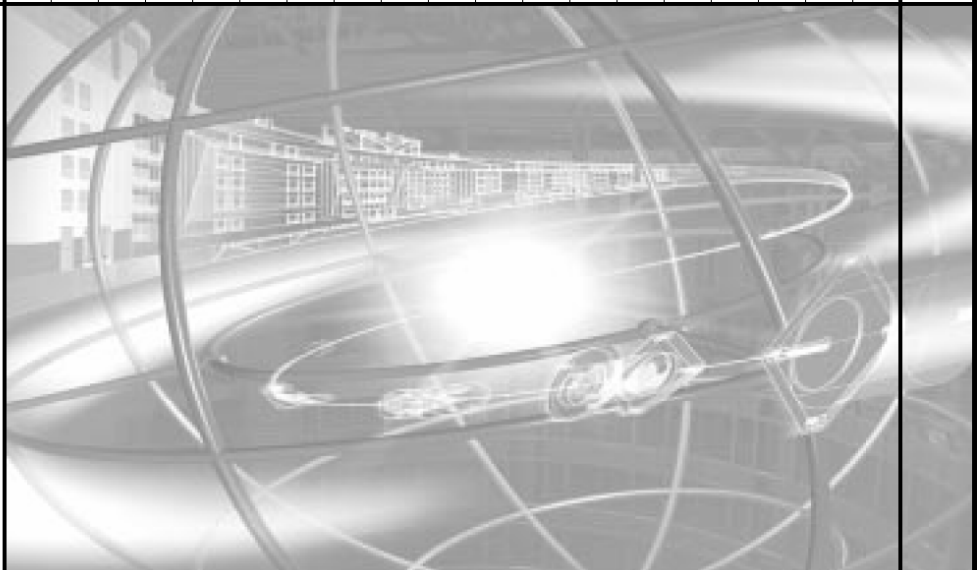


Biên dịch: Lê Quỳnh Mai (chủ biên)  
Trương Thanh Hoàng  
Hoàng Thuỳ Linh  
Hiệu đính: Bùi Công Độ

Phát triển  
**AutoCAD** bằng  
**ActiveX & VBA**



Autodesk®

**Biên dịch:** Lê Quỳnh Mai (chủ biên)  
Trương Thanh Hoàng  
Hoàng Thùy Linh  
**Hiệu đính:** Bùi Công Độ

# **Phát triển AutoCAD bằng ActiveX & VBA**

## LỜI MỞ ĐẦU

Sự quen thuộc với AutoCAD của người làm công tác thiết kế là hiển nhiên bởi khả năng hỗ trợ tạo bản vẽ kỹ thuật tuyệt vời cùng tính dễ dùng của nó. Tuy vậy, với đòi hỏi ngày càng cao của công việc, AutoCAD đang dần phát triển, từ một môi trường hỗ trợ tạo bản vẽ, đã biến thành một môi trường tích hợp, mà ở đó người dùng có thể lấy AutoCAD làm nền để xây dựng cho mình những công cụ làm việc có khả năng tùy biến cao, vượt ra khỏi giới hạn là công cụ tạo bản vẽ thông thường. Nắm bắt được nhu cầu này, cùng với mục tiêu đào tạo của bộ môn *Tự động hóa thiết kế cầu đường*, trường Đại học Giao thông vận tải, chúng tôi đã nghiên cứu các công cụ phát triển AutoCAD và thấy rằng VBA thực sự thích hợp. Thứ nhất, nó được tích hợp sẵn trong AutoCAD và có thể khai thác mọi khả năng sẵn có trong AutoCAD. Thứ hai, ngôn ngữ lập trình VB rất phổ biến bởi tính dễ sử dụng và nhiều tài liệu tham khảo, điều này rất hữu ích cho người lập trình bằng VBA. Hơn nữa, tài liệu bằng tiếng Việt về lĩnh vực này hiện nay rất hiếm và không đầy đủ. Chính vì vậy, sau khi xem xét và cân nhắc kỹ lưỡng các loại tài liệu tham khảo cho việc phát triển AutoCAD bằng VBA, chúng tôi đã quyết định dịch cuốn sách này sang tiếng Việt với mong muốn đóng góp cho người sử dụng AutoCAD ở Việt Nam một tài liệu tham khảo đầy đủ và hữu dụng.

Cuốn sách này, với nội dung chính là hướng dẫn phát triển AutoCAD bằng VBA do chính hãng Autodesk xuất bản, đã thể hiện được đầy đủ nhất tất cả các kiến thức, từ cơ bản đến nâng cao, trong lĩnh vực xây dựng các ứng dụng trên AutoCAD. Hy vọng rằng, với sự am hiểu về AutoCAD, về lập trình hướng đối tượng và sự cố gắng của bản thân, chúng tôi sẽ mang lại cho người đọc một tài liệu tham khảo thiết thực.

# MỤC LỤC

<b>ỨNG DỤNG MẪU.....</b>	<b>9</b>
1. Vẽ bãi đỗ xe.....	10
2. Chuyển từ tọa độ bản đồ sang tọa độ địa cầu.....	11
3. Liên kết cơ sở dữ liệu .....	13
4. Tính toán cần trục tháp.....	14
5. Xuất thuộc tính .....	16
6. Xây dựng đậm chữ I.....	17
<b>MỞ ĐẦU.....</b>	<b>19</b>
1. Tổng quan về công nghệ AutoCAD ActiveX.....	20
1.1. Tổng quan về các đối tượng AutoCAD ActiveX .....	20
2. Tổng quan về giao diện AutoCAD Visual Basic for Applications (VBA) .....	21
2.1. Cách thức thực thi của VBA trong AutoCAD .....	21
2.2. Phụ thuộc và hạn chế khi sử dụng AutoCAD VBA .....	22
3. Ưu điểm của sự kết hợp AutoCAD ActiveX và VBA.....	22
4. Tổ chức của cuốn sách .....	23
5. Tìm mã lệnh ví dụ.....	23
5.1. Thực thi các ứng dụng mẫu.....	23
5.2. Xem các ứng dụng mẫu .....	24
<b>CHƯƠNG 1: LÀM QUEN VỚI VBA .....</b>	<b>27</b>
1. Khái niệm về dự án VBA nhúng và độc lập.....	28
2. Tổ chức Dự án bằng VBA Manager.....	28
2.1. Tải một dự án đã có .....	29
2.2. Dỡ bỏ dự án .....	30
2.3. Nhúng dự án vào bản vẽ.....	30
2.4. Tách dự án VBA ra khỏi bản vẽ.....	30
2.5. Tạo dự án mới .....	31
2.6. Lưu dự án.....	31
3. Xử lý Macro .....	31
3.1. Thực thi Macro.....	32
3.2. Hiệu chỉnh Macro .....	32
3.3. Truy cập vào Macro.....	32

3.4. Tạo mới Macro .....	33
3.5. Xóa Macro.....	33
3.6. Thiết lập các tùy chọn trong dự án .....	33
<b>4. Hiệu chỉnh dự án bằng VBA IDE .....</b>	<b>34</b>
4.1. Mở VBA IDE .....	34
4.2. Xem thông tin về dự án.....	34
4.3. Định nghĩa các thành phần trong một dự án .....	35
4.4. Nhập những thành phần đã có.....	36
4.5. Hiệu chỉnh các thành phần .....	36
4.6. Thực thi Macro .....	38
4.7. Đặt tên dự án.....	38
4.8. Lưu Dự án.....	39
4.9. Tham chiếu dự án VBA khác .....	39
4.10. Thiết lập các tùy chọn trong VBA IDE .....	40
<b>5. Bài tập mở đầu .....</b>	<b>42</b>
<b>6. Thông tin thêm .....</b>	<b>42</b>
<b>7. Nhắc lại các thuật ngữ về dự án AutoCAD VBA .....</b>	<b>43</b>
<b>8. Nhắc lại về lệnh AutoCAD VBA .....</b>	<b>43</b>
<b><u>CHƯƠNG 2: CÁC KHÁI NIỆM CƠ BẢN VỀ ActiveX Automation</u></b>	<b><u>45</u></b>
<hr/>	
<b>1. Tìm hiểu mô hình đối tượng trong AutoCAD .....</b>	<b>46</b>
1.1. Sơ lược về đối tượng Application .....	48
1.2. Sơ lược về đối tượng Document.....	48
1.3. Sơ lược về tập đối tượng .....	50
1.4. Sơ lược về các đối tượng Đồ họa và Phi đồ họa.....	50
1.5. Sơ lược về đối tượng Preferences, Plot và Utility.....	50
<b>2. Truy xuất cây phân cấp đối tượng .....</b>	<b>51</b>
2.1. Tham chiếu đối tượng trong Cấu trúc cây phân cấp đối tượng .....	52
2.2. Truy xuất đối tượng Application .....	52
<b>3. Làm việc với Tập đối tượng .....</b>	<b>52</b>
3.1. Truy xuất Tập đối tượng.....	53
3.2. Thêm đối tượng mới vào Tập đối tượng.....	54
3.3. Duyệt Tập đối tượng.....	54
3.4. Xóa một đối tượng khỏi Tập đối tượng .....	54
<b>4. Tìm hiểu Phương thức và Thuộc tính .....</b>	<b>55</b>
<b>5. Tìm hiểu Đối tượng gốc .....</b>	<b>55</b>
<b>6. Thư viện kiểu .....</b>	<b>55</b>
<b>7. Gọi lại Thực Thể Đầu Tiên trong CSDL.....</b>	<b>56</b>
<b>8. Sử dụng Variant trong phương thức và thuộc tính .....</b>	<b>56</b>
8.1. Variant là gì? .....	56
8.2. Sử dụng biến Variant trong dữ liệu mảng .....	56
8.3. Chuyển Mảng thành Variant.....	57
8.4. Mảng Variant.....	57
<b>9. Sử dụng các ngôn ngữ lập trình khác.....</b>	<b>58</b>
9.1. Chuyển đổi từ mã VBA sang VB.....	58
9.2. Đoạn mã ví dụ so sánh VBA và VB.....	59

## **CHƯƠNG 3: ĐIỀU KHIỂN MÔI TRƯỜNG AutoCAD ..... 61**

<b>1. Mở, Lưu và Đóng các bản vẽ .....</b>	<b>62</b>
1.1. Mở bản vẽ .....	62
1.2. Tạo bản vẽ mới .....	62
1.3. Lưu bản vẽ .....	62
<b>2. Thiết lập các lựa chọn trong AutoCAD .....</b>	<b>63</b>
2.2. Lựa chọn về CSDL .....	64
<b>3. Điều khiển cửa sổ ứng dụng .....</b>	<b>64</b>
3.1. Thay đổi vị trí và kích thước của cửa sổ ứng dụng .....	65
3.2. Thu phóng cửa sổ ứng dụng AutoCAD .....	65
3.3. Xác định trạng thái hiện hành của cửa sổ AutoCAD .....	65
3.4. Ẩn cửa sổ ứng dụng .....	65
<b>4. Điều khiển cửa sổ bản vẽ .....</b>	<b>66</b>
4.1. Thay đổi vị trí và kích thước của cửa sổ bản vẽ .....	66
4.2. Thu phóng cửa sổ bản vẽ .....	66
4.3. Xác định trạng thái hiện hành của cửa sổ bản vẽ .....	66
4.4. Sử dụng chức năng thu phóng .....	67
4.5. Sử dụng các cảnh nhìn đã được đặt tên .....	71
4.6. Sử dụng các khung nhìn xếp cạnh nhau .....	72
4.7. Cập nhật đặc tính hình học trong cửa sổ bản vẽ .....	75
<b>5. Thiết lập lại các đối tượng hiện hành .....</b>	<b>76</b>
<b>6. Gán và lấy biến hệ thống .....</b>	<b>76</b>
<b>7. Vẽ với độ chính xác cao .....</b>	<b>77</b>
7.1. Điều chỉnh bắt điểm và lưới .....	77
7.2. Sử dụng chế độ bắt vuông góc .....	78
7.3. Vẽ đường tạm .....	79
7.4. Tính toán điểm và các giá trị liên quan .....	82
7.5. Tìm diện tích .....	82
<b>8. Nhắc người dùng nhập liệu .....</b>	<b>84</b>
8.1. Phương thức GetString .....	85
8.2. Phương thức GetPoint .....	85
8.3. Phương thức GetKeyword .....	86
8.4. Điều khiển quá trình nhập liệu của người dùng .....	86
<b>9. Truy xuất dòng lệnh của AutoCAD .....</b>	<b>87</b>
<b>10. Thao tác khi không mở bản vẽ nào .....</b>	<b>88</b>
<b>11. Nhập vào các định dạng khác .....</b>	<b>89</b>
<b>12. Xuất sang các định dạng khác .....</b>	<b>89</b>

## **CHƯƠNG 4: TẠO VÀ HIỆU CHỈNH THỰC THỂ AutoCAD .... 91**

<b>1. Tạo đối tượng .....</b>	<b>92</b>
1.1. Xác định đối tượng bao động .....	92
1.2. Tạo đường thẳng – đối tượng line .....	93
1.3. Tạo đối tượng cong .....	93
1.4. Tạo đối tượng điểm .....	94
1.5. Tạo vùng tô đặc .....	95
1.6. Tạo miền .....	96
1.7. Tạo vùng tô mẫu .....	99
<b>2. Hiệu chỉnh đối tượng .....</b>	<b>102</b>

2.1. Hiệu chỉnh các đối tượng phi đồ họa.....	102
2.2. Chọn đối tượng .....	103
2.3. Sao chép đối tượng.....	106
2.4. Di chuyển đối tượng.....	112
2.5. Xóa đối tượng .....	113
2.6. Co giãn đối tượng.....	114
2.7. Biến đổi đối tượng .....	115
2.8. Kéo dài hoặc cắt ngắn đối tượng.....	117
2.9. Phá vỡ đối tượng .....	118
2.10. Hiệu chỉnh đối tượng Polylines .....	119
2.11. Hiệu chỉnh đường cong Splines .....	121
2.12. Hiệu chỉnh vùng tô màu.....	123
<b>3. Sử dụng Lớp, Màu sắc và Kiểu đường .....</b>	<b>126</b>
3.1. Làm việc với các lớp.....	126
3.2. Làm việc với màu sắc.....	131
3.3. Làm việc với kiểu đường.....	132
3.4. Gán Lớp, Màu và Kiểu đường cho Đối tượng .....	134
<b>4. Thêm văn bản vào bản vẽ.....</b>	<b>137</b>
4.1. Làm việc với Kiểu chữ .....	137
4.2. Sử dụng Văn bản đơn .....	143
4.3. Sử dụng Văn bản nhiều dòng.....	146
4.4. Sử dụng ký tự Unicode, Ký tự điều khiển và Ký tự đặc biệt.....	151
4.5. Thay thế phông chữ.....	152
4.6. Kiểm tra chính tả .....	153

## **CHƯƠNG 5: KÍCH THƯỚC VÀ DUNG SAI..... 155**

<b>1. Khái niệm về kích thước.....</b>	<b>156</b>
1.1. Thành phần của một kích thước.....	157
1.2. Định nghĩa biến hệ thống kích thước .....	157
1.3. Thiết lập kiểu chữ cho kích thước.....	157
1.4. Khái niệm về đường dẫn.....	158
1.5. Khái niệm về kích thước liên kết.....	158
<b>2. Tạo kích thước.....</b>	<b>159</b>
2.1. Tạo kích thước dạng đường.....	159
2.2. Tạo kích thước dạng tia.....	159
2.3. Tạo kích thước đo góc.....	161
2.4. Tạo kích thước dạng tọa độ.....	162
<b>3. Hiệu chỉnh kích thước.....</b>	<b>163</b>
<b>4. Kiểu kích thước.....</b>	<b>164</b>
4.1. Kiểu kích thước ghi đề.....	165
<b>5. Kích thước trong không gian mô hình và không gian in .....</b>	<b>168</b>
<b>6. Tạo đường dẫn và chú thích .....</b>	<b>169</b>
6.1. Tạo đường dẫn.....	169
6.2. Thêm chú thích vào đường dẫn.....	170
6.3. Liên kết của đường dẫn.....	170
6.4. Hiệu chỉnh liên kết của đường dẫn .....	171
6.5. Hiệu chỉnh đường dẫn.....	171
<b>7. Tạo dung sai hình học.....</b>	<b>172</b>
7.1. Hiệu chỉnh dung sai .....	173

## **CHƯƠNG 6: TÙY BIẾN THANH CÔNG CỤ VÀ TRÌNH ĐƠN 175**

<b>1. Tìm hiểu tập đối tượng MenuBar và MenuGroups .....</b>	<b>176</b>
1.1. Khám phá tập đối tượng MenuGroups .....	177
<b>2. Tải các nhóm trình đơn .....</b>	<b>177</b>
2.1. Tạo nhóm trình đơn mới .....	178
<b>3. Thay đổi thanh trình đơn.....</b>	<b>179</b>
3.1. Chèn một mục vào thanh trình đơn.....	179
3.2. Gỡ bỏ một mục ra khỏi thanh trình đơn.....	180
3.3. Sắp xếp lại các mục đơn trên thanh trình đơn.....	180
<b>4. Tạo và hiệu chỉnh trình đơn kéo xuống và trình đơn tắt.....</b>	<b>181</b>
4.1. Tạo trình đơn mới .....	181
4.2. Thêm mục mới vào một trình đơn.....	182
4.3. Thêm vạch ngăn vào một trình đơn .....	184
4.4. Gán phím tắt cho một mục trình đơn.....	184
4.5. Tạo trình đơn con nhiều tầng .....	185
4.6. Xóa mục trình đơn khỏi một trình đơn.....	186
4.7. Tìm hiểu các thuộc tính của mục trình đơn .....	186
<b>5. Tạo và hiệu chỉnh thanh công cụ .....</b>	<b>189</b>
5.1. Tạo mới thanh công cụ .....	189
5.2. Thêm nút vào thanh công cụ.....	189
5.3. Thêm vạch ngăn vào một thanh công cụ .....	191
5.4. Định nghĩa ảnh cho nút.....	191
5.5. Tạo thanh công cụ Flyout.....	192
5.6. Thanh công cụ nổi và thanh công cụ neo .....	193
5.7. Xóa nút khỏi thanh công cụ.....	194
5.8. Tìm hiểu các thuộc tính của nút.....	194
<b>6. Tạo Macro.....</b>	<b>196</b>
6.1. Ký tự Macro và ký tự ASCII tương đương .....	196
6.2. Kết thúc Macro.....	197
6.3. Dùng để người dùng nhập liệu .....	198
6.4. Hủy lệnh.....	199
6.5. Lập lại Macro .....	199
6.6. Sử dụng chế độ chọn đối tượng đơn .....	200
<b>7. Tạo dòng trạng thái trợ giúp cho các mục trong trình đơn và nút trên thanh công cụ .....</b>	<b>200</b>
<b>8. Thêm mục vào trình đơn tắt.....</b>	<b>201</b>

## **CHƯƠNG 7: LÀM VIỆC VỚI CÁC SỰ KIỆN..... 203**

<b>1. Khái niệm về các sự kiện trong AutoCAD.....</b>	<b>204</b>
<b>2. Chỉ dẫn xây dựng bộ xử lý sự kiện.....</b>	<b>204</b>
<b>3. Xử lý sự kiện ở mức ứng dụng .....</b>	<b>205</b>
3.1. Kích hoạt sự kiện ở mức ứng dụng.....	207
<b>4. Xử lý sự kiện ở mức bản vẽ.....</b>	<b>208</b>
4.1. Kích hoạt sự kiện trong các môi trường ngoài VBA .....	209
4.2. Lập trình trong các môi trường khác VBA.....	210
4.3. Lập trình trong môi trường VBA.....	210
<b>5. Xử lý sự kiện ở mức đối tượng .....</b>	<b>211</b>
5.1. Kích hoạt sự kiện ở mức đối tượng .....	211



## **CHƯƠNG 8: LÀM VIỆC TRONG KHÔNG GIAN BA CHIỀU 215**

<b>1. Xác định tọa độ ba chiều .....</b>	<b>216</b>
1.1. Quy tắc bàn tay phải.....	216
1.2. Nhập tọa độ X, Y, Z .....	216
<b>2. Định nghĩa hệ tọa độ người dùng.....</b>	<b>218</b>
<b>3. Chuyển trục tọa độ .....</b>	<b>219</b>
<b>4. Tạo đối tượng ba chiều.....</b>	<b>222</b>
4.1. Tạo khung dây .....	223
4.2. Tạo lưới bề mặt.....	223
4.3. Tạo lưới đa diện .....	225
4.4. Tạo khối .....	226
<b>5. Hiệu chỉnh trong không gian 3D.....</b>	<b>226</b>
5.1. Quay .....	227
5.2. Nhân bản .....	228
5.3. Lấy đối xứng .....	229
<b>6. Hiệu chỉnh vật thể khối .....</b>	<b>230</b>

## **CHƯƠNG 9: TẠO BỐ CỤC VÀ IN ÁN..... 233**

<b>1. Khái niệm không gian mô hình và không gian in .....</b>	<b>234</b>
<b>2. Bố cục bản vẽ.....</b>	<b>234</b>
2.1. Mối quan hệ giữa Layout và Block .....	234
2.2. Khái niệm về cấu hình in .....	234
2.3. Xác định các cấu hình của Layout.....	235
<b>3. Khái niệm khung nhìn.....</b>	<b>236</b>
3.1. Chuyển sang Layout của không gian in .....	238
3.2. Chuyển sang Layout của không gian mô hình.....	239
3.3. Tạo khung nhìn trong không gian in .....	239
3.4. Thay đổi cảnh nhìn và nội dung khung nhìn .....	241
3.5. Đặt tỷ lệ cảnh nhìn theo không gian in .....	242
3.6. Đặt tỷ lệ cho mẫu của kiểu đường trong không gian in.....	243
3.7. Ẩn các đường thẳng trong khung nhìn khi in .....	244
<b>4. In bản vẽ .....</b>	<b>244</b>
4.1. Thao tác in cơ bản.....	244
4.2. In trong không gian mô hình .....	245
4.3. In trong không gian in.....	246

## **CHƯƠNG 10: KỸ THUẬT VẼ NÂNG CAO VÀ TỔ CHỨC BẢN VẼ .....**

<b>1. Làm việc với ảnh Raster .....</b>	<b>248</b>
1.1. Ảnh Raster trong bản vẽ.....	248
1.2. Đính kèm và đặt tỷ lệ ảnh Raster .....	249
1.3. Quản lý ảnh Raster .....	251
1.4. Hiệu chỉnh ảnh và đường biên .....	251
1.5. Cắt xén ảnh.....	253
<b>2. Sử dụng khối và thuộc tính .....</b>	<b>255</b>
2.1. Làm việc với khối.....	255
2.2. Làm việc với thuộc tính .....	261

<b>3. Sử dụng tham chiếu ngoài .....</b>	<b>267</b>
3.1. Cập nhật tham chiếu ngoài.....	267
3.2. Đính kèm tham chiếu ngoài.....	267
3.3. Tách các tham chiếu ngoài.....	269
3.4. Tải lại tham chiếu ngoài.....	270
3.5. Loại bỏ các tham chiếu ngoài .....	270
3.6. Ràng buộc tham chiếu ngoài .....	271
3.7. Cắt xén các Khối và Tham chiếu ngoài .....	272
<b>4. Nối kết và khôi phục lại dữ liệu mở rộng .....</b>	<b>273</b>

## **CHƯƠNG 11: PHÁT TRIỂN ỨNG DỤNG BẰNG VBA .....** 275

<b>1. Một số thuật ngữ trong VBA.....</b>	<b>276</b>
<b>2. Làm việc với Form trong VBA .....</b>	<b>276</b>
2.1. Thiết kế và chạy chương trình .....	277
2.2. Tạo Form mới trong Dự án.....	277
2.3. Thêm điều khiển vào Form.....	277
2.4. Hiện thị và ẩn Form.....	279
2.5. Tải và dỡ bỏ Form .....	279
2.6. Thiết kế chương trình với Modal Form .....	280
<b>3. Xử lý lỗi .....</b>	<b>280</b>
3.1. Bẫy lỗi thực thi.....	281
3.2. Xử lý lỗi đã bẫy được.....	282
3.3. Xử lý lỗi nhập dữ liệu người dùng trong AutoCAD.....	283
<b>4. Bảo mật mã nguồn chương trình VBA.....</b>	<b>283</b>
<b>5. Thực thi Macro từ trình đơn hoặc thanh công cụ.....</b>	<b>283</b>
<b>6. Tự động tải dự án VBA .....</b>	<b>283</b>
<b>7. Tự động thực thi Macro.....</b>	<b>284</b>
<b>8. Tự động mở VBA IDE mỗi khi tải một dự án .....</b>	<b>284</b>
<b>9. Làm việc khi không có bản vẽ được mở .....</b>	<b>284</b>
<b>10. Phân phối ứng dụng.....</b>	<b>285</b>
10.1. Phân phối ứng dụng Visual Basic.....	285

## **CHƯƠNG 12: TƯƠNG TÁC VỚI ỨNG DỤNG KHÁC, CƠ SỞ DỮ LIỆU VÀ WINDOWS API.....** 287

<b>1. Tương tác với ứng dụng Visual LISP.....</b>	<b>288</b>
<b>2. Tương tác với ứng dụng trên Windows.....</b>	<b>288</b>
2.1. Tham chiếu thư viện đối tượng ActiveX của ứng dụng khác .....	289
2.2. Tạo đại diện của ứng dụng.....	290
2.3. Lập trình với các đối tượng của ứng dụng khác .....	290
<b>3. Sử dụng DAO để truy cập thông tin của cơ sở dữ liệu .....</b>	<b>292</b>
3.1. Tham chiếu thư viện đối tượng DAO .....	292
3.2. Mở cơ sở dữ liệu .....	293
3.3. Lập trình với mô hình đối tượng của DAO .....	293
<b>4. Truy cập hàm Windows API từ VBA.....</b>	<b>293</b>

**CHƯƠNG 13: THIẾT KẾ ĐƯỜNG ĐI DẠO TRONG VƯỜN -  
MỘT VÍ DỤ VỀ ActiveX/VBA ..... 295**

<b>1. Kiểm tra môi trường làm việc.....</b>	<b>296</b>
<b>2. Xác định mục đích.....</b>	<b>296</b>
<b>3. Viết đoạn chương trình đầu tiên.....</b>	<b>297</b>
<b>4. Nhập số liệu .....</b>	<b>298</b>
4.1. Khai báo biến .....	298
4.2. Tạo chương trình con <code>gpuser</code> .....	298
<b>5. Vẽ đường đi dạo.....</b>	<b>300</b>
<b>6. Vẽ lớp gạch lát.....</b>	<b>302</b>
<b>7. Tổng hợp lại .....</b>	<b>304</b>
<b>8. Duyệt mã lệnh .....</b>	<b>304</b>
<b>9. Thực thi Macro.....</b>	<b>305</b>
<b>10. Thêm giao diện hộp thoại.....</b>	<b>306</b>
10.1. Tạo hộp thoại .....	306
10.2. Dùng cửa sổ Project để quản lý dự án .....	308
10.3. Cập nhật mã lệnh hiện có.....	309
10.4. Thêm mã lệnh cho hộp thoại.....	311

**PHỤ LỤC A: SO SÁNH Visual LISP VÀ ActiveX/VBA..... 315**

<b>1. So sánh Visual LISP và ActiveX/VBA.....</b>	<b>316</b>
---	------------

**PHỤ LỤC B: CHUYỂN ĐỔI TỪ AutoCAD PHIÊN BẢN 14.01. 327**

<b>1. Mục mới cập nhật .....</b>	<b>328</b>
<b>2. Mục đã thay đổi .....</b>	<b>341</b>
2.1. Đối tượng Preferences.....	342
<b>3. Mục đã loại bỏ.....</b>	<b>343</b>

## ỨNG DỤNG MẪU

Những người thiết kế chuyên nghiệp cũng như những ai ham thích AutoCAD đều dùng giao diện ActiveX® và VBA để tạo ra những ứng dụng linh động và mạnh mẽ. Để minh họa chỉ một vài trong số các khả năng mạnh mẽ của giao diện lập trình này, phần dưới đây sẽ đề cập công việc của những người sử dụng AutoCAD trên khắp thế giới. Rất nhiều trong số các ứng dụng này có trong thư mục *Sample* của AutoCAD.



### Trong phần này

- **Vẽ bãi đỗ xe**
- **Chuyển từ tọa độ bản đồ sang tọa độ địa cầu**
- **Liên kết cơ sở dữ liệu**
- **Tính toán cần trục tháp**
- **Xuất thuộc tính**
- **Xây dựng đậm chữ I**

Tài liệu tham khảo  
BM Tự động hóa

# 1. Vẽ bãi đỗ xe

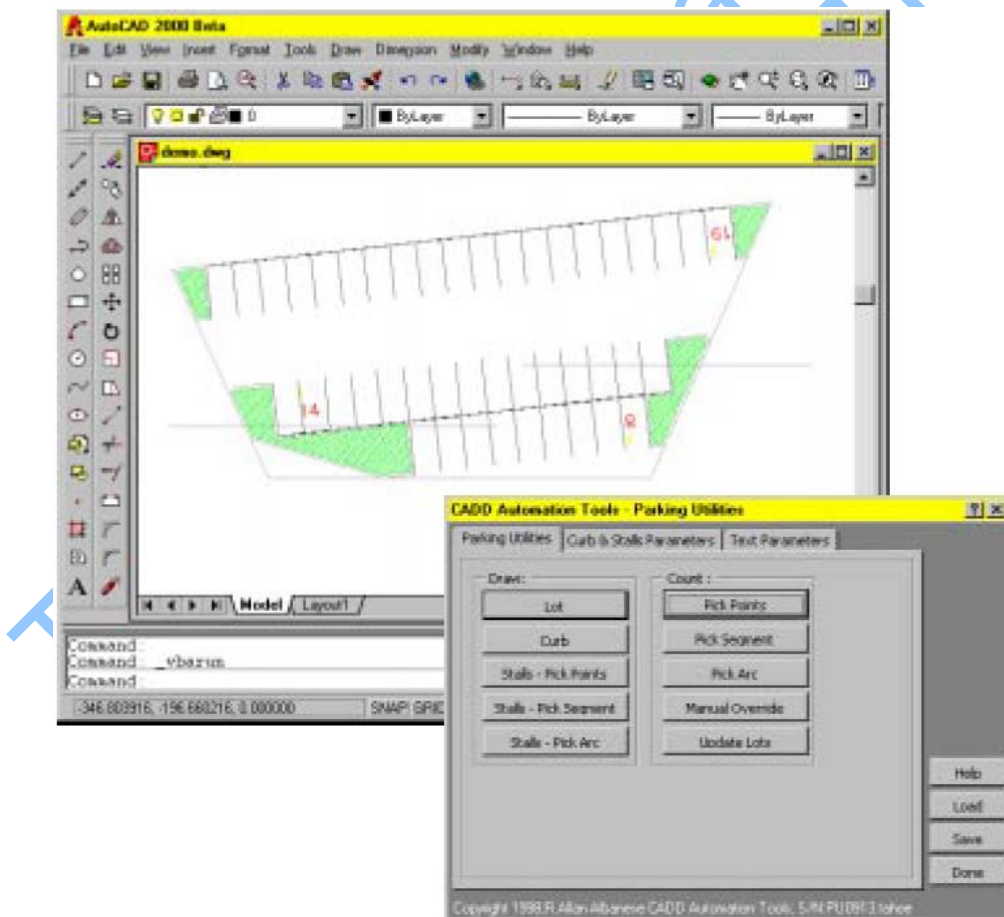
Tiện ích<sup>1</sup> vẽ bãi đỗ xe cho phép vẽ tự động khu vực đỗ xe trong các đường khép kín, các thanh chắn dừng xe riêng biệt, vẽ và thống kê các vị trí đỗ xe. Các dữ liệu mở rộng thông minh gắn với các đối tượng đồ họa giúp đảm bảo các bảng dữ liệu luôn đồng bộ với các đối tượng đồ họa.

Tiện ích vẽ bãi đỗ xe bao gồm rất nhiều thông số do người dùng quy định cho phép tạo khu vực đỗ xe một cách linh hoạt. Các thông số này có thể được lưu lại để chia sẻ với người khác.

Tiện ích vẽ bãi đỗ xe được phân phối như một tệp dự án<sup>2</sup> VBA độc lập và có đi kèm một tệp trợ giúp.

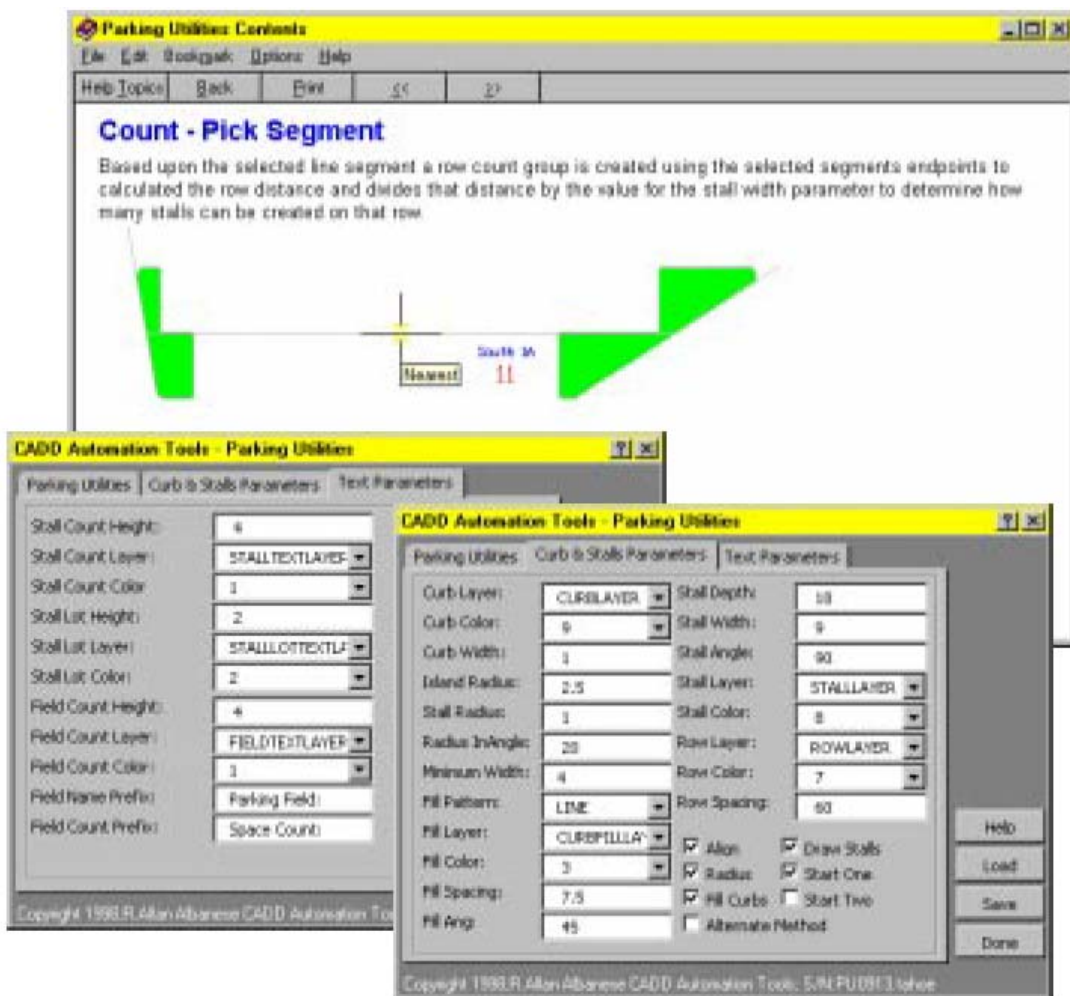
Tiện ích vẽ bãi đỗ xe do R. Allan Albanese của CADD Automation Tools phát triển. Đây là công ty chuyên về phát triển các ứng dụng dành cho AutoCAD sử dụng Microsoft VBA.

Ta có thể tìm thêm thông tin về ứng dụng này và các ứng dụng CADD Automation khác tại <http://www.caddautomationtools.com>.



<sup>1</sup> Tiện ích (utility application) là một ứng dụng được xây dựng nhằm thực hiện một nhiệm vụ cụ thể.

<sup>2</sup> Dự án (project) là một chương trình được tạo ra từ một hay nhiều tệp mã nguồn viết bằng VBA trong AutoCAD



## 2. Chuyển từ tọa độ bản đồ sang tọa độ địa cầu

Tiện ích này cho phép chuyển kinh độ, vĩ độ hai chiều thành tọa độ không gian ba chiều. Các đường vẽ ba chiều được tạo ra từ những tọa độ mới này và thể hiện trên một khối cầu.

Sử dụng công cụ *3D Orbit* trong AutoCAD, ta có thể xoay khối cầu để nhìn từ mọi vị trí.

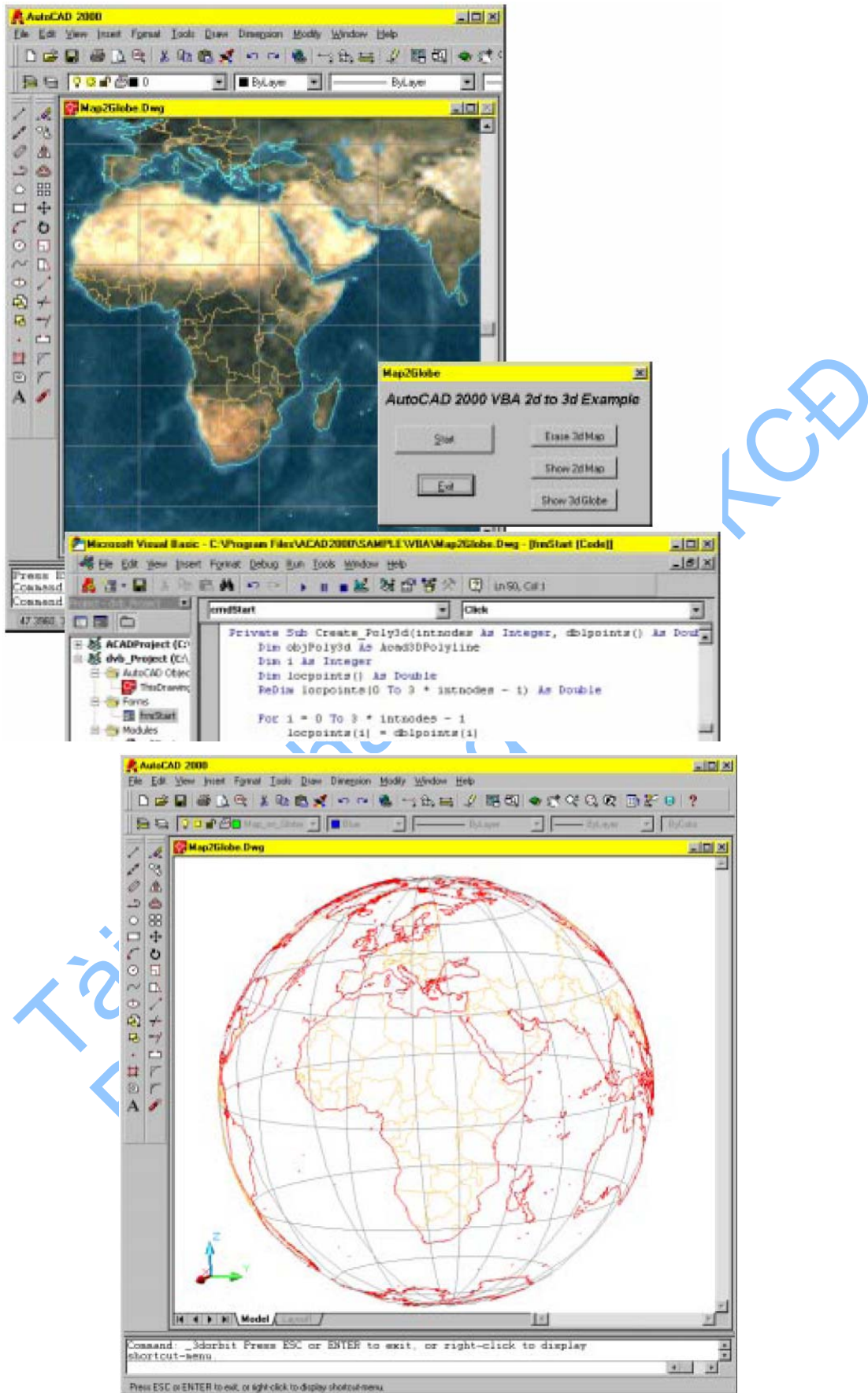
Tiện ích này được phân phối như một dự án VBA nhúng<sup>1</sup> trong bản vẽ *Map2Globe.dwg*. Hộp thoại của ứng dụng này được thiết kế cho phép người sử dụng chuyển bản đồ phẳng thành hình cầu ba chiều một cách nhanh chóng bằng cách sử dụng khung nhìn.

Tiện ích được phát triển bởi Carlos Ramos, kỹ sư phát triển ứng dụng của Autodesk Latin America.

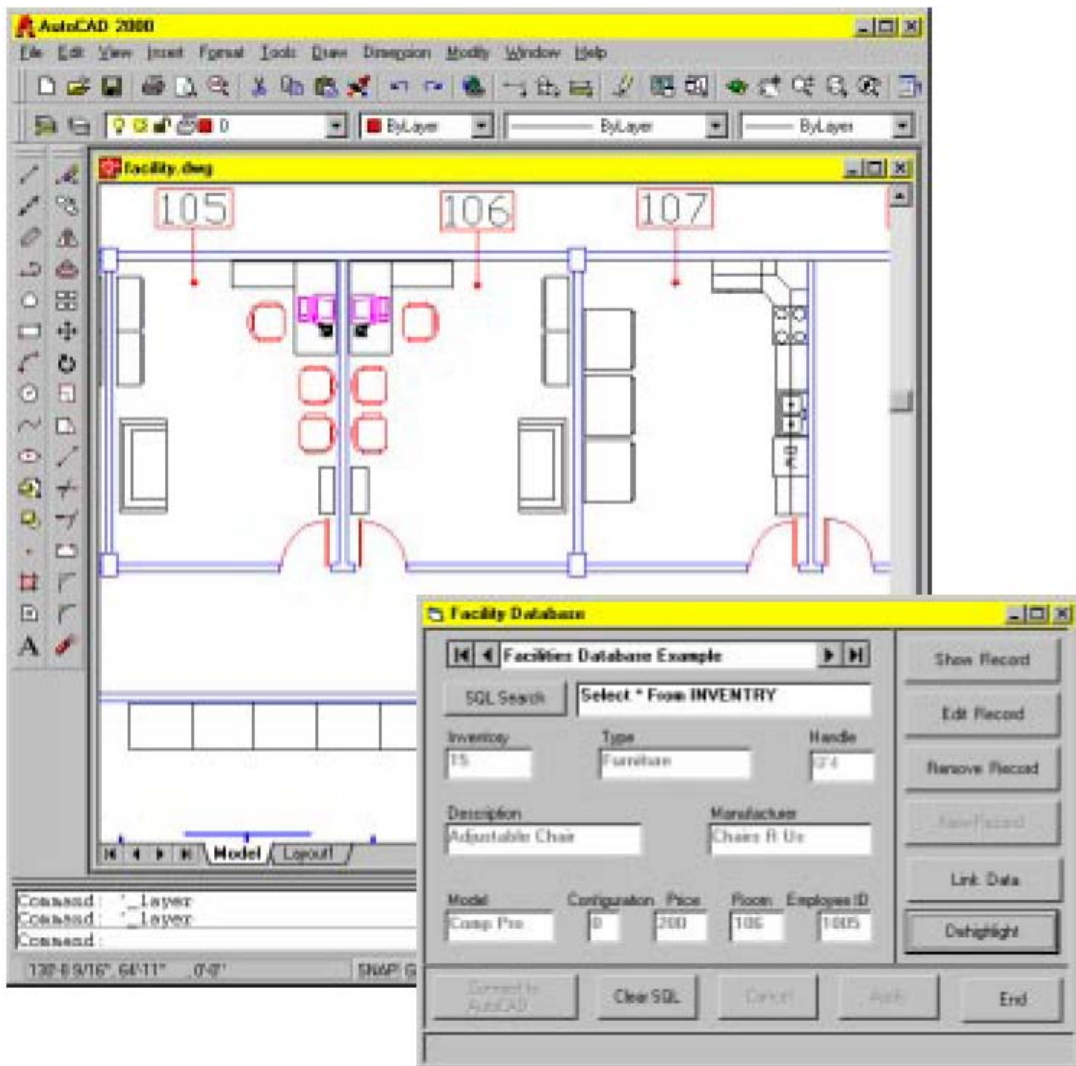
Bạn có thể tìm thấy bản vẽ *Map2Globe.dwg* trong thư mục *Sample\VBA* của AutoCAD.

<sup>1</sup> Chi tiết về một dự án VBA nhúng (embedded) trong bản vẽ được trình bày trong Chương 1





### 3. Liên kết cơ sở dữ liệu



Tiện ích này cho phép liên kết các khối trong một bản vẽ AutoCAD với cơ sở dữ liệu trong Microsoft Access. Với tiện ích này, ta có thể hiệu chỉnh, tạo mới, xóa bỏ các bản ghi và liên kết các đối tượng trong bản vẽ với các bản ghi trong cơ sở dữ liệu. Mã của tiện ích này được viết bằng Visual Basic® và được phân phối trong tệp tin khả thi *Facility.exe*.

Có thể tìm cơ sở dữ liệu, mã nguồn, tệp tin khả thi và bản vẽ của tiện ích này tại thư mục *Sample\ActiveX\Facility* của AutoCAD.



The screenshot shows two tables in Microsoft Access. The top table is 'INVENTORY' and the bottom table is 'COMPUTER'.

INV_ID	TYPE	DESCRIPT	NFR	MODEL	QTY	PRICE	ROOM	EMP_ID
130	Hardware	Personal inventory	Hardware Hardware	CPU2	1	2500.00	112	1018
131	Hardware	Personal inventory	Hardware Hardware	CPU2	1	2500.00	120	1026
132	Hardware	Personal inventory	Hardware Hardware	CPU2	1	2500.00	118	1024
133	Hardware	Personal inventory	Hardware Hardware	CPU3	3	5500.00	113	1019
134	Hardware	Personal inventory	Hardware Hardware	CPU3	3	5500.00	114	1020
135	Hardware	Personal inventory	Hardware Hardware	CPU3	3	5500.00	115	1021
136	Hardware	Personal inventory	Hardware Hardware	CPU3	3	5500.00	116	1022
137	Hardware	Personal inventory	Hardware Hardware	CPU3	3	5500.00	117	1023
138	Hardware	Personal inventory	Hardware Hardware	CPU3	3	5500.00	119	1025
139	Hardware	Personal inventory	Hardware Hardware	CPU3	3	5500.00	122	1010

ID	COMP_CFG	CPU	HDRIVE	RAM	GRAPHICS	INPT_DEV
1	1 00000000	486/33	300MB	8MB	Super VGA	Digitizer
2	2 00000000	286/12	60MB	640K	VGA	Mouse
3	3 00000000	MACH/C	40MB	2MB	Standard	Mouse
4	4 00000000	386SX/16	80MB	4MB	VGA	Mouse
5	5 00000000	386/33	300MB	8MB	VGA	Mouse
6	6 00000000	SPARC2	600MB	16MB	Standard	Mouse

```

cmdShowRecord_Click
Call sset.SelectAtPoint(Point)
If sset.Count = 1 Then
    If StrComp(sset(0).EntityType, "AcDbBlockReference", 1) = 0 Then
        strHandle1 = sset(0).Handle
        strSQLText = "Select * From INVENTORY Where Handle = '" & Trim(strHandle1) & "'"
        Data1.RecordSource = strSQLText
        Data1.Refresh
        If CheckAllFailed Then
            cmdEditRecord.Enabled = True
            cmdDeleteRecord.Enabled = True
        Else
            MsgBox "The record doesn't exist. Use New button to add the record."
            cmdAddRecord.Enabled = True
            ClearSQL
        End If
    End If

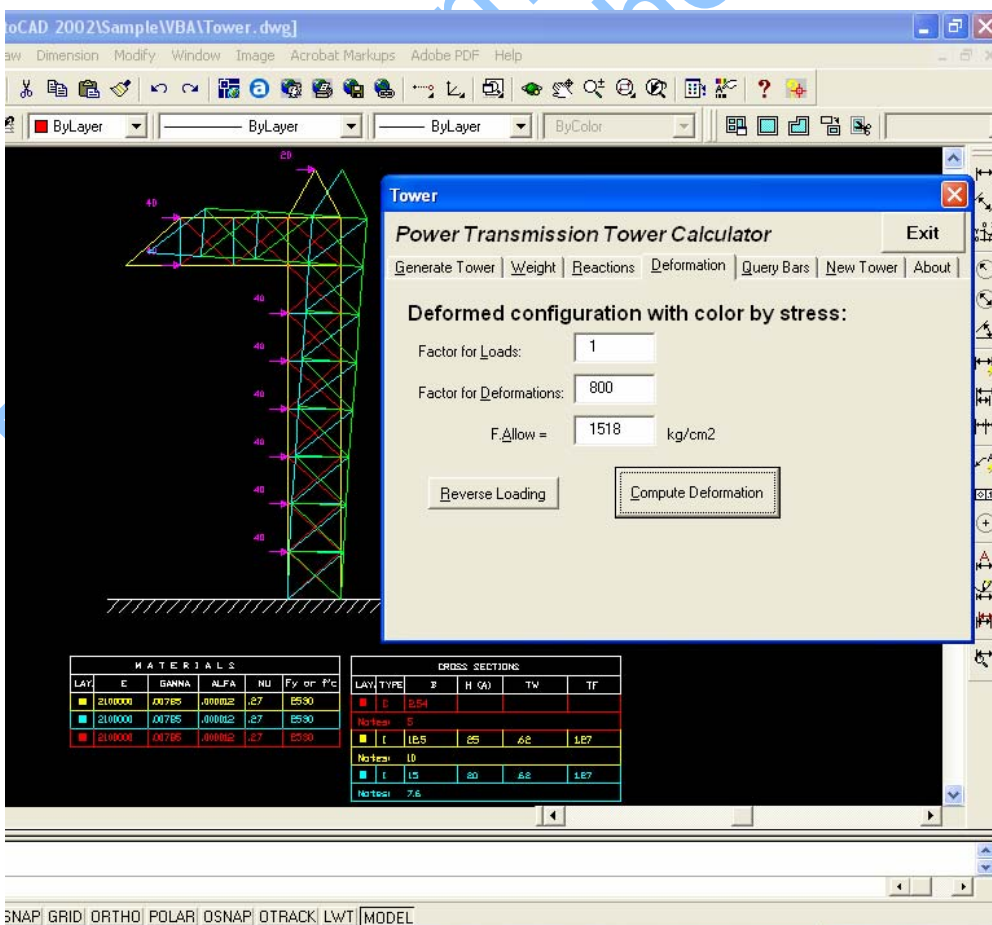
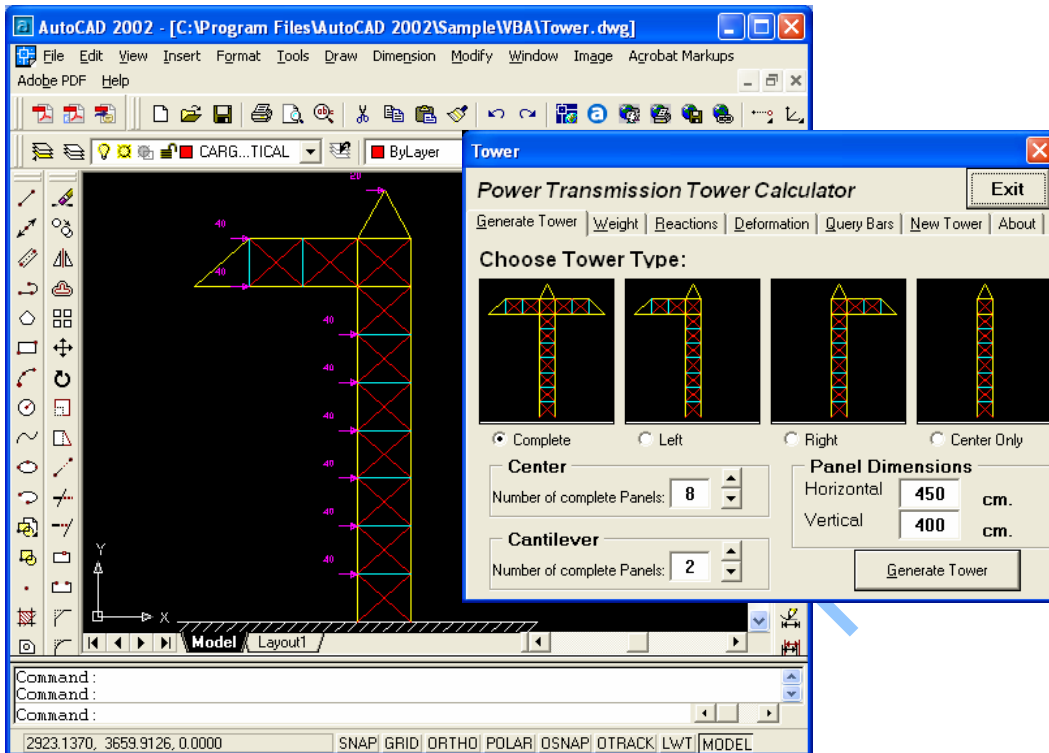
```

#### 4. Tính toán cần trục tháp

Tiện ích tính toán cần trục tháp cho phép tạo ra một cần trục tháp và thực hiện nhiều phép tính khác nhau cho kết cấu tháp. Tiện ích sử dụng hộp thoại với hệ thống bảng Tab nhằm hướng người sử dụng đến các bước khác nhau trong quá trình tạo lập và phân tích. Tất cả các phép tính được viết bằng mã Visual Basic®.

Tiện ích này được phân phối như một dự án VBA nhúng trong bản vẽ Tower.dwg, lưu trong thư mục Sample\VBA của AutoCAD.

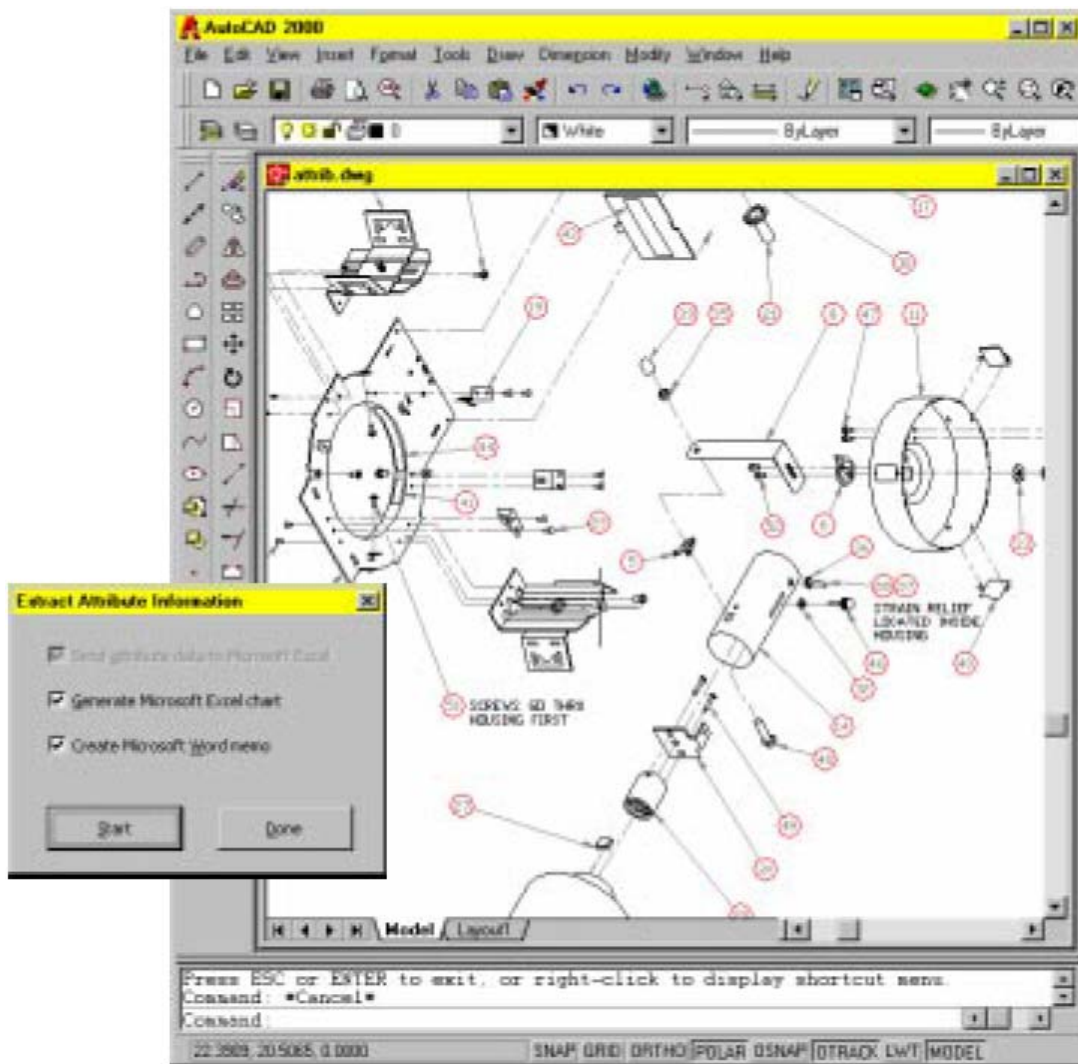
Tiện ích này được phát triển bởi Carlos Ramos, kỹ sư phát triển ứng dụng của Autodesk Latin America.

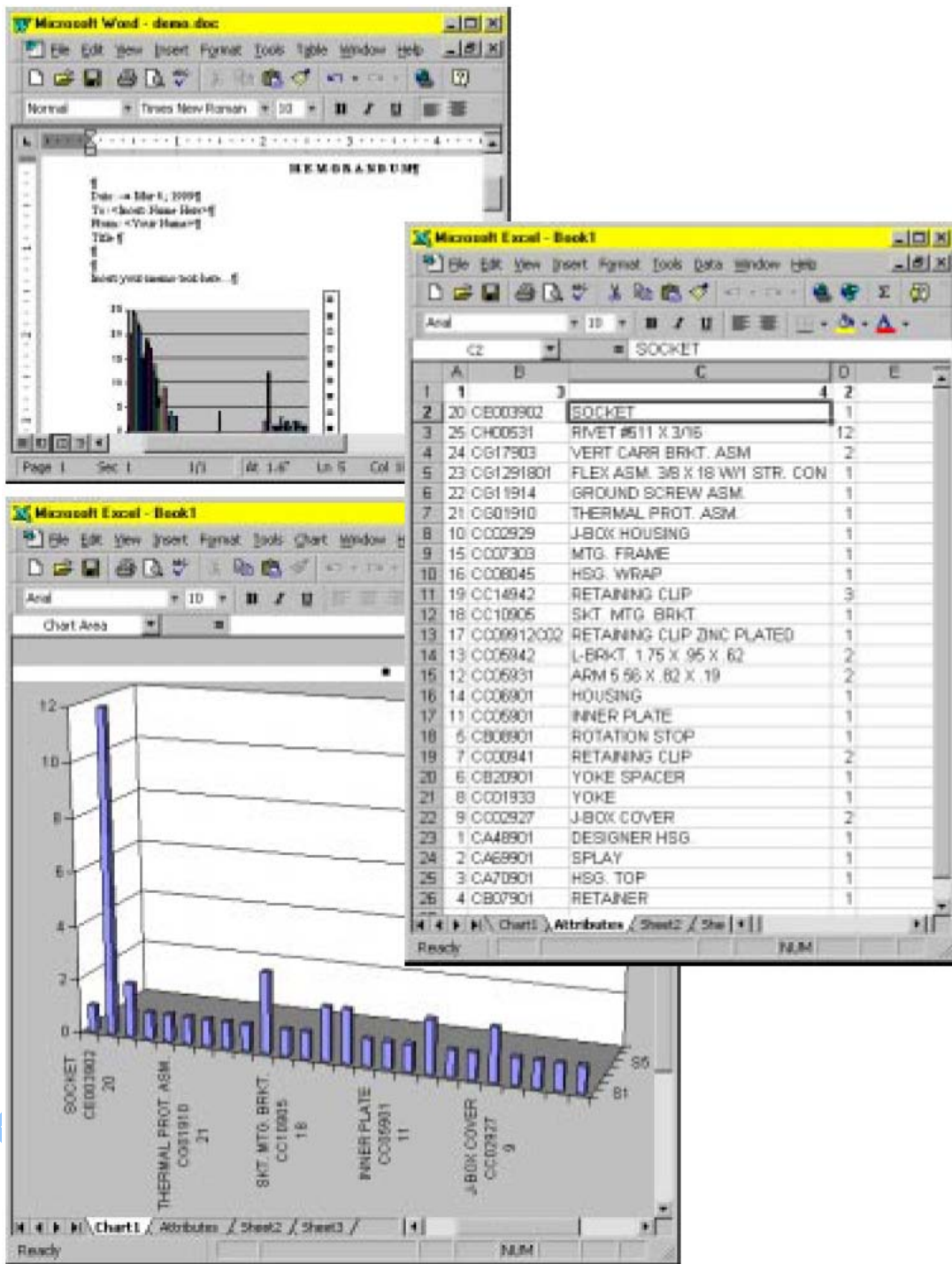


## 5. Xuất thuộc tính

Tiện ích này xuất tất cả các thuộc tính có trong bản vẽ. Các dữ liệu của thuộc tính được phân loại và lưu trong một bảng tính Excel. Các dữ liệu cũng được gắn vào một biểu đồ Microsoft Excel. Ngoài ra, tiện ích này có thể tạo ra một tài liệu Microsoft Word chứa biểu đồ các dữ liệu của thuộc tính.

Tiện ích này được phân phối trong một dự án VBA gọi là *atttext.dvb* và có thể thực hiện với bất kỳ bản vẽ nào có chứa thuộc tính. Tập dự án *atttext.dvb* có trong thư mục *SampleVBA* của AutoCAD. Tiện ích này do nhóm Application Developer Framework ở Autodesk phát triển.





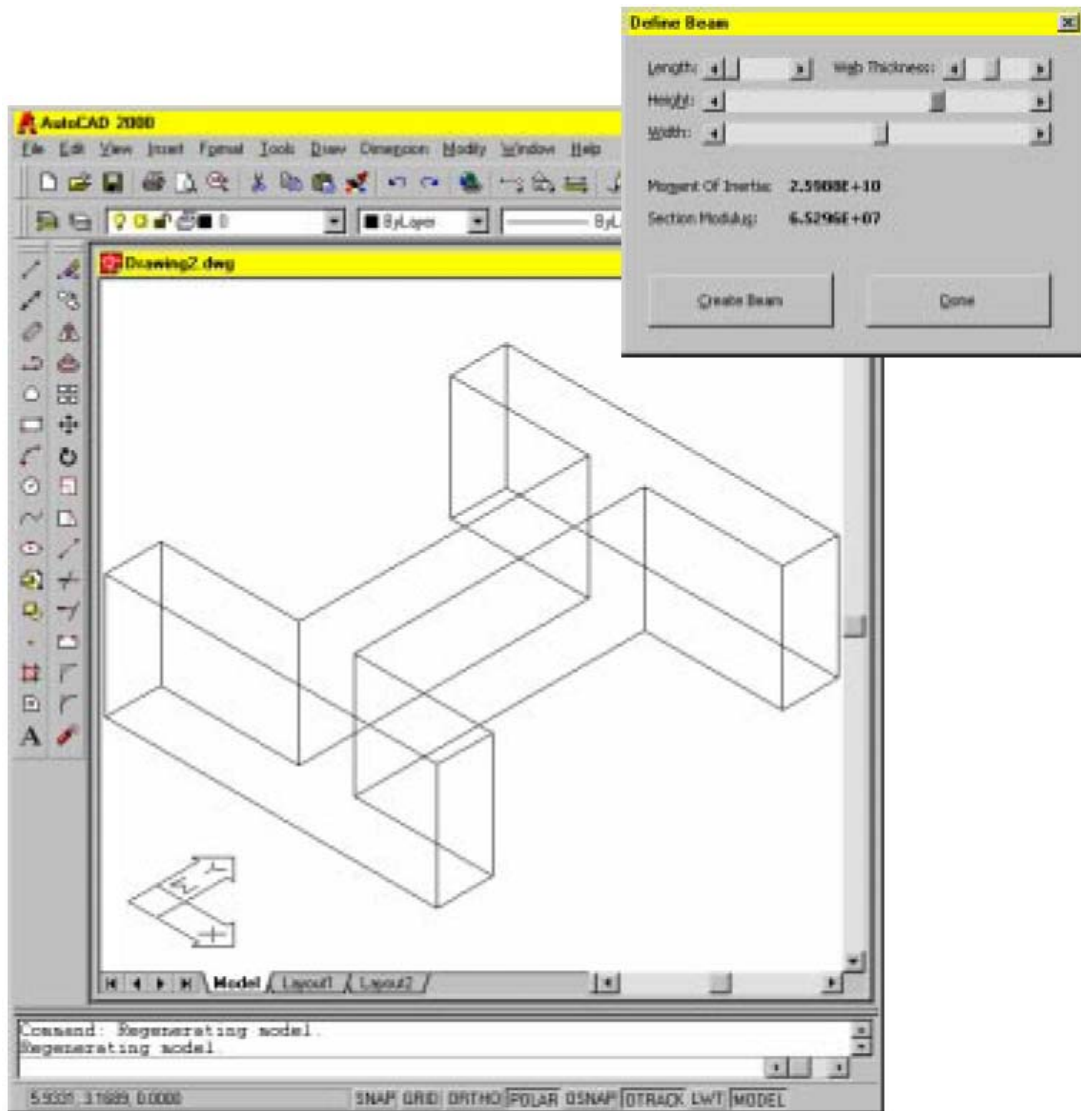
## 6. Xây dựng dầm chữ I

Tiện ích này cho phép tạo ra một dầm chữ I từ các thông số mà người dùng định nghĩa trong hộp thoại. Khi dầm chữ I được tạo ra, người sử dụng được tự do điều chỉnh các thông số thiết kế. Đối tượng dầm chữ I được tự động cập nhật khi người sử dụng thay đổi thông số thiết kế.



Tiện ích này được phân phối như một dự án VBA độc lập<sup>1</sup> gọi là *ibeam3d.dvb*. Tập dự án nằm trong thư mục *Sample\VBA* của AutoCAD.

Tiện ích này do Shashi Kant Rai thuộc nhóm Application Developer Framework của Autodesk phát triển.



---

<sup>1</sup> Chi tiết về dự án VBA độc lập được trình bày trong Chương 1

# MỞ ĐẦU

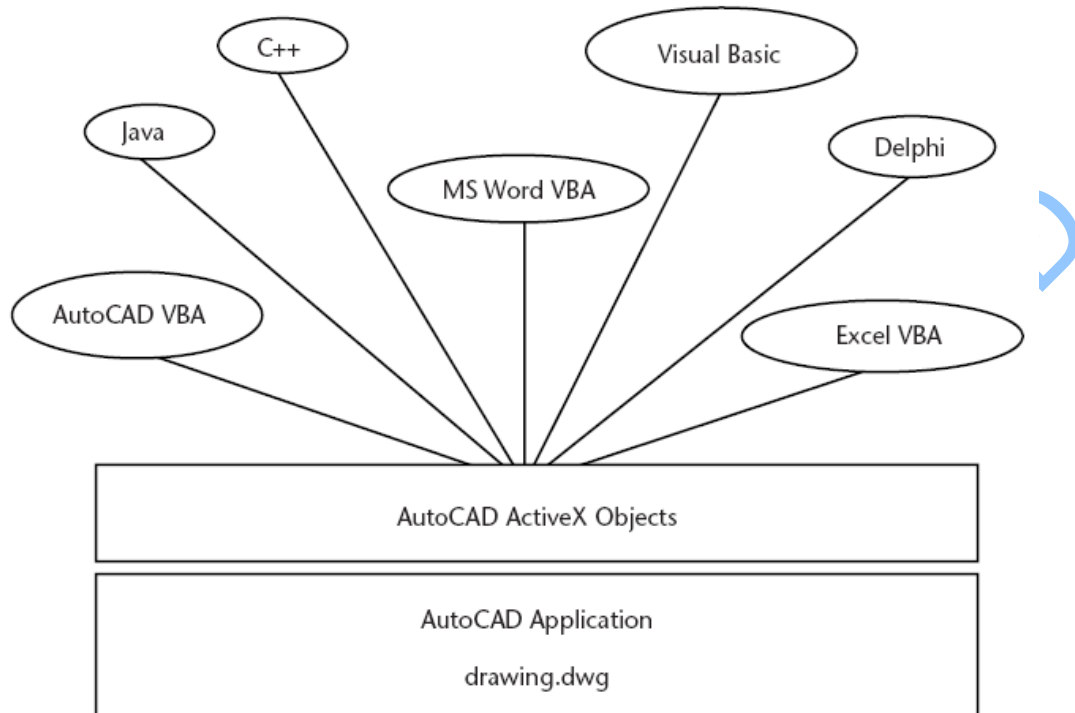
Chương này trình bày các khái niệm nhằm làm rõ các đối tượng AutoCAD thông qua giao diện ActiveX và lập trình trên các đối tượng đó thông qua môi trường phát triển ứng dụng Visual Basic for Application (VBA). Ngoài ra, chương này còn có phần giới thiệu về các loại tài liệu và mã ví dụ dùng trong AutoCAD ActiveX và VBA.

## Trong chương này

- **Tổng quan về công nghệ AutoCAD ActiveX**
- **Tổng quan về giao diện AutoCAD VBA**
- **Ưu điểm của sự kết hợp AutoCAD ActiveX và VBA**
- **Tổ chức của cuốn sách**
- **Tìm mã lệnh ví dụ**

# 1. Tổng quan về công nghệ AutoCAD ActiveX

AutoCAD ActiveX đưa ra cơ cấu để lập trình điều khiển AutoCAD từ cả trong và bên ngoài AutoCAD. Quá trình này được thực hiện bằng cách “trung bày” tất cả các đối tượng AutoCAD với “thế giới bên ngoài”. Khi đó, các đối tượng trong AutoCAD có thể được truy cập thông qua nhiều ngôn ngữ lập trình và các chương trình khác như Microsoft® Word VBA hoặc Excel VBA.



Có hai ưu điểm nổi bật khi sử dụng giao tiếp ActiveX cho AutoCAD:

- Khả năng lập trình truy cập vào bản vẽ AutoCAD được mở rộng cho nhiều môi trường lập trình khác nhau. Trước khi có ActiveX Automation, người lập trình bị giới hạn chỉ trong môi trường AutoLISP hoặc C++.
- Khả năng chia sẻ dữ liệu với các ứng dụng Windows® khác, chẳng hạn như Microsoft Excel® và Word®, được thực hiện dễ dàng hơn rất nhiều.

## 1.1. Tổng quan về các đối tượng AutoCAD ActiveX

Đối tượng chính là nền tảng xây dựng nên ứng dụng ActiveX. Mỗi đối tượng trong AutoCAD ActiveX là hiện thân một phần của AutoCAD. Có rất nhiều loại đối tượng khác nhau trong giao tiếp AutoCAD ActiveX. Chẳng hạn như:

- Các đối tượng đồ họa: line, arc, text, dimension...
- Thiết lập về định dạng: linetype, dimension style...
- Cấu trúc tổ chức: layer, group, block...
- Đối tượng liên quan đến hiển thị bản vẽ: view, viewport,...
- Và ngay cả bản vẽ và bản thân chương trình AutoCAD cũng được xem là đối tượng.

## 2. Tổng quan về giao diện AutoCAD Visual Basic for Applications (VBA)

Microsoft VBA là một môi trường lập trình hướng đối tượng có khả năng phát triển ứng dụng mạnh mẽ với những tính năng phong phú tương tự như của Visual Basic (VB). Điểm khác biệt chính giữa VBA và VB là VBA thực thi cùng trong tiến trình của ứng dụng AutoCAD và đưa ra một môi trường phát triển ứng dụng thông minh và rất nhanh chóng ngay bên trong AutoCAD.

VBA cũng có khả năng tích hợp với các ứng dụng có khả năng lập trình VBA khác. Điều đó có nghĩa là khi sử dụng thư viện đối tượng của các ứng dụng khác, AutoCAD có thể là Automation Controller<sup>1</sup> cho các ứng dụng khác như Microsoft Word và Excel.

Có bốn ưu điểm chính khi sử dụng VBA trong AutoCAD:

- Ngôn ngữ lập trình Visual Basic rất dễ học và dễ sử dụng.
- VBA thực thi cùng tiến trình với AutoCAD, vì vậy chương trình có tốc độ thực thi rất nhanh.
- Xây dựng giao diện hộp thoại nhanh chóng và hiệu quả. Điều này cho phép người lập trình tạo mẫu thử chương trình và nhận được phản hồi nhanh chóng ngay trong quá trình thiết kế.
- Dự án có thể được phân phối riêng hoặc nhúng trong các bản vẽ. Khả năng này cho phép người lập trình phân phối ứng dụng một cách linh hoạt.

### 2.1. Cách thức thực thi của VBA trong AutoCAD

VBA gửi thông điệp cho AutoCAD thông qua giao tiếp AutoCAD ActiveX Automation. AutoCAD VBA cho phép môi trường VBA thực thi đồng thời với AutoCAD và cung cấp khả năng lập trình điều khiển AutoCAD thông qua giao tiếp ActiveX Automation. Bộ đôi này của AutoCAD, ActiveX Automation và VBA, tạo ra giao diện lập trình mạnh mẽ không chỉ trong quá trình xử lý các đối tượng AutoCAD mà còn trong quá trình gửi dữ liệu và nhận dữ liệu từ các ứng dụng khác.

Có ba yếu tố cơ bản cấu thành giao diện lập trình ActiveX và VBA trong AutoCAD. Yếu tố đầu tiên chính là bản thân AutoCAD với tập đối tượng vô cùng phong phú, đóng gói tất cả các thực thể, dữ liệu và dòng lệnh AutoCAD. Do AutoCAD là ứng dụng được thiết kế với cấu trúc mở, với nhiều tầng giao diện khác nhau nên một khi đã quen thuộc với những khả năng của AutoCAD, ta sẽ lập trình VBA hiệu quả hơn nhiều. Người đã từng lập trình với AutoLISP<sup>®</sup> thường hiểu rất rõ cấu trúc của AutoCAD. Tuy nhiên, lập trình hướng đối tượng của VBA vẫn có nhiều điểm khác so với AutoLISP<sup>®</sup>.

---

<sup>1</sup> **Automation Controller:** là ngôn ngữ lập trình, chẳng hạn như VBA, có khả năng hỗ trợ công nghệ Automation của Microsoft. Một ứng dụng khi được lập trình sử dụng Automation Controller có thể tham khảo đến bất kỳ một thư viện đối tượng nào và có thể cập nhật đến từng đối tượng trong các thư viện đó chỉ từ một chương trình duy nhất.



Yếu tố thứ hai của giao tiếp AutoCAD ActiveX Automation là quá trình hình thành các thông điệp (hay các giao tiếp) với các đối tượng AutoCAD. Người lập trình VBA cần phải có những kiến thức cơ bản về ActiveX Automation. Ta có thể tìm hiểu thêm về giao tiếp AutoCAD ActiveX Automation trong cuốn “*ActiveX and VBA Reference*”. Ngay cả những người lập trình VB kinh nghiệm cũng nhận thấy rằng những kiến thức về giao tiếp AutoCAD ActiveX Automation là vô giá để có thể hiểu rõ và phát triển ứng dụng AutoCAD VBA.

Yếu tố thứ ba chính là môi trường lập trình VBA với hệ thống các đối tượng, từ khóa, hằng số,... cung cấp khả năng lập trình, điều khiển, gỡ lỗi và thực thi ứng dụng. Microsoft cũng cung cấp công cụ trợ giúp cho VBA ngay bên trong AutoCAD VBA và có thể truy cập trực tiếp trong VBA IDE bằng một trong các cách sau:

- Nhấn phím F1 trên bàn phím
- Chọn mục *Help* từ trình đơn của VBA IDE
- Bấm chuột vào biểu tượng dấu hỏi trên thanh công cụ của VBA IDE

## 2.2. Phụ thuộc và hạn chế khi sử dụng AutoCAD VBA

Để đảm bảo sự làm việc bình thường của ứng dụng AutoCAD Active và VBA cần phải đảm bảo hệ thống có các điều kiện sau:

Windows NT® 4.0

Yêu cầu phải có Windows NT4.0 Service Pack 3 để có thể chạy được AutoCAD ActiveX và VBA

Windows® 95 hoặc Windows 98

Không có yêu cầu đặc biệt nào

Cài đặt, Cài đặt lại hoặc Dỡ bỏ Microsoft Office hoặc các ứng dụng VBA khác

Nếu ta cài đặt, cài đặt lại hoặc dỡ bỏ Microsoft Office hoặc các ứng dụng VBA khác sau khi cài đặt AutoCAD, thì cần phải cài đặt lại AutoCAD. Đương nhiên, sau khi cài đặt AutoCAD, cần phải khởi động lại hệ thống.

## 3. Ưu điểm của sự kết hợp AutoCAD ActiveX và VBA

Giao tiếp AutoCAD ActiveX/VBA thể hiện nhiều điểm nổi bật so với các môi trường lập trình AutoCAD API khác:

- Tốc độ  
Do thực thi cùng tiến trình với VBA nên ứng dụng ActiveX nhanh hơn so với ứng dụng AutoLISP và ADS.
- Dễ sử dụng  
Ngôn ngữ lập trình và môi trường phát triển ứng dụng rất dễ sử dụng và được cài đặt sẵn trong AutoCAD.

- Khả năng hoạt động liên thông với Windows  
ActiveX và VBA được thiết kế để sử dụng với các ứng dụng Windows khác và được cung cấp khả năng giao tiếp thông tin với các ứng dụng khác.
- Tạo mẫu nhanh  
Khả năng xây dựng giao diện nhanh của VBA là môi trường hoàn hảo để xây dựng ứng dụng mẫu, ngay cả khi các ứng dụng đó được phát triển bằng ngôn ngữ khác.
- Cơ sở lập trình  
Trên khắp thế giới hiện nay có hàng triệu lập trình viên Visual Basic. Công nghệ AutoCAD ActiveX và VBA mở ra sự khả năng tùy biến và phát triển ứng dụng AutoCAD cho những lập trình viên này và những ai sẽ học Visual Basic trong tương lai.

## 4. Tổ chức của cuốn sách

Cuốn sách này cung cấp thông tin về cách phát triển ứng dụng ActiveX và VBA cho AutoCAD 2000. Thông tin chi tiết về các ứng dụng đang phát triển sử dụng VBA được đề cập trong chương 1 - “LÀM QUEN VỚI VBA” và chương 11 - “PHÁT TRIỂN ỨNG DỤNG BẰNG VBA”. Các lập trình viên sử dụng ActiveX từ một môi trường phát triển khác với VBA có thể bỏ qua hai chương này. Tuy nhiên, hãy lưu ý rằng các mã ví dụ trong sách được trình bày trong môi trường VBA.

Bài tập thực hành được trình bày trong chương 13 - “THIẾT KẾ ĐƯỜNG ĐI DẠO TRONG VƯỜN - MỘT VÍ DỤ VỀ ActiveX/VBA”. Bài tập này hướng đến những người mới học thông qua việc tạo chương trình vẽ đường đi dạo trong vườn trong AutoCAD sử dụng ActiveX và VBA.

Phụ lục B - “CHUYỂN ĐỔI TỪ AutoCAD PHIÊN BẢN 14.01” tổng kết những thay đổi của AutoCAD ActiveX và VBA kể từ AutoCAD phiên bản 14.01.

## 5. Tìm mã lệnh ví dụ

Có trên 800 chương trình VBA trong cuốn sách này và tài liệu “ActiveX and VBA Reference” minh họa cách sử dụng các phương thức, thuộc tính và sự kiện trong ActiveX.

Rất nhiều những ứng dụng mẫu lưu trong thư mục *Sample* của AutoCAD. Các ứng dụng này minh họa rất nhiều tính năng khác nhau, từ việc xuất dữ liệu của bản vẽ AutoCAD sang bảng tính Excel tới việc vẽ và thực hiện các phép phân tích phức tạp trên cần trục tháp. Những ví dụ này giúp ta hình dung được cách thức để kết hợp tính đa năng của môi trường lập trình Visual Basic for Applications với sức mạnh của giao tiếp AutoCAD ActiveX để tạo ra những ứng dụng có tính tùy biến cao.

### 5.1. Thực thi các ứng dụng mẫu

Tất cả các mã lệnh ví dụ trong cuốn sách này và trong “ActiveX and VBA Reference” có thể được sao chép trực tiếp từ tệp trợ giúp sang môi trường AutoCAD VBA, sau đó được thực thi với một yêu cầu: bản vẽ hiện hành trong

AutoCAD phải là bản vẽ trống và đang ở trong không gian mô hình. Ngoài ra, mã trong những cuốn sách có trong tệp *SampleCode.dvb* và *Events.dvb* trong thư mục *Sample*.

### Thực thi ví dụ

- 1 Sao chép ví dụ từ tệp trợ giúp sang mô-đun mã lệnh VBA còn trống.
- 2 Chắc chắn rằng AutoCAD có một bản vẽ trống được mở ở chế độ không gian mô hình.
- 3 Mở hộp thoại Macros bằng cách gõ lệnh VBARUN.
- 4 Lựa chọn Macro và nhấn Run.

Thông tin thêm về việc thực thi Macro<sup>1</sup> và hộp thoại Macro được thể hiện trong phần “Thực thi Macro” trang 32.

## 5.2. Xem các ứng dụng mẫu

Có 21 ứng dụng mẫu trong thư mục *Sample*. Bảng sau trình bày tên, mô tả và vị trí của tệp mã nguồn chính cho mỗi ứng dụng. Rất nhiều ứng dụng có thêm các tệp hỗ trợ, chứa trong cùng thư mục với tệp mã nguồn chính. Ngoài ra còn có tệp *readme.txt* mô tả ứng dụng và cách thực thi.

### Ứng dụng mẫu ActiveX và VBA

Tên	Mô tả	Đường dẫn
Dầm chữ I 3D	Tạo ra một hình chữ I 3D đặc và thay đổi kích thước một cách linh hoạt.	<i>/Sample/VBA/ibeam3d.dvb</i>
Từ bản đồ sang khối cầu	Tạo ra hình nhiều đường 3D trên một khối cầu từ các đường 2D ban đầu.	<i>/Sample/VBA/Map2Globe.dwg</i>
Tùy biến trình đơn	Dùng các đối tượng của MenuGroup và MenuBar.	<i>/Sample/VBA/Menu.dvb</i>
Theo dõi đối tượng	Sử dụng khối dữ liệu mở rộng để theo dõi sự thay đổi của đối tượng.	<i>/Sample/VBA/ObjectTracker.dvb</i>
Lưu thành phiên bản 12	Lưu một tệp đồ họa của AutoCAD2000 vào AutoCAD phiên bản 12.	<i>/Sample/VBA/SaveAsR12.dvb</i>
Tháp	Vẽ một căn trục tháp và thực hiện	<i>/Sample/VBA/Tower.dwg</i>

<sup>1</sup> **Macro:** là một loại chương trình VBA được nạp vào trong AutoCAD, chi tiết về *Macro* xem trong chương 1 mục “Xử lý Macro”. Trong bản dịch này, do chưa có từ chuyên môn tương đương trong tiếng Việt, cho nên chúng tôi dùng từ gốc tiếng Anh.

## Ứng dụng mẫu ActiveX và VBA

Tên	Mô tả	Đường dẫn
	các phép phân tích.	
Chiều cao chữ	Thay đổi toàn bộ chiều cao của chữ đối với tất cả chữ trong bản vẽ.	<i>/Sample/VBA/</i>
Lệnh gọi ngoài	Một Macro VBA gọi một chức năng ngoài Visual Basic 6 ở DLL đã đăng kí.	<i>/Sample/ActiveX/ExtrnCall/ExternalCall.dvb</i>
Tiện ích	Thực hiện kết nối AutoCAD với một cơ sở dữ liệu.	<i>/Sample/ActiveX/Facility/Setup/Setup.exe</i>
Tuỳ biến VBA IDE	Tạo một thanh công cụ mới trong VBA IDE cho phép bạn tải một dự án và mang tới VBA Manager, VBA Macros và hộp thoại VBA Options từ IDE.	<i>/Sample/VBA/VBAIDEMenu/acad.dvb</i>
Xuất thuộc tính	Xuất khối dữ liệu AutoCAD và đưa vào bảng tính.	<i>/Sample/ActiveX/ExtAttr/ExtAttr.xls</i>
Xuất thuộc tính ra tệp văn bản	Chuyển các dữ liệu đặc thù thành văn bản Microsoft Word, bảng tính Excel và biểu đồ.	<i>/Sample/VBA/attext.dvb</i>
Liên kết với Excel	Chỉ ra cách chuyển dữ liệu từ AutoCAD sang Excel và ngược lại.	<i>/Sample/VBA/ExcelLink.dvb</i>
Thay thế khối	Thay thế các khối đã được chèn trong bản vẽ với quy định khác.	<i>/Sample/VBA/BlockReplace.dvb</i>
Thay đổi độ rộng	Thay đổi độ rộng của toàn bộ các đường trong bản vẽ.	<i>/Sample/VBA/chplywid.dvb</i>
Vẽ đường tâm	Vẽ đường trục cho các hình cung, elip và hình tròn.	<i>/Sample/VBA/cntrline.dvb</i>
Vẽ đường	Chỉ ra cách vẽ một đường từ một cửa sổ VBA.	<i>/Sample/VBA/drawline.dvb</i>
Mã ví dụ	Tất cả mã nguồn mẫu, trừ các sự kiện ví dụ, trong sách hướng dẫn này và tài liệu tham khảo ActiveX and VBA Reference.	<i>/Sample/VBA/Example_Code.dvb</i>
Sự kiện ví dụ	Các sự kiện ví dụ trong ActiveX and VBA Referene.	<i>/Sample/VBA/Example_Events.dvb</i>
Các đặc tính khác	Minh họa các hàm API Automation sử dụng giao diện hộp thoại.	<i>/Sample/VBA/acad_cg.dvb</i>

Tài liệu tham khảo  
BM Tự động hoá TKCĐ

# LÀM QUEN VỚI VBA

Chương này sẽ giới thiệu về dự án AutoCAD VBA và môi trường phát triển VBA (VBA IDE). Nhìn chung các môi trường phát triển VBA đều tương tự nhau, nhưng AutoCAD VBA IDE có một số đặc tính riêng. Ngoài ra, AutoCAD cũng có một số lệnh dùng để nạp, thực thi dự án, mở dự án trong VBA IDE. Chương này sẽ giới thiệu chung về dự án VBA, lệnh VBA và VBA IDE.

Trong chương này

1

- **Khái niệm về dự án VBA nhúng và độc lập**
- **Tổ chức Dự án bằng VBA Manager**
- **Xử lý Macro**
- **Hiệu chỉnh dự án bằng VBA IDE**
- **Bài tập mở đầu**
- **Thông tin thêm**
- **Nhắc lại các thuật ngữ về dự án AutoCAD VBA**
- **Nhắc lại về lệnh AutoCAD VBA**

# 1. Khái niệm về dự án VBA nhúng và độc lập

Một dự án AutoCAD VBA là một tập hợp các mô đun mã lệnh, các mô đun lớp và các Form<sup>1</sup>. Chúng làm việc cùng nhau để thực hiện một nhiệm vụ định trước. Dự án có thể được lưu trực tiếp bên trong bản vẽ AutoCAD hoặc lưu ở một tệp riêng.

Dự án nhúng được lưu bên trong bản vẽ AutoCAD. Tất cả các dự án đều được tự động tải lên mỗi khi bản vẽ đó được mở trong AutoCAD. Điều này giúp ta phân phối chương trình rất thuận tiện. Các dự án nhúng thường bị giới hạn và không thể mở hoặc đóng bản vẽ AutoCAD vì chúng chỉ hoạt động bên trong bản vẽ có chứa nó. Người sử dụng các dự án nhúng không cần phải tìm và nạp các tệp dự án trước khi muốn chạy chương trình. Chương trình ghi lại thời gian khi mở bản vẽ là một ví dụ về dự án nhúng trong một bản vẽ. Với Macro dạng này, người dùng có thể truy cập và ghi lại thời gian làm việc trên bản vẽ. Người dùng không cần phải nạp dự án mỗi khi mở bản vẽ, mọi thứ đều đã được thực hiện một cách tự động.

Dự án độc lập được lưu trong một tệp riêng và linh hoạt hơn rất nhiều bởi vì chúng có thể mở, đóng và làm việc trong bất cứ bản vẽ AutoCAD nào nhưng lại không tự động tải lên mỗi khi mở bản vẽ. Người sử dụng cần phải biết rõ tệp dự án nào chứa Macro mà họ cần và phải tải dự án đó trước khi có thể thực thi được Macro đó. Tuy nhiên, dự án độc lập có thể được chia sẻ dễ dàng hơn và có thể tạo thành những thư viện chứa những Macro thông dụng. Chẳng hạn như ta có thể tạo Macro và lưu trong một tệp riêng để thực hiện nhiệm vụ là thu thập nhu cầu vật tư trong từng bản vẽ. Macro này do người quản lý thực thi sau khi kết thúc công việc để thu thập thông tin từ rất nhiều bản vẽ khác nhau.

Ở bất kỳ thời điểm nào, người dùng đều có thể tải dự án nhúng và dự án độc lập trong cùng một phiên làm việc của AutoCAD.

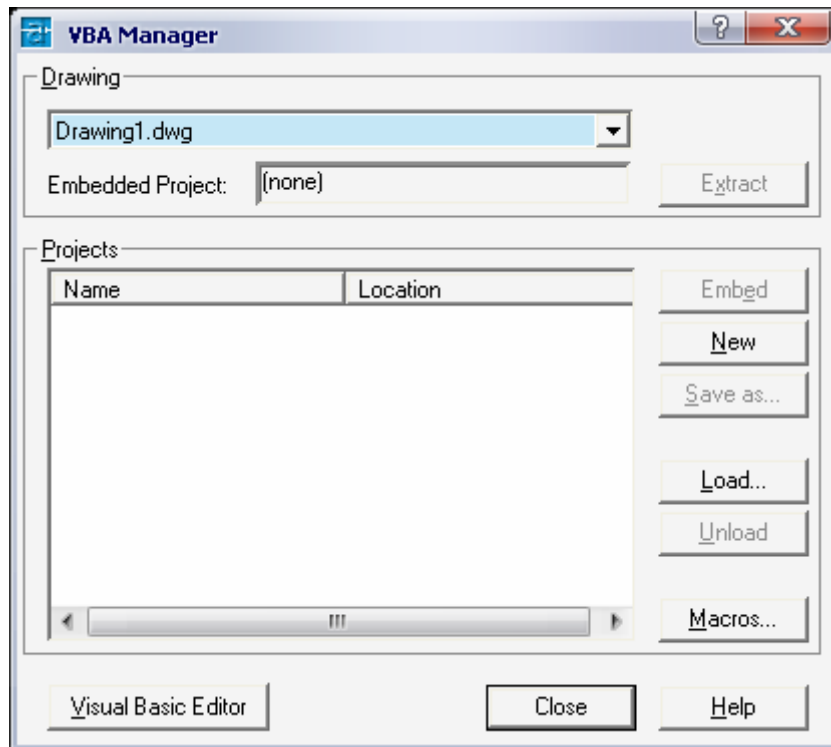
Dự án AutoCAD VBA không hoàn toàn tương thích với dự án Visual Basic. Tuy nhiên, Form, các mô đun và các lớp vẫn có thể chuyển đổi với nhau bằng cách sử dụng lệnh `IMPORT` và `EXPORT` trong môi trường lập trình VBA. Để có thêm thông tin, xin xem thêm phần “*Hiệu chỉnh dự án bằng VBA IDE*” trang 34.

# 2. Tổ chức Dự án bằng VBA Manager

Ta có thể xem tất cả các dự án VBA đã được tải trong phiên làm việc hiện hành của AutoCAD bằng cách sử dụng VBA Manager. Đây là một công cụ của AutoCAD cho phép ta tiến hành tải vào, dỡ bỏ, lưu, tạo mới, nhúng và trích các dự án VBA.

---

<sup>1</sup> **Form** ở đây được hiểu là một cửa sổ được tạo ra trong giai đoạn lập trình để bố trí giao diện của chương trình lên trên nó. Khi thực thi chương trình thì Form chính là cửa sổ hoạt động của chương trình. Do trong tiếng Việt chưa có từ tương đương nên chúng tôi sử dụng từ gốc tiếng Anh.



### Khởi động VBA Manager

- 1 Từ trình đơn Tool, chọn Macro → VBA Manager.
- 2 Hoặc trong AutoCAD thực hiện lệnh VBAMAN.

### 2.1. Tải một dự án đã có

Khi tải một dự án vào trong AutoCAD, thì người dùng có thể sử dụng được ngay tất cả các thủ tục kiểu public, còn gọi là Macro. Những dự án nhúng được tải ngay khi mở bản vẽ. Còn những dự án lưu trong tệp DVB (dự án độc lập) phải được tải riêng.

#### Tải tệp dự án VBA đã có

- 1 Trong VBA Manager, sử dụng lựa chọn Load để hiện hộp thoại Open VBA Project.
- 2 Trong hộp thoại OpenVBA Project, chọn tệp dự án cần mở. Hộp thoại VBA Project sẽ cho phép ta chỉ mở tệp DVB. Nếu ta cố tình mở một kiểu tệp khác, chương trình sẽ báo lỗi.
- 3 Chọn Open.

Ta cũng có thể tải một tệp dự án bằng cách sử dụng lệnh VBALOAD – sẽ mở hộp thoại Open VBA Project.

Ngoài ra, mỗi khi tải một dự án, tất cả những dự án được tham chiếu trong dự án đầu tiên cũng sẽ được tự động tải lên.

AutoCAD cũng sẽ tự động tải lúc khởi động bất kỳ một dự án nào có tên là *acad.dvb*.



### 2.1.1. Cảnh báo Vi-rút

Mỗi khi tải một dự án, AutoCAD thường có những lựa chọn cho phép kích hoạt hoặc không kích hoạt mã lệnh bên trong dự án nhằm bảo vệ tránh vi-rút. Nếu ta kích hoạt đoạn mã, vi-rút có trong đoạn mã sẽ bắt đầu thực thi. Nếu không kích hoạt đoạn mã, dự án sẽ vẫn được tải nhưng tất cả các đoạn mã trong dự án sẽ không được thực thi.

Để có thêm thông tin về vấn đề chống vi-rút, xin xem thêm phần “*Thiết lập các tùy chọn trong dự án*” trang 33.

## 2.2. Dỡ bỏ dự án

Dỡ bỏ dự án sẽ giải phóng bộ nhớ và đảm bảo số lượng hợp lý các dự án đã được tải lên để dễ dàng quản lý.

Ta không thể dỡ bỏ những dự án nhúng hoặc những dự án được tham chiếu bởi những dự án đang được tải trong AutoCAD.

### Để dỡ bỏ dự án VBA

- 1 Trong VBA Manager, chọn dự án cần dỡ bỏ.
- 2 Chọn Unload.
- 3 Hoặc, sử dụng lệnh VBAUNLOAD. Lệnh này sẽ nhắc người dùng nhập vào tên dự án cần dỡ bỏ.

## 2.3. Nhúng dự án vào bản vẽ

Khi ta tiến hành nhúng một dự án vào bản vẽ, tức là ta đã sao chép dự án đó vào trong cơ sở dữ liệu của bản vẽ. Và kể từ đó, dự án sẽ được tải và dỡ bỏ mỗi khi mở và đóng bản vẽ.

Tại một thời điểm, một bản vẽ chỉ có thể có một dự án nhúng. Nếu bản vẽ đã có một dự án nhúng, ta cần phải dỡ bỏ dự án đó ra trước khi tiến hành nhúng một dự án khác vào trong bản vẽ.

### Nhúng dự án vào bản vẽ AutoCAD

- 1 Mở VBA Manager và chọn dự án cần nhúng.
- 2 Chọn Embed.

## 2.4. Tách dự án VBA ra khỏi bản vẽ

Khi tách một dự án VBA, tức là ta đã tiến hành dỡ bỏ dự án đó ra khỏi cơ sở dữ liệu của bản vẽ và có thể lưu dự án đó vào một tệp khác. Nếu ta không lưu tệp trong một tệp dự án khác thì dự án đó sẽ bị xóa đi.

### Để tách một dự án VBA khỏi bản vẽ AutoCAD

- 1 Mở VBA Manager và chọn bản vẽ có chứa dự án cần tách ra.
- 2 Chọn Extract.
- 3 Nếu muốn lưu dự án vào một tệp khác, chọn Yes khi được hỏi “Do you want to export the VBA project before removing it?” - “Bạn có muốn xuất dự án VBA

trước khi xóa đi không?”. Khi đó hộp thoại Save As sẽ xuất hiện để có thể lưu vào một tệp ngoài.

Nếu không muốn lưu dự án vào tệp ngoài, ta chỉ cần chọn No. Dự án sẽ được xóa khỏi bản vẽ và sẽ không được lưu vào tệp ngoài.

## 2.5. Tạo dự án mới

Các dự án mới tạo đều là dự án độc lập và chưa được lưu. Khi đã tạo dự án, ta có thể nhúng dự án vào một bản vẽ hoặc lưu vào một tệp dự án ngoài.

### Để tạo dự án VBA mới

- 1 Mở VBA Manager.
- 2 Chọn New.

Một dự án mới sẽ được tạo ra với tên mặc định là ACADProject. Để đổi tên dự án, cần phải vào VBA IDE, xin xem thêm phần “Đặt tên dự án” trang 38.

## 2.6. Lưu dự án

Dự án nhúng sẽ được lưu mỗi khi lưu bản vẽ. Dự án độc lập phải được lưu bằng VBA Manager hoặc VBA IDE.

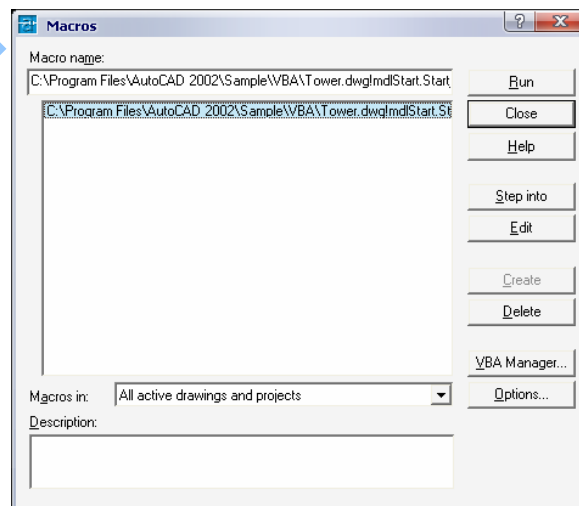
### Để lưu dự án bằng VBA Manager

- 1 Mở VBA Manager và chọn dự án cần lưu.
- 2 Chọn Save As để mở hộp thoại Save As.
- 3 Chọn tên tệp để lưu.
- 4 Chọn Save.

## 3. Xử lý Macro

Hộp thoại Macro cho phép ta thực thi, hiệu chỉnh, xóa hoặc tạo Macro cũng như thiết lập các lựa chọn cho dự án VBA.

Macro là một chương trình con kiểu public (có thể thực thi được). Mỗi dự án thường chứa ít nhất là một Macro.



## Để mở hộp thoại Macro

- 1 Từ menu Tools chọn Macro►Macro.
- 2 Hoặc, trong AutoCAD gõ lệnh VBARUN.

Tên của tất cả Macro trong một phạm vi nào đó sẽ được hiển thị trong hộp thoại. Để thay đổi phạm vi này, ta chọn một mục trong danh sách trong mục *Macros in*. Danh sách này xác định dự án hoặc bản vẽ mà có Macro đang được hiển thị. Ta có thể tùy chọn hiển thị Macro trong:

- Tất cả các bản vẽ và dự án
- Tất cả các bản vẽ
- Tất cả các dự án
- Một bản vẽ đang mở trong AutoCAD
- Một dự án đang được tải trong AutoCAD

Bằng cách thay đổi các phạm vi này, ta có thể điều khiển được số lượng tên Macro có trong danh sách. Điều này sẽ thực sự hữu ích khi có rất nhiều Macro được tải lên trong các bản vẽ và dự án.

### 3.1. Thực thi Macro

Thực thi Macro nghĩa là thực hiện mã lệnh của Macro trong phiên làm việc hiện hành của AutoCAD. Bản vẽ hiện hành được xem là bản vẽ đang được mở và đang thực thi Macro trên đó. Tất cả các tham chiếu đến đối tượng `ThisDrawing` sẽ tham chiếu đến bản vẽ hiện hành đối với các Macro trong dự án độc lập. Còn đối với các Macro trong dự án nhúng, đối tượng `ThisDrawing` luôn là bản vẽ có chứa Macro đó.

#### Để thực thi Macro

- 1 Mở hộp thoại Macros và chọn Macro sẽ thực thi.
- 2 Chọn Run.

### 3.2. Hiệu chỉnh Macro

Quá trình hiệu chỉnh Macro diễn ra trong VBA IDE, trong cửa sổ Code. Xin xem thêm ở mục “*Hiệu chỉnh dự án bằng VBA IDE*” trang 34.

#### Để hiệu chỉnh Macro

- 1 Mở hộp thoại Macro và chọn Macro cần hiệu chỉnh.
- 2 Chọn Edit.

### 3.3. Truy cập vào Macro

Quá trình truy cập vào Macro sẽ bắt đầu thực thi Macro và sau đó dừng quá trình thực thi lại ở dòng mã lệnh đầu tiên. VBA IDE sẽ được mở ra với cửa sổ mã lệnh tương ứng của Macro đó.

#### Để truy cập vào Macro

1 Trong hộp thoại Macros, chọn Macro cần truy cập.

2 Chọn Step.

### 3.4. Tạo mới Macro

Ta có thể tạo mới một Macro rỗng.

#### Để tạo Macro mới

1 Mở hộp thoại Macros và nhập vào tên của Macro mới cần tạo.

2 Trong danh sách thả xuống của mục Macros in, chọn dự án, nơi mà ta muốn tạo Macro.

3 Chọn Create.

Nếu tên đó đã có, chương trình sẽ hỏi xem có muốn thay thế Macro đã có hay không.

Nếu ta chọn Yes, mã lệnh trong Macro đã có sẽ bị xóa và thay vào đó sẽ là một Macro rỗng mới.

Nếu ta chọn No, chương trình sẽ quay trở lại hộp thoại Macros để nhập vào tên Macro mới.

Nếu ta chọn Cancel, chương trình sẽ đóng hộp thoại Macros và sẽ không tạo Macro nào cả.

### 3.5. Xóa Macro

Ta có thể xóa Macro của bất kỳ một dự án nào.

#### Để xóa macro

1 Mở hộp thoại Macros và chọn Macro cần xóa.

2 Chọn Delete. Chương trình sẽ nhắc xác nhận lại quá trình xóa Macro.

3 Chọn Yes để xóa Macro, No để hủy quá trình xóa.

### 3.6. Thiết lập các tùy chọn trong dự án

Có thể thiết lập 3 tùy chọn cho các dự án AutoCAD VBA:

- Cho phép tự động nhúng (Enabling auto embedding)
- Cho phép ngắt khi có lỗi (Allowing break on errors)
- Kích hoạt bảo vệ tránh vi-rút (Enabling macro virus protection)

#### Để thiết lập các tùy chọn trong dự án AutoCAD VBA

1 Từ trình đơn Tools chọn Macro►Macros để mở hộp thoại VBA Macros.

2 Trong hộp thoại VBA Macros, chọn Options để mở hộp thoại Options.

3 Trong hộp thoại Options, chọn tùy chọn cần kích hoạt.

4 Chọn OK.

### 3.6.1. Cho phép tự động nhúng

Tính năng tự động nhúng sẽ tự động tạo dự án VBA nhúng cho tất cả các bản vẽ khi mở bản vẽ

### 3.6.2. Cho phép ngắt khi có lỗi

Cho phép VBA chuyển sang chế độ Break khi có lỗi. Chế độ Break sẽ tạm thời “treo” chương trình đang thực thi để chuyển sang môi trường phát triển ứng dụng. Ở chế độ này, ta có thể kiểm tra, gỡ lỗi, khởi động lại, gỡ lỗi từng bước hoặc tiếp tục thực thi chương trình.

Khi chọn tùy chọn này, những lỗi chưa được xử lý khi xuất hiện trong quá trình thực thi Macro sẽ dừng Macro lại và hiển thị VBA IDE tại nơi phát sinh lỗi trong Macro.

Khi không chọn tùy chọn này, những lỗi không bắt được trong quá trình thực thi sẽ làm hiển thị hộp thông báo về lỗi đó và kết thúc quá trình thực thi Macro.

### 3.6.3. Kích hoạt bảo vệ tránh vi-rút

Cơ chế bảo vệ tránh vi-rút sẽ hiển thị thông báo đã được tích hợp sẵn mỗi khi mở bản vẽ mà có khả năng chứa vi-rút Macro.

## 4. Hiệu chỉnh dự án bằng VBA IDE

Sau khi dự án đã được tải trong AutoCAD, ta có thể hiệu chỉnh mã nguồn, Form, và các tham chiếu trong dự án sử dụng môi trường phát triển ứng dụng tương tác của VBA.

### 4.1. Mở VBA IDE

Sau khi mở VBA IDE, ta có thể truy xuất đến tất cả các dự án đã được tải lên.

#### Để mở VBA IDE

Ta có thể mở VBA IDE từ dòng lệnh hoặc từ thanh trình đơn

- Từ dòng lệnh, ta nhập VBAIDE.
- Hoặc, từ trình đơn Tools, chọn Macro → Visual Basic Editor.

#### Để mở VBA IDE tự động khi khởi động AutoCAD

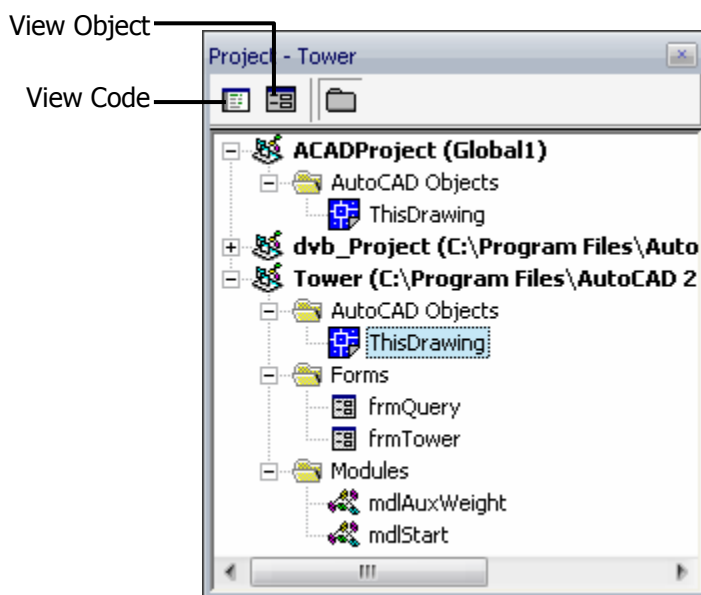
Nếu muốn mở VBA IDE tự động mỗi khi khởi động AutoCAD cần phải thêm dòng sau vào tệp *acad.rx*:

```
acadvba.arx
```

### 4.2. Xem thông tin về dự án

Trong VBA IDE có một cửa sổ gọi là cửa sổ Project sẽ hiển thị danh sách tất cả các dự án VBA đã được tải. Cửa sổ này cũng hiển thị mã lệnh, lớp và các mô đun, Form có trong dự án, các tài liệu liên quan đến dự án, tất cả các dự án khác được dự án này tham chiếu và các đường dẫn trong dự án.

Cửa sổ Project cũng có thanh công cụ riêng có thể sử dụng để mở các thành phần khác nhau của dự án. Sử dụng nút View Code để mở mã lệnh của mô đun đang được chọn. Sử dụng nút View Object để hiển thị các đối tượng được chọn, chẳng hạn như Form.



Mặc định thì cửa sổ Project luôn được hiển thị, còn nếu không thì ta chọn trình đơn View►Project window, hoặc nhấn CTRL+R.

### 4.3. Định nghĩa các thành phần trong một dự án

Mỗi dự án có thể có nhiều thành phần khác nhau. Các thành phần đó có thể là đối tượng (objects), Form, các mô đun chuẩn (standard modules), mô đun lớp (class modules) và các tham chiếu (references).

#### 4.3.1. Đối tượng

Thành phần đối tượng thể hiện loại đối tượng, hoặc tài liệu mà mã lệnh VBA sẽ truy cập. Trong các dự án AutoCAD VBA, đối tượng này thể hiện cho bản vẽ AutoCAD hiện hành.

#### 4.3.2. Form

Thành phần Form chứa tất cả các hộp thoại mà ta tạo ra trong dự án.

#### 4.3.3. Mô đun chuẩn

Thành phần mô đun mã lệnh chứa tất cả các hàm và thủ tục. Mô đun chuẩn còn được gọi là mô đun mã lệnh hoặc đơn giản hơn chỉ cần gọi là mô đun.

#### 4.3.4. Mô đun lớp

Thành phần mô đun lớp chứa tất cả các đối tượng mà ta tự tạo ra, và được định nghĩa là lớp.

#### 4.3.5. Tham chiếu

Thành phần Tham chiếu chứa tất cả các tham chiếu đến các dự án và các thư viện khác.

#### 4.3.6. Thêm Thành phần mới

Thêm thành phần mới là tạo ra một thành phần rỗng trong dự án. Ta có thể thêm mô đun, Form, mô đun lớp mới vào dự án. Nhiệm vụ của chúng ta là sẽ phải cập nhật tất cả các thuộc tính của thành phần đó (chẳng hạn như tên của thành phần) hoặc thêm vào các đoạn mã lệnh thích hợp. Khi đặt tên các thành phần cần phải nhớ rằng có thể là trong tương lai chương trình sẽ được những người lập trình khác tham khảo, vì thế cần phải theo quy ước đặt tên của nhóm phát triển ứng dụng.

##### Để thêm Thành phần mới vào dự án

- 1 Trong cửa sổ Project của VBA IDE, chọn dự án cần thêm Thành phần.
- 2 Từ trình đơn Insert, chọn UserForm, Module hoặc Class Module để thêm thành phần mới vào dự án.

Thành phần mới được tạo ra sẽ được thêm vào dự án và sẽ xuất hiện trong cửa sổ Project.

#### 4.4. Nhập những thành phần đã có

Tính năng này cho phép thêm một thành phần đã có vào trong dự án. Ta có thể nhập Form, mô đun hoặc mô đun lớp. Form được nhập thông qua tệp FRM, mô đun thì thông qua tệp BAS, còn mô đun lớp thì thông qua tệp CLS.

Khi nhập một tệp có chứa Thành phần, một bản sao của tệp nhập vào sẽ được thêm vào trong dự án. Tệp nguồn sẽ được giữ nguyên. Những thay đổi với thành phần vừa nhập vào sẽ không làm thay đổi tệp nguồn.

Nếu ta nhập một tệp chứa thành phần mà tên thành phần đã có, chương trình sẽ thêm một con số vào tên của tệp được nhập vào dự án.

##### Để nhập một thành phần đã có vào dự án

- 1 Trong cửa sổ Project trong VBA IDE, chọn dự án mà ta cần thêm thành phần.
- 2 Từ trình đơn Tệp, chọn Import File để mở hộp thoại Import File.
- 3 Trong hộp thoại Import File, chọn tệp cần nhập và nhấn Open.

Thành phần vừa được nhập vào sẽ thêm vào dự án và sẽ xuất hiện trong cửa sổ Project. Để hiệu chỉnh thuộc tính của một thành phần, chọn thành phần đó trong cửa sổ Project. Thuộc tính của thành phần được chọn sẽ được liệt kê và có thể được hiệu chỉnh trong cửa sổ Properties.

#### 4.5. Hiệu chỉnh các thành phần

Ta có thể hiệu chỉnh mô đun chuẩn, mô đun lớp và các Form trong VBA IDE. Mô đun chuẩn và mô đun lớp được hiệu chỉnh trong cửa sổ Code. Form thì được hiệu chỉnh trong cửa sổ UserForm sử dụng các công cụ đặc biệt khác.

##### Để hiệu chỉnh một thành phần trong dự án

- 1 Trong cửa sổ Project của VBA IDE, chọn thành phần cần hiệu chỉnh.
- 2 Chọn nút ViewCode trong cửa sổ Project để mở cửa sổ Code.
- 3 Chọn nút ViewObject trong cửa sổ Project để mở cửa sổ UserForm và các thanh công cụ liên quan.

Ta có thể mở tất cả các cửa sổ Code ứng với từng mô-đun có trong dự án để có thể dễ dàng xem các đoạn mã ở những mô-đun, Form khác nhau và sao chép, dán đoạn mã lệnh trong các cửa sổ Code.

#### Để truy cập vào mã lệnh ứng với Form

- Bấm đúp chuột trên bất kỳ điều khiển nào trên cửa sổ Form. Mã lệnh tương ứng với điều khiển đó sẽ được mở lên trong cửa sổ Code.

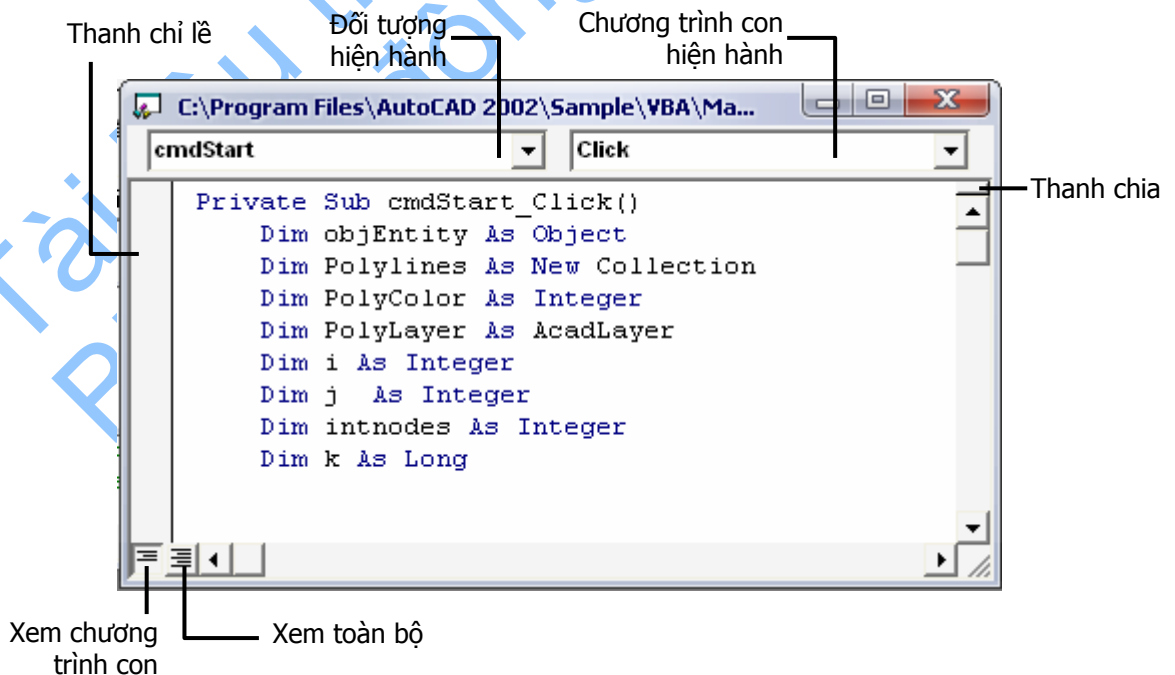
#### 4.5.1. Sử dụng cửa sổ mã lệnh

Cửa sổ mã lệnh có chứa 2 danh sách đồ xuống, một thanh chia, một thanh chỉ lề, biểu tượng xem toàn bộ và xem chương trình con.

Thanh chia bên cạnh phải của cửa sổ Code cho phép tách cửa sổ theo đường ngang. Chỉ cần kéo thanh chia xuống để tạo một khung cửa sổ khác. Tính năng này cho phép xem đồng thời hai phần của đoạn mã lệnh trong cùng một mô-đun. Để đóng khung cửa sổ này, chỉ cần kéo thanh chia trở về vị trí ban đầu.

Thanh chỉ lề nằm bên trái của cửa sổ Code, được sử dụng để biểu thị phần lề sử dụng trong quá trình soạn thảo mã lệnh và quá trình gỡ lỗi.

Biểu tượng xem toàn bộ và xem chương trình con nằm ở góc dưới bên trái của cửa sổ Code dùng để chuyển đổi giữa chế độ xem toàn bộ mã lệnh và chế độ chỉ xem mã lệnh của một chương trình con.





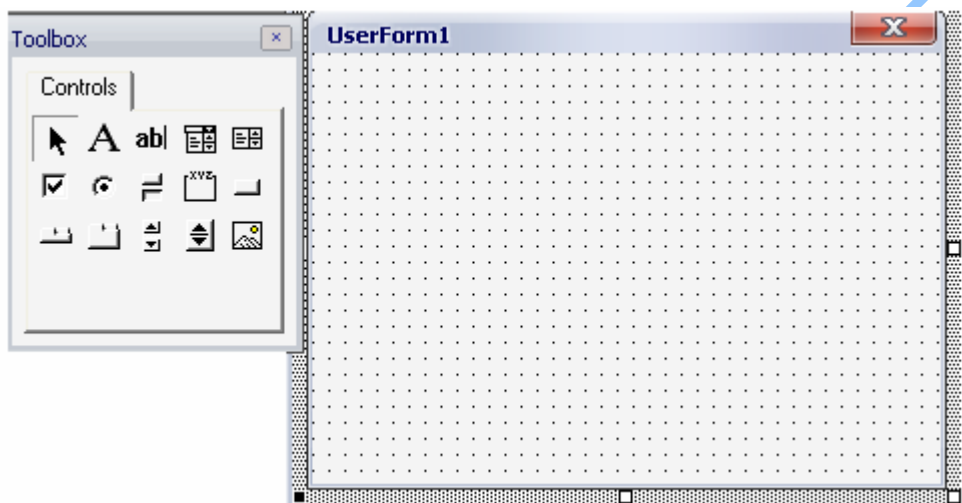
## 4.5.2. Sử dụng cửa sổ UserForm

Cửa sổ UserForm cho phép người dùng tạo các hộp thoại trong dự án của mình.

Để thêm một điều khiển chỉ cần kéo điều khiển cần thêm trong thanh công cụ và thả vào trên Form. Ta có thể thiết lập canh hàng cho điều khiển trong thẻ General của hộp thoại Options, hoặc có thể chọn hiển thị lưới, xác định kích thước lưới hiển thị trên Form. (Xem thêm phần “Thiết lập các tùy chọn trong VBA IDE” trang 40 để biết thêm về hộp thoại Options.)

Mỗi Form khi thiết kế đều có các nút bấm Maximize, Minimize và Close. Những nút này được cài đặt sẵn.

Để thêm mã lệnh cho điều khiển, chỉ cần bấm đúp chuột lên điều khiển đã được đặt trên Form. Thao tác này sẽ hiển thị cửa sổ Code của điều khiển đó.



## 4.6. Thực thi Macro

Ngoài cách thực thi macro từ hộp thoại Macros, ta còn có thể thực thi Macro từ VBA IDE.

### Để thực thi Macro từ VBA IDE

- Từ trình đơn Run, chọn Run Macro.

Nếu không có Macro hoặc Form hiện hành, một hộp thoại sẽ hiện lên cho phép chọn Macro để thực thi.

Nếu có một Macro hiện hành (con trỏ đang nằm trong một chương trình con nào đó), Macro đó sẽ được thực thi.

## 4.7. Đặt tên dự án

Nên chọn tên dự án và tên của tệp *.dvb*, nơi lưu dự án, là khác nhau. Tên của tệp *.dvb* chỉ là tên tệp để lưu dự án, còn tên của dự án phải được thiết lập trong cửa sổ Properties trong VBA IDE.

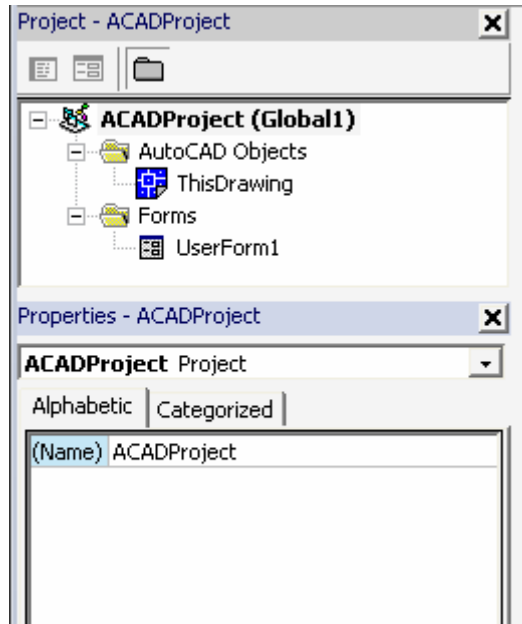
Nếu không thiết lập tên dự án và tên tệp, AutoCAD sẽ tự động gán tên mặc định sau:

Tên dự án: *ACADProject*

Tên tệp: *Project.dvb*

### Để thay đổi tên dự án

- 1 Trong cửa sổ Project của VBA IDE, chọn dự án cần đổi tên.
- 2 Trong cửa sổ Properties, hiệu chỉnh thuộc tính Name của dự án.



### Để thay đổi tên tệp của dự án

- 1 Trong VBA IDE, chọn Save từ trình đơn File
- 2 Trong hộp thoại Save As, nhập vào tên và thư mục lưu tệp dự án.

## 4.8. Lưu Dự án

Không có lệnh *SAVE* trong AutoCAD dành riêng cho các dự án VBA. Thay vào đó, lệnh *SAVE* tích hợp trong trình đơn File của VBA IDE và trong VBA Manager. Bất kỳ thay đổi nào trong dự án VBA sẽ phát sinh truy cập vào hộp thoại Save VBA Project khi xảy ra một trong những sự kiện sau:

- Chọn lệnh *SAVE* trong VBA IDE.
- Chọn *Save As* trong VBA Manager.
- Kết thúc phiên làm việc của AutoCAD mà vẫn chưa lưu dự án VBA.

---

**CHÚ Ý** Trước khi lưu dự án, dự án đã được gán một tên mặc định là *project.dvb*. Ta cần phải thay đổi tên mới cho dự án khi tiến hành lưu dự án. Nếu ta lưu dự án với tên mặc định là *project.dvb*, ta sẽ không còn có khả năng tạo một dự án mới, trống được nữa. Khi đó, mỗi khi tạo một dự án mới, VBA sẽ tải dự án có tên tệp là *project.dvb*.

---

## 4.9. Tham chiếu dự án VBA khác

Tham chiếu một dự án VBA từ một dự án khác giúp lập trình viên chia sẻ mã lệnh dễ dàng hơn. Các lập trình viên có thể tạo thư viện các Macro thường được sử dụng nhất và tham chiếu đến thư viện đó mỗi khi cần. Điều này giúp tập trung hóa quá

trình quản lý mã lệnh, mà vẫn cho phép nhiều lập trình viên khác tận dụng được mã lệnh.

### **Để tham chiếu dự án VBA khác**

- 1 Trong cửa sổ Project của VBA IDE, chọn dự án cần thêm tham chiếu.
- 2 Từ trình đơn Tools, chọn Reference để mở hộp thoại References.
- 3 Trong hộp thoại Reference, chọn nút Browse để mở hộp thoại Add Reference.
- 4 Trong hộp thoại Add Reference, chọn nút OK để hoàn thành quá trình thêm tham chiếu.

Sau khi đã thêm tham chiếu, ta sẽ thấy một thư mục mới trong cửa sổ Project của VBA IDE. Thư mục này chứa tất cả các tham chiếu và tên của các dự án được tham chiếu.

Khi đã tham chiếu một dự án, ta có thể sử dụng tất cả các mã lệnh hoặc Form kiểu public trong dự án đó.

Sau khi một dự án đã được tải lên, tất cả các dự án được tham chiếu trong dự án đó cũng được tự động tải lên. Những dự án được tham chiếu không thể đóng lại chừng nào dự án chính vẫn còn đang được mở.

Ta không thể tạo tham chiếu vòng. Có nghĩa là không thể tạo tham chiếu đến một dự án mà lại có tham chiếu đến dự án ban đầu. Nếu ta vô tình tạo ra tham chiếu vòng, VBA sẽ thông báo cho ta biết.

Tham chiếu dự án là một tính năng tiêu chuẩn của Microsoft VBA. Ta không cần phải làm gì thêm trong AutoCAD để kích hoạt tính năng này. Ta có thể tìm thêm thông tin về tham chiếu dự án trong các tài liệu trợ giúp về Microsoft Visual Basic. Ta cũng có thể truy cập vào tệp trợ giúp về Microsoft Visual Basic trong trình đơn Help của VBA IDE.

---

**CHÚ Ý** Ta không thể tham chiếu các dự án nhúng hoặc các dự án VBA từ những chương trình ứng dụng khác.

---

## **4.10. Thiết lập các tùy chọn trong VBA IDE**

Ta có thể thay đổi các đặc điểm của VBA IDE sử dụng hộp thoại Options. Để mở hộp thoại Options, ta chọn trình đơn Tools►Options.

Hộp thoại Options có 4 thẻ: Editor, Editor Format, General và Docking.

### **4.10.1. Editor**

Thẻ Editor dùng để thiết lập cấu hình cho cửa sổ Code và Project.

Cấu hình cho cửa sổ Code bao gồm:

- Auto Syntax Check – Tự động kiểm tra cú pháp
- Require Variable Declaration – Yêu cầu khai báo biến
- Auto List Member – Tự động hiển thị các thành phần
- Auto Quick Info – Tự động cung cấp thông tin nhanh

- Auto Data Tips – Tự động hiện chú thích dữ liệu
- Auto Indent – Tự động thụt đầu dòng
- Tab Width – Khoảng cách Tab

Cấu hình cho cửa sổ Window bao gồm:

- Drag and Drop Text Editing – Cho phép kéo thả khi soạn thảo
- Default to Full Module View – Mặc định chế độ xem toàn bộ
- Procedure Separator Display – Hiện thị phân cách giữa các chương trình con

#### 4.10.2. Editor Format

Thẻ Editor Format xác định diện mạo của phần mã lệnh Visual Basic.

Ta có thể:

- Thay đổi màu mã lệnh
- Thay đổi màu tiền cảnh
- Thay đổi màu hậu cảnh
- Thay đổi thanh chỉ lè
- Thay đổi màu và cỡ chữ
- Hiện thị hoặc ẩn thanh chỉ lè
- Hiện thị hoặc ẩn đoạn ký tự ví dụ tương ứng với các cấu hình

#### 4.10.3. General

Thẻ General dùng để thiết lập các cấu hình, xử lý lỗi và cấu hình khi biên dịch cho dự án Visual Basic hiện hành.

Ta có thể

- Thay đổi cấu hình lưới của Form
- Hiện thị hoặc ẩn chú giải thanh công cụ
- Thiết lập tự động che lấp cửa sổ
- Chọn nhận thông báo khi bị mất trạng thái – Xác định xem có nhận thông báo khi thao tác cần thực hiện sẽ làm tắt cả các biến trong các mô đun bị thiết lập lại khi thực thi dự án
- Xác định cách thức xử lý lỗi
- Thiết lập xem dự án sẽ được biên dịch khi có yêu cầu hoặc thực hiện quá trình biên dịch nền.

#### 4.10.4. Docking

Thẻ Docking cho phép ta lựa chọn tính năng neo cửa sổ hay không.

## 5. Bài tập mở đầu

Phần trước đã giới thiệu những khái niệm cơ bản nhất về lập trình trong AutoCAD VBA. Bây giờ ta sẽ thực hiện bài tập đầu tiên: tạo chương trình “Hello World”. Trong bài tập này, ta sẽ tạo một bản vẽ AutoCAD mới, thêm một dòng chữ vào bản vẽ, sau đó lưu bản vẽ, tất cả đều thực hiện từ VBA.

### Tạo đối tượng text “Hello World”

- 1 Mở VBA IDE bằng cách nhập dòng sau vào dòng lệnh AutoCAD:  
Command: **VBAIDE**
- 2 Mở cửa sổ Code bằng cách chọn lựa chọn Code từ trình đơn Menu của VBA IDE.
- 3 Tạo mới một chương trình con trong dự án bằng cách chọn trình đơn Insert►Procedure trong VBA IDE.
- 4 Khi được nhắc nhập thông tin cho chương trình con, nhập vào tên chẳng hạn như **HelloWorld**. Cần phải chắc chắn rằng mục Type được chọn là Sub, và mục Scope được chọn là Public.
- 5 Chọn OK.
- 6 Nhập vào đoạn mã sau (để mở một bản vẽ mới) giữa hai dòng Public Sub HelloWorld() and End Sub.  

```
ThisDrawing.Application.Documents.Add
```
- 7 Nhập vào đoạn mã sau (để tạo chuỗi và xác định điểm chèn) ngay phía sau đoạn mã đã nhập trong bước 6.  

```
Dim insPoint(0 To 2) As Double ' Khai báo điểm chèn của dòng chữ
Dim textHeight As Double ' Khai báo chiều cao chữ
Dim textStr As String
Dim textObj As AcadText ' Khai báo đối tượng Text của AutoCAD
insPoint(0) = 2 ' Thiết lập tọa độ cho điểm chèn
insPoint(1) = 4
insPoint(2) = 0
textHeight = 1 ' Thiết lập chiều cao chữ bằng 1
textStr = "Hello World!" ' Thiết lập nội dung dòng chữ
' Tạo đối tượng Text
Set textObj = ThisDrawing.ModelSpace.AddText _
(textStr, insPoint, textHeight)
```
- 8 Nhập đoạn mã (để lưu bản vẽ) ngay phía sau đoạn mã đã nhập ở bước 7.  

```
ThisDrawing.SaveAs ("Hello.dwg")
```
- 9 Thực thi chương trình bằng cách chọn trình đơn Run►RunSub/UserForm trong VBA IDE.

Sau khi chương trình đã thực thi xong, chương trình AutoCAD sẽ được hiện lên phía trên cùng. Ta có thể thấy dòng chữ “HelloWorld!” hiển thị trên bản vẽ và tên bản vẽ là *Hello.dwg*.

## 6. Thông tin thêm

Microsoft có cung cấp tệp trợ giúp về VBA IDE và ngôn ngữ lập trình Visual Basic.

## Để truy cập các tệp trợ giúp của Microsoft về VBA IDE

- Từ trình đơn Help trong VBA IDE, chọn Microsoft Visual Basic Help.

## 7. Nhắc lại các thuật ngữ về dự án AutoCAD VBA

Dự án độc lập (Global Project)

Là một dự án VBA được lưu trong tệp *.dwb*.

Dự án nhúng (Embedded Project)

Là một dự án VBA lưu kèm trong bản vẽ AutoCAD.

Bản vẽ thông thường (Regular Document)

Là bản vẽ AutoCAD không chứa dự án VBA nhúng.

Bản vẽ thông minh (Smart Document)

Là bản vẽ có chứa một hoặc nhiều dự án VBA nhúng.

Dự án hiện hành (Current Project)

Là dự án hiện đang được chọn trong VBA IDE.

ThisDrawing

Là một thuật ngữ trong VBA thể hiện bản vẽ hiện hành. Đối với các dự án độc lập, ThisDrawing luôn tham chiếu đến bản vẽ đang được kích hoạt trong AutoCAD. Với các dự án nhúng, ThisDrawing tham chiếu đến bản vẽ có chứa dự án đó.

VBA IDE

Là môi trường phát triển ứng dụng tương tác. Chương trình này cho phép ta soạn thảo mã lệnh và Form trong dự án, hoặc sao chép sang các dự án khác. Nó còn cho phép tham chiếu đến các mô hình đối tượng trong các ứng dụng khác.

VBA Manager

VBA Manager cho phép ta quản lý các dự án. Ta có thể tạo, xóa, nhúng hoặc trích xuất các dự án. Ta có thể xem xem có dự án nhúng nào trong bản vẽ đang được mở hay không.

Hộp thoại Macros (Macros Dialog Box)

Hộp thoại Macros cho phép ta thực thi, xóa, tạo các Macro mới và có nhiều tùy chọn khác cho dự án VBA.

## 8. Nhắc lại về lệnh AutoCAD VBA

VBA IDE      Khởi động VBA IDE.

VBA IDE cho phép ta hiệu chỉnh, thực thi và gỡ rối chương trình. Mặc dù VBA IDE chỉ khởi động khi AutoCAD đang được thực thi, nhưng nó có thể được thu nhỏ, mở và đóng độc lập so với cửa sổ chương trình AutoCAD.

VBALOAD	Tải dự án VBA vào phiên làm việc hiện hành của AutoCAD.
VBARUN	Thực thi Macro VBA từ hộp thoại Macros hoặc từ dòng lệnh AutoCAD.
VBAUNLOAD	Dỡ bỏ dự án VBA trong phiên làm việc hiện hành của AutoCAD. Nếu dự án VBA đã được chỉnh sửa và chưa lưu, hộp thoại Save Project hiện lên nhắc người dùng lưu dự án (hoặc hiển thị trên dòng lệnh).
VBAMAN	Hiển thị VBA Manager cho phép người dùng xem, tạo, tải, đóng, nhúng hoặc trích xuất các dự án.
VBASTMT	Thực thi dòng lệnh VBA từ dòng lệnh AutoCAD.



# CÁC KHÁI NIỆM CƠ BẢN VỀ ActiveX Automation

Để sử dụng AutoCAD ActiveX Automation có hiệu quả, ta cần phải quen thuộc với các thực thể (entity), đối tượng (object), và các đặc tính của AutoCAD liên quan đến loại chương trình được dự định phát triển. Nếu càng biết nhiều về các thuộc tính đồ họa hoặc phi đồ họa của đối tượng thì càng dễ dàng sử dụng AutoCAD ActiveX Automation.

Luôn nhớ rằng phần trợ giúp của AutoCAD ActiveX Automation luôn sẵn sàng – chỉ cần nhấn F1. Nếu gặp vấn đề gì với bất kỳ đối tượng, phương thức, hay thuộc tính, chỉ cần lựa chọn đối tượng, phương thức hay thuộc tính đó trong VBA IDE và nhấn F1.

## Trong chương này 2

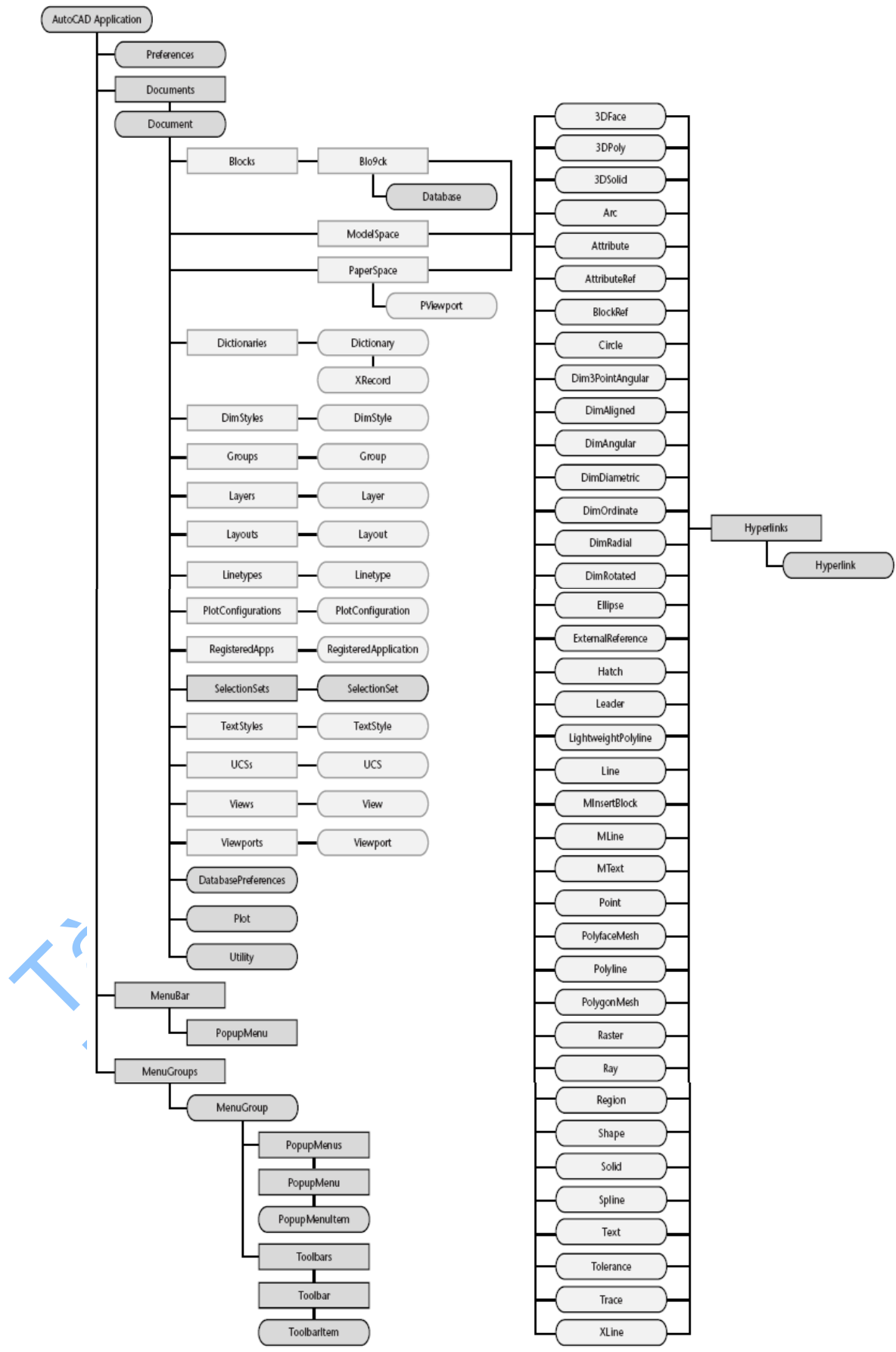
- **Tìm hiểu mô hình đối tượng trong AutoCAD**
- **Truy xuất cây phân cấp đối tượng**
- **Làm việc với tập đối tượng**
- **Tìm hiểu Phương thức và Thuộc tính**
- **Tìm hiểu Đối tượng gốc**
- **Thư viện kiểu**
- **Gọi lại thực thể đầu tiên trong CSDL**
- **Sử dụng Variant trong Phương thức và Thuộc tính**
- **Sử dụng các ngôn ngữ lập trình khác**

## 1. Tìm hiểu mô hình đối tượng trong AutoCAD

Một đối tượng là một khối cấu thành chính của giao diện AutoCAD ActiveX. Mỗi một đối tượng thể hiện đúng một phần của AutoCAD. Có rất nhiều loại đối tượng khác nhau trong giao diện AutoCAD ActiveX. Ví dụ

- Đối tượng đồ họa: line, arc, text và dimension
- Cấu hình về kiểu dáng (style settings): linetype và dimension style
- Cấu trúc tổ chức: layer, group, block
- Thể hiện bản vẽ: view, viewport
- Và ngay cả bản thân bản vẽ trong chương trình AutoCAD cũng được xem là một đối tượng.

Các đối tượng được xây dựng theo quan hệ phân cấp, trong đó đối tượng Application là gốc. Cấu trúc phân nhánh này còn được gọi là Mô Hình Đối Tượng. Mô hình đối tượng giúp người dùng có thể biết đối tượng nào có thể truy cập vào đối tượng nào ở cấp tiếp theo.

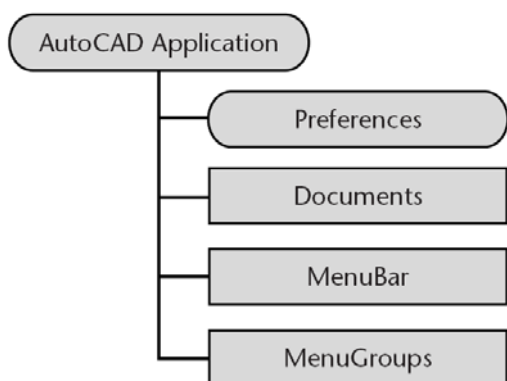


## 1.1. Sơ lược về đối tượng Application

Đối tượng Application là đối tượng gốc của mô hình đối tượng trong AutoCAD ActiveX Automation. Từ đối tượng Application, ta có thể truy xuất đến bất kỳ đối tượng nào khác, hoặc thuộc tính hoặc phương thức gán cho bất kỳ đối tượng nào.

Ví dụ, đối tượng Application có thuộc tính Preferences trả về đối tượng Preferences. Đối tượng này cho phép truy cập đến các cấu hình bên trong của hộp thoại Option. (Các cấu hình lưu trong bản vẽ nằm trong đối tượng DatabasePreferences, sẽ được đề cập sau). Các thuộc tính khác của đối tượng Application cho phép truy cập đến các dữ liệu riêng của chương trình chẳng hạn như tên và phiên bản chương trình, và kích thước, vị trí, tính nhìn thấy của AutoCAD. Các phương thức của đối tượng Application thực hiện các thao tác trong chương trình như liệt kê, nạp, gỡ bỏ chương trình ADS và ARX, thoát khỏi AutoCAD.

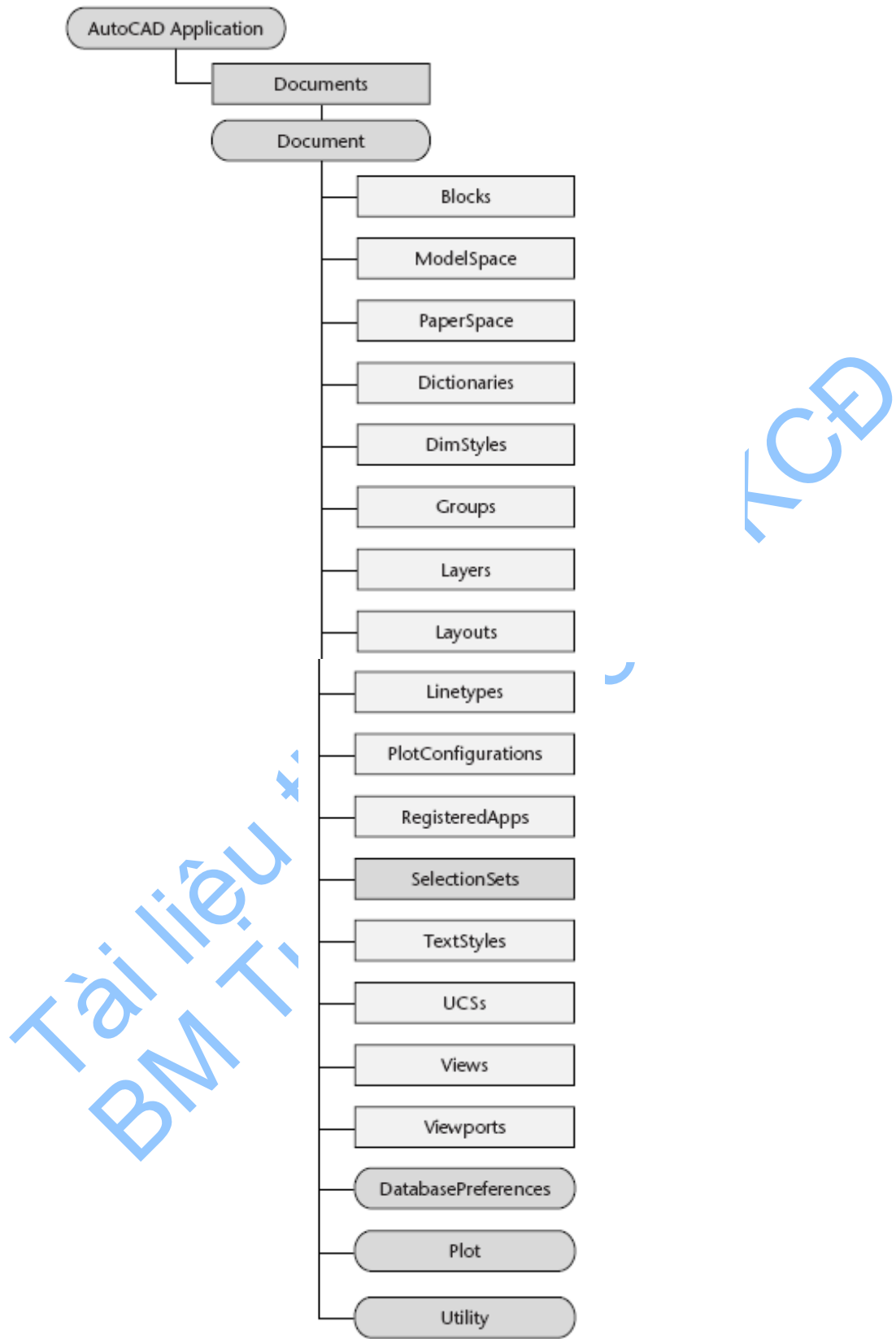
Đối tượng Application cũng có các liên kết đến bản vẽ AutoCAD thông qua tập đối tượng Documents, các trình đơn và thanh công cụ AutoCAD thông qua tập đối tượng MenuBar và MenuGroups, và VBA IDE thông qua một thuộc tính gọi là VBE.



Đối tượng Application là đối tượng toàn cục trong giao diện ActiveX. Điều này có nghĩa là tất cả các phương thức và thuộc tính của đối tượng Application luôn có hiệu lực trong không gian tên toàn cục.

## 1.2. Sơ lược về đối tượng Document

Đối tượng Document, thực chất là một bản vẽ AutoCAD, thuộc tập đối tượng Documents cho phép truy cập vào tất cả các đối tượng đồ họa và hầu hết các đối tượng phi đồ họa của AutoCAD. Các đối tượng đồ họa (đường thẳng, hình tròn, cung, ...) được truy cập thông qua tập ModelSpace và PaperSpace, còn các đối tượng phi đồ họa (layer, linetype, text style, ...) được truy cập thông qua tập đối tượng có tên tương tự, chẳng hạn như Layers, Linetypes, TextStyles. Đối tượng Document còn cho phép truy cập đến đối tượng Plot và Utility.



### 1.3. Sơ lược về tập đối tượng

AutoCAD nhóm hầu hết các đối tượng vào trong tập đối tượng. Mặc dù một tập đối tượng chứa nhiều loại đối tượng khác nhau nhưng khi xử lý được sử dụng những kỹ thuật tương tự nhau. Mỗi một tập đối tượng có một phương thức dùng để thêm đối tượng vào bản thân tập đối tượng đó và hầu hết các tập đối tượng đều sử dụng phương thức Add để thực hiện nhiệm vụ này. Tuy nhiên mỗi thực thể<sup>1</sup> thường được thêm vào phương thức có tên là Add<Tên thực thể>, chẳng hạn như để thêm vào một đường thẳng (Line) ta sử dụng phương thức AddLine.

Các tập đối tượng cũng có những phương thức và thuộc tính giống nhau. Thuộc tính Count dùng để truy cập bộ đếm số đối tượng trong tập đối tượng. Phương thức Item sử dụng để truy cập bất kỳ đối tượng nào trong tập đối tượng.

### 1.4. Sơ lược về các đối tượng Đồ họa và Phi đồ họa

Các đối tượng đồ họa, còn gọi là thực thể, là những đối tượng hữu hình cấu thành bản vẽ (đường thẳng, hình tròn, ảnh raster<sup>2</sup>...). Để tạo những đối tượng này, ta sử dụng phương thức Add<Tên thực thể> tương ứng. Để hiệu chỉnh hoặc truy vấn các đối tượng, ta sử dụng các phương thức và thuộc tính của bản thân từng đối tượng. Mỗi đối tượng đồ họa đều có các thuộc tính cho phép ứng dụng có thể thực hiện hầu hết các lệnh hiệu chỉnh đối tượng trong AutoCAD như Copy, Erase, Move, Mirror... Những đối tượng này còn có phương thức để xác lập và gọi lại các dữ liệu mở rộng (xdata), lựa chọn và cập nhật, và lấy lại hình bao của đối tượng. Các đối tượng đồ họa đều có các thuộc tính điển hình như Layer, Linetype, Color, và Handle cũng như những thuộc tính riêng biệt, phụ thuộc vào loại đối tượng, chẳng hạn như Center, Radius, và Area.

Các đối tượng phi đồ họa là những đối tượng không thể nhìn thấy được (đối tượng thông tin) chẳng hạn như Layer, Linetype, DimStyle, SelectionSets... Để tạo những đối tượng này, sử dụng phương thức Add của đối tượng tập đối tượng cha. Còn để hiệu chỉnh và truy vấn các đối tượng thì sử dụng các phương thức và thuộc tính riêng của từng đối tượng. Mỗi đối tượng phi đồ họa đều có các phương thức và thuộc tính đặc biệt tương ứng với từng mục đích; tất cả đều có các phương thức để thiết lập và gọi lại dữ liệu mở rộng (xdata), và xoá bản thân đối tượng.

### 1.5. Sơ lược về đối tượng Preferences, Plot và Utility

Dưới đối tượng Preferences là một tập hợp các đối tượng khác tương ứng với từng thẻ trong hộp thoại Options. Đồng thời, những đối tượng này đều cho phép truy cập vào các thiết lập được lưu trong dữ liệu của hộp thoại Options. Những thiết lập được lưu cùng bản vẽ đều nằm trong đối tượng DatabasePreferences. Ta còn có thể thiết lập và hiệu chỉnh các lựa chọn (và biến hệ thống không nằm trong hộp thoại Options) sử dụng phương thức SetVariable và GetVariable. Thông tin chi tiết ở mục “Thiết lập các lựa chọn trong AutoCAD”.

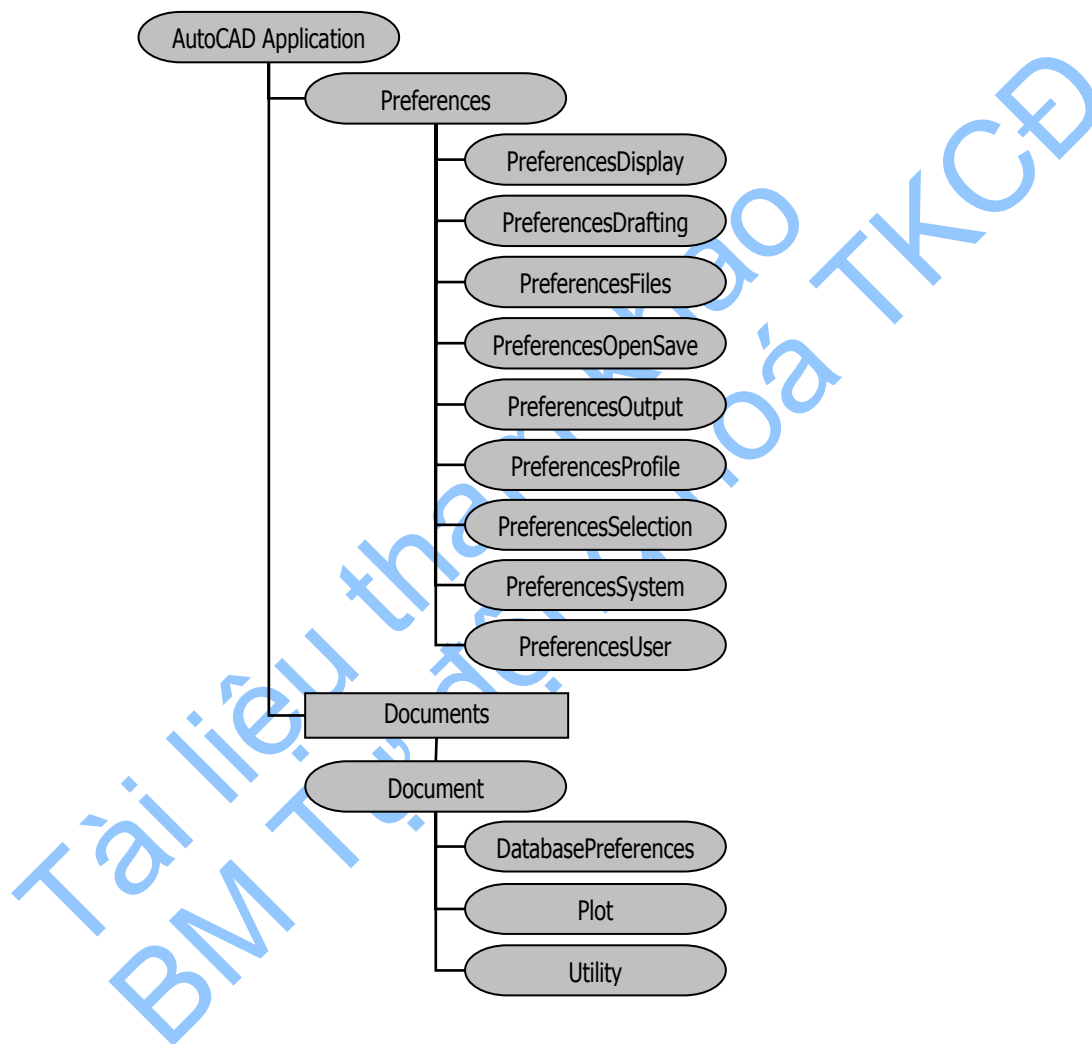
---

<sup>1</sup> **Thực thể (Entity):** trong một tập hợp có chứa nhiều đối tượng khác nhau, mỗi đối tượng như vậy được gọi là một thực thể.

<sup>2</sup> **Ảnh Raster:** là ảnh kiểu ánh xạ bit hay ảnh dạng màn hình, các hình ảnh được thể hiện bởi các chấm nhỏ riêng biệt.

Đối tượng Plot cho phép truy cập đến những thiết lập trong hộp thoại Plot và cho phép ứng dụng có thể in bản vẽ theo nhiều phương thức khác nhau. Thông tin chi tiết về việc in ấn, xin xem thêm mục “*In bản vẽ*” trang 244.

Đối tượng Utility bao gồm các hàm về nhập liệu của người dùng và hàm chuyển đổi. Hàm nhập liệu của người dùng là những phương thức nhắc người dùng AutoCAD nhập vào rất nhiều dạng dữ liệu khác nhau trong của sổ dòng lệnh, chẳng hạn như chuỗi, số nguyên, số thực, điểm... Các hàm chuyển đổi là những phương thức thực thi trên các dữ liệu đặc biệt của AutoCAD như điểm và góc phục vụ cho việc xử lý chuỗi và số. Để có thêm thông tin chi tiết về hàm nhập liệu của người dùng, xem thêm mục “*Nhắc người dùng nhập liệu*” trang 84.



## 2. Truy xuất cây phân cấp đối tượng

Việc truy xuất cây phân cấp đối tượng được thực hiện dễ dàng ngay bên trong VBA. Đó là do VBA thực thi cùng tiến trình với phiên làm việc hiện tại của AutoCAD và vì vậy việc kết nối với ứng dụng không cần thêm bước nào nữa.

VBA có thể liên kết đến bản vẽ hiện hành trong phiên làm việc hiện tại của AutoCAD nhờ đối tượng `ThisDrawing`. Nhờ có đối tượng `ThisDrawing` mà người lập trình có thể truy cập tức thời đến đối tượng `Document` hiện hành và tất cả các thuộc tính, phương thức cũng như các đối tượng khác trong cấu trúc cây phân cấp.



Khi được sử dụng trong dự án độc lập, đối tượng `ThisDrawing` luôn luôn tương ứng với bản vẽ hiện hành trong AutoCAD. Còn nếu là dự án nhúng, đối tượng `ThisDrawing` ứng với bản vẽ có chứa dự án nhúng đó. Lấy ví dụ dòng lệnh dưới đây trong dự án độc lập sẽ lưu bất cứ bản vẽ hiện hành nào trong AutoCAD:

```
ThisDrawing.Save
```

## 2.1. Tham chiếu đối tượng trong Cấu trúc cây phân cấp đối tượng

Người lập trình có thể tham chiếu đối tượng một cách trực tiếp hoặc thông qua biến người dùng định nghĩa. Để tham chiếu đối tượng trực tiếp, phải bao hàm đối tượng trong cấu trúc phân cấp. Chẳng hạn như đoạn mã sau đây sẽ thêm một đường thẳng vào trong không gian mô hình. Chú ý rằng cấu trúc phân cấp bắt đầu bằng `ThisDrawing`, sau đó đến đối tượng `ModelSpace` và sau đó gọi phương thức `AddLine`:

```
Dim startPoint(0 To 2) As Double, endPoint(0 To 2) As Double
Dim LineObj as AcadLine
startPoint(0) = 0: startPoint(1) = 0: startPoint(2) = 0
endPoint(0) = 30: endPoint(1) = 20: endPoint(2) = 0
Set LineObj = ThisDrawing.ModelSpace.AddLine(startPoint,endPoint)
```

Để tham chiếu đối tượng thông qua biến người dùng định nghĩa, người lập trình phải định nghĩa biến theo đúng kiểu mong muốn, sau đó gán biến cho một đối tượng thích hợp. Ví dụ, đoạn mã sau định nghĩa một biến (`moSpace`) kiểu `AcadModelSpace` và gán biến tương ứng với không gian mô hình hiện hành:

```
Dim moSpace As AcadModelSpace
Set moSpace = ThisDrawing.ModelSpace
```

Đoạn mã sau sẽ thêm một đường thẳng vào không gian mô hình sử dụng biến người dùng định nghĩa:

```
Dim startPoint(0 To 2) As Double, endPoint(0 To 2) As Double
Dim LineObj as AcadLine
startPoint(0) = 0: startPoint(1) = 0: startPoint(2) = 0
endPoint(0) = 30: endPoint(1) = 20: endPoint(2) = 0
Set LineObj = moSpace.AddLine(startPoint,endPoint)
```

## 2.2. Truy xuất đối tượng Application

Do đối tượng `ThisDrawing` cho phép liên kết đến đối tượng `Document`, người lập trình có thể tự hỏi, làm thế nào để truy xuất đến đối tượng gốc rễ nhất (đối tượng `Application`), nó nằm trên cả đối tượng `Document` trong cấu trúc cây phân cấp đối tượng. Đối tượng `Document` có một thuộc tính gọi là `Application`, cho phép liên kết đến đối tượng `Application`.

Lấy ví dụ, đoạn mã sau sẽ cập nhật lại bản vẽ:

```
ThisDrawing.Applications.Update
```

## 3. Làm việc với Tập đối tượng

Tập đối tượng là một đối tượng được định nghĩa trước chứa tất cả các thực thể tương tự nhau. Những đối tượng sau có thể hình thành nên tập đối tượng:

Documents Collection	bao gồm tất cả các bản vẽ đang mở trong phiên làm việc hiện hành của AutoCAD.
ModelSpace Collection	bao gồm tất cả các đối tượng đồ hoạ (thực thể) trong không gian mô hình.
PaperSpace Collection	bao gồm tất cả các đối tượng đồ hoạ trong không gian in hiện hành.
Block Object	bao gồm tất cả các thực thể nằm trong một khối nào đó.
Blocks Collection	bao gồm tất cả các khối trong bản vẽ.
Dictionaries Collection	bao gồm tất cả các từ điển trong bản vẽ.
DimStyles Collection	bao gồm tất cả các kiểu kích thước trong bản vẽ.
Groups Collection	bao gồm tất cả các nhóm trong bản vẽ.
Hyperlinks Collection	bao gồm tất cả các siêu liên kết của một thực thể nào đó.
Layers Collection	bao gồm tất cả các lớp (layer) trong bản vẽ.
Layouts Collection	bao gồm tất cả các layout trong bản vẽ.
Linetypes Collection	bao gồm tất cả các kiểu đường (linetype) trong bản vẽ.
MenuBar Collection	bao gồm tất cả các trình đơn hiện có trong AutoCAD.
MenuGroups Collection	bao gồm tất cả các trình đơn và thanh công cụ đã được tải vào AutoCAD.
RegisteredApplications Collection:	bao gồm tất cả các ứng dụng đã đăng ký trong bản vẽ.
SelectionSets Collection	bao gồm tất cả các đối tượng được chọn trong bản vẽ.
TextStyles Collection	bao gồm tất cả các kiểu văn bản trong bản vẽ.
UCSs Collection	bao gồm tất cả các hệ toạ độ người dùng trong bản vẽ.
Views Collection	bao gồm tất cả các cảnh nhìn trong bản vẽ.
Viewports Collection	bao gồm tất cả các khung nhìn trong bản vẽ.

### 3.1. Truy xuất Tập đối tượng

Hầu hết các tập đối tượng đều được truy xuất thông qua đối tượng Document. Đối tượng Document có một thuộc tính tương ứng với mỗi tập đối tượng. Lấy ví dụ, đoạn mã sau định nghĩa một biến và gán cho tập đối tượng Layers trong bản vẽ hiện tại:

```
Dim layerCollection as AcadLayers
Set layerCollection = ThisDrawing.Layers
```

Tập đối tượng Documents, MenuBar và MenuGroups được truy xuất thông qua đối tượng Application. Đối tượng Application cũng có một thuộc tính tương ứng với từng tập đối tượng. Ví dụ như đoạn mã sau sẽ định nghĩa một biến và gán cho tập đối tượng MenuGroups của ứng dụng:

```
Dim MenuGroupsCollection as AcadMenuGroups
Set MenuGroupsCollection = ThisDrawing.Application.MenuGroups
```

## 3.2. Thêm đối tượng mới vào Tập đối tượng

Để thêm một đối tượng mới vào tập đối tượng, ta sử dụng phương thức Add. Đoạn mã sau sẽ tạo một lớp mới và thêm vào trong tập đối tượng Layers:

```
Dim newLayer as AcadLayer
Set newLayer = ThisDrawing.Layers.Add("MyNewLayer")
```

## 3.3. Duyệt Tập đối tượng

Để chọn một đối tượng nào đó trong tập đối tượng, ta sử dụng phương thức Item. Phương thức Item cần có một từ định danh đóng vai trò như là chỉ số để xác định vị trí của đối tượng trong tập đối tượng cũng như là chuỗi thể hiện tên của đối tượng.

Ví dụ sau sẽ duyệt qua một tập đối tượng và hiển thị tên của tất cả các lớp trong tập đối tượng:

### Duyệt qua tập đối tượng Layers

```
Sub Ch2_IterateLayer()
    ' Duyệt qua tập đối tượng
    On Error Resume Next

    Dim I As Integer
    Dim msg As String
    msg = ""
    For I = 0 To ThisDrawing.Layers.Count - 1
        msg = msg + ThisDrawing.Layers.Item(I).Name + vbCrLf
    Next
    MsgBox msg
End Sub
```

Đoạn ví dụ dưới đây sử dụng phương thức Item để tìm một lớp có tên là "ABC":

### Tìm lớp có tên "ABC"

```
Sub Ch2_FindLayerABC()
    ' Sử dụng phương thức Item để tìm lớp tên "ABC"
    On Error Resume Next

    Dim ABCLayer As AcadLayer
    Set ABCLayer = ThisDrawing.Layers.Item("ABC")
    If Err <> 0 Then
        MsgBox "The layer 'ABC' does not exist."
    End If
End Sub
```

---

**CHÚ Ý:** Không sử dụng các phương thức hiệu chỉnh thực thể (Copy, Array, Mirror...) cho bất cứ một đối tượng nào khi đang đồng thời duyệt qua tập đối tượng sử dụng cấu trúc For...Each. Thay vào đó, có thể hoàn thành quá trình duyệt tập đối tượng trước khi hiệu chỉnh hoặc tạo một mảng tạm và gán bằng tập đối tượng sau đó có thể duyệt trên mảng vừa sao chép và thực hiện việc hiệu chỉnh bình thường.

---

## 3.4. Xoá một đối tượng khỏi Tập đối tượng

Để xoá một đối tượng nào đó, sử dụng phương thức Delete trong đối tượng tìm được. Ví dụ đoạn mã sau sẽ xoá lớp ABC:

```
Dim ABCLayer as AcadLayer
```

```
Set ABCLayer = ThisDrawing.Layers.Item("ABC")
ABCLayer.Delete
```

Một khi đối tượng đã bị xoá đi thì nên nhớ là về sau không bao giờ được truy cập lại đối tượng đó trong chương trình.

## 4. Tìm hiểu Phương thức và Thuộc tính

Mỗi đối tượng đều có thuộc tính và phương thức tương ứng. Thuộc tính mô tả các khía cạnh của từng đối tượng, còn phương thức là hành động có thể được thực hiện trên từng đối tượng. Sau khi tạo mới đối tượng, ta có thể truy vấn và hiệu chỉnh đối tượng thông qua các thuộc tính và phương thức của đối tượng đó.

Ví dụ, một đối tượng Circle (hình tròn) có thuộc tính Center. Thuộc tính này thể hiện tâm của hình tròn trong hệ toạ độ toàn cục 3 chiều. Để thay đổi tâm của hình tròn, chỉ cần gán thuộc tính này với toạ độ khác. Đối tượng Circle còn có một phương thức có tên là Offset. Phương thức này tạo một đối tượng mới nằm cách hình tròn cũ một khoảng trống cho trước. Để thấy tất cả các thuộc tính và phương thức của đối tượng Circle, tham khảo thêm phần đối tượng Circle trong *ActiveX and VBA Reference* trong AutoCAD.

## 5. Tìm hiểu Đối tượng gốc

Mỗi đối tượng đều được liên kết thường xuyên với một đối tượng gốc. Tất cả các đối tượng đều được phát sinh từ một đối tượng gọi là đối tượng gốc. Người lập trình có thể truy xuất đến tất cả các đối tượng trong cùng giao diện bằng cách lần theo liên kết từ đối tượng gốc đến các đối tượng con. Thêm vào đó, tất cả các đối tượng đều có một thuộc tính gọi là Application có liên kết trực tiếp ngược trở lại với đối tượng gốc.

Đối tượng gốc của giao tiếp AutoCAD là chương trình AutoCAD.

## 6. Thư viện kiểu

Các đối tượng, thuộc tính, và phương thức truy xuất qua *Automation Object* đều nằm trong thư viện kiểu. Một thư viện kiểu là một tệp hay một phần của tệp mô tả kiểu của một hoặc nhiều đối tượng. Các thư viện kiểu không lưu đối tượng mà chỉ lưu thông tin. Nhờ truy xuất vào thư viện kiểu, các ứng dụng và trình duyệt mới có thể xác định được những đặc tính của đối tượng, chẳng hạn như các giao tiếp mà đối tượng hỗ trợ và tên, địa chỉ của các bộ phận bên trong giao tiếp đó. Trước khi sử dụng *Automation Object* trong chương trình, người lập trình cần phải tham chiếu đến thư viện kiểu của nó. Quá trình tham chiếu tự động thực hiện trong AutoCAD VBA. Với các môi trường phát triển ứng dụng tương tác khác, người lập trình cần phải tự thực hiện việc tham chiếu.

Người lập trình có thể sử dụng các đối tượng của chương trình mà không cần thực hiện việc tham chiếu thư viện kiểu của chương trình đó. Tuy nhiên, nên thêm tham chiếu đến thư viện kiểu vì những lý do sau:

- Các hàm có khả năng truy cập toàn cục có thể được truy xuất trực tiếp mà không cần chứng nhận.

- Việc sử dụng hàm, thuộc tính, và phương thức có thể được kiểm tra lúc dịch để đảm bảo sự chính xác, và do đó thời gian thực thi sẽ nhanh hơn lúc chạy chương trình.
- Có thể khai báo biến có kiểu đã được định nghĩa trong thư viện, như vậy sẽ tăng độ tin cậy và khả năng dễ đọc trong lúc chạy chương trình.

## 7. Gọi lại Thực thể Đầu Tiên trong CSDL

Ví dụ sau trả lại thực thể đầu tiên trong không gian mô hình. Với các thực thể trong không gian in, vẫn có thể áp dụng đoạn mã này.

### Gọi lại đối tượng đầu tiên trong không gian mô hình

```
Sub Ch2_FindFirstEntity()
    'Ví dụ này gọi lại đối tượng đầu tiên trong không gian mô hình
    On Error Resume Next
    Dim entity As AcadEntity
    If ThisDrawing.ModelSpace.count <> 0 Then
        Set entity = ThisDrawing.ModelSpace.Item(0)
        MsgBox entity.ObjectName + _
            " là đối tượng đầu tiên trong không gian mô hình."
    Else
        MsgBox "Không có đối tượng nào trong không gian mô hình."
    End If
End Sub
```

## 8. Sử dụng Variant trong phương thức và thuộc tính

AutoCAD ActiveX Automation sử dụng kiểu variant để truyền mảng dữ liệu. Mặc dù điều này có vẻ khá rắc rối với những người chưa có kinh nghiệm, nhưng một khi đã hiểu được cơ bản vấn đề này cũng không quá khó. Thêm vào đó, AutoCAD ActiveX Automation cũng có những tiện ích trợ giúp để chuyển đổi kiểu dữ liệu.

### 8.1. Variant là gì?

Variant là một kiểu dữ liệu đặc biệt có thể lưu bất kỳ loại dữ liệu nào, trừ dữ liệu kiểu chuỗi có độ dài cố định và các kiểu dữ liệu do người dùng định nghĩa. Biến Variant còn có thể chứa các giá trị đặc biệt Empty, Error, Nothing và NULL. Có thể xác định cách thức xử lý biến kiểu variant sử dụng các hàm của Visual Basic: VarType hoặc TypeName.

Người lập trình có thể sử dụng biến kiểu variant trong những trường hợp hầu hết các dữ liệu cần được xử lý theo cách linh động hơn.

### 8.2. Sử dụng biến Variant trong dữ liệu mảng.

Biến Variant được sử dụng để truyền dữ liệu vào và ra AutoCAD ActiveX Automation. Điều này có nghĩa là mảng dữ liệu phải là kiểu variant để tương thích với các phương thức và thuộc tính trong AutoCAD ActiveX Automation. Ngoài ra,

dữ liệu mảng xuất ra từ AutoCAD ActiveX Automation phải được xử lý như là dữ liệu kiểu Variant.

---

**CHÚ Ý:** Trong AutoCAD, mảng dữ liệu đầu vào của VBA được tự động chuyển đổi thành kiểu variant. Nghĩa là người lập trình không phải sử dụng mảng variant để truyền dữ liệu vào trong các phương thức và thuộc tính của ActiveX Automation khi sử dụng chúng trong VBA. Tuy nhiên, tất cả các mảng dữ liệu xuất ra luôn ở dạng variant, do đó cần phải chú ý để xử lý theo cách thích hợp nhất.

---

### 8.3. Chuyển Mảng thành Variant

AutoCAD ActiveX Automation cung cấp các phương thức tiện ích để chuyển đổi một mảng dữ liệu thành Variant. Phương thức này là phương thức CreateTypedArray, nó sẽ tạo biến variant chứa mảng số nguyên, số thực... Sau đó có thể truyền biến variant kết quả vào bất kỳ phương thức hoặc thuộc tính nào của AutoCAD mà chấp nhận mảng số như là biến variant.

Phương thức CreateTypedArray nhận tham số đầu vào là kiểu biến của mảng và dữ liệu của mảng cần được chuyển đổi. Giá trị trả về là biến variant. Đoạn mã sau sẽ chuyển 3 mảng sử dụng CreateTypedArray: tọa độ các điểm khống chế của đường Spline và tiếp tuyến đầu, tiếp tuyến cuối của đường Spline. Sau đó các biến Variant sẽ được truyền vào phương thức AddSpline để tạo một đường spline.

#### Tạo đường Spline sử dụng phương thức CreateTypedArray

```
Sub Ch2_CreateSplineUsingTypedArray()  
    ' Ví dụ tạo đường Spline trong không gian mô hình sử dụng  
    ' sử dụng Phương thức CreateTypedArray.  
    Dim splineObj As AcadSpline  
    Dim startTan As Variant  
    Dim endTan As Variant  
    Dim fitPoints As Variant  
    Dim noOfPoints As Integer  
  
    Dim utilObj As Object  
    ' ràng buộc về sau với đối tượng Utility  
    Set utilObj = ThisDrawing.Utility  
    ' Định nghĩa đối tượng Spline  
    utilObj.CreateTypedArray _  
        startTan, vbDouble, 0.5, 0.5, 0  
    utilObj.CreateTypedArray _  
        endTan, vbDouble, 0.5, 0.5, 0  
    utilObj.CreateTypedArray _  
        fitPoints, vbDouble, 0, 0, 0, 5, 5, 0, 10, 0, 0  
    noOfPoints = 3  
    Set splineObj = ThisDrawing.ModelSpace.AddSpline _  
        (fitPoints, startTan, endTan)  
    ' Xem đường spline vừa tạo  
    ZoomAll  
End Sub
```

### 8.4. Mảng Variant

Thông tin về mảng được truyền ngược lại từ AutoCAD ActiveX Automation là truyền ở dạng biến variant. Nếu người lập trình biết rõ kiểu dữ liệu mảng, chỉ cần truy xuất vào biến variant như một mảng bình thường. Nếu không nắm rõ kiểu dữ

liệu chứa trong biến variant, cần phải sử dụng các hàm VBA VarType hoặc TypeName. Các hàm này trả lại kiểu dữ liệu chứa trong biến variant. Nếu cần duyệt qua mảng người lập trình có thể sử dụng cấu trúc For . . . Each của VBA.

Đoạn mã sau sẽ minh họa cách tính khoảng cách giữa hai điểm do người dùng nhập vào. Trong ví dụ này, kiểu dữ liệu là biết trước vì tất cả các tọa độ đều có kiểu là double. Tọa độ 3D chứa trong mảng có 3 biến double còn tọa độ 2D chứa trong mảng có 2 biến double.

### Tính khoảng cách giữa hai điểm

```
Sub Ch2_TinhKhoangCach()  
    Dim point1 As Variant  
    Dim point2 As Variant  
  
    ' Người dùng nhập tọa độ điểm  
    point1 = ThisDrawing.Utility.GetPoint _  
        (, vbCrLf & "Diem thu nhat: ")  
    point2 = ThisDrawing.Utility.GetPoint _  
        (point1, vbCrLf & "Diem thu hai: ")  
  
    ' Tính khoảng cách giữa point1 và point2  
    Dim x As Double, y As Double, z As Double  
    Dim dist As Double  
    x = point1(0) - point2(0)  
    y = point1(1) - point2(1)  
    z = point1(2) - point2(2)  
    dist = Sqr((Sqr((x ^ 2) + (y ^ 2)) ^ 2) + (z ^ 2))  
  
    ' Trình bày kết quả tính  
    MsgBox "Khoang cach giua hai diem la: " _  
        & dist, , "Tinh Khoang cach"  
End Sub
```

## 9. Sử dụng các ngôn ngữ lập trình khác

Tài liệu này được viết cho ngôn ngữ lập trình VBA, do đó các đoạn mã ví dụ và các ứng dụng mẫu đều được viết theo ngôn ngữ VBA. Để sử dụng những đoạn mã này trong các môi trường lập trình khác, người lập trình cần phải chuyển đổi thích hợp sang môi trường mà mình đã lựa chọn.

### 9.1. Chuyển đổi từ mã VBA sang VB

Để cập nhật các đoạn mã ví dụ sử dụng trong VB, cần phải tham chiếu đến thư viện kiểu của AutoCAD. Để thực hiện điều này trong VB, chọn mục References từ trình đơn Project để mở hộp thoại References. Trong hộp thoại References, chọn AutoCAD 2000 Type Library (hoặc tương đương) và nhấn OK.

Sau đó, trong các đoạn mã ví dụ, thay thế tất cả các phần tham chiếu đến ThisDrawing bằng các biến tự định nghĩa tham chiếu đến bản vẽ hiện hành. Để làm được điều này, định nghĩa một biến cho ứng dụng AutoCAD (myApp) và cho bản vẽ hiện hành (myDoc). Cuối cùng, gán các biến trong chương trình cho các ứng dụng AutoCAD hiện hành. Nếu đang chạy AutoCAD, phương thức GetObject sẽ lấy lại đối tượng AutoCAD Application. Nếu AutoCAD hiện tại chưa được chạy,



lỗi xuất hiện sẽ được chặn lại và sau đó được xóa đi. Và sau đó, phương thức CreateObject sẽ cố gắng tạo đối tượng AutoCAD Application. Nếu thành công, sẽ khởi động được AutoCAD; nếu không, một hộp thoại thông báo xuất hiện mô tả về lỗi đó. Đoạn mã sau có sử dụng thuộc tính Clear và Description của Err. Nếu môi trường phát triển không hỗ trợ các thuộc tính này, ta cần phải điều chỉnh ví dụ cho thích hợp.

### Kết nối với AutoCAD từ Visual Basic

```
Sub Ch2_ConnectToAcad()  
    Dim acadApp As AcadApplication  
    On Error Resume Next  
    Set acadApp = GetObject(, "AutoCAD.Application")  
    If Err Then  
        Err.Clear  
        Set acadApp = CreateObject("AutoCAD.Application")  
        If Err Then  
            MsgBox Err.Description  
            Exit Sub  
        End If  
    End If  
    MsgBox "Now running " + acadApp.Name + _  
        " version " + acadApp.Version  
End Sub
```

Tiếp theo đó, gán đối tượng Document trong AutoCAD cho biến document. Đối tượng Document được trả về bằng thuộc tính ActiveDocument.

```
Dim acadDoc as AcadDocument  
Set acadDoc = acadApp.ActiveDocument
```

Từ đây trở đi, sử dụng biến acadDoc để tham chiếu đến bản vẽ hiện hành của bản vẽ AutoCAD.

---

**CHÚ Ý** Khi có đa phiên làm việc trong AutoCAD, hàm GetObject sẽ trả về thực thể đầu tiên của AutoCAD trong Running Object Table của Windows. Xem thêm tài liệu của Microsoft Visual Basic về Running Object Table (ROT) và hàm GetObject để hiểu thêm về cách xác thực phiên làm việc được trả về khi dùng hàm GetObject.

---

## 9.2. Đoạn mã ví dụ so sánh VBA và VB

Đoạn mã ví dụ sau minh họa việc tạo một đoạn thẳng bằng VBA và VB:

### Tạo một đoạn thẳng sử dụng VBA

```
Sub Ch2_AddLineVBA()  
    ' Ví dụ này sẽ thêm một đoạn thẳng vào Model Space  
    Dim lineObj As AcadLine  
    Dim startPoint(0 To 2) As Double  
    Dim endPoint(0 To 2) As Double  
    ' Xác định điểm đầu và điểm cuối của đoạn thẳng  
    startPoint(0) = 1  
    startPoint(1) = 1  
    startPoint(2) = 0  
    endPoint(0) = 5  
    endPoint(1) = 5  
    endPoint(2) = 0  
    ' Tạo đoạn thẳng trong Model Space
```

```

Set lineObj = ThisDrawing.ModelSpace.AddLine(startPoint, _
                                             endPoint)
' Xem đoạn thẳng vừa mới tạo
ZoomAll
End Sub

```

### Tạo một đoạn thẳng sử dụng VB

```

Sub Ch2_AddLineVB()
On Error Resume Next
' Kết nối với chương trình AutoCAD
Dim acadApp As AcadApplication
Set acadApp = GetObject(, "AutoCAD.Application")
If Err Then
Err.Clear
Set acadApp = CreateObject("AutoCAD.Application")
If Err Then
MsgBox Err.Description
Exit Sub
End If
End If
' Kết nối với bản vẽ AutoCAD
Dim acadDoc As AcadDocument
Set acadDoc = acadApp.ActiveDocument
' Xác định hai đầu đoạn thẳng
Dim lineObj As AcadLine
Dim startPoint(0 To 2) As Double
Dim endPoint(0 To 2) As Double
startPoint(0) = 1
startPoint(1) = 1
startPoint(2) = 0
endPoint(0) = 5
endPoint(1) = 5
endPoint(2) = 0
' Tạo đoạn thẳng trong Model Space
Set lineObj = acadDoc.ModelSpace.AddLine _
(startPoint, endPoint)
ZoomAll
End Sub

```

# ĐIỀU KHIỂN MÔI TRƯỜNG AutoCAD

Chương này trình bày những vấn đề cơ bản cần biết để xây dựng một ứng dụng trong AutoCAD cũng như sẽ giải thích rõ cách thức điều khiển môi trường AutoCAD và cách làm việc hiệu quả trong môi trường này.

Trong chương này

3

- **Mở, Lưu và Đóng các bản vẽ**
- **Thiết lập các lựa chọn trong AutoCAD**
- **Điều khiển cửa sổ ứng dụng**
- **Điều khiển cửa sổ bản vẽ**
- **Thiết lập lại các đối tượng hiện hành**
- **Gán và lấy biến hệ thống**
- **Vẽ với độ chính xác cao**
- **Nhắc người dùng nhập dữ liệu**
- **Truy xuất dòng lệnh của AutoCAD**
- **Thao tác khi không mở bản vẽ nào**
- **Nhập vào các định dạng khác**
- **Xuất sang các định dạng khác**

## 1. Mở, Lưu và Đóng các bản vẽ

Tập Documents và đối tượng Document cho phép thực hiện các hàm liên quan đến tệp trong AutoCAD.

Để tạo một bản vẽ mới, hoặc mở một bản vẽ đã có, ta phải sử dụng các phương thức trong tập đối tượng Documents. Phương thức Add sẽ tạo một bản vẽ mới và thêm bản vẽ đó vào tập đối tượng Documents. Phương thức Open sẽ mở một bản vẽ đã có. Ngoài ra còn có phương thức Close trong tập đối tượng Documents dùng để đóng tất cả các bản vẽ đang mở trong phiên làm việc của AutoCAD.

Để lưu, nhập hoặc xuất một bản vẽ, ta sử dụng các phương thức của đối tượng Document: Save, Save As, Import và Export.

### 1.1. Mở bản vẽ

Ví dụ sau sử dụng phương thức Open để mở một bản vẽ đã có. Hàm Dir của Visual Basic dùng để kiểm tra sự tồn tại của tệp trước khi tiến hành mở bản vẽ. Ta nên thay đổi tên bản vẽ hoặc đường dẫn để chỉ đến một bản vẽ đã có trong hệ thống.

```
Sub Ch3_OpenDrawing()  
    Dim dwgName As String  
    dwgName = "c:\Program Files\acad2000\sample\campus.dwg"  
    If Dir(dwgName) <> "" Then  
        ThisDrawing.Application.Documents.Open dwgName  
    Else  
        MsgBox "File " & dwgName & " does not exist."  
    End If  
End Sub
```

### 1.2. Tạo bản vẽ mới

Ví dụ này sử dụng phương thức Add để tạo một bản vẽ mới dựa trên khuôn thức mặc định.

```
Sub Ch3_NewDrawing()  
    Dim docObj As AcadDocument  
    Set docObj = ThisDrawing.Application.Documents.Add  
End Sub
```

### 1.3. Lưu bản vẽ

Có thể dùng phương thức Save hoặc Save As để lưu bản vẽ.

#### Lưu bản vẽ hiện hành

Ví dụ sau sẽ lưu bản vẽ hiện hành sử dụng tên tệp sẵn có đồng thời cũng lưu bản vẽ với một tên khác.

```
Sub Ch3_SaveActiveDrawing()  
    ' Lưu bản vẽ hiện hành sử dụng tên tệp sẵn có  
    ThisDrawing.Save  
    ' Lưu bản vẽ sử dụng tên khác  
    ThisDrawing.SaveAs "MyDrawing.dwg"  
End Sub
```

Thông thường, khi ta muốn kiểm tra xem bản vẽ hiện hành có lưu những thay đổi hay chưa trước khi thoát khỏi phiên làm việc của AutoCAD hoặc khi bắt đầu một bản vẽ mới, hãy sử dụng thuộc tính Saved để kiểm tra chắc chắn rằng bản vẽ đã lưu những thay đổi trước đó.

### Kiểm tra xem bản vẽ đã lưu hay chưa

Ví dụ sau sẽ kiểm tra xem bản vẽ đã được lưu hay chưa và sẽ hỏi người dùng xem có đồng ý để lưu bản vẽ hay không (Nếu không đồng ý, sẽ thoát khỏi chương trình). Nếu đồng ý, sẽ sử dụng phương thức Save để lưu bản vẽ hiện hành.

```
Sub Ch3_TestIfSaved()  
    If Not (ThisDrawing.Saved) Then  
        If MsgBox("Do you wish to save this drawing?", _  
            vbYesNo) = vbYes Then  
            ThisDrawing.Save  
        End If  
    End If  
End Sub
```

## 2. Thiết lập các lựa chọn trong AutoCAD

Có chín đối tượng gắn với các lựa chọn khác nhau, mỗi đối tượng tương ứng với một thẻ trong hộp thoại Options. Thông qua các đối tượng này, ta có thể truy cập được tất cả các dữ liệu về lựa chọn được lưu giữ trong hộp thoại Options. Do vậy, người lập trình có thể tùy biến các thiết lập trong AutoCAD thông qua các thuộc tính có trong những đối tượng đó. Chín đối tượng này bao gồm:

- PreferencesDisplay
- PreferencesDrafting
- PreferencesFiles
- PreferencesOpenSave
- PreferencesOutPut
- PreferencesProFile
- PreferencesSelection
- PreferencesSystem
- PreferencesUser

Tất cả các đối tượng trên đều có thể được truy xuất thông qua đối tượng Preferences. Để có được quyền truy cập đến đối tượng Preferences, ta sử dụng thuộc tính Preferences của đối tượng Application như sau:

### Truy xuất đối tượng Preference

```
Dim acadPref As AcadPreferences  
Set acadPref = ThisDrawing.Application.Preferences
```

Sau đó, người lập trình có thể truy xuất đến bất kỳ đối tượng Preferences nào sử dụng các thuộc tính Display, Drafting, Files, OpenSave, Output, Profile, Selection,

System và User. Ví dụ: người lập trình có thể điều chỉnh kích thước của dấu thập con trỏ với thuộc tính CursorSize.

### Thiết lập kích thước dấu thập con trỏ thành toàn màn hình

Ví dụ này thiết lập kích thước dấu thập con trỏ thành toàn màn hình

```
acadPref.Display.CursorSize=100
```

Người lập trình có thể muốn ứng dụng của mình kích hoạt hoặc vô hiệu hoá một số tính năng nào đó trong giao diện AutoCAD.

### Hiển thị screen menu<sup>1</sup> và các thanh cuộn

Ví dụ sau đây kích hoạt screen menu và vô hiệu hoá các thanh cuộn sử dụng thuộc tính DisplayScreenMenu và DisplayScrollBars

```
acadPref.Display.DisplayScreenMenu = True  
acadPref.Display.DisplayScrollBars = False
```

## 2.2. Lựa chọn về CSDL

Ngoài 9 đối tượng Preferences, đối tượng DatabasePreferences bao gồm những lựa chọn được lưu trong bản vẽ. Đối tượng tách biệt này được dùng để chứa các lựa chọn lưu cùng bản vẽ sử dụng cho các ứng dụng truy xuất vào trong bản vẽ AutoCAD mà không cần phải có trước chương trình AutoCAD (ứng dụng ObjectDBX<sup>TM 2</sup>).

Đối tượng DatabasePreferences nằm trong đối tượng Document.

## 3. Điều khiển cửa sổ ứng dụng

Khả năng điều khiển cửa sổ ứng dụng đem đến cho người phát triển ứng dụng tính linh hoạt để tạo ra những chương trình thật hiệu quả và thông minh. Sẽ có những lúc ứng dụng cần phải thu nhỏ cửa sổ AutoCAD, có lẽ là lúc chương trình thực hiện các thao tác trong các chương trình khác, chẳng hạn như Excel. Thêm nữa, ta cũng cần phải thường xuyên kiểm tra trạng thái của cửa sổ ứng dụng AutoCAD trước khi thực hiện các thao tác, chẳng hạn như nhắc nhập liệu từ người dùng.

Bằng cách sử dụng các phương thức và thuộc tính trong đối tượng Application, ta có thể thay đổi vị trí, kích thước và tính nhìn thấy của cửa sổ ứng dụng. Ngoài ra ta cũng có thể thu nhỏ hoặc phóng to cửa sổ ứng dụng, kiểm tra trạng thái hiện hành của cửa sổ.

---

<sup>1</sup> **Screen menu:** là một dạng trình đơn được tổ chức theo phương thẳng đứng của màn hình (thường ở bên phải) để thực hiện các chức năng hỗ trợ cho lệnh đang thực thi của AutoCAD.

<sup>2</sup> **ObjectDBX<sup>TM</sup>**, hiện nay đã được đổi tên thành RealDWG, là thư viện phần mềm cho phép người phát triển C++ và .NET thực hiện các thao tác đọc và ghi tệp AutoCAD DWG và DXF mà không cần phải cài đặt chương trình AutoCAD.

### 3.1. Thay đổi vị trí và kích thước của cửa sổ ứng dụng

Người lập trình còn có thể sử dụng đối tượng Application để điều chỉnh vị trí và kích thước của cửa sổ ứng dụng AutoCAD.

#### Định vị trí cửa sổ ứng dụng

Ví dụ dưới đây sử dụng thuộc tính WindowTop, WindowLeft, Width, và Height để định vị trí của cửa sổ ứng dụng AutoCAD ở góc trên bên trái của màn hình và định kích thước của cửa sổ là 400 điểm theo chiều rộng và 400 điểm theo chiều cao.

```
Sub Ch3_PositionApplicationWindow()  
    ThisDrawing.Application.WindowTop = 0  
    ThisDrawing.Application.WindowLeft = 0  
    ThisDrawing.Application.Width = 400  
    ThisDrawing.Application.Height = 400  
End Sub
```

### 3.2. Thu phóng cửa sổ ứng dụng AutoCAD

Cửa sổ AutoCAD có thể được thu nhỏ hoặc phóng to bằng cách sử dụng thuộc tính WindowState. Những ví dụ sau minh họa điều này.

#### Phóng to cửa sổ ứng dụng

```
ThisDrawing.Application.WindowState = acMax
```

#### Thu nhỏ cửa sổ ứng dụng

```
ThisDrawing.Application.WindowState = acMin
```

### 3.3. Xác định trạng thái hiện hành của cửa sổ AutoCAD

Trạng thái hiện hành của cửa sổ AutoCAD có thể xác định được bằng cách sử dụng thuộc tính WindowState.

#### Xác định trạng thái hiện hành của cửa sổ ứng dụng

Ví dụ sau sẽ truy vấn trạng thái của cửa sổ ứng dụng và hiển thị trạng thái cửa sổ trong một hộp thông báo.

```
Sub Ch3_CurrentWindowState()  
    Dim CurrWindowState As Integer  
    Dim msg As String  
    CurrWindowState = ThisDrawing.Application.WindowState  
    msg = Choose _  
        (CurrWindowState, "Bình thường", "Thu nhỏ", "Phóng to")  
    MsgBox "Trạng thái hiện hành của cửa sổ là: " + msg  
End Sub
```

### 3.4. Ẩn cửa sổ ứng dụng

Có thể ẩn cửa sổ ứng dụng khỏi màn hình của người dùng.

#### Ẩn cửa sổ ứng dụng

Đoạn mã sau sử dụng thuộc tính Visible để ẩn cửa sổ ứng dụng:

```
ThisDrawing.Application.Visible = False
```



## 4. Điều khiển cửa sổ bản vẽ

Cũng giống như cửa sổ ứng dụng AutoCAD, ta có thể thu nhỏ, phóng to, định lại vị trí, thay đổi kích thước và kiểm tra trạng thái của bất kỳ một cửa sổ bản vẽ nào. Tuy nhiên, ta còn có thể thay đổi cách thức thể hiện bản vẽ trong cửa sổ bằng cách sử dụng các phương thức về cảnh nhìn (views), khung nhìn (viewports) và thu phóng (zooming).

AutoCAD ActiveX cung cấp nhiều cách thức thể hiện bản vẽ. Ta có thể điều khiển sự hiển thị của bản vẽ để di chuyển nhanh đến những vùng khác nhau của bản vẽ trong khi có thể theo dõi toàn bộ hiệu ứng của các thay đổi. Ngoài ra cũng có thể thay đổi độ phóng đại của bản vẽ hoặc trượt bản vẽ để định lại vị trí quan sát trong vùng đồ họa, lưu cảnh nhìn và khôi phục lại khi cần để in hoặc tham khảo đến một chi tiết cụ thể nào đó, hoặc thể hiện vài cảnh nhìn cùng một lúc bằng cách phân màn hình thành vài khung nhìn xếp cạnh nhau.

### 4.1. Thay đổi vị trí và kích thước của cửa sổ bản vẽ

Sử dụng đối tượng Document để điều chỉnh vị trí và kích thước của các cửa sổ bản vẽ.

#### Định vị trí của cửa sổ bản vẽ

Ví dụ sau sử dụng thuộc tính Width và Height để thiết lập kích thước cửa sổ bản vẽ hiện hành với bề rộng 400 điểm và chiều cao 400 điểm.

```
ThisDrawing.Width = 400  
ThisDrawing.Height = 400
```

### 4.2. Thu phóng cửa sổ bản vẽ

Cửa sổ bản vẽ có thể được thu nhỏ hoặc phóng to sử dụng thuộc tính WindowState.

#### Phóng to cửa sổ bản vẽ hiện hành

```
ThisDrawing.WindowState = acMax
```

#### Thu nhỏ cửa sổ bản vẽ hiện hành

```
ThisDrawing.WindowState = acMin
```

### 4.3. Xác định trạng thái hiện hành của cửa sổ bản vẽ

Có thể xác định cửa sổ hiện hành của cửa sổ bản vẽ bằng cách sử dụng thuộc tính WindowState.

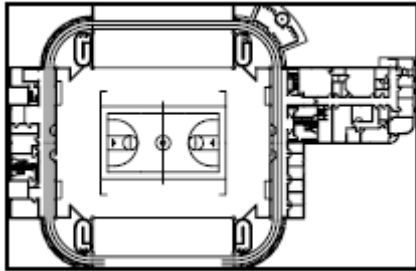
#### Xác định trạng thái hiện hành của cửa sổ bản vẽ hiện hành

```
Sub Ch3_CurrentWindowState()  
    Dim CurrWindowState As Integer  
    Dim msg As String  
    CurrWindowState = ThisDrawing.WindowState  
    msg = Choose(CurrWindowState, "Bình thường", "Thu nhỏ", "Phóng to")  
    MsgBox "Trạng thái hiện hành của cửa sổ bản vẽ là " + msg  
End Sub
```

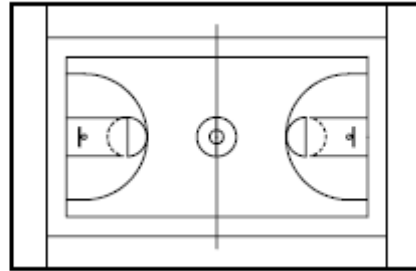
## 4.4. Sử dụng chức năng thu phóng

Cảnh nhìn là sự quan sát bản vẽ với độ phóng đại, vị trí và hướng nhất định. Cách phổ biến nhất để thay đổi cảnh nhìn là sử dụng một trong rất nhiều lựa chọn trong lệnh Zoom để tăng hay giảm kích thước hình ảnh hiển thị trong vùng đồ họa.

Phóng hình ảnh ra để quan sát chi tiết gọi là phóng to. Còn thu hình ảnh lại để quan sát tổng quát hơn gọi là thu nhỏ.



Thu nhỏ (Zoom out)

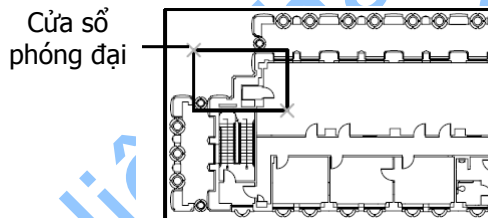


Phóng to (Zoom in)

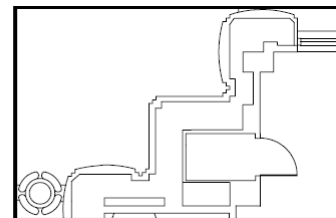
Phóng to không thay đổi kích thước tuyệt đối của bản vẽ, chỉ thay đổi kích thước thể hiện trong vùng đồ họa. AutoCAD đưa ra nhiều cách để thay đổi cảnh nhìn, bao gồm xác định vùng hiển thị, phóng đại đến một tỷ lệ nhất định và thể hiện toàn bộ bản vẽ.

### 4.4.1. Xác định cửa sổ phóng đại

Người lập trình có thể nhanh chóng phóng đại một vùng trên bản vẽ bằng cách xác định góc định vị cho vùng đó.



Cảnh nhìn ban đầu



Cảnh nhìn mới

Vùng được xác định bằng góc định vị đã lựa chọn sẽ được canh vào giữa vùng đồ họa nếu vùng phóng đại không có kích thước tỷ lệ với khung nhìn hiện hành.

Để thực hiện phóng đại một vùng theo đường bao cho trước, ta có thể sử dụng một trong hai phương thức ZoomWindow hoặc ZoomPickWindow. Phương thức ZoomWindow cho phép lập trình để định nghĩa hai điểm để làm cửa sổ bao. Còn phương thức ZoomPickWindow lại yêu cầu người dùng phải chọn hai điểm trên màn hình và hai điểm này sẽ trở thành điểm để làm cửa sổ bao.

### Phóng đại bản vẽ hiện hành theo một cửa sổ bao xác định bởi hai điểm

```
Sub Ch3_ZoomWindow()  
MsgBox "Dùng phương thức ZoomWindow với:" & vbCrLf & _  
    "1.3, 7.8, 0" & vbCrLf & _  
    "13.7, -2.6, 0", , "ZoomWindow"  
Dim point1(0 To 2) As Double  
Dim point2(0 To 2) As Double  
point1(0) = 1.3: point1(1) = 7.8: point1(2) = 0
```

```

point2(0) = 13.7: point2(1) = -2.6: point2(2) = 0
ThisDrawing.Application.ZoomWindow point1, point2
MsgBox " Dùng phương thức ZoomPickWindow", , "ZoomPickWindow"
ThisDrawing.Application.ZoomPickWindow
End Sub

```

#### 4.4.2. Tỷ lệ của cảnh nhìn

Nếu cần tăng hoặc giảm mức phóng đại của hình ảnh theo một tỷ lệ chính xác, ta có thể thực hiện theo 3 cách:

- Tương đối so với vùng giới hạn vẽ
- Tương đối so với cảnh nhìn hiện hành
- Tương đối so với đơn vị trang in

Khi thay đổi tỷ lệ cảnh nhìn tương đối so với vùng giới hạn vẽ, chỉ cần nhập giá trị bằng 1 để hiện tất cả giới hạn của vùng vẽ trong vùng đồ họa và tất cả sẽ được canh giữa theo điểm giữa của cảnh nhìn trước đó. Để phóng to hay thu nhỏ, chỉ cần nhập vào số lớn hơn hoặc nhỏ hơn 1. Ví dụ, nhập vào số 2 sẽ cho cảnh nhìn lớn gấp 2 lần so với khi quan sát toàn bộ bản vẽ hoặc 0.5 khi cảnh nhìn chỉ bằng một nửa so với khi quan sát toàn bộ bản vẽ.

Khi thay đổi tỷ lệ cảnh nhìn tương đối so với cảnh nhìn hiện hành, chỉ cần nhập vào số 2 để nhân đôi hay 0.5 để hiện chỉ một nửa kích thước so với cảnh nhìn hiện hành. Và đương nhiên nhập vào số 1 thì sẽ không có hiệu ứng gì.

Khi thay đổi tỷ lệ cảnh nhìn tương đối so với đơn vị trang in, giá trị tỷ lệ nhập vào sẽ có thể tăng hoặc giảm cảnh nhìn tương đối so với tỷ lệ hiện tại trong không gian in và được sử dụng để thay đổi tỷ lệ của khung nhìn trước khi in.

##### 4.4.2.1. Tạo cảnh nhìn theo tỷ lệ

Để tạo cảnh nhìn theo tỷ lệ, phải sử dụng phương thức ZoomScaled. Phương thức này cần có hai thông số đầu vào: tỷ lệ và loại tỷ lệ. Tỷ lệ: đơn giản chỉ là một con số. Việc xử lý con số này trong AutoCAD phụ thuộc vào loại tỷ lệ được lựa chọn.

Loại tỷ lệ sẽ xác định cách thức cần thể hiện theo tỷ lệ, bao gồm: tương đối so với vùng giới hạn vẽ, so với cảnh nhìn hiện hành, hay so với đơn vị trang in. Để thay đổi tỷ lệ tương đối so với vùng giới hạn vẽ, ta sử dụng hằng số acZoomScaledAbsolute. Để thay đổi tỷ lệ so với cảnh nhìn hiện hành, sử dụng hằng số acZoomScaledRelative. Để thay đổi tỷ lệ so với đơn vị trang in, sử dụng hằng số acZoomScaledRelativePSpace.

#### Phóng đại bản vẽ hiện hành với tỷ lệ cho trước

```

Sub Ch3_ZoomScaled()
    MsgBox "Perform a ZoomScaled using:" & vbCrLf & _
        "Scale Type: acZoomScaledRelative" & vbCrLf & _
        "Scale Factor: 2", , "ZoomScaled"

    Dim scalefactor As Double
    Dim scaletype As Integer

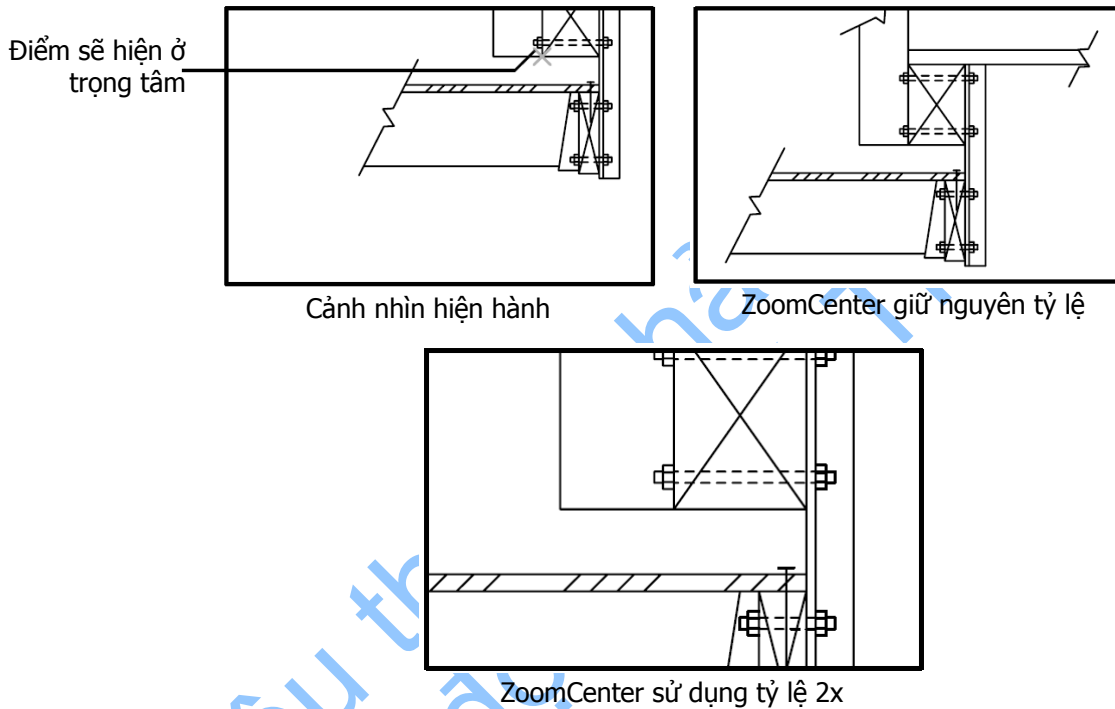
    scalefactor = 2
    scaletype = acZoomScaledRelative

```

```
ThisDrawing.Application.ZoomScaled scalefactor, scaletype
End Sub
```

### 4.4.3. Căn giữa

Ta có thể di chuyển một điểm nào đó trong bản vẽ vào giữa vùng đồ họa. Phương thức `ZoomCenter` rất hữu ích trong việc thay đổi tỷ lệ hiển thị của đối tượng và di chuyển đối tượng đó vào giữa khung nhìn. Ví dụ sau sẽ làm rõ cách sử dụng lệnh `ZoomCenter` để hiển thị đối tượng mà vẫn giữ nguyên tỷ lệ và ở mức phóng đại gấp hai lần:



Bằng cách sử dụng `ZoomCenter`, ta có thể xác định tỷ lệ bằng cách nhập vào giá trị phóng đại so với cảnh nhìn hiện hành.

#### Phóng đại bản vẽ hiện tại với chế độ căn giữa

```
Sub Ch3_ZoomCenter()
    MsgBox "Perform a ZoomCenter using:" & vbCrLf & _
        "Center 3, 3, 0" & vbCrLf & _
        "Magnification: 10", , "ZoomCenter"

    Dim Center(0 To 2) As Double
    Dim magnification As Double
    Center(0) = 3: Center(1) = 3: Center(2) = 0
    magnification = 10

    ThisDrawing.Application.ZoomCenter Center, magnification
End Sub
```

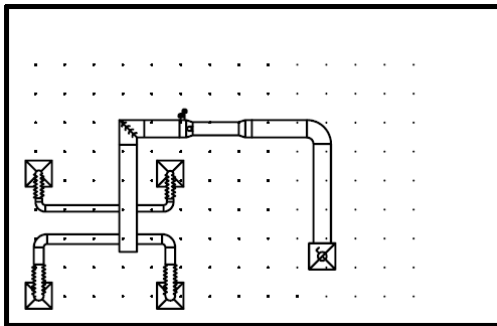
#### 4.4.3.2. Hiển thị vùng giới hạn vẽ và vùng đối tượng

Để hiển thị vùng quan sát dựa trên biên của bản vẽ hoặc vùng đối tượng trong bản vẽ, ta sử dụng phương thức `ZoomAll`, `ZoomExtents`, hoặc `ZoomPrevious`.

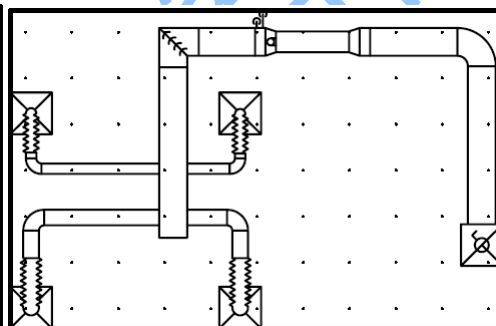
Phương thức ZoomAll sẽ hiển thị toàn bộ bản vẽ. Nếu có đối tượng vẽ ở ngoài vùng giới hạn vẽ, phương thức ZoomAll sẽ hiển thị toàn bộ các đối tượng. Nếu các đối tượng được vẽ bên trong vùng giới hạn vẽ, phương thức ZoomAll sẽ hiển thị toàn bộ vùng giới hạn vẽ.

Phương thức ZoomExtents xác định tỷ lệ phóng đại dựa trên vùng đối tượng của khung nhìn hiện hành (không phải là cảnh nhìn hiện hành). Thông thường, toàn bộ khung nhìn hiện hành đều có thể nhìn thấy được, và do đó, kết quả là rất rõ ràng, trực quan. Tuy nhiên, khi sử dụng phương thức Zoom trong không gian mô hình trong khi đang làm việc trong khung nhìn của không gian in, nếu phóng to bên ngoài biên của khung nhìn trong không gian in thì có thể không nhìn thấy được một số khu vực đã phóng đại.

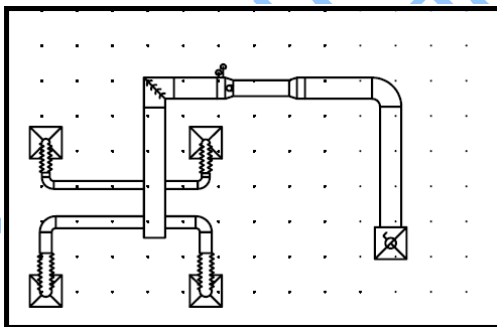
Phương thức ZoomExtents sẽ thay đổi cảnh nhìn để có thể bao quát tất cả các đối tượng của bản vẽ hiện hành. Trong một vài trường hợp (đối với cả phương thức ZoomAll và phương thức ZoomExtents), điều này có thể sẽ kích hoạt quá trình tái tạo bản vẽ<sup>1</sup>. Quá trình tái tạo bản vẽ không được kích hoạt với các lớp đóng băng hoặc lớp đã bị tắt. Nếu bản vẽ không có đối tượng nào cả, phương thức ZoomExtents sẽ hiển thị vùng giới hạn vẽ.



Cảnh nhìn hiện tại



Phóng đại vùng đối tượng



Phóng đại để thể hiện toàn bộ bản vẽ

<sup>1</sup> **Regeneration:** quá trình tái tạo bản vẽ. Mục đích của quá trình tái tạo bản vẽ nhằm: phát sinh lại toàn bộ dữ liệu của bản vẽ, tính toán lại hệ tọa độ màn hình, tạo lại chỉ mục CSDL bản vẽ để tăng tốc độ hiển thị (tạo lại trình tự hiển thị)

Trong cảnh nhìn 3D, phương thức ZoomAll và ZoomExtents có tác dụng như nhau. Đường tạm<sup>1</sup> (xline và ray) không ảnh hưởng đến quá trình thực hiện của phương thức ZoomAll và ZoomExtents.

Lệnh ZoomPrevious sẽ phóng đại khung nhìn hiện tại theo như lần phóng đại trước đó.

### Phóng đại bản vẽ hiện tại sử dụng lệnh ZoomAll và ZoomExtents

```
Sub Ch3_ZoomAll()  
    MsgBox "Perform a ZoomAll", , "ZoomAll" 'ZoomAll  
    ThisDrawing.Application.ZoomAll 'ZoomExtents  
    MsgBox "Perform a ZoomExtents", , "ZoomExtents"  
    ThisDrawing.Application.ZoomExtents  
End Sub
```

## 4.5. Sử dụng các cảnh nhìn đã được đặt tên

Ta có thể đặt tên và lưu lại một cảnh nhìn nào đó để dùng lại sau này. Và khi không cần nữa, ta có thể xoá cảnh nhìn đó đi.

### 4.5.1. Tạo và đặt tên các cảnh nhìn

Các cảnh nhìn được đặt tên ngay từ lúc tạo ra chúng. Để tạo một cảnh nhìn mới, sử dụng phương thức Add để thêm một cảnh nhìn mới vào tập đối tượng Views.

Tên của mỗi cảnh nhìn có thể dài đến 255 ký tự và có thể chứa chữ, số, và một số ký tự đặc biệt như dấu dollar (\$), dấu gạch ngang (-), dấu gạch dưới (\_).

Vị trí và tỷ lệ của cảnh nhìn cũng được lưu trữ cùng với bản vẽ.

#### Thêm một cảnh nhìn mới

```
Sub Ch3_AddView()  
    ' Thêm một cảnh nhìn mới vào Tập đối tượng Views  
    Dim viewObj As AcadView  
    Set viewObj = ThisDrawing.Views.Add("View1")  
End Sub
```

### 4.5.2. Xoá các cảnh nhìn

Để xoá các phương thức đã được đặt tên, ta sử dụng phương thức Delete. Phương thức Delete nằm ngay bên trong các đối tượng View chứ không phải ở các đối tượng cha. Ví dụ sau sẽ xoá một cảnh nhìn từ chính nó và từ tập đối tượng Views bằng cách chỉ ra tên cảnh nhìn cần xoá.

#### Xoá một cảnh nhìn từ bản thân đối tượng View

```
ViewObj.Delete
```

#### Xoá cảnh nhìn từ tập đối tượng Views

```
ThisDrawing.Views("View1").Delete
```

---

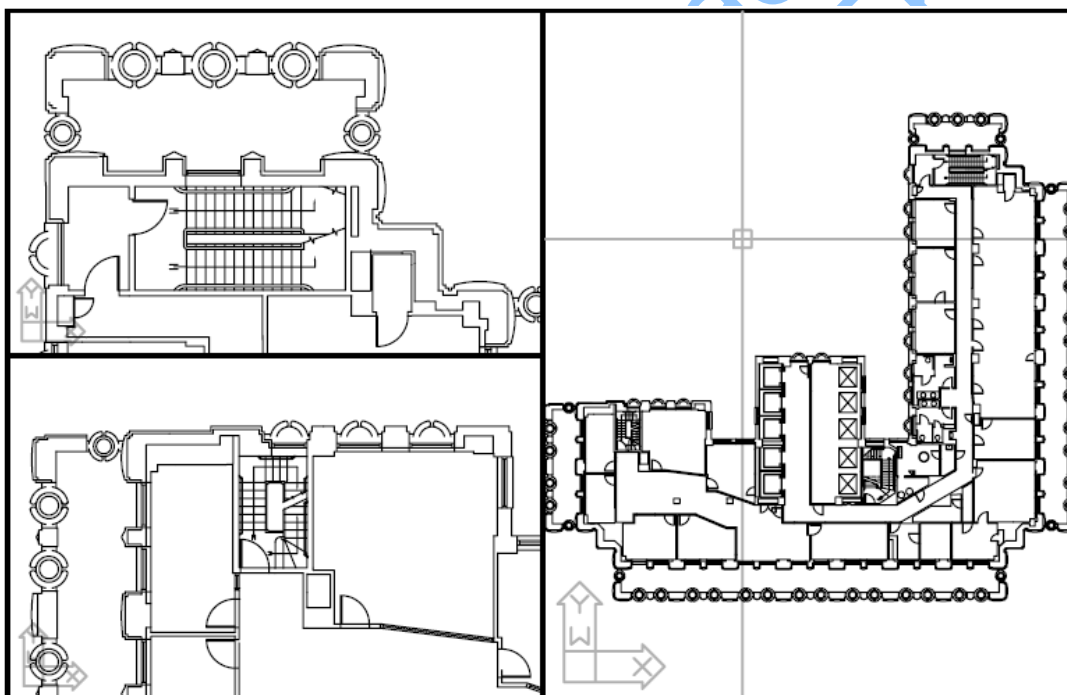
<sup>1</sup> **Đường tạm (Construction line):** là một loại đối tượng của AutoCAD và thường được dùng để giúp vẽ các đối tượng khác được chính xác.

## 4.6. Sử dụng các khung nhìn xếp cạnh nhau

AutoCAD thường khởi tạo một bản vẽ mới trong một khung nhìn có kích thước không giới hạn trên toàn vùng đồ họa. Ta có thể phân vùng đồ họa thành nhiều khung nhìn khác nhau cùng một lúc. Ví dụ như, khi ta tạo ra vùng quan sát ở mức độ chi tiết và tổng quan một chi tiết nào đó thì ta có thể theo dõi được sự thay đổi khi hiệu chỉnh một đối tượng nào đó trong bản vẽ. Trong từng khung nhìn xếp cạnh nhau, ta có thể thực hiện được các thao tác sau:

- Phóng đại, thiết lập chế độ bắt lưới, hiển thị lưới và hiển thị biểu tượng hệ toạ độ người dùng (UCS), và hiển thị lại các cảnh nhìn đã được đặt tên trong từng khung nhìn riêng biệt.
- Thực hiện lệnh vẽ từ khung nhìn này sang khung nhìn khác
- Đặt tên cấu hình cho khung nhìn để có thể sử dụng lại sau này

Ví dụ sau minh họa một bản vẽ với 3 khung nhìn xếp cạnh nhau. Dấu trỏ vẽ nằm ở khung nhìn hiện hành. Các khung nhìn lấp đầy toàn bộ vùng đồ họa và không đè lên nhau.



Khi vẽ trên một khung nhìn thì các thay đổi sẽ được hiển thị ngay lập tức trong các khung nhìn khác.

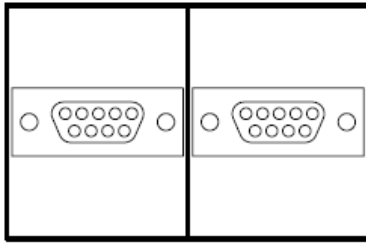
Khung nhìn xếp cạnh nhau khác với các khung nhìn được sắp xếp trong không gian in. Các khung nhìn trong không gian in, còn được gọi là khung nhìn động, thường được sử dụng để tạo ra bố trí hợp lý cho bản vẽ. Các khung nhìn động có thể đè lên nhau và được in ra cùng một lúc.

### 4.6.1. Hiển thị nhiều khung nhìn xếp cạnh nhau

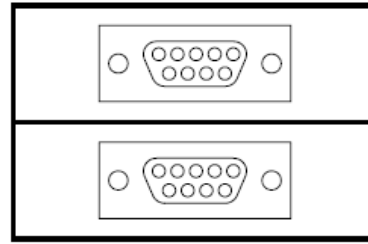
Ta có thể hiển thị khung nhìn xếp cạnh nhau theo nhiều kiểu cấu hình khác nhau. Việc hiển thị các khung nhìn như thế nào phụ thuộc vào số lượng và kích thước của



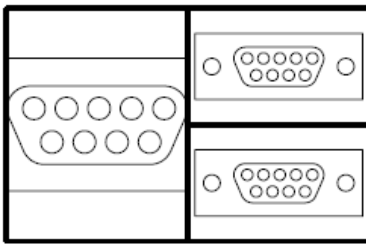
các cảnh nhìn mà ta cần. Minh hoạ sau thể hiện các kiểu cấu hình các khung nhìn mặc định:



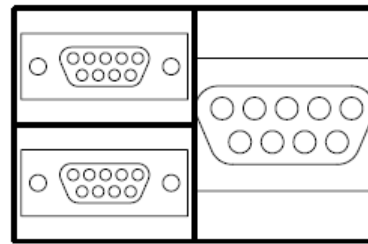
2, xếp đứng



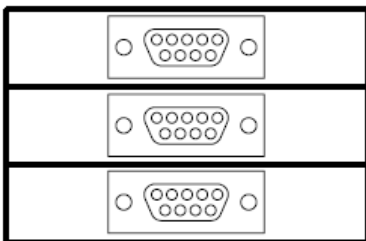
2, xếp ngang



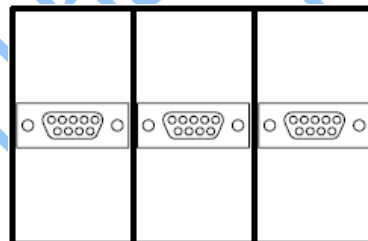
3, xếp trái



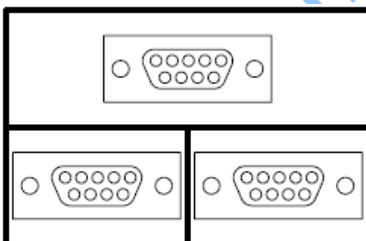
3, xếp phải



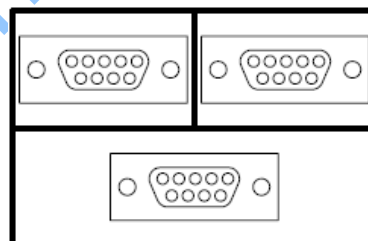
3, xếp ngang



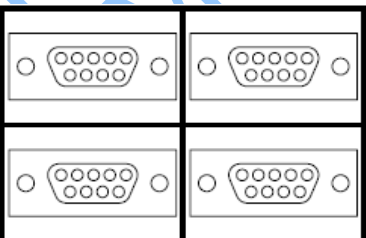
3, xếp đứng



3, xếp trên



3, xếp dưới



4

Để phân chia khung nhìn hiện hành, sử dụng phương thức Split. Phương thức này cần có 1 tham số: kiểu cấu hình của mà khung nhìn sẽ được phân chia. Để xác lập các kiểu cấu hình, ta sử dụng các hằng số sau, tương ứng với các kiểu cấu hình mặc định như đã được minh hoạ ở trên: acViewport2Horizontal, acViewport2Vertical,

acViewport3Left, acViewport3Right, acViewport3Horizontal,  
acViewport3Vertical, acViewport3Above, acViewport3Below, hoặc acViewport4.

Ví dụ sau sẽ tạo ra một khung nhìn mới và sau đó phân chia khung nhìn thành hai cửa sổ xếp ngang.

#### Phân chia khung nhìn thành 2 cửa sổ ngang

```
Sub SplitViewport()  
    ' Tạo khung nhìn mới  
    Dim vportObj As AcadViewport  
    Set vportObj = ThisDrawing.Viewports.Add("TEST_VIEWPORT")  
  
    ' Phân chia khung nhìn thành 2 cửa sổ ngang  
    vportObj.Split acViewport2Horizontal  
  
    ' Thiết lập vportObj thành khung nhìn hiện hành  
    ThisDrawing.ActiveViewport = vportObj  
End Sub
```

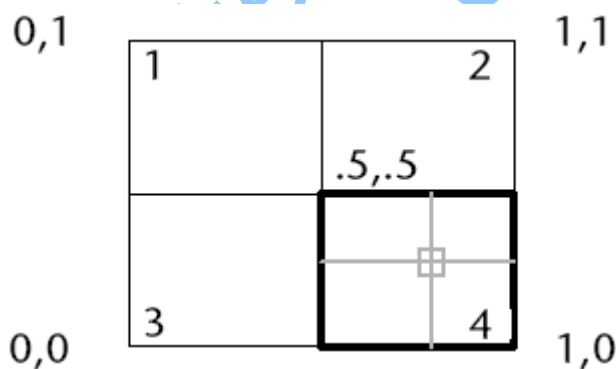
#### 4.6.2. Chuyển khung nhìn hiện hành

Khi khung nhìn là hiện hành, con trỏ chuột sẽ chuyển thành dạng trỏ vẽ và biên của khung nhìn sẽ được làm nổi bật.

Để thiết lập một khung nhìn là hiện hành, ta sử dụng thuộc tính ActiveViewport.

Ta có thể sử dụng lệnh duyệt qua các khung nhìn hiện hành để tìm ra một khung nhìn nào đó. Để làm được việc này, trước hết phải xác định tên cấu hình khung nhìn có chứa khung nhìn cần tìm sử dụng thuộc tính Name. Ngoài ra, nếu cấu hình khung nhìn đã được phân chia, mỗi khung nhìn con trên cấu hình khung nhìn sẽ được xác định thông qua thuộc tính LowerLeftCorner và UpperRightCorner.

Thuộc tính LowerLeftCorner và UpperRightCorner được thể hiện dựa trên vị trí của khung nhìn trên màn hình đồ họa. Các thuộc tính này được định nghĩa như sau (lấy cách phân chia thành 4 khung nhìn làm ví dụ):



Khung nhìn 1: LowerLeftCorner = (0, .5), UpperRightCorner = (.5, 1)

Khung nhìn 2: LowerLeftCorner = (.5, .5), UpperRightCorner = (1, 1)

Khung nhìn 3: LowerLeftCorner = (0, 0), UpperRightCorner = (.5, .5)

Khung nhìn 4: LowerLeftCorner = (.5, 0), UpperRightCorner = (1, .5)

## Phân chia khung nhìn và duyệt qua các cửa sổ

Ví dụ sau đây sẽ phân chia khung nhìn thành 4 cửa sổ, sau đó sẽ duyệt qua các khung nhìn trong bản vẽ và sẽ hiển thị tên khung nhìn, cùng với góc dưới-trái và góc trên-phải của các khung nhìn.

```
Sub Ch3_IteratingViewportWindows()  
    ' Tạo một khung nhìn mới và đặt nó là hiện hành  
    Dim vportObj As AcadViewport  
    Set vportObj = ThisDrawing.Viewports.Add("TEST_VIEWPORT")  
    ThisDrawing.ActiveViewport = vportObj  
  
    ' Chia khung nhìn thành 4 cửa sổ  
    vportObj.Split acViewport4  
  
    ' Duyệt qua các khung nhìn,  
    ' Làm nổi bật từng khung nhìn và hiển thị  
    ' tọa độ góc trên bên phải và góc dưới bên trái  
    ' của từng khung nhìn  
    Dim vport As AcadViewport  
    Dim LLCorner As Variant  
    Dim URCorner As Variant  
    For Each vport In ThisDrawing.Viewports  
        ThisDrawing.ActiveViewport = vport  
        LLCorner = vport.LowerLeftCorner  
        URCorner = vport.UpperRightCorner  
        MsgBox "Viewport: " & vport.Name & " is now active." & _  
            vbCrLf & "Lower left corner: " & _  
            LLCorner(0) & ", " & LLCorner(1) & vbCrLf & _  
            "Upper right corner: " & _  
            URCorner(0) & ", " & URCorner(1)  
    Next vport  
End Sub
```

## 4.7. Cập nhật đặc tính hình học trong cửa sổ bản vẽ

Rất nhiều thao tác mà ta thực hiện thông qua AutoCAD ActiveX Automation làm thay đổi những gì được hiển thị trong bản vẽ AutoCAD. Không phải tất cả những thao tác đều được cập nhật ngay lập tức lên bản vẽ. Điều này giúp cho ta có thể thực hiện một vài thay đổi trong bản vẽ mà không cần phải chờ đợi để chương trình cập nhật lại kết quả hiển thị sau mỗi lần thay đổi. Thay vào đó, ta có thể gói tất cả các thao tác lại và sau đó gọi một lệnh duy nhất để cập nhật kết quả hiển thị.

Phương thức sẽ cập nhật kết quả hiển thị là phương thức Update và Regen.

Phương thức Update cập nhật kết quả hiển thị của một đối tượng đơn lẻ. Phương thức Regen sẽ tái tạo lại bản vẽ, tính toán lại hệ tọa độ màn hình và độ phân giải hiển thị của tất cả các đối tượng. Ngoài ra, phương thức này còn tạo lại chỉ mục cho CSDL của bản vẽ để tăng tốc độ hiển thị và chọn đối tượng.

### Cập nhật kết quả hiển thị của một đối tượng

Ví dụ sau sẽ tạo ra một vòng tròn, gán cho vòng tròn màu đỏ, và sau đó cập nhật lại vòng tròn đó sử dụng phương thức Update để cho vòng tròn có thể được nhìn thấy trong AutoCAD.

```
Sub Ch3_UpdateDisplay()  
    Dim circleObj As AcadCircle
```

```

Dim center(0 To 2) As Double
Dim radius As Double
center(0) = 1: center(1) = 1: center(2) = 0
radius = 1

' Tạo vòng tròn và gán màu đỏ
Set circleObj = ThisDrawing.ModelSpace.AddCircle(center,
radius)
circleObj.Color = acRed

' Cập nhật
circleObj.Update
End Sub

```

## 5. Thiết lập lại các đối tượng hiện hành

Những thay đổi đến hầu hết các đối tượng hiện hành, chẳng hạn như lớp hiện hành, kiểu đường thẳng hiện hành, sẽ được hiển thị ngay lập tức. Tuy nhiên, cũng có một vài đối tượng mà khi ta muốn hiển thị những thay đổi thì nhất thiết phải thiết lập lại đối tượng hiện hành. Những đối tượng loại này bao gồm kiểu chữ hiện hành, hệ tọa độ người dùng hiện hành và khung nhìn hiện hành. Nếu có thực hiện thay đổi trên các đối tượng này thì nhất thiết đối tượng kiểu này phải được thiết lập lại, và sau đó gọi phương thức Regen để làm cho các thay đổi có hiệu lực.

Để thiết lập lại các đối tượng này, chỉ đơn giản là thiết lập lại thuộc tính ActiveTextStyle, ActiveUCS, ActiveViewport của các đối tượng đã được thay đổi.

### Thiết lập lại khung nhìn hiện hành

Ví dụ sau sẽ thay đổi kết quả hiển thị lưới trong khung nhìn hiện hành và sau đó thiết lập lại khung nhìn đó là khung nhìn hiện hành để hiển thị sự thay đổi.

```

Sub Ch3_ResetActiveViewport ()
' Thiết lập hiển thị lưới cho khung nhìn hiện hành
ThisDrawing.ActiveViewport.GridOn = _
Not (ThisDrawing.ActiveViewport.GridOn)
' Thiết lập lại khung nhìn hiện hành
ThisDrawing.ActiveViewport = ThisDrawing.ActiveViewport
End Sub

```

## 6. Gán và lấy biến hệ thống

Đối tượng Document cung cấp phương thức SetVariable và GetVariable để gán và lấy giá trị biến hệ thống của AutoCAD.

### Gán biến hệ thống MIRRTEXT

Ví dụ sau sử dụng phương thức SetVariable để gán một số nguyên cho biến hệ thống MIRRTEXT, là biến dùng để thiết lập cách thức AutoCAD xử lý ký tự khi sử dụng lệnh Mirror.

```

ThisDrawing.SetVariable "MIRRTEXT", 0

```

## 7. Vẽ với độ chính xác cao

Với AutoCAD, ta có thể vẽ với độ chính xác hình học rất cao mà không cần phải thực hiện những phép tính đơn điệu. Thông thường, ta có thể xác định chính xác điểm mà chẳng cần phải biết tọa độ. Không cần phải rời màn hình bản vẽ, ta cũng có thể thực hiện tính toán cho bản vẽ và hiển thị rất nhiều loại thông tin trạng thái khác nhau.

Đến thời điểm này, AutoCAD ActiveX Automation chưa cung cấp các phương thức để thực hiện những tính năng sau trong AutoCAD:

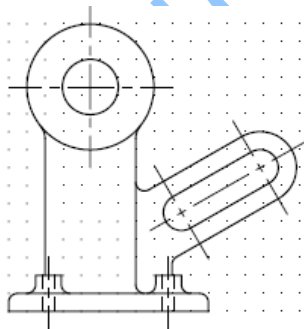
- Thiết lập lưới và bắt điểm đều.
- Thiết lập bắt điểm theo đối tượng
- Xác định bước đo trên đối tượng hoặc phân chia đối tượng thành nhiều đoạn.

### 7.1. Điều chỉnh bắt điểm và lưới

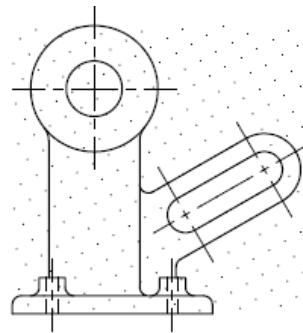
Ta có thể sử dụng hệ thống lưới làm đường cơ sở và bật chế độ bắt điểm để giới hạn di chuyển của con trỏ chuột. Bên cạnh việc thay đổi khoảng cách, ta có thể thay đổi sự canh hàng của lưới và chế độ bắt điểm. Ta còn có thể thay đổi sự canh hàng, hoặc ta còn có thể thiết lập để sử dụng trong các bản vẽ đồng dạng.

#### 7.1.1. Thay đổi góc bắt điểm và điểm cơ sở

Nếu ta cần vẽ dọc theo một hướng hoặc một góc nào đó, ta có thể quay góc bắt điểm. Phép quay này sẽ ràng buộc con trỏ theo hướng mới trong khi chế độ bắt điểm và bắt vuông góc đang bật. Trong ví dụ sau, góc bắt điểm được điều chỉnh để phù hợp với góc của móc neo. Với cách điều chỉnh như thế này, ta có thể sử dụng hệ thống lưới để vẽ các đối tượng ở góc 30 độ.



Góc bắt điểm mặc định



Góc bắt điểm đã quay

Điểm tâm của góc quay bắt điểm gọi là điểm cơ sở. Nếu ta cần quay đường gióng của các mẫu tô bóng, ta có thể thay đổi điểm này, nhưng thông thường có tọa độ là 0,0.

Để quay góc bắt điểm, ta sử dụng thuộc tính `SnapRotationAngle`. Để thay đổi điểm cơ sở của góc quay bắt điểm, sử dụng thuộc tính `SnapBasePoint`.

---

**CHÚ Ý** Cả hai thuộc tính đều cần phải gọi phương thức `Update` để cập nhật hiển thị trong AutoCAD

---

## Thay đổi điểm cơ sở và góc quay bắt điểm

Ví dụ sau sẽ thay đổi điểm cơ sở thành (1,1) và góc bắt điểm thành 30 độ. Hệ thống lưới được bật lên để ta có thể nhìn thấy được những thay đổi trên.

```
Sub Ch3_ChangeSnapBasePoint ()
    ' Bật hệ thống lưới trong khung nhìn hiện hành
    ThisDrawing.ActiveViewport.GridOn = True

    ' Thay đổi điểm cơ sở thành 1, 1
    Dim newBasePoint(0 To 1) As Double
    newBasePoint(0) = 1: newBasePoint(1) = 1
    ThisDrawing.ActiveViewport.SnapBasePoint = newBasePoint

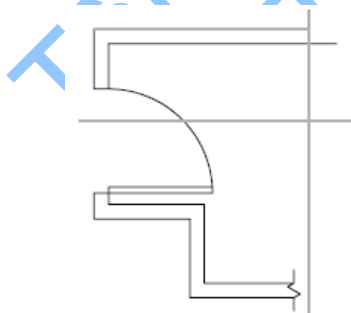
    ' Thay đổi góc bắt điểm thành 30 độ (0.575 radians)
    Dim rotationAngle As Double
    rotationAngle = 0.575
    ThisDrawing.ActiveViewport.SnapRotationAngle = rotationAngle

    ' Thiết lập lại khung nhìn hiện hành
    ThisDrawing.ActiveViewport = ThisDrawing.ActiveViewport
End Sub
```

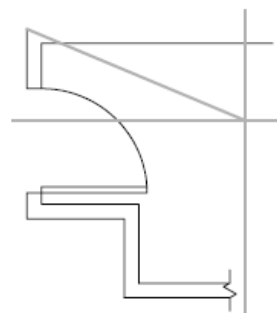
## 7.2. Sử dụng chế độ bắt vuông góc

Khi vẽ đường thẳng hoặc di chuyển một đối tượng, ta sử dụng chế độ bắt vuông góc để giới hạn con trỏ di chuyển theo trục đứng và trục nằm ngang (chế độ bắt vuông góc này phụ thuộc vào góc bắt điểm hiện hành hoặc phụ thuộc vào hệ tọa độ người dùng). Chế độ bắt vuông góc làm việc với tất cả các thao tác mà yêu cầu người dùng cần phải nhập vào điểm thứ hai. Ta không chỉ sử dụng chế độ bắt vuông góc để canh theo phương đứng và phương ngang mà còn tạo đường song song hoặc đồng dạng thông thường.

Bằng cách sử dụng AutoCAD để ràng buộc chế độ bắt vuông góc, ta có thể vẽ nhanh hơn. Ví dụ, ta có thể tạo ra hàng loạt các đường thẳng vuông góc với nhau bằng cách bật chế độ bắt vuông góc trước khi bắt đầu vẽ. Bởi vì tất cả các đường thẳng đều được ràng buộc theo trục ngang và đứng nên ta vẫn có thể vẽ nhanh hơn nhiều mà vẫn yên tâm là các đường thẳng luôn vuông góc với nhau.



Bật chế độ bắt vuông góc



Tắt chế độ bắt vuông góc

Khi ta di chuyển con trỏ chuột, một sợi dây vô hình kéo đường thẳng vào theo trục ngang và đứng, tùy thuộc vào trục nào gần con trỏ chuột nhất. AutoCAD bỏ qua chế độ bắt vuông góc trong quan sát phối cảnh, hoặc khi ta nhập tọa độ thông qua dòng lệnh, hoặc khi ta thực hiện theo chế độ bắt đối tượng nào đó.

Để bật hoặc tắt chế độ bắt vuông góc, ta sử dụng thuộc tính OrthoOn. Thuộc tính này cần số liệu đầu vào kiểu Boolean. Gán bằng TRUE khi bật và FALSE để tắt.

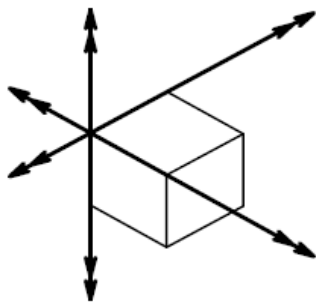
### **Bật chế độ bắt vuông góc trong khung nhìn hiện hành**

```
ThisDrawing.ActiveViewport.OrthoOn = True
```

## **7.3. Vẽ đường tạm**

Ta có thể tạo ra các đường tạm kéo dài vô tận theo một hoặc hai phương. Đường tạm kéo dài về một phương gọi là tia (ray). Đường tạm kéo dài về cả hai phương được gọi là xline. Các đường tạm được sử dụng để tham khảo khi vẽ các đối tượng khác. Ví dụ, ta có thể dùng đường tạm để tìm tâm của một tam giác, tạo một giao điểm tạm thời để sử dụng trong chế độ bắt đối tượng.

Đường tạm không ảnh hưởng đến tổng diện tích của bản vẽ, và do đó, kích thước vô hạn của đường tạm không ảnh hưởng đến chế độ thu phóng hoặc các điểm nhìn. Ta có thể di chuyển, quay, và sao chép các đường tạm giống như cách ta thực hiện với các đối tượng khác. Đôi lúc, ta cũng có thể tạo các đường tạm trong một lớp riêng, để sau đó làm đóng băng hoặc tắt lớp đó đi trước khi in.

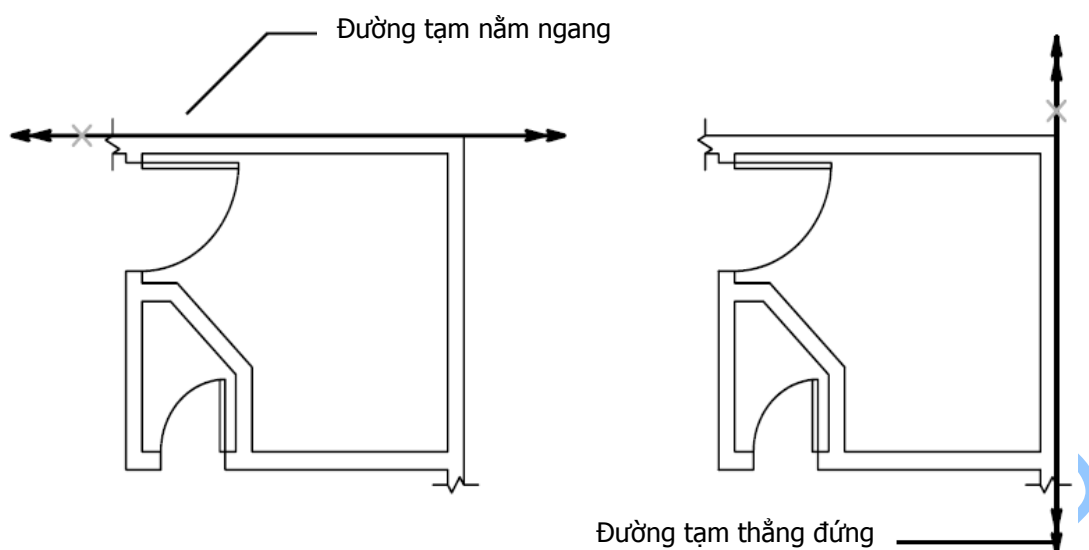


Đường tạm

### **7.3.1. Tạo đường tạm Xline**

Đường tạm Xline có thể được đặt ở bất cứ đâu trong không gian 3D và được kéo dài vô tận về cả hai hướng. Để tạo một đường xline, ta sử dụng phương thức AddXLine. Phương thức này cần tham số đầu vào là hai điểm, ta có thể nhập hoặc lựa chọn hai điểm để xác định phương của đường Xline. Điểm đầu tiên, điểm gốc, được xem là điểm giữa của đường xline.





### Thêm đường tạm

Đoạn mã ví dụ sau sẽ tạo một đối tượng Xline sử dụng hai điểm cho trước là (5,0,0) và (1,1,0).

```
Sub Ch3_AddXLine()
    Dim xlineObj As AcadXline
    Dim basePoint(0 To 2) As Double
    Dim directionVec(0 To 2) As Double

    ' Định nghĩa đường Xline
    basePoint(0) = 2#: basePoint(1) = 2#: basePoint(2) = 0#
    directionVec(0) = 1#: directionVec(1) = 1#: directionVec(2) = 0#

    ' Tạo đường Xline trong không gian mô hình
    Set xlineObj = ThisDrawing.ModelSpace.AddXline _
        (basePoint, directionVec)
    ThisDrawing.Application.ZoomAll
End Sub
```

### 7.3.2. Truy vấn đường tạm Xline

Sau khi đã được tạo ra, ta có thể truy vấn điểm đầu tiên của đường xline bằng thuộc tính BasePoint. Điểm thứ hai dùng để tạo đường xline không được lưu trực tiếp trong đối tượng. Thay vào đó, ta phải sử dụng thuộc tính DirectionVector để lấy giá trị vector chỉ phương của đường xline.

#### Truy vấn đường tạm

Ví dụ sau đây sẽ tìm điểm cơ sở và vector chỉ phương của đường xline vừa được tạo ở trên.

```
Dim BPoint As Variant
Dim Vector As Variant

Set BPoint = xlineObj.BasePoint
Set Vector = xlineObj.DirectionVector
```

### 7.3.3. Tạo tia

Tia là một đường thẳng trong không gian 3D, bắt đầu tại một điểm đã được chỉ ra trước còn đầu kia kéo dài vô tận. Không giống như đường tạm xline, kéo dài theo cả hai hướng, tia chỉ kéo dài về một hướng. Và do vậy, tia sẽ giúp ta giảm được cảm giác phân mảnh do quá nhiều đường xline tạo nên.

Cũng là đường tạm nên tia không được xét đến trong các lệnh có liên quan đến việc hiển thị vùng đối tượng.

### 7.3.4. Truy vấn tia

Sau khi đã được tạo ra, ta có thể truy vấn điểm đầu tiên của tia sử dụng thuộc tính BasePoint. Điểm thứ 2 dùng để tạo tia không được lưu trong đối tượng. Thay vào đó, ta sử dụng thuộc tính DirectionVector để lấy lại giá trị vector chỉ phương của tia.

#### Thêm, truy vấn và hiệu chỉnh đối tượng Tia

Đoạn mã ví dụ sau sẽ tạo ra một đối tượng Tia bằng việc dùng 2 điểm (3,3,0) và (4,4,0). Sau đó truy vấn điểm cơ sở hiện hành và vector chỉ phương và hiển thị kết quả trong hộp thông báo. Sau đó, tiến hành thay đổi vector chỉ phương và hiển thị lại tia.

```
Sub Ch3_EditRay()  
    Dim rayObj As AcadRay  
    Dim basePoint(0 To 2) As Double  
    Dim secondPoint(0 To 2) As Double  
  
    ' Xác định tia  
    basePoint(0) = 3#: basePoint(1) = 3#: basePoint(2) = 0#  
    secondPoint(0) = 4#: secondPoint(1) = 4#: secondPoint(2) = 0#  
  
    ' Tạo tia trong không gian mô hình  
    Set rayObj = ThisDrawing.ModelSpace.AddRay _  
        (basePoint, secondPoint) _  
    ThisDrawing.Application.ZoomAll  
  
    ' Hiển thị các thông tin của tia  
    MsgBox "The base point of the ray is: " & _  
        rayObj.BasePoint(0) & ", " & _  
        rayObj.BasePoint(1) & ", " & _  
        rayObj.BasePoint(2) & vbCrLf & _  
        "The directional vector for the ray is: " & _  
        rayObj.DirectionVector(0) & ", " & _  
        rayObj.DirectionVector(1) & ", " & _  
        rayObj.DirectionVector(2), , "Edit Ray"  
  
    ' Thay đổi hướng cho tia  
    Dim newVector(0 To 2) As Double  
    newVector(0) = -1  
    newVector(1) = 1  
    newVector(2) = 0  
    rayObj.DirectionVector = newVector  
    ThisDrawing.Regen False  
    MsgBox "The base point of the ray is: " & _  
        rayObj.BasePoint(0) & ", " & _
```

```

rayObj.basePoint(1) & ", " & _
rayObj.basePoint(2) & vbCrLf & _
"The directional vector for the ray is: " & _
rayObj.DirectionVector(0) & ", " & _
rayObj.DirectionVector(1) & ", " & _
rayObj.DirectionVector(2), , "Edit Ray"

```

End Sub

## 7.4. Tính toán điểm và các giá trị liên quan

Bằng cách sử dụng các phương thức có sẵn trong đối tượng Utility, ta có thể nhanh chóng giải các bài toán hoặc định vị điểm trong bản vẽ. Nhờ vào các phương thức trong đối tượng Utility, ta có thể thực hiện được những việc sau:

- Tìm góc hợp của một đường thẳng và trục X bằng cách sử dụng phương thức AngleFromAxis
- Chuyển đổi góc kiểu string sang giá trị kiểu real (double) bằng cách sử dụng phương thức AngleToReal
- Chuyển đổi một góc từ giá trị real (double) sang giá trị kiểu string bằng cách sử dụng phương thức AngleToString
- Chuyển đổi khoảng cách từ kiểu string sang kiểu real (double) bằng cách sử dụng phương thức DistanceToReal.
- Tạo biến kiểu variant chứa mảng các số kiểu số nguyên (integer), số thực (float),... bằng cách sử dụng phương thức CreateTypedArray.
- Tìm một điểm ở một góc nằm cách một điểm cho trước một khoảng bằng cách sử dụng phương thức PolarPoint
- Tịnh tiến một điểm từ hệ tọa độ này sang hệ tọa độ khác bằng cách sử dụng phương thức TranslateCoordinates
- Tìm khoảng cách giữa hai điểm do người dùng nhập vào bằng cách sử dụng phương thức GetDistance

### Tìm khoảng cách giữa hai điểm bằng cách sử dụng phương thức GetDistance

Ví dụ sau đây sử dụng phương thức GetDistance để lấy tọa độ các điểm và hàm MsgBox để hiển thị kết quả tính toán khoảng cách.

```

Sub Ch3_GetDistanceBetweenTwoPoints()
    Dim returnDist As Double
    returnDist = ThisDrawing.Utility.GetDistance _
    (, "Pick two points.")
    MsgBox "The distance between the two points is: " &
    returnDist
End Sub

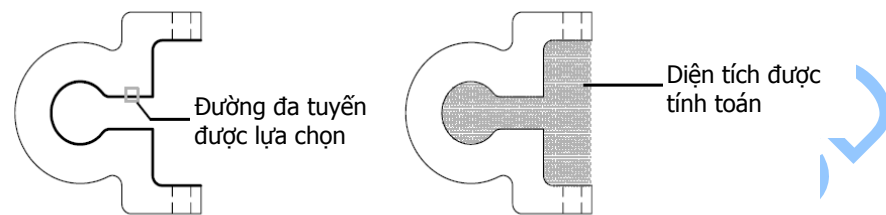
```

## 7.5. Tìm diện tích

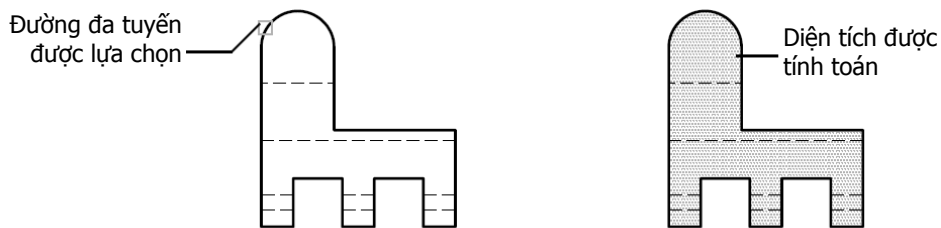
Bằng cách sử dụng thuộc tính Area, ta có thể tìm diện tích của một cung tròn, hình tròn, elip, đa tuyến có sử dụng bề dày nét, đa tuyến thông thường, vùng, hoặc đường spline kín trong mặt phẳng.

Diện tích được tính toán thường khác nhau tùy thuộc vào loại đối tượng mà ta truy vấn:

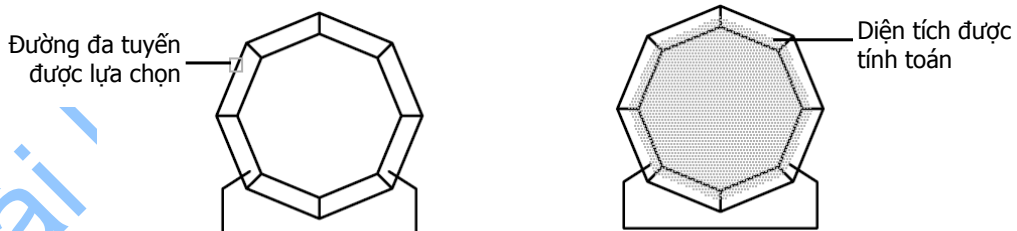
- Đường đa tuyến kín hoặc đa giác: đối với đường đa tuyến có bề dày, vùng diện tích tính toán được xác định theo tâm bề rộng của đường đa tuyến
- Đối tượng hở như đường cong spline hở và đường đa tuyến hở: diện tích được xác định như là khi có một đường thẳng nối điểm đầu và điểm cuối của các đối tượng hở này
- Miền: diện tích được xác định là phần diện tích kết hợp của các đối tượng cấu thành miền đó.



Đường polyline hở



Đường đa tuyến kín



Đường đa tuyến có bề dày

### 7.5.1. Xác định diện tích vùng tự định nghĩa

Ta có thể xác định diện tích của một vùng kín bất kỳ xác định bởi các điểm trong không gian 2D và 3D do người dùng nhập vào. Các điểm này cần phải nằm trên cùng một mặt phẳng.

**Để xác định diện tích vùng dựa trên các điểm do người dùng nhập vào:**

- Sử dụng phương thức GetPoint trong vòng lặp để cho người dùng nhập các điểm
- Tạo đường đa tuyến có bề dày từ các điểm do người dùng nhập vào. Sử dụng phương thức AddLightweightPolyline để tạo đường đa tuyến

- Sử dụng thuộc tính Area để xác định diện tích của đường đa tuyến vừa mới tạo ra
- Xóa đường đa tuyến sử dụng phương thức Erase.

### Tính diện tích từ các điểm do người dùng nhập vào

Ví dụ sau nhắc người dùng nhập vào 5 điểm. Sau đó ta tạo một đường đa tuyến từ các điểm nói trên. Sau đó đường đa tuyến sẽ được khép kín, và diện tích của đường đa tuyến sẽ được hiển thị trong hộp thông báo.

```
Sub Ch3_CalculateDefinedArea()
    Dim p1 As Variant
    Dim p2 As Variant
    Dim p3 As Variant
    Dim p4 As Variant
    Dim p5 As Variant

    ' Nhập vào các điểm
    p1 = ThisDrawing.Utility.GetPoint(, vbCrLf & "First point: ")
    p2 = ThisDrawing.Utility.GetPoint(p1, vbCrLf & "Second point: ")
    p3 = ThisDrawing.Utility.GetPoint(p2, vbCrLf & "Third point: ")
    p4 = ThisDrawing.Utility.GetPoint(p3, vbCrLf & "Fourth point: ")
    p5 = ThisDrawing.Utility.GetPoint(p4, vbCrLf & "Fifth point: ")

    ' Tạo đường đa tuyến 2D từ các điểm vừa nhập
    Dim polyObj As AcadLWPolyline
    Dim vertices(0 To 9) As Double
    vertices(0) = p1(0): vertices(1) = p1(1)
    vertices(2) = p2(0): vertices(3) = p2(1)
    vertices(4) = p3(0): vertices(5) = p3(1)
    vertices(6) = p4(0): vertices(7) = p4(1)
    vertices(8) = p5(0): vertices(9) = p5(1)
    Set polyObj = ThisDrawing.ModelSpace.AddLightWeightPolyline _
        (vertices)

    polyObj.Closed = True
    ThisDrawing.Application.ZoomAll

    ' Tính diện tích đường đa tuyến
    MsgBox "The area defined the by points is " & _
        polyObj.Area, , "Calculate Defined Area"
End Sub
```

## 8. Nhắc người dùng nhập liệu

Đối tượng Utility, đối tượng con của đối tượng Document, định nghĩa các phương thức nhập liệu từ người dùng. Các phương thức nhập liệu từ người dùng sẽ hiện thị ở dấu nhắc dòng lệnh AutoCAD yêu cầu người sử dụng nhập vào rất nhiều dạng dữ liệu khác nhau. Loại dữ liệu từ người dùng kiểu này là cách hữu dụng nhất để nhập các dữ liệu tương tác như hệ tọa độ màn hình, chọn các đối tượng, các giá trị kiểu short-string hoặc kiểu số. Nếu ứng dụng đòi hỏi phải nhập rất nhiều lựa chọn và số liệu khác nhau thì ta nên sử dụng một hộp thoại nhập liệu tốt hơn là việc sử dụng từng dấu nhắc dòng lệnh đơn lẻ như vậy.

Mỗi phương thức nhập dữ liệu từ người dùng đều hiển thị dấu nhắc trên dòng lệnh AutoCAD và trả về giá trị phù hợp với kiểu giá trị cần nhập vào. Ví dụ, phương

thức GetString trả về kiểu string, phương thức GetPoint trả về biến kiểu variant (sẽ là mảng có ba phần tử kiểu double) và phương thức GetInteger trả về giá trị kiểu số nguyên (integer). Ta còn có thể kiểm soát nhiều hơn nữa việc nhập dữ liệu từ người dùng bằng cách sử dụng phương thức InitializedUserInput. Phương thức này cho phép ta điều khiển được nhiều thứ hơn, chẳng hạn như các giá trị NULL (khi người dùng nhấn phím ENTER), nhập vào số 0 hoặc giá trị âm, và nhập vào một đoạn chữ bất kỳ.

Để cho dấu nhắc chỉ hiển thị trên một dòng riêng, sử dụng hằng vbCrLf ở phía đầu đoạn chuỗi chứa dấu nhắc.

## 8.1. Phương thức GetString

Phương thức GetString nhắc người dùng nhập vào một chuỗi ở dòng lệnh của AutoCAD. Phương thức này cần có 2 tham số. Tham số đầu điều khiển việc nhập dấu cách (khoảng trắng) trong chuỗi nhập vào. Nếu được gán là 0, không được nhập dấu cách (khi nhấn phím SPACE sẽ kết thúc nhập liệu); nếu gán là 1, chuỗi có thể chứa dấu cách (nhấn phím ENTER sẽ kết thúc nhập liệu). Tham số thứ hai là chuỗi nhằm nhắc người dùng nhập liệu.

### Nhập một chuỗi từ người dùng trong dòng lệnh AutoCAD

Ví dụ sau hiển thị dòng nhắc "Enter Your Name", và yêu cầu người dùng kết thúc nhập liệu khi nhấn phím ENTER (dấu cách cũng được cho phép nhập trong chuỗi). Giá trị sau đó được lưu trong biến retVal và được hiển thị thông qua hộp thông báo.

```
Sub Ch3_GetStringFromUser()  
Dim retVal As String  
    retVal = ThisDrawing.Utility.GetString _  
        (1, vbCrLf & "Enter your name: ")  
    MsgBox "The name entered was: " & retVal  
End Sub
```

Phương thức GetString không cần phải thực hiện trước lời gọi phương thức InitializeUserInput.

## 8.2. Phương thức GetPoint

Phương thức GetPoint nhắc người dùng nhập vào điểm trên dòng lệnh AutoCAD. Phương thức này cần có 2 tham số, một là điểm trước điểm cần nhập (có thể có) và một là chuỗi nhắc người dùng nhập liệu. Nếu có điểm trước đó, sẽ xuất hiện đường nối từ điểm đó với vị trí con trỏ chuột. Để điều khiển nhập liệu của người dùng, phương thức này có thể thực hiện sau khi thực hiện lời gọi đến phương thức InitializeUserInput.

### Nhập điểm từ người dùng

Ví dụ sau nhắc người dùng nhập vào 2 điểm, sau đó vẽ một đường thẳng nối hai điểm này lại.

```
Sub Ch3_GetPointsFromUser()  
    Dim startPnt As Variant  
    Dim endPnt As Variant  
    Dim prompt1 As String  
    Dim prompt2 As String
```

```

prompt1 = vbCrLf & "Enter the start point of the line: "
prompt2 = vbCrLf & "Enter the end point of the line: "

' Chọn điểm đầu tiên không có điểm cơ sở
startPnt = ThisDrawing.Utility.GetPoint(, prompt1)

' Sử dụng điểm đã nhập ở trên làm điểm cơ sở
endPnt = ThisDrawing.Utility.GetPoint(startPnt, prompt2)

' Tạo đường thẳng nối hai điểm trên
ThisDrawing.ModelSpace.AddLine startPnt, endPnt
ThisDrawing.Application.ZoomAll
End Sub

```

### 8.3. Phương thức GetKeyword

Phương thức `GetKeyword` nhắc người dùng nhập vào từ khóa ở đầu nhắc dòng lệnh AutoCAD. Phương thức này chỉ cần có một tham số, là chuỗi chứa nội dung dòng nhắc lệnh. Từ khóa và các tham số đầu vào được định nghĩa khi thực hiện lời gọi phương thức `InitializeUserInput`.

#### Nhập từ khóa từ dòng lệnh AutoCAD

Ví dụ sau sẽ bắt người dùng nhập vào một từ khóa bằng cách thiết lập tham số cho phương thức `InitializeUserInput` là 1 (nghĩa là không cho phép người dùng nhập vào một giá trị NULL). Tham số thứ hai thiết lập danh sách các từ khóa.

```

Sub Ch3_KeyWord()
    Dim keyWord As String
    ThisDrawing.Utility.InitializeUserInput 1, "Line Circle Arc"
    keyWord = ThisDrawing.Utility.GetKeyword _
        (vbCrLf & "Enter an option (Line/Circle/Arc): ")
    MsgBox keyWord, , "GetKeyword Example"
End Sub

```

Một kiểu nhắc nhập từ khóa thân thiện hơn là kiểu cài đặt một giá trị mặc định nếu người dùng nhấn phím ENTER (nhập giá trị NULL). Chú ý một thay đổi nhỏ trong ví dụ sau:

```

Sub Ch3_KeyWord2()
    Dim keyWord As String
    ThisDrawing.Utility.InitializeUserInput 0, "Line Circle Arc"
    keyWord = ThisDrawing.Utility.GetKeyword _
        (vbCrLf & "Enter an option (Line/Circle/<Arc>): ")
    If keyWord = "" Then keyWord = "Arc"
    MsgBox keyWord, , "GetKeyword Example"
End Sub

```

### 8.4. Điều khiển quá trình nhập liệu của người dùng

Ta có thể sử dụng phương thức `InitializeUserInput` để định nghĩa các từ khóa hoặc hạn chế kiểu nhập liệu trong các phương thức nhập liệu từ người dùng. Cách sử dụng và các tham số cũng tương tự như hàm `initget` trong AutoLISP. Phương thức `InitializeUserInput` có thể được sử dụng cho các phương thức sau: `GetAngle`, `GetCorner`, `GetDistance`, `GetInteger`, `GetKeyword`, `GetOrientation`, `GetPoint`, and `GetReal`. Phương thức `InitializeUserInput` không được sử dụng trong phương thức



GetString. Ta có thể sử dụng phương thức GetInput lấy giá trị kiểu string (từ khóa hoặc một giá trị bất kỳ) khi người sử dụng không nhập vào một giá trị kiểu string.

Phương thức InitializeUserInput nhận 2 tham số. Tham số đầu tiên là một giá trị định ra lựa chọn cho phương thức mà người dùng sẽ nhập dữ liệu. Tham số thứ hai là một chuỗi định nghĩa các từ khóa.

### Nhập một giá trị Integer hoặc một từ khóa từ dòng lệnh AutoCAD

Ví dụ sau nhắc người dùng nhập vào một số nguyên dương hoặc một từ khóa:

```
Sub Ch3_UserInput()  
    ' Tham số đầu tiên của InitializeUserInput (6)  
    ' giới hạn việc nhập các số nguyên dương, không âm  
    ' Tham số thứ 2 là danh sách các từ khóa  
    ThisDrawing.Utility.InitializeUserInput 6, "Big Small Regular"  
  
    ' Gán biến promptStr  
    Dim promptStr As String  
    promptStr = vbCrLf & "Enter the size or (Big/Small/<Regular>):"  
  
    ' Nhập vào một từ khóa khi dùng phương thức GetInteger sẽ gây lỗi  
    ' vì thế cần phải cho phép chương trình tiếp tục và kiểm tra  
    ' thông báo lỗi và thiết lập bộ xử lý lỗi để tiếp tục  
    ' thực hiện khi gặp lỗi.  
    On Error Resume Next  
  
    ' Gán giá trị do người dùng nhập vào  
    Dim returnInteger As Integer  
    returnInteger = ThisDrawing.Utility.GetInteger(promptStr)  
  
    ' Kiểm tra lỗi. Nếu lỗi trùng với giá trị như ở dưới  
    ' thì sử dụng phương thức GetInput để nhận lại giá  
    ' trị kiểu string được trả về; trong trường hợp khác,  
    ' sử dụng giá trị của biến returnInteger  
    If Err.Description = "User input is a keyword" Then  
        Dim returnString As String  
        returnString = ThisDrawing.Utility.GetInput()  
        Err.Clear  
    Else  
        If returnInteger = 0 Then          'Nếu nhấn phím ENTER  
            returnString = "Regular" 'Gán giá trị mặc định  
        Else                               'Nếu không,  
            returnString = returnInteger  
            'Sử dụng giá trị nhập vào  
        End If  
    End If  
    ' Hiện thị kết quả  
    MsgBox returnString, , "InitializeUserInput Example"  
End Sub
```

## 9. Truy xuất dòng lệnh của AutoCAD

Ta có thể gửi một lệnh trực tiếp đến dòng lệnh AutoCAD bằng cách sử dụng phương thức SendCommand. Phương thức SendCommand sẽ gửi một chuỗi đơn đến dòng lệnh AutoCAD. Chuỗi này phải chứa các đầy đủ các tham số theo trình tự thực thi của lệnh cần thực hiện. Dấu cách hoặc mã ASCII tương đương với phím

ENTER trong chuỗi tương đương với việc nhấn phím ENTER trên bàn phím. Không giống như môi trường AutoLISP, việc thực hiện phương thức SendCommand mà không có tham số là không hợp lệ.

### Gửi một lệnh đến dòng lệnh AutoCAD

Ví dụ sau tạo một vòng tròn có tâm (2,2,0) và bán kính là 4. Sau đó thực hiện phóng đại lên tất cả bản vẽ (lệnh ZoomAll). Chú ý rằng có một dấu cách ở cuối mỗi chuỗi thể hiện việc nhấn phím ENTER khi kết thúc dòng lệnh.

```
Sub Ch3_SendACommandToAutoCAD()  
    ThisDrawing.SendCommand "_Circle 2,2,0 4 "  
    ThisDrawing.SendCommand "_zoom a "  
End Sub
```

## 10. Thao tác khi không mở bản vẽ nào

Lúc nào AutoCAD cũng khởi động với một bản vẽ mới hoặc một bản vẽ đã có. Tuy nhiên, cũng có lúc mà không có bản vẽ nào được mở trong suốt phiên làm việc của AutoCAD.

Nếu ta đóng tất cả các bản vẽ trong giao diện người dùng AutoCAD, ta sẽ chú ý thấy có một vài thay đổi trong cửa sổ ứng dụng. Các trình đơn có hiệu lực được giảm thiểu chỉ còn File, View, Window, và Help. Trong các trình đơn này, các lựa chọn cũng được giảm thiểu. Cũng cần phải chú ý rằng không còn dòng lệnh trong giao diện của AutoCAD.

Tương tự như vậy, giao tiếp ActiveX chỉ cho phép một số thao tác sau khi không có bản vẽ nào được mở:

- Mở một bản vẽ
- Tạo một bản vẽ mới
- Nhập một bản vẽ
- Thoát khỏi AutoCAD

Những thao tác này đều có trong tập đối tượng Documents. Bên cạnh tập phương thức và thuộc tính khá hạn chế của đối tượng Application, các phương thức và thuộc tính trong tập đối tượng Documents là giao tiếp duy nhất có hiệu lực khi không có bản vẽ nào được mở. Nếu thực hiện các thao tác khác, chẳng hạn như cố tình truy cập vào các lựa chọn của người dùng, thì sẽ phát sinh lỗi.

Ta sử dụng thuộc tính Count trong tập đối tượng Documents để kiểm tra xem AutoCAD có ở trạng thái không có bản vẽ nào không. Nếu Documents.Count=0 có nghĩa là AutoCAD ở trạng thái không có bản vẽ nào. Nếu Documents.Count>0 thì có nghĩa là đã có ít nhất một bản vẽ được mở.

Cần phải đặc biệt lưu ý rằng trong VBA, đối tượng ThisDrawing không được định nghĩa khi AutoCAD ở trạng thái không có bản vẽ nào. Đó là do đối tượng ThisDrawing tham chiếu đến bản vẽ hiện hành và do vậy khi ở trạng thái không có bản vẽ nào mở thì việc cố gắng thực hiện lời gọi đến một Macro nào đó sử dụng đối tượng ThisDrawing sẽ làm phát sinh lỗi khi chạy chương trình (lỗi run-time).

Để tránh lỗi này, ta sử dụng một hàm VBA là `GetObject` để tạo kết nối với AutoCAD khi không có bản vẽ nào được mở.

## 11. Nhập vào các định dạng khác

Ta có thể nhập các bản vẽ hoặc hình ảnh của các ứng dụng khác bằng cách thực hiện quá trình mở theo một định dạng nào đó. AutoCAD có thể xử lý một số dạng thức chuyển đổi các tệp DXF, SAT, BMP, và PostScript. Trong tất cả các phiên bản AutoCAD, ta đều có thể thực hiện quá trình nhập một tệp nào đó bằng cách thực hiện phương thức `Import`. Phương thức này cần có 3 tham số đầu vào: tên của tệp cần nhập, điểm chèn vào bản vẽ, và hệ số tỷ lệ khi chèn vào bản vẽ hiện hành.

## 12. Xuất sang các định dạng khác

Khi cần sử dụng bản vẽ AutoCAD trong các chương trình khác, ta có thể chuyển đổi bản vẽ sang một định dạng khác sử dụng phương thức `Export`. Phương thức này sẽ xuất bản vẽ AutoCAD sang các định dạng WMF, SAT, DXF, DWF hoặc BMP. Phương thức `Export` nhận 3 tham số đầu vào: tên của tệp mới cần tạo, phần mở rộng của tệp mới, và tập các đối tượng được lựa chọn để xuất (nếu cần).

Khi xuất sang các định dạng WMF, SAT hoặc BMP thì tập lựa chọn phải khác rỗng. Tập lựa chọn này sẽ xác định rõ đối tượng nào sẽ được xuất sang định dạng khác. Nếu không lựa chọn đối tượng nào, sẽ chẳng có gì được xuất ra ngoài cả và sẽ mắc phải lỗi đối số không hợp lệ.

Khi xuất sang các định dạng EPS và DXF, ta không thể chọn được tập đối tượng. Toàn bộ bản vẽ sẽ được tự động xuất sang các định dạng này.

### Xuất bản vẽ sang định dạng DXF và nhập ngược lại

Ví dụ sau sẽ tạo ra một hình tròn trong bản vẽ hiện hành, sau đó được xuất ra tệp *DXFExport.DXF*. Tiếp theo, chương trình sẽ mở một bản vẽ mới và nhập lại tệp vừa xuất.

```
Sub Ch3_ImportingAndExporting()  
    ' Tạo một đường tròn  
    Dim circleObj As AcadCircle  
    Dim centerPt(0 To 2) As Double  
    Dim radius As Double  
    centerPt(0) = 2: centerPt(1) = 2: centerPt(2) = 0  
    radius = 1  
    Set circleObj = ThisDrawing.ModelSpace.AddCircle _  
        (centerPt, radius)  
    ThisDrawing.Application.ZoomAll  
  
    ' Tạo một tập đối tượng rỗng  
    Dim sset As AcadSelectionSet  
    Set sset = ThisDrawing.SelectionSets.Add("NEWSSET")  
  
    ' Xuất bản vẽ hiện tại thành tệp chỉ định từ trước.  
    Dim exportTệp As String  
    exportTệp = "C:\AutoCAD\DXFExprt"  
    ThisDrawing.Export exportTệp, "DXF", sset
```

```
' Mở bản vẽ mới
ThisDrawing.Application.Documents.Add "acad.dwt"

' Xác định tệp cần nhập vào
Dim importTệp As String
Dim insertPoint(0 To 2) As Double
Dim scaleFactor As Double
importTệp = "C:\AutoCAD\DXFExprt.dxf"
insertPoint(0) = 0: insertPoint(1) = 0: insertPoint(2) = 0
scaleFactor = 2#

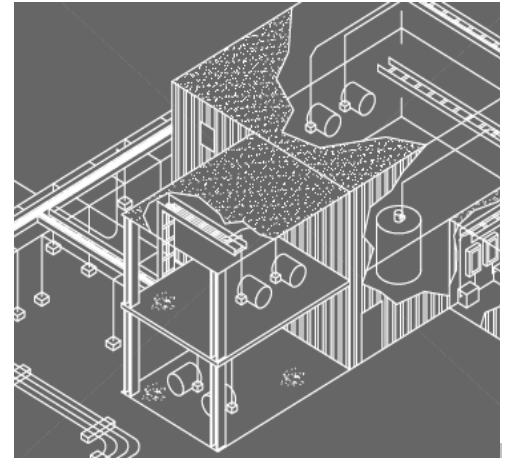
' Nhập vào
ThisDrawing.Import importTệp, insertPoint, scaleFactor
ThisDrawing.Application.ZoomAll
End Sub
```

Tài liệu tham khảo  
BM Tự động hoá TKCĐ

# TẠO VÀ HIỆU CHỈNH THỰC THỂ AutoCAD

Ta có thể tạo rất nhiều đối tượng khác nhau, từ những đối tượng đơn giản như đường thẳng và đường tròn đến những đối tượng phức tạp hơn như đường cong, elip và vùng tô mẫu. Nhìn chung, ta có thể thêm các đối tượng vào không gian mô hình sử dụng phương thức Add. Ngoài ra, ta cũng có thể thêm các đối tượng trong không gian in hay trong một khối.

Sau khi đã được tạo ra, ta có thể thay đổi thuộc tính lớp, màu sắc và kiểu đường của đối tượng đó. Ngoài ra, ta cũng có thể thêm ký tự để ghi chú cho bản vẽ.



Trong chương này

4

- Tạo đối tượng
- Hiệu chỉnh đối tượng
- Sử dụng Lớp, Màu sắc và Kiểu đường
- Thêm văn bản vào bản vẽ

# 1. Tạo đối tượng

Mặc dù có rất nhiều cách khác nhau để tạo một đối tượng đồ họa trong AutoCAD, nhưng ActiveX Automation chỉ có một phương thức để tạo đối tượng cho mỗi đối tượng. Chẳng hạn như trong AutoCAD, có 4 cách khác nhau để tạo một đường tròn: (1) xác định tâm và bán kính, (2) xác định 2 điểm là đường kính, (3) xác định 3 điểm nằm trên chu vi đường tròn, (4) xác định hai đường tiếp tuyến và bán kính. Tuy nhiên, trong ActiveX Automation chỉ có một cách duy nhất để tạo đường tròn, đó là cách sử dụng 2 giá trị tâm và bán kính.

---

**CHÚ Ý:** Các phương thức tạo đối tượng trong VB và VBA dùng CreateObject hoặc Dim với từ khóa New chỉ được sử dụng để tạo mới đối tượng Application trong AutoCAD. Tất cả các đối tượng khác của AutoCAD phải được tạo thông qua phương thức Add hoặc Add<Object> có trong giao tiếp AutoCAD.

---

## 1.1. Xác định đối tượng bao động<sup>1</sup>

Tất cả các đối tượng đồ họa đều được tạo trong tập đối tượng ModelSpace (không gian mô hình), tập đối tượng PaperSpace (không gian in) hoặc trong đối tượng Block (khối).

Tập đối tượng ModelSpace trả về thông qua thuộc tính ModelSpace và tập đối tượng PaperSpace trả về thông qua thuộc tính PaperSpace.

Có thể tham chiếu trực tiếp những đối tượng này hoặc có thể thông qua biến tự định nghĩa. Để tham chiếu trực tiếp, ta tạo đối tượng với toàn bộ cấu trúc phân nhánh. Ví dụ như khi tạo một đối tượng line trong không gian mô hình, ta sử dụng dòng lệnh sau:

```
Set lineObj = ThisDrawing.ModelSpace.AddLine(startPoint,endPoint)
```

Để tham chiếu đối tượng thông qua biến tự định nghĩa, ta sẽ định nghĩa biến có kiểu là AcadModelSpace và AcadPaperSpace, sau đó gán biến này với bản vẽ hiện hành thích hợp. Ví dụ sau sẽ định nghĩa hai biến và gán cho không gian mô hình và không gian in hiện hành:

```
Dim moSpace As AcadModelSpace  
Dim paSpace As AcadPaperSpace  
Set moSpace = ThisDrawing.ModelSpace  
Set paSpace = ThisDrawing.PaperSpace
```

Câu lệnh sau sẽ thêm đối tượng line vào không gian mô hình thông qua biến tự định nghĩa:

```
Set lineObj = moSpace.AddLine(startPoint,endPoint)
```

---

<sup>1</sup> **Đối tượng bao động (Container Object):** là đối tượng chứa các đối tượng khác, và các đối tượng này có thể được lưu trữ hoặc tháo dỡ ngay trong thời gian thực thi chương trình. Khái niệm này ngược với khái niệm **đối tượng bao tĩnh (composition)**, cũng chứa các đối tượng khác, nhưng những đối tượng này lại cố định trong quá trình dịch và thực thi chương trình. – Nguồn Wikipedia.

## 1.2. Tạo đường thẳng – đối tượng line

Đối tượng Line là đối tượng cơ bản nhất trong AutoCAD. Ta có thể tạo rất nhiều loại đường thẳng khác nhau: đường thẳng đơn hoặc nhiều đoạn thẳng có hoặc không bao gồm đoạn cong. Nhìn chung, ta có thể vẽ các đường thẳng bằng cách nhập vào tọa độ của các điểm. Kiểu đường mặc định là CONTINUOUS, là đường liền, nhưng cũng có rất nhiều kiểu đường khác nhau sử dụng dấu chấm hoặc dấu gạch đứt nét.

Để tạo một đường thẳng, ta sử dụng một trong những phương thức sau:

AddLine	Tạo đường thẳng đi qua hai điểm.
AddLightweightPolyline	Tạo đường đa tuyến 2D có bề dày từ danh sách đỉnh của nó.
AddMLine	Tạo đường thẳng nét đôi.
AddPolyline	Tạo đường đa tuyến 2D hoặc 3D.

Đối tượng Line và MLine tạo ra trong hệ tọa độ toàn cục (World Coordinate System-WCS). Còn các đối tượng Polyline và LightweightPolyline tạo ra trong hệ tọa độ địa phương (Object Coordinate System-OCS). Để có thêm thông tin về hệ tọa độ OSC, xem thêm mục “Chuyển trục tọa độ” trang 219.

### Tạo đối tượng Polyline

Ví dụ sau sử dụng phương thức AddLightweightPolyline để tạo một đường đa tuyến có hai đoạn thẳng sử dụng hệ tọa độ 2D (2,4), (4,2) và (6,4).

```
Sub Ch4_AddLightWeightPolyline()  
    Dim plineObj As AcadLWPolyline  
    Dim points(0 To 5) As Double  
  
    ' Định nghĩa tọa độ 2D của đường đa tuyến  
    points(0) = 2: points(1) = 4  
    points(2) = 4: points(3) = 2  
    points(4) = 6: points(5) = 4  
  
    ' Tạo đối tượng LightweightPolyline trong không gian mô hình  
    Set plineObj = ThisDrawing.ModelSpace.  
        AddLightWeightPolyline(points)  
    ThisDrawing.Application.ZoomAll  
End Sub
```

## 1.3. Tạo đối tượng cong

Ta có thể tạo rất nhiều loại đối tượng đường cong khác nhau trong AutoCAD, bao gồm đường cong spline, hình tròn, cung tròn, elip. Tất cả các đối tượng đường cong đều được tạo trong mặt phẳng XY của hệ tọa độ toàn cục hiện hành.

Để tạo một đường cong, ta có thể sử dụng một trong những phương thức sau:

AddArc	Tạo một cung tròn khi cho trước tâm, bán kính, góc bắt đầu và góc kết thúc.
AddCircle	Tạo đường tròn khi cho trước tâm và bán kính.



**AddEllipse** Tạo hình Ellipse khi cho trước tâm, một điểm trên trục chính và tỉ số bán kính.

**AddSpline** Tạo đường cong NURBS (nonuniform rational B-spline – đường cong B-spline hữu tỉ không đều) bậc hai hoặc bậc ba.

### Tạo đối tượng Spline

Ví dụ sau tạo một đường Spline trong không gian mô hình sử dụng ba điểm (0,0,0), (5,5,0) và (10,0,0). Đường Spline có tiếp tuyến đầu và cuối là (0.5,0.5,0.0).

```
Sub Ch4_CreateSpline()  
    ' Tạo đối tượng Spline trong không gian mô hình  
    ' Khai báo biến cần thiết  
    Dim splineObj As AcadSpline  
    Dim noOfPoints As Integer  
    Dim startTan(0 To 2) As Double  
    Dim endTan(0 To 2) As Double  
    Dim fitPoints(0 To 8) As Double  
  
    ' Định nghĩa biến  
    noOfPoints = 3  
    startTan(0) = 0.5: startTan(1) = 0.5: startTan(2) = 0  
    endTan(0) = 0.5: endTan(1) = 0.5: endTan(2) = 0  
    fitPoints(0) = 1: fitPoints(1) = 1: fitPoints(2) = 0  
    fitPoints(3) = 5: fitPoints(4) = 5: fitPoints(5) = 0  
    fitPoints(6) = 10: fitPoints(7) = 0: fitPoints(8) = 0  
  
    ' Tạo đối tượng spline  
    Set splineObj = ThisDrawing.ModelSpace.AddSpline _  
        (fitPoints, startTan, endTan)  
    ZoomAll  
End Sub
```

Để có thêm thông tin về đối tượng Spline, xem thêm về đối tượng Spline và phương thức AddSpline trong tài liệu “*ActiveX and VBA Reference*”.

## 1.4. Tạo đối tượng điểm

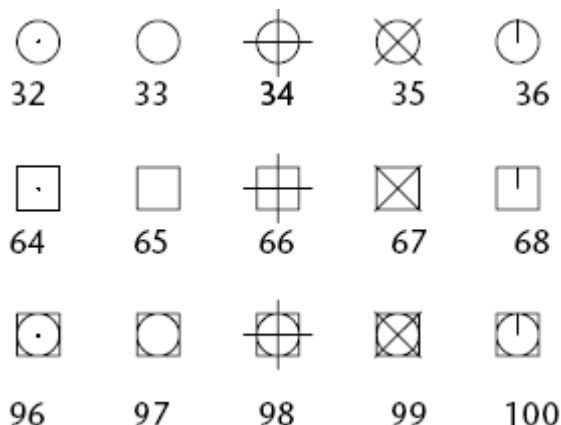
Đối tượng Point đôi khi cũng rất hữu dụng, chẳng hạn như để tạo một nút hoặc là một điểm tham chiếu để từ đó ta tiến hành bắt điểm hoặc thực hiện lệnh Offset. Ta cũng có thể tạo kiểu cho đối tượng Point và thiết lập kích thước tương đối so với màn hình hoặc theo kích thước tuyệt đối.

### 1.4.1. Điều chỉnh kiểu hiển thị của đối tượng Point

Biến hệ thống PDMODE và PDSIZE điều khiển kiểu hiển thị của đối tượng Point. Giá trị của PDMODE là 0, 2, 3 và 4 quy định cách vẽ qua một điểm còn giá trị 1 nghĩa là không hiển thị gì cả.



Cộng các giá trị 32, 64, 96 (quy định hình vẽ bao quanh điểm) với các giá trị trước sẽ tạo ra rất nhiều loại ký hiệu điểm khác nhau:



Biến PDSIZE điều khiển kích thước của ký hiệu điểm, trừ khi giá trị PDMODE là 0 và 1. Khi giá trị PDSIZE bằng 0 thì điểm sẽ có kích thước 5% so với chiều cao của vùng đồ họa. Giá trị PDSIZE dương sẽ xác định kích thước tuyệt đối của ký hiệu điểm, còn giá trị âm sẽ xác định phần trăm so với kích thước khung nhìn. Kích thước của tất cả các điểm đều được tính toán lại mỗi khi tái tạo lại bản vẽ.

Sau khi thay đổi giá trị PDMODE và PDSIZE, hình dạng của ký hiệu điểm sẽ thay đổi sau khi bản vẽ được tái tạo lại.

Để thiết lập giá trị cho biến PDMODE và PDSIZE, ta sử dụng phương thức SetVariable.

#### Tạo đối tượng Point và thay đổi kiểu hiển thị

Đoạn mã sau sẽ tạo một đối tượng Point trong không gian mô hình ở tọa độ (5,5,0), và sau đó cập nhật lại giá trị biến PDMODE và PDSIZE.

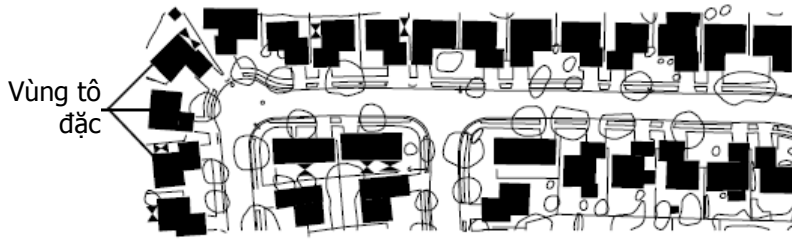
```
Sub Ch4_CreatePoint()
    Dim pointObj As AcadPoint
    Dim location(0 To 2) As Double

    ' Xác định vị trí điểm
    location(0) = 5#: location(1) = 5#: location(2) = 0#

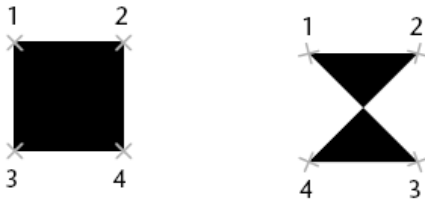
    ' Tạo điểm mới
    Set pointObj = ThisDrawing.ModelSpace.AddPoint(location)
    ThisDrawing.SetVariable "PDMODE", 34
    ThisDrawing.SetVariable "PDSIZE", 1
    ZoomAll
End Sub
```

### 1.5. Tạo vùng tô đặc

Ta có thể tạo một hình tam giác hoặc tứ giác rồi sau đó tô cho vùng đó. Để có hiệu quả nhanh hơn, ta nên tạo đối tượng khi đã tắt biến hệ thống FILLMODE và sau đó bật biến FILLMODE khi đã vẽ xong.



Khi tạo hình tứ giác được tô đặc, trình tự các điểm thứ 3 và 4 sẽ xác định hình dạng cuối cùng. So sánh 2 ví dụ sau:



Để tạo vùng được tô đặc, ta sử dụng phương thức AddSolid.

### Tạo đối tượng tô đặc

Đoạn mã ví dụ sau tạo hình tứ giác tô đặc trong không gian mô hình sử dụng hệ tọa độ sau (0,0,0), (5,0,0), (5,8,0) và (0,8,0).

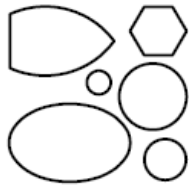
```
Sub Ch4_CreateSolid()
    Dim solidObj As AcadSolid
    Dim point1(0 To 2) As Double
    Dim point2(0 To 2) As Double
    Dim point3(0 To 2) As Double
    Dim point4(0 To 2) As Double

    ' Xác định vùng tô đặc
    point1(0) = 0#: point1(1) = 0#: point1(2) = 0#
    point2(0) = 5#: point2(1) = 0#: point2(2) = 0#
    point3(0) = 5#: point3(1) = 8#: point3(2) = 0#
    point4(0) = 0#: point4(1) = 8#: point4(2) = 0#

    ' Tạo đối tượng trong không gian mô hình
    Set solidObj = ThisDrawing.ModelSpace.AddSolid _
        (point1, point2, point3, point4)
    ZoomAll
End Sub
```

## 1.6. Tạo miền

Miền là khu vực bao kín được tạo từ các đối tượng khép kín gọi là các vòng kín. Vòng kín là một đường cong hoặc chuỗi đường cong xác định một vùng trên mặt phẳng có đường bao không cắt nhau. Vòng kín có thể là sự kết hợp giữa các đối tượng Line, Lightweight, Polyline, Circle, Arc, Elliptical arc (cung elip), Spline, 3D face, Trace và Solid. Các đối tượng cấu thành vòng kín phải là đối tượng khép kín hoặc tạo thành vùng khép kín bằng cách sử dụng chung điểm cuối cùng với các đối tượng khác. Tất cả những đối tượng này phải đồng phẳng (nằm trên cùng một mặt phẳng).



Không thể tạo thành miền nếu đường cong hở có giao điểm ở bên trong.

Các đối tượng như đường 3DPolyline và mặt lưới có thể được chuyển thành miền sau khi tiến hành bung các đối tượng. Miền cũng không thể được cấu thành từ những đối tượng giao cắt nhau tạo thành vùng kín, chẳng hạn như các cung tròn giao nhau hoặc các đường cong giao nhau.

Các vòng kín tạo thành miền phải được định nghĩa ở dạng mảng đối tượng.

Ngoài ra, ta cũng có thể tạo vùng tô mẫu hoặc tạo bóng cho miền và có thể tính toán một số thuộc tính như diện tích hay mô men quán tính của miền đó. Ta cũng có thể tạo các hình dạng khác nhau, sau đó chọn các đối tượng đó để tạo miền.

Để tạo miền, ta sử dụng phương thức AddRegion. Phương thức này sẽ tạo ra một miền mới ứng với mỗi vòng kín có trong mảng đường cong đầu vào. AutoCAD sẽ chuyển các đối tượng Polyline 2D hoặc 3D đồng phẳng thành các miền riêng biệt, sau đó sẽ chuyển các đối tượng Polyline, Line và Curve tạo thành vòng kín đồng phẳng. Nếu có nhiều hơn hai đường cong sử dụng cùng một điểm đầu thì sẽ tạo miền có hình dạng bất kỳ. Chính vì lý do này mà một số miền chỉ được thực sự tạo ra khi sử dụng phương thức AddRegion. Nên sử dụng biến variant để lưu mảng miền vừa mới tạo được.

Để xác định tổng số đối tượng Region đã tạo được, ta sử dụng hai hàm VBA là UBound và LBound. Ví dụ sau minh họa cách sử dụng hàm này:

```
UBound(objRegions) - LBound(objRegions) + 1
```

Trong đó biến objRegion là biến variant chứa giá trị trả về khi gọi phương thức AddRegion. Phương thức này sẽ tính tổng số miền đã được tạo ra.

### Tạo một miền đơn giản

Ví dụ sau tạo miền từ một hình tròn.

```
Sub Ch4_CreateRegion()  
    ' Tạo mảng chứa biên của miền.  
    Dim curves(0 To 0) As AcadCircle  
    ' Tạo vòng tròn để làm biên cho miền.  
    Dim center(0 To 2) As Double  
    Dim radius As Double  
    center(0) = 2  
    center(1) = 2  
    center(2) = 0  
    radius = 5#  
    Set curves(0) = ThisDrawing.ModelSpace.AddCircle _  
        (center, radius)  
    ' Tạo miền  
    Dim regionObj As Variant  
    regionObj = ThisDrawing.ModelSpace.AddRegion(curves)  
    ZoomAll  
End Sub
```

### 1.6.1. Tạo miền phức hợp

Ta có thể tạo miền phức hợp bằng cách trừ, nối hoặc tìm giao của các miền hoặc đối tượng 3DSolid. Sau đó, ta có thể tiến hành đập nổi hoặc đập nổi theo đường sinh để tạo nên các đối tượng đặc phức tạp hơn. Để tạo miền phức hợp, ta sử dụng phương thức Boolean.

Nếu muốn trừ một miền ra khỏi một miền khác, ta gọi phương thức Boolean từ miền đầu tiên, là miền mà ta muốn bị trừ đi một phần. Ví dụ như để tính diện tích rải thảm cho sàn nhà, ta tiến hành gọi phương thức Boolean từ đối tượng chứa đường bao ngoài của sàn nhà và sử dụng các diện tích không cần rải thảm làm tham số cho phương thức Boolean, chẳng hạn như diện tích cột quây hàng.

#### Tạo miền phức hợp

```
Sub Ch4_CreateCompositeRegions ()
    ' Tạo hai vòng tròn, một biểu diễn sàn nhà,
    ' một biểu diễn cột nhà ở giữa sàn nhà.
    Dim RoomObjects(0 To 1) As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 4
    center(1) = 4
    center(2) = 0
    radius = 2#
    Set RoomObjects(0) = ThisDrawing.ModelSpace. _
        AddCircle(center, radius)
    radius = 1#
    Set RoomObjects(1) = ThisDrawing.ModelSpace. _
        AddCircle(center, radius)
    ' Tạo vùng mới từ hai vòng tròn
    Dim regions As Variant
    regions = ThisDrawing.ModelSpace.AddRegion(RoomObjects)
    ' Sao chép miền vào các biến chứa miền để dễ sử dụng
    Dim RoundRoomObj As AcadRegion
    Dim PillarObj As AcadRegion
    If regions(0).Area > regions(1).Area Then
        ' The first region is the room
        Set RoundRoomObj = regions(0)
        Set PillarObj = regions(1)
    Else
        ' The first region is the pillar
        Set PillarObj = regions(0)
        Set RoundRoomObj = regions(1)
    End If
    ' Đặt màu phòng là đỏ và cột là xanh lục lam
    RoundRoomObj.Color = acRed
    PillarObj.Color = acCyan
    ZoomAll
    ' Trừ diện tích cột khỏi diện tích sàn
    ' để xác định diện tích rải thảm thực
    RoundRoomObj.Boolean acSubtraction, PillarObj
    ' Sử dụng thuộc tính Area để tính diện tích rải thảm
    MsgBox "The carpet area is: " & RoundRoomObj.Area
End Sub
```

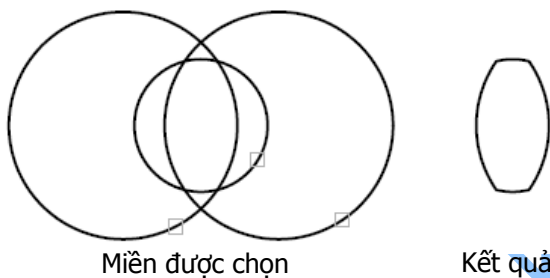
Tính diện tích của miền được tạo thành thông qua thuộc tính Area.

Để nối các miền lại với nhau, ta gọi phương thức Boolean và sử dụng hằng số `acUnion` thay vì sử dụng hằng số `acSubtraction`. Để tìm phần giao của hai miền, ta sử dụng hằng số `acIntersection`.

Ví dụ sau minh họa cách thức nối hai miền với nhau:



Ví dụ sau minh họa phần giao của 3 miền:



## 1.7. Tạo vùng tô mẫu

Vùng tô mẫu là vùng đặc biệt trong bản vẽ được tô đặc theo các mẫu có sẵn.

Khi muốn tạo vùng tô mẫu, ta không cần phải xác định trước vùng cần tô. Trước hết, cần phải tạo đối tượng Hatch. Sau đó ta sẽ xác định vòng khép kín, là đường biên ngoài cùng nhất của vùng tô mẫu. Sau đó ta có thể tiếp tục chọn bất kỳ vòng khép kín nào có trong vùng tô mẫu.

### 1.7.1. Tạo đối tượng Hatch

Khi tạo đối tượng Hatch, cần phải xác định loại mẫu tô, tên mẫu tô và thuộc tính liên kết với biên<sup>1</sup>. Khi đã tạo xong đối tượng Hatch, ta không thể thay đổi thuộc tính liên kết được nữa.

Để tạo đối tượng Hatch, ta sử dụng phương thức `AddHatch`.

### 1.7.2. Liên kết vùng tô mẫu

Ta có thể tạo vùng tô mẫu liên kết hoặc không liên kết. Vùng tô mẫu liên kết luôn được liên kết với đường biên và được cập nhật mỗi khi thay đổi đường biên. Còn vùng tô mẫu không liên kết thì lại không phụ thuộc vào đường biên.

Thuộc tính liên kết chỉ được thiết lập sau khi đã tạo vùng tô mẫu. Và một khi vùng tô mẫu được tạo rồi, ta có thể bỏ kích hoạt thuộc tính liên kết, nhưng không thể kích hoạt liên kết lại một lần nữa.

<sup>1</sup> Thuộc tính này cho phép vùng tô có thay đổi hay không khi đường biên của nó bị thay đổi.

Để tạo vùng tô mẫu liên kết, ta gán tham số *Associativity* trong phương thức *AddHatch* là *TRUE*. Để tạo vùng tô mẫu không liên kết, ta gán tham số *Associativity* trong phương thức *AddHatch* là *FALSE*.

### 1.7.3. Gán kiểu và tên mẫu tô

AutoCAD cung cấp mẫu tô đặc và hơn 50 mẫu tô theo chuẩn khác. Các loại mẫu tô dùng để làm nổi bật một yếu tố hoặc một vùng nào đó trong bản vẽ. Ví dụ như mẫu tô có thể giúp phân biệt các phần của một đối tượng 3D hoặc thể hiện các loại vật liệu cấu thành nên một đối tượng nào đó.

Ta có thể sử dụng mẫu tô có trong AutoCAD hoặc từ một mẫu tô ở thư viện ngoài. Để xem các mẫu tô có trong AutoCAD, xem thêm phần phụ lục E “*Standard Libraries*” trong tài liệu “*AutoCAD Command Reference*”.

Để xác định một mẫu tô, cần phải nhập vào loại và tên mẫu tô khi tạo đối tượng *Hatch*. Loại mẫu tô xác định nơi sẽ tra tên mẫu tô. Khi nhập loại mẫu tô, cần sử dụng một trong những hằng số sau:

*acHatchPatternTypePredefined*

Chọn tên mẫu tô định nghĩa trong tệp *acad.pat*.

*acHatchPatternTypeUserDefined*

Xác định mẫu đường thẳng sử dụng kiểu đường hiện hành.

*acHatchPatternTypeCustomDefined*

Chọn tên mẫu tô từ tệp *.PAT* khác.

Khi nhập vào tên mẫu tô, cần phải sử dụng tên có trong loại mẫu tô đã xác định trước.

### 1.7.4. Xác định đường biên vùng tô mẫu

Sau khi đã tạo đối tượng *Hatch*, ta có thể thêm vào đường biên vùng tô mẫu. Đường biên có thể là sự kết hợp của một trong các đối tượng sau: *Line*, *Arc*, *Circle*, *2DPolyline*, *Ellipse*, *Spline* và *Region*.



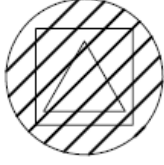
Đường bao ngoài đầu tiên được thêm vào phải là đường bao ngoài cùng nhất, xác định giới hạn ngoài cùng của vùng được tô mẫu. Để thêm vào đường biên ngoài cùng, ta sử dụng phương thức *AppendOuterLoop*.

Khi đã xác định đường biên ngoài cùng, ta có thể tiếp tục thêm vào các đường biên trong sử dụng phương thức *AppendInnerLoop*.

Đường biên trong xác định vùng cô lập bên trong vùng tô mẫu. Cách thức xử lý vùng cô lập trong đối tượng *Hatch* phụ thuộc vào thuộc tính *HatchStyle*. Thuộc tính *HatchStyle* được thiết lập theo một trong những điều kiện sau:



## Định nghĩa các loại HatchStyle

Loại HatchStyle	Điều kiện	Mô tả
	Normal	Đây là loại HatchStyle chuẩn. Lựa chọn này thực hiện tô mẫu từ bên ngoài vào bên trong. Nếu AutoCAD gặp một đường biên trong thì sẽ tắt lựa chọn tô mẫu cho đến khi gặp một đường biên trong khác nữa thì lựa chọn tô mẫu sẽ được bật lại. Đây là thiết lập mặc định cho thuộc tính HatchStyle.
	Outer	Chỉ tô mẫu phần diện tích ngoài cùng nhất. Lựa chọn này cũng tiến hành tô mẫu từ bên ngoài vào, nhưng khi gặp một đường biên trong lựa chọn tô mẫu sẽ tắt và không bật lại nữa.
	Ignore	Bỏ qua tất cả các đường biên trong.

Sau khi đã được định nghĩa, vùng tô mẫu cần phải được đánh giá trước khi hiển thị sử dụng phương thức Evaluate.

### 1.7.5. Tạo đối tượng Hatch

Ví dụ sau tạo một vùng tô mẫu liên kết trong không gian mô hình. Sau khi đã tạo vùng tô mẫu, ta tiến hành thay đổi kích thước của vòng tròn liên kết với vùng tô mẫu đó, khi đó vùng tô mẫu sẽ tự động thay đổi cho phù hợp với kích thước vòng tròn hiện tại.

```
Sub Ch4_CreateHatch()  
    Dim hatchObj As AcadHatch  
    Dim patternName As String  
    Dim PatternType As Long  
    Dim bAssociativity As Boolean  
    ' Định nghĩa vùng tô mẫu  
    patternName = "ANSI31"  
    PatternType = 0  
    bAssociativity = True  
  
    ' Tạo đối tượng vùng tô mẫu liên kết  
    Set hatchObj = ThisDrawing.ModelSpace.AddHatch _  
        (PatternType, patternName, bAssociativity)  
  
    ' Tạo đường biên ngoài là vòng tròn  
    Dim outerLoop(0 To 0) As AcadEntity  
    Dim center(0 To 2) As Double  
    Dim radius As Double  
    center(0) = 3: center(1) = 3: center(2) = 0  
    radius = 1  
    Set outerLoop(0) = ThisDrawing.ModelSpace. _
```

```

AddCircle(center, radius)

' Thêm đường biên ngoài và hiển thị vùng tô mẫu
hatchObj.AppendOuterLoop (outerLoop)
hatchObj.Evaluate
ThisDrawing.Regen True
End Sub

```

## 2. Hiệu chỉnh đối tượng

Để hiệu chỉnh một đối tượng, ta sử dụng các phương thức và thuộc tính gắn liền với đối tượng đó. Nếu tiến hành hiệu chỉnh các thuộc tính trực quan của đối tượng đồ họa thì cần phải sử dụng phương thức Update để vẽ lại các đối tượng trên màn hình.

Phần này sẽ mô tả cách hiệu chỉnh các đối tượng 2D.

### 2.1. Hiệu chỉnh các đối tượng phi đồ họa

Bên cạnh các đối tượng đồ họa, trong AutoCAD còn có các đối tượng phi đồ họa lưu trữ trong tệp bản vẽ. Những đối tượng này có tác dụng mô tả ứng với chức năng của đối tượng, chẳng hạn như các đối tượng Block, Layer, Group, và DimensionStyle. Trong hầu hết các trường hợp, ta thường đặt tên khi tạo mới các đối tượng này và sau đó có thể đổi tên đối tượng. Tên sẽ được lưu trong bảng ký hiệu. Khi đã xác định tên của đối tượng, ta có thể tham chiếu đến đối tượng và các dữ liệu liên quan của đối tượng trong bảng ký hiệu.

#### 2.1.1. Thanh lọc đối tượng phi đồ họa

Ta có thể thanh lọc các đối tượng khi chúng không sử dụng hoặc không được tham chiếu trong bản vẽ bất cứ lúc nào trong quá trình soạn thảo bản vẽ. Quá trình thanh lọc sẽ làm giảm kích thước của bản vẽ. Không thể thanh lọc các đối tượng đang được tham chiếu bởi các đối tượng khác, chẳng hạn như các tệp chứa phông có thể được tham chiếu trong kiểu chữ, lớp được tham chiếu bởi các đối tượng trong lớp đó.

Ta sử dụng phương thức PurgeAll để thanh lọc bản vẽ.

#### Thanh lọc bản vẽ

```
ThisDrawing.PurgeAll
```

#### 2.1.2. Đổi tên đối tượng

Khi bản vẽ ngày càng trở nên phức tạp, cần phải đổi tên các đối tượng để giữ ý nghĩa của tên đó hoặc tránh tranh chấp tên đối tượng với các bản vẽ khác vừa mới được chèn trong bản vẽ chính.

Ta có thể thay đổi tên của bất kỳ đối tượng nào ngoại trừ tên mặc định của AutoCAD, chẳng hạn như *layer 0* hoặc kiểu đường *CONTINUOUS*.

Tên có thể dài khoảng 255 ký tự. Ngoài chữ và chữ số, tên có thể chứa các khoảng trắng (AutoCAD sẽ xóa các khoảng trắng xuất hiện ở ngay phía trước và phía sau tên) và các ký tự đặc biệt không sử dụng trong Microsoft Windows và AutoCAD với mục đích khác. Các ký tự đặc biệt không sử dụng được bao gồm dấu lớn hơn, bé hơn (< >), dấu gạch chéo (/ \), dấu ngoặc kép ("), dấu hai chấm (:), dấu chấm

phẩy (;), dấu chấm hỏi (?), dấu phẩy (,), dấu sao (\*), dấu gạch đứng (|), dấu bằng (=), dấu nháy ('). Ngoài ra, ta cũng không được sử dụng các ký tự đặc biệt của phông chữ Unicode.

Sử dụng thuộc tính Name để đổi tên một đối tượng.

### Đổi tên lớp

Ví dụ sau tạo một lớp có tên là “NewLayer” và sau đó đổi tên thành “MyLayer”.

```
Sub Ch4_RenamingLayer()  
    ' Tạo mới lớp  
    Dim layerObj As AcadLayer  
    Set layerObj = ThisDrawing.Layers.Add("NewLayer")  
    ' Đổi tên lớp  
    layerObj.Name = "MyLayer"  
End Sub
```

## 2.2. Chọn đối tượng

Tập lựa chọn là một nhóm đối tượng AutoCAD được xử lý như một đối tượng đơn nhất. Tập lựa chọn có thể chỉ chứa một đối tượng, hoặc có thể chứa nhóm đối tượng phức tạp hơn, chẳng hạn như tập các đối tượng có cùng một màu hoặc thuộc cùng một lớp.

Quá trình định nghĩa tập lựa chọn gồm hai bước. Thứ nhất, phải tạo một tập lựa chọn mới và sau đó thêm vào tập đối tượng SelectionSets. Sau đó ta có thể thao tác trên tập lựa chọn với các đối tượng mà ta cần xử lý.

### 2.2.1. Tạo tập lựa chọn

Để tạo tập lựa chọn, ta sử dụng phương thức Add. Phương thức này chỉ cần một tham số là tên của tập lựa chọn.

#### Tạo một tập lựa chọn rỗng

Ví dụ sau tạo một tập lựa chọn mới.

```
Sub Ch4_CreateSelectionSet()  
    Dim selectionSet1 As AcadSelectionSet  
    Set selectionSet1 = ThisDrawing.SelectionSets. _  
        Add("NewSelectionSet")  
End Sub
```

### 2.2.2. Thêm đối tượng vào tập lựa chọn

Ta có thể thêm các đối tượng vào trong tập lựa chọn hiện hành bằng cách sử dụng một trong những phương thức sau:

**AddItems**            Thêm một hoặc nhiều đối tượng vào trong một tập lựa chọn được chỉ định

**Select**                Lựa chọn các đối tượng và thêm vào trong tập lựa chọn hiện hành. Ta có thể chọn tất cả các đối tượng, các đối tượng bên trong và cắt ngang qua một hình chữ nhật, các đối tượng bên trong và cắt ngang qua hình đa giác, tất cả các đối tượng cắt ngang qua hàng rào, đối tượng vừa mới tạo, đối tượng trong tập lựa chọn vừa mới

tao, đối tượng bên trong một cửa sổ, đối tượng bên trong hình đa giác.

- SelectAtPoint Chọn các đối tượng đi qua một điểm và thêm vào tập lựa chọn hiện hành.
- SelectByPolygon Chọn các đối tượng bên trong hàng rào và thêm vào tập lựa chọn hiện hành.
- SelectOnScreen Nhắc người dùng chọn đối tượng trên màn hình và thêm vào tập lựa chọn hiện hành.

### Thêm đối tượng vào tập lựa chọn

Ví dụ sau nhắc người dùng chọn đối tượng, sau đó thêm các đối tượng được chọn vào tập lựa chọn. Cuối cùng tiến hành thay đổi màu của đối tượng trong tập lựa chọn sang màu xanh.

```
Sub Ch4_AddToASelectionSet()  
    ' Tạo tập đối tượng mới  
    Dim sset As AcadSelectionSet  
    Set sset = ThisDrawing.SelectionSets.Add("SS1")  
    ' Nhắc người dùng chọn đối tượng để thêm vào tập lựa chọn  
    ' Để kết thúc việc chọn đối tượng, nhấn phím ENTER.  
    sset.SelectOnScreen  
    ' Duyệt qua tập lựa chọn và gán màu đối tượng là màu xanh  
    Dim entry As AcadEntity  
    For Each entry In sset  
        entry.Color = acBlue  
        entry.Update  
    Next entry  
End Sub
```

### 2.2.3. Lọc tập lựa chọn

Ta có thể giới hạn tập lựa chọn bằng các thuộc tính, chẳng hạn như màu sắc hoặc bằng loại đối tượng sử dụng bộ lọc. Ví dụ như ta chỉ cần sao chép những đối tượng màu đỏ trong bản mạch điện hoặc các đối tượng thuộc một lớp nào đó.

---

**CHÚ Ý:** Bộ lọc chỉ nhận diện màu và kiểu đường của chính đối tượng chứ không phải là màu và kiểu đường gán cho lớp chứa đối tượng đó.

---

Để sử dụng được cơ cấu lọc, cần phải xác định kiểu bộ lọc và dữ liệu được lọc. Kiểu bộ lọc là một loại mã dùng để xác định xem bộ lọc nào sẽ được sử dụng. AutoCAD ActiveX Automation sử dụng nhóm mã DXF để xác định kiểu bộ lọc. Danh sách dưới đây thể hiện một số kiểu bộ lọc thông dụng, danh sách đầy đủ tham khảo thêm trong tài liệu “*AutoCAD DXF Reference*”.

## Mã DXF của một số bộ lọc thông dụng

Mã DXF	Kiểu bộ lọc
0	Loại đối tượng (kiểu String) Chẳng hạn như "Line", "Circle", "Arc",...
2	Tên đối tượng (kiểu String) Bảng tên của named object.
8	Tên lớp (kiểu String) Ví dụ như "Layer 0".
60	Tính nhìn thấy của đối tượng (kiểu Integer) Sử dụng 0=nhìn thấy (visible), 1=ẩn (invisible)
62	Số màu (kiểu Integer) Giá trị chỉ số màu biến thiên từ 0 đến 256. Số 0 nghĩa là màu BYBLOCK (theo màu khối). Số 256 nghĩa là màu BYLAYER (theo màu lớp). Giá trị âm có nghĩa là lớp đã bị tắt.
67	Trong không gian mô hình/không gian in (kiểu Integer) Sử dụng 0 hoặc bỏ qua = không gian mô hình, 1= không gian in.

Ví dụ sau minh họa nhiều bộ lọc khác nhau.

Chỉ thêm những đối tượng kiểu Text vào tập lựa chọn:

```
FilterType = 0  
FilterData = "TEXT"  
sset.SelectOnScreen FilterType, FilterData
```

Chỉ thêm những đối tượng kiểu Line vào tập lựa chọn:

```
FilterType = 0  
FilterData = "LINE"  
sset.SelectOnScreen FilterType, FilterData
```

Chỉ thêm những đối tượng thuộc lớp FLOOR9 vào tập lựa chọn:

```
FilterType = 8  
FilterData = "FLOOR9"  
sset.SelectOnScreen FilterType, FilterData
```

Chỉ thêm những đối tượng có màu đỏ vào tập lựa chọn:

```
Filter Type = 62  
Filter Data = 5  
sset.SelectOnScreen FilterType, FilterData
```

#### 2.2.4. Gỡ bỏ đối tượng ra khỏi tập lựa chọn

Sau khi tạo tập lựa chọn, ta có thể gỡ một số đối tượng hoặc tất cả các đối tượng ra khỏi tập lựa chọn đó. Chẳng hạn ta có thể chọn toàn bộ một nhóm đối tượng dày đặc và gỡ bỏ một số đối tượng nào đó chỉ để lại những đối tượng mà ta cần thêm vào trong tập lựa chọn.

Sử dụng các phương thức sau để gỡ bỏ đối tượng ra khỏi tập lựa chọn:

**RemoveItems** Phương thức `RemoveItems` gỡ bỏ một hoặc một vài đối tượng khỏi tập lựa chọn. Các đối tượng bị gỡ bỏ vẫn còn tồn tại nhưng chúng không nằm trong tập lựa chọn nữa.

**Clear** Phương thức `Clear` sẽ làm rỗng tập lựa chọn. Tập lựa chọn vẫn tồn tại nhưng không chứa bất kỳ đối tượng nào. Các đối tượng đã từng ở trong tập lựa chọn vẫn còn tồn tại nhưng không còn nằm trong tập lựa chọn nữa.

**Erase** Phương thức `Erase` sẽ xóa tất cả các đối tượng có trong tập lựa chọn. Tập lựa chọn vẫn còn tồn tại nhưng không chứa bất kỳ đối tượng nào. Các đối tượng đã từng nằm trong tập lựa chọn sẽ không còn tồn tại nữa.

**Delete** Phương thức `Delete` xóa tập lựa chọn và tất cả các đối tượng nằm trong tập lựa chọn đó. Tất cả các đối tượng trong tập lựa chọn và cả bản thân tập lựa chọn sẽ không còn tồn tại nữa sau khi gọi phương thức `Delete`.

### 2.3. Sao chép đối tượng

Ta có thể sao chép một hoặc nhiều đối tượng bên trong bản vẽ hiện hành. Lệnh `Offset` tạo đối tượng mới nằm cách đối tượng đã chọn một khoảng nhất định. Lệnh `Mirror` tạo ảnh đối xứng của đối tượng qua một trục. Lệnh `Array` tạo hàng loạt bản sao theo mẫu hình chữ nhật hoặc theo hình tròn.

---

**CHÚ Ý:** Không thể thực hiện các phương thức trình bày ở dưới đây khi đồng thời duyệt qua tập đối tượng. Phép duyệt sẽ thực hiện các thao tác chỉ đọc trong khi những phương thức này lại thực hiện các thao tác đọc-ghi. Cần phải thực hiện phép duyệt qua tập đối tượng trước khi gọi những phương thức này.

---

#### 2.3.1. Sao chép đối tượng ở cùng một vị trí

Để sao chép một đối tượng, ta sử dụng phương thức `Copy` của chính đối tượng đó. Phương thức này tạo một bản sao của đối tượng. Đối tượng trả về của phương thức là đối tượng mới được tạo thành, có vị trí trùng với vị trí của đối tượng gốc.

#### 2.3.2. Sao chép nhiều đối tượng hoặc sao chép vào bản vẽ khác

Để sao chép nhiều đối tượng, ta sử dụng phương thức `CopyObjects` hoặc tạo mảng đối tượng để sau đó sử dụng phương thức `Copy`. Để sao chép các đối tượng trong một tập lựa chọn, ta duyệt qua tập lựa chọn và lưu các đối tượng trong một mảng. Sau đó duyệt qua mảng này và sao chép từng đối tượng một và lưu vào một mảng thứ hai.

## Sao chép nhiều đối tượng

Ví dụ sau tạo hai vòng tròn và sử dụng phương thức CopyObjects để tạo bản sao của các vòng tròn.

```
Sub Ch4_CopyCircleObjects()  
    Dim ACADApp As AcadApplication  
    Dim DOC1 As AcadDocument  
    Dim circleObj1 As AcadCircle  
    Dim circleObj2 As AcadCircle  
    Dim circleObj1Copy As AcadCircle  
    Dim circleObj2Copy As AcadCircle  
    Dim centerPoint(0 To 2) As Double  
    Dim radius1 As Double  
    Dim radius2 As Double  
    Dim radius1Copy As Double  
    Dim radius2Copy As Double  
    Dim objCollection(0 To 1) As Object  
    Dim retObjects As Variant  
  
    ' Định nghĩa vòng tròn  
    centerPoint(0) = 0: centerPoint(1) = 0: centerPoint(2) = 0  
    radius1 = 5#: radius2 = 7#  
    radius1Copy = 1#: radius2Copy = 2#  
  
    ' Lấy đối tượng Application từ hệ thống  
    Set ACADApp = GetObject(, "AutoCAD.Application")  
  
    ' Tạo bản vẽ mới  
    Set DOC1 = ACADApp.Documents.Add  
  
    ' Thêm hai vòng tròn vào bản vẽ  
    Set circleObj1 = DOC1.ModelSpace.AddCircle _  
        (centerPoint, radius1)  
    Set circleObj2 = DOC1.ModelSpace.AddCircle _  
        (centerPoint, radius2)  
    ZoomAll  
  
    ' Chuyển sang dạng thích hợp với phương thức CopyObjects  
    Set objCollection(0) = circleObj1  
    Set objCollection(1) = circleObj2  
  
    ' Sao chép đối tượng và lấy lại mảng đối tượng mới  
    retObjects = DOC1.CopyObjects(objCollection)  
  
    ' Lấy đối tượng mới tạo và gán các thuộc tính mới  
    Set circleObj1Copy = retObjects(0)  
    Set circleObj2Copy = retObjects(1)  
    circleObj1Copy.radius = radius1Copy  
    circleObj1Copy.Color = acRed  
    circleObj2Copy.radius = radius2Copy  
    circleObj2Copy.Color = acRed  
    ZoomAll  
End Sub
```

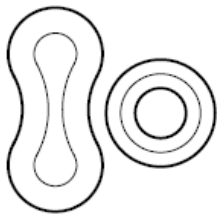


### 2.3.3. Offset đối tượng

Offset một đối tượng sẽ tạo mới một đối tượng đồng dạng và nằm cách một khoảng so với đối tượng gốc. Ta có thể Offset các đối tượng như Arc, Circle, Ellipse, Line, Lightweight polyline, Polyline, Spline và Xline.

Để Offset một đối tượng, ta sử dụng phương thức Offset có trong mỗi đối tượng đó.

Tham số đầu vào duy nhất là khoảng cách để Offset đối tượng. Nếu khoảng cách này có giá trị âm, AutoCAD sẽ xử lý theo cách tương tự như tạo ra có đường cong “nhỏ hơn” (nghĩa là, đối với cung tròn sẽ Offset về phía có bán kính nhỏ hơn). Nếu “nhỏ hơn” không có ý nghĩa thì AutoCAD sẽ Offset về phía có tọa độ X, Y, Z nhỏ hơn. Nếu khoảng cách Offset không hợp lệ thì sẽ phát sinh lỗi (chứa trong giá trị trả về).



Offset đường đa tuyến

Đối với nhiều đối tượng, kết quả của thao tác này sẽ tạo thành một đường cong mới (có thể không cùng loại với đường cong ban đầu). Ví dụ như khi offset một đường Ellipse sẽ tạo thành một đường Spline đó là do đối tượng kết quả không thỏa mãn phương trình của đường Ellipse nên AutoCAD sẽ tự chuyển đổi sang một dạng đối tượng khác thích hợp. Trong một số trường hợp thì kết quả của phương thức Offset có thể lại là một vài đường cong. Chính vì lẽ này, phương thức Offset sẽ trả về một đối tượng mới hoặc mảng đối tượng chứa trong biến kiểu variant.

#### Offset đường đa tuyến

Ví dụ sau tạo một đường Lightweight polyline và sau đó Offset đối tượng Polyline. Đường Polyline mới tạo có màu đỏ.

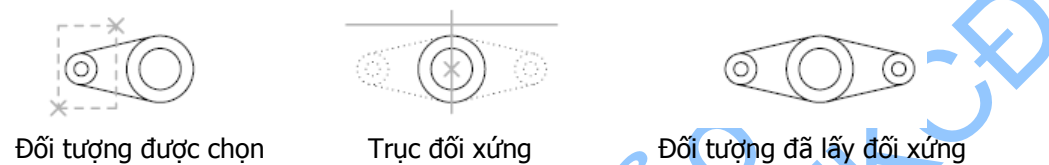
```
Sub Ch4_OffsetPolyline()  
    ' Tạo đường Polyline  
    Dim plineObj As AcadLWPolyline  
    Dim points(0 To 11) As Double  
    points(0) = 1: points(1) = 1  
    points(2) = 1: points(3) = 2  
    points(4) = 2: points(5) = 2  
    points(6) = 3: points(7) = 2  
    points(8) = 4: points(9) = 4  
    points(10) = 4: points(11) = 1  
    Set plineObj = ThisDrawing.ModelSpace. _  
        AddLightWeightPolyline(points)  
    plineObj.Closed = True  
    ZoomAll  
    ' Offset đường polyline  
    Dim offsetObj As Variant  
    offsetObj = plineObj.Offset(0.25)  
    offsetObj(0).Color = acRed  
    ZoomAll  
End Sub
```

### 2.3.4. Lấy đối xứng đối tượng

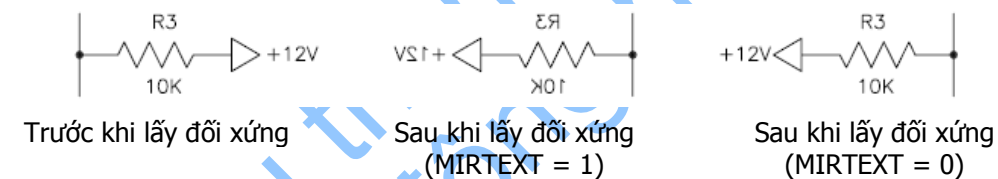
Lệnh Mirror tạo một bản sao đối xứng của đối tượng gốc quanh một đường trục bất kỳ. Ta có thể lấy đối xứng tất cả các đối tượng trong bản vẽ.

Để lấy đối xứng, ta sử dụng phương thức Mirror có trong mỗi đối tượng. Không giống như lệnh Mirror trong AutoCAD, phương thức này tạo ảnh đối xứng đối tượng và giữ nguyên đối tượng gốc (Để xóa đối tượng gốc, ta sử dụng phương thức Erase).

Phương thức này cần tham số đầu vào là hai giá trị tọa độ. Hai giá trị tọa độ này xác định điểm đầu và điểm cuối của trục đối xứng. Trong không gian 3D, đường thẳng này định hướng mặt phẳng đối xứng vuông góc với mặt phẳng XY của hệ tọa độ người dùng (UCS).



Để quản lý tính đối xứng của đối tượng kiểu Text, ta sử dụng biến hệ thống MIRRTEXT. Mặc định, biến MIRRTEXT có giá trị là 1 (bật), có nghĩa là đối tượng kiểu Text vẫn được lấy đối xứng như các đối tượng khác. Khi biến MIRRTEXT có giá trị 0 (tắt), các đối tượng kiểu Text sẽ không được lấy đối xứng. Ta sử dụng phương thức GetVariable và SetVariable để lấy và gán giá trị cho biến MIRRTEXT.



Ta cũng có thể lấy đối xứng đối tượng Viewport trong không gian in và thao tác này không ảnh hưởng đến các đối tượng và cảnh nhìn trong không gian mô hình.

### Lấy đối xứng đối tượng Polyline quanh một trục

Ví dụ sau tạo một đối tượng Lineweight polyline và lấy đối xứng qua một trục. Đối tượng mới tạo thành sẽ được gán màu đỏ.

```
Sub Ch4_MirrorPolyline()  
    ' Tạo đối tượng polyline  
    Dim plineObj As AcadLWPolyline  
    Dim points(0 To 11) As Double  
    points(0) = 1: points(1) = 1  
    points(2) = 1: points(3) = 2  
    points(4) = 2: points(5) = 2  
    points(6) = 3: points(7) = 2  
    points(8) = 4: points(9) = 4  
    points(10) = 4: points(11) = 1  
    Set plineObj = ThisDrawing.ModelSpace. _  
    AddLightWeightPolyline(points)  
    plineObj.Closed = True  
    ZoomAll  
    ' Xác định trục đối xứng  
    Dim point1(0 To 2) As Double
```

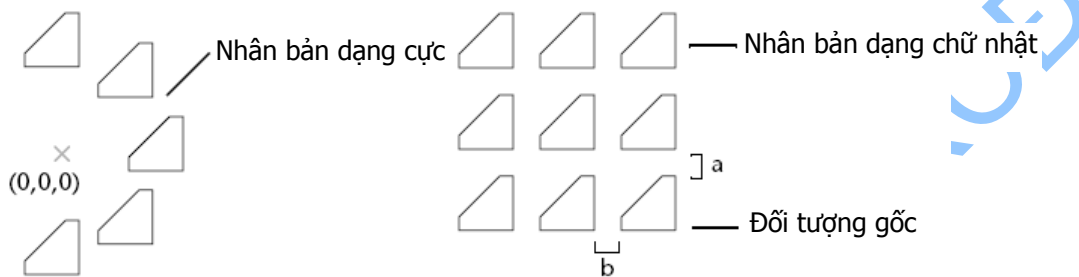
```

Dim point2(0 To 2) As Double
point1(0) = 0: point1(1) = 4.25: point1(2) = 0
point2(0) = 4: point2(1) = 4.25: point2(2) = 0
' Lấy đối xứng đối tượng polyline
Dim mirrorObj As AcadLWPolyline
Set mirrorObj = plineObj.Mirror(point1, point2)
mirrorObj.Color = acRed
ZoomAll
End Sub

```

### 2.3.5. Nhân bản đối tượng

Ta có thể nhân bản đối tượng theo dạng cực hoặc dạng chữ nhật. Với dạng cực, cần nắm được số lượng cần nhân bản, và giá trị góc để nhân đối tượng. Với dạng chữ nhật, thì phải nắm được số lượng và khoảng cách giữa các hàng, cột.



#### 2.3.5.1. Nhân bản dạng cực

Ta có thể nhân bản tất cả các loại đối tượng trong bản vẽ. Để nhân bản dạng cực, ta sử dụng phương thức `ArrayPolar` có trong đối tượng. Phương thức này cần phải nhập vào 3 thông số: số đối tượng cần tạo, góc để nhân bản và điểm tâm. Số đối tượng phải là số nguyên dương lớn hơn 1. Góc nhân bản phải có đơn vị là radian, với giá trị dương nghĩa là nhân bản theo chiều ngược chiều kim đồng hồ, còn giá trị âm là theo chiều kim đồng hồ, còn nếu giá trị góc bằng 0 thì chương trình sẽ phát sinh lỗi. Điểm tâm là một mảng chứa 3 số kiểu double, thể hiện tọa độ 3D trong hệ tọa độ toàn cục (WCS) của điểm tâm.

AutoCAD sẽ xác định khoảng cách từ điểm tâm nhân bản đến điểm tham chiếu của đối tượng gốc. Điểm tham chiếu được sử dụng phụ thuộc vào loại đối tượng. AutoCAD sẽ sử dụng tâm của hình tròn hoặc cung tròn, điểm chèn của một khối, điểm bắt đầu của đối tượng Text, một đầu của đường thẳng hoặc đối tượng **Trace**.

Phương thức này không hỗ trợ lựa chọn Rotate While Copying (xoay khi đang nhân bản) của lệnh `ARRAY` trong AutoCAD.

#### Nhân bản dạng cực

Ví dụ sau tạo một vòng tròn, sau đó thực hiện nhân bản dạng cực để tạo thành 4 đối tượng với góc nhân bản là 180 độ xung quanh điểm tâm là (4, 4, 0).

```

Sub Ch4_ArrayingACircle()
' Tạo vòng tròn
Dim circleObj As AcadCircle
Dim center(0 To 2) As Double
Dim radius As Double
center(0) = 2#: center(1) = 2#: center(2) = 0#
radius = 1

```

```

Set circleObj = ThisDrawing.ModelSpace. _
    AddCircle(center, radius)
ZoomAll
' Định nghĩa tham số nhân bản dạng cực
Dim noOfObjects As Integer
Dim angleToFill As Double
Dim basePnt(0 To 2) As Double
noOfObjects = 4
angleToFill = 3.14 ' 180 degrees
basePnt(0) = 4#: basePnt(1) = 4#: basePnt(2) = 0#
' Thực hiện nhân bản đối tượng
Dim retObj As Variant
retObj = circleObj.ArrayPolar _
    (noOfObjects, angleToFill, basePnt)
ZoomAll
End Sub

```

### 2.3.5.2. Nhân bản dạng chữ nhật

Để nhân bản đối tượng theo dạng chữ nhật 2D hoặc 3D, ta sử dụng phương thức `ArrayRectangular` có trong mỗi đối tượng. Phương thức này cần phải nhập vào các tham số: số hàng, số cột, khoảng cách giữa các hàng, khoảng cách giữa các cột. Khi nhân bản 3D cần phải nhập thêm giá trị: số tầng và khoảng cách giữa các tầng.

Nhân bản dạng chữ nhật thực hiện bằng cách sao chép các đối tượng trong tập lựa chọn một số lần thích hợp. Vì vậy nếu ta chỉ có một hàng thì cần phải nhập vào nhiều hơn một cột và ngược lại.

Đối tượng gốc được giả định là ở góc dưới bên trái, các đối tượng nhân bản sẽ được phát sinh về phía góc trên bên phải. Nếu khoảng cách giữa các hàng là số âm thì các hàng sẽ được phát sinh xuống phía dưới, còn nếu khoảng cách giữa các cột là số âm thì các cột sẽ được phát sinh về phía bên trái.

AutoCAD thực hiện nhân bản dạng chữ nhật theo một đường cơ sở xác định nhờ vào góc quay bắt điểm hiện hành. Mặc định góc này có giá trị là 0 nên các hàng và cột sẽ vuông góc tương ứng với trục X và trục Y của bản vẽ. Ta có thể thay đổi giá trị của góc này để nhân bản dạng chữ nhật với góc quay bằng cách thay đổi góc quay bắt điểm bằng một giá trị khác 0. Để làm được điều này, ta sử dụng thuộc tính `SnapRotationAngle`.

#### Nhân bản dạng chữ nhật

Ví dụ sau tạo vòng tròn và thực hiện nhân bản dạng chữ nhật thành 5 hàng và 5 cột.

```

Sub Ch4_ArrayRectangularExample()
' Tạo vòng tròn
Dim circleObj As AcadCircle
Dim center(0 To 2) As Double
Dim radius As Double
center(0) = 2#: center(1) = 2#: center(2) = 0#
radius = 0.5
Set circleObj = ThisDrawing.ModelSpace. _
    AddCircle(center, radius)
ZoomAll
' Xác định các thông số
Dim numberOfRows As Long
Dim numberOfColumns As Long
Dim numberOfLevels As Long

```

```

Dim distanceBwtnRows As Double
Dim distanceBwtnColumns As Double
Dim distanceBwtnLevels As Double
numberOfRows = 5
numberOfColumns = 5
numberOfLevels = 2
distanceBwtnRows = 1
distanceBwtnColumns = 1
distanceBwtnLevels = 1
' Thực hiện nhân bản
Dim retObj As Variant
retObj = circleObj.ArrayRectangular _
(numberOfRows, numberOfColumns, numberOfLevels, _
distanceBwtnRows, distanceBwtnColumns, distanceBwtnLevels)
ZoomAll
End Sub

```

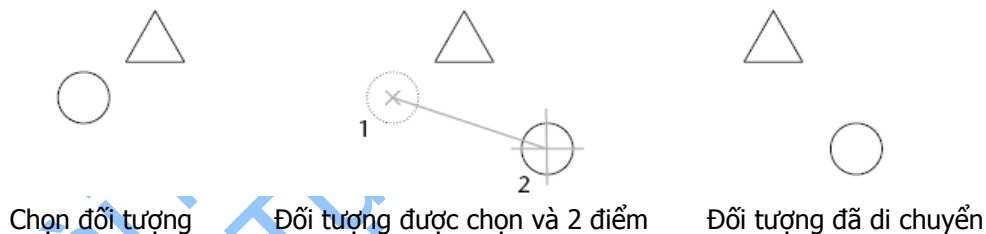
## 2.4. Di chuyển đối tượng

Ta có thể di chuyển đối tượng dọc theo vector nào đấy mà không làm thay đổi chiều và kích thước của đối tượng. Ngoài ra ta cũng có thể xoay đối tượng quanh một điểm cố định.

### 2.4.1. Di chuyển đối tượng

Ta có thể di chuyển tất cả các loại đối tượng trong bản vẽ và các đối tượng tham chiếu có thuộc tính theo một vector định trước.

Để di chuyển đối tượng, ta sử dụng phương thức Move có trong mỗi đối tượng. Phương thức này cần phải nhập vào tọa độ hai điểm. Hai điểm này xác định vector chuyển vị, là vector xác định khoảng cách và hướng di chuyển đối tượng.



### Di chuyển vòng tròn

Ví dụ sau tạo một vòng tròn và di chuyển vòng tròn này 2 đơn vị dọc theo trục X.

```

Sub Ch4_MoveCircle()
' Tạo vòng tròn
Dim circleObj As AcadCircle
Dim center(0 To 2) As Double
Dim radius As Double
center(0) = 2#: center(1) = 2#: center(2) = 0#
radius = 0.5
Set circleObj = ThisDrawing.ModelSpace. _
AddCircle(center, radius)
ZoomAll
' Định nghĩa các điểm của vector chuyển vị.
Dim point1(0 To 2) As Double
Dim point2(0 To 2) As Double
point1(0) = 0: point1(1) = 0: point1(2) = 0

```

```

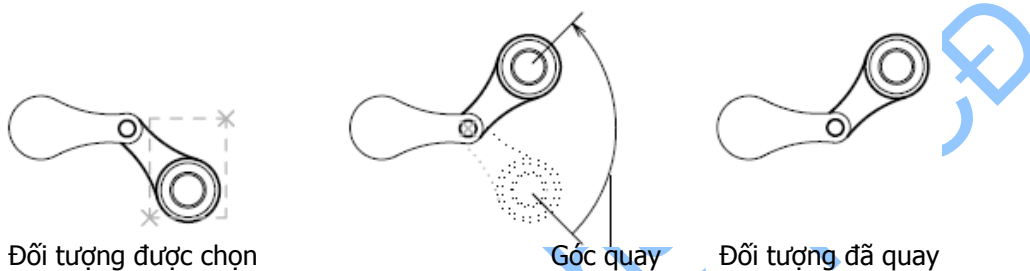
point2(0) = 2: point2(1) = 0: point2(2) = 0
' Di chuyển vòng tròn
circleObj.Move point1, point2
circleObj.Update
End Sub

```

## 2.4.2. Quay đối tượng

Ta có thể quay tất cả các đối tượng trong bản vẽ và các đối tượng tham chiếu có thuộc tính.

Để quay một đối tượng, ta sử dụng phương thức Rotate có trong từng đối tượng. Phương thức này cần nhập vào hai tham số là tâm quay và góc quay. Tâm quay là mảng chứa 3 số kiểu double. Góc quay có giá trị tính bằng radian.



### Quay đối tượng Polyline quanh một điểm

Ví dụ sau tạo đường Polyline khép kín và quay 45 độ quanh điểm tâm (4, 4.25, 0).

```

Sub Ch4_RotatePolyline()
' Tạo đối tượng polyline
Dim plineObj As AcadLWPolyline
Dim points(0 To 11) As Double
points(0) = 1: points(1) = 2
points(2) = 1: points(3) = 3
points(4) = 2: points(5) = 3
points(6) = 3: points(7) = 3
points(8) = 4: points(9) = 4
points(10) = 4: points(11) = 2
Set plineObj = ThisDrawing.ModelSpace. _
AddLightWeightPolyline(points)
plineObj.Closed = True
ZoomAll
' Xác định các tham số
Dim basePoint(0 To 2) As Double
Dim rotationAngle As Double
basePoint(0) = 4: basePoint(1) = 4.25: basePoint(2) = 0
rotationAngle = 0.7853981 ' 45 degrees
' Quay đối tượng polyline
plineObj.Rotate basePoint, rotationAngle
plineObj.Update
End Sub

```

## 2.5. Xóa đối tượng

Ta có thể xóa từng đối tượng bằng cách sử dụng phương thức Delete.

**CHÚ Ý** Các tập đối tượng trong ActiveX Automation đều có phương thức Delete như đã được định nghĩa trong thư viện kiểu. Tuy nhiên, không được xóa một số tập đối tượng như

tập đối tượng ModelSpace, Layers và Dictionaries. Nếu cố xóa các tập đối tượng này thì AutoCAD sẽ phát sinh lỗi.

## Xóa đối tượng

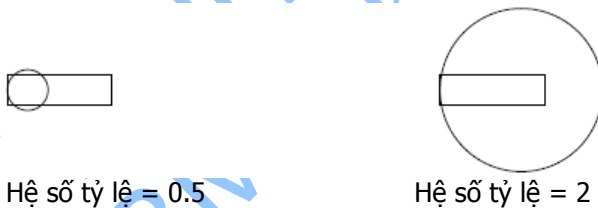
Ví dụ sau tạo đối tượng Lightweight polyline và sau đó xóa đối tượng này.

```
Sub Ch4_DeletePolyline()  
    ' Tạo đối tượng polyline  
    Dim lwpolyObj As AcadLWPolyline  
    Dim vertices(0 To 5) As Double  
    vertices(0) = 2: vertices(1) = 4  
    vertices(2) = 4: vertices(3) = 2  
    vertices(4) = 6: vertices(5) = 4  
    Set lwpolyObj = ThisDrawing.ModelSpace. _  
        AddLightWeightPolyline(vertices)  
    ZoomAll  
    ' Xóa đối tượng polyline  
    lwpolyObj.Delete  
    ThisDrawing.Regen acActiveViewport  
End Sub
```

## 2.6. Co dẫn đối tượng

Ta có thể co dẫn một đối tượng bằng cách sử dụng một điểm cơ sở và chiều dài, dùng để xác định hệ số tỷ lệ dựa trên đơn vị hiện hành của bản vẽ. Ta có thể co dẫn tất cả các đối tượng trong bản vẽ, và cả các đối tượng tham chiếu có thuộc tính.

Để co dẫn một đối tượng, ta sử dụng phương thức ScaleEntity có trong mỗi đối tượng. Phương thức này co dẫn đối tượng bằng nhau theo các phương X, Y và Z; nhận các tham số đầu vào là điểm cơ sở trong quá trình co dẫn và hệ số tỷ lệ. Điểm cơ sở là mảng chứa 3 giá trị kiểu double. Ba số này thể hiện tọa độ 3D trong hệ tọa độ toàn cục xác định điểm bắt đầu co dẫn đối tượng. Hệ số tỷ lệ là hệ số để co dẫn đối tượng. Tất cả các kích thước của đối tượng đều được nhân với hệ số tỷ lệ này. Hệ số tỷ lệ lớn hơn 1 sẽ phóng to đối tượng, còn nếu từ 0 đến 1 thì sẽ thu nhỏ đối tượng. Nếu hệ số tỷ lệ nhỏ hơn hoặc bằng 0, AutoCAD sẽ phát sinh lỗi.



## Co dẫn đối tượng Polyline

Ví dụ sau tạo đường Polyline khép kín và co dẫn đối tượng với hệ số tỷ lệ bằng 0.5.

```
Sub Ch4_ScalePolyline()  
    ' Tạo đối tượng Polyline  
    Dim plineObj As AcadLWPolyline  
    Dim points(0 To 11) As Double  
    points(0) = 1: points(1) = 2  
    points(2) = 1: points(3) = 3  
    points(4) = 2: points(5) = 3  
    points(6) = 3: points(7) = 3  
    points(8) = 4: points(9) = 4
```



```

points(10) = 4: points(11) = 2
Set plineObj = ThisDrawing.ModelSpace. _
    AddLightWeightPolyline(points)
plineObj.Closed = True
ZoomAll
' Xác định các thông số
Dim basePoint(0 To 2) As Double
Dim scaleFactor As Double
basePoint(0) = 4: basePoint(1) = 4.25: basePoint(2) = 0
scaleFactor = 0.5
' Co dẫn đối tượng Polyline
plineObj.ScaleEntity basePoint, scaleFactor
plineObj.Update
End Sub

```

## 2.7. Biến đổi đối tượng

Ta có thể di chuyển, co dẫn hoặc quay một đối tượng bằng cách sử dụng phương thức TransformBy với một ma trận 4x4.

Bảng sau minh họa cấu hình của ma trận, trong đó R=quay (Rotation) và T=Dịch tiến (Translation):

Cấu hình ma trận biến đổi			
R00	R01	R02	T0
R10	R11	R12	T1
R20	R21	R22	T2
0	0	0	1

Để biến đổi một đối tượng, trước hết ta cần phải thiết lập ma trận biến đổi. Dưới đây là một ví dụ về ma trận biến đổi, gán cho biến tMatrix, ma trận này sẽ quay một đối tượng một góc 90 độ quanh điểm (0,0,0)

```

tMatrix(0,0) = 0.0
tMatrix(0,1) = -1.0
tMatrix(0,2) = 0.0
tMatrix(0,3) = 0.0
tMatrix(1,0) = 1.0
tMatrix(1,1) = 0.0
tMatrix(1,2) = 0.0
tMatrix(1,3) = 0.0
tMatrix(2,0) = 0.0
tMatrix(2,1) = 0.0
tMatrix(2,2) = 1.0
tMatrix(2,3) = 0.0
tMatrix(3,0) = 0.0
tMatrix(3,1) = 0.0
tMatrix(3,2) = 0.0
tMatrix(3,3) = 1.0

```

Sau khi thiết lập ma trận biến đổi, ta áp ma trận này cho đối tượng cần biến đổi bằng cách sử dụng phương thức TransformBy. Đoạn mã sau sẽ minh họa cách áp một ma trận biến đổi (tMatrix) cho một đối tượng (anObj):

```
anObj.TransformBy tMatrix
```

### Quay một đường thẳng sử dụng ma trận biến đổi

Ví dụ sau đây sẽ tạo một đường thẳng và sau đó quay đường thẳng một góc 90 độ sử dụng ma trận biến đổi.

```
Sub Ch4_TransformBy()  
  ' Tạo đường thẳng  
  Dim lineObj As AcadLine  
  Dim startPt(0 To 2) As Double  
  Dim endPt(0 To 2) As Double  
  startPt(0) = 2  
  startPt(1) = 1  
  startPt(2) = 0  
  endPt(0) = 5  
  endPt(1) = 1  
  endPt(2) = 0  
  Set lineObj = ThisDrawing.ModelSpace.  
  AddLine(startPt, endPt)  
  ZoomAll  
  ' Khởi tạo ma trận biến đổi transMat  
  Dim transMat(0 To 3, 0 To 3) As Double  
  transMat(0, 0) = 0#: transMat(0, 1) = -1#  
  transMat(0, 2) = 0#: transMat(0, 3) = 0#  
  transMat(1, 0) = 1#: transMat(1, 1) = 0#  
  transMat(1, 2) = 0#: transMat(1, 3) = 0#  
  transMat(2, 0) = 0#: transMat(2, 1) = 0#  
  transMat(2, 2) = 1#: transMat(2, 3) = 0#  
  transMat(3, 0) = 0#: transMat(3, 1) = 0#  
  transMat(3, 2) = 0#: transMat(3, 3) = 1#  
  ' Biến đổi đối tượng sử dụng ma trận biến đổi  
  lineObj.TransformBy transMat  
  lineObj.Update  
End Sub
```

Dưới đây là một vài ví dụ về ma trận biến đổi:

#### Ma trận quay: quay góc 90 độ quanh điểm (0,0,0)

0.0	-1.0	0.0	0.0
1.0	0.0	0.0	0.0
0.0	0.0	1.0	0.0
0.0	0.0	0.0	1.0

#### Ma trận quay: quay góc 45 độ quanh điểm (5,5,0)

0.707107	-0.707107	0.0	5.0
0.707107	0.707107	0.0	-2.071068
0.0	0.0	1.0	0.0

0.0	0.0	0.0	1.0
-----	-----	-----	-----

#### Ma trận tịnh tiến: di chuyển một thực thể (10,10,0)

1.0	0.0	0.0	10.0
0.0	1.0	0.0	10.0
0.0	0.0	1.0	0.0
0.0	0.0	0.0	1.0

#### Ma trận co giãn: co giãn đối tượng (10,10,0) tại điểm (0,0,0)

10.0	0.0	0.0	0.0
0.0	10.0	0.0	0.0
0.0	0.0	10.0	0.0
0.0	0.0	0.0	1.0

#### Ma trận co giãn: co giãn đối tượng (10,10,0) tại điểm (2,2,0)

10.0	0.0	0.0	-18.0
0.0	10.0	0.0	-18.0
0.0	0.0	10.0	0.0
0.0	0.0	0.0	1.0

## 2.8. Kéo dài hoặc cắt ngắn đối tượng

Ta có thể thay đổi góc của cung hoặc thay đổi chiều dài của các đường thẳng hở, cung hoặc đường polyline hở, cung ellip, và đường spline hở. Kết quả của những thao tác này cũng giống như khi ta thực hiện kéo dài hoặc cắt ngắn đối tượng.

Ta cũng có thể kéo dài hoặc cắt ngắn đối tượng bằng cách thay đổi các thuộc tính của đối tượng. Ví dụ, để kéo dài đường thẳng, ta chỉ cần thay đổi tọa độ của thuộc tính StartPoint hoặc EndPoint. Để thay đổi góc của một cung, ta thay đổi thuộc tính StartAngle hoặc EndAngle của cung đó. Sau khi đã thay đổi thuộc tính của đối tượng, ta cần phải sử dụng phương thức Update để thấy được những thay đổi trong bản vẽ.

### Kéo dài một đường thẳng

Ví dụ sau sẽ tạo một đường thẳng và sau đó thay đổi thuộc tính EndPoint của đối tượng đó để kéo dài đường thẳng.

```

Sub Ch4_LengthenLine ()
    ' Tạo đường thẳng
    Dim lineObj As AcadLine
    Dim startPoint(0 To 2) As Double
    Dim endPoint(0 To 2) As Double
    startPoint(0) = 0
    startPoint(1) = 0
    startPoint(2) = 0
    endPoint(0) = 1
    endPoint(1) = 1
    endPoint(2) = 1
    Set lineObj = ThisDrawing.ModelSpace. _
        AddLine(startPoint, endPoint)
    lineObj.Update
    ' Kéo dài bằng cách thay đổi endpoint đến 4, 4, 4
    endPoint(0) = 4
    endPoint(1) = 4
    endPoint(2) = 4
    lineObj.endPoint = endPoint
    lineObj.Update
End Sub

```

## 2.9. Phá vỡ đối tượng

Quá trình phá vỡ các đối tượng sẽ chuyển đổi các đối tượng từ một đối tượng ban đầu thành nhiều đối tượng cấu thành nên nó mà không gây ra những thay đổi về mặt hiển thị đối tượng. Ví dụ, việc phá vỡ đối tượng hình thành các đường thẳng và cung từ các đối tượng 3DPolygons, Polylines, Polygon meshes, và Regions. Quá trình này sẽ thay khối được tham chiếu bằng các đối tượng đơn giản cấu thành nên khối đó.

Ta có thể phá vỡ đối tượng 3D polygons, Lightweight polylines, Polylines, Polygon meshes, Regions và Block. Để phá vỡ một đối tượng, ta sử dụng phương thức Explode.

Một đối tượng sau khi được phá vỡ trông chẳng khác gì ban đầu, nhưng màu và kiểu đường của đối tượng có thể thay đổi.

Khi phá vỡ một đường Polyline, AutoCAD sẽ hủy bỏ tất cả các thông tin liên quan đến bề dày. Và kết quả là các đường hoặc cung tạo thành sẽ nằm ở tâm của đường Polyline. Khi phá vỡ một khối có chứa đường Polyline, ta cần phải thực hiện phá vỡ đường Polyline thêm một lần nữa.

Một khối khi được chèn với các hệ số tỷ lệ X, Y, Z khác nhau, sau khi được phá vỡ ra có thể sẽ tạo thành các đối tượng giống không như ta mong đợi. Ta không thể phá vỡ các đối tượng Xref và các khối phụ thuộc của nó. Nếu ta phá vỡ một khối có các thuộc tính gắn kèm thì các thuộc tính này sẽ bị mất đi, tuy nhiên, các định nghĩa thuộc tính ban đầu sẽ vẫn được giữ nguyên. Các giá trị thuộc tính và những thay đổi đều bị xóa.

## Phá vỡ một đường Polyline.

Ví dụ sau tạo một đường Polyline có bề dày, sau đó sẽ phá vỡ nó thành các đối tượng khác nhau. Tiếp đó, sẽ tiến hành duyệt qua các đối tượng và hiển thị hộp thông báo hiển thị tên của từng đối tượng và chỉ số của đối tượng trong danh sách các đối tượng đã được phá vỡ ra.

```
Sub Ch4_ExplodePolyline()  
    Dim plineObj As AcadLWPolyline  
    Dim points(0 To 11) As Double  
    ' Xác định các đỉnh đối tượng polyline  
    points(0) = 1: points(1) = 1  
    points(2) = 1: points(3) = 2  
    points(4) = 2: points(5) = 2  
    points(6) = 3: points(7) = 2  
    points(8) = 4: points(9) = 4  
    points(10) = 4: points(11) = 1  
    ' Tạo đối tượng light weight Polyline  
    Set plineObj = ThisDrawing.ModelSpace.  
        AddLightWeightPolyline(points)  
    ' Thiết lập đoạn cong trên một đoạn  
    ' để thay đổi loại đối tượng  
    plineObj.SetBulge 3, -0.5  
    plineObj.Update  
    ' Phá vỡ đối tượng polyline  
    Dim explodedObjects As Variant  
    explodedObjects = plineObj.Explode  
    ' Duyệt qua các đối tượng đã được phá vỡ ra  
    ' và hiển thị loại của mỗi đối tượng  
    Dim I As Integer  
    For I = 0 To UBound(explodedObjects)  
        explodedObjects(I).Color = acRed  
        explodedObjects(I).Update  
        MsgBox "Exploded Object " & I & ": " & _  
            explodedObjects(I).ObjectName  
        explodedObjects(I).Color = acByLayer  
        explodedObjects(I).Update  
    Next  
End Sub
```

## 2.10. Hiệu chỉnh đối tượng Polylines

Đường đa tuyến, hình chữ nhật, đa giác 2D và 3D cũng như đối tượng lưới đa giác 3D đều là biến thể của đường đa tuyến và đều được hiệu chỉnh theo cùng một quy trình.

AutoCAD nhận được cả đường Fit polyline<sup>1</sup> và đường Spline fit polyline. Đường Spline-fit polyline sử dụng các đường cong tiếp tuyến giống như đường cong B-spline. Có hai loại đường cong Spline-fit polyline: đường cong bậc hai và bậc ba. Cả hai loại đường cong trên đều được điều khiển thông qua biến hệ thống SPLINETYPE.

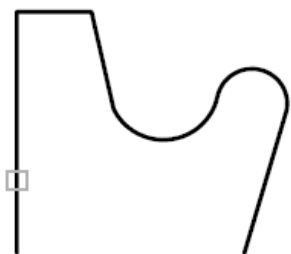
---

<sup>1</sup> **Fit polyline:** là một đường cong trơn bao gồm các cung tròn nối các cặp đỉnh đường Fit polyline. Các đường cong này sẽ đi qua tất cả các đỉnh của đường đa tuyến và sử dụng một hướng tiếp tuyến nào đó mà ta đã chỉ ra.

Ta có thể hiệu chỉnh đường đa tuyến theo cách khép kín và mở hoặc di chuyển, thêm hoặc xóa các đỉnh. Ta cũng có thể kéo thẳng đường đa tuyến giữa hai đỉnh bất kỳ và chuyển đổi giữa các kiểu đường theo ý muốn. Ngoài ra, ta cũng có thể thiết lập bề dày cho toàn bộ đường đa tuyến hoặc theo từng phần của nó.

Ta có thể nối một đường thẳng, cong tròn hoặc một đường đa tuyến khác vào một đường đa tuyến hở nếu hai đầu của chúng trùng nhau. Nếu một đường thẳng cắt ngang qua đầu của đường đa tuyến dưới dạng chữ T thì ta không thể nối các đối tượng này được. Hai đường thẳng giao với đường đa tuyến dưới dạng chữ Y, AutoCAD sẽ lựa chọn một đường thẳng và nối đường thẳng đó với đường đa tuyến. Quá trình nối đường đa tuyến có thể gây ra quá trình làm thẳng ngầm, vì AutoCAD sẽ hủy các thông tin về đường Spline của đường đa tuyến gốc và các đường đa tuyến khác đã nối với đường đa tuyến đó. Sau khi hoàn tất quá trình nối đường đa tuyến, ta có thể thiết lập lại thuộc tính cong Spline cho đường đa tuyến vừa mới tạo thành.

Trong ví dụ sau, mỗi phân đoạn của đường đa tuyến có bề dày điểm đầu và điểm cuối khác nhau tạo thành đường được vuốt đều.



Đường đa tuyến được chọn



Đường đa tuyến với bề dày điểm đầu và cuối của các phân đoạn khác nhau

Để hiệu chỉnh đường đa tuyến, ta sử dụng các thuộc tính và phương thức của đối tượng LightweightPolyline và Polyline. Sử dụng các thuộc tính sau để khép kín hoặc mở đường đa tuyến, thay đổi tọa độ các đỉnh, thêm hoặc xóa đỉnh:

Closed            khép kín hoặc mở đường đa tuyến.

Coordinates    xác định tọa độ từng đỉnh của đường đa tuyến.

AddVertex      thêm một đỉnh vào đường đa tuyến.

Sử dụng các phương thức sau để cập nhật phần cong lồi ra hoặc bề rộng của đường đa tuyến:

SetBulge        thiết lập phần cong lồi của đường đa tuyến tại một đoạn cho trước

SetWidth        thiết lập bề rộng ở điểm đầu và điểm cuối của một đoạn cho trước

### Hiệu chỉnh đường đa tuyến

Ví dụ sau tạo đường Lightweight polyline, sau đó sẽ thêm một đoạn cong lồi ở phân đoạn thứ 3 của đường đa tuyến, thêm một đỉnh vào đường đa tuyến, thay đổi bề dày của phân đoạn cuối và cuối cùng là khép kín đường đa tuyến.

```
Sub Ch4_EditPolyline()  
    Dim plineObj As AcadLWPolyline  
    Dim points(0 To 9) As Double  
    ' Định nghĩa các điểm của đường cong
```

```

points(0) = 1: points(1) = 1
points(2) = 1: points(3) = 2
points(4) = 2: points(5) = 2
points(6) = 3: points(7) = 2
points(8) = 4: points(9) = 4
' Tạo đối tượng light weight Polyline
Set plineObj = ThisDrawing.ModelSpace. _
    AddLightWeightPolyline(points)
' Thêm phần cong lồi đoạn 3
plineObj.SetBulge 3, -0.5
' Định nghĩa đỉnh mới
Dim newVertex(0 To 1) As Double
newVertex(0) = 4: newVertex(1) = 1
' Thêm đỉnh cho đối tượng polyline
plineObj.AddVertex 5, newVertex
' Thiết lập bề dày cho phân đoạn mới
plineObj.SetWidth 4, 0.1, 0.5
' Khép kín đối tượng Polyline
plineObj.Closed = True
plineObj.Update
End Sub

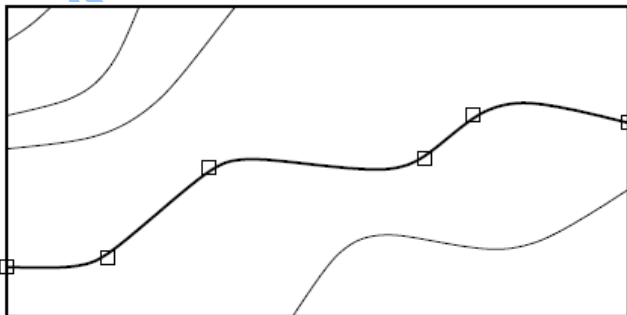
```

## 2.11. Hiệu chỉnh đường cong Splines

Ta có thể xóa hay thêm các điểm chuẩn để tăng độ chính xác hoặc di chuyển các điểm chuẩn để thay đổi hình dáng đường cong Spline. Ta cũng có thể mở hay khép kín đường cong Spline, hiệu chỉnh tiếp tuyến đầu và cuối, đổi chiều đường cong Spline. Ngoài ra ta cũng có thể thay đổi dung sai của đường cong Spline. Dung sai thể hiện mức độ lệch của đường cong Spline so với tập hợp các điểm chuẩn mà ta đã ấn định. Dung sai càng thấp thì đường cong Spline càng gần với các điểm chuẩn.

Ta có thể tinh chỉnh một đường cong Spline bằng cách tăng số điểm điều khiển trên một đoạn hoặc thay đổi trọng số của các điểm điều khiển cụ thể. Gia tăng trọng số của một điểm điều khiển sẽ kéo đường cong về gần điểm đó hơn. Chúng ta cũng có thể tinh chỉnh một đường cong Spline bằng cách thay đổi bậc của nó. Bậc của một đường cong Spline chính bằng bậc của đa thức Spline cộng thêm 1. Chẳng hạn, một đường cong Spline bậc 3 sẽ có bậc là 4. Một đường cong Spline có bậc càng cao thì càng có nhiều điểm điều khiển.

Xem ví dụ sau:



Ví dụ tạo một đường cong Spline biểu diễn đường đồng mức. Ta cần phải dịch chuyển điểm điều khiển thứ tư nhằm tăng độ chính xác. Sử dụng phương thức SetFitPoint để nhập tọa độ mới cho điểm chuẩn thứ tư.





Sử dụng các thuộc tính sau để thay đổi đường cong Spline:

Closed	Mở hoặc khép kín đường cong Spline
ControlPoints	Xác định các điểm điều khiển của một đường cong Spline
EndTangent	Xác định tiếp tuyến cuối của đường cong Spline
FitPoints	Xác định tất cả các điểm chuẩn của một đường cong Spline
FitTolerance	Điều chỉnh đường cong Spline cho phù hợp với các điểm hiện có với giá trị dung sai mới
Knots	Xác định vectơ nút cho đường cong Spline
StartTangent	Xác định tiếp tuyến đầu cho đường cong Spline

Ngoài ra, ta còn có thể sử dụng các phương thức sau để hiệu chỉnh đường cong Spline:

AddFitPoint	Thêm một điểm chuẩn vào đường cong Spline ở một vị trí cho trước
DeleteFitPoint	Xoá điểm chuẩn của một đường cong Spline
ElevateOrder	Nâng bậc của đường cong Spline
GetFitPoint	Lấy lại điểm chuẩn của đường cong Spline tại chỉ số cho trước (chỉ lấy một điểm duy nhất. Để tìm tất cả các điểm chuẩn ta dùng thuộc tính FitPoints)
Reverse	Đổi chiều của một đường cong Spline
SetControlPoint	Thiết lập điểm điều khiển của một đường cong Spline tại chỉ số cho trước.
SetFitPoint	Thiết lập điểm chuẩn cho đường cong Spline tại chỉ số cho trước (chỉ thiết lập một điểm chuẩn, để thay đổi tất cả các điểm chuẩn của đường cong Spline, ta sử dụng thuộc tính FitPoints)
SetWeight	Lập trọng số cho điểm điều khiển tại chỉ số cho trước

Sử dụng các thuộc tính chỉ đọc sau để truy vấn đường cong Spline:

Degree	Tìm bậc đa thức của đường cong Spline
Area	Diện tích vùng bao quanh bởi đường cong Spline
IsPeriodic	Xác định xem đường Spline có tuần hoàn hay không
IsPlanar	Xác định xem đường Spline có phẳng hay không

IsRational            Xác định xem đường Spline có hữu tỷ hay không

NumberOfControlPoints  
Số điểm điều khiển của đường cong Spline

NumberOfFitPoints  
Số điểm chuẩn của đường cong Spline

### Thay đổi điểm điều khiển của đường cong Spline

Ví dụ sau tạo một đường cong Spline và thay đổi điểm điều khiển đầu tiên của đường cong Spline.

```
Sub Ch4_ChangeSplineControlPoint()  
    ' Tạo đường cong spline  
    Dim splineObj As AcadSpline  
    Dim noOfPoints As Integer  
    Dim startTan(0 To 2) As Double  
    Dim endTan(0 To 2) As Double  
    Dim fitPoints(0 To 8) As Double  
    noOfPoints = 3  
    startTan(0) = 0.5: startTan(1) = 0.5: startTan(2) = 0  
    endTan(0) = 0.5: endTan(1) = 0.5: endTan(2) = 0  
    fitPoints(0) = 1: fitPoints(1) = 1: fitPoints(2) = 0  
    fitPoints(3) = 5: fitPoints(4) = 5: fitPoints(5) = 0  
    fitPoints(6) = 10: fitPoints(7) = 0: fitPoints(8) = 0  
    Set splineObj = ThisDrawing.ModelSpace.  
        AddSpline(fitPoints, startTan, endTan)  
    splineObj.Update  
    ' Thay đổi tọa độ của điểm điều khiển đầu tiên  
    Dim controlPoint(0 To 2) As Double  
    controlPoint(0) = 0  
    controlPoint(1) = 3  
    controlPoint(2) = 0  
    splineObj.SetControlPoint 0, controlPoint  
    splineObj.Update  
End Sub
```

## 2.12. Hiệu chỉnh vùng tô mẫu

Ta có thể hiệu chỉnh cả đường biên và mẫu tô của vùng tô mẫu. Nếu hiệu chỉnh đường biên của vùng tô mẫu có liên kết, mẫu tô sẽ được cập nhật tương ứng, miễn là đường biên vẫn phù hợp. Các vùng tô mẫu có liên kết cũng được cập nhật cho dù chúng nằm trong lớp đã bị tắt. Chúng ta có thể hiệu chỉnh mẫu tô hoặc chọn mẫu tô mới cho vùng tô mẫu hiện có, tuy nhiên thuộc tính liên kết chỉ được gán khi tạo đối tượng vùng tô mẫu. Ta có thể kiểm tra xem liệu một vùng tô mẫu có liên kết không bằng cách sử dụng thuộc tính AssociativeHatch. (Xem thêm phương thức AddHatch để biết cách tạo ra một vùng tô mẫu)

Cần phải tính toán lại vùng tô mẫu bằng cách sử dụng phương thức Evaluate để thấy được những thay đổi trên vùng tô mẫu.

### 2.12.1. Hiệu chỉnh đường biên vùng tô mẫu

Ta có thể bổ sung hay chèn thêm đường biên vào tập hợp các đường biên của vùng tô mẫu. Các vùng tô mẫu liên kết cũng sẽ được cập nhật tương ứng với những thay

đổi của đường biên. Vùng tô mẫu không liên kết sẽ không thực hiện quá trình cập nhật.

Để hiệu chỉnh đường biên vùng tô mẫu, sử dụng một trong các phương thức sau:

AppendInnerLoop      Bổ sung đường biên trong cho vùng tô mẫu.

AppendOuterLoop      Bổ sung đường biên ngoài cho vùng tô mẫu.

InsertLoopAt          Chèn một đường biên tại chỉ số cho trước của vùng tô mẫu.

### **Bổ sung đường biên cho vùng tô mẫu:**

Trong ví dụ sau chúng ta sẽ tạo một vùng tô mẫu có liên kết. Sau đó sẽ tạo một vòng tròn và thêm vòng tròn đó thành đường biên trong cho vùng tô mẫu.

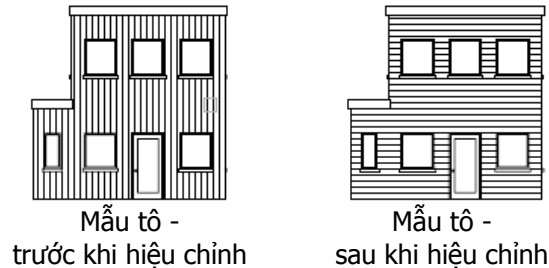
```
Sub Ch4_AppendInnerLoopToHatch()  
    Dim hatchObj As AcadHatch  
    Dim patternName As String  
    Dim PatternType As Long  
    Dim bAssociativity As Boolean  
    ' Định nghĩa và tạo vùng tô mẫu  
    patternName = "ANSI31"  
    PatternType = 0  
    bAssociativity = True  
    Set hatchObj = ThisDrawing.ModelSpace.  
    AddHatch(PatternType, patternName, bAssociativity)  
    ' Tạo đường biên ngoài.  
    Dim outerLoop(0 To 1) As AcadEntity  
    Dim center(0 To 2) As Double  
    Dim radius As Double  
    Dim startAngle As Double  
    Dim endAngle As Double  
    center(0) = 5: center(1) = 3: center(2) = 0  
    radius = 3  
    startAngle = 0  
    endAngle = 3.141592  
    Set outerLoop(0) = ThisDrawing.ModelSpace.  
    AddArc(center, radius, startAngle, endAngle)  
    Set outerLoop(1) = ThisDrawing.ModelSpace.  
    AddLine(outerLoop(0).startPoint, outerLoop(0).endPoint)  
    ' Thêm đường biên ngoài cho đối tượng hatch  
    hatchObj.AppendOuterLoop (outerLoop)  
    ' Tạo vòng tròn làm đường biên trong.  
    Dim innerLoop(0) As AcadEntity  
    center(0) = 5: center(1) = 4.5: center(2) = 0  
    radius = 1  
    Set innerLoop(0) = ThisDrawing.ModelSpace.  
    AddCircle(center, radius)  
    ' Append the circle as an inner loop to the hatch  
    hatchObj.AppendInnerLoop (innerLoop)  
    ' Evaluate and display the hatch  
    hatchObj.Evaluate  
    ThisDrawing.Regen True  
End Sub
```

### **2.12.2. Hiệu chỉnh mẫu tô của vùng tô mẫu**

Ta có thể điều chỉnh góc và khoảng cách của mẫu tô vùng tô mẫu hiện có hoặc thay thế bằng một vùng tô đặc hoặc mẫu tô định nghĩa trước của AutoCAD. Lựa chọn

Pattern trong hộp thoại Boundary Hatch để hiển thị danh sách tất cả các mẫu tô. Để giảm kích thước tệp, vùng tô mẫu được định nghĩa như một đối tượng đơn nhất trong bản vẽ.

Ví dụ sau minh họa kết quả thay đổi góc vùng tô mẫu 90 độ.



Sử dụng các thuộc tính và phương thức sau để hiệu chỉnh mẫu tô:

PatternAngle	Xác định góc của mẫu tô.
PatternDouble	Xác định xem vùng tô mẫu do người dùng định nghĩa có phải là loại vùng tô nét đôi hay không.
PatternName	Xác định tên mẫu tô (không thay đổi loại mẫu tô).
PatternScale	Xác định tỷ lệ mẫu tô.
PatternSpace	Xác định khoảng cách mẫu tô do người dùng định nghĩa.
SetPattern	Thiết lập tên mẫu tô và loại mẫu tô.

### Thay đổi khoảng cách mẫu tô

Ví dụ sau tạo vùng tô mẫu, sau đó cộng thêm 2 vào giá trị khoảng cách mẫu tô hiện hành.

```
Sub Ch4_ChangeHatchPatternSpace()
    Dim hatchObj As AcadHatch
    Dim patternName As String
    Dim PatternType As Long
    Dim bAssociativity As Boolean
    ' Định nghĩa vùng tô mẫu
    patternName = "ANSI31"
    PatternType = 0
    bAssociativity = True
    ' Tạo đối tượng vùng tô mẫu có liên kết
    Set hatchObj = ThisDrawing.ModelSpace. _
        AddHatch(PatternType, patternName, bAssociativity)
    ' Tạo đường biên ngoài của vùng tô mẫu
    Dim outerLoop(0 To 0) As AcadEntity
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 5
    center(1) = 3
    center(2) = 0
    radius = 3
    Set outerLoop(0) = ThisDrawing.ModelSpace. _
        AddCircle(center, radius)
    hatchObj.AppendOuterLoop (outerLoop)
    hatchObj.Evaluate
    ' Thay đổi khoảng cách mẫu tô của vùng tô mẫu
```

```

hatchObj.patternSpace = hatchObj.patternSpace + 2
hatchObj.Evaluate
ThisDrawing.Regen True
End Sub

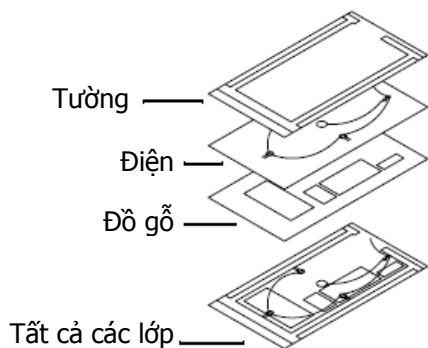
```

### 3. Sử dụng Lớp, Màu sắc và Kiểu đường

Lớp là các tầng trong suốt mà trên đó ta có thể tổ chức và nhóm nhiều loại thông tin khác nhau của bản vẽ. Những đối tượng được tạo ra sẽ chứa các thuộc tính bao gồm lớp, màu sắc và kiểu đường. Màu sắc giúp ta phân biệt các yếu tố giống nhau trong bản vẽ và kiểu đường giúp phân loại dễ dàng các yếu tố đồ họa khác nhau, chẳng hạn như các đường trục và các nét khuất. Sắp xếp các lớp và đối tượng trên lớp giúp ta quản lý thông tin của bản vẽ dễ dàng hơn.

#### 3.1. Làm việc với các lớp

Quá trình vẽ luôn được thực hiện trên một lớp nào đó. Đó có thể là lớp mặc định hoặc một lớp tự tạo ra và được đặt tên. Mỗi lớp đều có một màu và kiểu đường tương ứng. Chẳng hạn, ta có thể tạo ra một lớp mà trên đó chỉ vẽ các đường trục và gán cho nó màu xanh với kiểu đường là CENTER. Tiếp đó, cứ khi nào muốn vẽ các đường trục ta chỉ cần chuyển sang lớp đó và bắt đầu vẽ. Không nhất thiết là phải thiết lập kiểu đường và màu sắc mỗi khi muốn vẽ một đường trục. Và ta cũng có thể tắt lớp đường trục nếu không muốn hiển thị hoặc in các đối tượng này. Sử dụng lớp là một trong những ưu điểm lớn khi vẽ bằng AutoCAD thay vì với giấy bút thông thường.



Trong không gian in, hoặc khi làm việc trong khung nhìn nổi, ta có thể thiết lập tính nhìn thấy cho các lớp trong từng khung nhìn. Tất cả các lớp trong một bản vẽ áp dụng cùng một giới hạn vẽ, hệ tọa độ và hệ số phóng đại. Nếu ta sử dụng nhất quán một mô hình lớp, ta có thể lập một bản vẽ mẫu với các lớp, kiểu đường và màu sắc đã được gán từ trước.

##### 3.1.1. Duyệt Lớp và Kiểu đường

Tất cả các lớp và kiểu đường đều nằm trong các tập đối tượng tương ứng. Các lớp thì nằm trong tập đối tượng lớp còn các kiểu đường thì nằm trong tập đối tượng kiểu đường.

Ta có thể duyệt qua tập đối tượng để tìm tất cả các lớp và kiểu đường trong bản vẽ.

### Duyệt qua tập đối tượng lớp:

Đoạn mã dưới đây duyệt qua các tập đối tượng lớp để thu thập và sau đó hiển thị tên của tất cả các lớp trong hộp thông báo.

```
Sub Ch4_IteratingLayers()  
    Dim layerNames As String  
    Dim entry As AcadLayer  
    layerNames = ""  
    For Each entry In ThisDrawing.Layers  
        layerNames = layerNames + entry.Name + vbCrLf  
    Next  
    MsgBox "The layers in this drawing are: " + _  
        vbCrLf + layerNames  
End Sub
```

### 3.1.2. Tạo và đặt tên các lớp

Ta có thể tạo và đặt tên lớp cho một nhóm các đối tượng có ý nghĩa chung nào đó (chẳng hạn như lớp các bức tường hoặc lớp kích thước) và gán màu và kiểu đường cho những lớp này. Khi sắp xếp sơ đồ lớp, cần chọn tên lớp thật thận trọng.

Khi ta bắt đầu một bản vẽ mới, AutoCAD sẽ tạo ra một lớp đặc biệt có tên là 0. Lớp 0 này được mặc định gán cho màu số 7 (màu trắng hoặc màu đen tùy thuộc vào giá trị màu nền) và với kiểu đường CONTINUOUS (liên tục). Ta không thể xóa lớp 0 này.

Ta có thể tạo ra lớp mới và gán thuộc tính màu sắc cũng như kiểu đường cho những lớp này. Mỗi một lớp là một phần tử của tập đối tượng Layers. Ta dùng phương thức Add để tạo ra một lớp mới và bổ sung nó vào tập đối tượng Layers.

Ta cũng có thể gán tên khi tiến hành tạo mới lớp. Để đổi tên của một lớp sau khi đã tạo lập lớp đó, ta dùng thuộc tính Name. Tên của một lớp có thể chứa đến 31 ký tự bao gồm các chữ cái, số và các ký hiệu đặc biệt như ký hiệu đồng đô la (\$), dấu gạch nối (-), gạch dưới (\_) nhưng không có dấu cách.

#### Gán đối tượng cho một lớp mới

Đoạn mã sau sẽ tạo một đường tròn và một lớp mới. Lớp mới được gán màu đỏ. Sau đó đường tròn được gán với lớp đó và màu sắc của vòng sẽ thay đổi tương ứng.

```
Sub Ch4_NewLayer()  
    ' Tạo đường tròn  
    Dim circleObj As AcadCircle  
    Dim center(0 To 2) As Double  
    Dim radius As Double  
    center(0) = 2: center(1) = 2: center(2) = 0  
    radius = 1  
    Set circleObj = ThisDrawing.ModelSpace. _  
        AddCircle(center, radius)  
  
    ' Gán đường tròn có màu là "ByLayer" để có thể tự động  
    ' lấy màu của lớp khi được gán vào lớp mới  
    circleObj.Color = acByLayer  
  
    ' Tạo lớp mới có tên "ABC"  
    Dim layerObj As AcadLayer  
    Set layerObj = ThisDrawing.Layers.Add("ABC")  
    ' Gán lớp "ABC" màu đỏ  
    layerObj.Color = acRed
```

```

' Gán đường tròn vào lớp "ABC"
circleObj.Layer = "ABC"
circleObj.Update
End Sub

```

### 3.1.3. Chọn lớp hiện hành

Quá trình vẽ luôn được thực hiện trên lớp hiện hành. Sau khi chọn lớp hiện hành, ta có thể tạo các đối tượng mới trên lớp đó. Nếu ta chọn lớp khác làm lớp hiện hành thì tất cả các đối tượng mới tạo sẽ dựa trên lớp hiện hành mới sử dụng màu và kiểu đường của lớp đó. Ta không thể chọn lớp hiện hành nếu đó là lớp đã đông cứng.

Để chọn một lớp làm lớp hiện hành, ta sử dụng thuộc tính `ActiveLayer`. Thuộc tính này gán cho bản vẽ hiện hành. Ví dụ:

```

Dim newlayer As AcadLayer
Set newlayer = ThisDrawing.Layers.Add("LAYER1")
ThisDrawing.ActiveLayer = newlayer

```

### 3.1.4. Điều khiển tính nhìn thấy của Lớp

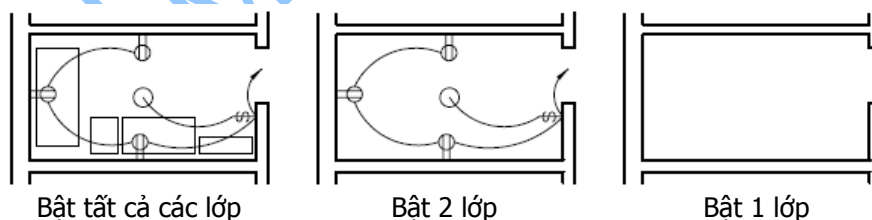
AutoCAD không hiển thị hay in các đối tượng được vẽ trên các lớp ẩn. Nếu muốn không bị vướng tầm nhìn khi làm việc chi tiết trên một lớp hoặc một số lớp cụ thể, hoặc nếu không muốn in một số chi tiết nhất định, chẳng hạn như đường chuẩn hoặc các đường tạm, ta có thể tắt hoặc làm đông cứng các lớp đó.

Phương thức để điều khiển tính nhìn thấy của một lớp phụ thuộc vào cách ta làm việc và kích thước bản vẽ. Ta có thể làm đông cứng lớp nếu muốn ẩn nó đi trong một thời gian dài.

Ta chỉ có thể in các lớp đang hiển thị. Trên hầu hết các máy vẽ (máy in), ta có thể gán các bút vẽ khác nhau cho mỗi một màu trong bản vẽ. Đối với máy vẽ chỉ có một bút thì AutoCAD sẽ tạm dùng để thay đổi bút vẽ trong quá trình in.

### 3.1.5. Bật và Tắt Lớp

Các lớp đã tắt sẽ vẫn được tái tạo lại cùng với bản vẽ nhưng lại không được hiển thị hay in ra. Ta nên tắt các lớp đi thay vì làm đông cứng chúng nếu ta thường xuyên chuyển đổi giữa các lớp có nhìn thấy và lớp ẩn. Bằng cách tắt các lớp, ta tránh phải tái tạo lại bản vẽ mỗi khi làm tan lớp. Khi bật một lớp đã được tắt, AutoCAD sẽ vẽ lại các đối tượng trên lớp đó.



Để bật hay tắt các lớp, ta sử dụng thuộc tính `LayerOn`. Để bật một lớp, gán giá trị `TRUE` và để tắt một lớp thì gán giá trị `FALSE` cho thuộc tính này.

#### Tắt một lớp

Trong ví dụ sau, ta tạo một lớp mới, tiếp đó thêm một vòng tròn và tắt lớp đó đi để không nhìn thấy vòng tròn nữa.



```

Sub Ch4_LayerInvisble()
    ' Tạo đường tròn
    Dim circleObj As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 2: center(1) = 2: center(2) = 0
    radius = 1
    Set circleObj = ThisDrawing.ModelSpace. _
    AddCircle(center, radius)
    circleObj.Color = acByLayer
    ' Tạo lớp "ABC"
    Dim layerObj As AcadLayer
    Set layerObj = ThisDrawing.Layers.Add("ABC")
    layerObj.Color = acRed
    ' Gán đường tròn vào lớp "ABC"
    circleObj.Layer = "ABC"
    circleObj.Update
    ' Tắt lớp "ABC"
    layerObj.LayerOn = False
    ThisDrawing.Regen acActiveViewport
End Sub

```

### 3.1.6. Làm đông cứng và làm tan Lớp

Ta có thể làm đông cứng các lớp để tăng tốc độ hiển thị những thay đổi, cải thiện quá trình lựa chọn đối tượng và giảm thời gian tái tạo cho những bản vẽ phức tạp. AutoCAD không hiển thị, in hay tái tạo các đối tượng trên các lớp đông cứng. Nên làm đông cứng các lớp mà ta không muốn hiển thị trong một thời gian dài. Khi ta làm “tan” một lớp bị đông cứng, AutoCAD sẽ phục hồi và hiển thị các đối tượng trên lớp đó.

Để làm đông cứng hay làm tan một lớp bị đông cứng, ta sử dụng thuộc tính Freeze. Để làm đông cứng một lớp, ta gán giá trị True và để làm tan một lớp, ta gán giá trị False cho thuộc tính này.

#### Làm đông cứng một lớp

Trong ví dụ sau, ta tạo một lớp mới có tên “ABC” và sau đó làm đông cứng lớp đó.

```

Sub Ch4_LayerFreeze()
    ' Tạo lớp "ABC"
    Dim layerObj As AcadLayer
    Set layerObj = ThisDrawing.Layers.Add("ABC")
    ' Đông cứng lớp "ABC"
    layerObj.Freeze = True
End Sub

```

### 3.1.7. Khoá và Mở khoá Lớp

Biện pháp khoá lớp rất hữu ích khi ta muốn hiệu chỉnh các đối tượng có liên kết với một lớp nào đó nhưng đồng thời cũng muốn quan sát các đối tượng ở các lớp khác. Ta không thể hiệu chỉnh các đối tượng nằm trên một lớp đã bị khoá. Tuy nhiên ta vẫn có thể nhìn thấy các đối tượng này nếu lớp đó vẫn được bật hoặc không đông cứng. Ta có thể biến một lớp đã bị khoá thành lớp hiện hành và thêm đối tượng vào lớp đó. Ta cũng có thể làm đông cứng và tắt các lớp đã bị khoá cũng như đổi màu và kiểu đường của các lớp này.

Để khoá hay mở các lớp, ta sử dụng thuộc tính Lock. Nếu muốn khoá một lớp, ta gán giá trị TRUE, mở khóa ta gán giá trị FALSE cho thuộc tính này.

### Khoá một lớp

Ví dụ sau tạo một lớp có tên “ABC” và sau đó khoá lớp này lại.

```
Sub Ch4_LayerLock()  
    ' Tạo lớp "ABC"  
    Dim layerObj As AcadLayer  
    Set layerObj = ThisDrawing.Layers.Add("ABC")  
    ' Khóa lớp "ABC"  
    layerObj.Lock = True  
End Sub
```

### 3.1.8. Gán màu cho một Lớp

Ta có thể gán màu cho một lớp. Khi gán màu, ta có thể nhập tên màu hoặc nhập chỉ số màu ACI (ACI – AutoCAD Color Index). Tên màu tiêu chuẩn chỉ có cho các màu từ số 1 đến số 7.

AutoCAD mặc định gán màu số 7 (trắng hay đen phụ thuộc vào màu nền) cho các lớp mới tạo lập. Ta có thể gán một màu khác với màu của lớp cho một đối tượng.

Để gán màu cho một lớp, ta sử dụng thuộc tính Color.

Ta có thể gán và đọc giá trị màu như là các chỉ số với giá trị nằm trong khoảng từ 0 đến 256. Các hằng số được gán cho bảy màu tiêu chuẩn và các giá trị BYBLOCK, BYLAYER.

Nếu sử dụng hằng `acByBlock`, AutoCAD sẽ vẽ đối tượng mới bằng màu mặc định (trắng hay đen tùy thuộc vào cấu hình) cho tới khi chúng được nhóm lại thành một khối. Khi chèn khối này vào bản vẽ, các đối tượng trong khối sẽ nhận giá trị hiện hành của thuộc tính Color.

Nếu sử dụng `acByLayer`, đối tượng mới sẽ lấy màu của lớp mà đối tượng đang được vẽ lên.

### 3.1.9. Gán kiểu đường cho lớp

Khi ta định nghĩa các lớp, kiểu đường là một trong những cách thể hiện thông tin một cách trực quan.

Kiểu đường là sự lặp lại mẫu nét đứt, điểm, và khoảng trống, dùng để phân biệt mục đích của các đường khác nhau.

Tên kiểu đường và định nghĩa sẽ mô tả trình tự gạch-chấm, chiều dài tương đối của dấu gạch-khoảng trống và đặc tính của đối tượng văn bản hoặc hình dạng bất kỳ có trong kiểu đường.

Để gán kiểu đường cho lớp, ta sử dụng thuộc tính Linetype. Thuộc tính này lấy tên của kiểu đường làm đầu vào.

### 3.1.10. Xóa lớp

Để xóa lớp, ta sử dụng phương thức Delete.

Lớp có thể được xóa bất kỳ lúc nào trong khi vẽ. Tuy nhiên, không thể xóa lớp hiện hành, lớp 0, lớp thuộc tham chiếu ngoài, hoặc lớp đã có chứa đối tượng.

**CHÚ Ý** Các lớp được tham chiếu trong định nghĩa khối, cùng với lớp đặc biệt có tên DEFPOINTS không thể xóa được ngay cả khi chúng không chứa đối tượng trực quan nào.

## 3.2. Làm việc với màu sắc

Có thể gán màu cho các lớp và cho từng đối tượng trong bản vẽ. Mỗi màu được định nghĩa bằng tên hoặc mã số màu ACI, được đánh số nguyên từ 1 đến 255. Các đối tượng và các lớp có thể có màu giống nhau. Ta có thể gán nét in khác nhau cho từng màu trong khi in hoặc sử dụng chỉ số màu để xác định đối tượng nào đó trong bản vẽ, dù có thể ta không nhìn thấy màu đó trên màn hình.

### 3.2.1. Định nghĩa màu

Khi định nghĩa một màu, có thể nhập tên của màu hoặc theo số ACI của màu đó. ACI có 255 màu. Các tên màu chuẩn chỉ có cho màu từ 1 đến 7.

#### Màu từ 1 đến 7

Số màu	Tên màu
1	Đỏ
2	Vàng
3	Xanh lá cây
4	Xanh lam
5	Xanh nước biển
6	Hồng xám
7	Đen/Trắng

Các màu từ 8 đến 255 cần được gán sử dụng số hoặc chọn trong hộp thoại màu. Mặc định màu (7) là đen hoặc trắng, phụ thuộc vào màu nền.

### 3.2.2. Thiết lập màu hiện hành

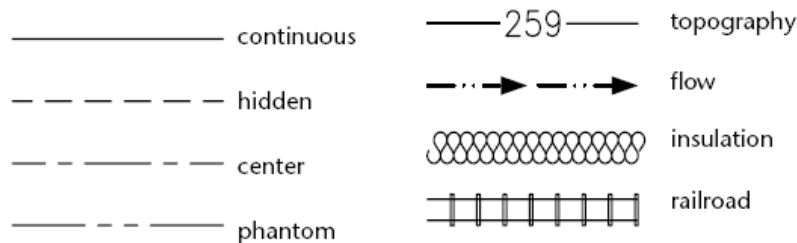
Ta có thể gán màu cho các lớp hoặc các đối tượng đã được tạo, định nghĩa màu hiện hành theo màu của lớp hiện hành có thể hoặc định nghĩa một màu khác.

Nếu chọn `BYLAYER`, các đối tượng mới nhận màu của lớp chứa đối tượng. Nếu chọn `BYBLOCK`, đối tượng mới được vẽ với màu mặc định cho đến khi được nhóm trong một khối. Các đối tượng trong khối sẽ kế thừa thông số màu hiện tại.

Để thiết lập lại màu hiện tại của đối tượng hoặc lớp, sử dụng thuộc tính `Color`.

### 3.3. Làm việc với kiểu đường

Kiểu đường là sự lặp lại mẫu gạch ngang, chấm, và các khoảng trống. Kiểu đường phức tạp là sự lặp lại mẫu ký hiệu. Tên kiểu đường và định nghĩa mô tả trình tự gạch – chấm, khoảng cách tương đối của dấu gạch ngang và khoảng trống, và các đặc tính khác của ký tự hoặc hình nào đó trong kiểu đường. Ta có thể tự tạo kiểu đường riêng.



Để sử dụng một kiểu đường, cần phải nạp kiểu đường đó vào trong bản vẽ. Định nghĩa của kiểu đường phải tồn tại trong tệp thư viện .LIN trước khi nạp vào bản vẽ. Để nạp kiểu đường vào bản vẽ, sử dụng phương thức Load.

#### Nạp kiểu đường vào AutoCAD

Ví dụ sau nạp kiểu đường “CENTER” từ tệp *acad.lin*. Nếu kiểu đường đã tồn tại hoặc tệp không có, sẽ có thông báo cho người dùng.

```
Sub Ch4_LoadLinetype()  
  On Error GoTo ERRORHANDLER  
  Dim linetypeName As String  
  linetypeName = "CENTER"  
  
  ' Nạp kiểu đường "CENTER" từ tệp acad.lin  
  ThisDrawing.Linetypes.Load linetypeName, "acad.lin"  
  Exit Sub  
ERRORHANDLER:  
  MsgBox Err.Description  
End Sub
```

---

**CHÚ Ý** Không nên nhầm lẫn giữa kiểu đường được dùng trong AutoCAD với các kiểu đường được cung cấp trong máy vẽ. Hai loại đường gạch ngang cho kết quả tương tự. Tuy nhiên không nên sử dụng cả hai loại cùng một lúc, bởi vì có thể không dự đoán được kết quả.

---

#### 3.3.1. Kiểu đường hiện hành

Để dùng kiểu đường vẽ trong lớp hiện hành, cần kích hoạt nó. Tất cả các đối tượng mới được tạo sẽ được vẽ sử dụng kiểu đường hiện hành vừa được kích hoạt.

---

**CHÚ Ý** Kiểu đường trong tham chiếu ngoài không thể kích hoạt được.

---

Nếu chọn BYLAYER, các đối tượng mới nhận kiểu đường của kiểu đường hiện hành. Nếu chọn BYBLOCK, các đối tượng mới được vẽ sẽ sử dụng kiểu đường đó cho đến khi chúng được nhóm vào trong một khối. Các đối tượng trong khối kế thừa thuộc tính kiểu đường hiện tại.

Để chuyển kiểu đường thành hiện hành, sử dụng thuộc tính `ActiveLinetype`. Thuộc tính này được thiết lập trên bản vẽ hiện hành. Ví dụ:

```
ThisDrawing.ActiveLinetype = ThisDrawing. _  
Linetypes.Item("CONTINUOUS")
```

### 3.3.2. Đổi tên kiểu đường

Ta có thể đổi tên một kiểu đường để dễ nhận biết hơn trong quá trình sử dụng. Có thể đổi tên bất kỳ lúc nào trong khi vẽ.

Khi đổi tên kiểu đường, thực chất là chỉ đổi tên kiểu đường định nghĩa trong bản vẽ. Tên kiểu đường trong tệp `.LIN` vẫn giữ nguyên.

Để đổi tên kiểu đường, ta sử dụng thuộc tính `Name`.

---

**CHÚ Ý** Không thể đổi tên kiểu đường `BYLAYER`, `BYBLOCK`, hoặc `CONTINUOUS` hoặc kiểu đường thuộc tham chiếu ngoài.

---

### 3.3.3. Xóa kiểu đường

Ta có thể xóa kiểu đường bất kỳ lúc nào trong khi vẽ, tuy nhiên không thể xóa kiểu đường `BYLAYER`, `BYBLOCK`, `CONTINUOUS`, kiểu đường hiện hành và kiểu đường thuộc tham chiếu ngoài. Ngoài ra, kiểu đường tham chiếu trong định nghĩa khối cũng không thể xóa được, ngay cả khi không có đối tượng trực quan nào sử dụng kiểu đường đó.

Để xóa kiểu đường, ta sử dụng phương thức `Delete`.

### 3.3.4. Thay đổi phần mô tả kiểu đường

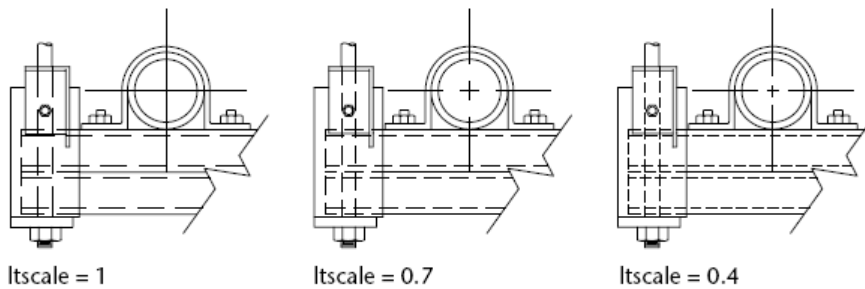
Mỗi kiểu đường đều có thể có thêm phần mô tả đi cùng. Phần mô tả đưa ra giải thích về kiểu đường đó dưới dạng ký tự ASCII. Ta có thể gán hoặc thay đổi phần chú thích của kiểu đường sử dụng thuộc tính `Description`.

Phần mô tả kiểu đường có thể chứa đến 47 ký tự. Phần mô tả có thể là một lời chú thích hoặc các dấu gạch dưới, dấu chấm, gạch ngang và khoảng trắng để thể hiện một cách sơ lược mẫu của kiểu đường. Ví dụ:

```
ThisDrawing.ActiveLinetype.Description = "Exterior Wall"
```

### 3.3.5. Xác định tỷ lệ cho kiểu đường

Ta có thể thiết lập tỷ lệ kiểu đường cho các đối tượng. Tỷ lệ càng nhỏ thì càng nhiều mẫu được lặp đi lặp lại trong bản vẽ. Mặc định, AutoCAD sử dụng tỷ lệ kiểu đường toàn cục là 1.0, nghĩa là tương đương với một đơn vị vẽ. Ta có thể thay đổi tỷ lệ kiểu đường cho tất cả các đối tượng trong bản vẽ, các tham khảo thuộc tính, các nhóm.



Để thay đổi tỷ lệ kiểu đường, sử dụng thuộc tính LinetypeScale.

Biến hệ thống CELTSCALE thiết lập tỷ lệ cho các đối tượng mới tạo. Còn biến LTSCALE thay đổi toàn bộ tỷ lệ kiểu đường của tất cả các đối tượng đã có cũng như những đối tượng mới. Để thay đổi giá trị của biến hệ thống trong AutoCAD ActiveX Automation, dùng phương thức SetVariable.

### Thay đổi tỷ lệ kiểu đường cho đường tròn

```
Sub Ch4_ChangeLinetypeScale()
    Dim center(0 To 2) As Double
    Dim radius As Double
    Dim circleObj As AcadCircle

    ' Tạo vòng tròn trong không gian mô hình
    center(0) = 2
    center(1) = 2
    center(2) = 0
    radius = 4
    Set circleObj = ThisDrawing.ModelSpace. _
        AddCircle(center, radius)

    ' Gán tỷ lệ kiểu đường cho đường tròn thành 0.5
    circleObj.LinetypeScale = 0.5
    circleObj.Update
End Sub
```

## 3.4. Gán Lớp, Màu và Kiểu đường cho Đối tượng

Sau khi đã được định nghĩa, lớp, màu và kiểu đường có thể được gán cho các đối tượng trong bản vẽ. Ta có thể nhóm những thành phần có liên quan trong bản vẽ vào trong một lớp. Ta cũng có thể điều chỉnh tắt mở lớp, màu, kiểu đường và xác định xem đối tượng trong lớp có được hiệu chỉnh hay không. Ta có thể chuyển đối tượng từ lớp này sang lớp khác và thay đổi tên của lớp.

Số lượng lớp trong bản vẽ và số lượng đối tượng trong một lớp là không giới hạn. Ta có thể gán tên từng lớp và chọn bất kỳ kiểu kết hợp nào trong việc hiển thị lớp.

Khối có thể được định nghĩa từ các đối tượng vẽ trên những lớp khác nhau với màu sắc và kiểu đường khác nhau. Ta cũng có thể giữ thông tin về lớp, màu sắc, kiểu đường trong một khối. Sau đó mỗi khi ta chèn khối, từng đối tượng sẽ được vẽ dựa trên màu sắc và kiểu đường ban đầu.

### 3.4.1. Thay đổi lớp của đối tượng

Sau khi đã tạo đối tượng và gán lớp, màu sắc và kiểu đường cho đối tượng, đôi khi ta cũng cần phải thay đổi lớp của đối tượng. Thay đổi lớp của đối tượng là cần thiết

vì đôi khi vô tình tạo ra đối tượng được nằm ở lớp khác khi quyết định tổ chức lại lớp sau này.

Để thay đổi lớp của đối tượng, sử dụng thuộc tính Layer của đối tượng đó. Thuộc tính Layer lấy tên làm thông số đầu vào.

### Di chuyển đối tượng sang lớp khác

Dưới đây là ví dụ tạo một đường tròn trên lớp hiện hành và tạo một lớp mới "ABC". Sau đó di chuyển đường tròn này sang lớp mới:

```
Sub Ch4_MoveObjectNewLayer()  
    ' Tạo đường tròn  
    Dim circleObj As AcadCircle  
    Dim center(0 To 2) As Double  
    Dim radius As Double  
    center(0) = 2: center(1) = 2: center(2) = 0  
    radius = 1  
    Set circleObj = ThisDrawing.ModelSpace. _  
        AddCircle(center, radius)  
    ' Tạo mới lớp "ABC"  
    Dim layerObj As AcadLayer  
    Set layerObj = ThisDrawing.Layers.Add("ABC")  
  
    ' Gán đường tròn và lớp "ABC"  
    circleObj.Layer = "ABC"  
    circleObj.Update  
End Sub
```

### 3.4.2. Thay đổi màu đối tượng

Ta có thể gán màu cho từng đối tượng trong bản vẽ. Mỗi màu được định nghĩa bằng số ACI, là một số từ 1 đến 255. Các màu chuẩn được đánh số từ 1 đến 7, là các màu: acRed, acYellow, acGreen, acCyan, acBlue, acMagenta, and acWhite

Thiết lập màu cho từng đối tượng sẽ ghi đè lên giá trị màu của lớp chứa đối tượng.

Nếu muốn giữ đối tượng trên một lớp nào đó nhưng lại không muốn giữ lại màu của lớp đó, ta có thể thay đổi màu của đối tượng. Để thay đổi màu, ta sử dụng thuộc tính Color của đối tượng đó.

### Thay đổi màu của đối tượng

Ví dụ sau tạo một đường tròn, sau đó chuyển thành màu đỏ.

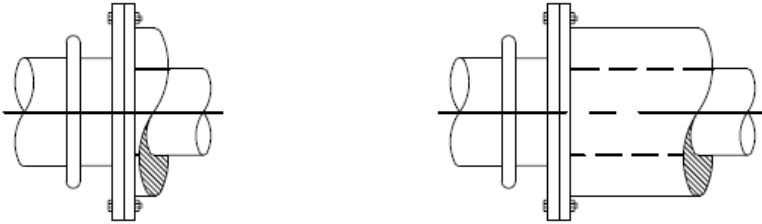
```
Sub Ch4_ColorCircle()  
    ' Tạo đường tròn  
    Dim circleObj As AcadCircle  
    Dim center(0 To 2) As Double  
    Dim radius As Double  
    center(0) = 2: center(1) = 2: center(2) = 0  
    radius = 1  
    Set circleObj = ThisDrawing.ModelSpace. _  
        AddCircle(center, radius)  
  
    ' Thay đổi màu của đường tròn thành màu đỏ  
    circleObj.Color = acRed  
    circleObj.Update  
End Sub
```



### 3.4.3. Thay đổi kiểu đường của đối tượng

Những đối tượng mới tạo sẽ sử dụng kiểu đường hiện hành. Nếu kiểu đường hiện hành là BYLAYER, đối tượng mới được tạo sẽ sử dụng kiểu đường của lớp hiện hành. Giá trị mặc định kiểu đường của đối tượng là BYLAYER.

Ta có thể gán kiểu đường cho tất cả các đối tượng trong AutoCAD nhưng kiểu đường sẽ không được hiển thị đối với các đối tượng: Text, Point, Viewport, Xlines, Ray, 3DPolyline Block. Nếu một đoạn thẳng quá ngắn, chỉ đủ để hiện một mẫu gạch ngang thì AutoCAD sẽ vẽ một đường liền nối hai đầu của nó.



Mặc định, đối tượng sẽ kế thừa kiểu đường của lớp chứa đối tượng. Sử dụng thuộc tính Linetype của đối tượng để thay đổi kiểu đường của chính đối tượng đó. Thuộc tính Linetype lấy tên của kiểu đường làm tham số đầu vào cho kiểu đường của đối tượng.

#### Thay đổi kiểu đường cho đối tượng

Ví dụ sau tạo một đường tròn. Sau đó cố gắng nhập kiểu đường "CENTER" từ tệp acad.lin. Nếu kiểu đường này đã có hoặc tệp không tồn tại, sẽ có xuất hiện thông báo. Cuối cùng đường tròn sẽ được gán kiểu đường là "CENTER".

```
Sub Ch4_ChangeCircleLinetype ()
    On Error Resume Next

    ' Tạo đường tròn
    Dim circleObj As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 2: center(1) = 2: center(2) = 0
    radius = 1
    Set circleObj = ThisDrawing.ModelSpace. _
        AddCircle(center, radius)
    Dim linetypeName As String
    linetypeName = "CENTER"

    ' Nạp kiểu đường "CENTER" từ tệp acad.lin
    ThisDrawing.Linetypes.Load linetypeName, "acad.lin"
    If Err.Description <> "" Then MsgBox Err.Description

    ' Gán kiểu đường "CENTER" cho đường tròn
    circleObj.Linetype = "CENTER"
    circleObj.
End Sub
```

---

**CHÚ Ý** Trước khi gán kiểu đường cho đối tượng, kiểu đường cần phải được tải vào bản vẽ hiện tại. Để tải kiểu đường vào bản vẽ, ta sử dụng phương thức Load.

---

## 4. Thêm văn bản vào bản vẽ

Văn bản là đối tượng dùng để truyền đạt những thông tin quan trọng trong bản vẽ. Văn bản dùng để đặt tiêu đề cho khối, tạo nhãn cho từng thành phần của bản vẽ, thể hiện quy định chung hoặc để làm ghi chú trong bản vẽ.

AutoCAD có nhiều cách khác nhau để tạo văn bản. Với những đoạn văn bản ngắn và đơn giản, ta có thể sử dụng dòng văn bản đơn. Với những đoạn văn bản dài hơn, có chứa định dạng riêng bên trong thì ta sử dụng văn bản nhiều dòng. Mặc dù tất cả các đoạn văn bản nhập vào đều sử dụng kiểu chữ hiện hành, với những thiết lập mặc định về phong chữ, định dạng nhưng cũng có nhiều cách khác nhau để tùy biến phần hiển thị của đoạn bản bản.

### 4.1. Làm việc với Kiểu chữ

Tất cả văn bản trong bản vẽ AutoCAD đều có kiểu chữ đi kèm. Khi nhập văn bản, AutoCAD sẽ sử dụng kiểu chữ hiện hành, với các thiết lập về phong chữ, cỡ, góc, hướng, và các đặc điểm khác của chữ. Kiểu chữ điều chỉnh những thuộc tính như trong bảng dưới đây:

#### Thuộc tính kiểu văn bản

Thuộc tính	Mặc định	Mô tả
Name	STANDARD	Tên có thể chứa 31 ký tự
Font File	<i>txt.shx</i>	Tệp chứa font
Big Font File	none	Tệp định nghĩa hình đặc biệt dùng cho bộ ký tự không phải dạng ASCII, như Kanji chẳng hạn.
Height	0	Chiều cao ký tự
Width	1	Co hoặc dẫn ký tự
Oblique angle	0	Góc nghiêng của ký tự
Text generation flag	No, No	Chữ lùi, chữ lộn ngược hoặc cả hai

Ta có thể sử dụng hoặc hiệu chỉnh kiểu mặc định hoặc có thể tạo và nạp một kiểu chữ mới. Sau khi tạo xong, ta có thể hiệu chỉnh các thuộc tính hoặc xóa kiểu chữ đi khi không cần nữa.

#### 4.1.1. Tạo và hiệu chỉnh Kiểu chữ

Ngoài kiểu mặc định là STANDARD, ta có thể tạo thêm nhiều kiểu chữ để sử dụng. Kiểu mới kế thừa chiều cao, hệ số bề rộng, góc nghiêng, và thuộc tính của kiểu chữ hiện hành. Để tạo kiểu chữ mới, sử dụng phương thức Add. Phương thức này sẽ tạo đối tượng TextStyle mới và thêm vào tập đối tượng TextStyles. Phương thức Add lấy tên của đối tượng TextStyle làm tham số đầu vào. Một khi đã tạo xong, ta không thể thay đổi tên của kiểu chữ thông qua AutoCAD ActiveX Automation được nữa.

Tên kiểu chữ có thể bao gồm chữ, số và các ký tự đặc biệt như dollar (\$), gạch chân (⏟), và gạch ngang (-). AutoCAD chuyển đổi các ký tự thành dạng in hoa. Nếu ta không nhập tên kiểu chữ, AutoCAD tự động đặt tên với dạng *Style $n$* , trong đó  $n$  là số nguyên bắt đầu từ 1. Mỗi kiểu mới sẽ tăng thêm 1 giá trị.

Những kiểu chữ đã có đều có thể được hiệu chỉnh với các thuộc tính của đối tượng TextStyle. Ta cũng có thể cập nhật những đoạn văn bản hiện có để hiển thị những thay đổi. Sử dụng những thuộc tính sau để hiệu chỉnh đối tượng TextStyle:

FontFile	Xác định tên tệp gắn với phong chữ nào đó (kiểu chữ).
BigFontFile	Tệp định nghĩa hình dạng đặc biệt của bộ ký tự không phải là ASCII, ví dụ như Kanji.
Height	Xác định chiều cao của ký tự.
Width	Xác định bề rộng ký tự (co hoặc giãn).
ObliqueAngle	Xác định độ nghiêng của ký tự.
TextGenerationFlag	Xác định kiểu chữ lùi, chữ lộn ngược hoặc cả hai.

Nếu thay đổi phong chữ hoặc hướng của một kiểu chữ, tất cả các văn bản sử dụng kiểu đó cũng sẽ thay đổi phong chữ hoặc hướng. Thay đổi chiều cao chữ, hệ số bề rộng, và độ nghiêng sẽ không thay đổi những văn bản đã có nhưng sẽ thay đổi những đối tượng văn bản được tạo sau này

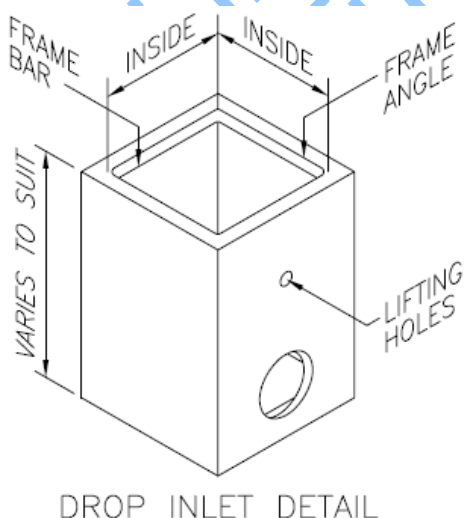
---

**CHÚ Ý** Cần phải gọi phương thức Regen hoặc Update để thấy được những thay đổi của các thuộc tính kể trên.

---

#### 4.1.2. Gán phong chữ

Phong chữ định nghĩa hình dáng của từng ký tự tạo nên bộ ký tự đó. Một phong chữ có thể sử dụng cho một hay nhiều kiểu chữ. Nếu ta có một phong chữ chuẩn, ta có thể tạo ra nhiều kiểu chữ khác nhau sử dụng cùng một phong chữ chuẩn để sử dụng cho những mục đích khác nhau. Ví dụ sau minh họa cùng một phong chữ được sử dụng trong nhiều kiểu chữ khác nhau (có độ nghiêng khác nhau):



Phông của kiểu chữ được gán thông qua thuộc tính FontFile của đối tượng TextStyle. Giá trị nhập vào thuộc tính FontFile là tên tệp chứa phông SHX đã được AutoCAD biên dịch.

### Thiết lập phông chữ

Ví dụ sau lấy các thiết lập về phông chữ của kiểu văn bản hiện hành, sau đó thay đổi kiểu phông chữ (tên phông) thành “.VnArial”. Sử dụng phương thức SetFont để gán phông chữ mới.

```
Sub Ch4_UpdateTextFont()  
    Dim typeFace As String  
    Dim Bold As Boolean  
    Dim Italic As Boolean  
    Dim charSet As Long  
    Dim PitchandFamily As Long  
  
    ' Lấy các thiết lập hiện hành để nhập các  
    ' giá trị mặc định cho phương thức SetFont  
    ThisDrawing.ActiveTextStyle.GetFont typeFace, _  
        Bold, Italic, charSet, PitchandFamily  
  
    ' Thay đổi kiểu phông chữ  
    typeFace = ".VnArial"  
    ThisDrawing.ActiveTextStyle.SetFont typeFace, _  
        Bold, Italic, charSet, PitchandFamily  
    ThisDrawing.Regen acActiveViewport  
End Sub
```

### 4.1.3. Sử dụng phông chữ TrueType

Phông chữ TrueType® luôn luôn hiển thị được tô đặc<sup>1</sup> trong bản vẽ, tuy nhiên, khi in thì chính biến hệ thống TEXTFILL mới điều khiển quá trình tô đặc cho các ký tự. Mặc định, biến TEXTFILL có giá trị là 1, nghĩa là các ký tự sẽ được tô đặc khi in. Khi xuất bản vẽ sang định dạng PostScript® sử dụng phương thức Export và in ra thiết bị in PostCript, phông chữ sẽ được in giống như trong thiết kế.

Để nâng cao tốc độ và hiệu quả của phông chữ dạng TrueType trong phiên bản AutoCAD này, hệ điều hành Windows vẽ trực tiếp một số văn bản TrueType. Tuy nhiên, do một số hạn chế trong Windows, AutoCAD phải vẽ phông TrueType được biến đổi theo một vài cách khác nhau, chẳng hạn như: văn bản đối xứng, chữ ngược, chữ lùi, nghiêng, chữ có tỷ lệ bề rộng khác 1, hoặc hướng không đồng phẳng với màn hình hiển thị. Quy tắc chung là phông chữ TrueType khi nhìn trong AutoCAD giống như khi nhìn trong các chương trình xử lý văn bản thì do Windows vẽ, còn lại là do AutoCAD tự vẽ. Những phông chữ đã được biến đổi khi hiển thị sẽ hơi đậm hơn trong một vài trường hợp, đặc biệt ở độ phân giải thấp. Sự khác nhau này chỉ xuất hiện khi hiển thị phông chữ nhưng thực chất sẽ không làm ảnh hưởng tới quá trình in ấn.

---

<sup>1</sup> Minh họa phông chữ đặc và rỗng: **Phông đặc**; Phông rỗng

#### 4.1.4. Sử dụng phông chữ Unicode và Big Font

AutoCAD hiện tại đã hỗ trợ bảng mã Unicode chuẩn. Một bộ phông chữ Unicode có thể chứa đến 65535 ký tự, với các mẫu cho nhiều ngôn ngữ. Phông chữ Unicode chứa nhiều ký tự hơn số ký tự được định nghĩa trong hệ thống. Vì vậy để sử dụng ký tự không thể nhập trực tiếp từ bàn phím, ta có thể nhập theo dạng thức sau  $\backslash U+nnnn$ , trong đó  $nnnn$  là chỉ số theo hệ mười sáu (hexadecimal) của ký tự Unicode. Tất cả mẫu phông chữ SHX của AutoCAD hiện nay đều là phông chữ Unicode.

Phông chữ SHX sử dụng trong phiên bản trước phiên bản 13 không hỗ trợ cách nhập theo dạng  $\backslash U+nnnn$ . Tuy nhiên, ta có thể tiếp tục phát sinh những ký tự khác nằm trong khoảng từ 128 đến 256.

Các tệp văn bản chứa bộ ký tự, chẳng hạn như Kanji, chứa hàng nghìn ký tự không phải ASCII. Để xử lý các văn bản như vậy, AutoCAD hỗ trợ tệp định nghĩa hình dạng phông đặc biệt gọi là tệp Big Font. Ta có thể thiết lập kiểu chữ sử dụng cả tệp phông bình thường và tệp Big Font.

Ví dụ về phông chữ chuẩn được cung cấp cùng AutoCAD, xem phụ lục E, “Standard Libraries” trong tài liệu “AutoCAD Command Reference”. AutoCAD còn hỗ trợ nhiều cách thay một phông chữ cho các phông khác hoặc để thiết lập phông mặc định, xin xem thêm phần “Thay thế phông chữ” trang 152.

Để thiết lập phông chữ thường, sử dụng thuộc tính FontFile. Còn đối với Big Fonts, ta sử dụng thuộc tính BigFontFile.

#### Thay đổi tệp phông chữ

Ví dụ sau sẽ thay đổi thuộc tính FontFile và BigFontFile. Cần phải thay đổi đường dẫn trong ví dụ này cho phù hợp với hệ thống hiện tại.

```
Sub Ch4_ChangeFontFiles ()
    ThisDrawing.ActiveTextStyle.BigFontFile = _
        "C:/AutoCAD/Fonts/bigfont.shx"
    ThisDrawing.ActiveTextStyle.fontFile = _
        "C:/AutoCAD/Fonts/italic.shx"
End Sub
```

---

**CHÚ Ý** Tên tệp dài có chứa dấu phẩy không được chấp nhận làm tên tệp chứa phông chữ.

#### 4.1.5. Thiết lập chiều cao chữ

Chiều cao chữ xác định kích thước theo đơn vị vẽ của ký tự có trong bộ phông chữ đang sử dụng. Giá trị này thường biểu diễn kích thước của ký tự in hoa, ngoại trừ trường hợp sử dụng phông chữ TrueType.

Với phông chữ TrueType, giá trị xác định chiều cao chữ có thể không thể hiện chính xác chiều cao của ký tự in hoa. Chiều cao này bao gồm cả chiều cao của ký tự in hoa và khoảng cách của dấu trong các ngôn ngữ khác, không phải tiếng Anh. Phần diện tích tương đối gán cho ký tự in hoa và dấu là do người thiết kế bộ phông chữ quyết định lúc thiết kế phông chữ và do đó giá trị này sẽ rất khác nhau với các bộ phông chữ khác nhau.

Bên cạnh phần chiều cao dành cho ký tự in hoa và vùng đặt dấu, bộ phông chữ TrueType còn có phần diện tích nằm phía dưới đường thẳng chèn văn bản, chẳng hạn như các ký tự y, j, g và q.

Thuộc tính Height dùng để xác định chiều chữ. Thuộc tính này chỉ chấp nhận số dương.

### Thay đổi chiều cao của đối tượng Text

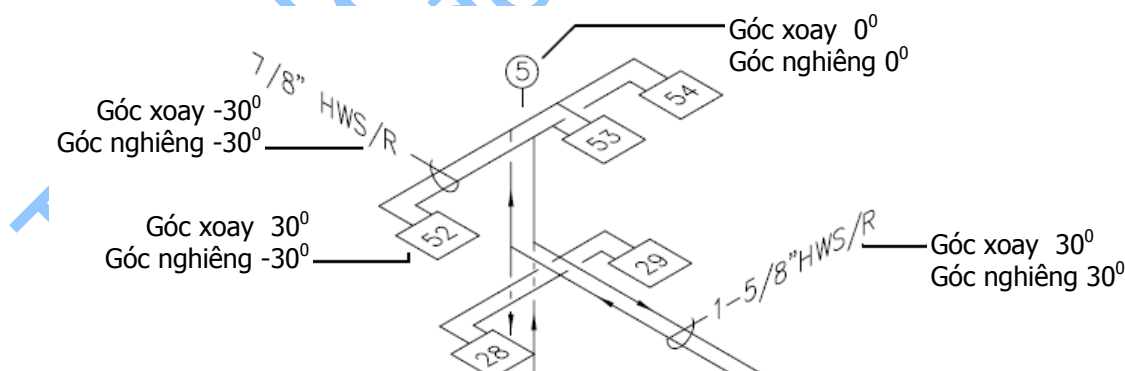
Ví dụ tạo một dòng văn bản và thay đổi chiều cao của văn bản.

```
Sub Ch4_ChangeTextHeight ()
    Dim textObj As AcadText
    Dim textString As String
    Dim insertionPoint(0 To 2) As Double
    Dim height As Double
    ' Định nghĩa đối tượng Text
    textString = "Hello, World."
    insertionPoint(0) = 3
    insertionPoint(1) = 3
    insertionPoint(2) = 0
    height = 0.5
    ' Tạo đối tượng Text trong không gian mô hình
    Set textObj = ThisDrawing.ModelSpace.
        AddText(textString, insertionPoint, height)

    ' Thay đổi giá trị thuộc tính Height thành 1
    textObj.height = 1
    textObj.Update
End Sub
```

### 4.1.6. Thiết lập góc nghiêng

Góc nghiêng sẽ xác định hướng nghiêng về phía trước hoặc phía sau của văn bản. Góc này chính là góc hợp với phương thẳng đứng.



Sử dụng thuộc tính ObliqueAngle để thiết lập góc nghiêng cho văn bản. Góc nghiêng nhập vào phải là giá trị radian. Góc dương làm chữ nghiêng về phía trước, giá trị góc âm sẽ được cộng với  $2 * \pi$  để chuyển thành giá trị dương tương đương.

### Tạo văn bản nghiêng

Ví dụ sau tạo đối tượng Text, sau đó làm nghiêng góc  $45^\circ$ .

```
Sub Ch4_ObliqueText ()
    Dim textObj As AcadText
    Dim textString As String
```

```

Dim insertionPoint(0 To 2) As Double
Dim height As Double

' Định nghĩa đối tượng Text
textString = "Hello, World."
insertionPoint(0) = 3
insertionPoint(1) = 3
insertionPoint(2) = 0
height = 0.5

' Tạo đối tượng Text trong không gian mô hình
Set textObj = ThisDrawing.ModelSpace. _
    AddText(textString, insertionPoint, height)

' Thay đổi giá trị thuộc tính ObliqueAngle
' thành 45 độ(0.707 radians)
textObj.ObliqueAngle = 0.707
textObj.Update
End Sub

```

#### 4.1.7. Thiết lập chế độ phát sinh văn bản

Chế độ phát sinh văn bản sẽ xác định xem văn bản có hiển thị lùi hoặc lộn ngược hay không.

Sử dụng thuộc tính `TextGenerationFlag` để thiết lập chế độ phát sinh văn bản. Để hiển thị văn bản viết lùi, ta gán giá trị `acTextFlagBackward`. Để hiển thị văn bản viết ngược, ta gán giá trị `acTextFlagUpsideDown`. Để hiển thị văn bản vừa viết lùi, vừa viết ngược, ta cộng cả hai hằng số này lại với nhau, `acTextFlagBackward + acTextFlagUpsidedown`, rồi gán cho thuộc tính này.

##### Hiển thị văn bản lùi

Ví dụ sau tạo một dòng văn bản, sau đó thiết lập để hiển thị viết lùi, sử dụng thuộc tính `TextGenerationFlag`.

```

Sub Ch4_ChangingTextGenerationFlag()
    Dim textObj As AcadText
    Dim textString As String
    Dim insertionPoint(0 To 2) As Double
    Dim height As Double

    ' Tạo đối tượng Text
    textString = "Hello, World."
    insertionPoint(0) = 3
    insertionPoint(1) = 3
    insertionPoint(2) = 0
    height = 0.5
    Set textObj = ThisDrawing.ModelSpace. _
        AddText(textString, insertionPoint, height)

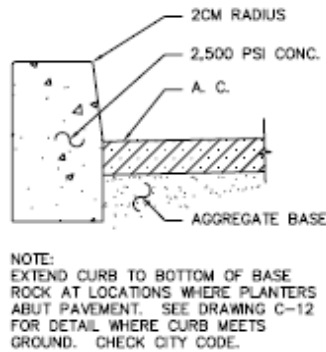
    ' Thay đổi giá trị thuộc tính the TextGenerationFlag
    textObj.TextGenerationFlag = acTextFlagBackward
    textObj.Update
End Sub

```



## 4.2. Sử dụng Văn bản đơn

Văn bản chèn trong bản vẽ truyền đạt rất nhiều thông tin khác nhau. Văn bản có thể dùng để thể hiện quy định chung, đặt tiêu đề cho khối, tạo nhãn và cũng có thể là một thành phần của bản vẽ. Đối với những văn bản ngắn không cần phải có nhiều phong chữ hoặc có nhiều dòng thì ta nên tạo văn bản đơn sử dụng đối tượng Text. Văn bản đơn rất thích hợp để tạo nhãn.



### 4.2.1. Tạo văn bản đơn

Khi sử dụng văn bản đơn thì mỗi dòng của văn bản là một đối tượng riêng biệt. Sử dụng phương thức AddText để tạo đối tượng văn bản đơn. Phương thức này cần phải nhập 3 giá trị: chuỗi văn bản, điểm chèn và chiều cao chữ.

Chuỗi văn bản là phần văn bản sẽ được thực sự hiển thị. Các ký tự đặc biệt, mã điều khiển, ký tự Unicode cũng được chấp nhận. Điểm chèn là một mảng 3 phần tử kiểu double thể hiện tọa độ toàn cục của điểm sẽ chèn văn bản trong bản vẽ. Chiều cao chữ là một số dương thể hiện chiều cao của ký tự in hoa. Chiều cao chữ được tính theo đơn vị hiện hành.

#### Tạo văn bản một dòng

Ví dụ sau sẽ tạo văn bản một dòng trong không gian mô hình tại vị trí (2,2,0).

```
Sub Ch4_CreateText()  
    Dim textObj As AcadText  
    Dim textString As String  
    Dim insertionPoint(0 To 2) As Double  
    Dim height As Double  
  
    ' Tạo đối tượng Text  
    textString = "Hello, World."  
    insertionPoint(0) = 2  
    insertionPoint(1) = 2  
    insertionPoint(2) = 0  
    height = 0.5  
    Set textObj = ThisDrawing.ModelSpace.  
        AddText(textString, insertionPoint, height)  
    textObj.Update  
End Sub
```

### 4.2.2. Định dạng văn bản đơn

Đối tượng Text được tạo ra sử dụng kiểu chữ hiện hành. Ta có thể thay đổi định dạng của đối tượng Text bằng cách thay đổi kiểu chữ, hoặc bằng cách thay đổi các

thuộc tính của đối tượng Text. Tuy nhiên, ta không thể thay đổi định dạng cho từng từ, từng chữ trong văn bản.

Để thay đổi kiểu chữ đi kèm với đối tượng Text, ta sẽ thiết lập thuộc tính StyleName thành một kiểu chữ mới. Sau khi đã thay đổi kiểu chữ, cần phải sử dụng phương thức Update của đối tượng Text để thấy được những thay đổi trong bản vẽ.

Ngoài những thuộc tính có thể hiệu chỉnh được cho tất cả các thực thể (màu, lớp, kiểu đường,...), ta còn có thể thay đổi các thuộc tính khác của đối tượng Text, bao gồm:

Alignment	Xác định chế độ canh hàng theo phương ngang và phương thẳng đứng cho văn bản.
InsertionPoint	Xác định điểm chèn của văn bản.
ObliqueAngle	Xác định góc nghiêng của từng đối tượng Text.
Rotation	Xác định góc xoay (tính bằng radian) của đối tượng Text.
ScaleFactor	Xác định hệ số tỷ lệ cho văn bản.
TextAlignmentPoint	Xác định điểm canh hàng của văn bản.
TextGenerationFlag	Xác định chế độ phát sinh văn bản: lùi, lộn ngược hoặc cả hai.
TextString	Xác định chuỗi văn bản thực sự được hiển thị.

Sau khi đã thay đổi các thuộc tính, cần phải sử dụng phương thức Update để hiển thị những thay đổi trong bản vẽ.

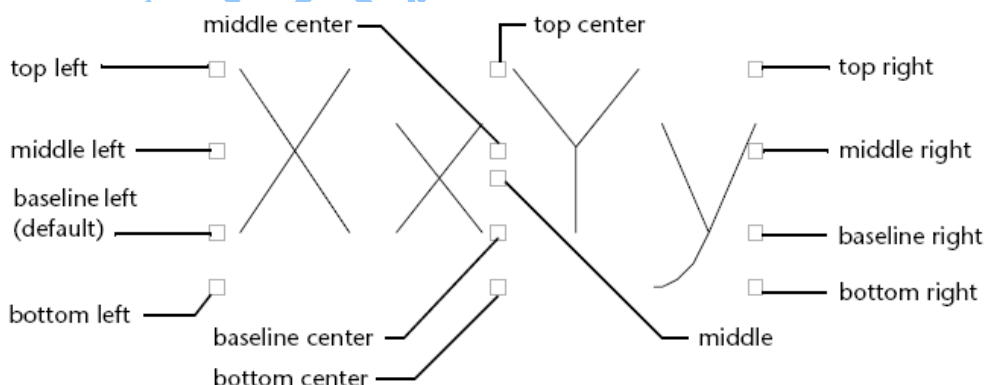
---

**CHÚ Ý** Biết tất cả các phương thức và thuộc tính của đối tượng Text, xin xem thêm tài liệu *AutoCAD ActiveX and VBA Reference*.

---

### 4.2.3. Canh hàng văn bản đơn

Ta có thể canh đều văn bản một dòng theo một hoặc hai phương đứng và ngang như thể hiện trong minh họa dưới đây. Chế độ canh trái là giá trị mặc định. Để thiết lập chế độ canh hàng theo phương ngang và dọc, ta sử dụng thuộc tính Alignment.



### Canh hàng lại văn bản

Ví dụ sau tạo đối tượng Text và đối tượng Point. Đối tượng Point được gán làm điểm canh hàng cho văn bản, và sau đó được chuyển thành dấu thập đỏ để ta có thể nhận biết rõ. Sau khi thay đổi chế độ canh lề, một hộp thông báo hiện lên nhằm tạm

ngừng quá trình thực thi Macro để ta có thể thấy rõ tác động của việc thay đổi chế độ canh hàng.

```
Sub Ch4_TextAlignment()  
    Dim textObj As AcadText  
    Dim textString As String  
    Dim insertionPoint(0 To 2) As Double  
    Dim height As Double  
    ' Định nghĩa đối tượng Text  
    textString = "Hello, World."  
    insertionPoint(0) = 3  
    insertionPoint(1) = 3  
    insertionPoint(2) = 0  
    height = 0.5  
    ' Tạo đối tượng text trong không gian mô hình  
    Set textObj = ThisDrawing.ModelSpace.  
        AddText(textString, insertionPoint, height)  
    ' Tạo điểm trùng với điểm canh hàng văn bản,  
    ' để ta có thể quan sát rõ quá trình thay đổi chế độ canh hàng  
    Dim pointObj As AcadPoint  
    Dim alignmentPoint(0 To 2) As Double  
    alignmentPoint(0) = 3  
    alignmentPoint(1) = 3  
    alignmentPoint(2) = 0  
    Set pointObj = ThisDrawing.ModelSpace.  
        AddPoint(alignmentPoint)  
    pointObj.Color = acRed  
    ' Gán kiểu điểm thành dạng chữ thập  
    ThisDrawing.SetVariable "PDMODE", 2  
  
    ' Canh trái đối tượng Text  
    textObj.Alignment = acAlignmentLeft  
    ThisDrawing.Regen acActiveViewport  
    MsgBox "The Text object is now aligned left"  
    ' Canh giữa đối tượng Text  
    textObj.Alignment = acAlignmentCenter  
    ' Canh hàng văn bản so với điểm (cần cho  
    ' tất cả ngoại trừ canh trái)  
    textObj.TextAlignmentPoint = alignmentPoint  
    ThisDrawing.Regen acActiveViewport  
    MsgBox "The Text object is now centered"  
    ' Canh phải đối tượng Text  
    textObj.Alignment = acAlignmentRight  
    ThisDrawing.Regen acActiveViewport  
    MsgBox "The Text object is now aligned right"  
End Sub
```

#### 4.2.4. Thay đổi văn bản đơn

Cũng giống như các đối tượng khác, ta có thể di chuyển, xoay, xóa và sao chép đối tượng văn bản. Ngoài ra, ta còn có thể lấy đối xứng đối tượng Text. Để đối tượng văn bản không bị đảo ngược khi lấy đối xứng, cần phải gán biến hệ thống MIRRTEXT bằng 0.

Dưới đây liệt kê các phương thức có trong đối tượng Text dùng để hiệu chỉnh đối tượng. Để có danh sách đầy đủ hơn, xin xem thêm tài liệu “*AutoCAD ActiveX and VBA Reference*”.

ArrayPolar	Nhân bản dạng cực.
ArrayRectangular	Nhân bản dạng chữ nhật.
Copy	Sao chép đối tượng Text.
Erase	Xóa đối tượng Text.
Mirror	Lấy đối xứng đối tượng Text.
Move	Di chuyển đối tượng Text.
Rotate	Xoay đối tượng Text.

### 4.3. Sử dụng Văn bản nhiều dòng

Đối với các đoạn văn bản dài và phức tạp, ta nên tạo đối tượng văn bản nhiều dòng – Mtext. Văn bản nhiều dòng có thể nằm trọn trong một bề rộng nhất định nhưng lại có thể mở rộng vô hạn theo chiều đứng. Đối tượng Mtext còn có thể được định dạng chi tiết đến từng từ hoặc từng ký tự.

Đối tượng văn bản nhiều dòng bao gồm nhiều dòng văn bản hoặc đoạn văn bản nằm trọn trong một bề rộng đã định trước. Mặc dù có nhiều dòng nhưng các đoạn văn được tạo trong một lần soạn thảo chỉ tạo thành một đối tượng. Đối tượng này có thể di chuyển, xoay, xóa, sao chép, lấy đối xứng, co giãn hoặc thay đổi tỷ lệ.

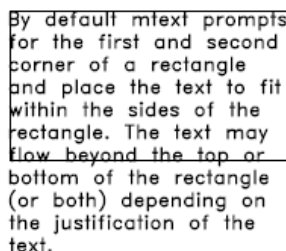
Văn bản nhiều dòng cung cấp nhiều lựa chọn trong quá soạn thảo hơn so với văn bản đơn. Ví dụ như ta có thể thiết lập đường gạch chân, phông, màu và chiều cao chữ cho từng ký tự, từng từ hoặc cụm từ trong một đoạn văn.

#### 4.3.1. Tạo văn bản nhiều dòng

Ta có thể tạo đối tượng văn bản nhiều dòng (đối tượng Mtext) bằng cách sử dụng phương thức AddMtext. Phương thức này cần phải nhập vào ba tham số: chuỗi ký tự, điểm chèn trong bản vẽ và chiều rộng của hình chữ nhật bao văn bản.

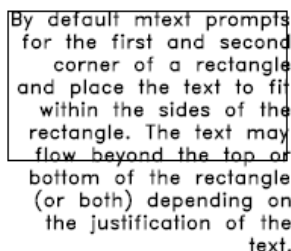
Chuỗi ký tự là đoạn văn bản sẽ thực sự được hiển thị. Ta có thể nhập các ký tự đặc biệt, mã điều khiển và ký tự Unicode. Điểm chèn là mảng gồm 3 phần tử kiểu double thể hiện tọa độ toàn cục của điểm, nơi sẽ chèn văn bản trong bản vẽ. Chiều rộng văn bản là một số dương thể hiện phần bề rộng của hình chữ nhật bao văn bản. Chiều rộng được đo theo hệ đơn vị hiện hành.

Sau khi đã tạo xong đối tượng Mtext, ta có thể thay đổi chiều cao chữ, chế độ canh hàng, góc nghiêng và kiểu văn bản cho đối tượng Mtext, hoặc có thể thiết lập định dạng cho từng ký tự. Chế độ canh hàng điều khiển cách phân bố văn bản tương đối so với hình chữ nhật bao.



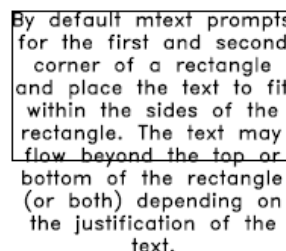
By default mtext prompts for the first and second corner of a rectangle and place the text to fit within the sides of the rectangle. The text may flow beyond the top or bottom of the rectangle (or both) depending on the justification of the text.

Canh trái



By default mtext prompts for the first and second corner of a rectangle and place the text to fit within the sides of the rectangle. The text may flow beyond the top or bottom of the rectangle (or both) depending on the justification of the text.

Canh phải



By default mtext prompts for the first and second corner of a rectangle and place the text to fit within the sides of the rectangle. The text may flow beyond the top or bottom of the rectangle (or both) depending on the justification of the text.

Canh giữa

Chiều cao của đối tượng MText phụ thuộc vào số ký tự trong chuỗi ký tự.

### Tạo đối tượng Mtext

Đoạn mã sau tạo đối tượng Mtext trong không gian mô hình tại điểm (2,2,0).

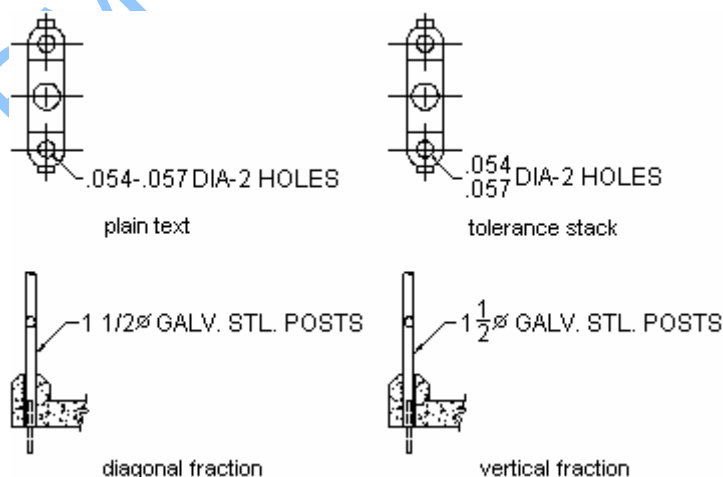
```
Sub Ch4_CreateMText()  
    Dim mtextObj As AcadMText  
    Dim insertPoint(0 To 2) As Double  
    Dim width As Double  
    Dim textString As String  
    insertPoint(0) = 2:insertPoint(1) = 2:insertPoint(2) = 0  
    width = 4  
    textString = "This is a text string for the mtext object."  
    ' Tạo đối tượng Mtext trong không gian mô hình  
    Set mtextObj = ThisDrawing.ModelSpace.  
    AddMText(insertPoint, width, textString)  
    ZoomAll  
End Sub
```

### 4.3.2. Định dạng văn bản nhiều dòng

Văn bản mới được tạo sẽ tự động nhận các thuộc tính của kiểu văn bản hiện hành, mặc định là kiểu STANDARD. Ta có thể thay đổi định dạng ban đầu bằng cách thiết lập định dạng cho từng ký tự và gán các thuộc tính cho đối tượng Text. Ta cũng có thể chỉ ra định dạng và các ký tự đặc biệt sử dụng một số phương thức sẽ được trình bày dưới đây.

Các lựa chọn định dạng, chẳng hạn chữ gạch chân, văn bản đặc biệt<sup>1</sup>, phông chữ có thể được áp dụng cho từng ký tự hoặc từng từ trong văn bản. Những lựa chọn chung như kiểu văn bản, chế độ canh hàng, bề rộng và góc xoay có tác động đến toàn thể đối tượng Mtext. Ta có thể thay đổi cả hai loại định dạng trên sử dụng các thuộc tính và phương thức của đối tượng Mtext.

<sup>1</sup> Văn bản đặc biệt – Stack text: là văn bản hoặc phân số hiển thị dung sai hoặc số đo nào đó. Cần phải nhập vào các ký tự đặc biệt trong đoạn văn bản để hiển thị văn bản đặc biệt, bao gồm dấu (/), (#) và (^). Dấu (/) thể hiện phân số dạng gạch ngang. Dấu (#) thể hiện phân số dạng gạch chéo. Dấu (^) thể hiện phân số sai (văn bản nằm trên 2 dòng) hoặc thể hiện số mũ. Hình dưới đây là minh họa các loại văn bản đặc biệt



### 4.3.2.1. Định dạng từng ký tự, từng từ

Ta có thể thiết lập định dạng cho từng từ hoặc từng ký tự bằng cách nhập vào ký tự ASCII tương đương với mã định dạng văn bản. Ta có thể: gạch chân văn bản, thêm một đường thẳng phía trên văn bản và tạo văn bản đặc biệt. Ngoài ra, ta còn có thể thay đổi màu, phông và chiều cao chữ hoặc có thể thêm vào khoảng trắng giữa các ký tự, tăng bề rộng của ký tự. Để thiết lập định dạng, ta phải sử dụng ký tự ASCII tương đương với mã định dạng:

#### Định dạng đối tượng Mtext

Mã định dạng	Mục đích	Nhập vào...	sẽ có kết quả là...
\0... \o	Bật/Tắt gạch trên	đường Autodesk \OAutoCAD\o 2000	Autodesk <u>AutoCAD</u> 2000
\L... \l	Bật/Tắt gạch dưới	đường Autodesk \LAutoCAD\l 2000	Autodesk <u>AutoCAD</u> 2000
\~	Chèn dấu không bị đồng	trắng Autodesk không bị xuống Autodesk đòng AutoCAD\~2000	Autodesk AutoCAD 2000
\\	Chèn dấu gạch ngược	Autodesk \\AutoCAD	Autodesk \AutoCAD
\{... \}	Chèn dấu móc	ngoặc Autodesk móc \{AutoCAD\} 2000	Autodesk {AutoCAD} 2000
\File name;	Thay đổi thành phông chữ có tên tệp như đã chỉ định	Autodesk \Ftimes; AutoCAD 2000	Autodesk AutoCAD
\Hvalue;	Thay đổi chiều cao chữ theo giá trị đơn vị vẽ	Autodesk \H2;AutoCAD	Autodesk <b>AutoCAD</b>
\Hvaluex;	Thay chiều cao chữ gấp x lần chiều cao hiện tại	Autodesk AutoCAD \H3x;2000	Autodesk AutoCAD 2000
\S... ^...;	Chèn văn bản chống với chữ đi kèm ký tự \, # và ^	1.000\S+0.010^- 0.000;	1,000 <sup>+0.010</sup> -0.000
\Tvalue;	Điều chỉnh khoảng cách các ký tự từ 0.75 đến 4	\T2;Autodesk	A u t o d e s k

## Định dạng đối tượng Mtext

Mã định dạng	Mục đích	Nhập vào...	sẽ có kết quả là...
<code>\Qangle;</code>	Thay đổi góc xiên	<code>\Q20;Autodesk</code>	<i>Autodesk</i>
<code>\Wvalue;</code>	Thay đổi hệ số bề rộng để tạo chữ rộng	<code>\W2;Autodesk</code>	<b>Autodesk</b>
<code>\A</code>	Thiết lập chế độ canh hàng theo chiều đứng; các giá trị: 0, 1, 2 (dưới, giữa, trên)	<code>\A1;1\S1/2</code>	$1\frac{1}{2}$

Sử dụng dấu ngoặc móc ({} ) để thiết lập định dạng chỉ cho đoạn văn bản bên trong dấu ngoặc. Có thể lồng được đến 8 cấp.

Ta cũng có thể nhập vào các ký tự ASCII tương đương với các ký tự điều khiển trong một dòng hoặc một đoạn văn để chỉ ra định dạng hoặc các ký tự đặc biệt, chẳng hạn như dung sai hoặc ký tự kích thước.

Các ký tự điều khiển sau sử dụng để tạo đoạn văn bản như trong minh họa ở dưới.

```
{{\H1.5x; Big text} \A2; over text\A1;/\A0; under text}
```

Big text<sup>over text/</sup>under text

### Sử dụng các ký tự điều khiển để định dạng văn bản

Ví dụ sau tạo một đối tượng Mtext có văn bản được định dạng như ở minh họa trên.

```
Sub Ch4_FormatMText()
    Dim mtextObj As AcadMText
    Dim insertPoint(0 To 2) As Double
    Dim width As Double
    Dim textString As String
    insertPoint(0) = 2
    insertPoint(1) = 2
    insertPoint(2) = 0
    width = 4
    ' Định nghĩa các ký tự ASCII làm ký tự điều khiển
    Dim OB As Long ' Mở ngoặc {
    Dim CB As Long ' Đóng ngoặc }
    Dim BS As Long ' Gạch ngược \
    Dim FS As Long ' Gạch xuôi /
    Dim SC As Long ' Chấm phẩy ;
    OB = Asc("{")
    CB = Asc("}")
    BS = Asc("\")
    FS = Asc("/")
    SC = Asc(";")
    ' Gán chuỗi văn bản các ký tự điều khiển và các ký tự thông thường
    ' đó là: {{\H1.5x; Big text}\A2; over text\A1;/\A0; under text}
    textString = Chr(OB) + Chr(OB) + Chr(BS) + "H1.5x" _
        + Chr(SC) + "Big text" + Chr(CB) + Chr(BS) + "\A2" _
```



```

+ Chr(SC) + "over text" + Chr(BS) + "A1" + Chr(SC)
+ Chr(FS) + Chr(BS) + "A0" + Chr(SC) + "under text"
+ Chr(CB)
' Tạo đối tượng Mtext trong không gian mô hình
Set mtextObj = ThisDrawing.ModelSpace.
AddMText(insertPoint, width, textString)
ZoomAll
End Sub

```

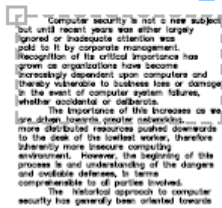
#### 4.3.2.2. Định dạng đối tượng văn bản nhiều dòng

Ta có thể thiết lập các thuộc tính điều khiển kiểu văn bản, chế độ canh lề, kích thước và góc xoay của văn bản. Những thiết lập này sẽ tác động lên tất cả văn bản bên trong đường bao văn bản chứ không phải chỉ tác động đến một từ hoặc một ký tự riêng lẻ nào cả.

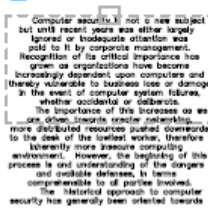
Thuộc tính `StyleName` thiết lập phong chữ và các đặc tính định dạng mặc định cho văn bản mới. Khi tạo đối tượng văn bản, ta có thể chọn kiểu văn bản mà mình cần dùng trong danh sách các kiểu văn bản hiện có.

Khi thay đổi kiểu văn bản của đối tượng Mtext có sử dụng định dạng riêng cho một phần nào đó của đoạn văn bản thì kiểu văn bản mới sẽ được áp cho toàn bộ đối tượng, và do đó định dạng riêng của một số ký tự nào đó có thể sẽ không được giữ nguyên. Ví dụ như khi ta thay đổi kiểu phông từ TrueType sang kiểu SHX hoặc một kiểu phông TrueType khác thì phông chữ mới đó sẽ được gán cho toàn bộ đối tượng văn bản và các định dạng riêng của các ký tự sẽ bị thay thế.

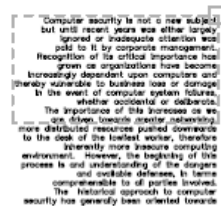
Chế độ canh lề điều khiển sự sắp xếp của văn bản dựa trên một điểm canh lề nào đó. Đoạn văn bản sẽ được canh đều dựa trên biên trái và phải của đường bao, còn các dòng văn bản sẽ được canh giữa, trên hoặc canh dưới dựa trên đường biên trên và dưới của đường bao văn bản. Đường biên trên và dưới được xác định dựa trên đường thẳng trên và dưới của đối tượng văn bản nhiều dòng. AutoCAD cung cấp cho người dùng 9 thiết lập về chế độ canh lề, bao gồm: Trên Trái (TL), Trên Giữa (TC), Trên Phải (TR), Giữa Trái (ML), Giữa Giữa (MC), Giữa Phải (MR), Dưới Trái (BL), Dưới Giữa (BC), Dưới Phải (BR).



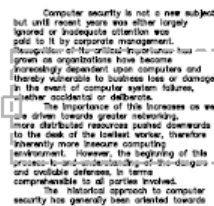
top left:  
left-aligned



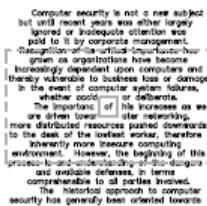
top center:  
centered



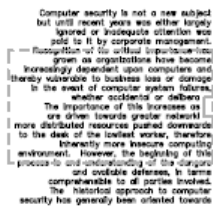
top right:  
right-aligned



middle left:  
left-aligned



middle center:  
centered



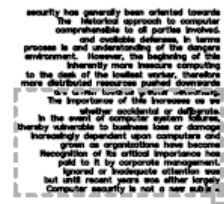
middle right:  
right-aligned



bottom left:  
left-aligned



bottom center:  
centered



bottom right:  
right-aligned

Sử dụng thuộc tính AttachmentPoint để thay đổi chế độ canh lề của đối tượng Mtext.

Thuộc tính Rotation điều khiển góc xoay của đường bao văn bản.

#### 4.4. Sử dụng ký tự Unicode, Ký tự điều khiển và Ký tự đặc biệt.

Ta có thể sử dụng các ký tự Unicode, ký tự điều khiển và ký tự đặc biệt trong chuỗi văn bản để hiển thị các ký hiệu. (Tất cả các ký tự không có trong bảng chữ cái buộc phải được nhập vào bằng mã ASCII tương đương).

Ta có thể tạo các ký tự đặc biệt sử dụng các chuỗi ký tự Unicode sau:

##### Ký tự Unicode

Ký tự Unicode	Mô tả
\U+00B0	Ký hiệu độ
\U+00B1	Ký hiệu dung sai dấu cộng/trừ
\U+2205	Ký hiệu ghi kích thước đường kính

Ngoài cách sử dụng các ký tự Unicode, ta có thể thể hiện các ký tự đặc biệt bằng cách thêm các thông tin điều khiển vào chuỗi ký tự văn bản bằng cách sử dụng cặp ký hiệu phần trăm (%%) để bắt đầu chuỗi điều khiển.

Mã điều khiển kiểu này tương thích với văn bản AutoCAD chuẩn và phong PostScript:

%%nnn      hiển thị ký tự số nnn.

Trong VB hoặc VBA, chuỗi trên có thể nhập như sau

```
Dim percent as Long
percent = ASC("%")
TextString = chr(percent) + chr(percent) + "nnn"
```

Các mã điều khiển sau chỉ tương thích với phong AutoCAD chuẩn:

##### Mô tả mã điều khiển

Mã điều khiển	Mô tả
---------------	-------

%%o	Bật/tắt chế độ gạch trên
%%u	Bật/tắt chế độ gạch dưới
%%d	Ký hiệu độ
%%p	Ký hiệu dung sai (dấu cộng/trừ)
%%c	Ký hiệu kích thước đường kính
%%%	Ký hiệu phần trăm

## 4.5. Thay thế phông chữ

Ta có thể chỉ định một phông chữ để thay thế cho các phông chữ khác hoặc để làm phông chữ mặc định khi AutoCAD không thể tìm thấy phông chữ nào đó trong bản vẽ.

Phông chữ sử dụng cho các văn bản trong bản vẽ được xác định dựa trên kiểu văn bản, và với đối tượng Mtext là dựa trên định dạng phông riêng biệt cho từng ký tự. Đôi lúc, để đảm bảo rằng bản vẽ chỉ sử dụng một số phông chữ nào đó, hoặc khi muốn chuyển các phông chữ đã sử dụng thành các phông chữ khác, ta có thể sử dụng các bộ soạn thảo văn bản để tạo bảng ánh xạ phông.

Ta có thể sử dụng bảng ánh xạ phông khi muốn chỉ sử dụng một bộ phông tiêu chuẩn, hoặc để tạo sự thuận tiện khi in. Ví dụ như khi chia sẻ bản vẽ với các nhà tư vấn, ta sẽ sử dụng bảng ánh xạ phông để xác định những phông chữ mà AutoCAD sẽ thay thế khi có những đoạn văn bản tạo với phông chữ khác. Tương tự, khi muốn soạn thảo bản vẽ sử dụng bộ phông hiện thị nhanh SHX và sau đó chuyển ngược lại thành các phông chữ khác để in ấn thì ta phải thiết lập bảng ánh xạ phông chữ để chuyển từng phông chữ SHX thành một phông chữ tương đương khác.

Bảng ánh xạ phông chữ là một tệp văn bản ASCII thuần túy chứa một ánh xạ phông trên một dòng. Mỗi dòng bao gồm: tên cơ sở của phông chữ (không bao gồm đường dẫn hoặc thư mục), tiếp sau đó là dấu chấm phẩy (;) và tên phông chữ thay thế. Tên phông chữ thay thế phải bao gồm cả phần mở rộng, chẳng hạn như *.ttf*.

Lấy ví dụ, ta có thể sử dụng mục sau trong bảng ánh xạ phông để chỉ ra rằng phông TrueType *times.ttf* sẽ thay thế cho phông *romanc.shx*:

```
Romanc.shx; times.ttf
```

AutoCAD luôn có một bảng ánh xạ phông chữ mặc định. Ta có thể hiệu chỉnh tệp này bằng cách sử dụng một chương trình soạn thảo văn bản ASCII thông thường. Ta cũng có thể chỉ định một bảng ánh xạ phông khác bằng cách sử dụng thuộc tính FontFileMap trong đối tượng Preferences.

### 4.5.1. Chỉ định phông thay thế mặc định

Nếu trong bản vẽ có chứa phông chữ không có trong hệ thống, AutoCAD sẽ tự động thay thế bằng phông chữ đã được chỉ định trước. Mặc định, AutoCAD sử dụng tệp *simplex.shx*. Tuy nhiên, ta cũng có thể chỉ định một phông chữ khác khi

cần bằng cách sử dụng thuộc tính AltFontFile của đối tượng Preferences để thiết lập tên tệp chứa phông chữ thay thế.

Nếu dùng kiểu văn bản có sử dụng Big Font, ta có thể ánh xạ sang một phông chữ khác sử dụng thuộc tính AltFontFile. Biến hệ thống này sử dụng một đôi tệp chứa tên phông là *txt.shx* và *bigfont.shx*.

Bảng sau thể hiện quy tắc thay thế phông chữ trong AutoCAD khi không tìm thấy phông chữ khi mở một bản vẽ.

<b>Quy tắc thay thế phông chữ</b>				
<b>Phần mở rộng</b>	<b>Thứ tự ánh xạ</b>			
	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
TTF	Sử dụng giá trị FONTMAP	Sử dụng phông định nghĩa trong kiểu văn bản	Windows thay thế một phông chữ tương tự	
SHX	Sử dụng giá trị FONTMAP	Sử dụng phông định nghĩa trong kiểu văn bản	Sử dụng FONTALT	Nhắc người dùng nhập phông mới
PFB	Sử dụng giá trị FONTMAP	Sử dụng FONTALT	Nhắc người dùng nhập phông mới	

#### 4.6. Kiểm tra chính tả

Trong suốt quá trình kiểm tra chính tả, AutoCAD so sánh từ trong bản vẽ với từ trong từ điển chính hiện hành. Bất kỳ từ nào người dùng thêm vào đều được lưu vào trong từ điển tùy biến hiện hành trong lúc kiểm tra chính tả. Ví dụ, ta có thể thêm vào một số tên thường dùng để AutoCAD không còn xem đó là những từ bị sai chính tả nữa.

Để kiểm tra chính tả với một ngôn ngữ khác, ta phải thay đổi sang một bộ từ điển chính khác.

AutoCAD ActiveX Automation không hỗ trợ các phương thức để thực hiện kiểm tra chính tả. Tuy nhiên, ta vẫn có thể gán tên của từ điển chính thông qua thuộc tính MainDictionary hoặc tên từ điển tùy biến thông qua thuộc tính CustomDictionary có trong đối tượng Preferences.

Tài liệu tham khảo  
BM Tự động hoá TKCĐ

# KÍCH THƯỚC VÀ DUNG SAI

Định kích thước là quá trình vẽ các đường kích thước cho bản vẽ. Còn dung sai xác định khoảng sai số cho phép của kích thước bản vẽ. Với ActiveX Automation, kích thước của bản vẽ được quản lý bằng các kiểu kích thước.

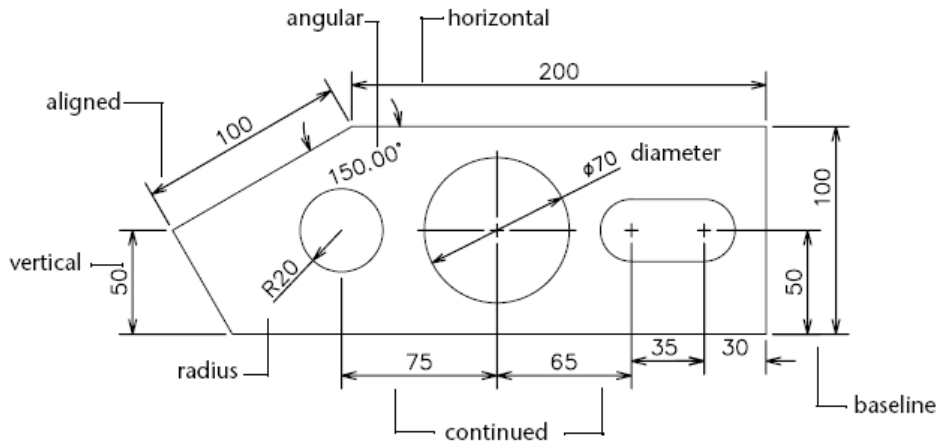


Trong chương này **5**

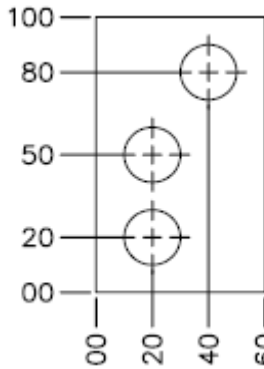
- **Khái niệm về kích thước**
- **Tạo kích thước**
- **Hiệu chỉnh kích thước**
- **Kiểu kích thước**
- **Kích thước trong không gian mô hình và không gian in**
- **Tạo đường dẫn và chú thích**
- **Tạo dung sai hình học**

# 1. Khái niệm về kích thước

Kích thước cho biết các số đo hình học của đối tượng như khoảng cách, góc giữa các đối tượng và tọa độ XY của một điểm. AutoCAD cung cấp 3 loại kích thước cơ bản: dạng đường, dạng tia và dạng góc. Kích thước dạng đường bao gồm các kiểu đo theo cạnh<sup>1</sup>, đo góc và kiểu tọa độ. Dưới đây là ví dụ về mỗi loại kích thước:



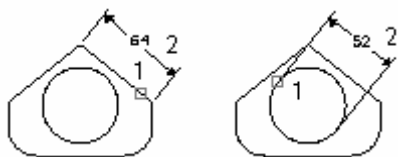
Ta có thể vẽ kích thước cho từng đường thẳng, nhiều đường thẳng một lúc, đường cong, đường tròn và các đoạn của đường đa tuyến hoặc cũng có thể xác định các kích thước đơn lẻ.



Kích thước theo kiểu tọa độ

AutoCAD vẽ kích thước trên lớp hiện hành. Mỗi kích thước đều thuộc một kiểu kích thước nhất định, có thể là mặc định hoặc tùy chọn. Kiểu kích thước xác định các đặc tính như màu, kiểu chữ, và tỷ lệ kiểu đường, nhưng lại không được hỗ trợ thông số về độ dày. Các họ đường kích thước cho phép tạo những kiểu kích thước

<sup>1</sup> Trong kích thước đo theo cạnh, đường kích thước song song với hai điểm gốc của đường kéo dài. Ví dụ dưới đây minh họa hai ví dụ về kích thước đo theo cạnh. Đầu tiên chọn đối tượng (1), sau đó xác định vị trí của kích thước đo theo cạnh (2). Đường dóng sẽ được vẽ tự động.

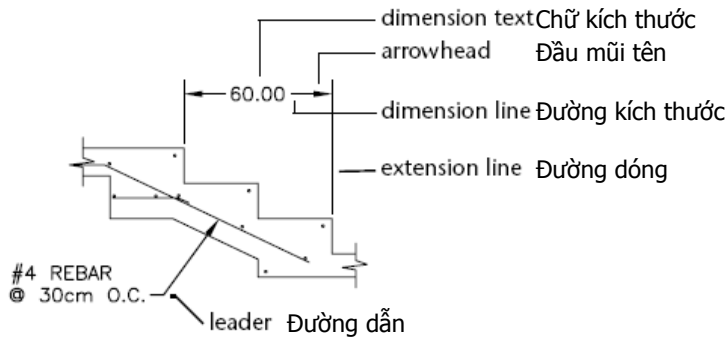




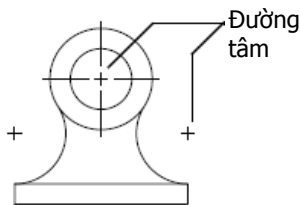
khác dựa trên một kiểu kích thước cơ bản. Chế độ ghi đè<sup>1</sup> cho phép chỉnh sửa chi tiết chỉ cho một kích thước cụ thể.

## 1.1. Thành phần của một kích thước

Phần này định nghĩa sơ lược về các thành phần của một kích thước.



Đường kích thước là một đường thẳng biểu thị hướng và độ rộng của một kích thước. Đối với kích thước đo góc, đường kích thước là một cung tròn. Đường dóng là một đường nối từ điểm được đo đến đường kích thước. Đầu mũi tên, còn gọi là ký hiệu điểm kết thúc hay điểm kết thúc, được thêm vào hai đầu của đường kích thước. Chữ kích thước là một chuỗi ký tự thường biểu thị số đo thực. Chuỗi ký tự cũng có thể bao gồm tiền tố, hậu tố và sai số. Đường dẫn là đường thẳng dẫn từ dòng chú thích đến đối tượng được chú thích. Dấu tâm là một dấu cộng nhỏ chỉ tâm của một đường tròn hay cung tròn. Đường tâm là các đường nét đứt xác định tâm của một đường tròn hay đường cung.



## 1.2. Định nghĩa biến hệ thống kích thước

Biến hệ thống kích thước điều kiện sự hiển thị của kích thước. Biến hệ thống kích thước gồm có: DIMAUNIT, DIMUPT, DIMTOFL, DIMFIT, DIMTIH, DIMTOH, DIMJUST, và DIMTAD. Ta có thể gán các biến này bằng cách sử dụng phương thức SetVariable. Ví dụ, đoạn mã sau gán biến hệ thống DIMAUNIT (đạng đơn vị cho kích thước góc) là radian (3).

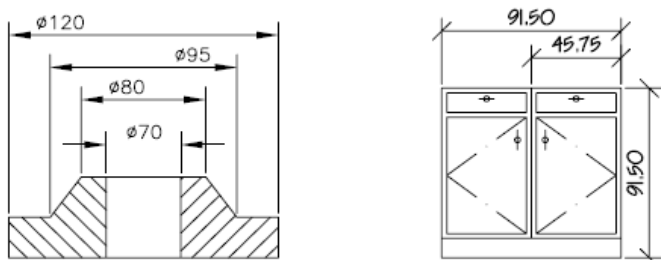
```
ThisDrawing.SetVariable "DIMAUNIT", 3
```

## 1.3. Thiết lập kiểu chữ cho kích thước

Chữ kích thước là tất cả các đối tượng văn bản gắn với kích thước, bao gồm số đo, sai số, tiền tố, hậu tố và các ghi chú một dòng hoặc nhiều dòng. Ta có thể sử dụng

<sup>1</sup> Ghi đè kiểu đường kích thước (dimension style override) là những thay đổi nhỏ cho một cấu hình nào đó cho kiểu kích thước hiện hành. Chế độ này tương đương với việc thay đổi biến hệ thống về đường kích thước mà không làm thay đổi kiểu kích thước hiện hành.

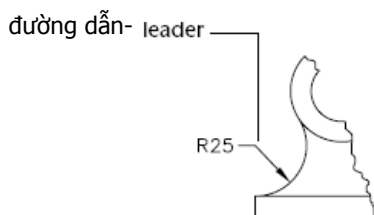
các số đo mặc định của AutoCAD làm chữ kích thước hoặc tự nhập vào hoặc có thể bỏ đi hoàn toàn. Ta cũng có thể thêm vào chữ kích thước các thông tin như quy trình sản xuất hoặc hướng dẫn lắp đặt.



Chữ kích thước một dòng sử dụng kiểu chữ hiện hành được xác định trong thuộc tính `ActiveTextStyle`. Chữ kích thước nhiều dòng cũng sử dụng kiểu chữ hiện hành cho các chuỗi văn bản đó.

### 1.4. Khái niệm về đường dẫn

Đường dẫn mặc định là một đường thẳng có một đầu mũi tên chỉ tới một đối tượng trong bản vẽ. Thông thường, chức năng của một đường dẫn là nối phần chú thích với đối tượng. Chú thích trong trường hợp này là một đoạn văn bản, khối hoặc khung điều chỉnh đối tượng. Các đường dẫn này khác với loại đường dẫn đơn giản do AutoCAD tự động tạo ra cho các kích thước bán kính, đường kính hay kích thước theo cạnh vốn có chuỗi ký tự không đặt giữa các đường dóng.



Các đối tượng đường dẫn được liên kết với chú thích, do đó khi chú thích được chỉnh sửa, đường dẫn cũng được cập nhật theo. Ta có thể sao chép chú thích được sử dụng ở một nơi khác trong bản vẽ và gắn thêm đường dẫn vào hoặc ta có thể tạo một chú thích mới. Ta cũng có thể tạo một đường dẫn mà không cần có chú thích.

### 1.5. Khái niệm về kích thước liên kết

Kích thước liên kết là kích thước mà trong đó tất cả các đường thẳng, đầu mũi tên, cung tròn và chữ kích thước đều được vẽ như một đối tượng kích thước đơn nhất. Biến hệ thống `DIMASO` điều khiển tính liên kết và mặc định là giá trị `on` (bật). Nếu biến `DIMASO` có giá trị là `off` (tắt), đường kích thước, đường dóng, đầu mũi tên, đường dẫn và chữ kích thước được vẽ như là các đối tượng riêng biệt. Ta có thể tạo kích thước không liên kết nếu cần thay đổi kích thước mà không bị các biến số chi phối. Tuy nhiên, nói chung, các kích thước liên kết dễ sử dụng hơn vì chúng được xem như là một đối tượng đơn nhất.

Để gán hoặc lấy biến hệ thống, sử dụng phương thức `SetVariable` và `GetVariable`.

## 2. Tạo kích thước

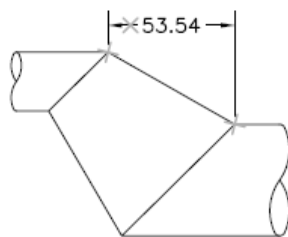
Ta có thể tạo kích thước dạng đường, dạng tia, dạng góc và kiểu tọa độ.

Khi tạo kích thước, kiểu kích thước hiện hành sẽ được sử dụng. Sau khi đã được tạo ra, ta có thể điều chỉnh góc của các đường dóng, vị trí chữ kích thước và nội dung chữ kích thước và góc hợp của nó so với đường kích thước. Ta cũng có thể thay đổi kiểu kích thước được sử dụng.

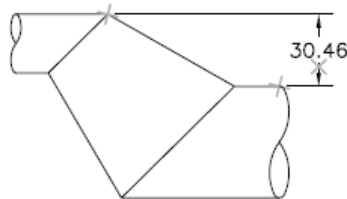
### 2.1. Tạo kích thước dạng đường

Các kích thước dạng đường có thể là kiểu đo theo cạnh hoặc kiểu đo nghiêng. Kích thước kiểu đo theo cạnh có đường kích thước song song góc của đường dóng. Các kích thước đo nghiêng có đường kích thước nằm nghiêng một góc so với góc của đường dóng.

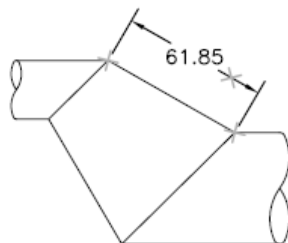
Để tạo một kích thước dạng đường, sử dụng các lệnh `AddDimAligned`, `AddDimRotated` hoặc `AddDim3PointAligned`. Sau khi đã tạo xong các kích thước đo thẳng, ta có thể điều chỉnh chuỗi ký tự, góc của chuỗi ký tự hoặc góc của đường kích thước. Hình vẽ sau hiển thị rõ góc của đường dóng cũng như vị trí đặt đường kích thước:



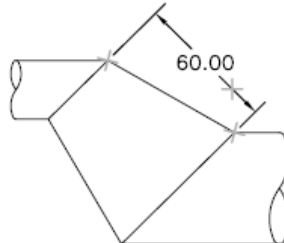
nằm ngang



thẳng đứng



theo cạnh



nghiêng góc 315 độ

Để tạo kích thước kiểu đo theo cạnh, ta sử dụng phương thức `AddDimAligned`. Phương thức này cần phải nhập vào 3 tọa độ: góc của hai đường dóng và vị trí của chuỗi ký tự.

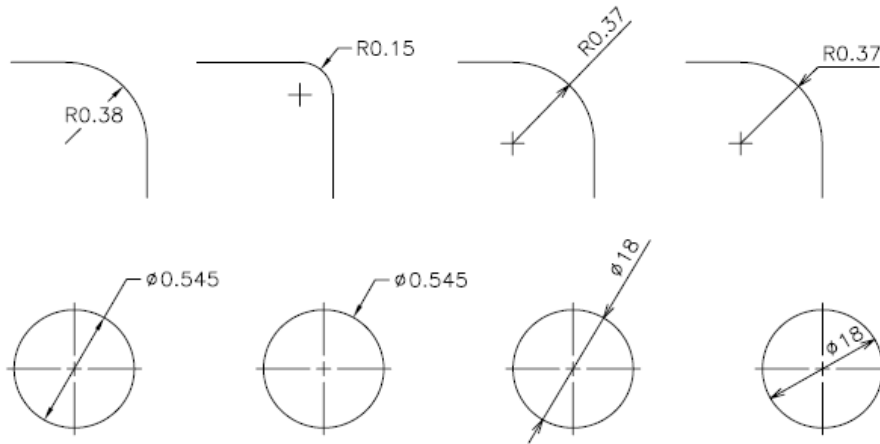
### 2.2. Tạo kích thước dạng tia

Kích thước dạng tia dùng để đo bán kính, đường kính của cung và đường tròn. Để tạo kích thước dạng tia, sử dụng phương thức `AddDimRadial`.

Các dạng khác nhau của kích thước dạng tia được tạo dựa trên kích cỡ của đường tròn hay đường cung, thuộc tính `TextPosition` (vị trí chuỗi ký tự) và các giá trị trong các biến hệ thống kích thước như `DIMUPT`, `DIMTOFL`, `DIMFIT`, `DIMTIH`, `DIMTOH`,

DIMJUST và DIMTAD. (Có thể lấy hoặc gán biến hệ thống bằng cách sử dụng phương thức GetVariable và SetVariable.)

Đối với chuỗi ký tự của kích thước nằm ngang, nếu góc nghiêng của đường kích thước lớn hơn 15 độ so với phương ngang và ở bên ngoài đường tròn hay cung tròn, AutoCAD vẽ một đường móc. Đường móc là một đường có mũi tên đặt bên cạnh chuỗi ký tự kích thước, như được minh họa dưới đây:



Để tạo kích thước dạng tia, sử dụng phương thức AddDimRadial hoặc AddDimDiametric. Các phương thức này cần phải nhập vào các giá trị: tọa độ tâm đường tròn hoặc cung tròn, tọa độ vị trí gắn đường dẫn và độ dài của đường dẫn.

Các phương thức này sử dụng tham số LeaderLength làm khoảng cách từ điểm gắn đường dẫn đến điểm mà kích thước sẽ tạo một đường móc nằm ngang chỉ đến dòng chú thích (hoặc dừng lại nếu không cần vẽ đường móc).

### Tạo một kích thước dạng tia

Ví dụ sau tạo đường kích thước dạng tia trong không gian mô hình.

```
Sub Ch5_CreateRadialDimension()
    Dim dimObj As AcadDimRadial
    Dim center(0 To 2) As Double
    Dim chordPoint(0 To 2) As Double
    Dim leaderLen As Integer

    ' Định nghĩa kích thước
    center(0) = 0
    center(1) = 0
    center(2) = 0
    chordPoint(0) = 5
    chordPoint(1) = 5
    chordPoint(2) = 0
    leaderLen = 5

    ' Tạo kích thước dạng tia trong không gian mô hình
    Set dimObj = ThisDrawing.ModelSpace. _
    AddDimRadial(center, chordPoint, leaderLen)
    ZoomAll
End Sub
```

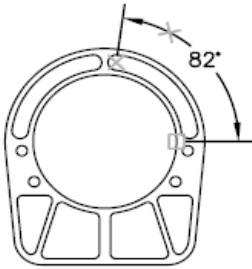
---

**CHÚ Ý** Việc đặt thông số LeaderLength chỉ được sử dụng trong quá trình tạo kích thước. Sau khi kích thước đã được thiết lập xong lần đầu, mọi thay đổi giá trị LeaderLength đều không có tác dụng, nhưng các thiết lập mới sẽ được lưu lại và hiển thị trong các ứng dụng DXF, LISP và ADSRX.

---

### 2.3. Tạo kích thước đo góc

Kích thước đo góc dùng để đo góc giữa hai đường thẳng hoặc giữa 3 điểm. Ví dụ, ta có thể sử dụng chúng để đo góc giữa hai đường bán kính của một đường tròn. Đường kích thước là một cung tròn.



Để tạo kích thước đo góc, sử dụng phương thức AddDimAngular. Phương thức này cần nhập 3 giá trị sau: đỉnh của góc, góc của các đường dóng, và vị trí của chuỗi ký tự. Đỉnh của góc là tâm của đường tròn hay đường cong, hoặc giao điểm của hai đường thẳng được đo. Góc của đường dóng là điểm mà hai đường dóng đi qua.

Đỉnh của góc có thể trùng với một trong những điểm góc. Các đường dóng sẽ được tự động thêm vào nếu cần.

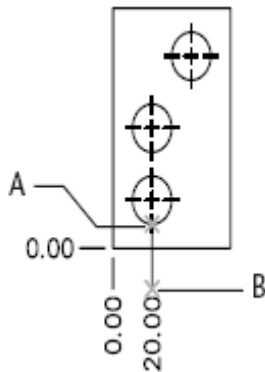
#### Tạo một kích thước dạng góc

Sau đây là ví dụ thiết lập một kích thước dạng góc trong không gian mô hình

```
Sub Ch5_CreateAngularDimension()  
    Dim dimObj As AcadDimAngular  
    Dim angVert(0 To 2) As Double  
    Dim FirstPoint(0 To 2) As Double  
    Dim SecondPoint(0 To 2) As Double  
    Dim TextPoint(0 To 2) As Double  
  
    ' Định nghĩa đường kích thước  
    angVert(0) = 0  
    angVert(1) = 5  
    angVert(2) = 0  
    FirstPoint(0) = 1  
    FirstPoint(1) = 7  
    FirstPoint(2) = 0  
    SecondPoint(0) = 1  
    SecondPoint(1) = 3  
    SecondPoint(2) = 0  
    TextPoint(0) = 3  
    TextPoint(1) = 5  
    TextPoint(2) = 0  
  
    ' Tạo đường kích thước đo góc trong không gian mô hình  
    Set dimObj = ThisDrawing.ModelSpace.  
    AddDimAngular(angVert, FirstPoint, SecondPoint, TextPoint)  
    ZoomAll  
End Sub
```

## 2.4. Tạo kích thước dạng tọa độ

Kích thước dạng tọa độ đo khoảng cách vuông góc từ một điểm, gọi là điểm mốc, đến đối tượng cần đo, như một lỗ trong một phần hình nào đó. Các kích thước này tránh được các sai số cộng dồn vì khoảng cách được đo trực tiếp từ điểm gốc đến đối tượng.



Kích thước dạng tọa độ bao gồm một tọa độ X hoặc Y và một đường dẫn. Kích thước dạng tọa độ theo trục X đo khoảng cách từ một đối tượng đến điểm mốc theo trục X. Kích thước dạng tọa độ theo trục Y đo khoảng cách đó theo với trục Y. AutoCAD sử dụng gốc của UCS hiện hành để xác định các tọa độ được đo và sử dụng giá trị tọa độ tuyệt đối.

Chuỗi ký tự được canh thẳng với tọa độ đường dẫn bất kể hướng của chuỗi ký tự này được quy định thế nào trong kiểu kích thước hiện hành. Ta có thể chấp nhận ký tự mặc định hoặc có thể điều chỉnh sau.

Để tạo một kích thước dạng tọa độ, sử dụng phương thức AddDimOrdinate. Phương thức này cần 3 giá trị: một tọa độ xác định điểm cần đo (A), một tọa độ xác định đầu mút của đường dẫn (B) và tham biến kiểu “Boolean” xác định kích thước là kích thước theo trục X hay Y. Nếu chọn TRUE cho “Boolean”, phương thức sẽ tạo một kích thước theo trục X, nếu chọn FALSE thì sẽ tạo một kích thước theo trục Y.

### Tạo một kích thước dạng tọa độ

```
Sub Ch5_CreatingOrdinateDimension ()
    Dim dimObj As AcadDimOrdinate
    Dim definingPoint(0 To 2) As Double
    Dim leaderEndPoint(0 To 2) As Double
    Dim useXAxis As Long
    ' Định nghĩa kích thước
    definingPoint(0) = 5
    definingPoint(1) = 5
    definingPoint(2) = 0
    leaderEndPoint(0) = 10
    leaderEndPoint(1) = 5
    leaderEndPoint(2) = 0
    useXAxis = 5
    ' Tạo kích thước dạng tọa độ trong không gian mô hình
    Set dimObj = ThisDrawing.ModelSpace. _
        AddDimOrdinate(definingPoint, _
            leaderEndPoint, useXAxis)
    ZoomAll
End Sub
```

### 3. Hiệu chỉnh kích thước

Cũng giống như các đối tượng đồ họa khác trong AutoCAD, ta có thể hiệu chỉnh kích thước bằng cách sử dụng các thuộc tính và phương thức chuẩn của đối tượng đó.

Các thuộc tính sau đây dùng cho hầu hết các đối tượng kích thước:

Rotation	Xác định góc nghiêng cho đường kích thước theo đơn vị radian
StyleName	Xác định tên của kiểu kích thước
Text	Xác định chuỗi ký tự của kích thước
TextPosition	Xác định vị trí của chuỗi ký tự kích thước
TextRotation	Xác định góc nghiêng của chuỗi ký tự kích thước
Measurement	Xác định số đo thực của kích thước

Bên cạnh đó, một số đối tượng kích thước nhất định có những thuộc tính khác dùng để hiệu chỉnh góc của đường đồng và chiều dài đường dẫn.

Các phương thức sau đây có thể sử dụng khi hiệu chỉnh đối tượng kích thước:

ArrayPolar	Nhân bản dạng cực
ArrayRectangular	Nhân bản dạng chữ nhật
Copy	Sao chép đối tượng kích thước
Erase	Xoá đối tượng kích thước
Mirror	Lấy đối xứng đối tượng kích thước
Move	Di chuyển đối tượng kích thước
Rotate	Xoay đối tượng kích thước
ScaleEntity	Cơ dẫn đối tượng kích thước

#### Ghi đè chuỗi ký tự kích thước

Giá trị kích thước được hiển thị có thể thay thế được bằng cách dùng thuộc tính TextOverride. Thuộc tính này có thể thay thế hoàn toàn giá trị hiển thị của kích thước, hoặc ta có thể thêm ký tự vào giá trị hiển thị. Ví dụ sau đây sẽ thêm một chuỗi ký tự vào giá trị kích thước để cả chuỗi ký tự đó và giá trị kích thước cùng được hiển thị.

```
Sub Ch5_OverrideDimensionText()  
    Dim dimObj As AcadDimAligned  
    Dim point1(0 To 2) As Double  
    Dim point2(0 To 2) As Double  
    Dim location(0 To 2) As Double  
    ' Định nghĩa kích thước  
    point1(0) = 5#: point1(1) = 3#: point1(2) = 0#  
    point2(0) = 10#: point2(1) = 3#: point2(2) = 0#  
    location(0) = 7.5: location(1) = 5#: location(2) = 0#  
    ' Tạo kích thước đo theo cạnh trong không gian mô hình  
    Set dimObj = ThisDrawing.ModelSpace.  
        AddDimAligned(point1, point2, location)  
    ' Thay đổi chuỗi ký tự của kích thước
```



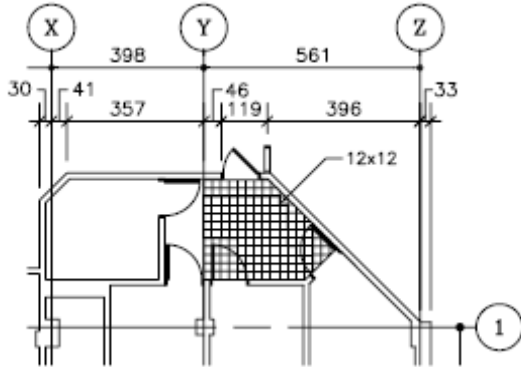
```

dimObj.TextOverride = "The value is <>"
dimObj.Update
End Sub

```

## 4. Kiểu kích thước

Một kiểu kích thước có đặt tên là một nhóm các thiết lập quy định hình thức hiển thị của kích thước. Với các kiểu kích thước có đặt tên, ta có thể thiết lập và áp dụng các tiêu chuẩn phác thảo cho bản vẽ.



Tất cả các kích thước đều được tạo ra đều dựa trên kiểu kích thước hiện hành. Nếu không định nghĩa hoặc áp dụng một kiểu nào đó trước khi tạo kích thước, AutoCAD sẽ sử dụng kiểu mặc định là kiểu STANDARD. Để thiết lập kiểu kích thước hiện hành, ta sử dụng thuộc tính ActiveDimStyle.

Để tạo một kiểu kích thước mẫu, trước hết phải đặt tên và lưu kiểu đó. Kiểu kích thước mới được dựa trên kiểu hiện hành và bao gồm tất cả những thay đổi cho cách bố trí các yếu tố kích thước (hộp thoại DDIM Geometry), vị trí của chuỗi ký tự (hộp thoại DDIM Format) và hình thức hiển thị của chú thích (hộp thoại DDIM Annotation). Chú thích trong trường hợp này có nghĩa là các đơn vị, sai số, ký tự chính và phụ.

Để tạo một kiểu kích thước mới, sử dụng phương thức Add. Phương thức này cần phải nhập tên của kiểu kích thước mới.

AutoCAD ActiveX Automation cho phép thêm các kiểu kích thước mới và thay đổi kiểu kích thước hiện hành. Ta cũng có thể thay đổi kiểu kích thước của một kích thước có sẵn qua thuộc tính StyleName.

Ta cũng có thể sao chép một kiểu kích thước đã có hoặc thiết lập các kiểu kích thước ghi đè<sup>1</sup>. Sử dụng phương thức CopyFrom để sao chép một kiểu kích thước từ một đối tượng nguồn thành một kiểu kích thước mới. Đối tượng nguồn có thể là một đối tượng DimStyle khác, kích thước, một đối tượng Tolerance hoặc Leader hoặc thậm chí là một đối tượng Document. Nếu sao chép các thiết lập từ một kiểu kích thước khác, kiểu này sẽ được nhân đôi chính xác. Nếu sao chép các thiết lập từ một kích thước khác, các đối tượng Tolerance hoặc Leader, cùng với kích thước ghi

<sup>1</sup> **Kiểu kích thước ghi đè (Dimension Override):** là một thay đổi cấu hình nào đó của kiểu kích thước hiện hành. Điều này tương đương với việc thay đổi biến hệ thống kích thước mà không làm thay đổi kiểu kích thước hiện hành.

đề cũng được sao chép vào kiểu mới. Nếu sao chép kiểu của một đối tượng Document, kiểu kích thước hiện hành cùng với các kiểu kích thước ghi đề trong bản vẽ sẽ được sao chép vào kiểu kích thước mới.

### Sao chép các kiểu kích thước và kích thước ghi đề

Ví dụ sau đây tạo ba kiểu kích thước mới và lần lượt sao chép những thiết lập của kiểu kích thước từ một đối tượng trong bản vẽ, từ một kiểu kích thước và từ những thiết lập hiện hành trong bản vẽ. Thực hiện theo đúng trình tự sau đây, ta sẽ hiểu được cách thức tạo một kiểu kích thước mới dựa trên những thiết lập đã có.

- 1 Tạo một bản vẽ mới và đặt làm bản vẽ hiện hành.
- 2 Tạo một kích thước dạng đường trong bản vẽ mới. Kích thước này là đối tượng duy nhất trong bản vẽ.
- 3 Sử dụng OPM (Object Properties Manager), thay đổi màu của đường kích thước thành màu vàng.
- 4 Thay đổi biến số hệ thống DIMCLRD thành 5 (màu xanh)
- 5 Thực thi ví dụ sau:

```
Sub Ch5_CopyDimStyles()  
    Dim newStyle1 As AcadDimStyle  
    Dim newStyle2 As AcadDimStyle  
    Dim newStyle3 As AcadDimStyle  
    Set newStyle1 = ThisDrawing.DimStyles.Add _  
        ("Style 1 copied from a dim")  
    Call newStyle1.CopyFrom(ThisDrawing.ModelSpace(0))  
    Set newStyle2 = ThisDrawing.DimStyles.Add _  
        ("Style 2 copied from Style 1")  
    Call newStyle2.CopyFrom(ThisDrawing.DimStyles.Item _  
        ("Style 1 copied from a dim"))  
    Set newStyle2 = ThisDrawing.DimStyles.Add _  
        ("Style 3 copied from the running drawing values")  
    Call newStyle2.CopyFrom(ThisDrawing)  
End Sub
```

Khi mở hộp thoại DIMSTYLE sẽ có ba kiểu kích thước được liệt kê: kiểu 1 có một đường kích thước màu vàng, kiểu 2 giống kiểu 1, kiểu 3 có đường kích thước màu xanh.

#### 4.1. Kiểu kích thước ghi đề

Mỗi kích thước đều có khả năng ghi đề những một số thiết lập nào đó của chính bản thân kiểu kích thước đó để tạo thành kiểu kích thước ghi đề. Các thuộc tính sau có trong hầu hết các đối tượng kích thước:

AltRoundDistance Quy định sự làm tròn của các đơn vị thay thế.

AngleFormat Quy định định dạng đơn vị của kích thước dạng góc.

Arrowhead1Block, Arrowhead2Block

Quy định khối tạo nên đầu mũi tên của đường kích thước.

Arrowhead1Type, Arrowhead2Type

Quy định dạng đầu mũi tên của đường kích thước.

ArrowheadSize	Quy định cỡ đầu mũi tên của đường kích thước, đường dẫn và đường móc.
CenterMarkSize	Quy định cỡ của dấu tâm cho các kích thước dạng tia.
CenterType	Quy định dạng của dấu tâm cho kích thước dạng tia.
DecimalSeparator	Quy định ký tự dùng làm dấu cách thập phân trong kích thước thập phân và các giá trị dung sai.
DimensionLineColor	Quy định màu cho đường kích thước của một kích thước, đối tượng đường dẫn hoặc dung sai.
DimensionLineWeight	Quy định độ dày của đường kích thước
DimLine1Suppress, DimLine2Suppress	Quy định sự hiển thị (có/không) của đường kích thước.
DimLineInside	Quy định chỉ hiển thị đường kích thước trong vòng các đường dóng.
ExtensionLineColor	Quy định màu của các đường dóng.
ExtensionLineExtend	Quy định khoảng cách từ đường dóng đến đường kích thước.
ExtensionLineOffset	Quy định khoảng cách từ đường dóng đến điểm góc của đường dóng.
ExtensionLineWeight	Quy định độ dày của đường dóng.
ExtLine1EndPoint, ExtLine2EndPoint	Xác định điểm cuối của đường dóng.
ExtLine1StartPoint, ExtLine2StartPoint	Xác định điểm khởi đầu của đường dóng.
ExtLine1Suppress, ExtLine2Suppress	Quy định hiển thị (có/không) đường dóng.
Fit	Quy định vị trí của chuỗi ký tự và mũi tên ở trong hay ngoài các đường dóng.
ForceLineInside	Quy định một đường kích thước có được vẽ giữa các đường dóng hay không ngay cả khi chuỗi ký tự được đặt bên ngoài các đường mở rộng.
FractionFormat	Quy định định dạng của các giá trị phân số trong kích thước và dung sai
HorizontalTextPosition	Quy định canh lề ngang cho chuỗi ký tự kích thước

LinearScaleFactor	Quy định hệ số tỷ lệ toàn cục cho các số đo kích thước dạng đường.
PrimaryUnitsPrecision	Quy định số chữ số thập phân hiển thị trong đơn vị chính của kích thước hoặc dung sai.
SuppressLeadingZeros, SuppressTrailingZeros	Quy định làm ẩn các số 0 ở trước hoặc ở sau của các số trong các giá trị kích thước.
SuppressZeroFeet, SuppressZeroInches	Quy định làm ẩn các số đo 0 foot và 0 inche trong các giá trị kích thước.
TextColor	Quy định màu của chuỗi ký tự trong đối tượng kích thước và dung sai.
TextGap	Quy định khoảng cách giữa chuỗi ký tự kích thước và đường kích thước khi ngắt đường kích thước để chèn chuỗi ký tự vào.
TextHeight	Quy định độ cao của chuỗi ký tự kích thước hoặc dung sai.
TextInside	Quy định chuỗi ký tự có/không xuất hiện ở trong đường dóng.
TextInsideAlign	Quy định vị trí của chuỗi kích thước là nằm trong đường dóng cho tất cả các loại kích thước trừ loại theo toạ độ.
TextMovement	Quy định cách thức vẽ chuỗi ký tự kích thước khi di chuyển chuỗi ký tự.
TextOutsideAlign	Quy định vị trí của chuỗi ký tự kích thước bên ngoài đường dóng cho tất cả các loại kích thước trừ loại theo toạ độ.
TextPosition	Quy định vị trí của chuỗi ký tự kích thước.
TextPrecision	Quy định độ chính xác của chuỗi ký tự kích thước dạng góc.
TextPrefix	Quy định tiền tố giá trị kích thước.
TextRotation	Quy định góc nghiêng của chuỗi ký tự kích thước.
TextSuffix	Quy định hậu tố giá trị kích thước.
ToleranceDisplay	Quy định dung sai có hiển thị cùng chuỗi ký tự kích thước hay không.
ToleranceHeightScale	Quy định hệ số tỷ lệ cho độ cao của chuỗi ký tự dung sai so với độ cao của chuỗi ký tự kích thước.
ToleranceJustification	Quy định chế độ canh hàng thẳng đứng của các giá trị dung sai so với chuỗi ký tự kích thước.
ToleranceLowerLimit	Quy định giới hạn dung sai nhỏ nhất cho chuỗi ký tự kích thước.

TolerancePrecision

Quy định độ chính xác của các giá trị dung sai trong giá trị kích thước chính.

ToleranceSuppressLeadingZeros

Quy định làm ẩn các số 0 ở đầu giá trị dung sai.

ToleranceSuppressTrailingZeros

Quy định làm ẩn các số 0 ở cuối giá trị dung sai.

ToleranceUpperLimit

Quy định dung sai lớn nhất cho chuỗi ký tự kích thước.

UnitsFormat

Quy định định dạng đơn vị cho tất cả các kích thước trừ kích thước dạng góc.

VerticalTextPosition

Quy định vị trí thẳng đứng của chuỗi ký tự so với đường kích thước.

### Nhập hậu tố do người dùng định nghĩa vào một kích thước đo theo cạnh

Ví dụ dưới đây tạo một kích thước đo theo cạnh trong không gian mô hình và sử dụng thuộc tính TextSuffix để cho phép người dùng thay đổi hậu tố của chuỗi ký tự kích thước.

```
Sub Ch5_AddTextSuffix()  
    Dim dimObj As AcadDimAligned  
    Dim point1(0 To 2) As Double  
    Dim point2(0 To 2) As Double  
    Dim location(0 To 2) As Double  
    Dim suffix As String  
  
    ' Định nghĩa kích thước  
    point1(0) = 0: point1(1) = 5: point1(2) = 0  
    point2(0) = 5: point2(1) = 5: point2(2) = 0  
    location(0) = 5: location(1) = 7: location(2) = 0  
    ' Tạo kích thước đo theo cạnh trong không gian mô hình  
    Set dimObj = ThisDrawing.ModelSpace.  
        AddDimAligned(point1, point2, location)  
    ThisDrawing.Application.ZoomAll  
    ' Cho phép người dùng nhập vào hậu tố  
    suffix=InputBox("Enter a new text suffix for the dimension" _  
        , "Set Dimension Suffix", ":SUFFIX")  
  
    ' Áp dụng thay đổi cho các kích thước  
    dimObj.TextSuffix = suffix  
    ThisDrawing.Regen acAllViewports  
End Sub
```

## 5. Kích thước trong không gian mô hình và không gian in

Ta có thể vẽ kích thước trong cả không gian mô hình và không gian in. Tuy nhiên, nếu đối tượng đồ họa đang định kích thước nằm trong không gian mô hình thì nên

về kích thước trong không gian mô hình, bởi vì AutoCAD sẽ đặt các điểm định nghĩa trong không gian mà đối tượng đồ họa được vẽ.

Nếu một đường kích thước vẽ trong không gian in mô tả đối tượng đồ họa trong không gian mô hình thì đường kích thước trong không gian in sẽ không thay đổi mỗi khi hiệu chỉnh hoặc thay đổi độ phóng đại của khung nhìn trong không gian in. Vị trí của đường kích thước trong không gian in luôn giữ nguyên khi thay đổi từ không gian in sang không gian mô hình.

Nếu vẽ kích thước trong không gian in và hệ số tỷ lệ chung của kích thước dạng đường (biến hệ thống DIMLFAC) là nhỏ hơn 0 thì khoảng cách đo được sẽ được nhân lên với giá trị tuyệt đối của DIMLFAC. Nếu vẽ kích thước trong không gian mô hình, giá trị 1.0 được sử dụng ngay cả khi DIMLFAC nhỏ hơn 0. AutoCAD tính toán giá trị của DIMLFAC nếu ta thay đổi biến số ở dòng lệnh Dim và chọn tùy chọn Viewport. AutoCAD tính tỷ lệ của không gian mô hình với không gian in và gán dấu âm của giá trị này cho DIMLFAC.

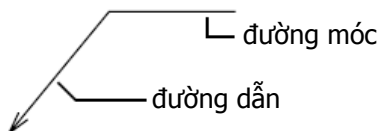
## 6. Tạo đường dẫn và chú thích

Một đường dẫn là một đường nối từ chú thích đến một đối tượng trong bản vẽ. Đường dẫn và chú thích luôn liên kết với nhau, nghĩa là nếu chỉnh sửa chú thích thì đường dẫn cũng sẽ thay đổi theo. Không nên nhầm lẫn giữa đối tượng đường dẫn với đường dẫn AutoCAD tự động tạo ra trong một đường kích thước.

### 6.1. Tạo đường dẫn

Ta có thể tạo một đường dẫn từ bất cứ điểm nào hoặc đối tượng nào trong bản vẽ và điều khiển sự hiển thị trong khi vẽ. Đường dẫn có thể là đoạn thẳng hoặc các đường cong Spline. Màu của các đường dẫn được lấy theo màu của đường kích thước hiện hành. Tỷ lệ các đường dẫn cũng được điều khiển thông qua hệ số tỷ lệ chung được thiết lập trong kiểu kích thước hiện hành. Loại và cỡ của đầu mũi tên, nếu có, được lấy giống như mũi tên thứ nhất định nghĩa trong kiểu kích thước hiện hành.

Một đoạn thẳng nhỏ gọi là đường móc thường nối chú thích với đường dẫn. Đường móc xuất hiện với đối tượng Mtext và khung điều chỉnh nếu đoạn đường dẫn cuối cùng ở một góc lớn hơn 15 độ so với phương ngang. Nếu đường dẫn không có chú thích sẽ không có đường móc.



Phương thức AddLeader dùng để tạo mới một đường dẫn. Phương thức này cần phải nhập ba giá trị: mảng tọa độ để đặt đường dẫn, chú thích (hoặc NULL nếu đường dẫn không có chú thích) và loại đường dẫn muốn tạo. Loại đường dẫn quy định đường dẫn là đường thẳng hay đường cong Spline và quyết định đường dẫn có mũi tên hay không. Sử dụng một trong những hằng số sau đây để xác định loại của đường dẫn: acLineNoArrow, acLineWithArrow, acSplineNoArrow, hoặc

acSplineWithArrow. Các hằng số này độc lập với nhau.

## Tạo mới đường dẫn

Ví dụ sau đây tạo một đường dẫn trong không gian mô hình. Không có chú thích gắn với đường dẫn.

```
Sub Ch5_CreateLeader()  
    Dim leaderObj As AcadLeader  
    Dim points(0 To 8) As Double  
    Dim leaderType As Integer  
    Dim annotationObject As AcadObject  
    points(0) = 0: points(1) = 0: points(2) = 0  
    points(3) = 4: points(4) = 4: points(5) = 0  
    points(6) = 4: points(7) = 5: points(8) = 0  
    leaderType = acLineWithArrow  
    Set annotationObject = Nothing  
    ' Tạo đường dẫn trong không gian mô hình  
    Set leaderObj = ThisDrawing.ModelSpace.  
        AddLeader(points, annotationObject, leaderType)  
    ZoomAll  
End Sub
```

## 6.2. Thêm chú thích vào đường dẫn

Chú thích của đường dẫn có thể là đối tượng Tolerance, Mtext hoặc BlockRef. Ta có thể tạo một chú thích mới, hoặc thêm vào một chú thích có sẵn. Chú thích được thêm vào đường dẫn chỉ khi nào nó được tạo ra.

Để thêm một chú thích khi tạo đường dẫn, nhập chú thích trong phương thức AddLeader.

## 6.3. Liên kết của đường dẫn

Đường dẫn được liên kết với chú thích để khi chú thích di chuyển, điểm cuối của đường dẫn cũng di chuyển theo. Khi di chuyển chuỗi ký tự và khung điều chỉnh đối tượng chú thích, đoạn đường dẫn cuối cùng hoặc sẽ gắn vào phía bên trái hoặc bên phải của chú thích tùy theo vị trí của chú thích đó so với điểm áp chót của đường dẫn. Nếu điểm giữa của chú thích ở bên trái điểm áp chót của đường dẫn, đường dẫn sẽ gắn vào bên phải, và với tất cả các trường hợp khác, đường dẫn sẽ được gắn vào bên trái chú thích.

Bỏ đi một trong hai đối tượng (chú thích và đường dẫn) khỏi bản vẽ bằng một trong các phương thức Erase, Add (để thêm vào một block) hoặc Wblock sẽ làm phá vỡ liên kết. Nếu đường dẫn và chú thích của nó được sao chép cùng nhau, bản sao chép mới vẫn có liên kết. Nếu được sao chép riêng lẻ thì sẽ không còn liên kết. Nếu sự liên kết bị phá vỡ vì một lý do nào đó, chẳng hạn như do chỉ sao chép đối tượng đường dẫn hoặc do xóa chú thích, đường dẫn sẽ bị xóa đi.

### Liên kết đường dẫn với chú thích

Ví dụ sau đây tạo đối tượng Mtext. Sau đó tạo một đường dẫn và tạo liên kết với chuỗi ký tự vừa tạo để làm chú thích.

```
Sub Ch5_AddAnnotation()  
    Dim leaderObj As AcadLeader  
    Dim mtextObj As AcadMText  
    Dim points(0 To 8) As Double  
    Dim insertionPoint(0 To 2) As Double
```



```

Dim width As Double
Dim leaderType As Integer
Dim annotationObject As Object
Dim textString As String, msg As String

' Tạo đối tượng Mtext trong không gian mô hình
textString = "Hello, World."
insertionPoint(0) = 5
insertionPoint(1) = 5
insertionPoint(2) = 0
width = 2
Set mtextObj = ThisDrawing.ModelSpace. _
    AddMText(insertionPoint, width, textString)

' Thông số của đường dẫn
points(0) = 0: points(1) = 0: points(2) = 0
points(3) = 4: points(4) = 4: points(5) = 0
points(6) = 4: points(7) = 5: points(8) = 0
leaderType = acLineWithArrow
' Tạo đối tượng đường dẫn trong không gian mô hình và
' liên kết với đối tượng Mtext làm chú thích
Set annotationObject = mtextObj
Set leaderObj = ThisDrawing.ModelSpace. _
    AddLeader(points, annotationObject, leaderType)
ZoomAll
End Sub

```

## 6.4. Hiệu chỉnh liên kết của đường dẫn

Trừ liên kết giữa đường dẫn và chú thích, đường dẫn và chú thích của nó là hai đối tượng hoàn toàn tách biệt trong bản vẽ. Chỉnh sửa đường dẫn sẽ không làm ảnh hưởng đến chú thích và ngược lại.

Dù được tạo ra dựa trên các biến hệ thống DIMCLRT, DIMTXT, và DIMTXSTY (để xác định màu, chiều cao và kiểu) nhưng chuỗi ký tự chú thích không bị thay đổi bởi các biến số này bởi vì thực chất nó không phải là đối tượng kích thước.

Sử dụng phương thức Evaluate để định mối quan hệ giữa đường dẫn và chú thích liên kết với nó. Lệnh này sẽ cập nhật lại đường dẫn khi cần.

## 6.5. Hiệu chỉnh đường dẫn

Bất cứ điều chỉnh nào làm thay đổi vị trí chú thích của đường dẫn đều ảnh hưởng đến vị trí điểm cuối của đường dẫn có liên kết. Tương tự, khi xoay chú thích sẽ khiến đường móc của đường dẫn (nếu có) xoay theo.

Để thay đổi kích cỡ của đường dẫn, ta có thể đặt lại tỷ lệ cho nó. Việc đặt tỷ lệ chỉ thay đổi tỷ lệ của đối tượng được chọn. Chẳng hạn, nếu bạn đặt tỷ lệ đường dẫn, vị trí chú thích sẽ thay đổi theo điểm cuối của đường dẫn nhưng không bị biến đổi tỷ lệ.

Ngoài việc thay đổi tỷ lệ, ta cũng có thể di chuyển, lấy đối xứng và xoay đường dẫn. Ta cũng có thể thay đổi kiểu chữ liên kết với chú thích bằng cách sử dụng thuộc tính StyleName.

## 7. Tạo dung sai hình học

Dung sai hình học thể hiện độ lệch của hình dạng, mặt cắt, hướng, vị trí của một chi tiết. Ta thêm dung sai hình học vào khung điều chỉnh đối tượng. Khung này chứa tất cả các thông tin về dung sai cho một đường kích thước.

Để tạo dung sai hình học, ta sử dụng phương thức AddTolerance. Phương thức này cần phải nhập ba giá trị: chuỗi ký tự gồm các biểu tượng dung sai, vị trí đặt dung sai trên bản vẽ và một vectơ chỉ phương xác định chiều của dung sai.

Đối tượng dung sai cũng có thể được sao chép, di chuyển, xoá, thay đổi tỷ lệ và xoay.

Khung điều chỉnh đối tượng bao gồm ít nhất hai phần. Phần đầu chứa các ký tự thể hiện loại dung sai được áp dụng, chẳng hạn như hình dạng, hướng hoặc biên. Sai số hình dạng điều chỉnh độ thẳng, độ phẳng, độ tròn, dạng trụ và mặt cắt của đường và mặt phẳng.

Ký tự tham chiếu chuẩn  
thứ nhất, hai và ba

Giá trị dung sai

Ký hiệu đặc tính hình  
học – trong trường hợp  
này là độ thẳng

Ký hiệu đường kính (lựa  
chọn)

Điều kiện vật liệu chuẩn

⊕	∅ 0.127	M	A	M	B	S	C L

Thành phần thứ hai chứa các giá trị dung sai. Khi được áp dụng, phía trước giá trị dung sai là biểu tượng đường kính và theo sau là biểu tượng điều kiện vật liệu.

### Tạo dung sai hình học

Ví dụ sau đây tạo một dung sai hình học đơn giản trong không gian mô hình.

```
Sub Ch5_CreateTolerance()
    Dim toleranceObj As AcadTolerance
    Dim textString As String
    Dim insertionPoint(0 To 2) As Double
    Dim direction(0 To 2) As Double

    ' Định nghĩa đối tượng dung sai
    textString = "Here is the Feature Control Frame"
    insertionPoint(0) = 5
    insertionPoint(1) = 5
    insertionPoint(2) = 0
    direction(0) = 1
    direction(1) = 1
    direction(2) = 0

    ' Tạo đối tượng dung sai trong không gian mô hình
    Set toleranceObj = ThisDrawing.ModelSpace._
        AddTolerance(textString, insertionPoint, direction)
    ZoomAll
End Sub
```

## 7.1. Hiệu chỉnh dung sai

Dung sai chịu ảnh hưởng bởi một vài biến hệ thống: DIMCLRD điều khiển màu của khung điều chỉnh đối tượng; DIMCLRT điều khiển màu của chuỗi ký tự dung sai, DIMGAP điều khiển khoảng trống giữa khung điều chỉnh đối tượng và chuỗi ký tự; DIMTXT điều chỉnh cỡ của ký tự dung sai. Sử dụng phương thức SetVariable để gán giá trị của biến hệ thống.

Tài liệu tham khảo  
BM Tự động hoá TKCĐ

Tài liệu tham khảo  
BM Tự động hoá TKCĐ

# TÙY BIẾN THANH CÔNG CỤ VÀ TRÌNH ĐƠN

AutoCAD ActiveX Automation cung cấp quyền kiểm soát trong việc điều chỉnh các trình đơn và các thanh công cụ trong phiên làm việc hiện hành của AutoCAD.

Sử dụng AutoCAD ActiveX/VBA, ta có thể hiệu chỉnh hoặc thêm các chi tiết hoặc thậm chí thay đổi hoàn toàn cấu trúc trình đơn hiện tại. Ta cũng có thể điều khiển các thanh công cụ và các trình đơn của nút chuột phải.

Việc tùy biến các trình đơn giúp nâng cao hiệu quả công việc bằng cách hiện rõ các nhiệm vụ cụ thể của ứng dụng hoặc bằng cách rút gọn những thao tác với nhiều bước khác nhau vào trong một mục trình đơn.

Trong chương này

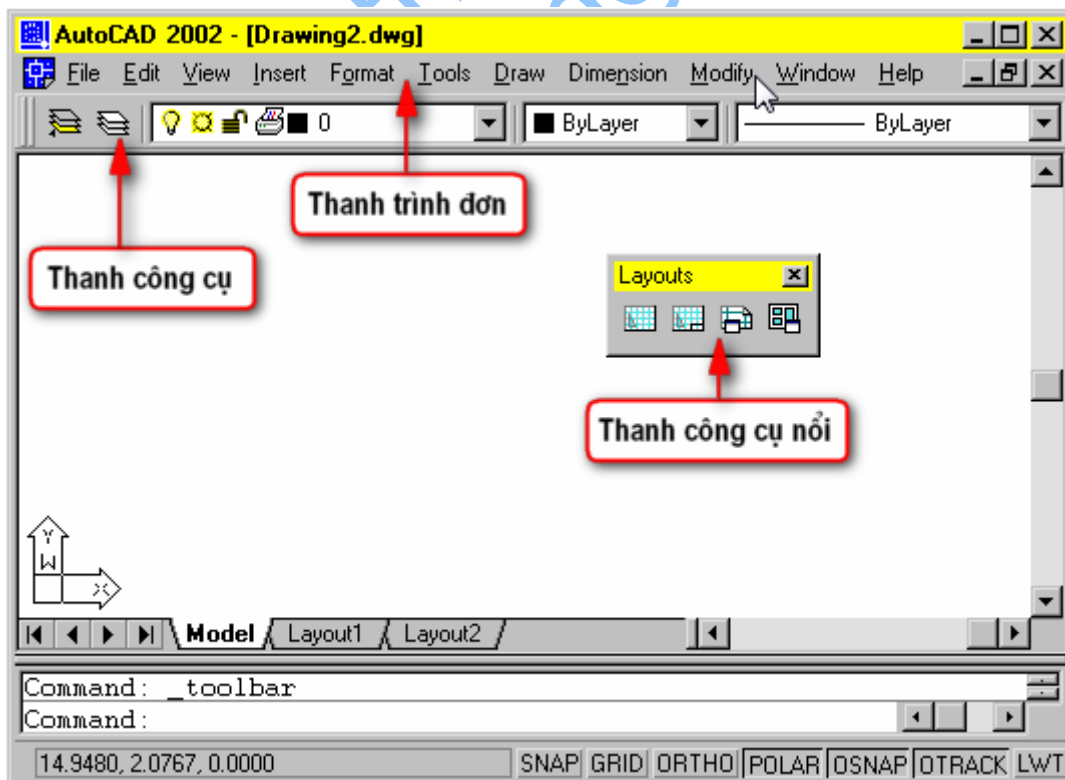
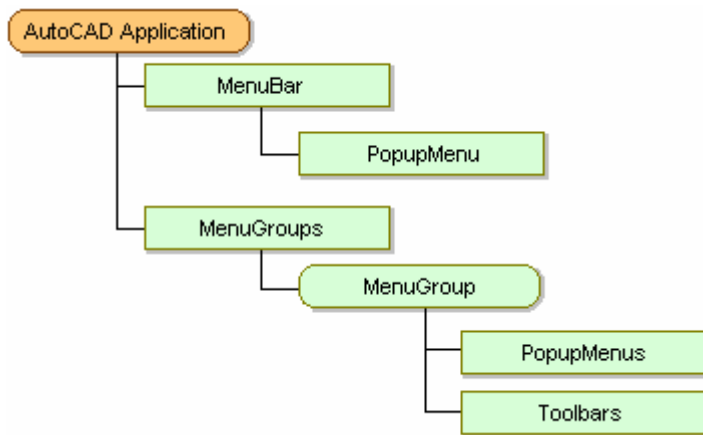
6

- Tìm hiểu tập đối tượng **MenuBar** và **MenuGroups**
- Tải các nhóm trình đơn
- Thay đổi thanh trình đơn
- Tạo và hiệu chỉnh trình đơn kéo xuống và trình đơn tắt
- Tạo và hiệu chỉnh thanh công cụ
- Tạo Macro
- Tạo dòng trạng thái trợ giúp cho các mục trong trình đơn và nút trên thanh công cụ
- Thêm mục vào trình đơn tắt

# 1. Tìm hiểu tập đối tượng MenuBar và MenuGroups

AutoCAD ActiveX cung cấp một số đối tượng dạng trình đơn. Trong đó, quan trọng nhất là tập đối tượng MenuBar (thanh trình đơn) và MenuGroups (các nhóm trình đơn). Tập đối tượng MenuBar bao gồm tất cả các trình đơn hiển thị trên thanh trình đơn của AutoCAD.

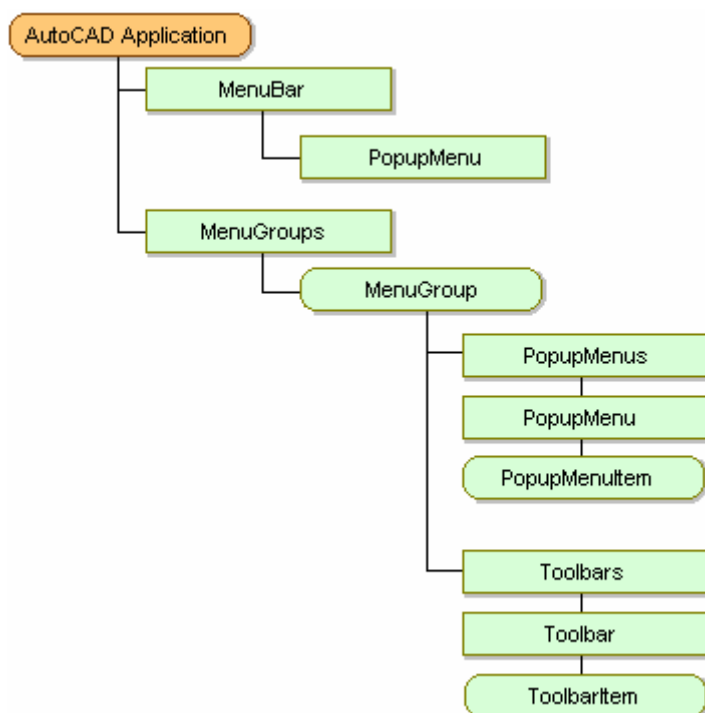
MenuGroups bao gồm các nhóm trình đơn được tải trong phiên làm việc hiện hành của AutoCAD. Các nhóm trình đơn này bao hàm tất cả các trình đơn hiện có trong phiên làm việc của AutoCAD, và một vài trong số đó hoặc tất cả có thể được hiển thị ở thanh trình đơn của AutoCAD. Bên cạnh các trình đơn, các nhóm trình đơn cũng bao hàm tất cả các thanh công cụ có trong phiên làm việc hiện hành của AutoCAD.



## 1.1. Khám phá tập đối tượng MenuGroups

Tập đối tượng MenuGroups bao gồm các nhóm trình đơn được tải lên trong phiên làm việc hiện hành của AutoCAD. Mỗi nhóm trình đơn bao gồm một tập đối tượng PopupMenus và một tập đối tượng Toolbars. Tập đối tượng PopupMenus gồm tất cả những trình đơn bên trong nhóm trình đơn đó. Tương tự, tập đối tượng Toolbars gồm tất cả các thanh công cụ bên trong nhóm trình đơn.

Thực tế, mỗi PopupMenu là một tập đối tượng bao hàm một đối tượng cá biệt dành cho một mục trong trình đơn. Và cũng tương tự, mỗi thanh công cụ là một tập đối tượng chứa các đối tượng tương ứng với một mục trên thanh công cụ.



## 2. Tải các nhóm trình đơn

Các nhóm trình đơn được tải vào trong AutoCAD bằng phương thức Load. Khi sử dụng phương thức Load, gán tham số BaseMenu bằng True để tải một nhóm trình đơn mới vào thanh trình đơn. Thao tác này sẽ tải một nhóm trình đơn như là trình đơn cơ sở giống như khi sử dụng lệnh MENU trong AutoCAD.

Để tải một nhóm trình đơn mới làm trình đơn cục bộ<sup>1</sup>, ta bỏ đi tham số BaseMenu. Thao tác này sẽ tải một nhóm trình đơn giống lệnh MENULOAD trong AutoCAD. Một khi đã được tải vào tập đối tượng MenuGroups, ta có thể chèn các trình đơn cục bộ vào thanh trình đơn bằng cách sử dụng phương thức InsertMenuInMenuBar hoặc InsertInMenuBar.

---

<sup>1</sup> **Trình đơn cục bộ (Partial menu):** là loại trình đơn được tải sau khi trình đơn chính của AutoCAD được nạp vào. Trình đơn cục bộ có thể được tải và gỡ bỏ bất cứ khi nào trong một phiên làm việc của AutoCAD.



Khi đã tải xong một nhóm trình đơn, tất cả các trình đơn và thanh công cụ được định nghĩa trong nhóm trình đơn đó đều có thể đem ra sử dụng. Ta có thể:

- Thêm trình đơn mới lên thanh trình đơn
- Gỡ bỏ những trình đơn đã có ra khỏi thanh trình đơn
- Xấp xếp lại các trình đơn trên thanh trình đơn
- Thêm mục mới vào một trình đơn hoặc thanh công cụ sẵn có
- Gỡ bỏ một mục đã có ra khỏi một trình đơn hoặc thanh công cụ
- Tạo mới trình đơn và thanh công cụ
- Tạo thanh công cụ dạng nổi hoặc dạng cố định<sup>1</sup>
- Kích hoạt hoặc vô hiệu hóa một mục trong trình đơn và thanh công cụ.
- Chọn hoặc bỏ chọn một mục trong trình đơn
- Thay đổi chuỗi chứa thẻ, nhãn và trợ giúp của mục trên trình đơn hoặc thanh công cụ
- Gán lại Macro ứng với một mục trong trình đơn hay thanh công cụ.

### Tải một nhóm trình đơn

Sau đây là một ví dụ tải một tệp trình đơn có tên *acad.mnc*

```
ThisDrawing.Application.MenuGroups.Load "acad.mnc"
```

---

**CHÚ Ý** Không thể hiệu chỉnh hình ảnh của các mục trong trình đơn bằng ActiveX Automation. Tuy nhiên, ta vẫn có thể tải hoặc gỡ bỏ các trình đơn này bằng ActiveX Automation. Để biết thêm chi tiết về các dạng trình đơn, xem chương 4, "Tùy biến Menu" trong tài liệu "AutoCAD Customization Guide".

---

## 2.1. Tạo nhóm trình đơn mới

Chương trình AutoCAD ActiveX không cho phép lập trình tạo một nhóm trình đơn mới. Tuy vậy, ta có thể tải một nhóm trình đơn có sẵn rồi lưu nó ra ngoài với một cái tên mới vào một tệp trình đơn mới. Sau đó, ta có thể hiệu chỉnh nhóm trình đơn để có thể bao hàm những chức năng mà mình mong muốn. Quá trình tạo những nhóm trình đơn mới dựa trên những trình đơn có sẵn có ưu điểm là tự động cung cấp những trình đơn cơ bản như File, Window và Help.

### Lưu một nhóm trình đơn vào một tệp mới

Dưới đây là ví dụ minh họa việc lưu nhóm trình đơn đầu tiên trong tập đối tượng nhóm trình đơn vào một tệp có tên *MyMenu.mnc*

```
ThisDrawing.Application.MenuGroups.Item(0). _  
SaveAs "MyMenu.mnc", acMenuFileCompiled
```

---

<sup>1</sup> Khi di chuyển thanh công cụ đến mép của một cửa sổ nào đó, thanh công cụ sẽ gắn theo đường mép và trở thành một phần của cửa sổ đó.

### 3. Thay đổi thanh trình đơn

Như chúng ta đã thấy, thanh trình đơn có thể được thay đổi hoàn toàn bằng một nhóm trình đơn mới nếu nhóm trình đơn đó được tải dưới dạng một trình đơn cơ sở. Bên cạnh đó, ta cũng có thể thêm, bớt hoặc sắp xếp lại các trình đơn riêng lẻ trên thanh trình đơn.

#### 3.1. Chèn một mục vào thanh trình đơn

Để chèn một trình đơn có sẵn vào thanh trình đơn, ta sử dụng phương thức `InsertMenuInMenuBar` hoặc `InsertInMenuBar`. Cả hai phương thức này đều có cùng một tác dụng là chèn một trình đơn có sẵn lên thanh trình đơn.

Sự khác nhau giữa hai phương thức trên là ở đối tượng thực hiện gọi phương thức đó. Phương thức `InsertMenuInMenuBar` được gọi từ tập đối tượng `PopupMenu`. Sử dụng phương thức này, ta có thể chèn bất cứ trình đơn nào từ tập đối tượng vào một vị trí cụ thể trên thanh trình đơn. Phương thức này yêu cầu các thông số đầu vào là tên của trình đơn được chèn vào và vị trí trên thanh trình đơn để chèn vào.

Phương thức `InsertMenuBar` thì được gọi trực tiếp từ đối tượng `PopupMenu`, là đối tượng sẽ được chèn vào thanh trình đơn. Phương thức này chỉ yêu cầu một tham số đầu vào là vị trí trên thanh trình đơn. Ta không cần phải nhập tên của trình đơn vì ta đang gọi phương thức một cách trực tiếp từ trình đơn sẽ được chèn.

Ta nên sử dụng phương thức nào thuận tiện hơn trong ứng dụng của mình.

#### Chèn một trình đơn lên thanh trình đơn

Ví dụ sau tạo một trình đơn mới có tên là `TestMenu` và chèn một mục vào nó. Mục này sẽ được gán lệnh `OPEN`. Sau khi hoàn thành, trình đơn sẽ hiện lên trên thanh trình đơn.

```
Sub Ch6_InsertMenu()  
    ' Định nghĩa biến cho nhóm trình đơn hiện tại  
    Dim currMenuGroup As AcadMenuGroup  
    Set currMenuGroup = ThisDrawing.Application. _  
        MenuGroups.Item(0)  
    ' Tạo một trình đơn mới  
    Dim newMenu As AcadPopupMenu  
    Set newMenu = currMenuGroup.Menus.Add("TestMenu")  
    ' Khai báo biến cho mục trình đơn  
    Dim newItem As AcadPopupMenuItem  
    Dim openMacro As String  
  
    ' Gán lệnh "ESC ESC _open " và tạo mục trình đơn  
    openMacro = Chr(3) + Chr(3) + Chr(95) + "open" + Chr(32)  
    Set newItem = newMenu.AddMenuItem(newMenu.Count + 1, _  
        "Open", openMacro)  
  
    ' Hiển thị trình đơn trên thanh trình đơn  
    currMenuGroup.Menus.InsertMenuInMenuBar "TestMenu", ""  
End Sub
```

### 3.2. Gỡ bỏ một mục ra khỏi thanh trình đơn

Để gỡ bỏ một trình đơn ra khỏi thanh trình đơn, ta sử dụng phương thức `RemoveMenuFromMenuBar` hoặc `RemoveFromMenuBar`. Cả hai phương thức trên đều có cùng một mục đích là gỡ đi một mục từ thanh trình đơn.

Hai phương thức trên khác nhau ở đối tượng mà phương thức đó được gọi. Phương thức `RemoveMenuFromMenuBar` được gọi từ tập đối tượng `PopupMenu`. Phương thức này yêu cầu tham số đầu vào là tên của mục trình đơn và vị trí tương ứng trên thanh trình đơn.

Phương thức `RemoveFromMenuBar` được gọi trực tiếp từ đối tượng `PopupMenu` mà ta cần gỡ bỏ. Sử dụng phương thức này ta không cần nhập bất cứ thông tin nào. Tên của trình đơn là không cần thiết vì ta đang gọi phương thức trực tiếp từ đối tượng mà ta cần gỡ bỏ.

Ta nên sử dụng phương thức nào thuận tiện hơn cho ứng dụng của mình.

#### Gỡ bỏ một mục ra khỏi thanh trình đơn

Sau đây là ví dụ gỡ bỏ mục trình đơn đã chèn trong ví dụ trước.

```
` Gỡ bỏ trình đơn từ thanh trình đơn
currMenuGroup.Menus.RemoveMenuFromMenuBar ("TestMenu")
```

---

**CHÚ Ý** Mục trình đơn đã gỡ bỏ khỏi thanh trình đơn vẫn còn có hiệu lực trong nhóm trình đơn. Việc gỡ bỏ chỉ đơn giản là làm cho người sử dụng không thể nhìn thấy được mục trình đơn đó.

---

### 3.3. Sắp xếp lại các mục đơn trên thanh trình đơn

Để sắp xếp lại các mục trình đơn, ta tiến hành chèn và gỡ bỏ các mục trình đơn cho đến khi có được trật tự mà mình mong muốn.

#### Chuyển trình đơn đầu xuống cuối thanh trình đơn

Ví dụ sau sẽ xóa mục đầu tiên trong thanh trình đơn, sau đó chèn nó vào vị trí cuối cùng trên thanh trình đơn.

```
Sub Ch6_MoveMenu ()
    ` Định nghĩa biến để lưu mục trình đơn cần di chuyển
    Dim moveMenu As AcadPopupMenu
    Dim MyMenuBar As AcadMenuBar
    Set MyMenuBar = ThisDrawing.Application.menuBar

    ` Đặt biến moveMenu bằng mục trình đơn đầu tiên được hiển thị
    trong trình đơn trên thanh trình đơn
    Set moveMenu = MyMenuBar.Item(0)

    ` Dỡ bỏ mục đầu tiên trong thanh trình đơn
    MyMenuBar.Item(0).RemoveFromMenuBar

    ` Thêm trình đơn trở lại thanh trình đơn vào vị trí lúc trước
    moveMenu.InsertInMenuBar (MyMenuBar.count)
End Sub
```

## 4. Tạo và hiệu chỉnh trình đơn kéo xuống và trình đơn tắt

AutoCAD ActiveX/VBA có khả năng hiệu chỉnh hai loại trình đơn AutoCAD: trình đơn kéo xuống<sup>1</sup> và trình đơn tắt (shortcut menu). Cả hai loại trình đơn này đều được hiển thị dưới dạng xếp lớp. Trình đơn tắt cho phép truy cập nhanh chóng đến các mục trình đơn được sử dụng nhiều nhất, chẳng hạn như chế độ bắt đối tượng.

Một trình đơn kéo xuống có thể chứa đến 999 mục, còn trình đơn tắt có thể chứa tối đa 499 mục. Các giới hạn trên đều tính bao gồm cả các trình đơn có trong cấu trúc nhiều tầng. Nếu số mục trong một trình đơn vượt quá các giới hạn này, AutoCAD sẽ bỏ qua các mục đã vượt quá. Nếu một trình đơn kéo xuống hay trình đơn tắt có chiều cao lớn hơn khoảng không cho phép trên màn hình thì nó sẽ được rút gọn cho vừa vặn.

Các trình đơn kéo xuống luôn được kéo xuống từ thanh trình đơn, còn các trình đơn tắt thì luôn hiện tại vị trí hoặc gần vị trí con trỏ chuột trên màn hình đồ họa. Cách sử dụng hai loại trình đơn này giống nhau ngoại trừ một điểm là tên của trình đơn tắt không nằm trên thanh trình đơn và thực tế là tên của nó chẳng được hiển thị ở đâu cả. Muốn truy cập vào trình đơn tắt, ta phải thông qua một trình đơn trong nhóm trình đơn cơ sở. Ta xác lập trình đơn tắt bằng cách sử dụng thuộc tính ShortcutMenu. Nếu thuộc tính ShortcutMenu trả về giá trị TRUE, thì trình đơn đó sẽ trở thành trình đơn tắt.

### 4.1. Tạo trình đơn mới

Để tạo một trình đơn mới, sử dụng phương thức Add để thêm một đối tượng PopupMenu mới vào tập đối tượng PopupMenu.

Để tạo một trình đơn tắt mới, ta phải xóa trình đơn tắt hiện có. Chỉ có thể có một trình đơn tắt duy nhất cho mỗi nhóm trình đơn. Nếu không có trình đơn tắt nào khác trong một nhóm trình đơn, ta có thể thêm một trình đơn với tên "POP0". Thao tác này sẽ thông báo cho AutoCAD biết là mình muốn tạo một trình đơn tắt.

Khi dùng phương thức Add, ta cần phải nhập tên (nhãn) của trình đơn cần thêm. Tên này sẽ trở thành tiêu đề cho trình đơn khi được tải lên thanh trình đơn. Tên trình đơn cũng là phương tiện để xác định trình đơn trong nhóm một cách dễ dàng nhất.

Tên trình đơn có thể là một chuỗi ký tự đơn hoặc có thể chứa những mã đặc biệt. Để biết danh sách đầy đủ của các mã đặc biệt, xem "*Tóm tắt về cú pháp tên trình đơn kéo xuống và trình đơn tắt*" ở trang 99 trong tài liệu "*AutoCAD Customization Guide*".

---

<sup>1</sup> **Trình đơn kéo xuống (Pull-down menu):** là loại trình đơn được kéo xuống từ thanh trình đơn và sẽ được giữ nguyên chừng nào người dùng còn giữ chuột trên trình đơn đó. Trình đơn này khác với loại trình đơn thả xuống (drop-down menu) là loại trình đơn được thả xuống từ thanh trình đơn khi có yêu cầu và sẽ giữ nguyên cho đến khi người dùng đóng trình đơn hoặc lựa chọn mục trình đơn khác.

Ta có thể thay đổi tên của trình đơn khi đã được tạo ra. Để thay đổi tên của một trình đơn đã tồn tại, ta sử dụng thuộc tính Name của trình đơn đó.

## Tạo PopupMenu

Ví dụ sau tạo một popup menu mới có tên là “TestMenu” trong nhóm trình đơn đầu tiên của tập đối tượng nhóm trình đơn.

```
Sub Ch6_CreateMenu()  
    Dim currMenuGroup As AcadMenuGroup  
    Set currMenuGroup =  
    ThisDrawing.Application.MenuGroups.Item(0)  
    ' Tạo trình đơn mới  
    Dim newMenu As AcadPopupMenu  
    Set newMenu = currMenuGroup.Menus.Add("TestMenu")  
End Sub
```

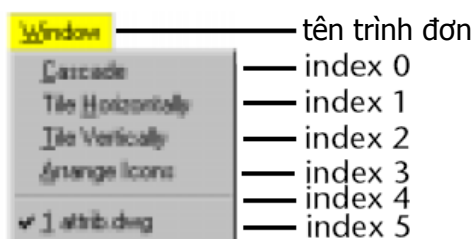
## 4.2. Thêm mục mới vào một trình đơn

Để thêm một mục mới vào một trình đơn, ta sử dụng phương thức AddMenuItem. Phương thức này tạo một mục PopupMenu và thêm nó vào trình đơn đã chỉ định.

Phương thức AddMenuItem cần nhập bốn tham số sau: Index(), Label(), Tag() và Macro()

### 4.2.1. Chỉ số (Index)

Tham số Index là một số nguyên xác định vị trí của mục mới bên trong một trình đơn. Chỉ số này bắt đầu với vị trí không (0), là vị trí đầu tiên trên trình đơn sau tiêu đề. Để thêm một mục vào cuối trình đơn, ta gán tham số Index bằng giá trị của thuộc tính Count của trình đơn đó. (Thuộc tính Count của trình đơn chỉ tổng số mục có trên trình đơn đó.)



Trên hình này, ta có thể thấy chỉ số của vị trí đầu tiên là 0, và các vạch ngăn cách cũng được tính như một mục của trình đơn với chỉ số của riêng mình (index 4). Đặc tính Count của trình đơn trên là 6. Để thêm một mục vào giữa hai mục Tile Horizontally và Tile Vertically, ta đặt Tham biến Index là 2, nghĩa là chỉ số của mục Tile Vertically. Thao tác này sẽ chèn mục mới vào vị trí chỉ số 2 và đẩy toàn bộ các mục còn lại xuống một chỉ số.

Khi đã tạo xong một mục mới, ta không thể thay đổi chỉ số của mục đó bằng thuộc tính Index. Để làm việc này, ta phải xóa đi rồi thêm trở lại mục trình đơn theo vị trí mong muốn hoặc xóa các trình đơn xung quanh cho đến khi mục trình đơn đó có vị trí như mong muốn.

### 4.2.2. Nhãn (Label)

Nhãn là một chuỗi ký tự xác định nội dung và định dạng của các mục trình đơn. Nhãn của mục trình đơn có thể chứa các chuỗi DIESEL mà tùy trường hợp có thể được thay thế mỗi khi được hiển thị. Để biết thêm chi tiết về cách sử dụng các chuỗi DIESEL, xem “Chuỗi DIESEL trong trình đơn” ở trang 12 trong tài liệu “*AutoCAD Customization Guide*”.

Bên cạnh chuỗi DIESEL, nhãn có thể chứa những mã đặc biệt. Chẳng hạn, một ký tự “&” đặt ngay trước một ký tự sẽ ám chỉ rằng ký tự đó là phím tắt. Để biết danh sách đầy đủ các mã đặc biệt, xem “Tóm tắt về cú pháp tên trình đơn kéo xuống và trình đơn tắt” ở trang 99 trong tài liệu “*AutoCAD Customization Guide*”.

Những chữ mà người dùng nhìn thấy trên mục trình đơn gọi là tiêu đề, được rút ra từ nhãn thông qua việc đọc hiểu tất cả các chuỗi DIESEL và mã đặc biệt chứa trong nhãn. Chẳng hạn, nhãn “&Edit” sẽ hiển thị tiêu đề là “Edit”.

Sau khi đã tạo được một mục trình đơn, ta có thể thay đổi nhãn của mục này thông qua thuộc tính Label của mục trình đơn đó.

### 4.2.3. Thẻ (Tag)

Thẻ là một chuỗi gồm các chữ, số và dấu gạch dưới (\_). Chuỗi này là giá trị duy nhất dùng để xác định mục trên một trình đơn.

Sau khi đã tạo được mục trình đơn, ta có thể sử dụng thuộc tính Tag để thay đổi thẻ của mục trình đơn đó.

### 4.2.4. Macro

Macro là một loạt các lệnh nhằm thực hiện một thao tác cụ thể nào đó khi người dùng chọn một mục trình đơn. Macro của trình đơn có thể đơn giản là một số thao tác bàn phím được lưu lại để thực hiện một nhiệm vụ hoặc chúng có thể là một sự kết hợp phức tạp các lệnh, AutoLISP, DIESEL hoặc mã chương trình ActiveX. Để biết thêm chi tiết về Macro của trình đơn, xem “*Macro của trình đơn*” ở trang 82 trong tài liệu “*AutoCAD Customization Guide*”.

Sau khi đã tạo được một mục trình đơn, ta có thể thay đổi macro thông qua đặc tính Macro của mục trình đơn đó.

### Thêm mục vào Popup Menu

Dưới đây là ví dụ tạo một trình đơn có tên “TestMenu” và chèn một mục vào đó. Mục trình đơn này có tên “Open”, và Macro của mục sẽ thực hiện lệnh OPEN.

```
Sub Ch6_AddAMenuItem()  
    Dim currMenuGroup As AcadMenuGroup  
    Set currMenuGroup =  
        ThisDrawing.Application.MenuGroups.Item(0)  
    ' Tạo trình đơn mới  
    Dim newMenu As AcadPopupMenu  
    Set newMenu = currMenuGroup.Menus.Add("TestMenu")  
    ' Thêm một mục vào trình đơn  
    Dim newItem As AcadPopupMenuItem  
    Dim openMacro As String  
    ' Gán lệnh (macro) "ESC ESC _open "  
End Sub
```

```

openMacro = Chr(3) + Chr(3) + Chr(95) + "open" + Chr(32)
Set newMenuItem = newMenu.AddItem _
    (newMenu.count + 1, "Open", openMacro)
' Hiển thị trên thanh trình đơn
newMenu.InsertInMenuBar _
    (ThisDrawing.Application.menuBar.count + 1)
End Sub

```

### 4.3. Thêm vạch ngăn vào một trình đơn

Để thêm vạch ngăn vào một trình đơn, ta sử dụng phương thức `AddSeparator`. Phương thức này tạo một đối tượng `PopupMenuItem` và thêm nó vào trình đơn đã chỉ định. Loại đối tượng `PopupMenuItem` này được gán kiểu là `acSeparator`. Thuộc tính `Type` quy định kiểu của một mục trình đơn.

Phương thức `AddMenuItem` chỉ cần nhập tham số `Index`. Tham số `Index` là một số nguyên xác định vị trí của vạch phân cách trong trình đơn. Chỉ số đầu tiên là 0, là vị trí đầu tiên của trình đơn sau phần tiêu đề.

Xem “*Kích hoạt hoặc vô hiệu hoá mục trình đơn*”, trang 188 để xem thêm ví dụ thêm một vạch phân cách vào trình đơn.

### 4.4. Gán phím tắt cho một mục trình đơn

Để gán phím tắt cho một mục trình đơn bằng AutoCAD ActiveX/VBA, ta sử dụng thuộc tính `Label` của mục trình đơn đó. Để xác định một phím tắt, chèn ký tự ASCII tương ứng với ký hiệu “&” ngay trước một ký tự được sử dụng làm phím tắt. Chẳng hạn, nhãn `Chr(Asc("&")) + "Edit"` sẽ được hiển thị là Edit, với ký tự E là phím tắt.

Có nhiều phương pháp khác nhau để tạo phím tắt cho các trình đơn và các lệnh của AutoCAD mà hiện không có ở AutoCAD ActiveX/VBA. Để biết thêm chi tiết, xem thêm mục “*Phím tắt*” trong tài liệu “*AutoCAD Customization Guide*”.

#### Thêm các phím tắt vào trình đơn

Sau đây là ví dụ lặp lại ví dụ “Thêm mục vào Popup Menu”, ta thêm phím tắt cho cả hai trình đơn “TestMenu” và “Open”. Chữ cái “s” sẽ được dùng làm phím tắt cho “TestMenu” còn “O” là phím tắt của “Open”.

```

Sub Ch6_AddMenuItem()
    Dim currMenuGroup As AcadMenuGroup
    Set currMenuGroup =
        ThisDrawing.Application.MenuGroups.Item(0)
    ' Tạo một trình đơn mới
    Dim newMenu As AcadPopupMenu
    Set newMenu = currMenuGroup.Menus.Add _
        ("Te" + Chr(Asc("&")) + "stMenu")
    ' Thêm mục vào trình đơn
    Dim newMenuItem As AcadPopupMenuItem
    Dim openMacro As String

    ' Gán lệnh "ESC ESC _open "
    openMacro = Chr(3) + Chr(3) + Chr(95) + "open" + Chr(32)
    Set newMenuItem = newMenu.AddItem _
        (newMenu.count + 1, Chr(Asc("&")) + _
            "Open", openMacro)
    ' Hiển thị trên thanh trình đơn

```



```

newMenu.InsertInMenuBar _
    (ThisDrawing.Application.menuBar.count + 1)
End Sub

```

## 4.5. Tạo trình đơn con nhiều tầng

Để thêm một trình đơn con nhiều tầng, ta tạo ra một trình đơn con bằng phương thức AddSubMenu. Phương thức này tạo ra một đối tượng PopupMenu mới và thêm nó vào trình đơn đã định. Dạng đối tượng PopupMenu đặc biệt này được gán kiểu là acSubMenu.

Phương thức AddSubMenu cần nhập ba tham số: Index, Label và Tag.

Tham số Index (chỉ số) là một số nguyên xác định vị trí của mục trình đơn mới trên thanh trình đơn. Chỉ số bắt đầu là 0, là vị trí đầu tiên trên thanh trình đơn sau phần tiêu đề. Để thêm mục mới vào cuối một trình đơn, đặt tham biến Index bằng với giá trị của thuộc tính Count của trình đơn. (Đặc tính Count của trình đơn chỉ tổng số mục trên trình đơn đó)

Tham số Label (nhãn) là một chuỗi quy định nội dung và định dạng của mục trình đơn. Dòng văn bản mà người dùng nhìn thấy trên mục trình đơn gọi là đầu đề và nó được kế thừa từ nhãn bằng cách xử lý tất cả các chuỗi DIESEL và các mã tự đặc biệt chứa trong nhãn. Ví dụ, nhãn "&Edit" sẽ cho ra đầu đề "Edit"

Tham biến Tag (thẻ tên), là một chuỗi những chữ, số và dấu gạch dưới (\_). Chuỗi này là duy nhất cho một mục trên trình đơn đã cho.

### 4.5.1. Trả về trình đơn mới

Phương thức AddSubMenu không trả về đối tượng PopupMenuItem vừa mới tạo thành mà thay vào đó là trình đơn mới mà trình đơn con đang trỏ đến. Trình đơn mới này (được trả về là đối tượng PopupMenu) vẫn được tạo như trình đơn bình thường.

Để có thêm thông tin về cách tạo thêm trình đơn, xin xem thêm phần "Thêm mục mới vào một trình đơn" trang 185.

### Tạo mới và thêm trình đơn con

Ví dụ sau tạo một trình đơn mới có tên là "TestMenu" và thêm nó vào trình đơn con có tên là "OpenFile". Trình đơn con gắn với mục có tên là "Open", sẽ mở bản vẽ. Cuối cùng, trình đơn sẽ được hiển thị trên thanh trình đơn.

```

Sub Ch6_AddASubMenu()
    Dim currMenuGroup As AcadMenuGroup
    Set currMenuGroup =
        ThisDrawing.Application.MenuGroups.Item(0)
    ' Tạo trình đơn
    Dim newMenu As AcadPopupMenu
    Set newMenu = currMenuGroup.Menus.Add("TestMenu")
    ' Thêm trình đơn con
    Dim FileSubMenu As AcadPopupMenu
    Set FileSubMenu = newMenu.AddSubMenu("", "OpenFile")
    ' Thêm mục cho trình đơn con
    Dim newMenuItem As AcadPopupMenu
    Dim openMacro As String

```

```

' Gán lệnh "ESC ESC _open "
openMacro = Chr(3) + Chr(3) + Chr(95) + "open" + Chr(32)
Set newMenuItem = FileSubMenu.AddMenuItem _
(newMenu.count + 1, "Open", openMacro)
' Hiện thị thanh trình đơn
newMenu.InsertInMenuBar _
(ThisDrawing.Application.menuBar.count + 1)
End Sub

```

## 4.6. Xoá mục trình đơn khỏi một trình đơn

Để xóa một mục trình đơn khỏi một trình đơn, ta sử dụng phương thức Delete của mục đó.

### Xoá một mục khỏi một trình đơn

Ví dụ sau sẽ thêm một mục vào vị trí cuối cùng của trình đơn cuối của thanh trình đơn, sau đó sẽ tiến hành xóa mục này.

```

Sub Ch6_DeleteMenuItem()
Dim LastMenu As AcadPopupMenu
Set LastMenu = ThisDrawing.Application.menuBar._
Item(ThisDrawing.Application.menuBar.count - 1)
' Thêm một mục
Dim newMenuItem As AcadPopupMenuItem
Dim openMacro As String
' Gán lệnh "ESC ESC _open "
openMacro = Chr(3) + Chr(3) + Chr(95) + "open" + Chr(32)
Set newMenuItem = LastMenu.AddMenuItem _
(LastMenu.count + 1, "Open", openMacro)
' Dỡ bỏ mục khỏi trình đơn
newMenuItem.Delete
End Sub

```

## 4.7. Tìm hiểu các thuộc tính của mục trình đơn

Tất cả các mục trong trình đơn đều có những thuộc tính sau:

### 4.7.1. Tag

Tag (thẻ) là một chuỗi gồm chữ, số và dấu gạch dưới (\_). Chuỗi này là giá trị duy nhất dùng để xác định một mục trên một trình đơn. Thẻ quyết định các phím tắt tương ứng với mục trình đơn đó.

Ta có thể đọc hoặc gán giá trị của thẻ thông qua thuộc tính Tag.

### 4.7.2. Label

Label (nhãn) là một chuỗi quy định nội dung và định dạng của các mục trong trình đơn.

Nhãn của một mục có thể chứa chuỗi DIESEL, có thể làm thay đổi nội dung của nhãn khi được hiển thị. Để biết thêm chi tiết về việc sử dụng chuỗi DIESEL, xem mục “Chuỗi DIESEL trên trình đơn” trong tài liệu “*AutoCAD Customization Guide*”.

Ta có thể đọc hoặc gán giá trị của nhãn thông qua thuộc tính Label.

### 4.7.3. Caption

Caption (tiêu đề) là dòng văn bản mà người dùng nhìn thấy hiển thị trên trình đơn. Đây là thuộc tính chỉ đọc và được kế thừa từ thuộc tính Label bằng cách gạt bỏ các chuỗi DIESEL.

Ta có thể đọc giá trị của tiêu đề thông qua thuộc tính Caption.

### 4.7.4. Macro

Macro là một chuỗi các lệnh nhằm thực hiện một tác vụ cụ thể nào đó khi ta chọn một mục trong trình đơn. Macro của trình đơn có thể đơn giản là một số thao tác bàn phím được lưu lại để thực hiện một nhiệm vụ hoặc có thể là một sự kết hợp các lệnh AutoLISP, DIESEL hoặc mã chương trình ActiveX. Để biết thêm chi tiết về Macro của trình đơn, xin xem thêm mục “*Macro trình đơn*” trong tài liệu “*AutoCAD Customization Guide*”.

Ta có thể đọc hoặc gán giá trị của một Macro trình đơn thông qua thuộc tính Macro của trình đơn đó.

### 4.7.5. Help String

Help String (chuỗi trợ giúp) là một dòng chữ hiện lên trên thanh trạng thái của AutoCAD khi người dùng chỉ vào một mục trong trình đơn.

Ta có thể đọc hoặc gán giá trị của chuỗi trợ giúp bằng cách sử dụng thuộc tính HelpString.

### 4.7.6. Enable

Sử dụng thuộc tính Enable (kích hoạt), ta có thể kích hoạt hoặc vô hiệu hoá một mục trong trình đơn. Ta cũng có thể đọc thuộc tính Enable để biết một mục trong trình đơn có đang bị vô hiệu hoá hay không. Sử dụng thuộc tính này để kích hoạt hoặc vô hiệu hoá mục trong trình đơn sẽ ghi đè các cấu hình có trong chuỗi DIESEL của mục trình đơn đó.

### 4.7.7. Check

Sử dụng thuộc tính Check (chọn), ta có thể đánh dấu chọn (✓) hoặc không chọn một mục trong trình đơn. Ta cũng có thể đọc thuộc tính Check để biết một mục trong trình đơn có đang được đánh dấu chọn hay không. Sử dụng thuộc tính này sẽ ghi đè tất cả các thiết lập về đánh dấu chọn trong chuỗi DIESEL của mục trong trình đơn đó.

### 4.7.8. Index

Index (chỉ số) của một mục trong trình đơn xác định vị trí của mục đó trên trình đơn. Chỉ số vị trí của trình đơn luôn luôn bắt đầu bằng số 0. Chẳng hạn, nếu mục trình đơn là mục đầu tiên trên trình đơn, nó có chỉ số vị trí là 0. Nếu nó là mục thứ hai trên trình đơn thì nó có chỉ số vị trí là 1, v.v..

#### 4.7.9. Type

Ta có thể biết được loại (type) của mục trong trình đơn thông qua thuộc tính Type. Một mục trong trình đơn có thể thuộc một trong những loại sau đây: mục trình đơn thông thường, vạch phân cách hoặc tiêu đề của một trình đơn con. Nếu mục trình đơn là một mục thông thường, thuộc tính Type tương ứng là `acMenuItem`. Nếu mục trình đơn là một vạch phân cách, thuộc tính này là `acMenuSeparator`. Nếu mục trình đơn là tiêu đề của một trình đơn con, thuộc tính này là `acSubMenu`.

#### 4.7.10. SubMenu

Ta có thể tìm một trình đơn con (submenu) thông qua thuộc tính `SubMenu`. Nếu mục trình đơn thuộc loại `acSubMenu`, thuộc tính này trả về trình đơn gắn với trình đơn con đó. Trình đơn được trả về là đối tượng `PopupMenu`.

Nếu mục trình đơn không thuộc loại `acSubMenu`, thuộc tính trả về giá trị `NULL`.

#### 4.7.11. Parent

Ta có thể truy được trình đơn gốc của một mục trong trình đơn bằng cách sử dụng thuộc tính `Parent`. Thuộc tính này trả về trình đơn có chứa mục trình đơn đang xét. Trình đơn gốc là một đối tượng `PopupMenu`.

#### Kích hoạt hoặc vô hiệu hoá mục trình đơn

Dưới đây là ví dụ tạo một trình đơn mới có tên "TestMenu" và chèn vào đó hai mục trình đơn. Sau đó vô hiệu hoá mục thứ hai thông qua thuộc tính `Enable` và hiển thị trình đơn vừa tạo lên thanh trình đơn.

```
Sub Ch6_DisableMenuItem()  
    Dim currMenuGroup As AcadMenuGroup  
    Set currMenuGroup =  
    ThisDrawing.Application.MenuGroups.Item(0)  
    ' Tạo trình đơn mới  
    Dim newMenu As AcadPopupMenu  
    Set newMenu = currMenuGroup.Menus.Add("TestMenu")  
    ' Thêm hai mục và một vạch ngăn vào trình đơn mới  
    Dim MenuEnable As AcadPopupMenuItem  
    Dim MenuDisable As AcadPopupMenuItem  
    Dim MenuSeparator As AcadPopupMenuItem  
    Dim openMacro As String  
    ' Gán lệnh "ESC ESC _open "  
    openMacro = Chr(3) + Chr(3) + Chr(95) + "open" + Chr(32)  
    Set MenuEnable = newMenu.AddMenuItem _  
    (newMenu.count + 1, "OpenEnabled", openMacro)  
    Set MenuSeparator = newMenu.AddSeparator("")  
    Set MenuDisable = newMenu.AddMenuItem _  
    (newMenu.count + 1, "OpenDisabled", openMacro)  
    ' Vô hiệu mục trình đơn thứ hai  
    MenuDisable.Enable = False  
    ' Hiển thị trình đơn trên thanh trình đơn  
    newMenu.InsertInMenuBar _  
    (ThisDrawing.Application.menuBar.count + 1)  
End Sub
```

## 5. Tạo và hiệu chỉnh thanh công cụ

### 5.1. Tạo mới thanh công cụ

Để tạo mới một thanh công cụ, ta sử dụng phương thức Add để thêm đối tượng thanh công cụ mới vào tập đối tượng thanh công cụ (Toolbars Collection).

#### 5.1.1. Tên thanh công cụ

Phương thức Add cần tham số đầu vào là tên của thanh công cụ. Tên là một chuỗi chữ-số và không sử dụng các dấu khác, ngoài dấu gạch ngang (-) hoặc dấu gạch dưới (\_). Tên là phương tiện giúp ta xác định một thanh công cụ dễ dàng nhất trong tập đối tượng thanh công cụ.

Ta có thể đổi tên của thanh công cụ đã được tạo trước. Để đổi tên của một thanh công cụ đã có, ta sử dụng thuộc tính Name của thanh công cụ đó.

#### Tạo một thanh công cụ mới

Dưới đây là ví dụ tạo một thanh công cụ mới có tên là “TestToolbar” trong nhóm trình đơn đầu tiên của tập đối tượng nhóm trình đơn.

```
Sub Ch6_CreateToolbar()  
    Dim currMenuGroup As AcadMenuGroup  
    Set currMenuGroup =  
        ThisDrawing.Application.MenuGroups.Item(0)  
    ' Tạo thanh công cụ mới  
    Dim newToolbar As AcadToolbar  
    Set newToolbar = currMenuGroup.Toolbars.Add("TestToolbar")  
End Sub
```

### 5.2. Thêm nút vào thanh công cụ

Để thêm nút mới vào một thanh công cụ, ta sử dụng phương thức AddToolbarButton. Phương thức này tạo mới một đối tượng ToolbarItem (mục công cụ) và sau đó thêm vào thanh công cụ. Ta chỉ nên thêm nút vào thanh công cụ đang hiển thị trên màn hình.

Phương thức AddToolbarButton cần nhập 5 Tham biến: Index, Name, HelpString, Macro và FlyoutButton.

#### 5.2.1. Index

Tham số Index (chỉ số) là một số nguyên xác định vị trí của một mục trên thanh công cụ. Chỉ số bắt đầu là 0, là vị trí đầu tiên trên thanh công cụ sau tiêu đề. Để thêm một nút mới vào cuối thanh công cụ, ta đặt tham số Index bằng với giá trị của thuộc tính Count trong thanh công cụ. (Thuộc tính Count của thanh công cụ thể hiện tổng số các nút có trên thanh công cụ.)

Khi đã tạo được một nút, ta không thể thay đổi tham số Index bằng thuộc tính Index. Để thay đổi chỉ số của một nút hiện có, ta phải xoá đi rồi thêm trở lại với một giá trị Index khác, hoặc thêm hay xoá các nút xung quanh cho đến khi nút đó có vị trí như ý muốn.

### 5.2.2. Name

Name (tên) là một chuỗi dùng để xác định nút. Chuỗi này bao gồm các chữ, số và không sử dụng các dấu khác, ngoài dấu gạch ngang (-) hoặc dấu gạch dưới (\_). Chuỗi này sẽ được hiển thị khi con trỏ chuột đặt trên nút công cụ.

Sau khi đã tạo được nút, ta có thể thay đổi tên của nó thông qua thuộc tính Name.

### 5.2.3. HelpString

HelpString (chuỗi trợ giúp) là một dòng chữ hiện lên trên thanh trạng thái của AutoCAD khi người dùng chỉ vào một nút trên thanh công cụ.

Sau khi đã tạo được nút, ta có thể thay đổi chuỗi trợ giúp thông qua thuộc tính HelpString.

### 5.2.4. Macro

Macro là một chuỗi các lệnh nhằm thực hiện một tác vụ cụ thể nào đó khi ta chọn một nút trên thanh công cụ. Macro của thanh công cụ có thể đơn giản là một số thao tác bàn phím được lưu lại để thực hiện một nhiệm vụ hoặc có thể là một sự kết hợp phức tạp các lệnh AutoLISP, DIESEL hoặc mã chương trình ActiveX. Để biết thêm chi tiết về Macro, xem thêm mục “*Macro trình đơn*” trong tài liệu “*AutoCAD Customization Guide*”.

Khi đã tạo xong một nút, ta có thể thay đổi Macro của nó thông qua thuộc tính Macro.

### 5.2.5. FlyoutButton

Tham số FlyoutButton là một giá trị lựa chọn xác định xem nút mới có phải là dạng nút Flyout<sup>1</sup> hay không. Nếu nút mới là một nút Flyout thì tham số này gán là TRUE. Nếu không, tham số này có thể gán là FALSE hoặc để trống.

#### Thêm nút vào một thanh công cụ mới

Dưới đây là ví dụ tạo một thanh công cụ mới và thêm một nút vào đó. Nút này được gán một Macro thực hiện lệnh OPEN khi được chọn.

```
Sub Ch6_AddButton()  
    Dim currMenuGroup As AcadMenuGroup  
    Set currMenuGroup =  
        ThisDrawing.Application.MenuGroups.Item(0)  
    ' Tạo thanh công cụ mới  
    Dim newToolbar As AcadToolbar  
    Set newToolbar = currMenuGroup.Toolbars.Add("TestToolbar")  
    ' Thêm một nút vào thanh công cụ  
    Dim newButton As AcadToolbarItem
```

---

<sup>1</sup> **Flyout Button:** là một dạng nút công cụ đặc biệt trong AutoCAD. Sau khi tạo nút công cụ dạng Flyout, người dùng cần phải gán nút công cụ này với một thanh công cụ hiện có trong AutoCAD bằng cách thay đổi thuộc tính Flyout riêng có của nút công cụ dạng này. Sau đó, khi người dùng nhấn và giữ chuột trên nút flyout, một thanh công cụ sẽ hiện ra tương ứng với thanh công cụ mà ta đã gán trong thuộc tính Flyout của nút công cụ. Nếu ta chỉ nhấn chuột, thao tác này tương ứng với việc ta nhấn nút đầu tiên trong thanh công cụ đó.

```

Dim openMacro As String
' Gán lệnh "ESC ESC _open " cho nút
openMacro = Chr(3) + Chr(3) + Chr(95) + "open" + Chr(32)
Set newButton = newToolbar.AddToolbarButton _
    ("", "NewButton", "Open a File.", openMacro)

```

End Sub

### 5.3. Thêm vạch ngăn vào một thanh công cụ

Để thêm vạch ngăn vào một thanh công cụ, ta sử dụng phương thức AddSeparator. Phương thức này tạo mới một đối tượng PopupToolbarItem và thêm nó vào thanh công cụ. Đối tượng PopupToolbarItem này được gán thuộc tính Type là acSeparator. Ta có thể biết được loại của một nút thông qua thuộc tính Type.

Phương thức AddSeparator chỉ cần nhập vào một đối số duy nhất: Index. Đối số Index là một số nguyên xác định vị trí của vạch ngăn trên thanh công cụ. Chỉ số bắt đầu là 0, là vị trí đầu tiên trên thanh công cụ sau tiêu đề.

### 5.4. Định nghĩa ảnh cho nút

Để định nghĩa ảnh của một nút, ta sử dụng phương thức SetBitmaps và GetBitmaps. Phương thức SetBitmaps cần nhập hai đối số: SmallIconName và LargeIconName.

#### 5.4.1. SmallIconName

SmallIconName (tên biểu tượng cỡ nhỏ) xác định chuỗi ID của nguồn ảnh cỡ nhỏ (loại bitmap 16×15). Chuỗi này phải gồm chữ, số và không dùng dấu khác, ngoài dấu gạch ngang (-) và gạch dưới (\_) đồng thời nên có phần mở rộng .bmp. Nguồn có thể là một tệp bitmap hệ thống hoặc bitmap do người dùng định nghĩa. Tệp bitmap do người dùng định nghĩa đòi hỏi phải có kích cỡ và đường dẫn thích hợp.

#### 5.4.2. LargeIconName

LargeIconName (tên biểu tượng cỡ lớn) xác định chuỗi ID của nguồn ảnh cỡ lớn (loại bitmap 24×22). Chuỗi này phải gồm chữ, số và không dùng dấu khác, ngoài dấu gạch ngang (-) và gạch dưới (\_) đồng thời nên có phần mở rộng .bmp. Nguồn có thể là một tệp bitmap hệ thống hoặc bitmap do người dùng định nghĩa. Tệp bitmap do người dùng định nghĩa đòi hỏi phải có kích cỡ và đường dẫn thích hợp.

#### Truy vấn thanh công cụ hiện có để tìm tên biểu tượng của nút trên nó

```

Sub Ch6_GetButtonImages()
    Dim Button As AcadToolbarItem
    Dim Toolbar0 As AcadToolbar
    Dim MenuGroup0 As AcadMenuGroup
    Dim SmallButtonName As String
    Dim LargeButtonName As String
    Dim msg As String
    Dim ButtonType As String

    ' Lấy lại thanh công cụ đầu tiên trong nhóm trình đơn đầu tiên
    Set MenuGroup0 = ThisDrawing.Application.MenuGroups.Item(0)
    Set Toolbar0 = MenuGroup0.Toolbars.Item(0)

```



```

' Khởi tạo các biến kiểu chuỗi
SmallButtonName = ""
LargeButtonName = ""

' Tạo tiêu đề cho hộp thông báo và
' hiển thị thanh công cụ cần truy vấn
msg = "Toolbar: " + Toolbar0.Name + vbCrLf
Toolbar0.Visible = True

' Duyệt qua thanh công cụ và thu thập dữ liệu
' cho mỗi nút trên thanh công cụ. Nếu thanh công cụ
' là một nút bình thường hoặc một nút dạng Flyout,
' ta lấy lại tên biểu tượng cỡ nhỏ và lớn của nút công cụ đó.
For Each Button In Toolbar0
ButtonType = Choose(Button.Type + 1, "Button", _
    "Separator", "Control", "Flyout")
msg = msg & ButtonType & ": "
If Button.Type = acToolbarButton Or _
    Button.Type = acToolbarFlyout Then
    Button.GetBitmaps SmallButtonName, LargeButtonName
msg = msg + SmallButtonName + ", "+ LargeButtonName
End If
msg = msg + vbCrLf
Next Button
' Hiển thị kết quả
MsgBox msg
End Sub

```

## 5.5. Tạo thanh công cụ Flyout

Để thêm một nút dạng Flyout vào thanh công cụ, ta sử dụng phương thức `AddToolBarButton`. Phương thức này tạo mới một đối tượng `ToolbarItem` và thêm vào thanh công cụ đã định.

Phương thức `AddToolBarButton` cần nhập 5 đối số: `Index`, `Name`, `HelpString`, `Macro` và `FlyoutButton`. Bằng cách đặt tham biến `Flyout` là `TRUE`, nút mới vừa tạo sẽ là nút Flyout. Giá trị trả về của phương thức này sẽ là một thanh công cụ Flyout mới. Thanh công cụ Flyout có thể được xử lý như một thanh công cụ bình thường.

### Tạo nút dạng Flyout

Ví dụ sau đây tạo hai thanh công cụ. Thanh công cụ đầu tiên có chứa một nút Flyout. Thanh công cụ thứ hai dùng để gán vào nút Flyout của thanh công cụ đầu tiên.

```

Sub Ch6_AddFlyoutButton()
    Dim currMenuGroup As AcadMenuGroup
    Set currMenuGroup = ThisDrawing.Application. _
        MenuGroups.Item(0)
    ' Tạo thanh công cụ đầu tiên
    Dim FirstToolbar As AcadToolbar
    Set FirstToolbar = currMenuGroup.Toolbars. _
        Add("FirstToolbar")
    ' Thêm nút bấm Flyout vào thanh công cụ đầu tiên
    Dim FlyoutButton As AcadToolbarItem
    Set FlyoutButton = FirstToolbar.AddToolBarButton _

```

```

        ("", "Flyout", "Demonstrates a flyout button", _
        "OPEN", True)

' Tạo thanh công cụ thứ 2 và gán thanh công cụ này
' vào nút bấm Flyout của thanh công cụ thứ nhất
Dim SecondToolbar As AcadToolbar
Set SecondToolbar = currMenuGroup.Toolbars. _
    Add("SecondToolbar")

' Thêm một nút và gán vào thanh công cụ thứ 2
Dim newButton As AcadToolbarItem
Dim openMacro As String

' Gán chuỗi Macro "ESC ESC _open "
openMacro = Chr(3) + Chr(3) + Chr(95) + "open" + Chr(32)
Set newButton = SecondToolbar.AddToolbarButton _
    ("", "NewButton", "Open a tệp.", openMacro)

' Gán thanh công cụ thứ hai vào
' nút bấm flyout trên thanh công cụ thứ
FlyoutButton.AttachToolbarToFlyout currMenuGroup.Name, _
    SecondToolbar.Name

' Hiện thanh công cụ thứ nhất, ẩn thanh công cụ thứ 2
FirstToolbar.Visible = True
SecondToolbar.Visible = False
End Sub

```

## 5.6. Thanh công cụ nổi và thanh công cụ neo

Bằng cách lập trình, ta có thể làm cho thanh công cụ có thể ở dạng nổi (floating) hoặc ở dạng neo (docking).

Để làm nổi một thanh công cụ, ta sử dụng phương thức Float. Phương thức này cần nhập 3 đối số: Top, Left và NumberFloatRows. Các đối số Top và Left xác định vị trí của cạnh trên và cạnh trái của thanh công cụ. Đối số NumberFloatRows xác định số hàng để tạo một thanh công cụ theo hàng ngang. Con số này phải lớn hơn hoặc bằng 1. Các nút của thanh công cụ sẽ được trải đều trên số hàng đã định. Đối với thanh công cụ có chiều dọc, giá trị này xác định số cột cấu thành thanh công cụ đó.

Để neo một thanh công cụ, ta sử dụng phương thức Dock. Phương thức Dock cũng cần nhập ba đối số: Side, Row và Column. Đối số Side xác định cạnh bên nào của thanh công cụ mà ta cần đặt cố định. Ta cũng có thể xác định cụ thể cạnh trên, dưới hay trái, phải của thanh công cụ. Các đối số Row và Column xác định một con số hàng và cột của các thanh công cụ cố định có sẵn để neo thanh công cụ vào.

Ta có thể kiểm tra một thanh công cụ có được neo hay không, ta sử dụng thuộc tính Docked. Thuộc tính Docked sẽ hiển thị TRUE nếu thanh công cụ là được neo và FALSE nếu thanh công cụ ở dạng nổi.

### Neo thanh công cụ

Ví dụ sau tạo một thanh công cụ mới với ba nút. Sau đó sẽ hiển thị thanh công cụ và neo ở cạnh trái màn hình.

```

Sub Ch6_DockToolbar()
    Dim currMenuGroup As AcadMenuGroup
    Set currMenuGroup =
    ThisDrawing.Application.MenuGroups.Item(0)
    ' Tạo thanh công cụ mới
    Dim newToolbar As AcadToolbar
    Set newToolbar = currMenuGroup.Toolbars.Add("TestToolbar")
    ' Thêm 3 nút vào thanh công cụ.
    ' Các nút sẽ được gán cùng một Macro.
    Dim newButton1 As AcadToolbarItem
    Dim newButton2 As AcadToolbarItem
    Dim newButton3 As AcadToolbarItem
    Dim openMacro As String
    ' Gán macro "ESC ESC _open "
    openMacro = Chr(3) + Chr(3) + Chr(95) + "open" + Chr(32)
    Set newButton1 = newToolbar.AddToolbarButton
        ("", "NewButton1", "Open a file.", openMacro)
    Set newButton2 = newToolbar.AddToolbarButton
        ("", "NewButton2", "Open a file.", openMacro)
    Set newButton3 = newToolbar.AddToolbarButton
        ("", "NewButton3", "Open a file.", openMacro)
    ' Hiện thị thanh công cụ
    newToolbar.Visible = True
    ' Neo thanh công cụ vào bên trái của màn hình.
    newToolbar.Dock acToolbarDockLeft
End Sub

```

## 5.7. Xóa nút khỏi thanh công cụ

Để xóa nút khỏi một thanh công cụ, ta sử dụng phương thức Delete của nút đó. Ta chỉ nên xóa nút của thanh công cụ đang được hiển thị trên màn hình.

## 5.8. Tìm hiểu các thuộc tính của nút

Tất cả các nút đều có những thuộc tính sau đây:

### 5.8.1. Tag

Tag (thẻ tên) là một chuỗi gồm chữ, số và dấu gạch dưới (\_). Chuỗi này chỉ dành cho một nút duy nhất trong thanh công cụ đã cho. Thẻ tên được gán tự động khi tạo ra một nút.

Ta có thể đọc hoặc gán giá trị của thẻ tên thông qua thuộc tính Tag.

### 5.8.2. Name

Name (tên) là một chuỗi để nhận dạng nút và cũng là chuỗi chú giải, một dòng chữ sẽ hiện lên khi người dùng di chuột hoặc các thiết bị trợ khác chỉ lên một mục trình đơn.

Ta có thể đọc hoặc gán giá trị của tên thông qua thuộc tính Name.

### 5.8.3. Macro

Macro là một chuỗi các lệnh nhằm thực hiện một tác vụ cụ thể nào đó khi ta chọn một nút. Macro có thể đơn giản chỉ là một số thao tác bàn phím được lưu lại để thực hiện một nhiệm vụ hoặc cũng có thể là một sự kết hợp phức tạp các lệnh,

AutoLISP, DIESEL hoặc mã chương trình ActiveX. Để biết thêm chi tiết về Macro, xem “Macro trình đơn” trong “*AutoCAD Customization Guide*”.

Ta có thể đọc hoặc gán giá trị của Macro thông qua thuộc tính Macro.

#### 5.8.4. HelpString

HelpString (chuỗi trợ giúp) là một dòng chữ hiện lên trên thanh trạng thái của AutoCAD khi người dùng chỉ vào một nút.

Ta có thể đọc hoặc gán giá trị của chuỗi trợ giúp bằng cách sử dụng thuộc tính HelpString.

#### 5.8.5. Index

Index (chỉ số) của một nút xác định vị trí của nút đó trên thanh công cụ. Chỉ số vị trí của thanh công cụ luôn bắt đầu với vị trí 0. Chẳng hạn, nếu mục công cụ là mục đầu tiên trên công cụ, nó có chỉ số vị trí là 0, nếu là mục thứ hai thì chỉ số vị trí là 1, v.v..

Ta có thể đọc chỉ số vị trí của một mục công cụ thông qua thuộc tính Index.

#### 5.8.6. Type

Một nút có thể nằm trong những loại sau: nút thông thường, vạch ngăn, nút flyout hoặc nút điều khiển đặc biệt. Nếu là một nút thông thường, thuộc tính này có giá trị là `acButton`. Nếu là một vạch ngăn, thuộc tính có giá trị là `acToolButtonSeparator`. Nếu là một nút Flyout, thuộc tính có giá trị là `acFlyout`. Còn nếu là một nút điều khiển đặc biệt, thuộc tính có giá trị là `acControl`.

Ta có thể biết được loại của một nút công cụ thông qua thuộc tính Type.

#### 5.8.7. Flyout

Nếu nút thuộc kiểu `acFlyout`, thuộc tính Flyout sẽ trả về thanh công cụ được sử dụng làm thanh công cụ Flyout. Thanh công cụ Flyout là đối tượng Toolbar.

Nếu nút không có kiểu `acFlyout`, thuộc tính Flyout sẽ trả về giá trị NULL.

Ta có thể lấy giá trị thanh công cụ Flyout thông qua thuộc tính Flyout.

#### 5.8.8. Parent

Thuộc tính này trả về thanh công cụ chứa nút đang xét. Thanh công cụ gốc cũng là một đối tượng Toolbar.

Ta có thể lấy đối tượng thanh công cụ chứa một nút bằng cách sử dụng thuộc tính Parent của nút công cụ đó.

#### 5.8.9. Các thuộc tính khác

Còn có các thuộc tính khác áp dụng đối với tất cả các thanh công cụ như: xem thanh công cụ đó đang ở dạng nổi hay neo, hiện hay ẩn và sử dụng nút cỡ to hay cỡ nhỏ.

## 6. Tạo Macro

Macro là một chuỗi các lệnh nhằm thực hiện một tác vụ cụ thể nào đó khi ta chọn một nút. Macro có thể đơn giản là một số thao tác bàn phím được lưu lại để thực hiện một nhiệm vụ hoặc có thể là một sự kết hợp phức tạp các lệnh, AutoLISP, DIESEL hoặc mã chương trình ActiveX.

Nếu muốn gộp các tham số của một lệnh vào Macro của trình đơn, ta cần phải biết thứ tự các tham số của lệnh đó. Mỗi ký tự trong Macro của trình đơn đều có ý nghĩa, thậm chí cả dấu cách. Khi AutoCAD được chỉnh sửa và phát triển, trình tự một số lệnh (đôi khi cả tên lệnh) có thể bị thay đổi. Do đó, trình đơn tùy biến có thể cần phải có một vài thay đổi nhỏ khi nâng cấp lên phiên bản AutoCAD mới.

Khi đầu vào của lệnh là từ một mục trình đơn, thì ta thiết lập các biến hệ thống PICKADD và PICKAUTO có giá trị lần lượt là 1 và 0. Điều này đảm bảo khả năng tương thích với các phiên bản AutoCAD trước và việc tùy biến cũng sẽ dễ dàng hơn bởi vì ta không cần phải kiểm tra cài đặt của các biến số này.

### 6.1. Ký tự Macro và ký tự ASCII tương đương

Dưới đây là bảng tóm tắt một số các ký tự đặc biệt được sử dụng trong Macro trình đơn và các ký tự ASCII tương đương khi sử dụng trong VB và VBA. Việc sử dụng các ký tự ASCII tương đương với các ký tự này là rất quan trọng khi ta tạo một chuỗi cho thuộc tính Macro.

#### Các ký tự đặc biệt sử dụng trong macro trình đơn và macro công cụ

Ký tự	ASCII tương đương	Miêu tả
;	chr(59)	phím ENTER
^M	chr(94) + chr(77)	phím ENTER
^	chr(94) + chr(124)	phím TAB
SPACEBAR	chr(32)	nhập vào một dấu cách; khoảng trắng giữa các lệnh trong một mục trình đơn tương đương với nhấn phím SPACEBAR
\	chr(92)	dừng để người dùng nhập liệu
_	chr(95)	Thông dịch lệnh AutoCAD và các từ khoá tiếp theo
+	chr(43)	nối macro trình đơn với dòng tiếp theo (nếu là ký tự cuối)
=*	chr(61) + chr(42)	Hiển thị hình ảnh, trình đơn kéo xuống hoặc trình đơn tắt trên cùng hiện hành

*^C^C	chr(42) + chr(94) + chr(67) chr(94) + chr(67)	Tiền tố cho một mục được lặp đi lặp lại
\$	chr(36)	Tải một phần trình đơn hoặc bắt đầu một chuỗi DIESEL macro có điều kiện
^B	chr(94) + chr(66)	Bật hoặc tắt Snap (Ctrl+B)
^C	chr(94) + chr(67)	Hủy lệnh (Ctrl+C)
ESC	chr(3)	Hủy lệnh (ESC)
^D	chr(94) + chr(68)	Bật hoặc tắt Coords (Ctrl+D)
^E	chr(94) + chr(69)	Tạo một mặt phẳng đồng dạng tiếp theo (Ctrl+E)
^G	chr(94) + chr(71)	Bật hoặc tắt Grid (Ctrl+G)
^H	chr(94) + chr(72)	phím backspace
^O	chr(94) + chr(79)	Bật hoặc tắt Ortho (Ctrl+O)
^P	chr(94) + chr(80)	Bật hoặc tắt MENUCHO
^Q	chr(94) + chr(81)	Hồi lại tất cả các dòng nhắc, danh sách trạng thái và đầu vào máy in (Ctrl+Q)
^T	chr(94) + chr(84)	Bật hoặc tắt Tablet (Ctrl+T)
^V	chr(94) + chr(86)	Thay đổi khung nhìn hiện tại (Ctrl+V)
^Z	chr(94) + chr(90)	Ký tự trắng nhằm ngăn chặn việc thêm SPACEBAR một cách tự động ở cuối một mục trình đơn

## 6.2. Kết thúc Macro

Khi thực hiện một Macro, AutoCAD thêm một dấu cách ở cuối Macro trước khi thực hiện các dòng lệnh. AutoCAD xử lý đoạn Macro trình đơn sau giống như là ta đã nhập vào lệnh `line` và dấu cách.

Đôi khi, điều này không được mong muốn. Chẳng hạn, lệnh `TEXT` hoặc `DIM` phải được kết thúc bằng `ENTER` chứ không phải là một dấu cách. Hơn nữa, đôi khi phải sử dụng nhiều hơn một dấu cách (hoặc `ENTER`) thì mới hoàn tất một lệnh, nhưng một số các bộ soạn thảo không cho phép bạn nhập một hàng dài những khoảng trắng.

Có hai cách đặc biệt để giải quyết những vấn đề này:

- Khi trong Macro có dấu chấm phẩy (;), AutoCAD sẽ thay bằng ký tự ENTER
- Nếu một dòng kết thúc với một ký tự điều khiển, dấu gạch ngược (\), dấu cộng (+), hoặc dấu chấm phẩy (;), AutoCAD sẽ không thêm một khoảng trắng phía sau các ký tự này.

Xem Macro sau:

```
erase \;
```

Nếu Macro này chỉ kết thúc đơn giản bằng dấu gạch ngược (tức là yêu cầu người dùng nhập liệu), thì sẽ không thực hiện được lệnh ERASE, vì AutoCAD không thêm khoảng trắng sau dấu gạch ngược (\). Do đó, Macro này phải sử dụng thêm dấu chấm phẩy (;) để buộc AutoCAD thêm ENTER sau khi người dùng nhập liệu. Sau đây là một số ví dụ nữa:

```
ucs
ucs ;
text \.4 0 DRAFT Inc;;;Main St.;;;City, State;
```

Chọn Macro đầu tiên: nhập ucr và một dấu cách trên dòng lệnh, và dòng nhắc sau sẽ xuất hiện:

```
Enter an option
[New/Move/orthoGraphic/Prev/Restore/Save/Del/Apply/?/World]
<World>:
```

Chọn Macro thứ hai: nhập vào ucs, dấu cách và một dấu chấm hỏi trên dòng lệnh, sẽ chấp nhận giá trị mặc định, World. Rõ ràng không có sự khác nhau giữa macro thứ nhất và thứ hai được hiển thị trên màn hình.

Chọn Macro thứ ba: hiển thị một dòng nhắc về điểm bắt đầu và sau đó là địa chỉ trên 3 hàng. Trong bộ ba dấu chấm phẩy (;;;), dấu đầu tiên kết thúc chuỗi văn bản, dấu thứ hai nhằm lặp lại lệnh TEXT và dấu thứ ba gọi lại vị trí mặc định là phía sau dòng trước.

---

**CHÚ Ý** Tất cả các ký tự đặc biệt phải nhập bằng mã ASCII tương đương. Để biết danh sách mã ASCII tương đương, xem phần "Ký tự Macro và mã ASCII tương đương."

---

### 6.3. Dừng để người dùng nhập liệu

Đôi khi sẽ rất hữu ích nếu chấp nhận nhập liệu từ bàn phím hoặc các thiết bị trợ trong một Macro bằng cách đặt một dấu gạch ngược (\) tại nơi muốn nhập liệu.

```
circle \1
layer off \;
```

Macro đầu tiên dừng để yêu cầu người dùng xác định điểm tâm rồi lấy giá trị bán kính bằng 1 từ Macro. Chú ý rằng không có dấu cách sau dấu gạch ngược. Macro tiếp theo dừng để yêu cầu người dùng nhập vào tên một lớp, rồi tắt lớp đó đi và thoát khỏi lệnh LAYER. Lệnh LAYER thường dùng với các thao tác khác và chỉ thoát ra khi nào bạn nhấn phím cách (dấu trắng) hoặc ENTER (;).

Thông thường, Macro tiếp tục thực hiện sau khi nhập một nội dung nào đó. Do đó, không thể thiết lập một macro có vài lần dừng để nhập liệu (như trong quá trình chọn đối tượng) và sau đó tiếp tục. Tuy nhiên, có một ngoại lệ đó là khi sử dụng



lệnh SELECT; một dấu gạch ngược sẽ “treo” Macro đó cho đến khi đối tượng được chọn xong. Ví dụ như Macro sau đây:

```
select \change previous ;properties color red ;
```

Macro này sử dụng lệnh SELECT để thực hiện việc chọn một hay nhiều đối tượng. Sau đó, nó thực hiện lệnh CHANGE, đặt theo các tùy chọn trước và thay màu của tất cả các đối tượng đã chọn thành màu đỏ.

Do dấu gạch ngược (\) làm Macro dừng lại để người dùng nhập liệu, nên ta không thể sử dụng dấu này cho bất cứ mục đích nào khác trong Macro. Khi xác định đường dẫn tệp, sử dụng một dấu gạch xuôi (/) làm giới hạn đường dẫn, ví dụ, */direct/file*.

Các trường hợp sau đây khiến Macro không tiếp tục thực hiện:

- Nếu cần nhập vào một điểm, chế độ bắt đối tượng Object Snap có thể thực hiện trước khi có được một điểm thực cần.
- Nếu sử dụng bộ lọc điểm X/Y/Z, Macro sẽ vẫn treo cho đến khi ta nhập đủ tọa độ điểm.
- Riêng đối với lệnh SELECT, Macro sẽ không tiếp tục thực hiện cho đến khi chọn đối tượng xong.
- Nếu người dùng nhập liệu với các lệnh gián tiếp khác thì Macro sẽ vẫn tiếp tục dừng cho đến khi người dùng thực hiện xong các lệnh gián tiếp và dữ liệu cần nhập đã được thực hiện đủ.
- Nếu người dùng chọn một Macro khác (để thay đổi các lựa chọn hoặc để thực thi các lệnh gián tiếp), Macro nguồn sẽ vẫn bị treo và Macro vừa được chọn sẽ được thực hiện đến hết trước khi thực hiện tiếp Macro nguồn.

## 6.4. Hủy lệnh

Để chắc chắn rằng không còn một lệnh còn chưa thực hiện, ta sử dụng  $\wedge\wedge\wedge$  trong Macro. Đây là cách tương tự như nhấn phím ESC hai lần. Mặc dù chỉ cần một  $\wedge\wedge\wedge$  thôi cũng đã hủy lệnh trong hầu hết các trường hợp, nhưng  $\wedge\wedge\wedge$  là cần thiết để trở về dấu nhắc từ lệnh DIM. Do đó, ta dùng  $\wedge\wedge\wedge$  để đảm bảo rằng AutoCAD sẽ trở về dấu nhắc lệnh trong đa số trường hợp.

## 6.5. Lặp lại Macro

Khi đã chọn một lệnh, có thể ta sẽ sử dụng nó vài lần trước khi dùng lệnh khác. Đây là cách mà hầu hết mọi người đều làm với công cụ: chọn một công cụ, làm việc với nó vài lần rồi sau đó chọn công cụ khác và cứ thế... Để tránh việc chọn công cụ mỗi khi sử dụng, AutoCAD cung cấp khả năng lặp lại các lệnh. Tuy nhiên, ta không thể sử dụng tính năng này trong một vài lựa chọn của một số lệnh.

Tính năng này giúp lặp lại thường xuyên các lệnh đã dùng cho đến khi chọn lệnh khác. Nếu một Macro bắt đầu với  $\wedge\wedge\wedge$  và theo sau là nhãn nào đó, Macro sẽ được lưu vào bộ nhớ. Các dòng nhắc lệnh tiếp theo sẽ được Macro đó tự động trả lời cho đến khi ta kết thúc Macro đó bằng cách thực thi một Macro khác hoặc ấn phím ESC.

Không sử dụng ^C (hủy) trong một Macro bắt đầu bằng chuỗi \*^C^C, điều này sẽ làm hủy quá trình lặp lại của Macro đó.

Sau đây là một ví dụ của việc lặp lại Macro:

```
*^C^CMOVE Single
*^C^CCOPY Single
*^C^CERASE Single
*^C^CSTRETCH Single Crossing
*^C^CROTATE Single
*^C^CSCALE Single
```

## 6.6. Sử dụng chế độ chọn đối tượng đơn

Việc chọn đối tượng đơn sẽ chuyển chế độ chọn đối tượng sang chế độ chọn đối tượng đơn, điều này sẽ vô hiệu hóa các trình tự thông thường khi ta tiến hành lựa chọn đối tượng và sẽ trả về đối tượng đầu tiên trong các đối tượng đã được lựa chọn đó. Điều này cũng khá hữu ích khi sử dụng trong Macro. Lấy ví dụ đoạn Macro sau:

```
*^C^CERASE single
```

Đoạn Macro này sẽ kết thúc lệnh đang thực hiện và kích hoạt lệnh ERASE với chế độ chọn đối tượng đơn. Các đối tượng được lựa chọn sẽ bị xóa đi, và Macro này sẽ được lặp lại (do có dấu sao ở đầu đoạn Macro) để ta có thể tiếp tục xóa các đối tượng khác. Chế độ lựa chọn đối tượng đơn sẽ tạo thêm nhiều tương tác động với môi trường AutoCAD.

## 7. Tạo dòng trạng thái trợ giúp cho các mục trong trình đơn và nút trên thanh công cụ

Các thông điệp trợ giúp trên thanh trạng thái là một phần quan trọng của chức năng hỗ trợ sẵn có. Chúng là những thông điệp đơn giản và có ý nghĩa, xuất hiện trên thanh trạng thái khi ta di chuột đến một trình đơn hoặc thanh công cụ. Dòng trạng thái trợ giúp của các mục của trình đơn và nút trên thanh công cụ đều được gán thông qua thuộc tính HelpString.

Thuộc tính HelpString được để trống khi một mục trên trình đơn hoặc trên thanh công cụ vừa được tạo lần đầu.

### Thêm dòng trạng thái trợ giúp cho mục trình đơn

Dưới đây là ví dụ tạo một trình đơn mới tên "TestMenu" rồi tạo một mục "Open". Mục này sau đó được gán chuỗi trợ giúp thông qua thuộc tính HelpString.

```
Sub Ch6_AddHelp()
    Dim currMenuGroup As AcadMenuGroup
    Set currMenuGroup =
        ThisDrawing.Application.MenuGroups.Item(0)
    ' Tạo trình đơn mới
    Dim newMenu As AcadPopupMenu
    Set newMenu = currMenuGroup.Menus.Add _
        ("Te" + Chr(Asc("&"))) + "stMenu")
    ' Thêm một mục vào trình đơn mới
    Dim newItem As AcadPopupMenuItem
    Dim openMacro As String
    ' Gán lệnh "ESC ESC _open "
```

```

openMacro = Chr(3) + Chr(3) + Chr(95) + "open" + Chr(32)
' Tạo mục "Open" trên trình đơn
Set newItem = newMenu.AddItem _
    (newMenu.count + 1, Chr(Asc("&")) _
    + "Open", openMacro)
' Thêm chuỗi trợ giúp cho mục "Open" trong trình đơn
newMenuItem.HelpString = "Opens an AutoCAD drawing file."
' Hiện thị trình đơn mới
newMenu.InsertInMenuBar _
    (ThisDrawing.Application.menuBar.count + 1)
End Sub

```

## 8. Thêm mục vào trình đơn tắt

Trình đơn tắt, là một trình đơn đặc biệt có trong nhóm trình đơn cơ bản của AutoCAD. Trình đơn này xuất hiện khi người dùng giữ phím SHIFT và nhấn chuột phải.

AutoCAD nhận biết trình đơn này bằng cách tìm trong nhóm trình đơn cơ bản một trình đơn có thuộc tính ShortcutMenu là TRUE. Ta có thể thêm một mục mới vào trình đơn tắt theo các bước được liệt kê trong phần *“Thêm mục mới vào một trình đơn”* trang 182.

Các nhóm trình đơn mới có thể chứa hoặc không chứa trình đơn tắt. Để tạo một trình đơn tắt cho một nhóm trình đơn, theo các bước ở phần *“Tạo trình đơn mới”* trang 181 và sử dụng POPO làm nhãn của trình đơn mới.

### Thêm một mục mới vào cuối trình đơn tắt

Dưới đây là ví dụ thêm mục có tên “OpenDWG” vào cuối trình đơn tắt.

```

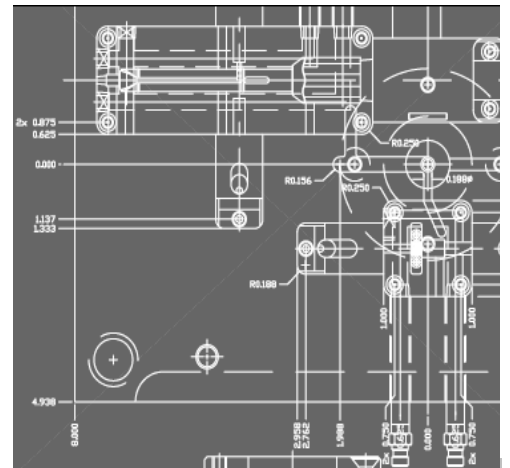
Sub Ch6_AddMenuItemToShortcutMenu()
    Dim currMenuGroup As AcadMenuGroup
    Set currMenuGroup = ThisDrawing.Application.MenuGroups.Item(0)
    ' Tìm trình đơn tắt và gán nó cho biến shortcutMenu
    Dim scMenu As AcadPopupMenu
    Dim entry As AcadPopupMenu
    For Each entry In currMenuGroup.Menus
        If entry.ShortcutMenu = True Then
            Set scMenu = entry
        End If
    Next entry
    ' Thêm một mục mới và trình đơn tắt
    Dim newItem As AcadPopupMenu
    Dim openMacro As String
    ' Gán lệnh "ESC ESC _open "
    openMacro = Chr(3) + Chr(3) + Chr(95) + "open" + Chr(32)
    Set newItem = scMenu.AddItem _
        ("", Chr(Asc("&")) _
        + "OpenDWG", openMacro)
End Sub

```

Tài liệu tham khảo  
BM Tự động hoá TKCĐ

# LÀM VIỆC VỚI CÁC SỰ KIỆN

Các sự kiện chẳng hạn như thông báo hoặc thông điệp của AutoCAD, dùng để thông báo tới người dùng về trạng thái hiện tại của phiên làm việc hoặc để cảnh báo cho người dùng về những gì đã xảy ra. Ví dụ, khi mở bản vẽ thì sự kiện BeginOpen được gọi, sự kiện này chứa tên bản vẽ AutoCAD được mở; còn khi đóng bản vẽ, sẽ có một sự kiện khác được gọi. Khi có được những thông tin này ta có thể xây dựng được chương trình giúp xác định được thời gian người dùng dành cho một bản vẽ nào đó.



Trong chương này

7

- **Khái niệm về các sự kiện trong AutoCAD**
- **Chỉ dẫn xây dựng bộ xử lý sự kiện**
- **Xử lý sự kiện ở mức ứng dụng**
- **Xử lý sự kiện ở mức bản vẽ**
- **Xử lý sự kiện ở mức đối tượng**

# 1. Khái niệm về các sự kiện trong AutoCAD

Trong AutoCAD có 3 loại sự kiện:

- Các sự kiện ở mức ứng dụng sẽ đáp ứng lại những thay đổi trong ứng dụng AutoCAD và môi trường của nó bao gồm: mở, lưu, đóng và in các bản vẽ, tạo bản vẽ mới, sử dụng dòng lệnh, tải hoặc dỡ bỏ ứng dụng ARX và AutoLISP, thay đổi các biến hệ thống và thay đổi đối với cửa sổ của ứng dụng.
- Các sự kiện ở mức bản vẽ sẽ đáp ứng lại những thay đổi đối với từng bản vẽ hoặc những gì bên trong nó bao gồm: thêm, xoá hoặc thay đổi các đối tượng, kích hoạt trình đơn tắt, thay đổi trong tập lựa chọn, thay đổi đối với cửa sổ vẽ và quá trình tái tạo bản vẽ. Ngoài ra cũng có một số các sự kiện ở mức này đáp ứng lại với việc mở, đóng và in một bản vẽ, tải hoặc dỡ bỏ các ứng dụng ARX và AutoLISP từ bản vẽ.
- Các sự kiện ở mức đối tượng đáp ứng lại các thay đổi của một đối tượng cụ thể. Hiện tại chỉ có một sự kiện duy nhất ở mức này, nó được gọi mỗi khi đối tượng bị thay đổi.

Các hàm đáp ứng lại các sự kiện được gọi là bộ xử lý sự kiện và được thực hiện một cách tự động mỗi khi sự kiện tương ứng được gọi. Các thông tin chứa trong các sự kiện, ví dụ như tên bản vẽ trong sự kiện BeginOpen, được chuyển vào bộ xử lý sự kiện thông qua các tham số.

## 2. Chỉ dẫn xây dựng bộ xử lý sự kiện

Điều quan trọng cần lưu ý là các sự kiện chỉ đơn giản cung cấp các thông tin dựa trên trạng thái hoặc các hoạt động diễn ra trong AutoCAD. Mặc dù bộ xử lý sự kiện có thể được viết để đáp ứng lại các sự kiện đó, nhưng AutoCAD lại nằm bên trong quá trình xử lý khi các bộ xử lý sự kiện được gọi. Do đó các bộ xử lý sự kiện thường có những hạn chế về những tác vụ có thể thực hiện được nhằm đảm bảo sự an toàn cho các thao tác tác động đến AutoCAD và cơ sở dữ liệu.

- Không nên dựa vào trình tự của các sự kiện  
Khi viết các bộ xử lý sự kiện ta không nên dựa vào trình tự xảy ra của các sự kiện như theo mình suy nghĩ. Ví dụ: khi sử dụng lệnh OPEN, sự kiện BeginCommand, BeginOpen, EndOpen và EndCommand sẽ đều được thực hiện. Tuy nhiên, chúng có thể không xảy ra theo trật tự đó. Trình tự sự kiện duy nhất có thể sử dụng một cách an toàn là sự kiện Begin sẽ xảy ra trước sự kiện End tương ứng. Trong ví dụ trước, sự kiện có thể được xảy ra theo trình tự sau: BeginCommand – BeginOpen – EndCommand – EndOpen hoặc BeginCommand – EndCommand – BeginOpen – EndOpen.
- Không nên dựa vào trình tự của các thao tác xử lý  
Nếu ta xoá đối tượng 1 và tiếp theo là đối tượng 2, không nên cho rằng sự kiện ObjectErased cho đối tượng 1 phát sinh trước rồi sau đó cho đối tượng 2, vì thực tế, ta có thể sẽ nhận được sự kiện của đối tượng 2 trước.
- Không nên dùng các hàm tương tác trong bộ xử lý sự kiện

Cố gắng thực hiện một hàm tương tác trong bộ xử lý sự kiện có thể dẫn đến nhiều rắc rối khá nghiêm trọng, vì AutoCAD có thể vẫn tiếp tục thực hiện một câu lệnh nào đó vào thời điểm mà sự kiện bắt đầu. Vì vậy, nên tránh sử dụng các phương thức đòi hỏi phải nhập dữ liệu, ví dụ như GetPoint, GetEntity, GetKeyword... cũng như các thao tác chọn đối tượng và phương thức SendCommand trong bất cứ bộ xử lý sự kiện nào.

- Không nên gọi hộp thoại trong bộ xử lý sự kiện

Hộp thoại được coi là hàm tương tác, có thể cản trở các thao tác đang sử dụng trong AutoCAD. Tuy nhiên các hộp thông báo hoặc cảnh báo được coi là không có sự tương tác do đó có thể sử dụng một cách an toàn.

- Ta có thể ghi dữ liệu vào cơ sở dữ liệu của bất cứ đối tượng nào ngoại trừ đối tượng phát sinh sự kiện.

Một cách hiển nhiên là bất cứ đối tượng nào làm phát sinh một sự kiện đang được xử lý có thể vẫn được mở để sử dụng trong AutoCAD và các thao tác vẫn đang được tiến hành. Do đó nên tránh việc ghi bất cứ thông tin nào cho đối tượng từ bộ xử lý sự kiện của chính đối tượng đó. Tuy nhiên, ta vẫn có thể đọc thông tin một cách an toàn từ đối tượng phát sinh sự kiện. Ví dụ: ta có một sàn nhà được lát gạch và đã có bộ xử lý sự kiện được đính kèm với đường biên sàn nhà. Nếu ta thay đổi kích thước của sàn nhà thì bộ xử lý sự kiện sẽ tự động cộng hoặc trừ các viên gạch để phù hợp diện tích sàn mới. Bộ xử lý sự kiện sẽ vẫn có thể đọc diện tích mới của đường biên nhưng nó không thể thay đổi được bất cứ thứ gì trên biên đó.

- Không nên thực hiện bất cứ thao tác nào từ bộ xử lý sự kiện mà sẽ làm phát sinh chính sự kiện đó.

Nếu ta thực hiện cùng một thao tác trong bộ xử lý sự kiện mà sẽ làm phát sinh sự kiện đó thì sẽ tạo ra một vòng lặp vô hạn. Ví dụ người dùng không nên cố gắng mở một bản vẽ bên trong sự kiện BeginOpen, nếu không AutoCAD sẽ tiếp tục mở thêm nhiều bản vẽ khác cho tới khi đạt đến số lượng bản vẽ được mở lớn nhất.

- Lưu ý rằng không có sự kiện nào được thực hiện khi AutoCAD đang hiển thị hộp thoại kiểu Modal<sup>1</sup>.

### 3. Xử lý sự kiện ở mức ứng dụng

Các sự kiện ở mức ứng dụng không duy trì liên tục trong AutoCAD VBA, tức là chúng không được phép hoạt động một cách tự động khi tải một dự án VBA. Do vậy, các sự kiện này cần được kích hoạt đối với VBA và tất cả các điều khiển ActiveX Automation.

---

<sup>1</sup> **Hộp thoại kiểu Modal (Modal dialog):** là dạng hộp thoại mà sẽ chặn tất cả các tương tác với tất cả các hộp thoại hiển thị trước nó ở trong cùng một ứng dụng. Chỉ những hộp thoại khác ứng dụng hoặc những hộp thoại được gọi từ bản thân hộp thoại modal mới có thể nhận được các tương tác từ phía người dùng.



Một khi sự kiện ở mức ứng dụng được kích hoạt, có rất nhiều sự kiện mà ta có thể sử dụng, bao gồm:

Sự kiện	Tình huống
AppActivate	Gọi đến ngay trước khi cửa sổ chính của ứng dụng được kích hoạt.
AppDeactivate	Gọi đến trước khi cửa sổ chính của ứng dụng được bỏ kích hoạt.
ARXLoaded	Gọi đến khi một ứng dụng ObjectARX được tải vào.
ARXUnloaded	Gọi đến khi một ứng dụng ObjectARX được dỡ ra.
BeginCommand	Gọi đến ngay sau một câu lệnh được sử dụng nhưng trước khi nó hoàn thành.
BeginTệpDrop	Gọi đến khi một tệp bị dỡ ra khỏi ứng dụng.
BeginLISP	Gọi đến ngay sau khi AutoCAD nhận được yêu cầu tính biểu thức LISP.
BeginModal	Gọi đến ngay trước khi modal dialog hiện ra.
BeginOpen	Gọi đến ngay sau khi AutoCAD nhận yêu cầu mở một tệp đã có.
BeginPlot	Gọi đến ngay sau khi AutoCAD nhận được yêu cầu in một bản vẽ.
BeginQuit	Gọi đến ngay trước khi một phiên làm việc của AutoCAD kết thúc.
BeginSave	Gọi đến ngay sau khi AutoCAD nhận được yêu cầu lưu một bản vẽ.
EndCommand	Gọi đến ngay sau khi một câu lệnh hoàn thành.
EndLISP	Gọi đến khi hoàn thành tính toán một biểu thức LISP.
EndModal	Gọi đến ngay sau khi hộp thoại bị đóng.
EndOpen	Gọi đến ngay sau khi AutoCAD kết thúc mở một bản vẽ đã có.
EndPlot	Gọi đến ngay sau khi một tài liệu được gửi đến máy in.
EndSave	Gọi đến ngay sau khi AutoCAD kết thúc lưu một bản vẽ.
LISPCancelled	Gọi đến ngay sau khi việc tính toán một biểu thức LISP bị dừng lại.
NewDrawing	Gọi đến trước khi tạo ra một bản vẽ mới.
SysVarChanged	Gọi đến khi giá trị của một biến hệ thống bị thay đổi
WindowChanged	Gọi đến khi có sự thay đổi cửa sổ ứng dụng.
WindowMovedOrResized	Gọi đến ngay sau khi cửa sổ ứng dụng đã được dịch chuyển hoặc thay đổi kích thước.

### 3.1. Kích hoạt sự kiện ở mức ứng dụng

Trước khi sử dụng sự kiện ở mức ứng dụng người dùng phải tạo ra một lớp mới và khai báo một đối tượng có kiểu là AcadApplication cùng với các sự kiện. Ví dụ, giả sử có một lớp mới được tạo ra gọi là EventClassModule, lớp mới này chứa các khai báo của ứng dụng với từ khoá của VBA là WithEvents.

#### Để tạo một lớp mới và khai báo đối tượng Application với các sự kiện:

- 1 Trong VBA IDE, để thêm một lớp: Insert ▶ ClassModule.
- 2 Chọn lớp mới được tạo trong cửa sổ Project
- 3 Đổi tên của lớp trong cửa sổ Properties thành EventClassModule
- 4 Mở cửa sổ Code của lớp bằng cách bấm phím F7 hoặc lựa chọn: View ▶ Code
- 5 Trong cửa sổ Code của lớp thêm dòng lệnh sau:

```
Public WithEvents App As AcadApplication
```

Sau khi một đối tượng đã được khai báo với các sự kiện, nó sẽ xuất hiện trong hộp danh sách Object của lớp và người dùng có thể viết các thủ tục sự kiện cho đối tượng mới trong lớp. (Khi lựa chọn một đối tượng mới trong hộp Object, các sự kiện có hiệu lực đối với đối tượng đó đều có trong hộp danh sách Procedure).

Tuy nhiên, trước khi các thủ tục được thực hiện ta phải kết nối những đối tượng được khai báo trong lớp với đối tượng Application. Dưới đây là đoạn mã mà ta có thể chèn vào bất cứ môđun nào để thực hiện việc kết nối trên.

#### Để kết nối đối tượng được khai báo với đối tượng Application

- 1 Trong cửa sổ Code của môđun chính, thêm dòng lệnh sau trong phần khai báo:

```
Dim X as New EventClassModule
```

- 2 Trong cùng cửa sổ đó thêm vào thủ tục:

```
Sub InitializeEvent()  
    Set X.App= ThisDrawing.Application  
End Sub
```

- 3 Trong phần mã của môđun chính, thêm lời gọi thủ tục InitializeApp

```
Call InitializeEvents
```

Khi thủ tục InitializeEvents được thực hiện thì đối tượng App trong lớp sẽ trở đến một đối tượng Application nhất định và bất cứ thủ tục sự kiện nào trong lớp sẽ chạy khi sự kiện đó xảy ra.

#### Lời nhắc tiếp tục khi một bản vẽ được thả vào AutoCAD

Ví dụ này chặn lại quá trình tải khi một tệp đã được kéo và thả vào AutoCAD. Khi đó, sẽ có hộp thoại thông báo gồm tên tệp, các nút lệnh Yes/No/Cancel cho phép người dùng quyết định xem có tiếp tục tải tệp đó nữa hay không. Nếu người dùng lựa chọn thoát khỏi thao tác, quyết định đó sẽ được thực hiện nhờ việc trả về thông số Cancel của sự kiện BeginFileDrop và tệp đó sẽ không được tải.

```
Public WithEvents ACADApp As AcadApplication  
Sub Example_AcadApplication_Events()
```

```

' Ví dụ này khởi tạo biến toàn cục (ACADApp)
' biến này sẽ được sử dụng để chặn sự kiện AcadApplication
' Chạy thủ tục này trước tiên!
' Có thể gọi được ứng dụng từ đối tượng ThisDocument,
' nhưng yêu cầu có bản vẽ đang mở
' nên bản vẽ sẽ được lấy được từ hệ thống
Set ACADApp = GetObject(, "AutoCAD.Application")
End Sub

Private Sub ACADApp_BeginTệpDrop _
    (ByVal FileName As String, Cancel As Boolean)
' Ví dụ này chặn sự kiện BeginFileDrop của ứng dụng
' Sự kiện này được bắt đầu khi một bản vẽ được kéo vào
AutoCAD.
' Để bắt đầu cho ví dụ này:
' 1) Phải chạy ví dụ khởi tạo trước, trong ví dụ này biến
' toàn cục là ACADApp đã được liên kết với sự kiện này
' 2) Kéo một tệp bản vẽ AutoCAD vào chương trình AutoCAD
' từ Windows Desktop hoặc từ Windows Explorer
' Sử dụng biến "Cancel" để dừng việc tải tệp
' và biến "TệpName" để thông báo cho người dùng
' tệp nào sắp được thả vào.
If MsgBox("AutoCAD is about to load " & FileName & vbCrLf _
    & "Do you want to continue loading this file?", _
    vbYesNoCancel + vbQuestion) <> vbYes Then
    Cancel = True
End If
End Sub

```

## 4. Xử lý sự kiện ở mức bản vẽ

Các sự kiện ở mức bản vẽ luôn duy trì liên tục trong VBA của AutoCAD, nghĩa là chúng sẽ được tự động kích hoạt khi tải một dự án VBA. Tuy nhiên, chúng không được kích hoạt đối với bất cứ điều khiển nào khác, chẳng hạn như VB. Do đó các sự kiện ở mức bản vẽ cần được kích hoạt đối với tất cả các điều khiển ActiveX Automation khác.

Mỗi khi một sự kiện ở mức bản vẽ được kích hoạt, ta có nhiều sự kiện để sử dụng, bao gồm:

Sự kiện	Tình huống
Activate	Khi cửa sổ bản vẽ được kích hoạt
BeginClose	Ngay trước khi một bản vẽ đóng lại
BeginCommand	Ngay sau khi một câu lệnh được sử dụng và trước khi hoàn thành
BeginDoubleClick	Sau khi người dùng nhấn đúp lên một đối tượng trong bản vẽ
BeginLISP	Ngay sau khi AutoCAD nhận được yêu cầu tính một biểu thức LISP
BeginPlot	Ngay sau khi AutoCAD nhận được yêu cầu in một bản vẽ

BeginRightClick	Sau khi người dùng nhấp chuột phải lên cửa sổ bản vẽ
BeginSave	Ngay sau khi AutoCAD nhận được yêu cầu lưu một bản vẽ
BeginShortcutMenuCommand	Sau khi người dùng nhấp chuột phải lên cửa sổ bản vẽ và trước khi trình đơn tắt xuất hiện ở dạng Command
BeginShortcutMenuDefault	Sau khi người dùng nhấp chuột phải lên cửa sổ bản vẽ và trước khi trình đơn tắt xuất hiện ở dạng Default
BeginShortcutMenuEdit	Sau khi người dùng nhấp chuột phải lên cửa sổ bản vẽ và trước khi trình đơn tắt xuất hiện ở dạng Edit
BeginShortcutMenuGrip	Sau khi người dùng nhấp chuột phải lên cửa sổ bản vẽ và trước khi trình đơn tắt xuất hiện ở dạng Grip
BeginShortcutMenuOsnap	Sau khi người dùng nhấp chuột phải lên cửa sổ bản vẽ và trước khi trình đơn tắt xuất hiện ở dạng Osnap
Deactivate	Khi cửa sổ bản vẽ thôi kích hoạt
EndCommand	Ngay sau khi câu lệnh hoàn thành
EndLISP	Khi hoàn thành tính toán biểu thức LISP
EndPlot	Sau khi một bản vẽ được gửi đến máy in
EndSave	Khi AutoCAD hoàn thành lưu bản vẽ
EndShortcutMenu	Sau khi trình đơn tắt xuất hiện
LayoutSwitched	Sau khi người dùng chuyển sang Layout khác
LISPCancelled	Khi việc tính toán biểu thức LISP bị dừng lại
ObjectAdded	Khi một đối tượng được thêm vào bản vẽ
ObjectErased	Khi một đối tượng bị xoá khỏi bản vẽ
ObjectModified	Khi một đối tượng trong bản vẽ bị sửa
SelectionChanged	Khi tập lựa chọn đầu tiên hiện tại bị thay đổi
WindowChanged	Khi có sự thay đổi đối với cửa sổ bản vẽ
WindowMovedOrResized	Ngay sau khi cửa sổ bản vẽ dịch chuyển hoặc thay đổi kích cỡ

#### 4.1. Kích hoạt sự kiện trong các môi trường ngoài VBA

Trước khi sử dụng sự kiện ở mức bản vẽ trong môi trường VB hoặc các môi trường khác ngoài VBA, ta cần tạo ra một lớp mới và khai báo đối tượng với kiểu là AcadDocument cùng với các sự kiện. Ví dụ: giả sử đã tạo một lớp mới có tên là EventClassModule, lớp mới này chứa các khai báo của ứng dụng theo từ khóa WithEvents của VBA.

##### **Để tạo một lớp mới và khai báo đối tượng Document với các sự kiện:**

- 1 Trong VBA IDE, thêm một lớp: Insert ► ClassModule.

- 2 Chọn lớp mới trong cửa sổ Project
- 3 Đổi tên của lớp trong cửa sổ Properties thành EventClassModule
- 4 Mở cửa sổ Code của lớp bằng cách bấm phím F7 hoặc chọn: View ▶ Code
- 5 Trong cửa sổ Code của lớp thêm dòng lệnh sau:

```
Public WithEvents Doc As AcadDocument
```

Sau khi một đối tượng đã được khai báo với các sự kiện, nó sẽ xuất hiện trong hộp danh sách Object của lớp và người dùng có thể viết các thủ tục sự kiện cho đối tượng mới của lớp (Khi lựa chọn một đối tượng mới trong hộp Object, các sự kiện có hiệu lực đối với đối tượng đó đều có trong hộp danh sách Procedure).

Tuy nhiên, trước khi các thủ tục được thực hiện ta phải kết nối những đối tượng được khai báo trong lớp với đối tượng Document. Dưới đây là đoạn mã mà ta có thể chèn vào bất cứ môđun nào để thực hiện việc kết nối trên:

### Kết nối đối tượng được khai báo với đối tượng Document

- 1 Trong cửa sổ Code của môđun chính, thêm dòng lệnh sau trong phần các khai báo:

```
Dim X as New EventClassModule
```

- 2 Trong cùng cửa sổ đó thêm vào thủ tục:

```
Sub InitializeEvent()  
    Set X.Doc= ThisDrawing  
End Sub
```

- 3 Trong phần mã của môđun chính, thêm lời gọi thủ tục InitalizeApp

```
Call InitializeEvents
```

Khi thủ tục InitializeEvents được thực hiện thì đối tượng Doc trong lớp sẽ trở đến một đối tượng Document nhất định và bất cứ thủ tục sự kiện nào trong lớp sẽ chạy khi sự kiện đó xảy ra.

## 4.2. Lập trình trong các môi trường khác VBA

Khi sự kiện ở mức bản vẽ đã được cho phép, ta sẽ tìm được biến của lớp Doc có trong hộp danh sách của cửa sổ mã chương trình của ClassModule. Chọn lớp Doc và liệt kê danh sách các sự kiện sẵn có sẽ xuất hiện trong hộp danh sách Procedure. Chỉ cần chọn sự kiện muốn viết xử lý, khung xử lý sẽ được tự động tạo ra.

## 4.3. Lập trình trong môi trường VBA

Như đã đề cập ở trên trong phần trên, các sự kiện ở mức bản vẽ đã được tự động cho phép khi dự án VBA được tải vào. Để viết xử lý sự kiện cho các sự kiện này trong VBA, cần lựa chọn AcadDocument từ hộp danh sách trong cửa sổ Code. Các sự kiện có sẵn trong bản vẽ sẽ xuất hiện trong hộp danh sách Procedure. Chỉ cần lựa chọn sự kiện muốn xử lý, khung xử lý sẽ được tự động tạo ra.

Chú ý rằng các xử lý sự kiện giới thiệu ở đây áp dụng cho bản vẽ hiện tại đang được kích hoạt. Để viết xử lý sự kiện cho một bản vẽ nhất định, trước hết thực hiện các

bước đã nêu trong phần “Kích hoạt sự kiện trong các môi trường ngoài VBA” trang 209, các thủ tục đó sẽ cho phép ta làm việc với bản vẽ cụ thể.

### **Cập nhật trình đơn tắt với các sự kiện BeginShortcutMenuDefault và EndShortcutMenu**

Ví dụ này sử dụng bộ xử lý sự kiện cho sự kiện BeginShortcutMenuDefault để thêm chỉ mục “OpenDWG” ngay từ trước cho trình đơn tắt. Sau đó xử lý sự kiện cho EndShortcutMenu sẽ xoá chỉ mục thêm vào để nó không được lưu liên tục trong định dạng trình đơn của người dùng.

```
Private Sub AcadDocument_BeginShortcutMenuDefault _  
    (ShortcutMenu As AutoCAD.IAcadPopupMenu)  
    On Error Resume Next  
    ' Thêm chỉ mục cho trình đơn tắt  
    Dim newItem As AcadPopupMenuItem  
    Dim openMacro As String  
    openMacro = Chr(27) + Chr(27) + Chr(95) + "open" + Chr(32)  
    Set newItem = ShortcutMenu.AddItem _  
        (0, Chr(Asc("&")) _  
        + "OpenDWG", openMacro)  
End Sub  
  
Private Sub AcadDocument_EndShortcutMenu _  
    (ShortcutMenu As AutoCAD.IAcadPopupMenu)  
    On Error Resume Next  
    ShortcutMenu.Item("OpenDWG").Delete  
End Sub
```

## **5. Xử lý sự kiện ở mức đối tượng**

Các sự kiện ở mức đối tượng không thường trực trong AutoCAD VBA. Có nghĩa là chúng sẽ không tự hoạt động khi một dự án VBA được tải vào. Một sự kiện ở mức đối tượng phải được cho phép đối với VBA và tất cả các điều khiển ActiveX Automation khác.

Mỗi khi một sự kiện ở mức đối tượng được cho phép, chỉ có một sự kiện sau có thể sử dụng:

<b>Sự kiện</b>	<b>Tình huống</b>
Modified	Khi đối tượng trong bản vẽ bị sửa đổi

### **5.1. Kích hoạt sự kiện ở mức đối tượng**

Trước khi có thể sử dụng đối tượng ở mức sự kiện, ta cần tạo ra một module lớp mới và khai báo đối tượng với kiểu là AcadObject cùng với các sự kiện. Ví dụ: giả sử đã tạo một mô-đun lớp mới có tên là EventClassModule. Mô-đun này chứa các khai báo của ứng dụng theo từ khoá WithEvents của VBA.

#### **Tạo một mô-đun lớp mới và khai báo một đối tượng Circle với các sự kiện:**

- 1 Trong VBA IDE, thêm một mô-đun lớp: chọn menu Insert ▶ ClassModule.
- 2 Chọn mô-đun lớp mới trong cửa sổ Project

- 3 Đổi tên của lớp trong cửa sổ Properties thành EventClassModule
- 4 Mở cửa sổ Code của lớp bằng cách dùng phím tắt F7 hoặc bằng cách lựa chọn menu View ▶ Code
- 5 Trong cửa sổ Code của lớp thêm dòng lệnh sau:

```
Public WithEvents Object As AcadCircle
```

Sau khi một đối tượng đã được khai báo với các sự kiện, nó sẽ xuất hiện trong hộp danh sách Object trong mô-đun lớp và người dùng có thể viết các thủ tục sự kiện cho đối tượng mới trong mô-đun lớp (Khi lựa chọn một đối tượng mới trong hộp Object, các sự kiện có hiệu lực đối với đối tượng đó đều có trong hộp danh sách Procedure).

Tuy nhiên, trước khi các thủ tục được thực hiện ta phải kết nối những đối tượng được khai báo trong mô-đun lớp với đối tượng Circle. Dưới đây là đoạn mã mà ta có thể chèn vào bất cứ mô-đun nào để thực hiện việc kết nối trên.

### Kết nối đối tượng đã được khai báo với đối tượng Automation

- 1 Trong cửa sổ Code của mô-đun chính, thêm dòng lệnh sau trong phần các khai báo:

```
Dim x as New EventClassModule
```

- 2 Trong cùng cửa sổ đó, tạo một đường tròn có tên là “MyCircle” và khởi tạo nó với các sự kiện:

```
Sub InitializeEvents()
    Dim MyCircle As AcadCircle
    Dim centerPoint(0 To 2) As Double
    Dim radius As Double
    centerPoint(0) = 0#: centerPoint(1) = 0#: centerPoint(2) = 0#
    radius = 5#
    Set MyCircle =
    ThisDrawing.ModelSpace.AddCircle(centerPoint, radius)
    Set X.Object = MyCircle
End Sub
```

- 3 Trong phần mã của mô-đun chính, thêm lời gọi thủ tục InitalizeApp:

```
Call InitializeEvents
```

Khi thủ tục InitializEvents được thực hiện thì đối tượng Circle trong mô-đun lớp sẽ trở đến một đối tượng Circle đã được tạo và bất cứ thủ tục sự kiện nào trong mô-đun lớp sẽ chạy khi sự kiện đó xảy ra.

---

**CHÚ Ý** Khi lập trình trong VBA, cần phải cung cấp xử lý sự kiện Modified cho tất cả các đối tượng được cho phép hiệu chỉnh. Nếu không được xử lý, chương trình có thể bị ngắt không theo mong muốn.

---

### Hiển thị diện tích của đường đa tuyến kín mỗi khi nó được cập nhật

Ví dụ này tạo ra một đường đa tuyến có bề dày cùng với các sự kiện. Bộ xử lý sự kiện cho đường đa tuyến sẽ thể hiện diện tích mới mỗi khi nó bị thay đổi. Để gọi sự kiện này chỉ cần thay đổi kích cỡ của nó trong AutoCAD và lưu ý rằng, phải chạy thủ tục CreatePlineWithEvents trước khi xử lý sự kiện được kích hoạt.



```

Public WithEvents PLine As AcadLWPolyline

Sub CreatePLineWithEvents()
    ' Ví dụ này sẽ tạo ra đường light weight polyline
    Dim points(0 To 9) As Double
    points(0) = 1: points(1) = 1
    points(2) = 1: points(3) = 2
    points(4) = 2: points(5) = 2
    points(6) = 3: points(7) = 3
    points(8) = 3: points(9) = 2
    Set PLine = ThisDrawing.ModelSpace. _
        AddLightWeightPolyline(points)
    PLine.Closed = True
    ThisDrawing.Application.ZoomAll
End Sub

Private Sub PLine_Modified (ByVal pObject As AutoCAD.IAcadObject)
    ' Sự kiện này được gọi khi đường polyline bị thay đổi kích thước
    ' Nếu đường polyline bị xoá thì sự kiện modified vẫn được gọi ra nên
    ' có thể sử dụng bẫy lỗi để tránh đọc dữ liệu từ đối tượng đã bị xoá
    On Error GoTo ERRORHANDLER
    MsgBox "The area of " & pObject.ObjectName & " is: " & pObject.Area
    Exit Sub
ERRORHANDLER:
    MsgBox Err.Description
End Sub

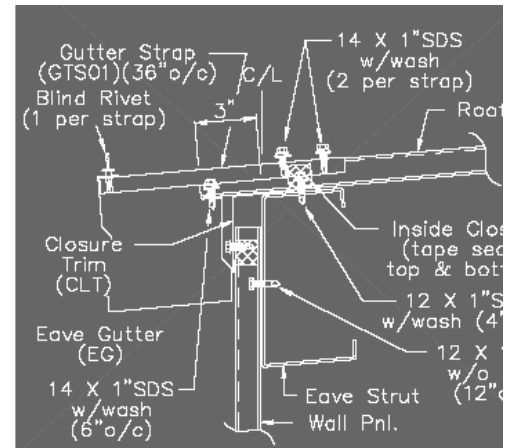
```

Tài liệu tham khảo  
 BM Tự động hoá TK

Tài liệu tham khảo  
BM Tự động hoá TKCĐ

# LÀM VIỆC TRONG KHÔNG GIAN BA CHIỀU

Vật thể 3 chiều (3D) thường được biểu diễn bởi các bản vẽ gồm các hình chiếu của nó trên các mặt phẳng (2D). Mặc dù phương pháp phác hoạ trên được sử dụng rất rộng rãi trong các ngành kỹ thuật và kiến trúc, nhưng nó bị hạn chế ở chỗ: bản vẽ 2D phải mô tả vật thể không gian và phải thể hiện được một cách trực quan. Hơn nữa, vì các hình chiếu được tạo ra độc lập nên khả năng lỗi và gây nhầm lẫn là rất lớn. Vì vậy, ta có thể tạo ra hình 3D thật thay vì thể hiện nó bằng các hình 2D. Có thể sử dụng các công cụ của AutoCAD để tạo ra các hình 3D một cách chi tiết và giống như thật, hơn nữa ta có thể thao tác với chúng theo nhiều cách khác nhau.



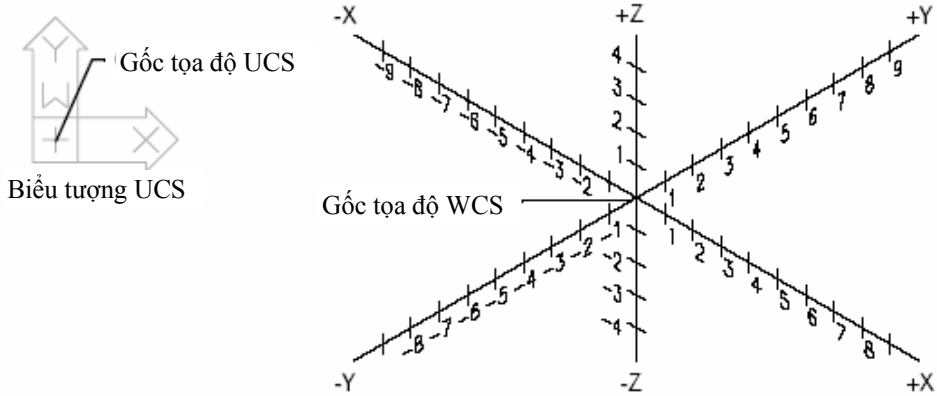
Trong chương này

8

- Xác định tọa độ ba chiều
- Định nghĩa hệ tọa độ người dùng
- Chuyển trục tọa độ
- Tạo đối tượng ba chiều
- Hiệu chỉnh trong không gian ba chiều
- Hiệu chỉnh vật thể khối

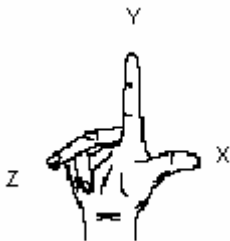
# 1. Xác định tọa độ ba chiều

Việc xác định tọa độ 3D cũng giống như trong hệ tọa độ 2D, chỉ thêm thành phần theo hướng thứ 3, trục Z. Khi vẽ trong 3D, cần xác định được các giá trị tọa độ theo các trục X, Y và Z trong hệ tọa độ chung (WCS) cũng như trong hệ tọa độ người dùng (UCS). Hình vẽ dưới đây mô tả các trục X, Y, Z của hệ trục WCS:



## 1.1. Quy tắc bàn tay phải

Quy tắc bàn tay phải xác định chiều dương của trục Z khi đã biết hướng của trục X và Y, đồng thời cũng xác định chiều quay dương xung quanh các trục trong không gian 3D.



Để xác định chiều dương của các trục X, Y, Z, đặt bàn tay phải thẳng, các ngón tay hướng thẳng lên trên sao cho mu bàn tay hướng về phía màn hình, ngón cái hướng sang ngang, ngón trỏ chỉ hướng thẳng đứng lên trên và ngón giữa gấp theo hướng vuông góc với lòng bàn tay, khi đó hướng chỉ của ngón cái sẽ trùng với hướng dương của trục X, hướng của các ngón trỏ trùng với hướng dương của trục Y, và hướng chỉ của ngón giữa là hướng dương của trục Z. Để xác định chiều quay dương quanh các trục tọa độ đặt ngón cái trùng theo hướng dương của hệ trục tọa độ, gấp các ngón tay còn lại hướng vào lòng bàn tay thì hướng gấp các ngón tay trùng với chiều quay dương quanh trục tọa độ đó.

## 1.2. Nhập tọa độ X, Y, Z

Nhập tọa độ trong hệ trục tọa độ WCS 3D tương tự như nhập tọa độ trong hệ trục tọa độ WCS 2D, nhưng cần chỉ ra tọa độ Z bên cạnh tọa độ X, Y. Cũng như hệ trục tọa độ 2D, một biến được sử dụng để chuyển các tọa độ vào trong các phương thức hay thuộc tính trong ActiveX cũng như để nhập vào một tọa độ.

Đoạn chương trình con dưới đây, trước hết sẽ tạo ra một đường đa tuyến nét mảnh 2D với 3 đỉnh, sau đó sẽ tạo đường đa tuyến 3D với 3 đỉnh. Chú ý rằng chiều dài của mảng chứa các đỉnh đó sẽ tăng lên để chứa thêm tọa độ Z của đường đa tuyến 3D. Chương trình này kết thúc bởi việc xuất ra tọa độ các đỉnh của đường đa tuyến 2D và 3D bằng thông báo.

### Định nghĩa và truy vấn tọa độ của đường đa tuyến 2D và 3D

Ví dụ này sẽ tạo ra 2 đường đa tuyến, trong đó có một đường 3D. Đường đầu tiên là đa tuyến 2D và đường thứ hai sẽ là 3D. Ví dụ này cũng sẽ truy vấn tọa độ của chúng và hiển thị trên các thông báo.

```
Sub Ch8_Polyline_2D_3D()
    Dim pline2DObj As AcadLWPolyline
    Dim pline3DObj As AcadPolyline
    Dim points2D(0 To 5) As Double
    Dim points3D(0 To 8) As Double

    ' Định nghĩa đỉnh polyline 2D
    points2D(0) = 1: points2D(1) = 1
    points2D(2) = 1: points2D(3) = 2
    points2D(4) = 2: points2D(5) = 2

    ' Định nghĩa đỉnh polyline 3D
    points3D(0) = 1: points3D(1) = 1: points3D(2) = 0
    points3D(3) = 2: points3D(4) = 1: points3D(5) = 0
    points3D(6) = 2: points3D(7) = 2: points3D(8) = 0

    ' Tạo polyline 2D
    Set pline2DObj =
    ThisDrawing.ModelSpace.AddLightWeightPolyline(points2D)
    pline2DObj.Color = acRed
    pline2DObj.Update

    ' Tạo polyline 3D
    Set pline3DObj = ThisDrawing.ModelSpace.AddPolyline(points3D)
    pline3DObj.Color = acBlue
    pline3DObj.Update

    ' Lấy các tọa độ đỉnh của các polyline
    Dim get2Dpts As Variant
    Dim get3Dpts As Variant
    get2Dpts = pline2DObj.Coordinates
    get3Dpts = pline3DObj.Coordinates

    ' Hiển thị tọa độ
    MsgBox ("2D polyline (red): " & vbCrLf & _
        get2Dpts(0) & ", " & get2Dpts(1) & vbCrLf & _
        get2Dpts(2) & ", " & get2Dpts(3) & vbCrLf & _
        get2Dpts(4) & ", " & get2Dpts(5))
    MsgBox ("3D polyline (blue): " & vbCrLf & _
        get3Dpts(0) & ", " & get3Dpts(1) & ", " & _
        get3Dpts(2) & vbCrLf & _
        get3Dpts(3) & ", " & get3Dpts(4) & ", " & _
        get3Dpts(5) & vbCrLf & _
        get3Dpts(6) & ", " & get3Dpts(7) & ", " & _
        get3Dpts(8))
End Sub
```

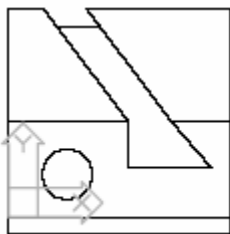
## 2. Định nghĩa hệ tọa độ người dùng

Định nghĩa một đối tượng hệ tọa độ người dùng (UCS) bao gồm thay đổi vị trí của gốc tọa độ (0,0,0) và thay đổi hướng của mặt phẳng XY và trục Z. Hệ tọa độ người dùng có thể đặt ở một vị trí và theo một hướng bất kỳ nào đó trong không gian 3D, có thể được định nghĩa, lưu lại và sử dụng với số lượng tùy vào nhu cầu sử dụng của người dùng. Nếu có nhiều cổng nhìn được kích hoạt thì chúng dùng chung một hệ trục tọa độ.

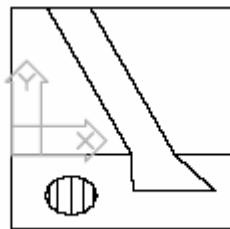
Nếu vẽ nhiều trong không gian 3D thì ta nên định nghĩa sẵn một số hệ trục tọa độ, mỗi hệ trục tọa độ có gốc và hướng các trục tọa độ khác nhau, tùy theo yêu cầu cụ thể.

Để chỉ ra gốc tọa độ và hướng của hệ trục tọa độ người dùng, ta cần hiển thị biểu tượng của UCS tại gốc của hệ tọa độ đó bằng cách sử dụng thuộc tính UCSIconAtOrigin (xem thuộc tính UCSIconOn) và để không hiển thị biểu tượng tại gốc tọa độ, nó được hiển thị tại tọa độ WCS được định nghĩa bởi biến hệ thống UCSORG.

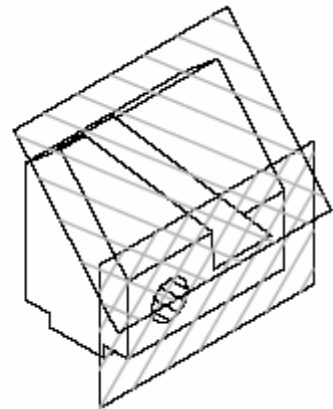
Các hệ trục tọa độ người dùng rất hữu ích trong không gian 3D. Ta sẽ dễ dàng sắp xếp các hệ tọa độ thẳng hàng với các khối hình học đã có hơn là phải tìm ra độ dịch chuyển chính xác của một điểm trong không gian 3D.



Hệ trục thứ nhất



Hệ trục thứ 2



Mô hình với 2 hệ trục tọa độ

Có thể định nghĩa hệ trục tọa độ mới trong không gian in cũng như trong không gian mô hình, tuy nhiên, các hệ trục người dùng trong không gian in chỉ được thao tác thủ công. AutoCAD lưu giữ được 10 hệ trục tọa độ sau cùng, mà được tạo ra trong không gian mô hình và trong không gian in.

Hệ trục tọa độ mới được tạo bằng phương thức Add. Phương thức này yêu cầu 4 giá trị đầu vào, bao gồm: tọa độ của điểm gốc ; tọa độ trên trục X, trục Y ; tên của hệ trục tọa độ.

Tất cả các tọa độ của các đối tượng ActiveX Automation của AutoCAD đều nhập vào từ hệ tọa độ chung (WCS). Phương thức GetUCSMatrix được dùng để tính ma trận biến đổi của một hệ trục UCS bất kỳ. Dùng ma trận này để tìm tọa độ tương đương trong WCS.

Sử dụng thuộc tính `ActiveUCS` trong đối tượng `Document` để kích hoạt một hệ tọa độ UCS. Nếu hệ trục UCS đang sử dụng bị thay đổi thì đối tượng UCS mới cần được khởi động lại như hệ tọa độ UCS hiện hành để những thay đổi được xuất hiện. Hệ trục UCS hiện hành được thiết lập lại bằng cách sử dụng một lần nữa thuộc tính `ActiveUCS` với đối tượng UCS được cập nhật.

### Ví dụ tạo một hệ tọa độ UCS mới, kích hoạt và chuyển tọa độ của một điểm sang hệ tọa độ UCS này.

Thủ tục dưới đây sẽ tạo một hệ trục tọa độ mới và kích hoạt. Sau đó yêu cầu người dùng chọn một điểm trong bản vẽ và trả về tọa độ của điểm đó trong cả hệ trục UCS và hệ trục WCS.

```
Sub Ch8_NewUCS()  
    ' Định nghĩa các biến cần dùng  
    Dim ucsObj As AcadUCS  
    Dim origin(0 To 2) As Double  
    Dim xAxisPnt(0 To 2) As Double  
    Dim yAxisPnt(0 To 2) As Double  
  
    ' Gán gốc tọa độ cho hệ trục UCS  
    origin(0) = 4: origin(1) = 5: origin(2) = 3  
    xAxisPnt(0) = 5: xAxisPnt(1) = 5: xAxisPnt(2) = 3  
    yAxisPnt(0) = 4: yAxisPnt(1) = 6: yAxisPnt(2) = 3  
  
    ' Thêm hệ trục UCS vào tập đối tượng UserCoordinateSystems  
    Set ucsObj = ThisDrawing.UserCoordinateSystems._  
        Add(origin, xAxisPnt, yAxisPnt, "New_UCS")  
  
    ' Thể hiện biểu tượng UCS  
    ThisDrawing.ActiveViewport.UCSIconAtOrigin = True  
    ThisDrawing.ActiveViewport.UCSIconOn = True  
  
    ' Chuyển hệ trục UCS mới thành hệ trục hiện thời  
    ThisDrawing.ActiveUCS = ucsObj  
    MsgBox "The current UCS is : " & ThisDrawing.ActiveUCS.Name _  
        & vbCrLf & " Pick a point in the drawing."  
  
    ' Tìm tọa độ trong hệ trục WCS và UCS của một điểm  
    Dim WCSpnt As Variant  
    Dim UCSpnt As Variant  
    WCSpnt = ThisDrawing.Utility.GetPoint(, "Enter a point: ")  
    UCSpnt = ThisDrawing.Utility.TranslateCoordinates _  
        (WCSpnt, acWorld, acUCS, False)  
    MsgBox "The WCS coordinates are: " & WCSpnt(0) & ", " _  
        & WCSpnt(1) & ", " & WCSpnt(2) & vbCrLf & _  
        "The UCS coordinates are: " & UCSpnt(0) & ", " _  
        & UCSpnt(1) & ", " & UCSpnt(2)  
  
End Sub
```

## 3. Chuyển trục tọa độ

Phương thức `TranslateCoordinates` sẽ chuyển một điểm hoặc một vectơ từ hệ tọa độ này sang hệ tọa độ khác. Một đối số kiểu `point`, gọi là `OriginalPoint` (điểm gốc), có thể được hiểu như là một điểm 3D hoặc là một vectơ chuyển vị 3D. Đối số này có thể được phân biệt bởi đối số kiểu `Boolean` và `Disp`. Nếu đối số `Disp` được gán



bằng True thì đối số `OriginalPoint` được xem như là véc tơ chuyển vị, ngược lại thì được coi như là một điểm. Hai đối số nữa nhằm xác định xem `OriginalPoint` thuộc hệ trục tọa độ nào và nó sẽ được chuyển sang hệ trục tọa độ nào. Các hệ trục tọa độ dưới đây của AutoCAD có thể được xác định từ các đối số `From` và `To` :

- **WCS**

World Coordinate System (WCS) - hệ trục tọa độ tham chiếu. Tất cả các hệ trục tọa độ khác đều được định nghĩa thông qua hệ trục này vì đây là hệ trục không thay đổi. Tất cả các giá trị được tính thông qua WCS là ổn định dù có sự thay đổi các hệ trục tọa độ khác. Tất cả các điểm sử dụng trong các phương thức và thuộc tính của ActiveX đều được biểu diễn trong hệ trục WCS ngoại trừ khi có những trường hợp khác được chỉ rõ.

- **UCS**

User Coordinate System (UCS) - hệ trục tọa độ làm việc. Người dùng tạo ra hệ trục UCS để tạo thuận lợi cho các thao tác với bản vẽ. Tất cả các điểm được chuyển vào dòng lệnh của AutoCAD, bao gồm cả những điểm trả về từ các hàm AutoLISP và các hàm mở rộng, là những điểm trong hệ trục UCS hiện tại (trừ khi người dùng thêm vào phía trước dấu \* ở đầu nhắc dòng lệnh). Nếu muốn ứng dụng gửi tọa độ điểm theo hệ trục WCS, OCS, DCS đến dòng lệnh của AutoCAD, trước hết phải chuyển tọa độ về hệ trục tọa độ UCS bằng cách gọi phương thức `TranslateCoordinates`.

- **OCS**

Object Coordinate System (OCS) - giá trị tọa độ được xác định trong các phương thức và thuộc tính cho đối tượng `Polyline` và `LightweightPolyline`, đều được biểu diễn trong hệ trục tọa độ này, tương đối so với đối tượng. Những điểm này thường được chuyển sang hệ trục tọa độ WCS, hệ trục tọa độ UCS hiện tại hoặc hệ trục tọa độ DCS hiện tại, tùy theo mục đích sử dụng của đối tượng. Ngược lại, các điểm trong hệ trục tọa độ WCS, UCS hay DCS phải được chuyển về hệ trục OCS trước khi ghi vào cơ sở dữ liệu bằng chính các thuộc tính đó. Xem thêm tài liệu "*ActiveX and VBA Reference*" để tìm hiểu thêm về các phương thức và thuộc tính được sử dụng trong hệ trục tọa độ này.

Khi chuyển tọa độ sang hệ OCS hoặc ngược lại, cần đưa vào véc tơ pháp tuyến<sup>1</sup> cho hệ trục OCS trong đối số cuối cùng của hàm `TranslateCoordinates`.

- **DCS**

Display Coordinate System (DCS) - hệ trục tọa độ mà các đối tượng sẽ được biến đổi trước khi chúng được hiển thị. Gốc của DCS là điểm được lưu trong biến hệ thống `TARGET` của AutoCAD và trục Z của hệ trục này là hướng quan sát. Nói cách khác, khung nhìn luôn là mặt phẳng quan sát của

---

<sup>1</sup> Véc tơ pháp tuyến (Normal vector): véc tơ pháp tuyến đơn vị trong hệ trục tọa độ WCS, dùng để định nghĩa trục tọa độ z cho hệ trục tọa độ OCS

hệ trục DCS. Các tọa độ này có thể được sử dụng để xác định xem những đối tượng nào sẽ được hiển thị trong AutoCAD.

- PSDCS

Paper Space DCS (PSDCS) - hệ trục tọa độ này chỉ có thể được biến đổi từ hệ trục DCS hoặc sang hệ trục DCS trong khung nhìn của không gian mô hình hiện hành. Thực chất đây là sự biến đổi 2D, trong đó tọa độ X và Y luôn được nhân tỷ lệ và tịnh tiến nếu đối số `Disp` là `True`, còn tọa độ Z chỉ được nhân tỷ lệ nhưng không bao giờ tịnh tiến. Do đó, tọa độ Z có thể được dùng để tìm hệ số phóng đại giữa hai hệ trục tọa độ. Hệ trục PSDCS chỉ có thể được biến đổi trong khung nhìn của không gian mô hình hiện tại. Nếu đối số `from` là PSDCS thì đối số `to` sẽ là DCS và ngược lại.

### Biến đổi tọa độ từ hệ trục OCS sang hệ trục WCS

Ví dụ sau đây sẽ tạo ra đối tượng Polyline trong không gian mô hình. Đỉnh thứ nhất của đa tuyến sẽ được biểu diễn theo tọa độ trong cả hệ trục OCS và WCS. Quá trình biến đổi ngược từ OCS sang WCS cần phải nhập véc tơ pháp tuyến của hệ trục OCS vào đối số cuối cùng trong phương thức `TranslateCoordinates`.

```
Sub Ch8_TranslateCoordinates()  
    ' Tạo một polyline trong không gian mô hình.  
    Dim plineObj As AcadPolyline  
    Dim points(0 To 14) As Double  
    ' Gán các đỉnh cho polyline  
    points(0) = 1: points(1) = 1: points(2) = 0  
    points(3) = 1: points(4) = 2: points(5) = 0  
    points(6) = 2: points(7) = 2: points(8) = 0  
    points(9) = 3: points(10) = 2: points(11) = 0  
    points(12) = 4: points(13) = 4: points(14) = 0  
  
    ' Tạo đối tượng light weight Polyline trong không gian mô hình  
    Set plineObj = ThisDrawing.ModelSpace.AddPolyline(points)  
    ' Tìm tọa độ X và Y của đỉnh đầu tiên của đường polyline  
    Dim firstVertex As Variant  
    firstVertex = plineObj.Coordinate(0)  
    ' Tìm tọa độ Z của đường polyline sử dụng thuộc tính  
    Elevation  
    firstVertex(2) = plineObj.Elevation  
  
    ' Thay đổi véc tơ pháp cho polyline để thấy sự sai khác  
    ' giữa hai hệ tọa độ  
    Dim plineNormal(0 To 2) As Double  
    plineNormal(0) = 0#  
    plineNormal(1) = 1#  
    plineNormal(2) = 2#  
    plineObj.Normal = plineNormal  
  
    ' Chuyển tọa độ từ hệ trục OCS sang WCS  
    Dim coordinateWCS As Variant  
    coordinateWCS = ThisDrawing.Utility.TranslateCoordinates _  
        (firstVertex, acOCS, acWorld, False, plineNormal)  
  
    ' Biểu diễn tọa độ của điểm  
    MsgBox "The first vertex has the following coordinates:" _  
        & vbCrLf & "OCS: " & firstVertex(0) & ", " & _
```

```

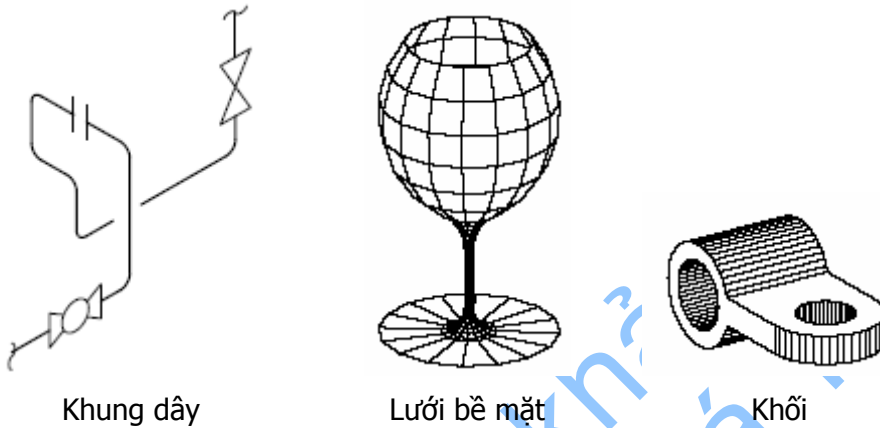
firstVertex(1) & ", " & firstVertex(2) & vbCrLf & _
"WCS: " & coordinateWCS(0) & ", " & _
coordinateWCS(1) & ", " & coordinateWCS(2)

```

End Sub

## 4. Tạo đối tượng ba chiều

AutoCAD hỗ trợ ba kiểu mô hình không gian: mô hình khung dây, lưới và khối. Mỗi loại có cách tạo và hiệu chỉnh riêng.



Mô hình khung dây: là dạng mô tả các đối tượng không gian dưới dạng khung. Trong mô hình này, các đối tượng không có các mặt mà chỉ gồm các điểm, các đường thẳng và các đường cong thể hiện các cạnh của vật thể. Có thể tạo mô hình khung dây bằng cách bố trí các đối tượng 2D ở bất cứ đâu trong không gian 3D. AutoCAD cũng cung cấp một số đối tượng khung dây 3D như đường đa tuyến 3D. Vì mỗi đối tượng được tạo thành bởi mô hình khung dây cần được vẽ và bố trí một cách độc lập nên loại mô hình này đôi khi rất tốn thời gian để mô tả.

Mô hình lưới bề mặt: phức tạp hơn mô hình khung dây ở chỗ nó phải định nghĩa không chỉ các cạnh, mà cả bề mặt của đối tượng. Mô hình hoá bề mặt trong AutoCAD xác định các mặt bằng cách sử dụng một lưới đa giác. Do các mặt của lưới là phẳng nên lưới chỉ có thể xấp xỉ được các đường cong của bề mặt.

Mô hình khối: là loại mô hình 3D sử dụng dễ nhất. Với mô hình khối trong AutoCAD, ta có thể tạo ra các hình 3D cơ bản như: hộp, hình nón, trụ tròn, hình cầu, hình nêm... Sau đó, để tạo ra các hình khối khác phức tạp hơn, ta có thể kết hợp, cắt bỏ, hoặc xác định giao cắt của các khối. Ta cũng có thể tạo ra khối bằng cách quét một hình phẳng quanh một trục nào đó hoặc quét theo một đường nào đó. Với AutoCAD Designer, ta có thể định nghĩa các khối theo các thông số và mối liên hệ giữa mô hình 3D và hình chiếu 2D được tạo ra từ khối 3D đó.

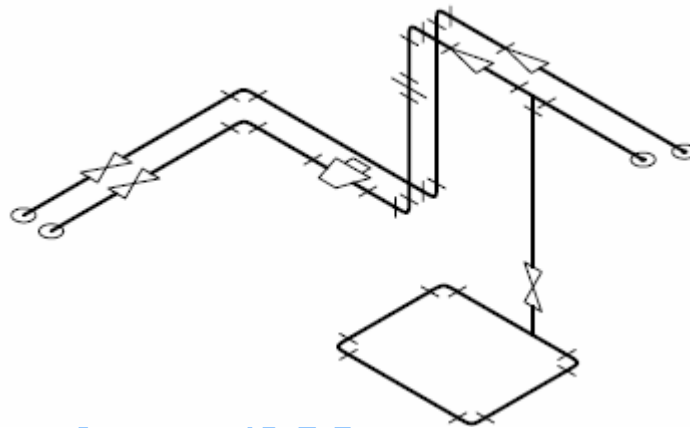
**CẢNH BÁO!** Mỗi loại mô hình sử dụng phương pháp khác nhau để xây dựng mô hình 3D, phương thức hiệu chỉnh đối tượng cũng khác nhau trong cách tác động đến các loại mô hình khác nhau, do vậy không nên sử dụng lẫn lộn các phương thức mô hình hóa. Ta có thể biến đổi từ dạng khối sang dạng mặt và từ dạng mặt sang dạng khung dây. Tuy nhiên, ta không thể biến đổi từ dạng khung dây sang dạng mặt và từ dạng mặt sang dạng khối.

## 4.1. Tạo khung dây

Trong AutoCAD, có thể tạo mô hình khung dây bằng cách bố trí các đối tượng 2D ở bất cứ vị trí nào trong không gian 3D. Có thể bố trí theo một số phương pháp sau:

- Tạo đối tượng bằng cách nhập điểm 3D, bao gồm các tọa độ X, Y và Z để định vị điểm.
- Đặt mặt phẳng vẽ mặc định (mặt phẳng XY) cho mặt phẳng dự định vẽ đối tượng bằng cách định nghĩa một hệ trục tọa độ người dùng (UCS).
- Di chuyển đối tượng đã được tạo ra từ trước theo hướng thích hợp trong không gian 3D.

Ngoài ra có thể tạo một số đối tượng khung dây như đường đa tuyến trong không gian 3D. Sử dụng phương thức `Add3DPoly` để tạo đối tượng 3DPolyline. Hình vẽ dưới mô tả một ví dụ ứng dụng mô hình hoá 3D bằng cách kết hợp các đa tuyến 3D và các biểu tượng 2D được đặt trong không gian 3D.

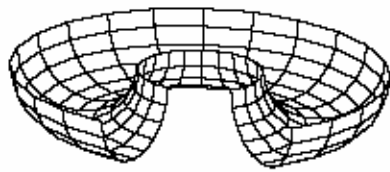


## 4.2. Tạo lưới bề mặt

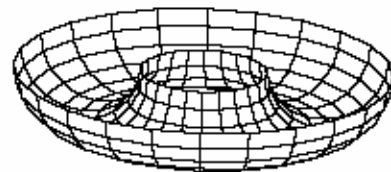
Một lưới đa giác (đối tượng PolygonMesh) thể hiện bề mặt của một vật thể bằng các mặt phẳng. Mật độ lưới hay số mặt được xác định bởi ma trận có  $M \times N$  đỉnh, tương tự như một lưới bao gồm các cột và các hàng.  $M$  và  $N$  theo thứ tự là chỉ số cột và chỉ số hàng của một đỉnh bất kỳ của lưới. Lưới có thể tạo cả trong không gian 2D và 3D nhưng chủ yếu được sử dụng trong 3D.

Sử dụng mô hình lưới nếu như không cần chi tiết về một số thuộc tính vật lý (như khối lượng, trọng lượng, trọng tâm của vật thể...) nhưng lại cần khả năng che khuất, tô bóng và tạo vỏ ngoài, là những tính chất mà mô hình khung dây không thể hiện được. Ngoài ra, mô hình lưới bề mặt cũng rất hữu ích để tạo ra các hình không thông thường, ví dụ như mô hình 3D của vùng núi.

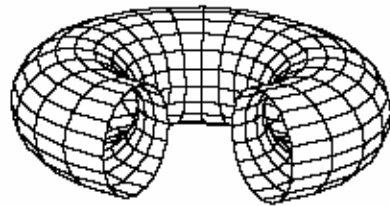
Một ô lưới có thể khép kín hoặc không khép kín (mở). Ô lưới là mở theo một hướng nào đó nếu cạnh đầu và cạnh cuối của lưới theo hướng đó không trùng nhau như mô tả ở hình vẽ dưới đây:



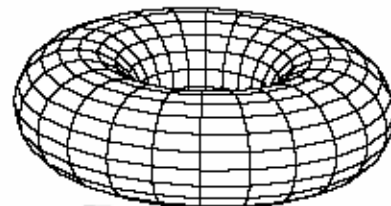
M mở  
N mở



M đóng  
N mở



M mở  
N đóng



M đóng  
N đóng

Để tạo lưới chữ nhật sử dụng phương thức `Add3Dmesh`. Phương thức này cần 3 tham số đầu vào: số đỉnh theo hướng M (số cột), số đỉnh theo hướng N (số hàng) và một mảng kiểu `variant` để lưu tọa độ các đỉnh của lưới. Khi một đối tượng `PolygonMesh` được tạo ra, sử dụng thuộc tính `Mclose` và `Nclose` để khép kín lưới.

### Ví dụ tạo lưới đa giác

Ví dụ này sẽ tạo một lưới đa giác kích thước 4x4. Hướng của khung nhìn hiện tại sẽ được điều chỉnh để có thể nhìn được lưới trong không gian 3D dễ dàng hơn.

```
Sub Ch8_Create3DMesh ()
    Dim meshObj As AcadPolygonMesh
    Dim mSize, nSize, Count As Integer
    Dim points(0 To 47) As Double
    ' tạo mảng chứa các điểm
    points(0) = 0: points(1) = 0: points(2) = 0
    points(3) = 2: points(4) = 0: points(5) = 1
    points(6) = 4: points(7) = 0: points(8) = 0
    points(9) = 6: points(10) = 0: points(11) = 1
    points(12) = 0: points(13) = 2: points(14) = 0
    points(15) = 2: points(16) = 2: points(17) = 1
    points(18) = 4: points(19) = 2: points(20) = 0
    points(21) = 6: points(22) = 2: points(23) = 1
    points(24) = 0: points(25) = 4: points(26) = 0
    points(27) = 2: points(28) = 4: points(29) = 1
    points(30) = 4: points(31) = 4: points(32) = 0
    points(33) = 6: points(34) = 4: points(35) = 0
    points(36) = 0: points(37) = 6: points(38) = 0
    points(39) = 2: points(40) = 6: points(41) = 1
    points(42) = 4: points(43) = 6: points(44) = 0
    points(45) = 6: points(46) = 6: points(47) = 0
End Sub
```

```

mSize = 4: nSize = 4
'Tạo lưới không gian 3DMesh trong không gian mô hình
Set meshObj = ThisDrawing.ModelSpace.Add3Dmesh_
    (mSize, nSize, points)
'Thay đổi hướng của công nhìn để nhìn rõ hơn hình trụ
Dim NewDirection(0 To 2) As Double
NewDirection(0) = -1
NewDirection(1) = -1
NewDirection(2) = 1
ThisDrawing.ActiveViewport.direction = NewDirection
ThisDrawing.ActiveViewport = ThisDrawing.ActiveViewport
ZoomAll
End Sub

```

### 4.3. Tạo lưới đa diện

Sử dụng phương thức `AddPolyfaceMesh` để tạo lưới đa diện với mỗi mặt có thể có số đỉnh khác nhau. Tạo lưới đa diện tương tự như tạo lưới chữ nhật, cần chỉ ra tọa độ của tất cả các đỉnh và sau đó định nghĩa các mặt bằng cách nhập chỉ số của các điểm cho các đỉnh của mặt đó. Khi tạo ra một lưới đa diện, có thể chỉ ra các cạnh khuất, gán chúng vào các lớp riêng hoặc đổi màu chúng.

Muốn tạo một cạnh khuất thì nhập chỉ số đỉnh của cạnh đó là số âm. Có thể tham khảo thêm trong tài liệu “*ActiveX and VBA Reference*”, phần phương thức `AddPolyfaceMesh`.

#### Tạo lưới đa diện

Ví dụ này tạo ra một đối tượng `PolyfaceMesh` trong không gian mô hình. Hướng của công nhìn sử dụng sẽ được điều chỉnh để có thể nhìn được lưới trong không gian 3D dễ dàng hơn.

```

Sub Ch8_CreatePolyfaceMesh()
    'Định nghĩa các đỉnh của lưới
    Dim vertex(0 To 17) As Double
    vertex(0) = 4: vertex(1) = 7: vertex(2) = 0
    vertex(3) = 5: vertex(4) = 7: vertex(5) = 0
    vertex(6) = 6: vertex(7) = 7: vertex(8) = 0
    vertex(9) = 4: vertex(10) = 6: vertex(11) = 0
    vertex(12) = 5: vertex(13) = 6: vertex(14) = 0
    vertex(15) = 6: vertex(16) = 6: vertex(17) = 1
    ' Định nghĩa danh sách các mặt
    Dim FaceList(0 To 7) As Integer
    FaceList(0) = 1
    FaceList(1) = 2
    FaceList(2) = 5
    FaceList(3) = 4
    FaceList(4) = 2
    FaceList(5) = 3
    FaceList(6) = 6
    FaceList(7) = 5
    ' Tạo lưới đa diện
    Dim polyfaceMeshObj As AcadPolyfaceMesh
    Set polyfaceMeshObj = ModelSpace.AddPolyfaceMesh_
        (vertex, FaceList)
    ' Thay đổi hướng nhìn của công nhìn để nhìn lưới đa diện.
    Dim NewDirection(0 To 2) As Double

```

```

NewDirection(0) = -1
NewDirection(1) = -1
NewDirection(2) = 1
ThisDrawing.ActiveViewport.direction = NewDirection
ThisDrawing.ActiveViewport = ThisDrawing.ActiveViewport
ZoomAll
End Sub

```

## 4.4. Tạo khối

Một đối tượng khối (đối tượng 3DSolid) thể hiện được toàn bộ hình khối của vật thể. Mô hình khối thể hiện được đầy đủ nhất thông tin của vật thể và là loại mô hình ít gây nhầm lẫn nhất. Các hình khối phức tạp được tạo ra và hiệu chỉnh dễ dàng hơn các mô hình kiểu lưới và kiểu khung dây.

Tạo các hình khối từ một trong các hình khối như: hộp, nón, trụ, cầu, hình nêm... hoặc bằng cách đập nổi từ một đối tượng 2D theo một đường sinh hoặc quay hình 2D quanh một trục. Dưới đây là các phương thức để tạo khối: AddBox, AddCone, AddCylinder, AddEllipticalCone, AddEllipticalCylinder, AddExtrudedSolid, AddExtrudedSolidAlongPath, AddRevolvedSolid, AddSolid, AddSphere, AddTorus, AddWedge.

### Tạo khối hình nêm

Ví dụ này sẽ tạo ra khối đặc hình nêm trong không gian mô hình. Hướng của cổng nhìn sử dụng sẽ được điều chỉnh để có thể nhìn được khối trong không gian 3D dễ dàng hơn.

```

Sub Ch8_CreateWedge ()
    Dim wedgeObj As Acad3DSolid
    Dim center(0 To 2) As Double
    Dim length As Double
    Dim width As Double
    Dim height As Double
    ' Định nghĩa khối nêm
    center(0) = 5#: center(1) = 5#: center(2) = 0
    length = 10#: width = 15#: height = 20#
    ' Tạo khối nêm trong không gian mô hình
    Set wedgeObj = ThisDrawing.ModelSpace.
    AddWedge(center, length, width, height)
    ' Đổi hướng nhìn của cổng nhìn
    Dim NewDirection(0 To 2) As Double
    NewDirection(0) = -1
    NewDirection(1) = -1
    NewDirection(2) = 1
    ThisDrawing.ActiveViewport.direction = NewDirection
    ThisDrawing.ActiveViewport = ThisDrawing.ActiveViewport
    ZoomAll
End Sub

```

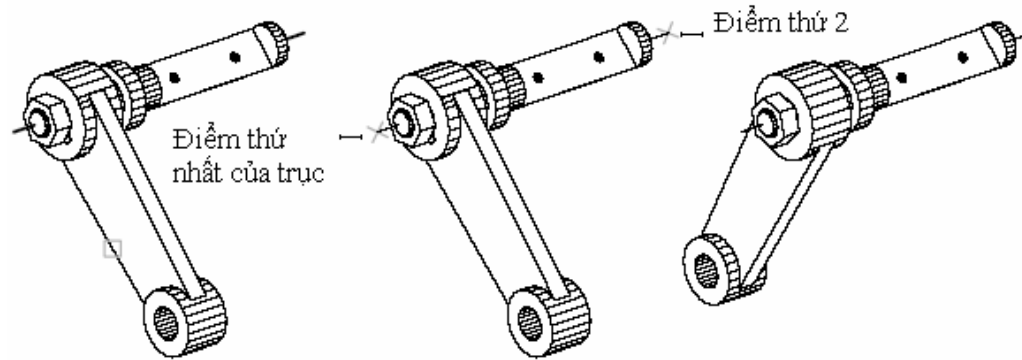
## 5. Hiệu chỉnh trong không gian 3D

Mục này sẽ trình bày về sự hiệu chỉnh các đối tượng 3D như : quay, nhân bản và lấy đối xứng.



## 5.1. Quay

Phương thức `Rotate` sẽ quay đối tượng 2D quanh một điểm và chiều quay được xác định theo hệ tọa độ WCS. Sử dụng phương thức `Rotate3D` để quay đối tượng 3D quanh một trục xác định, phương thức này cần 3 tham số đầu vào gồm: tọa độ trong hệ WCS của 2 điểm xác định trục quay và góc quay tính theo radian.



Đối tượng quay

Trục quay

Sau khi quay

Để quay đối tượng 3D có thể sử dụng một trong 2 phương thức `Rotate` hoặc `Rotate3D`. Dưới đây là một thủ tục tạo một khối hộp và quay nó quanh một trục.

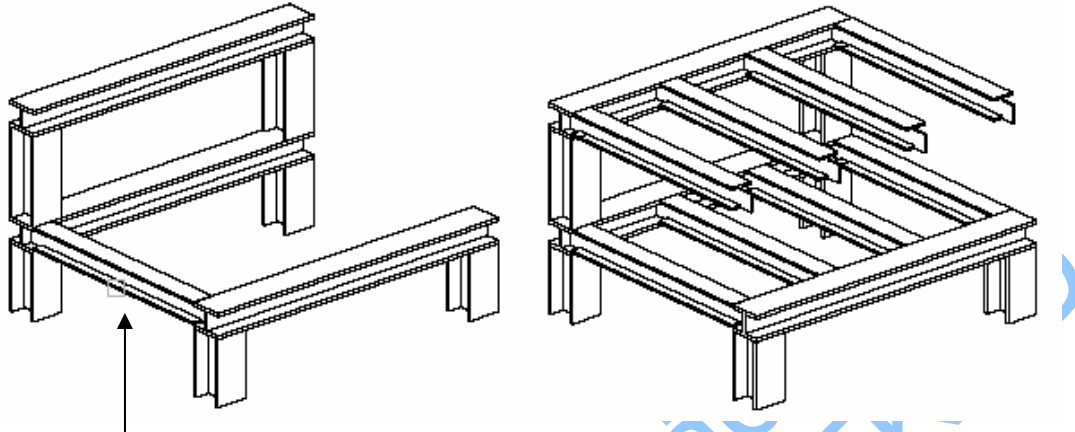
### Tạo khối hộp và quay nó quanh một trục

Ví dụ sau sẽ tạo một khối hộp, sau đó sẽ định nghĩa trục quay và quay khối hộp quanh trục đó với góc quay  $30^{\circ}$ .

```
Sub Ch8_Rotate_3DBox()  
    Dim boxObj As Acad3DSolid  
    Dim length As Double  
    Dim width As Double  
    Dim height As Double  
    Dim center(0 To 2) As Double  
    ' Định nghĩa khối hộp  
    center(0) = 5: center(1) = 5: center(2) = 0  
    length = 5  
    width = 7  
    height = 10  
    ' Tạo khối hộp trong không gian mô hình  
    Set boxObj = ThisDrawing.ModelSpace.  
        AddBox(center, length, width, height)  
    ' Định nghĩa trục quay thông qua 2 điểm  
    Dim rotatePt1(0 To 2) As Double  
    Dim rotatePt2(0 To 2) As Double  
    Dim rotateAngle As Double  
    rotatePt1(0) = -3: rotatePt1(1) = 4: rotatePt1(2) = 0  
    rotatePt2(0) = -3: rotatePt2(1) = -4: rotatePt2(2) = 0  
    rotateAngle = 30  
    rotateAngle = rotateAngle * 3.141592 / 180#  
    ' Quay khối hộp  
    boxObj.Rotate3D rotatePt1, rotatePt2, rotateAngle  
    ZoomAll  
End Sub
```

## 5.2. Nhân bản<sup>1</sup>

Phương thức `ArrayRectangular` sẽ thực hiện nhân bản đối tượng theo dạng chữ nhật. Ngoài việc xác định số hàng và số cột theo các hướng tương ứng là trục X và trục Y, cần xác định số tầng tức là số phần tử theo hướng trục Z.



Đối tượng nhân bản

Kết quả

### Nhân bản đối tượng 3D

Ví dụ dưới đây sẽ tạo ra một đường tròn và sau đó nhân bản dạng chữ nhật gồm (4hàng × 4cột × 3tầng)

```
Sub Ch8_CreateRectangularArray()  
    ' Tạo đường tròn  
    Dim circleObj As AcadCircle  
    Dim center(0 To 2) As Double  
    Dim radius As Double  
    center(0) = 2: center(1) = 2: center(2) = 0  
    radius = 0.5  
    Set circleObj = ThisDrawing.ModelSpace. _  
        AddCircle(center, radius)  
    ' Định nghĩa kích thước dãy hình chữ nhật  
    Dim numberOfRows As Long  
    Dim numberOfColumns As Long  
    Dim numberOfLevels As Long  
    Dim distanceBwtnRows As Double  
    Dim distanceBwtnColumns As Double  
    Dim distanceBwtnLevels As Double  
    numberOfRows = 4  
    numberOfColumns = 4  
    numberOfLevels = 3  
    distanceBwtnRows = 1  
    distanceBwtnColumns = 1  
    distanceBwtnLevels = 4  
    ' Nhân bản đối tượng  
    Dim retObj As Variant
```

<sup>1</sup> **Nhân bản (Array)**: là quá trình tạo ra một (hoặc nhiều) đối tượng mới giống hệt như đối tượng ban đầu theo một quy luật bố trí dạng chữ nhật hoặc theo dạng tọa độ cực (và có thể xoay hình mới được tạo đi một góc nào đó).

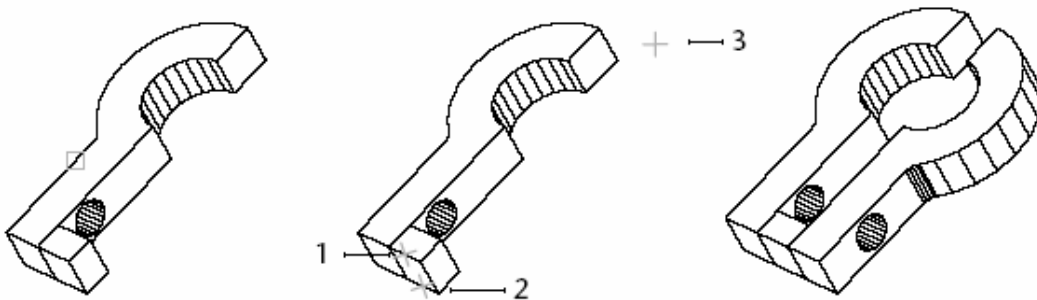
```

retObj = circleObj.ArrayRectangular _
    (numberOfRows, numberOfColumns, _
    numberOfLevels, distanceBwtnRows, _
    distanceBwtnColumns, distanceBwtnLevels)
ZoomAll
End Sub

```

### 5.3. Lấy đối xứng

Phương thức `Mirror3D` sẽ lấy đối xứng các đối tượng qua một mặt phẳng đối xứng được xác định bởi 3 điểm.



Đối tượng lấy đối xứng      Định nghĩa mặt phẳng đối xứng      Sau khi lấy đối xứng

#### Lấy đối xứng trong không gian 3D

Ví dụ này sẽ tạo một khối hộp trong không gian mô hình và sau đó sẽ lấy đối xứng khối hộp đó qua một mặt phẳng và khối hộp đối xứng sẽ có màu đỏ.

```

Sub Ch8_MirrorABox3D()
    ' Tạo đối tượng khối
    Dim boxObj As Acad3DSolid
    Dim length As Double
    Dim width As Double
    Dim height As Double
    Dim center(0 To 2) As Double
    center(0) = 5#: center(1) = 5#: center(2) = 0
    length = 5#: width = 7: height = 10#
    ' Tạo khối hộp (đối tượng 3DSolid) trong không gian mô hình
    Set boxObj = ThisDrawing.ModelSpace. _
        AddBox(center, length, width, height)
    ' Định nghĩa mặt phẳng đối xứng bởi 3 điểm
    Dim mirrorPt1(0 To 2) As Double
    Dim mirrorPt2(0 To 2) As Double
    Dim mirrorPt3(0 To 2) As Double
    mirrorPt1(0) = 1.25: mirrorPt1(1) = 0: mirrorPt1(2) = 0
    mirrorPt2(0) = 1.25: mirrorPt2(1) = 2: mirrorPt2(2) = 0
    mirrorPt3(0) = 1.25: mirrorPt3(1) = 2: mirrorPt3(2) = 2
    ' Lấy đối xứng khối hộp
    Dim mirrorBoxObj As Acad3DSolid
    Set mirrorBoxObj = boxObj.Mirror3D _
        (mirrorPt1, mirrorPt2, mirrorPt3)
    mirrorBoxObj.Color = acRed
    ZoomAll
End Sub

```

## 6. Hiệu chỉnh vật thể khối

Có thể tạo ra các hình khối phức tạp từ các vật thể khác bằng cách kết hợp các khối, cắt khối, giao nhau của các khối. Sử dụng phương thức Boolean hoặc CheckInterference để thực hiện các phép hiệu chỉnh nói trên.



Các khối trước khi dùng Boolean giao cắt      Sau khi thực hiện Boolean giao cắt

Đoạn chương trình dưới đây sẽ tạo một khối hộp và một khối trụ trong không gian mô hình. Sau đó sẽ tìm giao của hai khối và tạo khối mới từ phần giao của hai khối trên. Để quan sát được tốt hơn, khối hộp sẽ có màu trắng và khối trụ sẽ có màu xanh và khối giao sẽ có màu đỏ.

### Tìm phần giao của 2 khối

Ví dụ này sẽ tạo một khối hộp và một khối trụ trong không gian mô hình. Sau đó sẽ tìm giao của hai khối và tạo khối mới từ phần giao của hai khối trên. Để quan sát được tốt hơn, khối hộp sẽ có màu trắng và khối trụ sẽ có màu xanh và khối giao sẽ có màu đỏ.

```
Sub Ch8_FindInterferenceBetweenSolids ()
    ' Định nghĩa khối hộp
    Dim boxObj As Acad3DSolid
    Dim length As Double
    Dim width As Double
    Dim height As Double
    Dim center(0 To 2) As Double
    center(0) = 5: center(1) = 5: center(2) = 0
    length = 5
    width = 7
    height = 10
    ' Tạo khối hộp trong không gian mô hình có màu trắng
    Set boxObj = ThisDrawing.ModelSpace. _
        AddBox(center, length, width, height)
    boxObj.Color = acWhite
    ' Định nghĩa khối trụ
    Dim cylinderObj As Acad3DSolid
    Dim cylinderRadius As Double
    Dim cylinderHeight As Double
    center(0) = 0: center(1) = 0: center(2) = 0
    cylinderRadius = 5
    cylinderHeight = 20
    ' Tạo khối trụ có màu xanh
    Set cylinderObj = ThisDrawing.ModelSpace.AddCylinder _
        (center, cylinderRadius, cylinderHeight)
    cylinderObj.Color = acCyan
    ' Tìm phần giao của hai khối và tạo ra khối mới có màu đỏ.
    Dim solidObj As Acad3DSolid
```

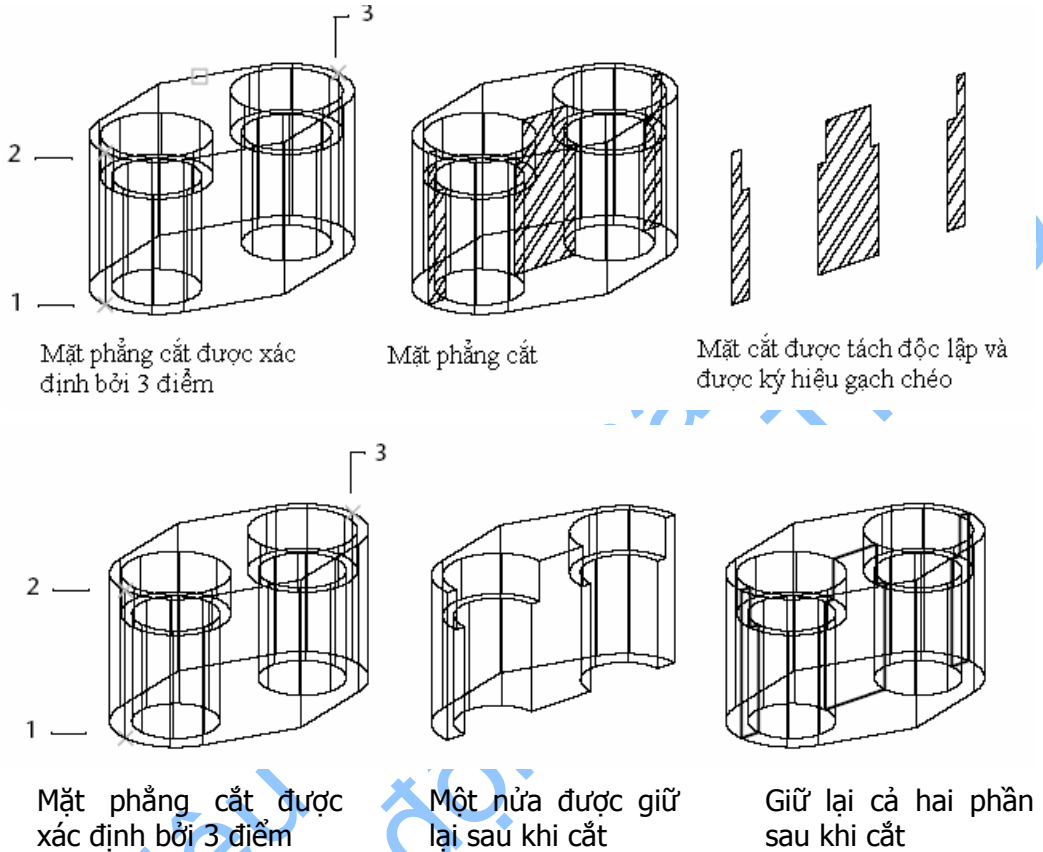
```

Set solidObj = boxObj.CheckInterference(cylinderObj, True)
solidObj.Color = acRed
ZoomAll

```

End Sub

Các khối còn có thể được hiệu chỉnh theo các cách khác như: lấy mặt cắt của các khối bởi các mặt phẳng sử dụng phương thức `SectionSolid` hoặc cắt một khối thành hai phần sử dụng phương thức `SliceSolid`.



### Cắt khối thành hai phần

Ví dụ sau sẽ tạo một khối hộp trong không gian mô hình. Sau đó sẽ cắt khối hộp bằng một mặt phẳng xác định bởi 3 điểm và khối cắt được trả về kiểu đối tượng `3DSolid`.

```

Sub Ch8_SliceABox()
' Định nghĩa đối tượng khối hộp
Dim boxObj As Acad3DSolid
Dim length As Double
Dim width As Double
Dim height As Double
Dim center(0 To 2) As Double
center(0) = 5#: center(1) = 5#: center(2) = 0
length = 5#: width = 7: height = 10#
' Tạo khối hộp (đối tượng 3DSolid) trong không gian mô hình
Set boxObj = ThisDrawing.ModelSpace.AddBox _
(center, length, width, height)
boxObj.Color = acWhite
' Định nghĩa mặt phẳng cắt thông qua 3 điểm

```

```

Dim slicePt1(0 To 2) As Double
Dim slicePt2(0 To 2) As Double
Dim slicePt3(0 To 2) As Double
slicePt1(0) = 1.5: slicePt1(1) = 7.5: slicePt1(2) = 0
slicePt2(0) = 1.5: slicePt2(1) = 7.5: slicePt2(2) = 10
slicePt3(0) = 8.5: slicePt3(1) = 2.5: slicePt3(2) = 10
' Cắt khối hộp thành các phần và chuyển màu
' các phần đó thành đồ
Dim sliceObj As Acad3DSolid
Set sliceObj = boxObj.SliceSolid _
(slicePt1, slicePt2, slicePt3, True)
sliceObj.Color = acRed
ZoomAll

```

End Sub

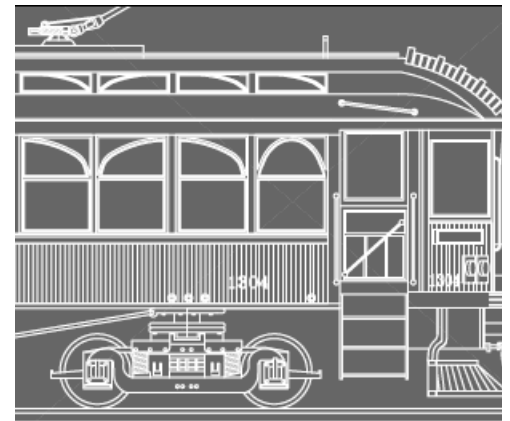
Tương tự như các lưới, vật thể khối sẽ được hiển thị là mô hình khung dây cho đến khi thực hiện ẩn, tô bóng hoặc tạo vỏ. Ngoài ra, có thể tính toán các thuộc tính của khối như thể tích, mô men quán tính, trọng tâm khối... bằng cách sử dụng các thuộc tính sau: MomentOfInertia, PrincipalDirections, PrincipalMoments, ProductOfInertia, RadiiOfGyratation, Volume.

Thuộc tính ContourlinesPerSurface điều khiển số lượng đường để thể hiện các phần đường cong của khung dây.

Thuộc tính RenderSmoothness điều chỉnh độ trơn của các đường thấy và khuất của vật thể.

# TẠO BỐ CỤC VÀ IN ẤN

Sau khi đã tạo bản vẽ trong AutoCAD, thông thường ta sẽ in bản vẽ ra giấy. Một bản vẽ in ra có thể chứa một ảnh đơn của bản vẽ hoặc là sự kết hợp, sắp xếp của nhiều ảnh khác nhau. Trong không gian in, ta có thể tạo nhiều cửa sổ gọi là các khung nhìn nổi, để hiển thị các ảnh khác nhau của bản vẽ. Tùy theo nhu cầu mà ta có thể in một hoặc nhiều khung nhìn, hoặc thiết lập các lựa chọn để xác định xem in cái gì và ảnh nằm vừa trong trang in như thế nào.



Trong chương này

9

- Khái niệm không gian mô hình và không gian in
- Bố cục bản vẽ
- Khái niệm khung nhìn
- In bản vẽ



# 1. Khái niệm không gian mô hình và không gian in

Không gian mô hình (Model Space): là nơi để vẽ, nơi mà người dùng sẽ tạo ra các đối tượng hình học của một mô hình nào đó. Thông thường, khi bắt đầu vẽ trên không gian mô hình, ta cần chỉ rõ giới hạn và đơn vị của bản vẽ.

Không gian in (Paper Space): là nơi thể hiện sự trình bày trên giấy của mô hình khi được in ra. Trong không gian in, ta có thể bố trí các khung nhìn khác nhau của bản vẽ, thu phóng các khung nhìn một cách độc lập và sắp xếp các khung nhìn khác nhau của bản vẽ để in. Có thể có nhiều không gian in, mỗi không gian in là một cách trình bày bản vẽ.

## 2. Bố cục bản vẽ

Tất cả cấu tạo hình học của bản vẽ đều được chứa trong các Layout<sup>1</sup>. Không gian mô hình hình học được chứa trong một Layout riêng có tên là Model. Ta không thể thay đổi tên của Layout này cũng như tạo ra một Layout khác để chứa không gian mô hình hình học. Chỉ có một Layout chứa không gian mô hình trong một bản vẽ.

Không gian in hình học được để trong nhiều Layout. Trong một bản vẽ có thể có nhiều Layout, mỗi cái thể hiện các cấu hình in khác nhau và ta có thể thay đổi tên của các Layout này.

Trong ActiveX Automation, đối tượng ModelSpace bao gồm toàn bộ các đối tượng hình học trong Layout của không gian mô hình. Bởi vì Layout không gian in có thể nhiều hơn một trong cùng một bản vẽ nên đối tượng PaperSpace chỉ đến Layout không gian in hiện hành cuối cùng.

### 2.1. Mối quan hệ giữa Layout và Block

Nội dung của bất cứ Layout nào đều được đặt giữa hai đối tượng ActiveX khác nhau, đó là đối tượng Layout và đối tượng Block. Đối tượng Layout chứa các thiết lập về in ấn và các đặc tính về hiển thị của Layout như là cách thể hiện trên giao diện của AutoCAD. Đối tượng Block chứa thuộc tính hình học cho Layout. Mỗi đối tượng Layout được liên kết duy nhất với một đối tượng Block. Để truy cập vào đối tượng Block đã liên kết với Layout trước, sử dụng thuộc tính Block của Layout đó. Ngược lại, mỗi đối tượng Block cũng được liên kết với duy nhất một đối tượng Layout, và để truy cập vào đối tượng Layout đã liên kết với Block trước, sử dụng thuộc tính Layout cho Block đó.

### 2.2. Khái niệm về cấu hình in

Đối tượng PlotConfiguration giống với đối tượng Layout ở chỗ chúng đều chứa các thông tin về in ấn. Khác biệt của chúng ở chỗ: đối tượng Layout được liên kết với

---

<sup>1</sup> **Layout:** là một khái niệm trong AutoCAD, nó thể hiện cách bố trí các đối tượng hình học trong bản vẽ. Bởi AutoCAD có hai loại không gian dùng để vẽ với mục đích khác nhau (*Model Space* và *Paper Space*) cho nên khái niệm Layout sẽ tương ứng với những không gian vẽ này. Do trong tiếng Việt chưa có từ tương đương cho nên chúng tôi tạm sử dụng từ nguyên gốc tiếng Anh

một đối tượng Block chứa các đối tượng hình học cần in còn đối tượng PlotConfiguration không liên kết với một đối tượng Block nào, nó chỉ là tên của một tập hợp các thông số in ẩn có sẵn dùng cho bất cứ đối tượng hình học nào.

## 2.3. Xác định các cấu hình của Layout

Các cấu hình của Layout điều khiển kết quả in ra sau cùng. Những cấu hình này liên quan đến khổ giấy, tỷ lệ in, vùng in, góc in và tên của thiết bị in. Việc hiểu rõ cách sử dụng một cách hợp lý các cấu hình trên sẽ đảm bảo kết quả in ra như mong muốn. Tất cả các cấu hình cần thiết lập và thay đổi đối với mỗi Layout có thể thực hiện thông qua các thuộc tính và phương thức của đối tượng Layout.

### 2.3.1. Lựa chọn khổ giấy và đơn vị in

Khổ giấy được lựa chọn phụ thuộc vào cấu hình của thiết bị in được thiết lập trong hệ thống. Mỗi thiết bị in khác nhau sẽ có sẵn một danh sách các khổ giấy tiêu chuẩn. Để thay đổi khổ giấy trong Layout, sử dụng thuộc tính `CanonicalMediaName`.

Đơn vị của bản in được xác định thông qua thuộc tính `PaperUnits`. Thuộc tính này có thể lấy một trong 3 giá trị là: `acInches`, `acMillimeters`, hoặc `acPixels`. Nếu thiết bị in được cấu hình in theo chế độ Raster thì buộc phải xác định kích thước theo đơn vị là Pixels (điểm ảnh)

### 2.3.2. Điều chỉnh góc bản in

Góc bản in (plot origin) nằm ở góc phía dưới bên trái của một vùng in xác định và được điều khiển thông qua thuộc tính `PlotOrigin`. Vị trí mặc định của góc này là (0,0). Tuy nhiên, ta có thể chuyển sang chế độ in canh giữa bằng cách gán thuộc tính `CenterPlot` bằng TRUE, khi đó góc bản in sẽ được tự động thay đổi.

### 2.3.3. Thiết lập vùng in

Khi chuẩn bị in một Layout cần chỉ rõ vùng in để xác định những gì cần in trong bản vẽ. Để chỉ định vùng in, ta sử dụng thuộc tính `PlotType`. Thuộc tính này có thể nhận một trong các giá trị dưới đây:

<code>acDisplay</code>	In tất cả các đối tượng đang thể hiện trong không gian mô hình. Tùy chọn này không có hiệu lực khi in từ một Layout không gian in
<code>acExtents</code>	In tất cả những đối tượng nằm trong đường biên hiện tại của không gian được lựa chọn
<code>acLimits</code>	In tất cả các đối tượng trong giới hạn của không gian hiện tại
<code>acView</code>	In cảnh nhìn được đặt tên bởi thuộc tính <code>ViewToPlot</code>
<code>acWindow</code>	In tất cả các đối tượng trong cửa sổ được chỉ định bằng phương thức <code>SetWindowToPlot</code>
<code>acLayout</code>	In tất cả các đối tượng nằm trong phần lề của khổ giấy được sử dụng. Tùy chọn này không có hiệu lực khi in trong không gian mô hình.

Khi tạo một Layout không gian in mới, giá trị mặc định của tùy chọn này là `acLayout`.

#### 2.3.4. Thiết lập Tỷ lệ in

Thông thường, ta vẽ theo kích thước thật của đối tượng, còn khi in ta có thể chỉ ra tỷ lệ chính xác hay tỷ lệ tương đối để bản vẽ phù hợp với khổ giấy. Tỷ lệ in có thể theo chuẩn hoặc tự chọn.

Để nhập tỷ lệ tiêu chuẩn, trước hết gán thuộc tính `UseStandardScale` thành `TRUE`, sau đó nhập tỷ lệ mong muốn thông qua thuộc tính `StandardScale`.

Để nhập tỷ lệ tự chọn, trước hết gán thuộc tính `UseStandardScale` thành `FALSE`, sau đó nhập tỷ lệ đó thông qua phương thức `SetCustomScale`.

Khi chỉ cần xem bản vẽ hay phác thảo thì tỷ lệ chính xác là không quan trọng, trong trường hợp này có thể sử dụng giá trị `acScaleToFit` của thuộc tính `StandardScale` để bản in có tỷ lệ lớn nhất vừa với trang giấy in.

#### 2.3.5. Thiết lập tỷ lệ cho độ dày nét vẽ

Độ dày nét vẽ có thể được phóng đại một cách cân đối phù hợp với tỷ lệ in trong Layout. Diễn hình là độ dày nét vẽ xác định bề rộng nét của đối tượng được in và được in ra với bề rộng nét không phụ thuộc vào tỷ lệ in. Thông thường nên sử dụng tỷ lệ in mặc định đối với mỗi Layout là 1:1. Tuy nhiên nếu muốn in một Layout vừa vào khổ giấy in khác với khổ của Layout đó thì cần chỉ rõ độ dày nét được phóng đại phù hợp với tỷ lệ in mới.

Để phóng đại độ dày nét thì gán thuộc tính `ScaleLineWeights` thành `TRUE` và nếu không cần phóng đại nét vẽ thì gán thuộc tính này thành `FALSE`.

#### 2.3.6. Thiết lập thiết bị in

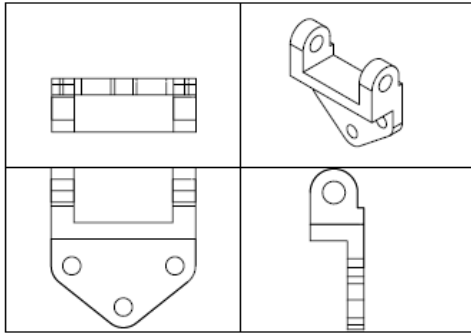
Tên thiết bị in được xác định trong thuộc tính `ConfigName`, thuộc tính này có thể nhận bất cứ giá trị tên thiết bị nào có hiệu lực trong hệ thống<sup>1</sup>. Khi không thiết lập một giá trị cụ thể thì công cụ in sẽ được sử dụng là thiết bị in mặc định trong hệ thống.

### 3. Khái niệm khung nhìn

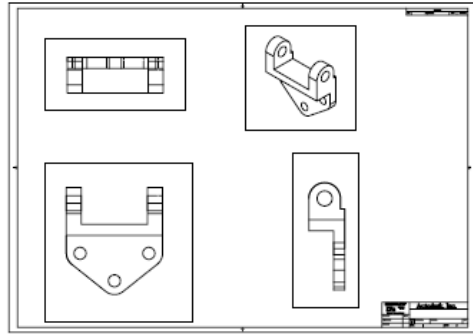
Khi làm việc trong không gian mô hình, ta vẽ trong các khung nhìn xếp cạnh nhau (tương ứng với đối tượng Viewport trong ActiveX Automation). Có thể hiển thị một hay vài khung nhìn khác nhau cùng một lúc. Nếu một vài khung nhìn được hiển thị, thì việc hiệu chỉnh trên một khung nhìn sẽ ảnh hưởng đến tất cả các khung nhìn còn lại. Tuy nhiên có thể gán độ phóng đại, điểm nhìn, lưới và bắt điểm độc lập cho mỗi khung nhìn.

---

<sup>1</sup> Hệ thống ở đây được hiểu là hệ thống máy tính hoặc hệ thống mạng máy tính.



Khung nhìn xếp cạnh nhau

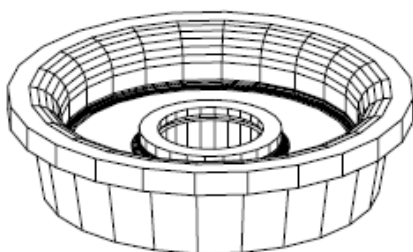


Khung nhìn nổi

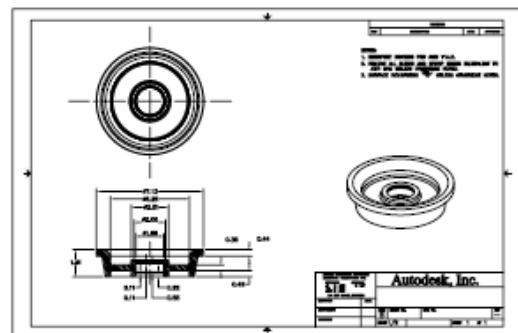
Trong không gian in, ta làm việc trong các khung nhìn nổi của nó (ứng với đối tượng PViewport trong ActiveX Automation) để có thể thể hiện được tất cả các ảnh khác nhau của mô hình. Khung nhìn kiểu nổi được coi như một đối tượng nên có thể di chuyển, thay đổi kích thước và định hình dạng cho phù hợp với Layout. Ta cũng có thể vẽ các đối tượng, chẳng hạn như tên các khối hay chú thích, một cách trực tiếp trong không gian in mà không ảnh hưởng đến bản vẽ trong không gian mô hình.

Trong không gian in, sẽ không thể hiệu chỉnh các đối tượng vẽ trong không gian mô hình. Để truy cập vào mô hình trong đối tượng PViewport, cần chuyển từ không gian in sang không gian mô hình thông qua thuộc tính ActiveSpace. Khi đó ta có thể làm việc với mô hình trong khi vẫn quan sát được các Layout. trong đối tượng PViewport, việc hiệu chỉnh và thay đổi góc nhìn gần giống như trong Viewport. Tuy nhiên, ta có thêm quyền điều khiển cho từng khung nhìn. Ví dụ: có thể đóng băng hoặc tắt các lớp trong một vài khung nhìn mà không ảnh hưởng đến các khung nhìn khác, có thể thể hiện tất cả hay một số khung nhìn, có thể căn hàng vị trí giữa các khung nhìn cũng như tỷ lệ quan sát cho phù hợp với toàn bộ Layout.

Mình họa dưới đây làm thể hiện cách hiển thị các cảnh nhìn khác nhau của mô hình trong không gian in. Mỗi không gian in đại diện cho một đối tượng Pviewport với một cảnh nhìn khác nhau. Trong một cảnh nhìn, lớp đường kích thước đã được tắt. Chú ý rằng phân tiêu đề, khung và ghi chú vẽ trong không gian in không xuất hiện trong không gian mô hình. Ngoài ra, lớp chứa khung của khung nhìn cũng đã được tắt.



Mô hình



Mô hình được hiển thị thông qua các khung nhìn nổi

Khi làm việc với đối tượng Viewport, thuộc tính ActiveSpace phải được gán giá trị là acModelSpace và khi làm việc trong đối tượng PViewport, ActiveSpace có thể

nhận một trong hai giá trị `acModelSpace` hoặc `acPaperSpace`, do vậy cho phép chuyển không gian in sang không gian mô hình và ngược lại khi cần thiết.

## Các đối tượng PViewport, Viewport và thuộc tính ActiveSpace

Kiểu khung nhìn	Trạng thái	Sử dụng
PViewport	<code>ActiveSpace = acPaperSpace</code>	Bố trí các Layout bằng cách tạo ra các khung nhìn nổi, có thêm các tiêu đề, khung hay chú thích. Các thay đổi không làm ảnh hưởng đến mô hình.
PViewport	<code>ActiveSpace= acModelSpace</code>	Làm việc trong khung nhìn nổi để thay đổi mô hình hoặc thay đổi cảnh nhìn. Có thể đóng băng hoặc tắt các lớp trong từng khung nhìn.
Viewport	<code>ActiveSpace= acModelSpace</code>	Chia màn hình thành các khung nhìn xếp cạnh nhau để hiệu chỉnh trên các cảnh nhìn khác nhau của mô hình

Trong AutoCAD ActiveX Automation, thuộc tính `ActiveSpace` được sử dụng để điều khiển biến hệ thống `TILEMODE`. Gán `ThisDrawing.ActiveSpace=acModelSpace` tương đương với gán `TILEMODE=on` và khi gán `ThisDrawing.ActiveSpace=acPaperSpace` tương đương với gán `TILEMODE=off`.

Tương tự, thuộc tính `MSpace` tương đương với hai lệnh của AutoCAD là `MSPACE` và `PSPACE`. Khi gán `ThisDrawing.MSpace=TRUE` sẽ tương đương với sử dụng lệnh `MSPACE` để chuyển sang không gian mô hình, và khi thuộc tính trên được gán giá trị `FALSE` sẽ tương đương với lệnh `PSPACE` để chuyển sang không gian in.

Cần chú ý trước khi gán thuộc tính `MSPACE` thành `TRUE` cần sử dụng phương thức `Display` để khởi tạo các thông số đồ họa ban đầu trước khi chuyển sang không gian mô hình và để AutoCAD thực hiện các thao tác cần thiết khác. Tuy nhiên, trong giao tiếp ActiveX Automation, người lập trình cần phải cẩn trọng với phương thức khởi tạo này

**CHÚ Ý** Nên ghi nhớ rằng cần phải bật màn hình bằng cách gọi phương thức `Display` cho ít nhất là một đối tượng `Pviewport` trước khi gán thuộc tính `Mspace` bằng `TRUE`. Nếu không bật được màn hình thì sẽ làm phát sinh lỗi do thuộc tính `Mspace`.

### 3.1. Chuyển sang Layout của không gian in

Từ không gian mô hình, có thể chuyển sang Layout của không gian in được kích hoạt gần nhất theo các bước sau:

- 1 Gán thuộc tính `ActiveSpace` thành `acPaperSpace`:

```
ThisDrawing.ActiveSpace=acPaperSpace
```

- 2 Chuyển thuộc tính `MSpace` thành `FALSE`:

```
ThisDrawing.MSpace=FALSE
```

Khi đang ở trong không gian in, AutoCAD sẽ hiển thị biểu tượng của UCS của không gian in ở góc dưới bên trái của màn hình đồ họa. Con trỏ vẽ của AutoCAD thể hiện rằng vùng Layout của không gian in (chứ không phải vùng hiển thị trong khung nhìn) có thể được hiệu chỉnh.

### 3.2. Chuyển sang Layout của không gian mô hình

Từ không gian in, có thể chuyển sang không gian mô hình trong khung nhìn xếp cạnh nhau hoặc khung nhìn nổi theo các bước sau:

**1** Dùng phương thức `Display` để khởi tạo các thông số đồ họa

```
ThisDrawing.ActivePViewport.Display TRUE
```

**2** Chuyển thuộc tính `MSpace` thành `TRUE`:

```
ThisDrawing.MSpace=TRUE
```

Các bước thực hiện trên sẽ chuyển sang không gian mô hình và các khung nhìn nổi.

---

**CHÚ Ý** Phải tạo khung nhìn nổi trước khi chuyển sang không gian mô hình.

---

**Để chuyển sang khung nhìn xếp cạnh nhau thì thực hiện thêm bước sau:**

- Gán thuộc tính `ActiveSpace` thành `acModelSpace`:

```
ThisDrawing.ActiveSpace=acModelSpace
```

### 3.3. Tạo khung nhìn trong không gian in

Các khung nhìn của không gian in được tạo bởi phương thức `AddPViewport`. Phương thức này cần có điểm giữa, bề rộng và chiều cao của khung nhìn mới. Trước khi tạo khung nhìn, sử dụng thuộc tính `ActiveSpace` không gian in thành không gian vẽ hiện tại (thông thường được sử dụng bằng cách gán `TILEMODE=0`).

Sau khi tạo đối tượng `PViewport`, có thể gán các thuộc tính của chúng như: `Direction`, `LensLength`, `GridOn`, có thể điều khiển các thuộc tính của bản thân khung nhìn như `Layer`, `LineType`, `LinetypeScale`.

#### Chuyển từ không gian mô hình sang không gian in

Ví dụ dưới đây sẽ chuyển AutoCAD sang không gian in, tạo khung nhìn nổi, gán và kích hoạt khung nhìn.

```
Sub Ch9_SwitchToPaperSpace()  
    ' Gán không gian vẽ hiện tại thành không gian in  
    ThisDrawing.ActiveSpace = acPaperSpace  
    ' Tạo khung nhìn trong không gian in  
    Dim newVport As AcadPViewport  
    Dim center(0 To 2) As Double  
    center(0) = 3.25  
    center(1) = 3  
    center(2) = 0  
    Set newVport = ThisDrawing.PaperSpace.AddPViewport _  
        (center, 6, 5)  
    ' Thay đổi hướng nhìn cho mỗi khung nhìn  
    Dim viewDir(0 To 2) As Double  
    viewDir(0) = 1
```



```

viewDir(1) = 1
viewDir(2) = 1
newVport.direction = viewDir
' Kích hoạt khung nhìn
newVport.Display True
' Chuyển sang không gian mô hình
ThisDrawing.MSpace = True
' Gán newVport thành khung nhìn hiện tại
ThisDrawing.ActivePViewport = newVport
' Zoom Extents trong không gian mô hình
ZoomExtents
' Tắt chế độ sửa đổi không gian mô hình
ThisDrawing.MSpace = False
' Zoom Extents trong không gian in
ZoomExtents
End Sub

```

Thứ tự các bước thực hiện trên rất quan trọng. Nói chung, mọi thứ phải được thực hiện theo trình tự như các dòng lệnh trong AutoCAD, nếu không thực hiện được như vậy thì điều không mong muốn sẽ ảnh hưởng đến việc định nghĩa và kích hoạt các khung nhìn.

---

**CHÚ Ý** Để gán hay sửa các thông số của cảnh nhìn (hướng nhìn, ...) thì phương thức `Display` của đối tượng `Viewport` cần được gán thành `False`, và trước khi gán khung nhìn hiện tại thì phương thức `Display` phải được gán trở lại giá trị `True`.

---

#### Ví dụ tạo 4 khung nhìn nổi

Ví dụ này lấy theo ví dụ ở phần trên và tiếp tục ví dụ đó bằng cách tạo ra 4 khung nhìn nổi và gán các góc nhìn khác nhau.

```

Sub Ch9_FourPViewports()
Dim topVport, frontVport As AcadPViewport
Dim rightVport, isoVport As AcadPViewport
Dim pt(0 To 2) As Double
Dim viewDir(0 To 2) As Double
ThisDrawing.ActiveSpace = acPaperSpace
ThisDrawing.MSpace = True
' Lấy PViewport đang có và chuyển thành topVport
pt(0) = 2.5: pt(1) = 5.5: pt(2) = 0
Set topVport = ThisDrawing.ActivePViewport
' Không cần thiết lập hướng nhìn cho khung nhìn phía trên
topVport.center = pt
topVport.width = 2.5
topVport.height = 2.5
topVport.Display True
ThisDrawing.MSpace = True
ThisDrawing.ActivePViewport = topVport
ZoomExtents
ZoomScaled 0.5, acZoomScaledRelativePSPACE
' Tạo và thiết lập frontVport
pt(0) = 2.5: pt(1) = 2.5: pt(2) = 0
Set frontVport = ThisDrawing.PaperSpace.AddPViewport _
    (pt, 2.5, 2.5)
viewDir(0) = 0: viewDir(1) = 1: viewDir(2) = 0
frontVport.direction = viewDir
frontVport.Display acOn
ThisDrawing.MSpace = True

```



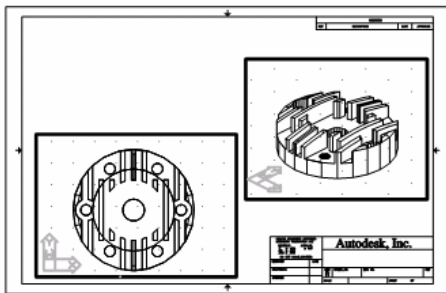
```

ThisDrawing.ActivePViewport = frontVport
ZoomExtents
ZoomScaled 0.5, acZoomScaledRelativePSPACE
'Tạo và thiết lập rightVport
pt(0) = 5.5: pt(1) = 5.5: pt(2) = 0
Set rightVport = ThisDrawing.PaperSpace.AddPViewport _
    (pt, 2.5, 2.5)
viewDir(0) = 1: viewDir(1) = 0: viewDir(2) = 0
rightVport.direction = viewDir
rightVport.Display acOn
ThisDrawing.MSpace = True
ThisDrawing.ActivePViewport = rightVport
ZoomExtents
ZoomScaled 0.5, acZoomScaledRelativePSPACE
'Tạo và thiết lập isoVport
pt(0) = 5.5: pt(1) = 2.5: pt(2) = 0
Set isoVport = ThisDrawing.PaperSpace.AddPViewport _
    (pt, 2.5, 2.5)
viewDir(0) = 1: viewDir(1) = 1: viewDir(2) = 1
isoVport.direction = viewDir
isoVport.Display acOn
ThisDrawing.MSpace = True
ThisDrawing.ActivePViewport = isoVport
ZoomExtents
ZoomScaled 0.5, acZoomScaledRelativePSPACE
'Kết thúc: thực hiện tái tạo lại toàn bộ các khung nhìn
ThisDrawing.Regen True
End Sub

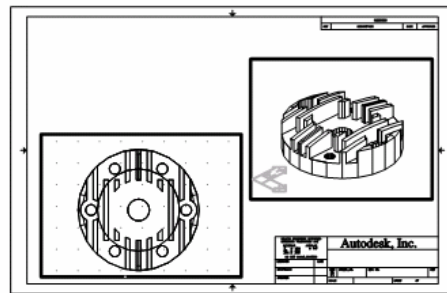
```

### 3.4. Thay đổi cảnh nhìn và nội dung khung nhìn

Thay đổi cảnh nhìn trong đối tượng Viewport chỉ thực hiện được khi ở trong không gian mô hình và đối tượng Viewport đó phải đang được kích hoạt.



g Lưới và biểu tượng UCS trong hai khung nhìn



Lưới trong một khung nhìn và biểu tượng UCS trong khung nhìn khác

#### Hiệu chỉnh bản vẽ trong khung nhìn nổi :

1 Trong không gian mô hình, kích hoạt khung nhìn cần hiệu chỉnh bằng cách gán thuộc tính `ActiveViewport` như sau:

```
ThisDrawing.ActiveViewport=MyViewportObject
```

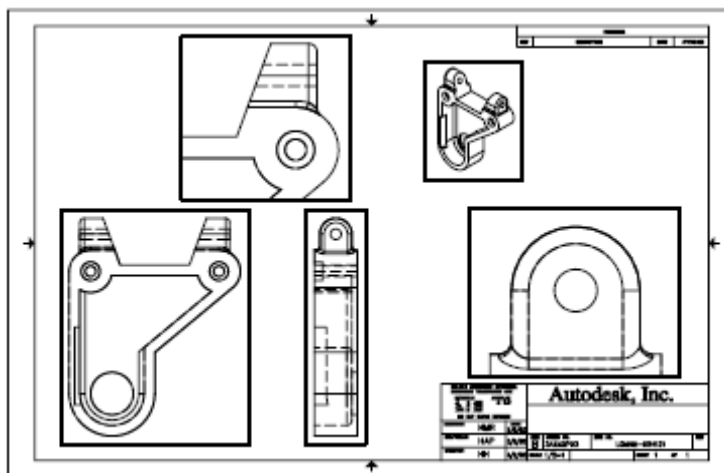
2 Hiệu chỉnh bản vẽ.

Đồng thời có thể tạo đối tượng khác như chú thích, kích thước trong không gian in, khi đó cần gán thuộc tính `ActiveSpace` thành `FALSE` và chuyển sang không gian in

bằng cách sử dụng thuộc tính `mSpace`. Các đối tượng được tạo ra trong không gian in sẽ không được nhìn thấy trong không gian mô hình.

### 3.5. Đặt tỷ lệ cảnh nhìn theo không gian in

Trước khi in cần lập tỷ lệ in chính xác cho mỗi phần của bản vẽ. Đặt tỷ lệ tương đối trong không gian in sẽ thiết lập nên tỷ lệ hợp lý cho từng nội dung được hiển thị. Hình vẽ minh họa dưới đây mô tả một bản vẽ với một số khung nhìn, mỗi khung nhìn có tỷ lệ khác nhau và cảnh nhìn khác. Để in bản vẽ này với một tỷ lệ chính xác thì mỗi cảnh nhìn cần được đặt tỷ lệ theo khổ giấy, chứ không phải là tỷ lệ so với cảnh nhìn trước hay so với toàn bộ mô hình.



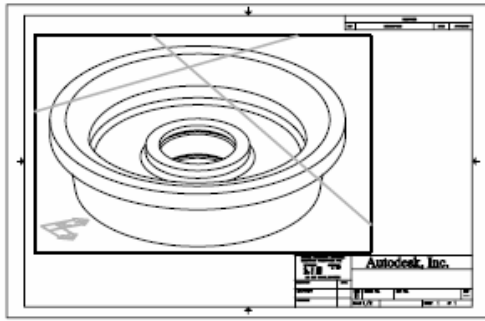
Khi làm việc trong không gian in, hệ số tỷ lệ thể hiện tỷ số giữa kích thước sẽ in và kích thước thực của mô hình thể hiện trong khung nhìn. Để nhận được tỷ lệ này, chia đơn vị của không gian in cho đơn vị của không gian mô hình. Ví dụ đối với bản vẽ tỷ lệ 1:4 thì hệ số phóng đại của một đơn vị trong không gian in tương ứng với 4 đơn vị của không gian mô hình.

Sử dụng phương thức `ZoomScaled` để co giãn các khung nhìn tương ứng với đơn vị giấy in. Phương thức này cần 3 thông số đầu vào: khung nhìn cần đặt tỷ lệ, hệ số tỷ lệ, cách thức thay đổi tỷ lệ. Thông số thứ 3 là tùy chọn và xác định cách thức thay đổi tỷ lệ:

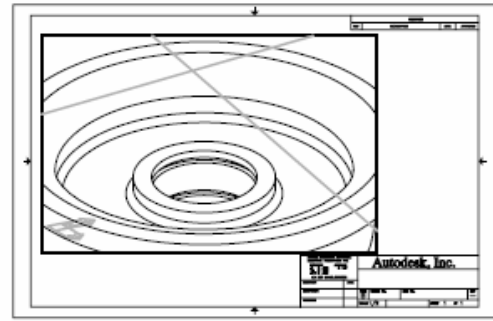
- Theo giới hạn vẽ
- Theo cảnh nhìn hiện tại
- Theo đơn vị giấy in

Để chỉ ra tỷ lệ tương đối với đơn vị giấy in, nhập vào cho thông số thứ ba là hằng số `acZoomScaleRelativePSPACE`.

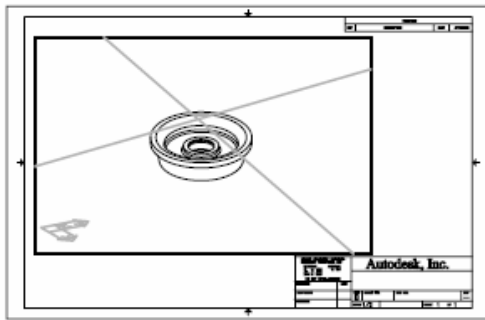
Như minh họa trong các hình vẽ dưới đây, nếu nhập tỷ lệ tương đối so với đơn vị của không gian in là 2 thì khung nhìn sẽ giãn ra gấp 2 lần kích thước theo đơn vị trong không gian in. Và hệ số tỷ lệ tương đối là 0.5 so với đơn vị của không gian in sẽ co khung nhìn lại bằng một nửa kích thước theo đơn vị của không gian in. Khi đó mô hình sẽ được in ra với kích thước bằng nửa kích thước thực tế.



Khung nhìn hiện tại



Phóng theo tỷ lệ 2xp

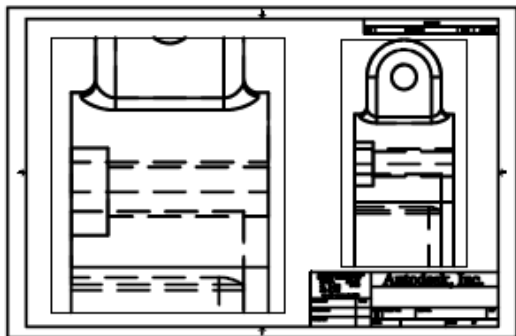


Phóng theo tỷ lệ 0.5xp

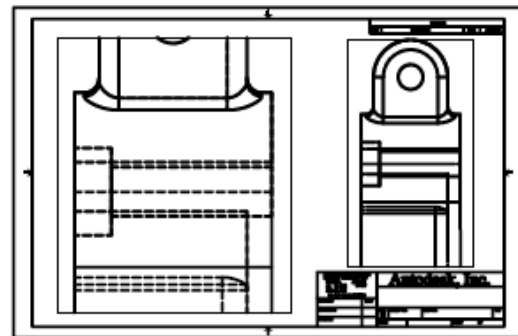
### 3.6. Đặt tỷ lệ cho mẫu của kiểu đường trong không gian in

Trong không gian in, ta có thể đặt tỷ lệ cho mẫu của kiểu đường theo hai cách. Cách thứ nhất là dựa vào đơn vị vẽ của không gian tạo ra đối tượng (không gian mô hình hoặc không gian in). Còn theo cách thứ hai, tỷ lệ của các kiểu đường đều được đặt thống nhất theo đơn vị của không gian in. Ta cũng có thể sử dụng biến hệ thống `PSLTSCALE` nhằm đảm bảo một tỷ lệ chung cho tất cả kiểu đường của các đối tượng khác nhau trong các khung nhìn có tỷ lệ phóng đại khác nhau. Điều này cũng có ảnh hưởng đến sự hiển thị của các đường thẳng trong không gian 3 chiều.

Trong hình vẽ dưới đây, mẫu kiểu đường của các đường thẳng trong không gian mô hình đều được đặt cùng một tỷ lệ trong không gian in bằng cách sử dụng biến hệ thống `PSLTSCALE`. Điểm cần lưu ý là mặc dù các đối tượng trong hai khung nhìn có tỷ lệ phóng đại khác nhau nhưng kiểu đường đều có cùng một tỷ lệ.



`psltscale=1`, đường nét đứt thay đổi tỷ lệ theo không gian in



`psltscale=0`, đường nét đứt có tỷ lệ theo không gian nơi tạo ra đối tượng

Để gán giá trị cho biến hệ thống `PSLTSCALE`, ta sử dụng phương thức `SetVariable`

### 3.7. Ẩn các đường thẳng trong khung nhìn khi in

Khi bản vẽ chứa các đối tượng mặt 3D, lưới, đối tượng đập nổi, bề mặt, khối đặc, ta cần phải ẩn các đường khuất khi in.

Để ẩn đi các đường khuất trong khung nhìn của không gian in (đối tượng `PViewport`) ta sử dụng thuộc tính `RemoveHiddenLines` của khung nhìn đó. Thuộc tính này nhận giá trị logic. Khi không muốn in đường khuất, ta gán giá trị `TRUE` và ngược lại ta gán giá trị `FALSE`.

Để ẩn đi các đường khuất trong khung nhìn của không gian mô hình (đối tượng `Viewport`) khi in ẩn, ta sử dụng thuộc tính `PlotHidden` của đối tượng `Layout`. Và tương tự, thuộc tính này nhận giá trị logic. Khi không muốn in đường khuất, ta gán giá trị `TRUE` và ngược lại ta gán giá trị `FALSE`.

## 4. In bản vẽ

Bản vẽ có thể được in giống như khi nhìn thấy trong không gian mô hình hoặc có thể tiến hành in sau khi đã tạo bố cục trang in trong không gian in. Thông thường, việc in ẩn trong không gian mô hình thích hợp khi cần xem trước hoặc kiểm tra bản vẽ trước khi tạo bố cục trong không gian in. Sau khi đã hoàn thành bản vẽ trong không gian mô hình, ta sẽ bắt đầu tạo bố cục để in ẩn trong không gian in.

Việc in ẩn liên quan đến hai đối tượng của `ActiveX Automation` là đối tượng `Layout` và đối tượng `Plot`. Đối tượng `Layout` chứa các thông số in cho một `Layout` đã có từ trước, còn đối tượng `Plot` chứa các phương thức và thuộc tính để khởi tạo và điều khiển quá trình in.

### 4.1. Thao tác in cơ bản

Với đối tượng `Plot`, có thể sử dụng các phương thức và thuộc tính sau:

<code>PlotToFile</code>	in bản vẽ thành tệp
<code>PlotToDevice</code>	in bản vẽ trên máy in
<code>DisplayPlotPreview</code>	xem trước khi in một vùng được chỉ định
<code>SetLayoutsToPlot</code>	lập danh sách các <code>Layout</code> sẽ được in trong lần gọi tiếp theo của phương thức <code>PlotToFile</code> hoặc <code>PlotToDevice</code>
<code>StartBatchMode</code>	khởi tạo in theo nhóm
<code>QuietErrorMode</code>	bật/tắt chế độ báo lỗi ngầm <sup>1</sup> đối với các lỗi in
<code>NumberOfCopies</code>	định số bản in của mỗi lần in.

---

<sup>1</sup> **Báo lỗi ngầm (`QuietErrorMode`):** là chế độ mà các lỗi xuất hiện trong quá trình in sẽ được ghi vào một tệp riêng do đó không ngắt quá trình in bằng các thông báo. Chế độ này rất thích hợp với những ứng dụng không được ngắt khi in, ví dụ khi tiến hành in theo nhóm.

**BatchPlotProgress** nhận trạng thái hiện tại của chế độ in theo nhóm hoặc ngắt chế độ in theo nhóm

Phương thức `SetLayoutsToPlot` phải được gọi trước khi gọi phương thức `PlotToFile` hoặc `PlotToDevice`, nếu phương thức này không được gọi hoặc được gọi nhưng thông số đầu vào là giá trị `NULL` thì Layout hiện hành sẽ được in.

Thuộc tính `NumberOfCopies` chỉ ra số lượng bản in. Nếu thuộc tính này không được khởi tạo trước mỗi lần gọi phương thức `PlotToDevice` thì giá trị được xác định gần nhất của thuộc tính này sẽ được sử dụng.

Chế độ in thành nhóm được cung cấp để hỗ trợ việc áp dụng tiện ích `BatchPlot`. Trước khi khởi tạo chế độ này cần gán `QuietErrorMode = TRUE` để quá trình in không bị ngắt giữa chừng. Tiếp đó sử dụng phương thức `StartBatchMode` để khởi tạo chế độ in theo nhóm. Dùng thuộc tính `BatchPlotProgress` để kiểm tra quá trình in hoặc để ngắt chế độ này.

## 4.2. In trong không gian mô hình

Khi thực hiện in một bản vẽ lớn theo các hệ đơn vị khác nhau trong không gian mô hình, nếu không xác định các thông số cụ thể cho việc in ấn thì bản vẽ sẽ được in ra theo các giá trị mặc định như: phần bản vẽ in ra là toàn bộ những gì được biểu diễn, máy in mặc định của hệ thống, tỷ lệ co giãn vừa với khổ giấy, góc quay bằng 0, điểm gốc của bản vẽ là (0,0). Các thông số in nêu trên có thể sửa lại bằng cách thay đổi thuộc tính của đối tượng `Layout` liên kết với không gian mô hình.

### In phần mở rộng của không gian mô hình hiện hành

Ví dụ dưới đây, trước hết kiểm tra để đảm bảo không gian mô hình là không gian đang làm việc, sau đó nó sẽ tạo một số thông số in, cuối cùng bản in sẽ được gửi đến máy in nhờ phương thức `PlotToDevice`

```
Sub Ch9_PrintModelSpace()  
    ' Kiểm tra không gian làm việc là không gian mô hình  
    If ThisDrawing.ActiveSpace = acPaperSpace Then  
        ThisDrawing.MSpace = True  
        ThisDrawing.ActiveSpace = acModelSpace  
    End If  
    ' Xác định phạm vi và tỷ số co giãn cho vùng in  
    ThisDrawing.ModelSpace.Layout.PlotType = acExtents  
    ThisDrawing.ModelSpace.Layout.  
        StandardScale = acScaleToFit  
    ' Gán số lượng bản in là 1  
    ThisDrawing.Plot.NumberOfCopies = 1  
    ' Bắt đầu in  
    ThisDrawing.Plot.PlotToDevice  
End Sub
```

Tên thiết bị có thể được chỉ ra bằng cách sử dụng thuộc tính `ConfigName`, hoặc bằng cách dùng phương thức `PlotToDevice` có tham số chỉ đến tệp `PC3` (tệp cấu hình thiết bị in).

### 4.3. In trong không gian in

Có thể in một hay nhiều Layout trong không gian in cùng một lúc, có thể in Layout hiện hành như đã trình bày trong phần “*In trong không gian mô hình*” trang 245, hoặc in Layout được chỉ định theo tên của nó.

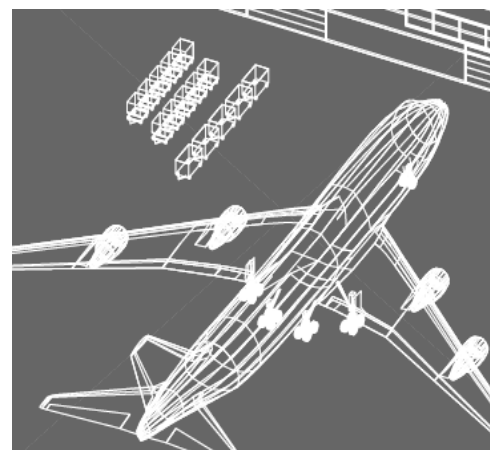
#### In 2 bản vẽ trong không gian in

Ví dụ dưới đây sẽ chuyển hai Layout trong không gian in “Layout1” và “Layout2” tới máy in. Cần chú ý là hai Layout này phải tồn tại trong bản vẽ khi dùng đoạn mã này. Ví dụ này đầu tiên sẽ tạo ra một mảng kiểu chuỗi (string) chứa tên của các Layout cần in, sau đó sử dụng mảng này để làm đầu vào đối với phương thức SetLayoutsToPlot, tiếp theo sẽ gán số lượng bản in và cuối cùng gửi các bản in đến máy in mặc định.

```
Sub Ch9_PrintPaperSpace()  
    ' Tạo các Layout để in trong không gian in  
    Dim strLayouts(0 To 1) As String  
    Dim varLayouts As Variant  
    strLayouts(0) = "Layout1"  
    strLayouts(1) = "Layout2"  
    varLayouts = strLayouts  
    ThisDrawing.Plot.SetLayoutsToPlot varLayouts  
    ' Gán số lượng bản in là 1  
    ThisDrawing.Plot.NumberOfCopies = 1  
    ' Bắt đầu in  
    ThisDrawing.Plot.PlotToDevice  
End Sub
```

# KỸ THUẬT VẼ NÂNG CAO VÀ TỔ CHỨC BẢN VẼ

Khi có nhiều kinh nghiệm với AutoCAD, có thể sử dụng các hỗ trợ nâng cao của chương trình để tăng cường khả năng của các ứng dụng. Bản vẽ tạo ra có thể chứa: ảnh hàng không, ảnh vệ tinh, ảnh đồ họa cũng như các ảnh do máy tính tạo ra. Bên cạnh đó AutoCAD cũng cung cấp một số tính năng trợ giúp tổ chức dữ liệu, cho phép bổ sung thêm thông tin cho các đối tượng trong bản vẽ.



## Trong chương này **10**

- **Làm việc với ảnh Raster**
- **Sử dụng Khối và Thuộc tính**
- **Sử dụng Tham chiếu ngoài**
- **Kết nối và Khôi phục lại dữ liệu mở rộng**

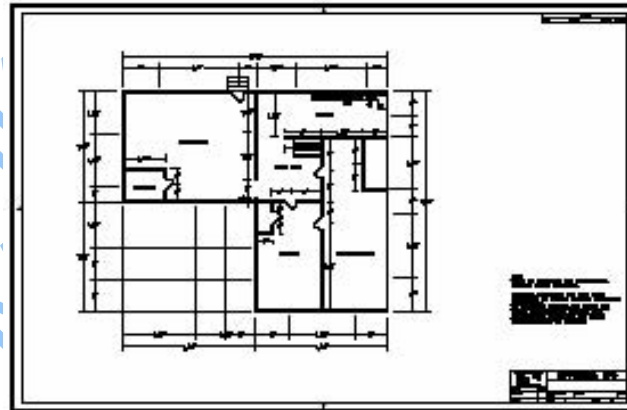


# 1. Làm việc với ảnh Raster

Có thể chèn ảnh raster<sup>1</sup> vào bản vẽ AutoCAD (là bản vẽ dạng vector<sup>2</sup>), sau đó lưu lại. Có rất nhiều lý do cần kết hợp ảnh dạng Raster với tệp dạng véc tơ, bao gồm cả tài liệu được quét lại, bản fax, sử dụng ảnh kỹ thuật số, ảnh hàng không, ảnh vệ tinh, tạo các hiệu ứng như các hình mờ và lô gô, chèn ảnh do máy tính tạo ra từ các chương trình biểu diễn hình ảnh trực quan như AutoVision.

## 1.1. Ảnh Raster trong bản vẽ

Ảnh Raster bao gồm một lưới ô hình vuông có kích thước nhỏ hay là các điểm được biết đến là các điểm ảnh (pixel). Ví dụ ảnh của một ngôi nhà gồm hàng loạt các điểm màu biểu diễn hình ảnh của nó. Mỗi ảnh Raster tương ứng với các điểm ảnh trong một lưới nhất định.



Ảnh Raster cũng giống như các đối tượng khác trong bản vẽ AutoCAD có thể sao chép, dịch chuyển hay cắt xén. Ảnh có thể được thay đổi, điều chỉnh độ tương phản hay cắt xén ảnh theo đường bao hình chữ nhật hoặc một đa giác. AutoCAD hỗ trợ

<sup>1</sup> **Ảnh Raster:** là ảnh kiểu ảnh xạ bit hay ảnh dạng màn hình, các hình ảnh được thể hiện bởi các chấm nhỏ riêng biệt.

<sup>2</sup> **Bản vẽ Vector:** là bản vẽ thể hiện theo kiểu đồ họa hướng đối tượng, trong đó các đối tượng hình học được lưu trữ và xử lý bởi các phương trình, công thức toán học.

hầu hết các định dạng của tệp ảnh được sử dụng phổ biến trong các ứng dụng kỹ thuật ảnh chủ yếu như: đồ họa máy tính, quản lý bản vẽ, bản đồ, các hệ thống thông tin địa lý (GIS). Các ảnh có thể là Bitonal<sup>1</sup>, xám 8-bit, màu 8-bit, hoặc màu 24-bit. Một số định dạng tệp ảnh hỗ trợ ảnh tạo bởi các chấm điểm trong suốt. Khi ảnh trong suốt được mở, AutoCAD có thể nhận ra các điểm ảnh đó và cho phép các đối tượng đồ họa trên màn hình AutoCAD được “nhìn xuyên” qua điểm ảnh. (Trong ảnh Bitonal, các điểm ảnh nền được coi như là trong suốt). Ảnh trong suốt thường có màu sắc hoặc thuộc dải màu xám.

Mặc dù các phần đuôi mở rộng được liệt kê nhưng AutoCAD nhận dạng từ nội dung của tệp chứa ảnh chứ không phải từ phần mở rộng của tệp.

### Các định dạng ảnh được hỗ trợ trong AutoCAD

Loại	Mô tả và phiên bản	Phần mở rộng
BMP	Windows and OS/2 bitmap format	<i>.bmp, .dib, .rle</i>
CALS-I	Mil-R-Raster I	<i>.gp4, .mil, .rst, .cg4, .cal</i>
GeoSPOT	GeoSPOT ( tệp <i>.bil</i> phải được đi kèm với tệp <i>.hdr</i> và <i>.pal</i> cùng với các tệp dữ liệu liên quan trong cùng một thư mục)	<i>.bil</i>
IG4	Image Systems Group 4	<i>.ig4</i>
IGS	Image Systems Grayscale	<i>.igs</i>
JPEG	Joint Photographics Expert Group	<i>.jpg</i>
FLIC	FLIC Autodesk Animator Animation	<i>.flc, .fli</i>
PCX	Picture PC Paintbrush Picture	<i>.pcx</i>
PICT	Picture Macintosh Picture	<i>.pct</i>
PNG	Portable Network Graphic	<i>.png</i>
RLC	Run Length Compressed	<i>.rlc</i>
TARGA	True Vision Raster-Based Data Format	<i>.tga</i>
TIF	Tagged Image Format	<i>.tif</i>

## 1.2. Đính kèm và đặt tỷ lệ ảnh Raster

Ảnh có thể đặt trong tệp bản vẽ nhưng chúng thực sự không phải là một phần của bản vẽ. Chúng được liên kết với tệp bản vẽ thông qua đường dẫn hoặc ID của một

<sup>1</sup> **Bitonal**: là loại ảnh có hai tông màu: tiền cảnh và màu nền.

tệp quản lý dữ liệu. Đường dẫn của tệp ảnh có thể bị thay đổi hoặc bị xoá bất cứ lúc nào. Sử dụng đường dẫn để gắn ảnh vào bản vẽ làm tăng đáng kể dung lượng của bản vẽ. Để gắn ảnh, trước hết cần tạo ra đối tượng Raster trong bản vẽ bằng phương thức `AddRaster`. Phương thức này yêu cầu 4 tham số đầu vào: tên của tệp ảnh đính kèm, điểm chèn ảnh trong bản vẽ, tỷ lệ phóng đại và góc quay của ảnh. Chú ý rằng đối tượng Raster chỉ là một liên kết độc lập của ảnh chứ không phải là ảnh.

Một ảnh Raster có thể đính kèm nhiều lần vào bản vẽ và mỗi lần đính kèm sẽ tương ứng cần tạo một đối tượng Raster mới. Mỗi phần đính kèm đó có đường bao riêng và các thông số về độ sáng tối, tương phản, độ đậm nhạt, độ trong suốt riêng. Một ảnh đơn có thể cắt được thành nhiều phần và sắp xếp lại một cách độc lập trong bản vẽ.

Có thể gán tỷ lệ co giãn ảnh Raster khi tạo đối tượng Raster để tỷ lệ về hình học của ảnh phù hợp với tỷ lệ hình học trong bản vẽ. Khi chọn một ảnh để đính kèm, ảnh đó sẽ được chèn vào với tỷ lệ là 1 đơn vị đo của ảnh bằng 1 đơn vị bản vẽ. Để thiết lập tỷ lệ phóng đại cho ảnh cần biết đơn vị đo nào (inches, feet, metres...) đang sử dụng là đơn vị vẽ của AutoCAD. Tệp ảnh nhất thiết phải có thông tin về độ phân giải xác định DPI (Dots Per Inche: số điểm trên một inch) và số các điểm trong ảnh.

Nếu một ảnh có thông tin về độ phân giải, AutoCAD kết hợp giá trị đó với tỷ lệ co giãn và đơn vị đo của AutoCAD đang sử dụng trong bản vẽ. Ví dụ: nếu ảnh raster là một bản thiết kế đã được quét với tỷ lệ: 1 cm trên ảnh bằng 6m (600 cm) thực tế, tức là tỷ lệ 1:600; bản vẽ sử dụng đơn vị cm và 1 cm bằng 1 đơn vị vẽ thì tỷ số phóng đại của ảnh được lập bằng cách gán giá trị cho thông số `ScaleFactor` của phương thức `AddRaster` là 600. Khi đó AutoCAD sẽ chèn ảnh với tỷ lệ phù hợp với các đối tượng hình học trong bản vẽ vector.

---

**CHÚ Ý** Nếu không có thông số về độ phân giải cho tệp ảnh đính kèm thì AutoCAD sẽ tính bề rộng gốc của ảnh bằng 1 đơn vị. Sau khi chèn ảnh, bề rộng của ảnh trong hệ đơn vị đo của AutoCAD bằng với hệ số phóng đại.

---

### Đính kèm ảnh Raster

Ví dụ dưới đây sẽ chèn ảnh Raster vào không gian mô hình, ảnh này chứa trong tệp `watch.jpg` đặt trong thư mục `Sample`. Nếu không tìm được tệp ảnh này hoặc được đặt trong thư mục khác thì chỉ cần gán tên và đường dẫn đầy đủ một tệp ảnh bất kỳ nào đó có sẵn cho biến `imageName`.

```
Sub Ch10_AttachingARaster()  
    Dim insertionPoint(0 To 2) As Double  
    Dim scalefactor As Double  
    Dim rotationAngle As Double  
    Dim imageName As String  
    Dim rasterObj As AcadRasterImage  
    imageName = "C:/Acad2000/sample/watch.jpg"  
    insertionPoint(0) = 5  
    insertionPoint(1) = 5  
    insertionPoint(2) = 0  
    scalefactor = 2  
    rotationAngle = 0  
    On Error GoTo ERRORHANDLER  
    ' Gắn ảnh raster vào không gian mô hình
```

```

Set rasterObj = ThisDrawing.ModelSpace.AddRaster_
    (imageName, insertionPoint, scalefactor, rotationAngle)
ZoomAll
Exit Sub
ERRORHANDLER:
MsgBox Err.Description
End Sub

```

### 1.3. Quản lý ảnh Raster

Có thể thay đổi tên ảnh, tên tệp và đường dẫn của tệp thông qua các thuộc tính của đối tượng `Raster`.

#### 1.3.1. Thay đổi đường dẫn của tệp

Đường dẫn và tên tệp của ảnh có thể lấy hoặc thay đổi được bằng thuộc tính `ImageFile`. Đường dẫn được tạo bởi thuộc tính này là đường dẫn thật để AutoCAD tìm kiếm ảnh.

Nếu AutoCAD không thể định vị được bản vẽ (ví dụ trong trường hợp bản vẽ bị chuyển đi chỗ khác sau khi tên của tệp đã được gán cho thuộc tính `ImageFile`) thì nó sẽ xoá các thông tin về đường dẫn tương đối hay tuyệt đối chứa trong tên tệp (ví dụ, `images\tree.tga` hoặc `c:\myproject\images\tree.tga` sẽ chuyển thành `tree.tga`) và tìm đường dẫn mới được định nghĩa bởi phương thức `SetProjectFilePath` của đối tượng `Preferences`. Nếu bản vẽ không được định vị theo các đường dẫn, nó sẽ dùng đường dẫn đầu tiên tìm lại được. Có thể xoá đường dẫn trong tên tệp hoặc chỉ ra một đường dẫn tương đối bằng cách gán lại thuộc tính `ImageFile`.

Thay đổi đường dẫn trong thuộc tính `ImageFile` không ảnh hưởng đến thiết lập tìm kiếm đường dẫn của các tệp trong dự án.

#### 1.3.2. Đặt tên ảnh

Tên của ảnh không nhất thiết trùng với tên của tệp ảnh đó. Khi đính kèm ảnh vào bản vẽ, AutoCAD sử dụng tên tệp không bao gồm phần mở rộng làm tên của ảnh. Có thể thay đổi tên của ảnh mà không làm ảnh hưởng đến tên của tệp ảnh đó. Tệp ảnh được thể hiện thông qua thuộc tính `ImageFile` của đối tượng `Raster`. Thay đổi thuộc tính `ImageFile` sẽ dẫn đến thay đổi ảnh trong bản vẽ. Tên ảnh được thể hiện bởi thuộc tính `Name` và khi thay đổi thuộc tính `Name` thì chỉ thay đổi tên của ảnh mà không ảnh hưởng đến tệp liên kết với ảnh đó.

### 1.4. Hiệu chỉnh ảnh và đường biên

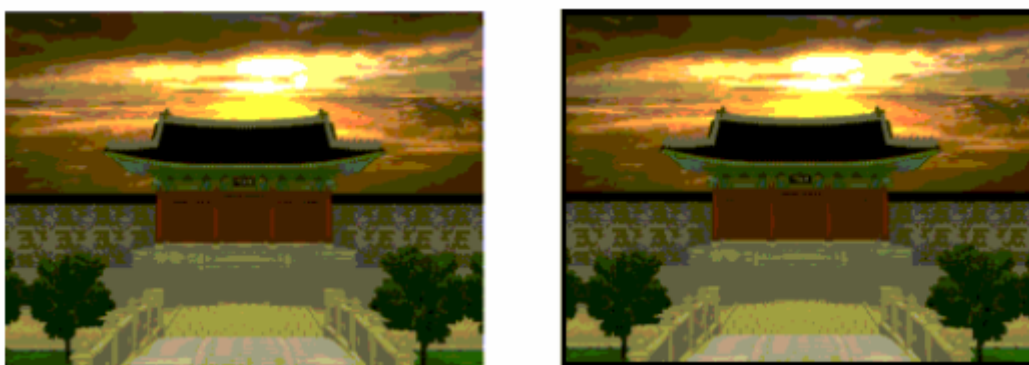
Tất cả các ảnh đều có đường biên. Khi đính một ảnh vào bản vẽ, đường biên của ảnh sẽ kế thừa các thuộc tính thiết lập hiện tại bao gồm: màu, lớp, kiểu đường và tỷ lệ của kiểu đường. Nếu định dạng ảnh là bitonal thì màu của ảnh và màu của đường biên là như nhau. Cũng như các đối tượng khác của AutoCAD, có thể hiệu chỉnh ảnh và các thuộc tính của đường biên như sau:

- Hiện thị hoặc dấu đường biên
- Hiệu chỉnh lớp của ảnh, màu và kiểu đường của đường biên
- Thay đổi vị trí của ảnh

- Đặt tỷ lệ, góc quay và thay đổi chiều cao, chiều rộng của ảnh
- Bật tắt chế độ hiển thị của ảnh
- Thay đổi độ trong suốt của ảnh
- Thay đổi độ sáng tối, tương phản và độ mờ
- Thay đổi chất lượng và tốc độ hiển thị ảnh

### 1.4.1. Hiển thị và che giấu đường biên ảnh

Giấu đường biên sẽ đảm bảo rằng ảnh không thể bị dịch chuyển hay hiệu chỉnh một cách ngẫu nhiên và đường biên sẽ không được in ra hoặc hiển thị. Khi đường biên bị giấu đi, phần ảnh cắt ra vẫn được biểu diễn theo giới hạn do đường biên xác định, chỉ có đường biên là chịu ảnh hưởng. Khi chọn hiển thị hay không hiển thị đường biên của một ảnh tức là hiển thị hoặc không hiển thị đường biên của toàn bộ các ảnh trong bản vẽ.



Để hiển thị hoặc giấu đường biên ảnh, sử dụng thuộc tính ClippingEnabled.

---

**CHÚ Ý** Thuộc tính này chỉ tác động lên đường biên của ảnh. Để nhìn thấy sự thay đổi trên ảnh khi bật tắt thuộc tính này cần quan sát kỹ đường viền nhỏ xung quanh ảnh.

---

### 1.4.2. Thay đổi lớp của ảnh, màu và kiểu đường của đường biên

Có thể thay đổi màu và kiểu đường của đường biên ảnh cũng như lớp của ảnh bằng cách sử dụng các thuộc tính sau:

Layer	Xác định layer của ảnh
Color	Xác định màu của đường biên
Linetype	Xác định kiểu đường của đường biên

### 1.4.3. Thay đổi tỷ lệ, góc quay, vị trí, bề rộng và chiều cao của ảnh

Có thể thay đổi tỷ lệ, góc quay, vị trí, bề rộng và chiều cao của ảnh bằng cách sử dụng các phương thức và thuộc tính sau:

ScaleEntity	Đặt tỷ lệ ảnh
Rotate	Quay ảnh
Origin	Xác định vị trí của ảnh

Width	Xác định bề rộng của ảnh theo pixel
Height	Xác định chiều cao của ảnh theo pixel
ImageWidth	Xác định bề rộng của ảnh theo đơn vị trong cơ sở dữ liệu
ImageHeight	Xác định bề rộng của ảnh theo đơn vị trong cơ sở dữ liệu
ShowRotation	Quyết định ảnh raster được hiển thị quay hay không

#### 1.4.4. Thay đổi chế độ nhìn thấy của ảnh

Chế độ nhìn thấy của ảnh tác động tới tốc độ vẽ lại bằng việc không thể hiện ảnh trong bản vẽ hiện tại. Các ảnh bị giấu đi sẽ không được thể hiện hay in ra mà chỉ thể hiện đường biên của ảnh. Gán giá trị `FALSE` cho thuộc tính `ImageVisibility` để không hiển thị ảnh và gán `TRUE` để hiển thị ảnh trở lại.

#### 1.4.5. Thay đổi màu của ảnh Bitonal và độ trong suốt

Ảnh Raster dạng Bitonal là những ảnh chỉ gồm màu tiền cảnh và màu nền. Khi đính một ảnh dạng này vào bản vẽ, các điểm tiền cảnh trong ảnh đó sẽ kế thừa toàn bộ các thiết lập về màu sắc của lớp hiện tại. Bên cạnh những thay đổi có thể thực hiện với các ảnh đính kèm, có thể thay đổi các ảnh Bitonal bằng cách chuyển màu chữ và bật tắt chế độ trong suốt của màu nền.

---

**CHÚ Ý** Ảnh Bitonal và đường biên của nó luôn có cùng màu.

---

Để thay đổi màu tiền cảnh của ảnh Bitonal, sử dụng thuộc tính `Color`.

Để bật và tắt chế độ trong suốt, sử dụng thuộc tính `Transparency`.

#### 1.4.6. Điều chỉnh độ sáng tối, tương phản và độ mờ

Có thể điều chỉnh độ sáng tối, tương phản và độ mờ trong AutoCAD để biểu diễn ảnh và in ảnh mà không làm ảnh hưởng đến tệp ảnh Raster ban đầu. Điều chỉnh độ sáng tối để thay đổi độ sáng của ảnh. Điều chỉnh độ tương phản để làm cho các ảnh có chất lượng kém dễ xem hơn. Điều chỉnh độ mờ để các hình học biểu diễn theo dạng Vector dễ nhìn hơn so với ảnh và tạo các hiệu ứng vệt nước khi in ra. Sử dụng các thuộc tính sau để thực hiện các điều chỉnh đó:

Brightness	Xác định mức sáng tối của ảnh
Contrast	Xác định mức độ tương phản của ảnh
Fade	Xác định mức độ mờ của ảnh

### 1.5. Cắt xén ảnh

Ảnh có thể được biểu diễn hoặc in ra chỉ một phần cần thiết bằng cách cắt riêng phần đó. Đường biên của phần cắt riêng đó là một hình đa giác phẳng hoặc là hình chữ nhật với các đỉnh của chúng phải nằm bên trong khung của ảnh. Từ một ảnh có thể cắt ra nhiều ảnh nhỏ bằng cách dùng các đường biên cắt khác nhau.





Cắt ảnh bằng đường bao cắt hình chữ nhật      Ảnh nhận được sau khi cắt

### Các bước thực hiện cắt ảnh

- 1 Bật đường biên của ảnh bằng thuộc tính `ClippingEnabled`.
- 2 Xác định đường biên cắt và thực hiện cắt với phương thức `ClipBoundary`. Phương thức này yêu cầu một tham số đầu vào là mảng kiểu `variant`<sup>1</sup> của các tọa độ trong hệ tọa độ 2D WCS để xác định đường biên cắt ảnh.

#### 1.5.1. Thay đổi đường biên cắt

Để đổi một đường biên cắt có sẵn chỉ cần lặp lại các bước nêu trên, khi đó đường biên cũ sẽ bị xoá đi và đường biên mới sẽ thay thế đường biên cũ.

#### 1.5.2. Hiện thị và không hiện thị đường biên cắt

Ảnh được cắt ra từ ảnh ban đầu có thể được hiện thị cùng với đường biên cắt; hoặc giấu đi đường bao cắt và hiện thị các đường biên của ảnh gốc. Để giấu đi đường biên cắt và hiện thị ảnh gốc, thực hiện gán thuộc tính `ClippingEnabled` là `FALSE`. Và ngược lại để biểu diễn ảnh được cắt thì gán thuộc tính trên thành `TRUE`.

#### Cắt đường biên của ảnh Raster

Ví dụ này sẽ chèn một ảnh raster vào không gian mô hình. Sau đó sẽ thực hiện cắt ảnh bằng một đường biên cắt. Ví dụ sử dụng tệp ảnh có tên `downtown.jpg` lưu trong thư mục `Sample` của chương trình. Nếu không tìm thấy tệp này trong thư mục nói trên thì có thể tệp đó được lưu trong thư mục khác hoặc dùng một tệp ảnh khác bằng cách gán tên và đường dẫn của tệp đó cho biến `imageName`.

```
Sub Ch10_ClippingRasterBoundary()  
    Dim insertionPoint(0 To 2) As Double  
    Dim scaleFactor As Double  
    Dim rotationAngle As Double  
    Dim imageName As String  
    Dim rasterObj As AcadRasterImage  
    imageName = "C:\AutoCAD\sample\downtown.jpg"  
    insertionPoint(0) = 5  
    insertionPoint(1) = 5  
    insertionPoint(2) = 0
```

<sup>1</sup> Kiểu `variant`: kiểu biến thể là kiểu có thể chứa mọi loại dữ liệu



```

scalefactor = 2
rotationAngle = 0
On Error GoTo ERRORHANDLER
' Tạo một ảnh raster trong không gian mô hình
Set rasterObj = ThisDrawing.ModelSpace.AddRaster _
    (imageName, insertionPoint, scalefactor, rotationAngle)
ZoomAll
' Xây dựng đường biên cắt bằng một mảng điểm
Dim clipPoints(0 To 9) As Double
clipPoints(0) = 6: clipPoints(1) = 6.75
clipPoints(2) = 7: clipPoints(3) = 6
clipPoints(4) = 6: clipPoints(5) = 5
clipPoints(6) = 5: clipPoints(7) = 6
clipPoints(8) = 6: clipPoints(9) = 6.75
' Cắt ảnh
rasterObj.ClipBoundary clipPoints
' Cho phép thể hiện đường biên cắt
rasterObj.ClippingEnabled = True
ThisDrawing.Regen acActiveViewport
Exit Sub
ERRORHANDLER:
MsgBox Err.Description
End Sub

```

## 2. Sử dụng khối và thuộc tính

AutoCAD hỗ trợ một số đặc tính giúp quản lý các đối tượng trong bản vẽ. Với các khối, có thể tổ chức và thao tác với nhiều đối tượng như một đối tượng. Các thuộc tính liên kết với các mục thông tin của khối trong bản vẽ, ví dụ: số lượng và giá cả.

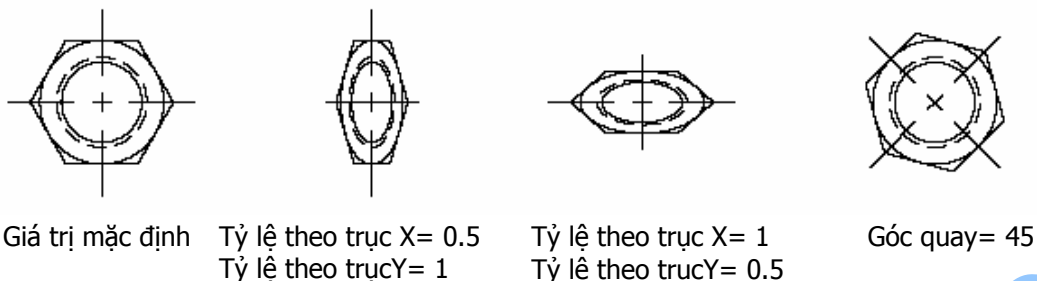
Sử dụng các tham chiếu ngoài của AutoCAD để đính kèm hoặc ghi đè toàn bộ các bản vẽ khác lên bản vẽ hiện tại. Khi mở bản vẽ hiện tại thì toàn bộ thay đổi trên những bản vẽ mà nó tham chiếu đến sẽ được cập nhật.

### 2.1. Làm việc với khối

Một khối là tập hợp của nhiều đối tượng có thể liên kết với nhau để tạo thành một đối tượng độc lập hoặc một khối tham chiếu. Các khối tham chiếu có thể được chèn, phóng đại và xoay trong bản vẽ. Các khối tham chiếu cũng có thể được tách thành các đối tượng thành phần để sửa đổi và định nghĩa lại khối. AutoCAD sẽ cập nhật sự thay đổi đó khi sử dụng khối đó trong những lần sau dựa vào định nghĩa của khối. Các khối giúp tổ chức hợp lý hơn quá trình vẽ. Ví dụ các khối có thể sử dụng trong các trường hợp sau:

- Xây dựng một thư viện chuẩn của các ký hiệu, cấu kiện, các bộ phận chuẩn thường dùng để có thể chèn cùng một khối tham chiếu nhiều lần thay vì tạo lại các thành phần của chúng mỗi khi sử dụng.
- Sửa lại bản vẽ một cách hiệu quả bằng cách chèn thêm, định vị lại và sao chép các khối như các cấu kiện hơn là thực hiện trên các đối tượng hình học riêng lẻ.
- Tiết kiệm được bộ nhớ bằng cách lưu trữ tất cả các tham chiếu vào một khối và được lưu trong cơ sở dữ liệu bản vẽ bằng tên của khối.

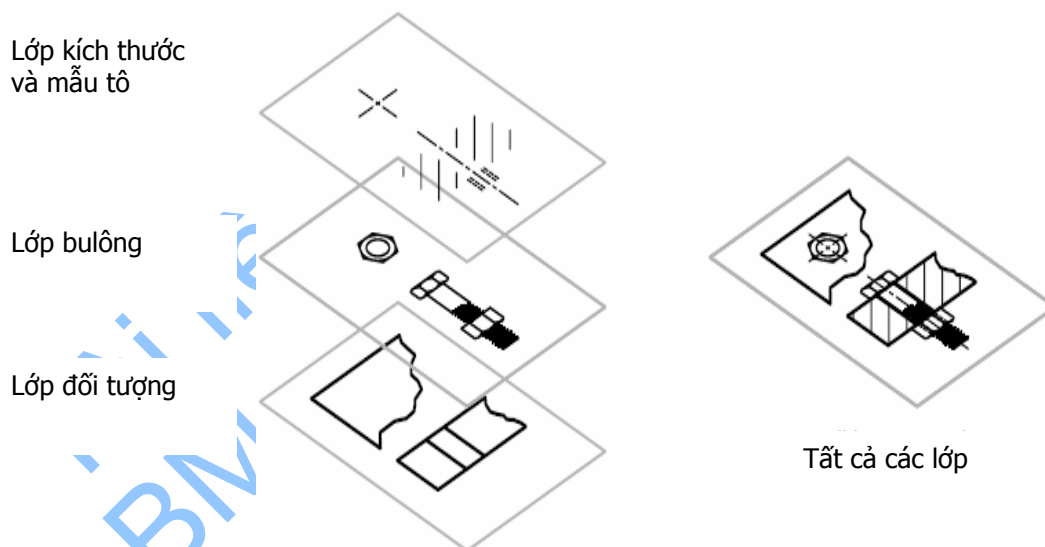
Mỗi khi chèn một khối vào bản vẽ là một lần tạo tham chiếu đến khối, khi đó cần đặt hệ số phóng đại và góc quay cho khối được chèn. Khối đó có thể được phóng đại với tỷ lệ khác nhau theo các hướng khác nhau bằng cách nhập các giá trị phóng đại theo các trục tọa độ (X,Y,Z).



Các khối giúp tổ chức các thao tác vẽ một cách hệ thống nên có thể thiết lập, thiết kế lại và sắp xếp các đối tượng trong bản vẽ cũng như các thông tin liên kết với chúng.

### 2.1.1. Làm việc với lớp, màu sắc, và kiểu đường

Khối có thể được định nghĩa từ các đối tượng được vẽ ngay từ ban đầu trên các lớp khác nhau cũng như với các màu và kiểu đường khác nhau. Các khối có thể giữ nguyên các thuộc tính này của đối tượng. Do đó, mỗi khi chèn khối thì mỗi đối tượng trong khối đó sẽ được vẽ bởi lớp, màu sắc và kiểu đường ban đầu của chúng.

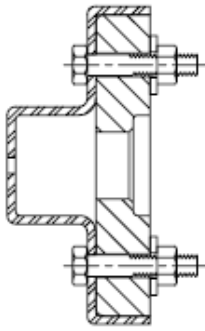


Khi một tham chiếu khối chứa các đối tượng được vẽ từ lớp 0 (với màu và kiểu đường được gán là BYLAYER) thì nó sẽ được đặt vào lớp hiện hành và nhận màu và kiểu đường của lớp hiện hành. Các thuộc tính của lớp hiện hành sẽ thay thế màu sắc và kiểu đường đã được gán cho khối tham chiếu đó.

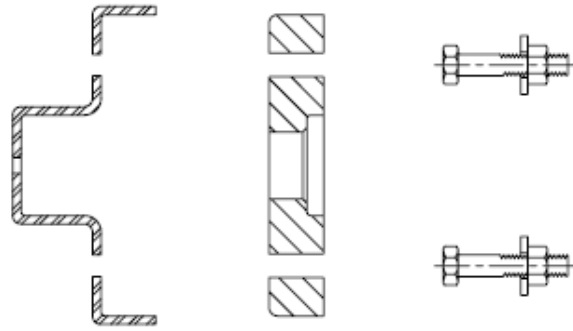
Khi một tham chiếu khối chứa các đối tượng mà màu và kiểu đường được gán là BYBLOCK thì sẽ được vẽ với màu và kiểu đường đã thiết lập khi chèn khối. Nếu màu và kiểu đường chưa được gán thì tham chiếu khối sẽ nhận màu và kiểu đường của lớp.

### 2.1.2. Các khối lồng nhau

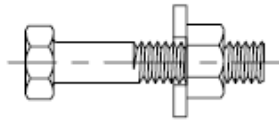
Các khối có thể chứa (lồng) các tham chiếu khối khác. Ví dụ, chèn một bản vẽ lắp ráp cơ khí bao gồm vỏ máy, giá đỡ và các chốt liên kết; mỗi chốt liên kết tạo thành bởi bulông, vòng đệm và đai ốc. Điều hạn chế duy nhất của các khối lồng là không thể chèn hoặc tạo khối tham chiếu đến chính nó.



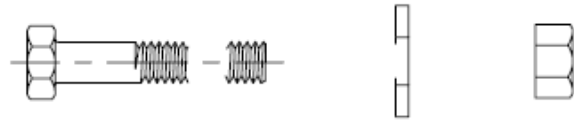
Khối hoàn chỉnh



Các khối cấu thành khối hoàn chỉnh



Khối liên kết bu lông



Các khối cấu thành khối bu lông

### 2.1.3. Định nghĩa khối

Sử dụng phương thức `Add` để tạo khối mới. Phương thức này cần 2 thông số đầu vào gồm: vị trí trên bản vẽ để tạo khối và tên của khối.

Sau khi tạo khối, có thể thêm các đối tượng hình học hoặc các khối khác vào khối mới được tạo. Có thể chèn khối vào bản vẽ. Khối được chèn đó là một đối tượng gọi là tham chiếu khối.

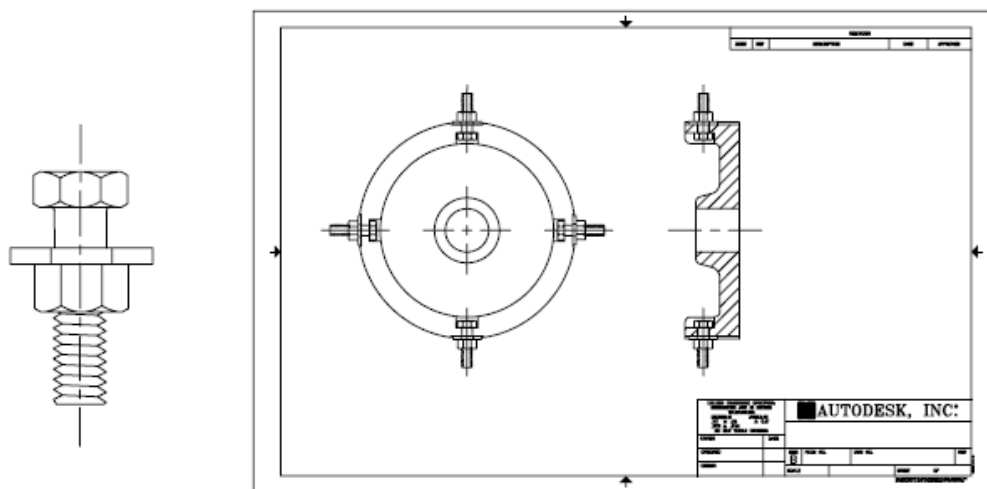
Khối cũng có thể được tạo bằng cách sử dụng phương thức `wblock` để nhóm các đối tượng thành một tệp bản vẽ độc lập. Tệp bản vẽ sau đó có thể được sử dụng như là một định nghĩa khối trong các bản vẽ khác. AutoCAD coi bản vẽ bất kỳ chèn vào bản vẽ khác là một khối.

### 2.1.4. Chèn khối

Có thể chèn khối hoặc toàn bộ bản vẽ nào đó vào bản vẽ hiện tại bằng phương thức `InsertBlock`. Phương thức này yêu cầu 6 thông số đầu vào gồm: điểm chèn, tên của khối hoặc tên bản vẽ được chèn, tỷ lệ phóng đại theo trục X, trục Y, trục Z và góc quay.

Khi chèn một bản vẽ vào một bản vẽ khác thì bản vẽ đó sẽ được coi như một khối tham chiếu. Mỗi lần chèn theo trình tự sẽ tham chiếu đến định nghĩa khối (bao gồm mô tả hình học của khối) với các thông số về vị trí, tỷ lệ phóng đại và góc quay khác nhau sẽ được mô tả trong minh họa dưới đây. Nếu bản vẽ bị thay đổi sau khi được chèn vào bản vẽ khác thì sự thay đổi đó sẽ không được cập nhật trên khối đã chèn. Để sự thay đổi cập nhật trên khối chèn cần dỡ chúng và chèn lại vào bản vẽ. Cách này được thực hiện bằng phương thức `InsertBlock`.

Nếu chèn một bản vẽ như một khối thì tên tệp sẽ được dùng là tên của khối một cách tự động. Có thể thay đổi tên khối bằng cách sử dụng thuộc tính `Name` mỗi khi tạo khối.



Bu lông

Các khối tham chiếu bu lông

Theo mặc định, AutoCAD sẽ sử dụng điểm có tọa độ (0, 0, 0) làm điểm cơ sở cho bản vẽ được chèn vào. Ta cũng có thể thay đổi điểm cơ sở của bản vẽ bằng cách mở bản vẽ gốc và sử dụng phương thức `SetVariable` để xác định lại điểm chèn mới cho biến hệ thống `INSBASE`. AutoCAD sẽ sử dụng điểm cơ sở mới cho các lần chèn sau của bản vẽ.

Nếu bản vẽ sử dụng để chèn có chứa cả đối tượng PaperSpace thì chúng sẽ không nằm trong định nghĩa khối của bản vẽ hiện hành. Để sử dụng các đối tượng PaperSpace trong các bản vẽ khác thì thực hiện bằng cách mở bản vẽ gốc và sử dụng phương thức `Add` để định nghĩa các đối tượng PaperSpace như là một khối. Và sau đó, ta có thể chèn bản vẽ đó vào trong bản vẽ khác, kể cả trong không gian in và không gian mô hình.

Không thể dùng phép lặp để tìm ra các đối tượng gốc cấu thành nên khối hiện tại. Tuy nhiên, có thể lặp qua định nghĩa của các khối ban đầu hoặc bung khối tham chiếu thành các khối gốc cấu thành nên nó.

Ngoài ra, có thể chèn một loạt các khối sử dụng phương thức `AddMInsertBlock`. Phương thức này không chèn một khối đơn lẻ vào trong bản vẽ giống như phương thức `InsertBlock` mà chèn một loạt các khối đã được chỉ định. Phương thức này trả về đối tượng kiểu `MInsertBlock`.

### Định nghĩa khối và chèn khối vào bản vẽ

Ví dụ này sẽ định nghĩa một khối và thêm một đường tròn. Sau đó sẽ chèn khối này vào bản vẽ như một tham chiếu khối.

```
Sub Ch10_InsertingABlock()
    ' Định nghĩa khối
    Dim blockObj As AcadBlock
    Dim insertionPnt(0 To 2) As Double
    insertionPnt(0) = 0
    insertionPnt(1) = 0
```

```

insertionPnt(2) = 0
Set blockObj = ThisDrawing.Blocks.Add _
(insertionPnt, "CircleBlock")
' Thêm một đường tròn cho khối
Dim circleObj As AcadCircle
Dim center(0 To 2) As Double
Dim radius As Double
center(0) = 0
center(1) = 0
center(2) = 0
radius = 1
Set circleObj = blockObj.AddCircle(center, radius)
' Chèn khối
Dim blockRefObj As AcadBlockReference
insertionPnt(0) = 2
insertionPnt(1) = 2
insertionPnt(2) = 0
Set blockRefObj = ThisDrawing.ModelSpace.InsertBlock _
(insertionPnt, "CircleBlock", 1#, 1#, 1#, 0)
ZoomAll
MsgBox "The circle belongs to " & blockRefObj.ObjectName
End Sub

```

---

**CHÚ Ý** Sau mỗi lần chèn, hệ trục tọa độ WCS của tệp bên ngoài sẽ song song với hệ tọa độ UCS trong bản vẽ hiện hành. Vì vậy, một khối được chèn từ một tệp khác sẽ được chèn vào bản vẽ theo bất cứ hướng nào trong không gian bằng cách thiết lập hệ trục tọa độ UCS trước khi chèn khối.

---

### 2.1.5. Bung khối tham chiếu

Sử dụng phương thức Explode để bung một khối tham chiếu, sau đó có thể hiệu chỉnh, thêm hoặc xóa đối tượng tạo thành khối.

#### Biểu diễn kết quả bung một khối tham chiếu

Ví dụ này sẽ định nghĩa một khối và thêm vào một đường tròn. Sau đó sẽ chèn khối này vào bản vẽ như một tham chiếu khối. Khối đó sẽ bị phá và đối tượng nhận được sẽ được biểu diễn cùng với kiểu đối tượng của nó.

```

Sub Ch10_ExplodingABlock()
' Định nghĩa khối
Dim blockObj As AcadBlock
Dim insertionPnt(0 To 2) As Double
insertionPnt(0) = 0
insertionPnt(1) = 0
insertionPnt(2) = 0
Set blockObj = ThisDrawing.Blocks.Add(insertionPnt,
"CircleBlock")
' Thêm đường tròn vào khối
Dim circleObj As AcadCircle
Dim center(0 To 2) As Double
Dim radius As Double
center(0) = 0
center(1) = 0
center(2) = 0
radius = 1
Set circleObj = blockObj.AddCircle(center, radius)
' Chèn khối

```

```

Dim blockRefObj As AcadBlockReference
insertionPnt(0) = 2
insertionPnt(1) = 2
insertionPnt(2) = 0
Set blockRefObj = ThisDrawing.ModelSpace.InsertBlock _
    (insertionPnt, "CircleBlock", 1#, 1#, 1#, 0)
ZoomAll
MsgBox "The circle belongs to " & blockRefObj.ObjectName
' Phá khối
Dim explodedObjects As Variant
explodedObjects = blockRefObj.Explode
' Vòng lặp qua tất cả các đối tượng sau khi phá khối
Dim I As Integer
For I = 0 To UBound(explodedObjects)
    explodedObjects(I).Color = acRed
    explodedObjects(I).Update
    MsgBox "Exploded Object " & I & ": "&
        explodedObjects(I).ObjectName
    explodedObjects(I).Color = acByLayer
    explodedObjects(I).Update
Next
End Sub

```

### 2.1.6. Định nghĩa lại khối

Sử dụng bất cứ phương thức hay thuộc tính nào của đối tượng Block để định nghĩa lại khối.

Khi định nghĩa lại khối, tất cả các tham chiếu đến khối đó trong bản vẽ đều được cập nhật ngay theo định nghĩa mới của khối.

Việc định nghĩa lại ảnh hưởng đến việc chèn khối trước và sau đó. Các thuộc tính là hằng số sẽ bị mất đi và được thay bằng hằng số mới. Các thuộc tính là biến số sẽ giữ nguyên không đổi mặc dù khối mới định nghĩa không kèm thuộc tính nào.

#### Định nghĩa lại đối tượng trong định nghĩa khối

Ví dụ này sẽ định nghĩa một khối và thêm một đường tròn vào định nghĩa khối. Sau đó sẽ chèn khối này vào bản vẽ như một tham chiếu. Thực hiện thay đổi đường tròn trong định nghĩa và cập nhật lại cho khối, khi đó trong bản vẽ khối sẽ tự động cập nhật thay đổi đó.

```

Sub Ch10_RedefiningABlock()
    ' Định nghĩa khối
    Dim blockObj As AcadBlock
    Dim insertionPnt(0 To 2) As Double
    insertionPnt(0) = 0
    insertionPnt(1) = 0
    insertionPnt(2) = 0
    Set blockObj = ThisDrawing.Blocks.Add(insertionPnt, _
        "CircleBlock")
    ' Thêm đường tròn vào khối
    Dim circleObj As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 0
    center(1) = 0
    center(2) = 0

```

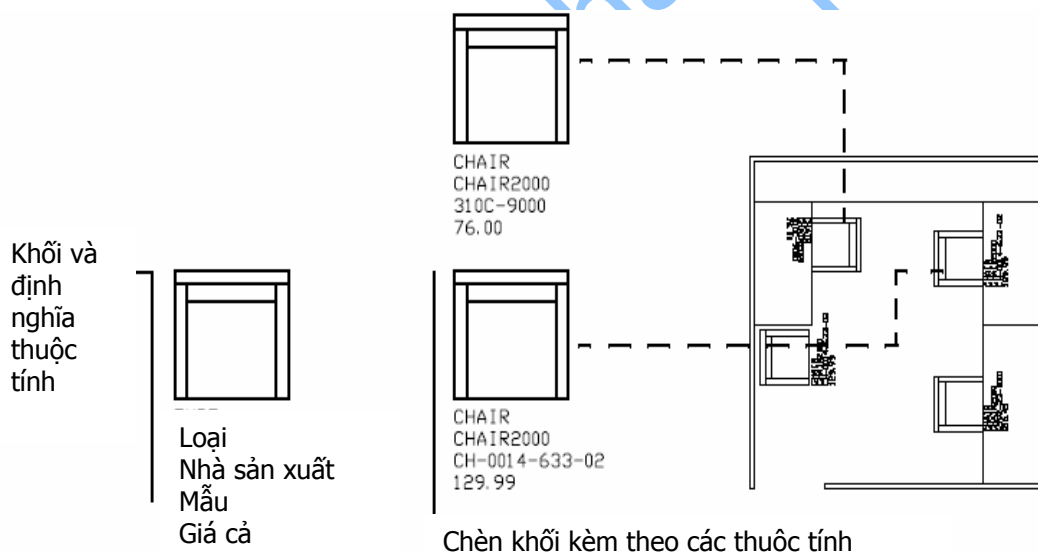
```

radius = 1
Set circleObj = blockObj.AddCircle(center, radius)
' Chèn khối
Dim blockRefObj As AcadBlockReference
insertionPnt(0) = 2
insertionPnt(1) = 2
insertionPnt(2) = 0
Set blockRefObj = ThisDrawing.ModelSpace.InsertBlock _
(insertionPnt, "CircleBlock", 1#, 1#, 1#, 0)
ZoomAll
' Thay đổi bán kính đường tròn và cập nhật lại khối tham
' chiếu
circleObj.radius = 3
blockRefObj.Update
End Sub

```

## 2.2. Làm việc với thuộc tính

Tham chiếu thuộc tính cung cấp các nhãn tương tác hoặc các thẻ để đính văn bản kèm theo khối. Ví dụ trong bản vẽ bố trí đồ đạc của một căn phòng dưới đây, các dữ liệu có thể gắn với mỗi vật dụng và chúng có thể là chỉ số của bộ phận, giá, chú thích và chủ sở hữu.



Thông tin về các tham chiếu thuộc tính của bản vẽ có thể được trích ra và sử dụng trong các bản tính hoặc cơ sở dữ liệu để tạo các mục như danh mục các bộ phận hay bảng thống kê vật liệu. Mỗi khối có thể được liên kết với nhiều thuộc tính miễn là mỗi thuộc tính có một thẻ khác nhau.

Có thể định nghĩa các thuộc tính là hằng số, chúng có cùng giá trị với mọi khối được chèn trong bản vẽ nên AutoCAD không nhắc nhập giá trị này khi chèn.

Các thuộc tính có thể để chế độ không nhìn thấy tức là nó sẽ không được thể hiện hay in ra. Tuy nhiên, thông tin của chúng vẫn được lưu trong bản vẽ.



### 2.2.1. Tạo định nghĩa thuộc tính và tham chiếu thuộc tính

Để tạo một thuộc tính tham chiếu, cần tạo định nghĩa thuộc tính cho một khối bằng cách sử dụng phương thức `AddAttribute`. Phương thức này cần 6 thông số đầu vào gồm: chiều cao của chữ ghi thuộc tính, chế độ của thuộc tính, dòng nhắc, điểm chèn, giá trị của thẻ và giá trị thuộc tính mặc định. Chế độ của thuộc tính không bắt buộc phải nhập và có thể tùy chọn một trong 5 giá trị sau:

`acAttributeModeNormal`

Chỉ ra rằng chế độ hiện hành của mỗi thuộc tính được duy trì.

`acAttributeModeInvisible`

Chỉ ra rằng giá trị thuộc tính không xuất hiện khi chèn khối. Lệnh `ATTDISP` sẽ vô hiệu hoá chế độ `Invisible`.

`acAttributeModeConstant`

Gán giá trị cố định cho thuộc tính khi chèn khối.

`acAttributeModeVerify`

Nhắc nhở kiểm tra giá trị thuộc tính là đúng khi chèn khối.

`acAttributeModePreset`

Gán giá trị mặc định cho thuộc tính khi chèn khối có chứa giá trị hiện tại. Giá trị không thể sửa khi để ở chế độ này.

Các tùy chọn trên có thể không cần nhập hoặc nhập dưới dạng một tổ hợp bất kỳ hay tổ hợp của tất cả các giá trị. Lập tổ hợp của các tùy chọn bằng cách cộng các giá trị đó lại với nhau. Ví dụ có thể nhập `acAttributeModeInvisible + acAttributeModeConstant`.

Dòng nhắc xuất hiện mỗi khi chèn khối chứa thuộc tính. Giá trị mặc định của chuỗi này chính là giá trị của thẻ. Nhập giá trị chế độ thuộc tính là `acAttributeModeConstant` để làm vô hiệu dòng nhắc.

Giá trị của thẻ dùng để nhận biết các thuộc tính. Giá trị này là một chuỗi tổ hợp của bất cứ ký tự nào trừ khoảng trống hay dấu chấm than. AutoCAD sẽ tự động chuyển các chữ cái viết thường thành viết hoa.

Khi thuộc tính được định nghĩa cho khối, dù khối được chèn ở đâu bằng phương thức `InsertBlock` đều có thể chỉ định giá trị khác nhau cho thuộc tính.

Một định nghĩa thuộc tính được liên kết với khối tạo ra nó. Các định nghĩa thuộc tính được tạo ra trong không gian mô hình hay không gian in không được đính kèm vào vị trí hợp lệ của một khối bất kỳ nào.

#### Định nghĩa một thuộc tính

Ví dụ này sẽ tạo một khối và sau đó thêm thuộc tính cho khối. Khối sau đó sẽ được chèn vào bản vẽ.

```
Sub Ch10_CreatingAnAttribute ()  
    ' Định nghĩa khối  
    Dim blockObj As AcadBlock  
    Dim insertionPnt (0 To 2) As Double  
    insertionPnt (0) = 0
```

```

insertionPnt(1) = 0
insertionPnt(2) = 0
Set blockObj = ThisDrawing.Blocks.Add(insertionPnt, _
    "BlockWithAttribute")
' Thêm thuộc tính cho khối
Dim attributeObj As AcadAttribute
Dim height As Double
Dim mode As Long
Dim prompt As String
Dim insertionPoint(0 To 2) As Double
Dim tag As String
Dim value As String
height = 1
mode = acAttributeModeVerify
prompt = "New Prompt"
insertionPoint(0) = 5
insertionPoint(1) = 5
insertionPoint(2) = 0
tag = "New Tag"
value = "New Value"
Set attributeObj = blockObj.AddAttribute(height, mode, _
    prompt, insertionPoint, tag, value)
' Chèn khối và tạo tham chiếu khối và tham chiếu thuộc tính
Dim blockRefObj As AcadBlockReference
insertionPnt(0) = 2
insertionPnt(1) = 2
insertionPnt(2) = 0
Set blockRefObj = ThisDrawing.ModelSpace.InsertBlock _
    (insertionPnt, "BlockWithAttribute", 1#, 1#, 1#, 0)
End Sub

```

### 2.2.2. Sửa đổi định nghĩa thuộc tính

Sử dụng thuộc tính và phương thức của đối tượng `Attribute` để sửa đổi thuộc tính. Một số thuộc tính của đối tượng `Attribute` được liệt kê dưới đây:

<code>Alignment</code>	Quy định chế độ căn lề ngang và dọc của thuộc tính
<code>Backward</code>	Quy định hướng chữ của thuộc tính
<code>FieldLength</code>	Quy định chiều dài chữ của thuộc tính
<code>Height</code>	Quy định chiều cao chữ của thuộc tính
<code>InsertionPoint</code>	Quy định điểm chèn chữ của thuộc tính
<code>Mode</code>	Quy định chế độ của thuộc tính
<code>PromptString</code>	Quy định nội dung dòng nhắc của thuộc tính
<code>Rotation</code>	Quy định góc quay chữ của thuộc tính
<code>ScaleFactor</code>	Quy định tỷ lệ phóng đại chữ của thuộc tính
<code>TagString</code>	Quy định nội dung thẻ tên của thuộc tính

Một số phương thức có thể sử dụng để sửa các thuộc tính được liệt kê dưới đây:

<code>ArrayPolar</code>	Nhân bản dạng cực
<code>ArrayRectangular</code>	Nhân bản dạng chữ nhật

Copy	Sao chép thuộc tính
Erase	Xoá thuộc tính
Mirror	Lấy đối xứng thuộc tính
Move	Di chuyển thuộc tính
Rotate	Quay thuộc tính
ScaleEntity	Phóng đại thuộc tính

### Định nghĩa lại thuộc tính

Ví dụ này sẽ tạo một khối và sau đó thêm thuộc tính cho khối. Khối sau đó sẽ được chèn vào bản vẽ. Chữ thuộc tính sẽ được cập nhật lại để biểu diễn theo chiều ngược lại.

```

Sub Ch10_RedefiningAnAttribute()
    ' Định nghĩa khối
    Dim blockObj As AcadBlock
    Dim insertionPnt(0 To 2) As Double
    insertionPnt(0) = 0
    insertionPnt(1) = 0
    insertionPnt(2) = 0
    Set blockObj = ThisDrawing.Blocks.Add _
        (insertionPnt, "BlockWithAttribute")

    ' Thêm thuộc tính cho khối
    Dim attributeObj As AcadAttribute
    Dim height As Double
    Dim mode As Long
    Dim prompt As String
    Dim insertionPoint(0 To 2) As Double
    Dim tag As String
    Dim value As String
    height = 1
    mode = acAttributeModeVerify
    prompt = "New Prompt"
    insertionPoint(0) = 5
    insertionPoint(1) = 5
    insertionPoint(2) = 0
    tag = "New Tag"
    value = "New Value"
    Set attributeObj = blockObj.AddAttribute(height, mode, _
        prompt, insertionPoint, tag, value)

    ' Chèn khối và tạo tham chiếu khối và tham chiếu thuộc tính
    Dim blockRefObj As AcadBlockReference
    insertionPnt(0) = 2
    insertionPnt(1) = 2
    insertionPnt(2) = 0
    Set blockRefObj = ThisDrawing.ModelSpace.InsertBlock _
        (insertionPnt, "BlockWithAttribute", 1#, 1#, 1#, 0)

    ' Định nghĩa lại chữ thuộc tính để thể hiện chữ theo hướng
    ' ngược lại
    attributeObj.Backward = True
    attributeObj.Update
End Sub

```

### 2.2.3. Trích các thông tin của thuộc tính

Các thông tin thuộc tính trong bản vẽ có thể được lấy ra bằng phương thức `GetAttributes` và `GetConstantAttributes`. Phương thức `GetAttributes` trả về một mảng các tham chiếu thuộc tính đính kèm với khối cùng với các giá trị hiện hành của chúng. Phương thức `GetConstantAttributes` trả về một mảng các thuộc tính hằng số đính kèm với khối hoặc các tham chiếu ngoài. Các thuộc tính được trả về bằng phương thức này là các định nghĩa thuộc tính hằng số chứ không phải là các tham chiếu thuộc tính.

Không cần các tệp mẫu để trích thông tin thuộc tính và không có tệp thông tin thuộc tính nào được tạo ra.

Duyệt qua mảng các thuộc tính tham chiếu sử dụng thuộc tính `TagString` và `TextString` của tham chiếu thuộc tính để kiểm tra thông tin của thuộc tính.

Thuộc tính `TagString` biểu diễn từng thẻ riêng biệt của tham chiếu thuộc tính

Thuộc tính `TextString` chứa giá trị của tham chiếu thuộc tính.

#### Lấy thông tin của tham chiếu thuộc tính

Ví dụ này sẽ tạo một khối và sau đó thêm thuộc tính cho khối. Khối sau đó sẽ được chèn vào bản vẽ. Dữ liệu của thuộc tính sau đó sẽ được trả về và biểu diễn trong một hộp thoại thông báo. Dữ liệu đó sẽ được cập nhật cho tham chiếu khối và các dữ liệu thuộc tính sẽ được lấy và biểu diễn lại.

```
Sub Ch10_GettingAttributes()  
    ' Định nghĩa khối  
    Dim blockObj As AcadBlock  
    Dim insertionPnt(0 To 2) As Double  
    insertionPnt(0) = 0  
    insertionPnt(1) = 0  
    insertionPnt(2) = 0  
    Set blockObj = ThisDrawing.Blocks.Add _  
        (insertionPnt, "TESTBLOCK")  
  
    ' Định nghĩa thuộc tính  
    Dim attributeObj As AcadAttribute  
    Dim height As Double  
    Dim mode As Long  
    Dim prompt As String  
    Dim insertionPoint(0 To 2) As Double  
    Dim tag As String  
    Dim value As String  
    height = 1#  
    mode = acAttributeModeVerify  
    prompt = "Attribute Prompt"  
    insertionPoint(0) = 5  
    insertionPoint(1) = 5  
    insertionPoint(2) = 0  
    tag = "Attribute Tag"  
    value = "Attribute Value"  
  
    ' Tạo đối tượng thuộc tính cho khối  
    Set attributeObj = blockObj.AddAttribute _  
        (height, mode, prompt, _  
        insertionPoint, tag, value)
```

```

' Chèn khối
Dim blockRefObj As AcadBlockReference
insertionPnt(0) = 2
insertionPnt(1) = 2
insertionPnt(2) = 0
Set blockRefObj = ThisDrawing.ModelSpace.InsertBlock _
    (insertionPnt, "TESTBLOCK", 1, 1, 1, 0)
ZoomAll

' Lấy giá trị thuộc tính của khối tham chiếu
Dim varAttributes As Variant
varAttributes = blockRefObj.GetAttributes

' Chuyển thẻ thuộc tính và giá trị vào một chuỗi và
' thể hiện trong hộp thông báo
Dim strAttributes As String
strAttributes = ""
Dim I As Integer
For I = LBound(varAttributes) To UBound(varAttributes)
    strAttributes = strAttributes + " Tag: " + _
        varAttributes(I).TagString + vbCrLf + _
        " Value: " + varAttributes(I).textString
Next
MsgBox "The attributes for blockReference " + _
    blockRefObj.Name & " are: " & vbCrLf _
    & strAttributes

' Thay đổi giá trị của thuộc tính
' Chú ý: không có SetAttributes. Khi đã có mảng kiểu variant
' và có các đối tượng.
' Thay đổi chúng sẽ gây thay đổi các đối tượng trong bản vẽ.
varAttributes(0).textString = "NEW VALUE!"

' Lấy lại thuộc tính
Dim newvarAttributes As Variant
newvarAttributes = blockRefObj.GetAttributes

' Biểu diễn lại các thẻ và các giá trị
strAttributes = ""
For I = LBound(varAttributes) To UBound(varAttributes)
    strAttributes = strAttributes + " Tag: " + _
        newvarAttributes(I).TagString + vbCrLf + _
        " Value: " + newvarAttributes(I).textString
Next
MsgBox "The attributes for blockReference " & _
    blockRefObj.Name & " are: " & vbCrLf _
    & strAttributes
End Sub

```

### 3. Sử dụng tham chiếu ngoài

Một tham chiếu ngoài (xref<sup>1</sup>) là liên kết một bản vẽ khác với bản vẽ hiện tại. Khi chèn một bản vẽ như một khối thì khối này và tất cả các đối tượng hình học liên kết với nó sẽ được chứa trong cơ sở dữ liệu của bản vẽ hiện hành, nó sẽ không được cập nhật cho dù bản vẽ vẽ gốc thay đổi. Tuy nhiên khi chèn bản vẽ đó như một tham chiếu ngoài thì nó đó sẽ cập nhật mọi thay đổi của bản vẽ gốc. Do vậy một bản vẽ có chứa tham chiếu ngoài luôn thể hiện được mọi sự thay đổi gần nhất của bản vẽ được tham chiếu.

Giống như khối, tham chiếu ngoài được thể hiện trên bản vẽ hiện hành như một đối tượng độc lập. Tuy nhiên, nó làm tăng không đáng kể dung lượng của bản vẽ và không thể bị phá. Nhưng cũng như với khối, các tham chiếu ngoài đính kèm với bản vẽ có thể được lồng ghép lại với nhau.

#### 3.1. Cập nhật tham chiếu ngoài

Khi mở hoặc in một bản vẽ, AutoCAD sẽ cập nhật lại các tham chiếu ngoài để có được các thay đổi gần nhất của các bản vẽ được tham chiếu. Sau khi tạo sự thay đổi cho các bản vẽ tham chiếu và lưu lại thì những người dùng khác có thể truy cập bản vẽ này bằng cách tải lại các tham chiếu ngoài của họ.

#### 3.2. Đính kèm tham chiếu ngoài

Việc đính kèm một tham chiếu ngoài sẽ liên kết một bản vẽ cần tham chiếu với bản vẽ hiện hành. Khi một bản vẽ tham chiếu đến một bản vẽ khác, không như tham chiếu đến khối, AutoCAD chỉ gán định nghĩa của tham chiếu ngoài vào bản vẽ trong khi đó, cả khối và các đối tượng của khối đều được chứa trong bản vẽ đó. AutoCAD sẽ đọc bản vẽ được tham chiếu để quyết định sẽ hiển thị những gì trên bản vẽ hiện hành.

Nếu bản vẽ được tham chiếu bị thiếu hoặc hỏng thì dữ liệu của nó sẽ không được hiển thị trong bản vẽ hiện hành. Mỗi khi mở bản vẽ, AutoCAD sẽ tải toàn bộ các đối tượng đồ họa và phi đồ họa (như lớp, kiểu đường và kiểu chữ) từ bản vẽ được tham chiếu. Nếu `VISRETAIN` là 1 thì AutoCAD sẽ lưu bất kỳ thông tin nào được cập nhật của lớp trong tham chiếu ngoài vào bản vẽ hiện tại.

---

<sup>1</sup> Xref: external reference – là một loại khối (Block) trong AutoCAD nhưng có một vài khác biệt với khối thông thường, mà khác biệt lớn nhất là nội dung của nó sẽ được cập nhật lại khi bản vẽ gốc được thay đổi.

Trong AutoCAD có hai kiểu tham chiếu ngoài (xref): *Attaching Xref* và *Overlying Xref*. Thực chất của Xref là một bản vẽ được tham chiếu bởi một bản vẽ khác. Giả sử bản vẽ *C.DWG* tham chiếu đến hai bản vẽ: *A.DWG* (xref A) và *B.DWG* (xref B). Xref A có kiểu là *Attaching* và Xref B có kiểu là *Overlying*. Trong bản vẽ *C.DWG* thì Xref A và Xref B không có sự khác nhau. Khi bản vẽ *D.DWG* tham chiếu đến bản vẽ *C.DWG* thì ở đây sẽ xuất hiện sự khác nhau giữa Xref A và Xref B: chỉ có Xref A (kiểu *Attaching*) được hiển thị trong bản vẽ *D.DWG*.

Trong bản dịch này, do trong tiếng Việt chưa có từ chuyên môn thích hợp với hai kiểu tham chiếu ngoài trên cho nên chúng tôi tạm sử dụng từ *tham chiếu lập* cho kiểu *Attaching xref* và từ *tham chiếu một lần* cho kiểu *Overlying xref*.

Có thể đính kèm các tham chiếu ngoài với số lượng bao nhiêu tùy theo yêu cầu sử dụng và mỗi tham chiếu ngoài khi chèn cần có vị trí, tỷ lệ và góc quay riêng. Ngoài ra có thể điều khiển được các thuộc tính kiểu đường, lớp đã được định nghĩa trong tham chiếu ngoài. Sử dụng phương thức `AttachExternalReference` để đính kèm tham chiếu ngoài. Phương thức này yêu cầu thông số đầu vào bao gồm các thông tin về: tên của tệp bản vẽ được tham chiếu, tên của xref được sử dụng trong bản vẽ hiện hành, điểm chèn, tỷ lệ, và góc quay của Xref. Phương thức này sẽ trả về đối tượng `ExternalReference` mới được tạo.

### Đính kèm tham chiếu ngoài vào bản vẽ

Ví dụ này sẽ hiển thị toàn bộ các khối trong bản vẽ hiện tại trước và sau khi thêm tham chiếu ngoài. Trong ví dụ này sử dụng tệp `City map.dwg` tìm trong thư mục `sample`. Nếu không có hoặc tệp được lưu ở thư mục khác thì chỉ cần thêm tên và đường dẫn có thực cho biến `PathName` dưới đây:

```
Sub Ch10_AttachingExternalReference ()
    On Error GoTo ERRORHANDLER
    Dim InsertPoint(0 To 2) As Double
    Dim insertedBlock As AcadExternalReference
    Dim tempBlock As AcadBlock
    Dim msg As String, PathName As String
    ' Định nghĩa tham chiếu ngoài sẽ được chèn
    InsertPoint(0) = 1
    InsertPoint(1) = 1
    InsertPoint(2) = 0
    PathName = "c:/acad2000/sample/City map.dwg"
    ' Hiển thị thông tin về các khối trên bản vẽ
    GoSub ListBlocks
    ' Thêm tham chiếu ngoài cho bản vẽ
    Set insertedBlock = ThisDrawing.ModelSpace.
        AttachExternalReference(PathName, "XREF_IMAGE", _
        InsertPoint, 1, 1, 1, 0, False)
    ZoomAll
    ' Hiển thị thông tin của khối mới của bản vẽ
    GoSub ListBlocks
    Exit Sub
ListBlocks:
    msg = vbCrLf ' Reset message
    For Each tempBlock In ThisDrawing.Blocks
        msg = msg & tempBlock.Name & vbCrLf
    Next
    MsgBox "The current blocks in this drawing are: " & msg
    Return
ERRORHANDLER:
    MsgBox Err.Description
End Sub
```

### 3.2.2. Tham chiếu một lần

Tham chiếu một lần tương tự như tham chiếu lặp, trừ khi bản vẽ đã có tham chiếu lặp hay tham chiếu một lần từ trước. Bất cứ tham chiếu một lần nào gắn vào bản vẽ cũng sẽ bị bỏ qua và do đó không được hiển thị trong bản vẽ tham chiếu. Nói cách khác, tham chiếu một lần sẽ không được đọc vào.



Lưu ý là ta chỉ nên sử dụng tham chiếu một lần khi đối tượng hình học được tham chiếu không giúp ích cho người dùng khác khi họ tham chiếu đến bản vẽ của ta. Ví dụ khi cần thiết kế hệ thống dây điện trong nhà và cần tham khảo thiết kế sàn nhà. Nếu sử dụng tham chiếu một lần (tốt hơn là tham chiếu lặp) tới bản vẽ thiết kế sàn thì một người thiết kế khác không cần tham khảo thiết kế sàn có thể tham chiếu đến bản vẽ thiết kế hệ thống dây điện mà không phải kèm theo bản vẽ thiết kế sàn nhà.

Tham chiếu một lần được thiết kế để chia sẻ dữ liệu. Bằng cách đó có thể nhìn được bản vẽ liên hệ với các bản vẽ khác như thế nào. Tham chiếu một lần cũng làm giảm đi việc tham chiếu lồng vòng

Để tạo tham chiếu một lần, gán giá trị TRUE cho tham số `bOverlay` của phương thức `AttachExternalReference`.

### 3.3. Tách các tham chiếu ngoài

Có thể tách các tham chiếu ngoài ra khỏi bản vẽ hoàn toàn và cũng có thể tách riêng từng tham chiếu ngoài một. Gỡ bỏ tham chiếu ngoài sẽ xóa tất cả các ký hiệu phụ thuộc được liên kết với tham chiếu ngoài đó. Nếu tất cả các đối tượng của tham chiếu ngoài đều bị xóa khỏi bản vẽ thì AutoCAD sẽ loại bỏ định nghĩa tham chiếu ngoài khi mở bản vẽ lần sau.

Sử dụng phương thức `Detach` để gỡ bỏ tham chiếu ngoài và phương thức này không thể gỡ bỏ các tham chiếu ngoài được lồng vào.

#### Tách tham chiếu ngoài

Ví dụ này sẽ gắn một tham chiếu ngoài và sau đó tách nó khỏi bản vẽ. Trong ví dụ này sử dụng tệp `City map.dwg` tìm trong thư mục `sample`. Nếu không có hoặc tệp được lưu ở thư mục khác thì chỉ cần thêm tên và đường dẫn có thực cho biến `PathName` dưới đây.

```
Sub Ch10_DetachingExternalReference()  
    On Error GoTo ERRORHANDLER  
  
    ' Định nghĩa tham chiếu ngoài sẽ được chèn  
    Dim xrefHome As AcadBlock  
    Dim xrefInserted As AcadExternalReference  
    Dim insertionPnt(0 To 2) As Double  
    Dim PathName As String  
    insertionPnt(0) = 1  
    insertionPnt(1) = 1  
    insertionPnt(2) = 0  
    PathName = "c:/AutoCAD/sample/City map.dwg"  
  
    ' Thêm tham chiếu ngoài  
    Set xrefInserted = ThisDrawing.ModelSpace.  
        AttachExternalReference(PathName, "XREF_IMAGE", _  
            insertionPnt, 1, 1, 1, 0, False)  
    ZoomAll  
    MsgBox "The external reference is attached."  
  
    ' Gỡ bỏ tham chiếu ngoài  
    Dim name As String  
    name = xrefInserted.name  
    ThisDrawing.Blocks.Item(name).Detach
```

```

        MsgBox "The external reference is detached."
    Exit Sub
ERRORHANDLER:
    MsgBox Err.Description
End Sub

```

### 3.4. Tải lại tham chiếu ngoài

Nếu có sự thay đổi trong bản vẽ tham chiếu ngoài thì có thể cập nhật nội dung thay đổi đó trong bản vẽ tham chiếu đến nó bằng phương thức `Reload`. Khi tải lại bản vẽ, tham chiếu ngoài được lựa chọn sẽ được cập nhật trên bản vẽ tham chiếu nó. Nếu đã bỏ tham chiếu ngoài thì nó có thể được tải lại bất cứ lúc nào.

#### Tải lại tham chiếu ngoài

Ví dụ này sẽ gắn một tham chiếu ngoài và sau đó sẽ tải lại tham chiếu đó. Trong ví dụ này sử dụng tệp *City map.dwg* trong thư mục *sample*. Nếu không có hoặc tệp được lưu ở thư mục khác thì chỉ cần thêm tên và đường dẫn có thực cho biến `PathName` dưới đây.

```

Sub Ch10_ReloadingExternalReference ()
    On Error GoTo ERRORHANDLER
    ' Định nghĩa tham chiếu ngoài sẽ được chèn
    Dim xrefHome As AcadBlock
    Dim xrefInserted As AcadExternalReference
    Dim insertionPnt(0 To 2) As Double
    Dim PathName As String
    insertionPnt(0) = 1
    insertionPnt(1) = 1
    insertionPnt(2) = 0
    PathName = "c:/AutoCAD/sample/City map.dwg"

    ' Thêm tham chiếu ngoài
    Set xrefInserted = ThisDrawing.ModelSpace.
        AttachExternalReference(PathName, "XREF_IMAGE", _
            insertionPnt, 1, 1, 1, 0, False)
    ZoomAll
    MsgBox "The external reference is attached."

    ' Tải lại tham chiếu ngoài
    ThisDrawing.Blocks.Item(xrefInserted.name).Reload
    MsgBox "The external reference is reloaded."
    Exit Sub
ERRORHANDLER:
    MsgBox Err.Description
End Sub

```

### 3.5. Loại bỏ các tham chiếu ngoài

Sử dụng phương thức `Unload` để loại bỏ một tham chiếu ngoài. Khi loại bỏ một tham chiếu không còn được sử dụng trong bản vẽ hiện tại thì sự làm việc của AutoCAD sẽ được tăng cường nhờ việc không phải đọc và biểu diễn các thông tin hình học hay các ký hiệu không cần thiết. Các đối tượng hình học của tham chiếu ngoài và những đối tượng tham chiếu ngoài được ghép sẽ không được thể hiện trong bản vẽ hiện tại cho đến khi tham chiếu ngoài được tải lại.

## Loại bỏ tham chiếu ngoài

Ví dụ này sẽ gắn một tham chiếu ngoài và sau đó loại bỏ tham chiếu đó. Trong ví dụ này sử dụng tệp *City map.dwg* trong thư mục *sample*. Nếu không có hoặc tệp được lưu ở thư mục khác thì chỉ cần thêm tên và đường dẫn có thực cho biến *PathName* dưới đây.

```
Sub Ch10_UnloadingExternalReference()  
    On Error GoTo ERRORHANDLER  
    ' Định nghĩa tham chiếu ngoài sẽ được chèn  
    Dim xrefHome As AcadBlock  
    Dim xrefInserted As AcadExternalReference  
    Dim insertionPnt(0 To 2) As Double  
    Dim PathName As String  
    insertionPnt(0) = 1  
    insertionPnt(1) = 1  
    insertionPnt(2) = 0  
    PathName = "c:/AutoCAD/sample/City map.dwg"  
  
    ' Thêm tham chiếu ngoài  
    Set xrefInserted = ThisDrawing.ModelSpace.  
    AttachExternalReference(PathName, "XREF_IMAGE",  
    insertionPnt, 1, 1, 1, 0, False)  
    ZoomAll  
    MsgBox "The external reference is attached."  
  
    ' Bỏ tham chiếu ngoài  
    ThisDrawing.Blocks.Item(xrefInserted.name).Unload  
    MsgBox "The external reference is unloaded."  
    Exit Sub  
ERRORHANDLER:  
    MsgBox Err.Description  
End Sub
```

## 3.6. Ràng buộc tham chiếu ngoài

Ràng buộc một tham chiếu ngoài vào bản vẽ khi sử dụng phương thức *Bind* sẽ làm cho tham chiếu ngoài trở thành bộ phận cố định của bản vẽ và không còn là tham chiếu ngoài nữa. Các thông tin của tham chiếu ngoài sẽ trở thành khối thông thường. Khi bản vẽ được tham chiếu bị thay đổi thì tham chiếu ngoài được liên kết sẽ không được cập nhật. Quá trình này sẽ liên kết toàn bộ cơ sở dữ liệu trong bản vẽ bao gồm cả bảng ký hiệu của nó. Bảng ký hiệu là các đối tượng phi hình học, ví dụ như khối, kiểu kích thước, lớp, kiểu chữ. Liên kết với tham chiếu ngoài cho phép các đối tượng phi hình học trong tham chiếu ngoài được sử dụng trong bản vẽ hiện hành.

Phương thức *Bind* chỉ yêu cầu một tham số là *bPrefixName*. Nếu tham số này được gán giá trị *TRUE* thì tên trong bảng ký hiệu của bản vẽ tham chiếu ngoài sẽ được thêm tiền tố <tên khối>\$x\$ trong bản vẽ hiện tại, trong đó x : là số nguyên bất kỳ được tự động tăng để tránh ghi đè lên định nghĩa khối đã có. Nếu tham số này nhận giá trị *FALSE* thì tên trong bảng ký hiệu sẽ được ghép chung vào bản vẽ hiện tại mà không có tiền tố. Nếu có các tên trùng nhau thì AutoCAD sẽ sử dụng ký hiệu vừa được định nghĩa trong bản vẽ. Nếu không chắc chắn liệu trong bản vẽ có trùng tên trong bảng ký hiệu hay không thì nên gán *bPrefixName* là *TRUE*.

## Ràng buộc tham chiếu ngoài

Ví dụ này sẽ gắn một tham chiếu ngoài và sau đó liên kết nó với bản vẽ. Trong ví dụ này sử dụng tệp *City map.dwg* trong thư mục *sample*. Nếu không có hoặc tệp được lưu ở thư mục khác thì chỉ cần thêm tên và đường dẫn có thực cho biến `PathName` dưới đây.

```
Sub Ch10_BindingExternalReference()  
    On Error GoTo ERRORHANDLER  
  
    ' Định nghĩa tham chiếu ngoài cần chèn vào  
    Dim xrefHome As AcadBlock  
    Dim xrefInserted As AcadExternalReference  
    Dim insertionPnt(0 To 2) As Double  
    Dim PathName As String  
    insertionPnt(0) = 1  
    insertionPnt(1) = 1  
    insertionPnt(2) = 0  
    PathName = "c:/AutoCAD/sample/City map.dwg"  
  
    ' Thêm tham chiếu ngoài vào bản vẽ  
    Set xrefInserted = ThisDrawing.ModelSpace.  
        AttachExternalReference(PathName, "XREF_IMAGE", _  
            insertionPnt, 1, 1, 1, 0, False)  
    ZoomAll  
    MsgBox "The external reference is attached."  
  
    ' Ràng buộc tham chiếu ngoài vào bản vẽ  
    ThisDrawing.Blocks.Item(xrefInserted.name).Bind False  
    MsgBox "The external reference is bound."  
    Exit Sub  
ERRORHANDLER:  
    MsgBox Err.Description  
End Sub
```

## 3.7. Cắt xén các Khối và Tham chiếu ngoài

Không có phương thức nào được cung cấp trong ActiveX Automation để cắt xén đường bao của các khối và tham chiếu ngoài. Sử dụng lệnh `XCLIP` của AutoCAD hoặc gửi dòng lệnh `XCLIP` tới AutoCAD bằng phương thức `SendCommand`.

### 3.7.1. Tải vào theo nhu cầu và nâng cao tốc độ xử lý tham chiếu ngoài

Ta có thể cải thiện tốc độ thao tác với bản vẽ có sử dụng tham chiếu ngoài bằng cách kết hợp giữa việc tải theo nhu cầu và lưu bản vẽ với chỉ mục thông qua các biến hệ thống là `XLOADCTL` và `INDEXCTL`. Khi bật chế độ tải theo nhu cầu, và nếu có lưu chỉ mục trong bản vẽ được tham chiếu, AutoCAD chỉ tải vào bộ nhớ phần dữ liệu cần thiết để tái tạo trong bản vẽ hiện hành. Hay nói cách khác, bản vẽ được tham chiếu sẽ được đọc “theo nhu cầu”. Để thấy rõ lợi ích to lớn của tính năng này, ta cần phải lưu bản vẽ được tham chiếu với các chỉ mục lợp và chỉ mục không gian. Lợi ích của chế độ tải theo nhu cầu sẽ thể hiện rõ nhất khi:

- Ta cắt xén khối tham chiếu ngoài để thể hiện chỉ một phần của bản vẽ được tham chiếu, và chỉ mục không gian cũng được lưu trong bản vẽ tham chiếu ngoài;

- Ta làm đông một số lớp trong tham chiếu ngoài, và chỉ mục lớp cũng được lưu trong bản vẽ tham chiếu ngoài.

Để bật chế độ tải theo nhu cầu, ta sử dụng thuộc tính XRefDemandLoad. Nếu ta bật chế độ này với lựa chọn acDemandLoadEnabledWithCopy, AutoCAD sẽ tạo ra một bản sao tạm thời của tệp được tham chiếu và sẽ tham chiếu đến bản sao tạm thời đó. Khi đó, ta có thể vừa tải tham chiếu ngoài vừa cho phép thay đổi tệp bản vẽ được tham chiếu. Khi tắt chế độ tải theo nhu cầu, AutoCAD sẽ đọc toàn bộ bản vẽ được tham chiếu mà không cần quan tâm đến tính nhìn thấy của lớp hay những vùng được cắt xén.

Để bật chỉ mục lớp và chỉ mục không gian, ta sẽ thiết lập biến hệ thống INDEXCTL sử dụng phương thức SetVariable. Ta có thể gán một trong những giá trị sau cho biến INDEXCTL:

- 0 = không tạo chỉ mục
- 1 = tạo chỉ mục lớp
- 2 = tạo chỉ mục không gian
- 3 = tạo cả chỉ mục lớp và chỉ mục không gian

Mặc định, khi tạo mới bản vẽ trong AutoCAD, biến INDEXCTL có giá trị là 0.

## 4. Nối kết và khôi phục lại dữ liệu mở rộng

Dữ liệu mở rộng (xdata) được sử dụng như một công cụ liên kết thông tin với các đối tượng trong bản vẽ.

### Gán dữ liệu mở rộng cho tất cả các đối tượng trong tập đối tượng được chọn

Ví dụ này sẽ nhắc người dùng chọn đối tượng trong bản vẽ. Các đối tượng được chọn sẽ nằm trong một tập đối tượng được chọn và sau đó các dữ liệu mở rộng xác định sẽ được gán cho tất cả các đối tượng trong tập này:

```
Sub Ch10_AttachXDataToSelectionSetObjects()
    ' Tạo tập đối tượng được chọn
    Dim sset As Object
    Set sset = ThisDrawing.SelectionSets.Add("SS1")

    ' Nhắc người dùng chọn đối tượng
    sset.SelectOnScreen

    ' Định nghĩa dữ liệu mở rộng
    Dim appName As String, xdataStr As String
    appName = "MY_APP"
    xdataStr = "This is some xdata"
    Dim xdataType(0 To 1) As Integer
    Dim xdata(0 To 1) As Variant

    ' Định nghĩa giá trị cho mỗi mảng
    '1001 nhận biết appName
    xdataType(0) = 1001
    xdata(0) = appName

    '1000: xác định kiểu chuỗi
```

```

xdataType(1) = 1000
xdata(1) = xdataStr

' Vòng lặp qua tất cả các thực thể trong tập đối tượng được
' chọn và gán dữ liệu mở rộng cho mỗi thực thể
Dim ent As Object
For Each ent In sset
    ent.SetXData xdataType, xdata
Next ent
End Sub

```

### Xem dữ liệu mở rộng của các đối tượng trong tập đối tượng được chọn

Ví dụ này sẽ hiển thị dữ liệu mở rộng đã được gán kèm trong ví dụ trước. Nếu gán dữ liệu mở rộng khác kiểu chuỗi (kiểu 1000) thì cần sửa lại đoạn mã này.

```

Sub Ch10_ViewXData()
' Tìm tập lựa chọn được tạo từ ví dụ trước
Dim sset As Object
Set sset = ThisDrawing.SelectionSets.Item("SS1")
' Định nghĩa biến dữ liệu mở rộng để lưu thông tin của dữ
' liệu mở rộng
Dim xdataType As Variant
Dim xdata As Variant
Dim xd As Variant
' Định nghĩa biến đếm chỉ số
Dim xdi As Integer
xdi = 0
' Lặp qua tất cả các đối tượng trong tập lựa chọn
' và lấy giá trị dữ liệu mở rộng của mỗi đối tượng
Dim msgstr As String
Dim appName As String
Dim ent As AcadEntity
appName = "MY_APP"
For Each ent In sset
    msgstr = ""
    xdi = 0
    ' Lấy giá trị và kiểu của xdata appName
    ent.GetXData appName, xdataType, xdata
    ' Nếu biến xdataType chưa được khởi tạo thì không có
    ' xdata appName nhận được từ thực thể đó
    If VarType(xdataType) <> vbEmpty Then
        For Each xd In xdata
            msgstr = msgstr & vbCrLf & xdataType(xdi) _
                & ": " & xd
            xdi = xdi + 1
        Next xd
    End If
    ' Nếu biến msgstr là NULL, thì không có dữ liệu mở rộng
    If msgstr = "" Then msgstr = vbCrLf & "NONE"
    MsgBox appName & " xdata on " & ent.ObjectName & _
        ":" & vbCrLf & msgstr
Next ent
End Sub

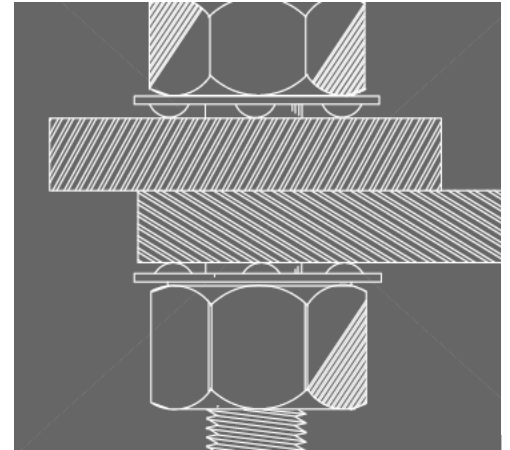
```

# PHÁT TRIỂN ỨNG DỤNG BẰNG VBA

Nhiều công việc trong lập trình không chỉ đơn giản là làm việc với mô hình đối tượng AutoCAD ActiveX. Chương này sẽ đề cập tổng quan về cách tạo các hộp thoại, xử lý lỗi, điều khiển cửa sổ ứng dụng và phân phối ứng dụng cho người khác.

Lưu ý rằng, tài liệu của Microsoft về VBA sẽ có thêm nhiều thông tin hơn.

T  
BM



Trong chương này **11**

- Một số thuật ngữ trong VBA
- Làm việc với Form trong VBA
- Xử lý lỗi
- Bảo mật mã nguồn chương trình VBA
- Thực thi Macro từ trình đơn hoặc thanh công cụ
- Tự động tải dự án VBA
- Tự động thực thi Macro
- Tự động mở VBA IDE mỗi khi tải một dự án
- Làm việc khi không có bản vẽ được mở
- Phân phối ứng dụng



## 1. Một số thuật ngữ trong VBA

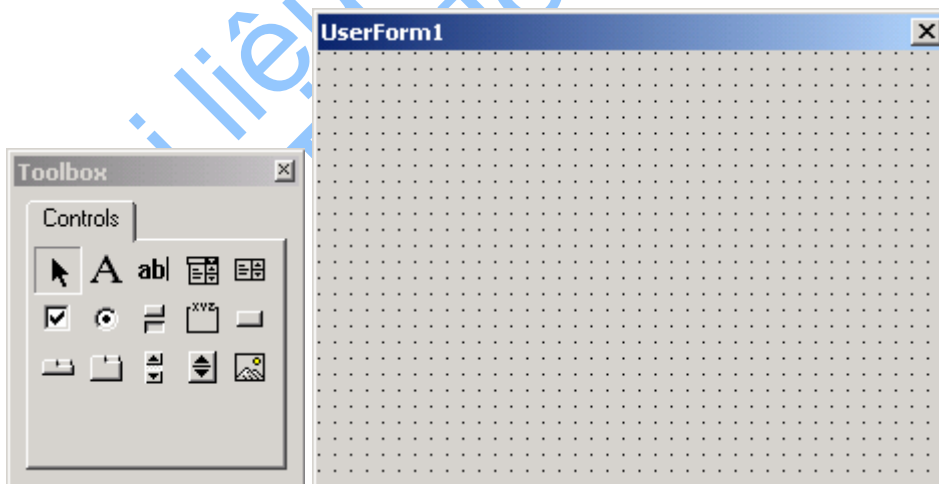
Nội dung chương này sẽ mở rộng các kinh nghiệm với VBA. Các khái niệm được nêu ra dưới đây sẽ giúp tìm hiểu và làm việc với môi trường VBA

Project	Dự án - là tập hợp của các form và các mô-đun được nhóm lại thành một tệp.
Module	là một nhóm các thủ tục và các hàm thường có liên quan đến nhau.
Macro	là những hàm và thủ tục có thể dùng chung. Macro thường được thể hiện dưới dạng những thành phần chạy được của Dự án.
Dialog box	là phương tiện để thể hiện thông tin hoặc tập hợp thông tin khi ứng dụng hoạt động.
Form	là đối tượng chứa các điều khiển hộp thoại.

## 2. Làm việc với Form trong VBA

Form là các khối cấu tạo cơ bản, trên đó ta có thể thiết kế các các hộp thoại cho các ứng dụng khác nhau. Qua các Form tự tạo, ta có thể cung cấp thông tin cho người dùng, nhận thông tin từ người dùng hoặc người dùng có thể điều khiển các hoạt động của ứng dụng.

Form cũng giống như một phong đề vẽ, ban đầu chúng hoàn toàn rỗng. Để làm đầy phong đó cần có bảng màu. Trong trường hợp này, bảng màu chính là hộp công cụ điều khiển (Toolbox). Người thiết kế chính là họa sỹ, sẽ thực hiện bố trí các điều khiển lấy từ hộp công cụ vào Form với số lượng tùy ý. Có thể điều chỉnh các thuộc tính và kích thước của các điều khiển cũng như của Form vào bất cứ lúc nào. Sau cùng sẽ thêm vào các chức năng (viết mã lệnh) cho các điều khiển để Form có thể hoạt động được.



Mặc dù Visual Basic hỗ trợ nhiều loại Form khác nhau nhưng VBA chỉ hỗ trợ duy nhất loại UserForm. Điều này có nghĩa là một số loại Form được tạo bởi Visual Basic sẽ không nhập được vào VBA.

Các UserForm, được gọi là Form trong tài liệu này, luôn là dạng Modal. Có nghĩa là khi nó xuất hiện trong lúc ứng dụng đang chạy, thì người dùng phải đóng nó lại trước khi muốn thực hiện bất cứ thao tác nào khác trong ứng dụng. Điều này sẽ được nói đến kỹ hơn trong phần “*Thiết kế chương trình với Modal Form*” trang 280.

## 2.1. Thiết kế và chạy chương trình

Khi tạo Form là lúc làm việc trong chế độ thiết kế, ở chế độ đó có thể thực hiện các thao tác sau:

- Thêm điều khiển vào Form
- Thay đổi thuộc tính của Form
- Thay đổi thuộc tính của các điều khiển trên Form
- Chèn mã lệnh cho các môđun của Form

Trong chế độ thiết kế không có bất cứ sự tương tác nào giữa người dùng, giao diện của AutoCAD và Form thiết kế.

Khi chạy ứng dụng thì Form sẽ ở chế độ hoạt động. Khi ở chế độ này, ta không thể tạo bất cứ sự thay đổi nào tới Form một cách trực tiếp. Tuy nhiên, khi Form đang được hiển thị trong giao diện sử dụng của AutoCAD, người dùng có thể tương tác với Form như là một phần thông thường trong ứng dụng.

## 2.2. Tạo Form mới trong Dự án

### Tạo Form mới trong Dự án

- 1 Mở cửa sổ Project của VBA IDE và chọn dự án muốn thêm Form mới.
- 2 Từ menu Insert chọn UserForm. Khi đó một Form trống được tạo ra và thêm vào dự án.

## 2.3. Thêm điều khiển vào Form

Rất dễ dàng để thêm các điều khiển vào Form, bằng cách chọn điều khiển trong hộp công cụ và kéo nó vào Form. Khi điều khiển đã được đặt trên Form, có thể thay đổi vị trí và kích cỡ của nó và có thể sao chép điều khiển đó tùy theo ý muốn của người thiết kế.

Bên cạnh cách nêu trên, còn có những cách khác để đặt điều khiển vào Form.

### Thay đổi kích cỡ của điều khiển khi đặt vào Form

- 1 Chọn điều khiển mong muốn trong hộp công cụ
- 2 Trên Form, nhấn chuột trái, kéo chuột và sau đó thả ra. Điều khiển lựa chọn sẽ được đặt trên Form và có kích cỡ phụ thuộc vào phạm vi kéo chuột.

### Đặt nhiều điều khiển giống nhau trên Form

- 1 Từ hộp công cụ Control, nháy đúp chuột vào điều khiển muốn đặt vào Form.

- 2 Trên Form, nháy chuột tại vị trí muốn đặt điều khiển. Di chuyển chuột đến một vị trí khác trên Form và nháy chuột lần nữa, một bản sao của điều khiển đó sẽ xuất hiện. Có thể tạo ra nhiều bản sao của điều khiển với cách làm tương tự.
- 3 Khi thực hiện xong các thao tác, trở lại hộp công cụ và nháy chuột vào điều khiển đó một lần nữa để bỏ lựa chọn.

### 2.3.1. Thay đổi kích thước và vị trí của điều khiển

Để dịch chuyển điều khiển chỉ cần chọn vào điều khiển đó và kéo nó tới vị trí mới trên Form. Để thay đổi kích cỡ của điều khiển trước hết phải chọn nó, khi đó đường bao của điều khiển sẽ được hiển thị. Sau đó kéo đường bao kích này cho phù hợp kích thước mong muốn (Form cũng được điều chỉnh kích thước theo cách tương tự).

Để di chuyển và thay đổi kích cỡ của nhiều điều khiển cùng một lúc thì lựa chọn từng điều khiển trong khi giữ phím SHIFT, sau đó có thể thực hiện di chuyển hoặc đổi kích thước của chúng theo nhóm.

### 2.3.2. Định dạng các điều khiển

VBA cung cấp một số chức năng điều chỉnh định dạng để giúp cho ta trình bày Form. Những chức năng này có thể được tìm thấy trong trình đơn Format của VBA IDE. Chúng cho phép căn chỉnh, tạo kích thước giống nhau, thay đổi khoảng cách cho nhiều điều khiển trên Form.

Chú ý, để chọn nhiều điều khiển, giữ phím SHIFT trong khi chọn.

### 2.3.3. Thay đổi thuộc tính của điều khiển

Các thuộc tính sẽ điều chỉnh các đặc điểm khác nhau của điều khiển ví dụ như: kích cỡ, hình dạng, màu sắc, nhãn và các giá trị mặc định. Gán các thuộc tính của điều khiển thực hiện khi thiết kế bằng cách sử dụng cửa sổ Properties.

#### Thay đổi thuộc tính của điều khiển

- 1 Chọn điều khiển cần thay đổi thuộc tính trên Form
- 2 Mở cửa sổ Properties bằng cách sử dụng phím F4 trong trường hợp cửa sổ đó chưa được mở.
- 3 Trong cửa sổ Properties, tìm thuộc tính muốn thay đổi và chọn giá trị hiện tại cho thuộc tính đó.
- 4 Thay đổi giá trị cho thuộc tính đó.

Ta cũng có thể thay đổi thuộc tính của đối tượng trong chương trình đang hoạt động bằng cách viết mã lệnh truy cập vào thuộc tính này. Tham khảo tài liệu của Microsoft để tìm thêm thông tin về thay đổi thuộc tính của điều khiển trong lúc chương trình đang hoạt động.

### 2.3.4. Thêm mã lệnh cho điều khiển

Sau khi bố trí các điều khiển trên Form theo mong muốn, tiến hành viết mã lệnh cho các điều khiển. Mở cửa sổ Code (cửa sổ viết mã lệnh) bằng cách nhấn đúp chuột

trên điều khiển, khi đó cửa sổ Code sẽ được mở ra cùng với một hàm được tạo sẵn cho điều khiển đó tương ứng với sự kiện mặc định của điều khiển.

Có thể viết mã lệnh cho sự kiện mặc định hoặc chọn một sự kiện khác từ hộp danh sách các sự kiện nằm ở góc trên bên phải cửa sổ Code.

## 2.4. Hiện thị và ẩn Form

Sau khi thiết kế Form và các điều khiển đã được viết mã lệnh đầy đủ thì bước cuối cùng là hiện thị Form cho người dùng khi chạy chương trình. Sự hiện thị của Form được thực hiện bởi phương thức Show. Phương thức này có thể được gọi từ bất cứ môđun nào trong ứng dụng.

### Hiện thị Form

Ví dụ dưới đây sẽ hiện thị Form có tên là “UserForm1”:

```
Public Sub MyApplication()  
    UserForm1.Show  
End Sub
```

Thủ tục (và tiếp đó là việc hiện thị Form) có thể gọi được như một Macro từ lệnh VBARUN hoặc từ trình đơn của AutoCAD.

Lưu ý rằng Form trong VBA ở chế độ Modal. Điều đó có nghĩa là người dùng sẽ không thể tương tác trực tiếp với AutoCAD (ví dụ chọn một điểm hay một đối tượng nào đó trên bản vẽ) khi Form đang được hiện thị. Để cho phép người dùng tương tác với bản vẽ của AutoCAD thì cần sử dụng phương thức Hide của VBA. Phương thức này sẽ ẩn Form đi và cho phép người dùng truy cập một cách hạn chế với AutoCAD. Khi sử dụng phương thức Hide, Form vẫn còn trong bộ nhớ của máy tính, nó vẫn giữ nguyên tất cả các giá trị hiện tại trong khi bị ẩn đi. Phương thức này được gọi theo cách tương tự như phương thức Show.

### Ẩn Form

```
UserForm1.Hide
```

## 2.5. Tải và dỡ bỏ Form

Một số trường hợp cần tải Form vào bộ nhớ trong lúc chạy ứng dụng nhưng không hiện thị Form. Trường hợp này có thể gặp phải khi người lập trình muốn khống chế thời điểm tải Form tốt hơn hoặc muốn truy cập Form nhưng không muốn người dùng nhìn thấy Form. Để tải Form nhưng không hiện Form, sử dụng phương thức Load của VBA. Phương thức Show có thể được sử dụng sau đó để người dùng nhìn thấy Form tại thời điểm thích hợp trong khi ứng dụng thực hiện. Cần nhớ rằng, người dùng không thể tương tác với Form cho đến khi nhìn thấy nó. Nếu phương thức Show được gọi mà Form chưa được tải vào thì nó sẽ được tự động tải.

Bên cạnh đó, cũng có một số trường hợp ta muốn dỡ bỏ Form. Việc dỡ bỏ Form sẽ loại bỏ Form đó khỏi bộ nhớ và tất cả các bộ nhớ liên quan đến Form đó sẽ được thu hồi lại. Cho đến khi Form được tải vào lần nữa bằng cách sử dụng phương thức Load hoặc Show thì người dùng không thể tương tác với Form và cũng không thể thao tác lập trình được. Ta có thể dỡ bỏ Form khi biết Form sẽ không được sử dụng lần nữa trong ứng dụng và khi cần tiết kiệm bộ nhớ.

Phương thức Hide không thực hiện được việc dỡ bỏ Form. Nếu ứng dụng kết thúc và Form chưa được dỡ bỏ thì nó sẽ được tự động dỡ bỏ. Dưới đây là bảng so sánh các phương thức Show, Hide, Load và Unload của VBA:

### Các phương thức VBA Show, Hide, Load và Unload

Phương thức	Vai trò
Show	Hiện thị Form. Nếu Form chưa được tải vào thì nó sẽ tự động tải Form vào.
Hide	Ẩn Form. Form chưa được dỡ bỏ khỏi bộ nhớ.
Load	Form được tải vào bộ nhớ nhưng chưa được hiển thị.
Unload	Dỡ bỏ Form khỏi bộ nhớ. Có thể thực hiện việc này bằng phương thức Unload hoặc được thực hiện tự động khi ứng dụng kết thúc.

## 2.6. Thiết kế chương trình với Modal Form

Tất cả các hộp thoại VBA trong AutoCAD đều ở dạng Modal. Điều này có nghĩa là người dùng cần trả lời các yêu cầu của hộp thoại trước khi tiếp tục thực hiện các phần khác của ứng dụng. Sẽ không có mã lệnh tiếp theo nào được thực hiện cho đến khi hộp thoại dạng Modal được đóng bằng phương thức Hide hoặc Unload. Điều này yêu cầu người phát triển ứng dụng cần suy nghĩ thận trọng về thời điểm cũng như cách thức sắp xếp trình tự của các hộp thoại.

Ví dụ cần thiết kế một hộp thoại yêu cầu người dùng chọn một đối tượng trong bản vẽ AutoCAD. Để người dùng có thể chọn được một đối tượng trong cửa sổ ứng dụng của AutoCAD, trước hết cần ẩn Form bằng cách gọi phương thức Hide. Khi đối tượng đã được chọn thì hiển thị lại Form bằng phương thức Show, tất cả các dữ liệu của nó vẫn được giữ nguyên trạng và ứng dụng sẽ tiếp tục chạy.

**CHÚ Ý** Mặc dù tất cả các Form khác trong ứng dụng đều không được phép hoạt động khi hộp thoại Modal hiển thị, nhưng các ứng dụng khác thì đều được phép hoạt động.

## 3. Xử lý lỗi

Có ba loại lỗi khác nhau có thể gặp phải trong ứng dụng: lỗi biên dịch, lỗi thực thi và lỗi logic.

Lỗi biên dịch xảy ra trong khi xây dựng ứng dụng. Những lỗi này hầu hết là lỗi cú pháp, các vấn đề về phạm vi của biến, hoặc lỗi nhập dữ liệu. Trong VBA, những kiểu lỗi như vậy thường được phát hiện bởi môi trường phát triển. Khi soạn một dòng lệnh sai thì dòng đó sẽ được đánh dấu và thông báo lỗi sẽ xuất hiện và nhắc nhở về lỗi đó. Các lỗi biên dịch cần phải được sửa trước khi ứng dụng có thể hoạt động.

Lỗi thực thi thường khó tìm và sửa hơn. Chúng thường xảy ra trong khi thi hành các dòng lệnh và thường do nhận thông tin từ người dùng. Ví dụ nếu ứng dụng yêu cầu người dùng nhập tên của bản vẽ và người dùng nhập tên của một bản vẽ không tồn

tại thì lỗi thực thi sẽ xảy ra. Để xử lý lỗi loại này một cách hiệu quả cần dự đoán các dạng vấn đề sẽ gặp phải để bẫy chúng và viết mã lệnh xử lý các tình huống đó.

Lỗi logic là khó tìm và sửa nhất. Hiện tượng của lỗi logic thường nằm trong các tình huống khi lỗi biên dịch và lỗi thực thi không xảy ra nhưng kết quả do ứng dụng tạo ra vẫn không đúng. Người lập trình coi đây là một loại lỗi mà có thể rất dễ hoặc rất khó để tìm và bẫy được.

Các thông tin để tìm và sửa ba loại lỗi nói trên có thể được tìm thấy trong tài liệu của môi trường lập trình. Các lỗi của AutoCAD thường là lỗi thực thi nên chúng sẽ được nói đến kỹ hơn trong các phần sau của chương này.

---

**CHÚ Ý** Hầu hết các ví dụ mẫu được cấp trong tài liệu của AutoCAD đều không sử dụng các bẫy lỗi. Điều này nhằm mục đích làm cho các ví dụ đơn giản và tập trung vào nội dung chính của nó. Tuy nhiên, cũng như các ngôn ngữ lập trình khác, xử lý lỗi và bẫy lỗi hợp lý là điều cần thiết để tăng hiệu quả của ứng dụng.

---

### **Có cần phải bẫy lỗi thực thi không?**

Hầu hết các môi trường lập trình phải cung cấp công cụ xử lý lỗi mặc định. Với VB và VBA, các phản ứng lỗi mặc định là sự xuất hiện của các thông báo lỗi và ngắt ứng dụng. Trong khi xây dựng ứng dụng thì kiểu phản ứng này là phù hợp, nhưng sẽ không được phép xuất hiện đối với người dùng. Khi lập trình, người lập trình có thể muốn bỏ qua hoặc cung cấp các phản ứng lỗi riêng cho một số lỗi; muốn ngăn sự xuất hiện của các thông báo của các lỗi hoặc đơn giản là điều khiển sự xuất hiện của các thông báo lỗi đối với người dùng. Bên cạnh đó, người dùng hầu như không thể chấp nhận việc tự động ngắt ứng dụng.

Nói chung, xử lý lỗi là cần thiết mỗi khi cần nhập các thông số đầu vào từ người dùng và mỗi khi làm việc với nhập/xuất các tệp. Lưu ý rằng, ngay cả khi tệp cần dùng đã sẵn sàng để được xử lý, nhưng có thể xuất hiện những yếu tố gây ra lỗi mà người lập trình chưa nghĩ đến.

### **3.1. Bẫy lỗi thực thi**

Trong VB và VBA, các lỗi thực thi được bẫy bằng cách sử dụng lệnh `On Error`. Câu lệnh này sẽ tạo bẫy cho hệ thống. Mỗi khi có lỗi xảy ra, lệnh này sẽ tự động chuyển hướng sang phần xử lý riêng biệt được thiết kế cho lỗi đó. Khi đó các xử lý lỗi mặc định của hệ thống sẽ bị bỏ qua.

Lệnh `On Error` có 3 dạng như sau:

```
On Error Resume Next  
On Error GoTo Label  
On Error GoTo 0
```

Lệnh `On Error Resume Next` được sử dụng khi người lập trình muốn bỏ qua lỗi. Lệnh này sẽ chỉ thực hiện bẫy lỗi thay vì sự xuất hiện của các thông báo và dừng chương trình, bằng cách nhảy đến thực hiện dòng lệnh tiếp theo. Ví dụ, nếu muốn tạo một thủ tục duyệt qua tất cả các đối tượng trong không gian vẽ để đổi màu cho chúng, người lập trình cần biết rằng khi một trong các đối tượng nằm trong một lớp bị khoá thì AutoCAD sẽ báo lỗi. Do vậy, thay vì phải dừng chương trình lại, chỉ cần bỏ qua các đối tượng thuộc lớp bị khoá đó để tiếp tục đổi màu cho các đối tượng còn lại.



### Xử lý lỗi sử dụng lệnh On Error Resume Next

Ví dụ dưới đây là một chương trình con duyệt qua tất cả các đối tượng trong không gian vẽ và đổi màu chúng thành đỏ. Áp dụng chương trình con này trong một bản vẽ có sẵn một số đối tượng mà trong đó có đối tượng thuộc một lớp bị khoá. Tiếp đó, giải thích lệnh On Error Resume Next và tiếp tục chạy chương trình. Chương trình sẽ dừng lại ở đối tượng đầu tiên thuộc lớp bị khoá.

```
Sub Ch11_ColorEntities()  
    Dim entry As Object  
    On Error Resume Next  
    For Each entry In ThisDrawing.ModelSpace  
        entry.Color = acRed  
    Next entry  
End Sub
```

Câu lệnh On Error GoTo Label được dùng khi người lập trình muốn khai báo xử lý lỗi. Câu lệnh này sẽ bắt lỗi và thay vì thông báo các lỗi và ngừng chương trình, nó sẽ nhảy đến đoạn mã lệnh ở vị trí được định sẵn. Đoạn mã chương trình này sẽ thực hiện các thao tác theo người lập trình muốn khi lỗi xuất hiện. Ví dụ, mở rộng ví dụ nêu trên bằng cách tạo ra các thông báo để xử lý trong trường hợp gặp đối tượng nằm trong lớp bị khoá.

### Xử lý lỗi sử dụng lệnh On Error GoTo

Ví dụ dưới đây là một chương trình con duyệt qua tất cả các đối tượng trong không gian mô hình và đổi màu thành đỏ. Khi gặp đối tượng thuộc lớp bị khoá, chương trình sẽ xuất hiện một hộp thoại để thông báo kiểu lỗi và thẻ của đối tượng. Áp dụng chương trình con này trong một bản vẽ có sẵn một số đối tượng và trong đó có đối tượng thuộc một lớp bị khoá. Tiếp đó, thêm dấu chú thích trước dòng lệnh On Error GoTo MyErrorHandler và tiếp tục chạy chương trình. Chương trình sẽ dừng lại ở đối tượng đầu tiên thuộc lớp bị khoá.

```
Sub Ch11_ColorEntities2()  
    Dim entry As Object  
    On Error GoTo MyErrorHandler  
    For Each entry In ThisDrawing.ModelSpace  
        entry.Color = acRed  
    Next entry  
    ' Quan trọng! Thoát khỏi thủ tục trước khi xử lý lỗi  
    Exit Sub  
MyErrorHandler:  
    MsgBox entry.EntityName + " is on a locked layer." + _  
        " The handle is: " + entry.Handle  
    Resume Next  
End Sub
```

Lệnh On Error GoTo 0 sẽ huỷ toàn bộ các xử lý lỗi hiện tại trong chương trình con. Lệnh On Error Resume Next và On Error GoTo Label giữ nguyên thông tin trong bộ xử lý lỗi hiện tại cho đến khi thủ tục kết thúc, hoặc khi có một bộ xử lý lỗi khác được khai báo, hoặc khi bộ xử lý lỗi bị huỷ do lệnh On Error GoTo 0.

## 3.2. Xử lý lỗi đã bắt được

Sau khi bắt được một lỗi, việc xử lý lỗi đó phụ thuộc vào ứng dụng và bản chất của lỗi. VB và VBA cung cấp các thông tin của lỗi đã bắt được thông qua đối tượng



Err. Đối tượng Err có các thuộc tính: Number, Description, Source, HelpTệp, HelpContext, LastDLLerr. Các thuộc tính nêu trên luôn nhận giá trị của lỗi gần nhất. Thuộc tính quan trọng nhất là Number và Description. Thuộc tính Number chứa mã của lỗi và thuộc tính Description chứa các thông báo xuất hiện khi có lỗi. Trong phần xử lý lỗi, người lập trình có thể so sánh giá trị của Number, từ đó có thể xác định được loại lỗi để có hướng xử lý phù hợp.

### 3.3. Xử lý lỗi nhập dữ liệu người dùng trong AutoCAD

Phương thức nhập dữ liệu từ người dùng cần có những bẫy lỗi để yêu cầu người dùng nhập vào loại dữ liệu nhất định theo đúng định dạng mà người lập trình mong muốn. Nếu người dùng nhập các loại dữ liệu khác, AutoCAD sẽ từ chối dữ liệu và đưa ra lời nhắc nhở. Sử dụng phương thức InitializeUserInput cùng với những chức năng nhập dữ liệu, không những tạo ra một công cụ tổng hợp để kiểm soát việc nhập số liệu từ người dùng mà còn gợi ý cho người lập trình thêm một số điều kiện cần được kiểm tra khi bẫy lỗi. Tham khảo về bẫy lỗi từ số liệu do người dùng nhập vào trong ví dụ ở phần “*Nhắc người dùng nhập liệu*” trang 84.

## 4. Bảo mật mã nguồn chương trình VBA

Mặc dù không hỗ trợ tạo ra các chức năng bảo vệ, nhưng VBA cung cấp mật khẩu bảo vệ cho các đối tượng nhìn thấy được của một dự án gồm các Form, các lớp, môđun. Có thể tìm thấy chức năng bảo mật trong trình đơn của VBA IDE như sau: Chọn Tools ▶ Project Properties ▶ Protection.

## 5. Thực thi Macro từ trình đơn hoặc thanh công cụ

Macro có thể được gọi từ thanh công cụ hoặc trình đơn của AutoCAD bằng cách thay đổi thuộc tính Macro của thanh công cụ hoặc trình đơn đó. Thuộc tính Macro phải gán bằng:

```
-VBARUN Tên_tệp.dvb!Tên_dự_án.Tên_Macro
```

trong đó: tên tệp, tên dự án, và tên Macro được chỉ ra trên dòng lệnh cùng với lệnh VBARUN. Tên tệp chỉ cần thiết khi tệp đó chưa được tải vào phiên làm việc hiện tại của AutoCAD, nếu đã có tên tức là tệp đó đã được tải vào.

Tham khảo thêm các thông tin cụ thể về thay đổi trình đơn và thanh công cụ trong chương 6 “*TÙY BIẾN THANH CÔNG CỤ VÀ TRÌNH ĐƠN*” trang 175.

## 6. Tự động tải dự án VBA

Có 2 cách khác nhau để tải tự động một dự án:

- Khi khởi động, VBA sẽ tìm trong đường dẫn của AutoCAD một dự án có tên là *acad.dvb*. Tệp này sẽ tự động tải như một dự án mặc định;
- Tất cả các dự án có tên khác với *acad.dvb* đều có thể được sử dụng sau khi tải vào bằng lệnh VBALOAD. Đoạn mã lệnh dưới đây sử dụng tệp khởi động của AutoLISP để tải VBA cũng như dự án VBA có tên là *myproj.dvb*

khi AutoCAD bắt đầu hoạt động. Chạy trình soạn thảo *notepad.exe* và tạo (hoặc thêm vào) tệp *acad.lsp* như sau:

```
(defun S::STARTUP ()  
  (command "_VBALOAD" "myproj.dvb")  
)
```

## 7. Tự động thực thi Macro

### Tự động thực thi Macro khi khởi động AutoCAD

Bất cứ Macro nào trong tệp *acad.dvb* đều có thể tự động thực hiện bằng cách gọi gián tiếp macro đó qua câu lệnh VBARUN từ một tiện ích khởi động nào đó của AutoCAD, như tệp *acad.lsp*. Ví dụ, để tự động thực hiện macro tên là *drawline*, trước hết lưu macro đó vào tệp *acad.dvb*, tiếp đó khởi động chương trình soạn thảo *notepad.exe* và tạo mới (hoặc thêm vào) tệp *acad.lsp* như sau :

```
(defun S::STARTUP ()  
  (command "_vbarun" "drawline")  
)
```

### Tự động thực thi Macro khi tải VBA

Ta có thể tự động chạy một Macro trong tệp *acad.dvb* bằng cách đặt tên cho Macro là *AcadStartup*. Bất cứ Macro nào trong tệp *acad.dvb* có tên như vậy sẽ được tự động chạy khi tải VBA.

## 8. Tự động mở VBA IDE mỗi khi tải một dự án

Trên hộp thoại Open VBA Project có một tùy chọn cho phép môi trường IDE được mở tự động khi một dự án VBA được tải. Tùy chọn này nằm ở góc dưới bên trái của hộp thoại và khi đã được chọn thì nó sẽ được giữ nguyên như vậy cho đến khi bị tắt đi.

---

**CHÚ Ý** Mở hộp thoại Open VBA Project từ dòng lệnh VBALOAD. Hộp thoại sẽ xuất hiện và cho phép người dùng lựa chọn dự án để tải vào. Nếu không nhìn thấy hộp thoại thì có thể do biến FILEDIA đã bị tắt đi. Đó là biến điều khiển sự xuất hiện của các hộp thoại, và để bật trở lại thì gán cho biến giá trị bằng 1.

---

## 9. Làm việc khi không có bản vẽ được mở

Một số điều cần lưu ý khi làm việc ở chế độ không có bản vẽ nào được mở:

- Đối tượng *ThisDrawing* chưa được định nghĩa ở trường hợp này. Khi đó việc sử dụng đối tượng này sẽ gây ra lỗi.
- Các đối tượng phụ thuộc vào bản vẽ cũng không được định nghĩa ở trường hợp này. Các đối tượng thuộc nhóm này là những đối tượng nằm dưới nhánh của đối tượng Document trong mô hình đối tượng của AutoCAD. Với những đối tượng không phụ thuộc bản vẽ như Application hoặc MenuBar thì được phép làm việc ở trạng thái này.

- AutoCAD không hiện dòng lệnh khi không có bản vẽ nào mở, nên bất cứ sự cố gắng dùng qua dòng lệnh của chương trình ở trạng thái này đều sẽ gây lỗi.

## 10. Phân phối ứng dụng

Ứng dụng VBA có thể được phân phối theo hai cách sau:

- Nhúng vào bản vẽ AutoCAD
- Lưu dự án VBA vào tệp

Hình thức phân phối cần được lựa chọn cho phù hợp với ứng dụng. Những ứng dụng có thể áp dụng trong bản vẽ hiện tại và không can thiệp vào các bản vẽ khác thường được nhúng luôn vào bản vẽ đó. Bằng cách này, ta có thể chắc chắn là ứng dụng đã được tải và sẵn sàng để sử dụng mỗi khi bản vẽ được mở.

Những ứng dụng được sử dụng bởi nhiều người và được cập nhật một cách thường xuyên, cần đóng mở nhiều bản vẽ khác, hoặc không được sử dụng thường xuyên, thì nên lưu thành tệp dự án của VBA riêng. Theo cách này, có một nơi tập trung ứng dụng và tất cả mọi người sẽ được đảm bảo sử dụng phiên bản mới nhất.

Các thông tin chi tiết về dự án được nhúng và tệp dự án VBA, tham khảo trong phần “*Khái niệm về dự án VBA nhúng và độc lập*” trang 28.

### 10.1. Phân phối ứng dụng Visual Basic

Ứng dụng Visual Basic hay bất cứ ứng dụng chạy độc lập nào đều không thể lưu trong một bản vẽ của AutoCAD. Chúng được biên dịch thành những tệp thực thi độc lập (EXE) và có thể chạy bằng lệnh `APPLOAD` của AutoCAD.

Tài liệu tham khảo  
BM Tự động hoá TKCĐ

# TƯƠNG TÁC VỚI ỨNG DỤNG KHÁC, CƠ SỞ DỮ LIỆU VÀ WINDOWS API<sup>1</sup>

Công nghệ ActiveX tạo khả năng trao đổi thông tin một cách dễ dàng với các ứng dụng khác của AutoCAD và cả các ứng dụng hỗ trợ ActiveX khác như Microsoft Excel hay Microsoft Word. Nội dung được đề cập trong chương này là tìm hiểu các thủ tục cơ bản khi thực hiện tương tác với các ứng dụng khác.



Trong chương này **12**

- Tương tác với ứng dụng Visual LISP
- Tương tác với ứng dụng trên Windows
- Sử dụng Data Access Objects (DAO) để truy cập thông tin của cơ sở dữ liệu
- Truy cập hàm Windows API từ VBA

---

<sup>1</sup> **Windows API:** Windows Application Programming Interface – tập hợp các chương trình con được sử dụng bởi ứng dụng để hướng việc thực thi của ứng dụng đến hệ điều hành Windows.

## 1. Tương tác với ứng dụng Visual LISP

Ứng dụng Visual LISP có thể truy cập tới tất cả phạm vi của các đối tượng ActiveX. Chúng có thể gọi các phương thức của ActiveX, gán và lấy các thuộc tính của đối tượng ActiveX. Ngoài ra các ứng dụng Visual LISP cũng có thể gọi các Macro VBA qua lệnh VBARUN.

ActiveX và các ứng dụng VBA có thể thực hiện các ứng dụng Visual LISP thông qua phương thức SendCommand để gửi câu lệnh đến dòng lệnh của AutoCAD.

Để tìm hiểu thêm về truy cập các đối tượng ActiveX thông qua Visual LISP, tham khảo trong “*Visual LISP Developer’s Guide*”

## 2. Tương tác với ứng dụng trên Windows

Công nghệ ActiveX của AutoCAD cho phép trao đổi thông tin một cách dễ dàng với các ứng dụng hỗ trợ ActiveX như Microsoft Excel hoặc Microsoft Word. Khả năng này cho phép tập hợp, lưu trữ và biểu diễn các thông tin của AutoCAD theo các định dạng khác ngoài dạng bản vẽ. Ngoài ra, có thể đọc các thông tin từ các ứng dụng khác vào AutoCAD để trực tiếp tạo ra hoặc tác động lên các đối tượng AutoCAD. Một ví dụ sử dụng công nghệ này là tạo hoá đơn vật liệu như một bản tính Excel từ các đối tượng trong bản vẽ AutoCAD.

Các chương trước đã hướng dẫn viết mã lệnh sử dụng mô hình đối tượng ActiveX của AutoCAD. Việc trao đổi thông tin với các ứng dụng được hỗ trợ ActiveX liên quan đến việc tham chiếu tới các mô hình đối tượng ActiveX của ứng dụng khác và viết mã lệnh cần thiết để sử dụng các đối tượng của chúng.

---

**CHÚ Ý** Chương này chỉ giới thiệu một cách vắn tắt về khả năng lập trình ứng dụng chéo. Những tài liệu về vấn đề này không phải là hướng tập trung của AutoCAD và chúng được đề cập trong cả tài liệu của Microsoft cũng như các hướng dẫn lập trình độc lập. Chúng cũng được đề cập sâu hơn trong các tài liệu “VBA Foundations for AutoCAD” và “VBA Solutions for AutoCAD” của hãng Autodesk.

---

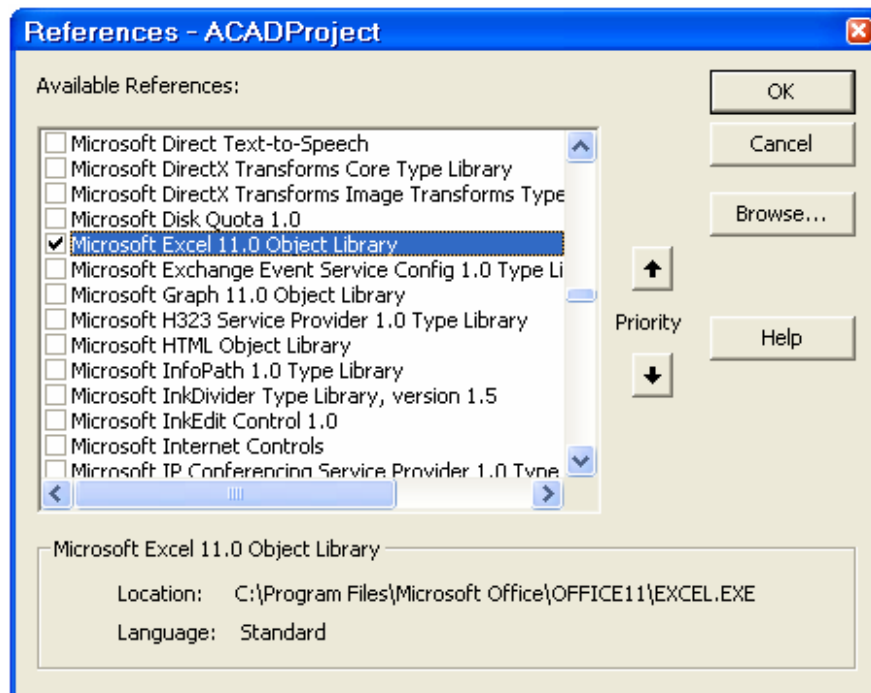
### Ba bước cơ bản để thực hiện trao đổi thông tin giữa các mô hình đối tượng ActiveX:

- 1 Tham chiếu đến các mô hình đối tượng ActiveX của các ứng dụng khác.  
Cần biết tên và mối quan hệ giữa các đối tượng trong mô hình đối tượng khác.
- 2 Tạo một đại diện của ứng dụng khác  
Bước này sẽ tạo ra đối tượng hợp lệ đại diện cho các đối tượng cơ bản trong mô hình đối tượng khác.
- 3 Viết mã chương trình sử dụng cả đối tượng của AutoCAD và của ứng dụng khác.  
Đây chính là nơi diễn ra quá trình trao đổi dữ liệu.

## 2.1. Tham chiếu thư viện đối tượng ActiveX của ứng dụng khác

Để lập trình truy cập tới một ứng dụng khác cần chỉ ra cho VBA cách tạo ra các đối tượng có thể sử dụng của ứng dụng đó. Việc này được thiết lập bằng cách tạo ra một tham chiếu tới thư viện đối tượng của nó. Đó là một tệp trong máy tính nơi định nghĩa tất cả các đối tượng, phương thức, thuộc tính, hằng số cũng như các sự kiện của ứng dụng.

Tạo tham chiếu tới thư viện đối tượng trong VBA IDE, chọn menu Tools►References:



Hộp thoại xuất hiện chứa danh sách liệt kê các thư viện đối tượng mà VBA tìm thấy trong hệ thống. Đánh dấu vào thư viện cần sử dụng trong danh sách, khi đó các thư viện được lựa chọn đã sẵn sàng cho việc tham chiếu của dự án hiện hành. Ví dụ, để tham chiếu thư viện đối tượng của Microsoft Excel, chọn thư viện đối tượng Microsoft Excel trong danh sách như hình minh họa trên.

Khi đã tạo một tham chiếu đến thư viện đối tượng của một ứng dụng khác thì có thể sử dụng Object Browser của VBA để xem danh sách các đối tượng của ứng dụng.

---

**CHÚ Ý** Phải thiết lập tham chiếu cho mỗi dự án VBA có sử dụng mô hình đối tượng này vì việc thiết lập các tham chiếu sẽ không được thực hiện tự động cho bất cứ dự án nào. Điều này là do hiệu suất của chương trình.

---



## 2.2. Tạo đại diện<sup>1</sup> của ứng dụng

Khi muốn tham chiếu đến thư viện đối tượng của ứng dụng, ta phải tạo ra một đại diện của ứng dụng đó. Đây chỉ là việc thông báo rằng ta bắt đầu làm việc với ứng dụng bằng một biến đối tượng cụ thể.

Để thực hiện điều này, trước hết cần khai báo một biến sẽ đại diện cho ứng dụng bằng cách sử dụng lệnh `Dim` và chỉ rõ ứng dụng trong khai báo đó. Ví dụ sau khai báo biến đối tượng kiểu `Excel.Application`:

```
Dim ExcelAppObj as Excel.Application
```

Sau khi khai báo biến, sử dụng lệnh `Set` và từ khoá `New` để gán biến bằng một đại diện đang thực thi của ứng dụng. Ví dụ sau, câu lệnh `Set` sẽ gán biến được khai báo ở trên cho ứng dụng `Excel`, từ khoá `New` khởi động một phiên làm việc mới của `Excel`.

```
Set ExcelAppObj = New Excel.Application
```

---

**CHÚ Ý** Một vài ứng dụng cho phép chỉ có một đại diện của nó được thực thi tại một thời điểm. Sử dụng từ khoá `New` đối với ứng dụng đó sẽ thiết lập một tham chiếu tới đại diện của nó và sẽ không khởi động thêm một đại diện mới nữa.

---

## 2.3. Lập trình với các đối tượng của ứng dụng khác

Sau khi đã tạo tham chiếu đến thư viện đối tượng và tạo một đại diện mới của ứng dụng thì ta mới có thể tạo ra và thao tác với các đối tượng trong ứng dụng đó. Tất cả các đối tượng, phương thức và thuộc tính được định nghĩa bởi mô hình đối tượng đều sẵn sàng sử dụng. Ví dụ, dùng biến được khai báo từ phần trước, dòng mã lệnh dưới đây sẽ hiển thị chương trình `Excel`:

```
ExcelAppObj.Visible = TRUE
```

Để có thể lập trình xây dựng ứng dụng ta cần thông thạo mô hình đối tượng của ứng dụng, ngoài ra ta có thể sử dụng `Object Browser` của `VBA` hoặc tài liệu trợ giúp của ứng dụng để tìm hiểu về mô hình đối tượng cần tham chiếu tới.

### 2.3.1. Thoát khỏi ứng dụng khác

Khởi động một ứng dụng khi lập trình sẽ làm tăng bộ nhớ của máy tính, do đó nên đóng chúng lại khi không sử dụng để tài nguyên của hệ thống được giải phóng.

Mặc dù mỗi mô hình đối tượng là khác nhau nhưng hầu hết đều có phương thức `Quit` được dùng để đóng ứng dụng. Ví dụ, dòng lệnh sau đây sẽ đóng chương trình `Excel`, sử dụng các khai báo biến ở các phần trên:

```
ExcelAppObj.Application.Quit
```

---

**CHÚ Ý:** Việc huỷ hoặc làm vượt quá phạm vi của biến đối tượng thực sự không làm kết thúc ứng dụng. Nên thường xuyên thoát khỏi ứng dụng bằng phương pháp thích hợp để đảm bảo rằng bộ nhớ được làm sạch đúng cách.

---

---

<sup>1</sup> **Đại diện (Instance):** là một loại biến đối tượng chỉ đến ứng dụng cụ thể được tham chiếu đến mà đã được cấp phát bộ nhớ (đang hoạt động).

## Liệt kê các thuộc tính của AutoCAD trong bảng tính Excel

Thủ tục dưới đây thực hiện tìm kiếm các tham chiếu khối trong bản vẽ hiện tại, sau đó sẽ tìm các thuộc tính gắn với khối đó và liệt kê chúng trong một bảng tính Excel.

### Trình tự thực hiện

- 1 Mở bản vẽ có chứa khối tham chiếu và các thuộc tính gắn với chúng (bản vẽ mẫu được dùng có đường dẫn là *sample/activeX/attrib.dwg*)
- 2 Mở VBA IDE bằng dòng lệnh VBAIDE của AutoCAD
- 3 Trong menu Tools ▶ References, chọn đối tượng Microsoft Excel 11.0 Object (hoặc phiên bản khác của đối tượng này)
- 4 Sao chép đoạn thủ tục dưới đây vào cửa sổ viết mã lệnh Code của VBA và thực hiện nó:

```
Sub Ch12_Extract()  
    Dim Excel As Excel.Application  
    Dim ExcelSheet As Object  
    Dim ExcelWorkbook As Object  
    Dim RowNum As Integer  
    Dim Header As Boolean  
    Dim elem As AcadEntity  
    Dim Array1 As Variant  
    Dim Count As Integer  
  
    ' Khởi động Excel.  
    Set Excel = New Excel.Application  
    ' Tạo một workbook mới và tìm kiếm bảng tính hoạt động.  
    Set ExcelWorkbook = Excel.Workbooks.Add  
    Set ExcelSheet = Excel.ActiveSheet  
    ExcelWorkbook.SaveAs "Attribute.xls"  
    RowNum = 1  
    Header = False  
  
    ' Lặp quá trình tìm kiếm trên không gian vẽ tất cả  
    ' các tham chiếu khối  
    For Each elem In ThisDrawing.ModelSpace  
        With elem  
            ' Khi một tham chiếu khối được tìm thấy thì nó sẽ được  
            ' kiểm tra luôn các thuộc tính  
            If StrComp(.EntityName, "AcDbBlockReference", 1) = 0 Then  
                If .HasAttributes Then  
                    ' Lấy giá trị thuộc tính  
                    Array1 = .GetAttributes  
                    ' Sao chép Tagstrings của các thuộc tính vào bảng  
                    Excel  
                    For Count = LBound(Array1) To UBound(Array1)  
                        If Header = False Then  
                            If StrComp(Array1(Count).EntityName, _  
                                "AcDbAttribute", 1) = 0 Then  
                                ExcelSheet.Cells(RowNum, Count + 1).value = _  
                                    Array1(Count).TagString  
                            End If  
                        End If  
                    Next Count  
                    RowNum = RowNum + 1  
                    For Count = LBound(Array1) To UBound(Array1)
```

```

        ExcelSheet.Cells(RowNum, Count + 1).value = _
            Array1(Count).textString
        Next Count
        Header = True
    End If
End If
End With
Next elem
Excel.Application.Quit
End Sub

```

### 3. Sử dụng DAO để truy cập thông tin của cơ sở dữ liệu

Các đối tượng truy cập dữ liệu (DAO - Data Access Objects ) cho phép làm việc với các dữ liệu trong một cơ sở dữ liệu nào đó. Đặc biệt là khi sử dụng DAO ta có thể truy cập vào bất cứ cơ sở dữ liệu nào được cung cấp bởi Microsoft Jet, bao gồm:

- Cơ sở dữ liệu Desktop như Access, dBase, FoxPro và Paradox
- Cơ sở dữ liệu ODBC như Microsoft SQL Server và Oracle®

DAO tạo điều kiện hoàn thiện điều khiển các cơ sở dữ liệu từ mã lệnh VBA nên ta có thể:

- Tạo một cơ sở dữ liệu mới hoặc thay đổi cấu trúc của một cơ sở dữ liệu cũ
- Thêm bảng vào cơ sở dữ liệu, định nghĩa quan hệ của các bảng; định nghĩa và thi hành các câu truy vấn
- Thêm, sửa hoặc xóa các bản ghi dữ liệu
- Bảo mật cơ sở dữ liệu

#### Có 3 bước cơ bản để sử dụng được DAO trong lập trình VBA

- 1 Tham chiếu đến thư viện đối tượng DAO của Microsoft (Microsoft DAO Object Library).
- 2 Mở cơ sở dữ liệu mà ta muốn làm việc, điều này sẽ tạo một thể hiện của đối tượng DAO.
- 3 Viết mã lệnh sử dụng cả mô hình đối tượng AutoCAD và mô hình đối tượng DAO.

---

**CHÚ Ý** Chương này cung cấp những giới thiệu vắn tắt về khả năng lập trình với DAO. Những tài liệu về vấn đề này không phải là hướng tập trung của AutoCAD và chúng được đề cập trong tài liệu của Microsoft cũng như các hướng dẫn lập trình độc lập khác. Chúng cũng được đề cập sâu hơn trong các tài liệu "VBA Foundations for AutoCAD" và "VBA Solutions for AutoCAD" của hãng Autodesk.

---

#### 3.1. Tham chiếu thư viện đối tượng DAO

Để viết mã lệnh truy cập DAO ta cần chỉ dẫn cho VBA tạo ra đối tượng trong DAO đã sẵn sàng làm việc, bằng cách tạo một tham chiếu đến thư viện đối tượng của DAO.

## Tạo một tham chiếu đến thư viện đối tượng của ứng dụng khác

- 1 Trong VBA IDE, mở menu Tools ▶ References
- 2 Tìm và lựa chọn mục Microsoft DAO Object Library
- 3 Chọn OK để đóng hộp thoại và cập nhật sự thay đổi.

Khi đã tạo được tham chiếu đến thư viện đối tượng của DAO, ta có thể sử dụng VBA Object Browser để nhìn thấy tất cả các đối tượng trong thư viện đối tượng của DAO.

---

**CHÚ Ý:** Phải gán tham chiếu cho mỗi dự án VBA có sử dụng mô hình đối tượng này vì việc thiết lập các tham chiếu sẽ không được thực hiện tự động cho bất cứ dự án nào. Điều đó là vì lý do hiệu suất của các chương trình.

---

### 3.2. Mở cơ sở dữ liệu

Để có thể làm việc với dữ liệu trong một cơ sở dữ liệu, ta cần mở cơ sở dữ liệu đó bằng VBA. Phương thức để mở cơ sở dữ liệu là `OpenDatabase` của đối tượng `Workspace` mặc định trong mô hình đối tượng DAO. Dòng mã lệnh dưới đây sẽ mở cơ sở dữ liệu có tên là `X.MDB`:

```
Dim db As Database  
Set db = DBEngine.Workspaces(0).OpenDatabase("C:\X.MDB")
```

Dòng mã lệnh này sẽ tham chiếu đến đối tượng `DBEngine` và sau đó sẽ tham chiếu đến đối tượng `Workspace` mặc định của DAO. Nó sẽ mở cơ sở dữ liệu trong đối tượng `Workspace` mặc định và gán cơ sở dữ liệu đó cho biến của đối tượng cơ sở dữ liệu.

### 3.3. Lập trình với mô hình đối tượng của DAO

Sau khi đã tham chiếu đến thư viện đối tượng DAO, thực hiện tạo và mở một cơ sở dữ liệu, thì ta có thể tiến hành truy vấn dữ liệu trong cơ sở dữ liệu. Tất cả các đối tượng, phương thức và thuộc tính được định nghĩa bởi mô hình đối tượng DAO đều có thể sử dụng.

Phần lớn thao tác với dữ liệu trong cơ sở dữ liệu sẽ liên quan tới đối tượng `Recordset` của DAO. Đối tượng này thể hiện một tập hợp gồm các hàng được trả về bởi một bảng, một truy vấn lựa chọn (bằng đối tượng truy vấn hoặc câu lệnh SQL) hoặc bởi một đối tượng `Recordset` khác. Đây là đối tượng cơ bản khi lập trình với dữ liệu và là yếu tố nổi bật nhất trong mã lệnh thao tác với dữ liệu.

Để tự làm quen và tìm hiểu thêm về mô hình đối tượng DAO, ta có thể sử dụng VBA Object Browser hoặc tài liệu hướng dẫn về DAO của Microsoft. Các tài liệu tham khảo về DAO còn bao gồm các mã lệnh ví dụ để giúp cho người học có thể bắt đầu được ngay.

## 4. Truy cập hàm Windows API từ VBA

Các thủ tục của Windows API luôn có trong hầu hết các ứng dụng của Windows. Chúng cho phép ta có thể mở rộng khả năng cho ứng dụng của mình.

Thông qua Windows API, ta có thể lấy được thông tin về hệ thống hiện hành như: những chương trình nào được cài đặt hay đang chạy trong hệ thống, thông tin nằm ở đâu trong hệ thống và những thiết lập điều khiển nào trong hệ thống hiện hành. Ngoài ra ta có thể truy cập được joystick, multimedia<sup>1</sup> và điều khiển âm thanh. Những khả năng trên chỉ là một phần nhỏ trong các khả năng được cung cấp bởi Windows APIs.

Để sử dụng các thủ tục Windows API, trước hết cần khai báo API trong ứng dụng. Thực hiện điều này bằng lệnh `Declare` trong VB. Lệnh này yêu cầu một số thông tin như sau:

- Tên của thư viện liên kết động (DLL) chứa thủ tục muốn dùng
- Tên của thủ tục trong thư viện DLL
- Tên của các thủ tục mà ta muốn dùng trong ứng dụng
- Các tham số của thủ tục
- Kiểu giá trị trả về nếu thủ tục được gọi là một hàm

Câu lệnh khai báo `Declare` có thể được đặt tại bất cứ đâu trong các mô đun của VBA.

Nếu nó được đặt trong một mô đun chuẩn thì nó sẽ sẵn sàng để sử dụng bởi bất cứ mô đun nào trong ứng dụng, trừ khi nó bị giới hạn phạm vi bởi từ khoá `Private`.

Nếu đặt lệnh `Declare` trong mô đun của một lớp hay Form thì chỉ có thể sử dụng chúng trong chính mô đun đó. Khi một thủ tục được khai báo thì nó có thể được gọi như những thủ tục khác trong ứng dụng mà ta đã tạo ra.

Sử dụng lệnh `Declare` đúng cách là một kỹ năng khó học, nhưng sử dụng sai thì rất dễ và thường dẫn đến những kết quả rất tồi tệ. Để chắc chắn, nên lưu bất cứ thông tin nào trong ứng dụng hiện tại trước khi thử lệnh `Declare` mới.

Để giúp người học sử dụng lệnh `Declare`, Microsoft đã cung cấp một tệp liệt kê những khai báo thông dụng nhất, gọi là `Win32api.txt` và được đính kèm với bộ chương trình Visual Basic và Office. Người dùng có thể tìm thủ tục cần dùng trong tệp này và sao chép lại khai báo đó vào mã lệnh của mình.

Tài liệu về Microsoft VBA cũng có thêm những thông tin về lệnh `Declare` và ví dụ sử dụng nó. Microsoft API Reference hiện là một phần trong bộ Microsoft Developer Network CD và cung cấp những tham khảo cho việc sử dụng các thủ tục sẵn có trong Windows APIs. Cuốn sách "*Visual Basic Programmer's Guide to the Win32 API*" của Dan Appleman cũng là một nguồn tài liệu hướng dẫn rất tốt cho những người lập trình với Visual Basic.

---

<sup>1</sup> **Joystick:** thiết bị ngoài, thường dùng để điều khiển các trò chơi trên máy tính. **Multimedia:** các thiết bị hay ứng dụng phục vụ cho việc giải trí bằng âm thanh và hình ảnh.

# THIẾT KẾ ĐƯỜNG ĐI DẠO TRONG VƯỜN - MỘT VÍ DỤ VỀ ActiveX/VBA

Đây là phần hướng dẫn người đọc cách thêm một Macro mới vào trong AutoCAD, giải thích sự làm việc của ActiveX và VBA, và cách sử dụng các công nghệ này hiệu quả nhất. Ví dụ này hướng theo kiến trúc cảnh quan nhưng những khái niệm mà người đọc có thể học được vẫn có thể vận dụng vào các lĩnh vực ứng dụng khác. Phần ví dụ này được soạn với đối tượng là những người sử dụng AutoCAD thành thạo nhưng mới học lập trình với VBA.

## Trong chương này **13**

- Kiểm tra môi trường làm việc
- Xác định mục đích
- Viết đoạn chương trình đầu tiên
- Nhập số liệu
- Vẽ đường đi dạo
- Vẽ lớp gạch lát
- Tổng hợp lại
- Duyệt mã lệnh
- Thực thi Macro
- Thêm giao diện hộp thoại

## 1. Kiểm tra môi trường làm việc

Để thực hiện các thao tác cần thiết trong ví dụ này, trước hết cần cài đặt môi trường VBA trong AutoCAD. Khi cài AutoCAD với tùy chọn Full hoặc Standard thì môi trường VBA được tự động cài đặt, nếu chọn Custom thì có thể VBA sẽ không được cài đặt.

### Kiểm tra việc cài đặt VBA:

- 1 Khởi động AutoCAD.
- 2 Tại dòng nhập lệnh, gõ **VBAIDE**.
- 3 Nếu môi trường lập trình VBA mở ra thì VBA đã được cài đặt. Nếu xuất hiện thông báo “AutoCAD VBA is not currently installed” thì VBA chưa được cài đặt.

### Cài đặt môi trường VBA:

- 1 Thoát khỏi chương trình AutoCAD.
- 2 Chạy chương trình cài đặt AutoCAD *setup.exe*.
- 3 Chọn Add để thêm một thành phần vào bản cài đặt hiện tại.
- 4 Chọn VBA Support.
- 5 Chọn Next để tiếp tục cài đặt
- 6 Khẳng định các cài đặt bằng cách chọn Next một lần nữa.
- 7 Khi đã hoàn thành việc cài đặt thì khởi động lại chương trình AutoCAD.
- 8 Tại dòng nhập lệnh, nhập **VBAIDE**. Môi trường lập trình VBA sẽ được mở ra và môi trường làm việc để thực hiện ví dụ trong chương này đã sẵn sàng.

## 2. Xác định mục đích

Mục tiêu trong ví dụ này là lập một Macro mới cho AutoCAD để vẽ một đường đi trong vườn và được lát bởi các viên bê tông tròn. Marco mới sẽ có các dòng nhắc theo trình tự như sau:

Command: **gardenpath**

Start point of path: *Người dùng sẽ chỉ ra điểm đầu của tuyến đường*

Endpoint of path: *Người dùng sẽ chỉ ra điểm kết thúc của tuyến đường*

Half width of path: *Người dùng nhập vào giá trị của một nửa bề rộng*

Radius of tiles: *Người dùng nhập vào giá trị bán kính của gạch lát*

Spacing between tiles: *Người dùng nhập vào khoảng cách giữa hai viên gạch*

Khi thực hiện, Macro sẽ nhắc người dùng nhập vào điểm bắt đầu và kết thúc của tim đường đi dạo thiết kế. Tiếp đó, macro nhắc người dùng nhập bề rộng nửa đường (vẽ bề rộng nửa từ đường tim sẽ giúp dễ hình dung hơn là cả bề rộng đường) và bán kính của những viên gạch lát hình tròn. Cuối cùng, người dùng được yêu cầu nhập khoảng cách giữa các viên gạch lát.



### 3. Viết đoạn chương trình đầu tiên

Macro sẽ được xây dựng dựa trên một loạt các hàm và thủ tục. Một số thủ tục làm việc với các góc vì lý do đơn vị của góc trong ActiveX là radians nhưng hầu hết người dùng thường quen làm việc với góc tính theo độ. Do đó, trước hết cần xây dựng hàm chuyển đổi từ độ sang radians. Nếu chưa mở VBA IDE thì khởi động nó từ bảng lệnh **VBAIDE**.

Trong VBA IDE, mở cửa sổ Code như sau: trong menu View ▶ Code; hoặc dùng phím F7.

#### Hàm chuyển đổi từ độ sang radians

Gõ đoạn mã sau vào cửa sổ Code sau dòng Option Explicit:

```
Const pi = 3.14159
' Chuyển đổi từ độ sang radians
Function dtr(a As Double) As Double
    dtr = (a / 180) * pi
End Function
```

Ngay khi xuống dòng để kết thúc dòng lệnh Function dtr(a As Double) As Double, VBA sẽ tự động thêm dòng lệnh End Function. Môi trường lập trình VBA luôn thực hiện như vậy để đảm bảo rằng tất cả các chương trình con đều có lệnh kết thúc tương ứng.

Ở dòng đầu tiên, hằng số  $\pi$  được gán giá trị là 3.14159. Điều này cho phép thay thế việc gõ lại số 3.14159 mỗi khi cần dùng bằng cách nhập  $\pi$ .

Dòng tiếp theo để định nghĩa một hàm có tên là dtr (Degrees To Radians). Hàm số này cần một đối số  $a$  là góc theo đơn vị độ. Kết quả tính bằng cách chia góc  $a$  cho 180 và nhân với  $\pi$ . Dòng bắt đầu bằng dấu nháy đơn là câu chú thích. VBA sẽ bỏ qua tất cả những dòng bắt đầu bằng ký tự này.

Hàm này đã có thể được sử dụng trong các chương trình con khác trong dự án. Sau đó lưu dự án với tên tùy chọn theo trình tự chọn: menu File ▶ Save, tuy nhiên để tiện dùng, trong ví dụ này đặt tên cho dự án là *gardenpath.dvb*

#### Tính khoảng cách giữa hai điểm

Tiếp theo thêm một hàm tính khoảng cách giữa hai điểm. Nhập đoạn mã sau đây tiếp theo hàm dtr:

```
' Tính khoảng cách giữa hai điểm
Function distance(sp As Variant, ep As Variant) As Double
    Dim x As Double
    Dim y As Double
    Dim z As Double

    x = sp(0) - ep(0)
    y = sp(1) - ep(1)
    z = sp(2) - ep(2)
    distance = Sqr((Sqr((x ^ 2) + (y ^ 2)) ^ 2) + (z ^ 2))
End Function
```

Lưu kết quả đã thực hiện.

## 4. Nhập số liệu

Macro sẽ nhắc nhở người dùng nhập vị trí để vẽ đường đi, bề rộng của đường, và khoảng cách giữa các viên gạch lát. Cho nên tiếp theo sẽ định nghĩa một thủ tục để yêu cầu người dùng nhập vào các thông tin yêu cầu ở trên và sau đó thực hiện tính toán.

Trong thủ tục này, sử dụng các phương thức nhập số liệu của đối tượng `Utility`.

### 4.1. Khai báo biến

Thủ tục tiếp theo sẽ sử dụng một số biến toàn cục. Tất cả các biến toàn cục cần được khai báo trước khi các thủ tục có thể sử dụng chúng.

#### Khai báo biến toàn cục

Trong VBA IDE, nhập đoạn mã lệnh sau trong cửa sổ Code ngay dưới dòng lệnh `Const pi = 3.14159:`

```
' Các biến để lưu các thông số của đường trong vườn
Private sp(0 To 2) As Double
Private ep(0 To 2) As Double
Private hwidth As Double
Private trad As Double
Private tspac As Double
Private pangle As Double
Private plength As Double
Private totalwidth As Double
Private angp90 As Double
Private angm90 As Double
```

Hai hộp danh sách dạng thả xuống ở phía trên của cửa sổ Code được gọi tương ứng là `Object Box` (hộp chứa danh sách các đối tượng) và `Procedure/Event Box` (hộp danh sách chứa các thủ tục và sự kiện). Hiện tại, giá trị trong hai hộp danh sách này lần lượt là `General` và `Declarations`. Hai hộp danh sách sẽ cho biết phần dự án mà con trỏ dừng lại là phần nào: cụ thể là đối tượng nào và thủ tục hay sự kiện nào của đối tượng đó. Phần `Declarations` sẽ tương ứng với phần khai báo các biến sẽ sử dụng trong nhiều chương trình con. Để chuyển sang làm việc với đối tượng khác hoặc sự kiện khác của đối tượng, có thể thực hiện nhanh chóng bằng cách lựa chọn tên tương ứng trong hai hộp danh sách này.

Chú ý đến dòng `Option Explicit` nằm trên cùng của phần `Declaration`. Khi dòng này xuất hiện trong một mô-đun tức là các biến phải khai báo rõ ràng bằng các lệnh `Dim`, `Private`, `Public`, `ReDim`, hoặc `Static`. Các trường hợp sử dụng biến chưa được khai báo thì sẽ phát sinh thông báo lỗi. Nếu không sử dụng dòng lệnh `Option Explicit`, tất cả các biến mà chưa được khai báo sẽ mang kiểu `Variant`. Người lập trình nên sử dụng dòng lệnh này để tránh những lỗi cú pháp khi nhập tên của các biến đã có hoặc tránh nhầm lẫn khi phạm vi của các biến không được khai báo rõ ràng.

### 4.2. Tạo chương trình con `gpuser`

Chương trình con `gpuser` sẽ nhắc người dùng nhập tất cả các thông tin cần thiết để vẽ đường đi dạo trong vườn.

## Nhắc người dùng nhập số liệu:

Viết đoạn mã sau ngay dưới hàm distance:

```
' Thông tin cần thiết về đường đi bộ
Private Sub gpuser()
    Dim varRet As Variant
    varRet = ThisDrawing.Utility.GetPoint( , _
    "Start point of path: ")
    sp(0) = varRet(0)
    sp(1) = varRet(1)
    sp(2) = varRet(2)
    varRet = ThisDrawing.Utility.GetPoint( , _
    "Endpoint of path: ")
    ep(0) = varRet(0)
    ep(1) = varRet(1)
    ep(2) = varRet(2)
    hwidth = ThisDrawing.Utility.GetDistance(sp, _
    "Half width of path: ")
    trad = ThisDrawing.Utility.GetDistance(sp, _
    "Radius of tiles: ")
    tspac = ThisDrawing.Utility.GetDistance(sp, _
    "Spacing between tiles: ")
    pangle = ThisDrawing.Utility.AngleFromXAxis(sp, ep)
    totalwidth = 2 * hwidth
    plength = distance(sp, ep)
    angp90 = pangle + dtr(90)
    angm90 = pangle - dtr(90)
End Sub
```

Đoạn mã lệnh này định nghĩa một chương trình con có tên là `gpuser`. Chương trình con này không có tham số và sẽ yêu cầu người dùng nhập tất cả các thông tin cần thiết.

Dòng lệnh `Dim varRet As Variant` thực hiện khai báo biến `varRet`. Vì biến này chỉ sử dụng trong phạm vi của chương trình con này nên chỉ cần khai báo cục bộ trong phạm vi của nó thay vì khai báo trong phần Declarations.

Dòng lệnh tiếp theo `varRet = ThisDrawing.Utility.GetPoint( , "Start point of path: ")` thực hiện gọi phương thức `GetPoint`. Dấu gạch ngang để nối dòng lệnh với dòng ngay dưới nó, khi đó VBA sẽ đọc hai dòng như một dòng lệnh. Dấu gạch này không bắt buộc phải có nên có thể nối liền hai dòng lại nhưng nếu sử dụng để ngắt dòng lệnh dài thì sẽ giúp đọc dễ dàng hơn.

Để truy cập phương thức `GetPoint`, cần thông qua đối tượng đại diện cho bản vẽ hiện hành là `ThisDrawing`. Khi nhập `ThisDrawing` và tiếp theo là dấu chấm (`.`) có nghĩa là sẽ thực hiện truy cập vào một thứ gì đó chứa trong đối tượng `ThisDrawing`. Sau dấu chấm gõ `Utility` và dấu chấm tiếp theo, tức là sẽ truy cập vào bên trong của đối tượng `Utility`. Cuối cùng nhập `GetPoint` là tên của phương thức đang cần gọi.

Phương thức `GetPoint` yêu cầu 2 tham số. Tham số thứ nhất không bắt buộc và sẽ không sử dụng trong chương trình con này bằng cách để trống tham số đó đặt dấu phẩy (`,`) để phân biệt nó. Tham số thứ hai là lời nhắc nên cũng không bắt buộc. Với tham số này, nhập một chuỗi để nhắc người dùng nhập điểm đầu. Điểm người dùng

nhập được gán cho biến `varRet`. Ba dòng lệnh tiếp theo sẽ gán tọa độ của điểm người dùng vừa nhập cho mảng `sp`.

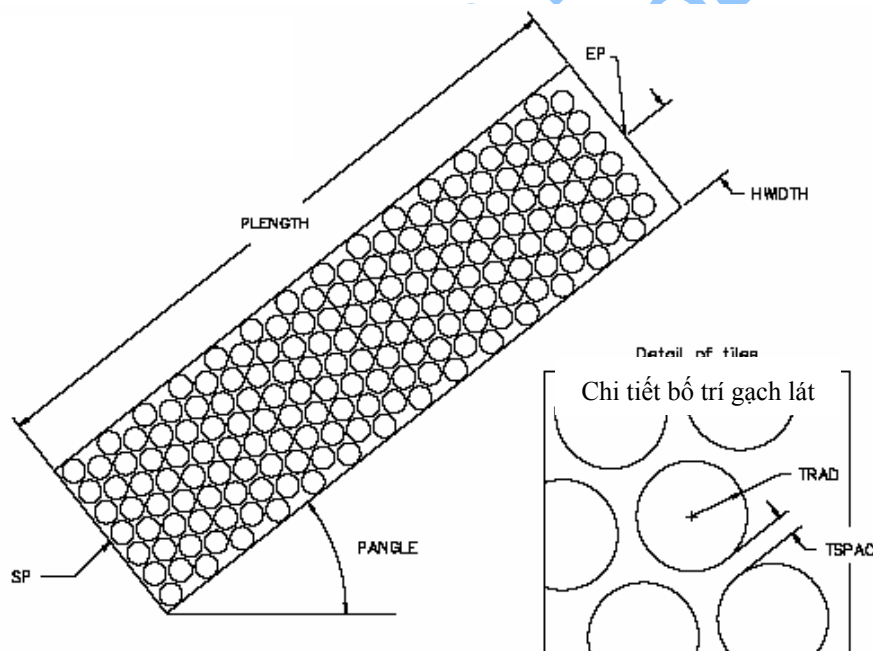
Điểm cuối cũng được lấy theo cách tương tự như điểm đầu.

Phương thức `GetDistance` để nhập bề rộng nửa của đường đi bộ (`hwidth`), bán kính của gạch lát (`trad`) và khoảng cách giữa các viên gạch (`tspac`). Phương thức này cần hai tham số, thứ nhất là điểm cơ sở, thứ hai là một lời nhắc: yêu cầu một chuỗi hướng dẫn người dùng nhập giá trị phù hợp. Phương thức `GetDistance` có thể trả về giá trị khoảng cách được nhập từ dòng lệnh và cả khoảng cách giữa điểm được chọn trong AutoCAD và điểm cơ sở.

Chương trình con sẽ tính toán các biến khác mà được sử dụng tiếp theo trong Macro. Biến `pangle` được gán cho góc giữa đường thẳng tạo bởi điểm đầu điểm cuối và chiều dương của trục x bằng phương thức `AngleFromXAxis`. Bề rộng của đường đi bộ được tính bằng 2 lần của giá trị nửa bề rộng nhập vào từ người dùng. Biến `plength` được gán giá trị là chiều dài của đường đi bộ được xác định ở trên trong phương thức `GetDistance`. Cuối cùng các biến `angp90` và `angm90` được tính và lưu bằng cách cộng và trừ giá trị góc của đường đi bộ với  $90^0$ .

Lưu kết quả đã thực hiện.

Hình vẽ dưới biểu diễn cách các biến được tính trong chương trình con để xác định các kích thước của đường đi bộ.



## 5. Vẽ đường đi dạo

Khi đã có đủ các thông số định vị và bề rộng thì có thể bắt đầu thực hiện vẽ đường đi dạo.

### Vẽ đường đi dạo trong vườn:

Thêm đoạn mã dưới đây ngay dưới chương trình con `gpuser`:

```

' Vẽ đường đi dạo
Private Sub drawout()
    Dim points(0 To 9) As Double
    Dim pline As AcadLWPolyline
    Dim varRet As Variant
    varRet = ThisDrawing.Utility.PolarPoint(sp, angm90, hwidth)
    points(0) = varRet(0)
    points(1) = varRet(1)
    points(8) = varRet(0)
    points(9) = varRet(1)
    varRet = ThisDrawing.Utility.PolarPoint( _
        varRet, pangle, plength)
    points(2) = varRet(0)
    points(3) = varRet(1)
    varRet = ThisDrawing.Utility.PolarPoint( _
        varRet, angp90, totalwidth)
    points(4) = varRet(0)
    points(5) = varRet(1)
    varRet = ThisDrawing.Utility.PolarPoint( _
        varRet, pangle + dtr(180), plength)
    points(6) = varRet(0)
    points(7) = varRet(1)
    Set pline = ThisDrawing.ModelSpace. _
        AddLightWeightPolyline(points)
End Sub

```

Thủ tục này sẽ vẽ đường đi dạo sử dụng phương thức `AddLightweightPolyline`. Phương thức này cần một tham số là mảng của các điểm tạo thành đường đa tuyến. Để vẽ đa tuyến cần tìm tất cả các điểm tạo thành đường đa tuyến đó và xếp chúng vào mảng tọa độ. Với đường đi dạo thì các điểm cần thiết chính là các góc của nó.

Tìm các góc của đường đi dạo bằng cách sử dụng phương thức `PolarPoint`. Phương thức này sẽ tìm điểm khi biết góc và khoảng cách của điểm đó so với điểm cơ sở. Từ điểm đầu (`sp`) có thể tìm được góc thứ nhất của đường theo chiều kim đồng hồ, đỉnh này cách `sp` một khoảng bằng nửa bề rộng đường (`hwidth`) và tạo góc ( $-90^0$ ) so với trục đường. Để vẽ đường khép kín bao quanh phạm vi đường thì điểm này sẽ là điểm bắt đầu và cũng là điểm kết thúc của mảng tọa độ các điểm. Vì thế tọa độ X và Y được trả về từ phương thức `PolarPoint` sẽ được gán cho cả điểm đầu và điểm cuối của mảng.

Các góc còn lại sẽ được tính theo cách tương tự sử dụng chiều dài, bề rộng của đường đi dạo (`plength` và `width`) và góc của đường đi dạo tạo với chiều dương của trục x.

Mỗi khi phương thức `PolarPoint` được gọi thì tọa độ của các điểm nhận được (biến `varRet`) sẽ được chép vào mảng tọa độ các điểm.

Khi tất cả các góc đã được xác định trong mảng tọa độ các điểm thì thực hiện phương thức `AddLightweightPolyline`, chú ý là phương thức này được gọi từ đối tượng `ModelSpace`. Nếu thực hiện chương trình con này, cần chú ý rằng đường đa tuyến chưa hiển thị trên AutoCAD cho đến khi bản vẽ được cập nhật lại (sẽ thực hiện ở phần sau).

## 6. Vẽ lớp gạch lát

Sau khi tạo thủ tục nhập số liệu từ người dùng cùng với thủ tục vẽ đường đi dạo, tiếp theo sẽ bố trí gạch lát hình tròn cho đường đi dạo.

Thao tác này yêu cầu một số yếu tố về hình học

### bố trí gạch lát hình tròn trong đường đi dạo

Trong VBAIDE, nhập dòng mã lệnh sau trong cửa sổ Code ngay dưới thủ tục drawout:

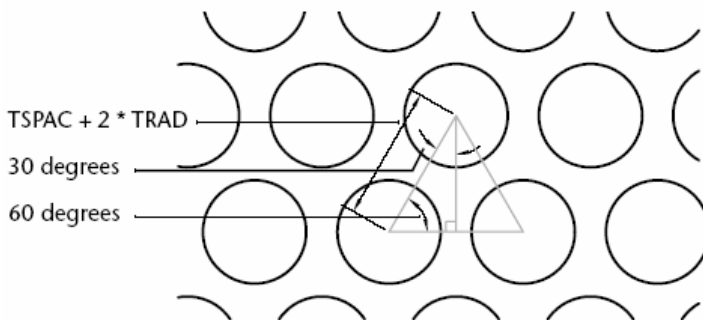
```
' Đặt một hàng gạch dọc theo chiều dài của đường đi dạo
' và các khoảng bù thích hợp
Private Sub draw(pd As Double, offset As Double)
    Dim pfirst(0 To 2) As Double
    Dim pctile(0 To 2) As Double
    Dim pltile(0 To 2) As Double
    Dim cir As AcadCircle
    Dim varRet As Variant
    varRet = ThisDrawing.Utility.PolarPoint( _
        sp, pangle, pd)
    pfirst(0) = varRet(0)
    pfirst(1) = varRet(1)
    pfirst(2) = varRet(2)
    varRet = ThisDrawing.Utility.PolarPoint( _
        pfirst, angp90, offset)
    pctile(0) = varRet(0)
    pctile(1) = varRet(1)
    pctile(2) = varRet(2)
    pltile(0) = pctile(0)
    pltile(1) = pctile(1)
    pltile(2) = pctile(2)
    Do While distance(pfirst, pltile) < (hwidth - trad)
        Set cir = ThisDrawing.ModelSpace.AddCircle( _
            pltile, trad)
        varRet = ThisDrawing.Utility.PolarPoint( _
            pltile, angp90, (tspac + trad + trad))
        pltile(0) = varRet(0)
        pltile(1) = varRet(1)
        pltile(2) = varRet(2)
    Loop
    varRet = ThisDrawing.Utility.PolarPoint( _
        pctile, angm90, tspac + trad + trad)
    pltile(0) = varRet(0)
    pltile(1) = varRet(1)
    pltile(2) = varRet(2)
    Do While distance(pfirst, pltile) < (hwidth - trad)
        Set cir = ThisDrawing.ModelSpace.AddCircle( _
            pltile, trad)
        varRet = ThisDrawing.Utility.PolarPoint( _
            pltile, angm90, (tspac + trad + trad))
        pltile(0) = varRet(0)
        pltile(1) = varRet(1)
        pltile(2) = varRet(2)
    Loop
End Sub
```

```

' Vẽ hàng gạch
Private Sub drawtiles()
    Dim pdist As Double
    Dim offset As Double
    pdist = trad + tspac
    offset = 0
    Do While pdist <= (plength - trad)
        draw pdist, offset
        pdist = pdist + ((tspac + trad + trad) * Sin(dtr(60)))
        If offset = 0 Then
            offset = (tspac + trad + trad) * Cos(dtr(60))
        Else
            offset = 0
        End If
    Loop
End Sub

```

Hình vẽ dưới đây mô tả thủ tục trên làm việc như thế nào. Thủ tục `draw` vẽ một hàng gạch theo một khoảng cách cho trước dọc theo đường đi bộ, khoảng cách này được xác định bởi tham số thứ nhất của chương trình con và sau đó sẽ tạo hàng gạch thứ hai song song và cách nó một khoảng bằng tham số thứ hai của chương trình con này. Các hàng có thể được sao chép và đặt so le nhau để lấp được nhiều khoảng trống hơn và bố trí đẹp mắt hơn.



Chương trình `draw` sẽ định vị hàng gạch đầu tiên bằng cách sử dụng phương thức `PolarPoint` để di chuyển dọc theo trục đường với khoảng cách xác định bởi tham số thứ nhất. Sau đó chương trình sẽ dùng tiếp phương thức `PolarPoint` để dịch chuyển vuông góc với trục đường đến vị trí hàng gạch song song. Chương trình sử dụng lệnh `While` để vẽ các vòng tròn cho đến khi gặp mép đường đi dạo. Trong vòng lặp thứ nhất của lệnh `While`, phương thức `PolarPoint` sẽ dịch chuyển đến vị trí của viên gạch tiếp theo với khoảng dịch chuyển bằng hai lần bán kính một viên gạch (`trad`) cộng với khoảng cách giữa hai viên gạch trên một hàng (`tspac`). Vòng lặp thứ hai sẽ vẽ viên gạch trên hàng theo hướng khác cho đến khi gặp cạnh nào đó của đường đi dạo.

Chương trình con `drawtiles` lặp lại việc gọi thủ tục `draw` đến khi vẽ xong các hàng gạch. Lệnh `While loop` sẽ thực hiện lặp từng bước dọc theo đường đi dạo và gọi thủ tục `draw` để vẽ toàn bộ hàng. Các viên gạch liền kề nhau trong các hàng sẽ tạo thành một tam giác đều như mô tả trong hình vẽ trên. Cạnh của tam giác đều đó có độ dài bằng 2 lần bán kính của viên gạch cộng với khoảng cách giữa các hàng. Do đó theo quan hệ hình học trong tam giác, khoảng cách giữa các hàng gạch và khoảng cách giữa các viên gạch trong một hàng bằng tích độ dài cạnh của tam giác nhân với  $\cos$  của  $60^\circ$ . Lệnh `If` được dùng trong thủ tục `drawtiles` để tạo các hàng gạch



khác song song. Nếu khoảng cách (biến `offset`) bằng 0 thì gán cho biến đó giá trị bằng khoảng cách giữa các hàng gạch như đã tính ở trên. Và nếu khoảng cách `offset` khác 0 thì sẽ gán cho giá trị đó bằng 0 để có thể tạo khoảng cách giữa các hàng gạch như mong muốn.

Lưu kết quả vừa thực hiện.

## 7. Tổng hợp lại

Cuối cùng ta có thể tổng hợp các chương trình con lại với nhau để tạo ra marco với tên là `gardenpath`.

### Tạo chương trình con `gardenpath`

Trong VBA IDE, nhập đoạn mã lệnh dưới đây vào cửa sổ Code ngay dưới chương trình con `drawtiles`:

```
' Dòng lệnh thực thi, gọi các hàm thành phần
Sub gardenpath()
    Dim sblip As Variant
    Dim scmde As Variant
    gpuser
    sblip = ThisDrawing.GetVariable("blipmode")
    scmde = ThisDrawing.GetVariable("cmdecho")
    ThisDrawing.SetVariable "blipmode", 0
    ThisDrawing.SetVariable "cmdecho", 0
    drawout
    drawtiles
    ThisDrawing.SetVariable "blipmode", sblip
    ThisDrawing.SetVariable "cmdecho", scmde
End Sub
```

Thủ tục `path` sẽ gọi thủ tục `gpuser` để tập hợp các thông số đầu vào cần thiết. Phương thức `GetVariable` được sử dụng để lưu giá trị hiện tại của các biến hệ thống `BLIPMODE`, `CMDECHO` và gán chúng cho các biến `sblip` và `scmde`. Chương trình sẽ sử dụng phương thức `SetVariable` để gán cả hai biến hệ thống đó bằng 0 để tắt chế độ xuất hiện đóm sáng khi click chuột và nhắc lại dòng lệnh. Tiếp theo đó, thủ tục `drawout` và `drawtiles` sẽ vẽ đường đi dạo.

Kết thúc, phương thức `SetVariable` sẽ gán biến hệ thống trở về giá trị ban đầu.

Đề ý rằng, chỉ thủ tục này là không có từ khoá `Private` vì nó sẽ được gọi bởi người dùng, do đó ở đây bỏ qua từ khoá đó.

## 8. Duyệt mã lệnh

Bước tiếp theo sẽ thi hành Marco và sẽ duyệt qua từng dòng lệnh theo trình tự thực hiện của nó.

Từ menu `Tools` của chương trình AutoCAD chọn `Macro` ▶ `Run Macro`. Trong hộp thoại xuất hiện chọn `ThisDrawing.gardenpath` và chọn nút `Step`.

Môi trường VBA IDE sẽ xuất hiện trên màn hình và dòng đầu tiên của marco `gardenpath` được đánh dấu. Dòng đánh dấu là dòng chuẩn bị thi hành. Để thực thi

một dòng lệnh nào đó nhấn phím F8. Dòng lệnh tiếp theo được thi hành là thủ tục `gpuser`. Để nhảy vào trong thủ tục này thì nhấn phím F8 một lần nữa.

Để chạy qua từng dòng lệnh trong thủ tục `gpuser` thì nhấn phím F8. Dòng lệnh thứ nhất sẽ là phương thức `GetPoint`. Trước khi thực hiện dòng lệnh này, mở cửa sổ Locals bằng cách chọn View ▶ Locals Window. Cửa sổ này sẽ xuất hiện ở phía dưới của VBA IDE. Tất cả các biến cục bộ và giá trị của chúng sẽ xuất hiện trong cửa sổ này trong khi macro đang thi hành.

Nhấn F8 để thực hiện phương thức `GetPoint`. Để ý rằng, dòng lệnh đó sẽ không được đánh dấu và không có mã lệnh nào mới được đánh dấu. Bởi vì phương thức `GetPoint` đang chờ nhập vào một điểm từ AutoCAD. Chuyển về cửa sổ của chương trình AutoCAD, và ở dòng command sẽ xuất hiện dòng nhắc nhập điểm của phương thức `GetPoint`, chọn một điểm nào đó trên màn hình.

Trở lại môi trường VBA IDE, và để ý rằng dòng lệnh tiếp theo lời gọi phương thức `GetPoint` được đánh dấu. Tiếp tục thực hiện qua từng dòng lệnh bằng cách chọn phím F8 và chú ý chuyển sang cửa sổ AutoCAD khi cần nhập thông tin.

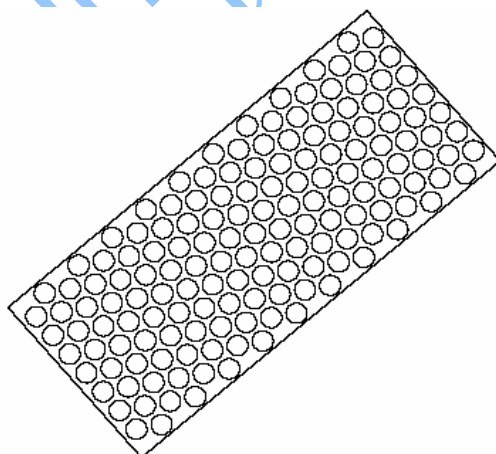
## 9. Thực thi Macro

Khi thực hiện Macro không cần thiết phải chạy qua từng dòng lệnh như các thao tác ở phần trên. Để chạy một Macro, chọn menu Tools ▶ Macro ▶ Run Macro, trong hộp thoại xuất hiện chọn nút Run. Khi đó chương trình sẽ được thực hiện như khi người dùng sử dụng. Chạy macro từ AutoCAD và theo các giá trị nhập vào như sau:

Tools ▶ Macro ▶ Run Macro: ***ThisDrawing.gardenpath***

Start point of the path: **2, 2**  
Endpoint of the path: **9, 8**  
Half width of the path: **.2**  
Radius of tiles: **.2**  
Spacing between tiles: **.1**

Với các số đầu vào này thì đường đi dạo sẽ được vẽ ra như sau:



Có thể nhập các bộ giá trị khác nhau, sử dụng chuột và bàn phím để thử nghiệm Macro này.

## 10. Thêm giao diện hộp thoại

Các hộp thoại giao diện cho macro của VBA được tạo ra trong VBA IDE.

Macro `gardenpath` nhận các thông số đầu vào từ dòng lệnh và có thể dễ dàng thêm hộp thoại giao diện cho nó. Các hộp thoại này tạo cho người dùng có các lựa chọn khác nhau trong nhiều lựa chọn.

Macro hiện tại có ít lựa chọn và cần được bổ sung thêm một số chi tiết. Các chi tiết này sẽ cho phép người dùng chỉ ra hình dạng của gạch lát trên đường. Có thể bắt đầu bằng cách sao chép lại phần mã lệnh của `gardenpath.dvb` thành một tệp khác lấy tên là `gdialog.dvb`.

### Sao chép dự án



- 1 Lưu dự án hiện tại trong VBA IDE
- 2 Dùng lệnh `SaveAs` trong VBA IDE và lưu dự án dưới tên là `gdialog.dvb`.

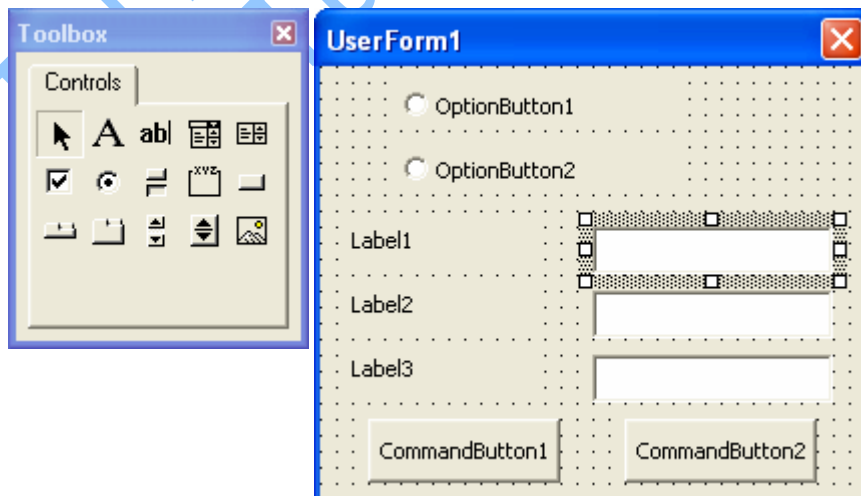
### 10.1. Tạo hộp thoại

Hộp thoại được tạo trong VBA IDE thông qua đối tượng Form.

Hộp thoại tạo ra sẽ gồm 2 tùy chọn để khi chọn mục này thì mục kia sẽ bị xoá đi, một để chọn gạch hình tròn và một để chọn gạch hình đa giác. Trên hộp thoại sẽ gồm 3 hộp ký tự để nhập các giá trị: bán kính của gạch, khoảng cách giữa các viên gạch và số cạnh của viên gạch (chỉ có khi chọn hình dạng gạch là đa giác)

#### Tạo hộp thoại từ VBA IDE

- 1 Mở một Form mới theo thao tác sau: menu `Insert` ▶ `User Form`. Sau đó sẽ xuất hiện hai cửa sổ, cửa sổ thứ nhất là hộp công cụ chứa các điều khiển để tạo hộp thoại, cửa sổ thứ hai là một Form trống để bố trí các điều khiển theo ý người lập trình.
- 2 Lựa chọn từng điều khiển từ hộp công cụ và đặt vào Form. Trên Form sẽ có 2 nút tùy chọn () , 3 nhãn (`A`), 3 hộp ký tự (`abl`), 2 nút lệnh () như mô tả ở hình dưới.



- 3 Đóng hộp công cụ.

### **Gán thuộc tính cho các nút tùy chọn**

- 1 Trên Form, chọn điều khiển `OptionButton1`. Trong cửa sổ Properties, thay đổi các thuộc tính dưới đây cho điều khiển đó (nếu cửa sổ Properties chưa mở thì thực hiện theo trình tự sau đây để mở: `Menu View\Properties Window`):

```
(Name) = gp_poly  
Caption = Polygon  
ControlTipText = Polygon Tile Shape  
Accelerator = P
```

- 2 Thực hiện như bước 1 đối với `OptionButton2`:

```
(Name) = gp_circ  
Caption = Circle  
ControlTipText = Circle Tile Shape  
Accelerator = I
```

### **Gán thuộc tính cho các nhãn**

- 1 Trên Form, chọn điều khiển `Label1`. Trong cửa sổ Properties, thay đổi các thuộc tính dưới đây cho các điều khiển đó:

```
(Name) = label_trad  
Caption = Radius of tiles  
TabStop = True
```

- 2 Thực hiện như bước 1 đối với `Label2`:

```
(Name) = label_tspac  
Caption = Space between tiles  
TabStop = True
```

- 3 Thực hiện như bước 1 đối với `Label3`:

```
(Name) = label_tsides  
Caption = Number of sides  
TabStop = True
```

### **Gán thuộc tính cho các hộp ký tự**

- 1 Trên Form, chọn điều khiển `TextBox1`. Trong cửa sổ Properties, thay đổi các thuộc tính dưới đây cho điều khiển `TextBox1`:

```
(Name) = gp_trad
```

- 2 Thực hiện như bước 1 đối với `TextBox2`:

```
(Name) = gp_tspac
```

- 3 Thực hiện như bước 1 đối với `TextBox3`:

```
(Name) = gp_tsides
```

### **Gán thuộc tính cho các nút lệnh và Form**

- 1 Trên Form, chọn điều khiển `CommandButton1`. Trong cửa sổ Properties, thay đổi các thuộc tính dưới đây cho điều khiển `CommandButton1`

```
(Name) = accept  
Caption = OK  
ControlTipText = Accept the options  
Accelerator = O  
Default = True
```

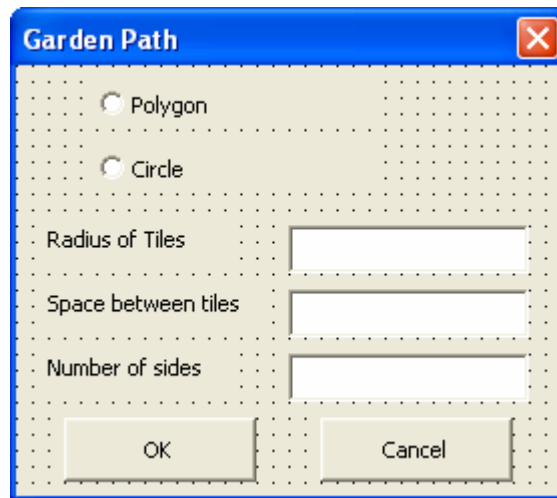
2 Thực hiện như bước 1 đối với CommandButton2:

```
(Name) = cancel  
Caption = Cancel  
ControlTipText = Cancel the operation  
Accelerator = C
```

3 Chọn Form bằng cách bấm chuột lên nền của nó tại vị trí không đặt điều khiển. Trong cửa sổ Properties, thay đổi thuộc tính của các thuộc tính sau:

```
(Name) = gpDialog  
Caption = Garden Path
```


Form thiết kế sẽ có dạng như sau:

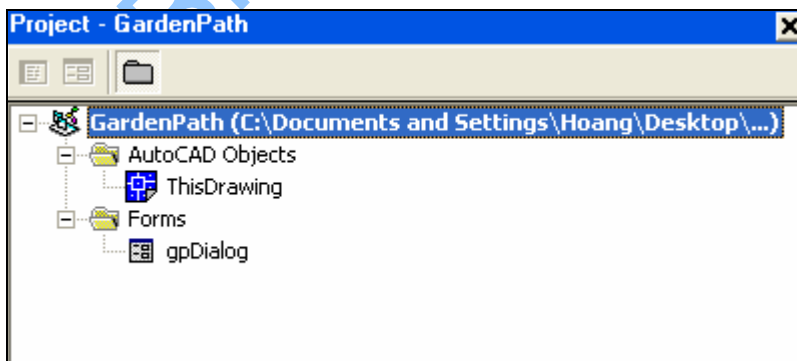


4 Lưu kết quả

## 10.2. Dùng cửa sổ Project để quản lý dự án

Trong cửa sổ Project của VBA IDE ta sẽ thấy tên và vị trí của dự án, một thư mục có tên là AutoCAD Objects và một thư mục có tên là Forms. Khi mở thư mục AutoCAD Objects (nó có thể đã được mở sẵn từ trước) ta sẽ thấy biểu tượng bản vẽ của AutoCAD có tên là ThisDrawing. Khi mở thư mục Forms (nó có thể đã được mở sẵn từ trước) ta sẽ thấy biểu tượng của Form và có tên là gpDialog. Đây là Form vừa được tạo ra.

Để xem phần mã lệnh của Form, chọn gpdialog trong cửa sổ Project, nhấn nút View Code (  ) hoặc phím F7.



Cửa sổ Code xuất hiện nhưng hầu như chưa có gì vì chưa nhập mã lệnh nào cho Form cả. Để trở lại Form, nhấn nút View Form trên cửa sổ Project hoặc nhấn tổ hợp phím SHIFT+F7.

Chọn `ThisDrawing` trong cửa sổ Project và xem phần mã lệnh bằng cách nhấn nút ViewCode, khi đó toàn bộ phần mã lệnh vừa nhập xuất hiện trong cửa sổ này.

Sử dụng cửa sổ Project để xác định vị trí mã lệnh và giúp người lập trình biết mình đang làm việc ở đâu một cách dễ dàng.

### 10.3. Cập nhật mã lệnh hiện có

Sau khi tạo hộp thoại với các điều khiển được sắp xếp theo ý người lập trình, bước tiếp theo sẽ viết mã lệnh mới để cho hộp thoại có thể làm việc.

Trước hết là sửa đổi mã lệnh đã có cho phù hợp. Bắt đầu từ mã lệnh cho `ThisDrawing`:

#### Cập nhật các biến toàn cục để sử dụng với hộp thoại

Cập nhật các dòng sau ở trong phần Declarations:

```
Public trad As Double ' cập nhật lại
Public tspac As Double ' cập nhật lại
Public tsides As Integer ' thêm mới
Public tshape As String ' thêm mới
```

Các biến `trad` và `tspac` được cập nhật lại phạm vi là `Public` thay vì `Private`. Các biến `Private` chỉ sử dụng được trong phạm vi mô-đun mà chúng được khai báo. Tuy nhiên ở phần này, Form cũng sẽ sử dụng hai biến đó nên chúng cần phải chuyển thành dạng `Public`. Bên cạnh đó, thêm hai biến mới là `tside` và `tshape`, tương ứng để chứa số cạnh của gạch hình đa giác và hình dạng của gạch lát do người dùng lựa chọn là hình tròn hay đa giác.

#### Cập nhật chương trình con `gpuser` để sử dụng với hộp thoại

Chuyển đến phần mã lệnh của chương trình con `gpuser`, xoá phần nhập bán kính gạch lát và khoảng cách giữa các viên gạch vì hai thông số này sẽ được nhập vào từ Form. Xoá các dòng lệnh sau:

```
trad = ThisDrawing.Utility._
GetDistance(sp, "Radius of tiles: ")
tspac = ThisDrawing.Utility._
GetDistance(sp, "Spacing between tiles: ")
```

Thêm các dòng lệnh để tải và hiển thị Form dưới đây thay vào vị trí của các dòng mã lệnh bị xoá ở trên:

```
Load gpDialog
gpDialog.Show
```

Sau khi sửa, chương trình con `gpuser` mới sẽ như sau:

```
' Thông tin cần thiết cho đường đi dạo
Private Sub gpuser()
    Dim varRet As Variant
    varRet = ThisDrawing.Utility.GetPoint( _
        , "Start point of path: ")
    sp(0) = varRet(0)
```

```

sp(1) = varRet(1)
sp(2) = varRet(2)
varRet = ThisDrawing.Utility.GetPoint( _
    , "Endpoint of path: ")
ep(0) = varRet(0)
ep(1) = varRet(1)
ep(2) = varRet(2)
hwidth = ThisDrawing.Utility. _
    GetDistance(sp, "Half width of path: ")
Load GPDdialog
GPDdialog.Show
pangle = ThisDrawing.Utility.AngleFromXAxis( _
    sp, ep)
totalwidth = 2 * hwidth
plength = distance(sp, ep)
angp90 = pangle + dtr(90)
angm90 = pangle - dtr(90)
End Sub

```

### Vẽ gạch lát là hình tròn hay đa giác

Tiếp theo cần thêm thủ tục để vẽ gạch lát hình tròn hay đa giác. Thêm vào đoạn mã lệnh sau vào cuối chương trình:

```

'Vẽ gạch lát với hình dạng theo thiết kế
Sub DrawShape(pltile)
    Dim angleSegment As Double
    Dim currentAngle As Double
    Dim angleInRadians As Double
    Dim currentSide As Integer
    Dim varRet As Variant
    Dim aCircle As AcadCircle
    Dim aPolygon As AcadLWPolyline
    ReDim points(1 To tsides * 2) As Double

    'Vẽ phụ thuộc vào kiểu hình dạng
    Select Case tshape
    Case "Circle"
        Set aCircle = ThisDrawing.ModelSpace. _
            AddCircle(pltile, trad)
    Case "Polygon"
        angleSegment = 360 / tsides
        currentAngle = 0
        For currentSide = 0 To (tsides - 1)
            angleInRadians = dtr(currentAngle)
            varRet = ThisDrawing.Utility.PolarPoint(pltile, _
                angleInRadians, trad)
            points((currentSide * 2) + 1) = varRet(0)
            points((currentSide * 2) + 2) = varRet(1)
            currentAngle = currentAngle + angleSegment
        Next currentSide
        Set aPolygon = ThisDrawing.ModelSpace. _
            AddLightWeightPolyline(points)
        aPolygon.Closed = True
    End Select
End Sub

```

Thủ tục trên sử dụng lệnh `Select Case` để điều khiển rẽ nhánh cho chương trình dựa vào kiểu hình dạng của viên gạch được xác định bởi biến `tshape`.



## Cập nhật thủ tục draw để vẽ gạch lát phù hợp

Tiếp theo, chuyển tới thủ tục draw, tìm dòng lệnh dưới đây:

```
Set cir = ThisDrawing.ModelSpace.AddCircle(pltile, trad)
```

và thay bằng dòng lệnh sau:

```
DrawShape (pltile)
```

## 10.4. Thêm mã lệnh cho hộp thoại

Các việc cần làm bây giờ là xoá các phần mã lệnh vẽ gạch lát hình tròn và thay bằng lời gọi thủ tục DrawShape để vẽ gạch có hình dạng theo lựa chọn của người dùng.

### Thêm xử lý sự kiện cho các nút tùy chọn

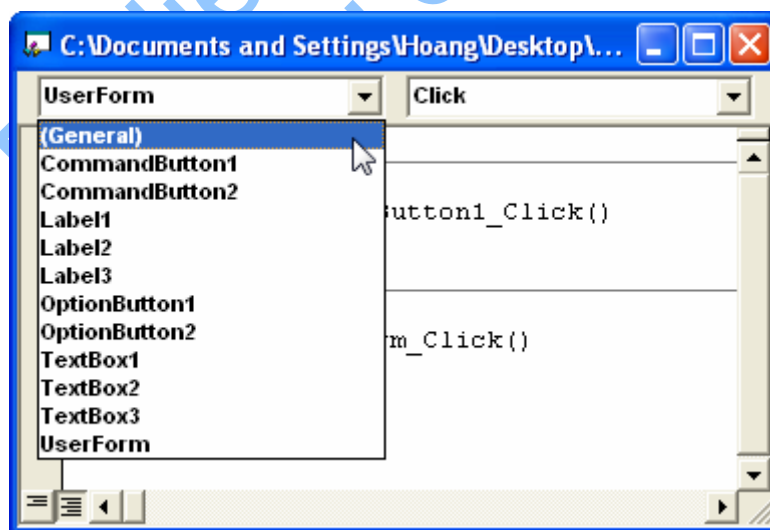
Mở cửa sổ Code của gpDialog và nhập đoạn mã lệnh sau dưới câu lệnh Option Explicit:

```
Private Sub gp_poly_Click()  
    gp_tsides.Enabled = True  
    ThisDrawing.tshape = "Polygon"  
End Sub
```

```
Private Sub gp_circ_Click()  
    gp_tsides.Enabled = False  
    ThisDrawing.tshape = "Circle"  
End Sub
```

Các thủ tục gp\_poly\_Click() và gp\_circ\_Click() được đặt tên theo hai điều khiển lựa chọn đã tạo trên Form trong phần trên kết hợp với từ "\_Click". Chúng là những thủ tục được tự động thực hiện khi người dùng bấm chuột vào điều khiển.

Lần lượt mở các hộp Object và Procedure/Event ở phía trên của cửa sổ Code, ta sẽ thấy danh sách bao gồm tên của tất cả các điều khiển đã chèn vào Form và được sắp xếp theo tên.



Tiếp theo, đặt con trỏ tại vị trí dòng Private Sub gp\_poly\_Click() và mở hộp Procedure/Event sẽ thấy một danh sách các sự kiện của điều khiển lựa chọn gp\_poly. Hai thủ tục đã được tạo ra để xử lý sự kiện Click. Ta cũng có thể thêm

mã lệnh để xử lý sự kiện `DbClick` để chương trình tự động thực hiện khi người dùng bấm đúp chuột trên điều khiển. Ta có thể thêm mã lệnh cho bất cứ sự kiện nào của điều khiển được liệt kê trong danh sách. Các kiểu thủ tục như vậy được gọi là xử lý sự kiện.

Xét mã lệnh của hai thủ tục xử lý sự kiện vừa tạo ra. Thủ tục xử lý sự kiện đầu tiên phản ứng với sự kiện `Click` của điều khiển `gp_poly`, dòng mã lệnh đầu tiên là để hộp ký tự chứa số cạnh của đa giác hoạt động. Hộp ký tự này chỉ làm việc khi hình dạng gạch lựa chọn là đa giác. Dòng lệnh tiếp theo là gán giá trị `Polygon` cho biến `tshape`.

Thủ tục xử lý sự kiện thứ hai để phản ứng với sự kiện `Click` của điều khiển `gp_circ`. Thủ tục này sẽ làm mất hiệu lực của hộp ký tự chứa số cạnh của đa giác và gán biến `tshape` theo giá trị của `Circle`.

### Thêm xử lý sự kiện cho nút OK

Thêm đoạn mã lệnh sau để xử lý sự kiện của nút OK:

```
Private Sub accept_Click()  
    If ThisDrawing.tshape = "Polygon" Then  
        ThisDrawing.tsides = Cint(gp_tsides.text)  
        If (ThisDrawing.tsides < 3#) Or  
            (ThisDrawing.tsides > 1024#) Then  
            MsgBox "Enter a value between 3 and " & _  
                "1024 for the number of sides."  
            Exit Sub  
        End If  
    End If  
  
    ThisDrawing.trad = Cdbl(gp_trad.text)  
    ThisDrawing.tspac = Cdbl(gp_tspac.text)  
    If ThisDrawing.trad < 0# Then  
        MsgBox "Enter a positive value for the radius."  
        Exit Sub  
    End If  
  
    If (ThisDrawing.tspac < 0#) Then  
        MsgBox "Enter a positive value for the spacing."  
        Exit Sub  
    End If  
    GPDialog.Hide  
End Sub
```

Đến đây, các công việc cần thiết cho Form đã hoàn thiện. Xử lý sự kiện này, trước hết sẽ kiểm tra xem lựa chọn cuối cùng có phải là đa giác không. Nếu lựa chọn là đa giác thì nó sẽ nhận giá trị số cạnh của đa giác từ điều khiển `gp_tsides`. Giá trị người dùng nhập vào được lưu trong thuộc tính `Text` và là kiểu chuỗi, nên cần chuyển sang kiểu số nguyên bằng cách sử dụng hàm `Cint` của Visual Basic. Sau đó, xử lý sự kiện sẽ kiểm tra giá trị có nằm trong phạm vi hợp lệ là từ 3÷1024 không. Nếu không thì sẽ xuất hiện một hộp thông báo và xử lý sự kiện sẽ kết thúc. Khi đó người dùng sẽ có cơ hội để thay đổi giá trị nhập vào.

Nhấn nút OK một lần nữa thì xử lý sự kiện sẽ được bắt đầu và kiểm tra lại giá trị nhập vào.

Giá trị bán kính và khoảng cách giữa các viên gạch cũng được nhận theo cách tương tự trên ngoại trừ kiểu giá trị của chúng là `double` chứ không phải là `integer`, hàm sử dụng là `Cdbl`. Chúng cũng được kiểm tra để đảm bảo mang giá trị dương.

Khi các giá trị đã được nhập vào và kiểm tra thì lệnh `gpDialog.Hide` sẽ làm ẩn Form, do vậy cần thông qua các điều khiển để trở lại thủ tục đầu tiên gọi Form.

### Thêm xử lý sự kiện cho nút Cancel

Thêm đoạn mã lệnh sau cho xử lý sự kiện của nút Cancel:

```
Private Sub cancel_Click()  
    Unload Me  
End  
End Sub
```

Đây là một đoạn xử lý sự kiện đơn giản để dỡ bỏ Form và kết thúc Macro. Và còn một sự kiện nữa chưa xử lý là tạo các giá trị khởi tạo cho Form. Đó là sự kiện `Initialize` của Form. Sự kiện này được thi hành khi Form được tải lần đầu tiên.

### Thêm xử lý cho sự kiện khởi tạo của Form

Thêm đoạn mã xử lý sự kiện dưới đây cho sự kiện `Initialize` của Form:

```
Private Sub UserForm_Initialize()  
    gp_circ.Value = True  
    gp_trad.Text = ".2"  
    gp_tspac.Text = ".1"  
    gp_tsides.Text = "5"  
    gp_tsides.Enabled = False  
    ThisDrawing.tsides = 5  
End Sub
```

Đoạn mã lệnh này sẽ gán các giá trị ban đầu cho Form và cho biến `tsides`, biến này phải nhận giá trị dương và lớn hơn 3 ngay cả trong trường hợp chọn một đường tròn. Để hiểu lý do, tham khảo thủ tục `DrawShape` ở phần trên. Biến `point` được định nghĩa thông qua số cạnh của đa giác, biến này sẽ được cấp phát bộ nhớ mặc dù không chọn hình dạng gạch lát là đa giác. Do đó, biến `tside` phải được xác định trong một phạm vi hợp lý. Người dùng được tự do thay đổi các giá trị trong khi Macro thực hiện.

Bây giờ Macro này đã sẵn sàng hoạt động.

Tài liệu tham khảo  
BM Tự động hoá TKCĐ

## SO SÁNH Visual LISP VÀ ActiveX/VBA

Hầu hết các khả năng của các giao diện Visual LISP đều có trong giao diện của ActiveX/VBA. Bảng so sánh trong chương này dùng để tham khảo cho những người đã quen lập trình với Visual LISP tìm kiếm các chức năng tương ứng trong ActiveX/VBA.

Trong phụ lục này

A

- So sánh Visual LISP và ActiveX/VBA

# 1. So sánh Visual LISP và ActiveX/VBA

Bảng sau đây liệt kê các so sánh tương đương giữa các hàm và các toán tử trong AutoLISP với ActiveX/VBA. Các ActiveX Automation tương đương được chỉ ra bằng cụm từ “AutoCAD.Application.”

## Các thiết lập cho Paper space, model space và TILEMODE

Hàm của AutoLISP	Tương đương trong ActiveX/Visual Basic
+ (cộng)	+ (toán tử cộng)
- (trừ)	- (toán tử trừ)
* (nhân)	* (toán tử nhân)
/ (chia)	/ (toán tử chia)
= (bằng)	= (toán tử so sánh bằng)
/= (khác)	<> (toán tử so sánh khác)
< (nhỏ hơn)	< (toán tử so sánh nhỏ hơn)
<= (nhỏ hơn hoặc bằng)	<= (toán tử so sánh nhỏ hơn hoặc bằng)
> (lớn hơn)	> (toán tử so sánh lớn hơn)
>= (lớn hơn hoặc bằng)	>= (toán tử so sánh lớn hơn hoặc bằng)
~ (phủ định bit)	Toán tử Not
1+ (tăng)	Dùng dấu + (toán tử cộng)
1- (giảm)	Dùng dấu - (toán tử trừ)
abs	Hàm Abs
acad_colordlg	<i>Không có</i>
acad_helpdlg	Tìm HELP trong mục online Help
acad_strlsort	Tìm SORT trong mục online Help
action_tile	Sử dụng Visual Basic Dialog Editor
add_list	Sử dụng Visual Basic Dialog Editor
ads	Phương thức AutoCAD.Application.ListADS
alert	Hàm MsgBox
and	Toán tử And

## Các thiết lập cho Paper space, model space và TILEMODE

Hàm của AutoLISP	Tương đương trong ActiveX/Visual Basic
angle	Phương thức AutoCAD.Application.ActiveDocument.Utility.AngleFromXAxis
angtof	AutoCAD.Application.ActiveDocument.Utility.AngleToReal
angtos	AutoCAD.Application.ActiveDocument.Utility.AngleToString
append	Sử dụng các hàm xử lý mảng của Visual Basic
apply	<i>Không có</i>
arx	AutoCAD.Application.ListARX
arxload	AutoCAD.Application.LoadARX
arxunload	AutoCAD.Application.UnloadARX
ascii	Hàm Asc
assoc	<i>Không có</i>
atan	Hàm Atn
atof	Hàm CDbI
atoi	Hàm CInt
atom	Tìm kiếm IS trong mục online Help
atoms-family	<i>Không có</i>
autoarxload	<i>Không có</i>
autoload	<i>Không có</i>
Boole	Sử dụng phép toán logic của Visual Basic
boundp	Tìm kiếm IS trong mục online Help
car/cdr	Sử dụng các hàm xử lý mảng của Visual Basic
chr	Hàm Chr
client_data_tile	Sử dụng Visual Basic Dialog Editor
close	AutoCAD.Application.Documents.Close
command	AutoCAD.ActiveDocument.SendCommand



## Các thiết lập cho Paper space, model space và TILEMODE

Hàm của AutoLISP	Tương đương trong ActiveX/Visual Basic
cond	Câu lệnh Select Case
cons	Sử dụng hàm xử lý mảng hoặc phương thức AutoCAD.Application.collection.Add<tên thực thể>
cos	Hàm Cos
cvunit	Sử dụng các hàm đảo
defun	Từ khoá của Visual Basic: Function và End Function
dictadd	AutoCAD.Application.ActiveDocument.Dictionaries.Add
dictnext	AutoCAD.Application.ActiveDocument.Dictionaries.Item
dictremove	AutoCAD.Application.ActiveDocument.Dictionaries.Dictionary.Delete
dictrename	AutoCAD.Application.ActiveDocument.Dictionaries.Dictionary.Rename
dictsearch	AutoCAD.Application.ActiveDocument.Dictionaries.Dictionary.GetName and GetObject
dimx_tile và dimy_tile	Sử dụng Visual Basic Dialog Editor
distance	Sử dụng phương thức tương tác AutoCAD.Application.Utility.GetDistance. Tham khảo thêm phần "Tính khoảng cách giữa hai điểm".
distof	<i>Không có</i>
done_dialog	Sử dụng Visual Basic Dialog Editor
end_image	Sử dụng Visual Basic Dialog Editor
end_list	Sử dụng Visual Basic Dialog Editor
entdel	AutoCAD.Application.ActiveDocument.collection_object.Delete
entget	AutoCAD.Application.ActiveDocument.collection_object.property
entlast	AutoCAD.Application.ActiveDocument.Modelspace.Item(count-1)
entmake	AutoCAD.Application.ActiveDocument.Modelspace.Add<tên thực thể>
entmakex	AutoCAD.Application.ActiveDocument.Modelspace.Add<entitynam

## Các thiết lập cho Paper space, model space và TILEMODE

Hàm của AutoLISP	Tương đương trong ActiveX/Visual Basic
	e>
entmod	Sử dụng bất kỳ thuộc tính đọc-ghi nào của đối tượng
entnext	AutoCAD.Application.ActiveDocument.collection.Item
entsel	AutoCAD.Application.ActiveDocument.SelectionSets đối tượng/phương thức/thuộc tính
entupd	AutoCAD.Application.ActiveDocument.Modelspace.object.Update
eq	<i>Không có</i>
equal	Toán tử Eqv
*error*	đối tượng/phương thức/thuộc tính Error
eval	<i>Không có</i>
exit	AutoCAD.Application.Quit
exp	Hàm Exp
expand	<i>Không có</i>
expt	^ (Phép toán lũy thừa)
fill_image	Sử dụng Visual Basic Dialog Editor
findtệp	Hàm Dir
fix	Hàm Fix, Int, Cint
float	Hàm CDbI
foreach	Câu lệnh For Each...Next
gc	AutoCAD.Application.ActiveDocument.PurgeAll
gcd	<i>Không có</i>
get_attr	Sử dụng Visual Basic Dialog Editor
get_tile	Sử dụng Visual Basic Dialog Editor
getangle	AutoCAD.Application.ActiveDocument.Utility.GetAngle
getcfg	AutoCAD.Application.Preferences.property
getcname	<i>Không có</i>

## Các thiết lập cho Paper space, model space và TILEMODE

Hàm của AutoLISP	Tương đương trong ActiveX/Visual Basic
getcorner	AutoCAD.Application.ActiveDocument.Utility.GetCorner
getdist	AutoCAD.Application.ActiveDocument.Utility.GetDistance
getenv	AutoCAD.Application.Preferences.property
gettệpd	Sử dụng hộp thoại tệp của Visual Basic
getint	AutoCAD.Application.ActiveDocument.Utility.GetInteger
getkword	AutoCAD.Application.ActiveDocument.Utility.GetKeyword
getorient	AutoCAD.Application.ActiveDocument.Utility.GetOrientation
getpoint	AutoCAD.Application.ActiveDocument.Utility.GetPoint
getreal	AutoCAD.Application.ActiveDocument.Utility.GetReal
getstring	AutoCAD.Application.ActiveDocument.Utility.GetString
getvar	AutoCAD.Application.GetVariable
graphscr	AppActivate AutoCAD.Application.Caption
grclear	<i>Hàm Obsolete (Hàm không còn được sử dụng)</i>
grdraw	<i>Không có</i>
grread	<i>Không có</i>
grtext	AutoCAD.Application.ActiveDocument.Utility.Prompt
grvecs	<i>Không có</i>
handent	AutoCAD.Application.ActiveDocument.ModelSpace.object.Handle
help	Tìm HELP trong mục online Help
if	Câu lệnh If... Then... Else
initget	AutoCAD.Application.ActiveDocument.Utility.InitializeUserInput
inters	AutoCAD.Application.ActiveDocument.ModelSpace.object.Intersect With
itoa	Hàm Str
lambda	<i>Không có</i>

## Các thiết lập cho Paper space, model space và TILEMODE

Hàm của AutoLISP	Tương đương trong ActiveX/Visual Basic
last	Tên mảng(UBound(ten_mang))
length	Hàm Ubound
list	Câu lệnh ReDim
listp	Hàm IsArray
load_dialog	Sử dụng Visual Basic Dialog Editor
load	AutoLISP không được hỗ trợ qua Automation
log	Hàm Log
logand	Hàm And
logior	Hàm Or
lsh	Hàm Imp
mapcar	<i>Không có</i>
max	Hàm Max
mem	<i>Không có</i>
member	Sử dụng tập hợp
menucmd	Đối tượng AutoCAD.Application.MenuBar
menugroup	Đối tượng AutoCAD.Application.MenuGroup
min	Hàm Min
minusp	Dùng cú pháp < 0
mode_tile	Sử dụng Visual Basic Dialog Editor
namedobjdict	AutoCAD.Application.ActiveDocument.Dictionaries
nentsel	AutoCAD.Application.ActiveDocument.SelectionSets.SelectionSet.SelectAtPoint
nentselp	AutoCAD.Application.ActiveDocument.SelectionSets.SelectionSet.SelectAtPoint
new_dialog	Sử dụng Visual Basic Dialog Editor
not	Sử dụng phép toán logic

## Các thiết lập cho Paper space, model space và TILEMODE

Hàm của AutoLISP	Tương đương trong ActiveX/Visual Basic
nth	Sử dụng cú pháp doi_tuong(n)
null	Hàm IsNull
numberp	Hàm TypeName
open	Hàm Open
or	Sử dụng phép toán logic
osnap	Không có (Có thể dùng phương thức SetVariable để điều khiển biến hệ thống OSMODE.)
polar	AutoCAD.Application.ActiveDocument.Utility.PolarPoint
prin1	AutoCAD.Application.ActiveDocument.Utility.Prompt
princ	AutoCAD.Application.ActiveDocument.Utility.Prompt
print	AutoCAD.Application.ActiveDocument.Utility.Prompt
progn	<i>Không có</i>
prompt	AutoCAD.Application.ActiveDocument.Utility.Prompt
quit	AutoCAD.Application.Quit
quote	<i>Không có</i>
read	<i>Không có</i>
read-char	Hàm Input
read-line	Hàm Line Input
redraw	AutoCAD.Application.ActiveDocument.ModelSpace.Object.Update
regapp	AutoCAD.Application.ActiveDocument.RegisteredApplications.Add
rem	Hàm Mod
repeat	For...Each, While,
reverse	<i>Không có</i>
rtos	AutoCAD.Application.ActiveDocument.Utility.RealToString
set	Hàm Set

## Các thiết lập cho Paper space, model space và TILEMODE

Hàm của AutoLISP	Tương đương trong ActiveX/Visual Basic
set_tile	Sử dụng Visual Basic Dialog Editor
setcfg	AutoCAD.Application.Preferences.property
setfunhelp	<i>Không có</i>
setq	Hàm Set
setvar	AutoCAD.Application.SetVariable method
sin	Hàm sin
setview	AutoCAD.Application.ActiveDocument.Viewports.Viewport.SetView
slide_image	Sử dụng Visual Basic Dialog Editor
snvalid	<i>Không có</i>
sqrt	Hàm Sqr
ssadd	AutoCAD.Application.ActiveDocument.SelectionSets.Add
ssdel	AutoCAD.Application.ActiveDocument.SelectionSets.SelectionSet.Delete
ssget	AutoCAD.Application.ActiveDocument.SelectionSets.SelectionSet.SelectOnScreen
ssgetfirst	<i>Không có</i>
sslength	AutoCAD.Application.ActiveDocument.SelectionSets.SelectionSet.Count
ssmemb	So sánh ID của đối tượng với các đối tượng trong SelectionSet
ssname	AutoCAD.Application.ActiveDocument.SelectionSets.SelectionSet.Name
ssnamex	<i>Không có</i>
sssetfirst	AutoCAD.Application.ActiveDocument.PickfirstSelectionSet
Startapp`	Hàm Shell
start_dialog	Sử dụng Visual Basic Dialog Editor
start_image	Sử dụng Visual Basic Dialog Editor
start_list	Sử dụng Visual Basic Dialog Editor

## Các thiết lập cho Paper space, model space và TILEMODE

Hàm của AutoLISP	Tương đương trong ActiveX/Visual Basic
strcase	Hàm StrConv
strcat	Phép toán &
strlen	Hàm Len
subst	<i>Không có</i>
substr	Hàm Mid
tablet	<i>Không có</i>
tblnext	AutoCAD.Application.ActiveDocument.collection_object.Item
tblobjname	AutoCAD.Application.ActiveDocument.collection_object.Name
tblsearch	AutoCAD.Application.ActiveDocument.collection_object.Name
term_dialog	Sử dụng Visual Basic Dialog Editor
terpri	<i>Không có</i>
textbox	AutoCAD.Application.ActiveDocument.space.object.GetBoundingBox
textpage	<i>Không có</i>
textscr	<i>Không có</i>
trace	<i>Không có</i>
trans	AutoCAD.Application.ActiveDocument.Utility.TranslateCoordinates
type	Hàm TypeName
unload_dialog	Sử dụng Visual Basic Dialog Editor
untrace	<i>Không có</i>
vector_image	Sử dụng Visual Basic Dialog Editor
ver	AutoCAD.Application.Version
vports	Tập đối tượng AutoCAD.Application.ActiveDocument.Viewports
wcmatch	Phép toán Like
while	While...Wend



## Các thiết lập cho Paper space, model space và TILEMODE

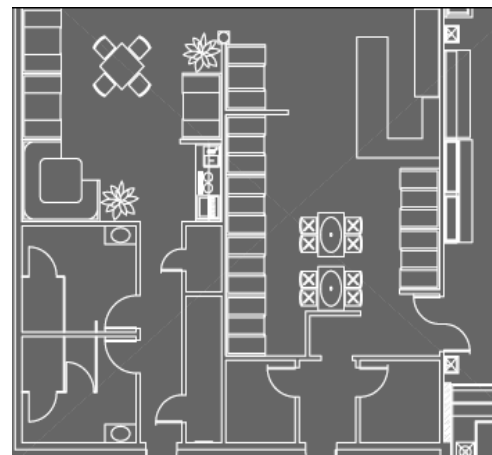
Hàm của AutoLISP	Tương đương trong ActiveX/Visual Basic
write-char	Hàm Print
write-line	Hàm Print
xdroom	<i>Không có</i>
xdsiz	<i>Không có</i>
zerop	Sử dụng cú pháp = 0

Tài liệu tham khảo  
BM Tự động hoá TKCĐ

Tài liệu tham khảo  
BM Tự động hoá TKCĐ

# CHUYỂN ĐỔI TỪ AutoCAD PHIÊN BẢN 14.01

Trong phụ lục này sẽ liệt kê danh sách các đối tượng, thuộc tính, phương thức và sự kiện mới của một số giao diện ActiveX Automation. Đồng thời, ở đây cũng liệt kê các phương thức và thuộc tính bị thay đổi hoặc đã bị xoá khỏi hệ thống.



Trong phụ lục này

**B**

- Mục mới cập nhật
- Mục đã thay đổi
- Mục đã loại bỏ

# 1. Mục mới cập nhật

Trong phần này sẽ liệt kê các đối tượng ActiveX cùng với các thuộc tính, phương thức và sự kiện mới của chúng.

Danh sách liệt kê bao gồm các đối tượng ActiveX đã có, đối tượng mới và cả các đối tượng dạng ẩn (Các đối tượng dạng ẩn là các giao diện COM nhưng không phải là đối tượng của VBA. Phương thức và thuộc tính của các đối tượng này được sử dụng trong VBA thông qua các đối tượng của VBA).

Trong bảng dưới đây, các đối tượng mới được đánh dấu sao (\*) và các đối tượng ẩn được đánh dấu cộng (+).

## Phương thức, thuộc tính và sự kiện mới kể từ AutoCAD phiên bản 14.01

Đối tượng	Phương thức/ Thuộc tính / Sự kiện
3dFace	Coordinate, Coordinates, Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName, VisibilityEdge1, VisibilityEdge2, VisibilityEdge3, VisibilityEdge4
3dpolyline	Coordinate, Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName, Type
3dSolid	Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName
Application	AppActivate, AppDeactivate, ARXLoaded, ARXUnloaded, BeginCommand, BeginFileDrop, BeginLisp, BeginModal, BeginOpen, BeginPlot, BeginQuit, BeginSave, Documents, EndCommand, EndLisp, EndModal, EndOpen, EndPlot, EndSave, Eval, GetAcadState, LispCancelled, LoadDVB, MenuBar, MenuGroups, NewDrawing, RunMacro, StatusId, SysVarChanged, UnloadDVB, VBE, WindowChanged, WindowLeft, WindowMovedOrResized, WindowState, WindowTop, ZoomAll, ZoomCenter, ZoomExtents, ZoomPickWindow, ZoomScaled, ZoomWindow, ZoomPrevious
Arc	ArcLength, Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName, TotalAngle
Attribute	Alignment, Backward, Constant, Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Invisible, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName, Preset, UpsideDown, Verify
AttributeReference	Alignment, ArrayPolar, ArrayRectangular, Backward, Constant, Copy, Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Invisible, Lineweight,

## Phương thức, thuộc tính và sự kiện mới kể từ AutoCAD phiên bản 14.01

Đối tượng	Phương thức/ Thuộc tính / Sự kiện
	Mirror, Mirror3D, Modified, ObjectName, OwnerID, PlotStyleName, UpsideDown
Block	AddDim3PointAngular, AddMInsertBlock, AddMLine, AddPolyfaceMesh, AttachExternalReference, Bind, Detach, Document, GetExtensionDictionary, GetXData, HasExtensionDictionary, IsLayout, IsXRef, Layout, Modified, ObjectID, ObjectName, OwnerID, Reload, SetXData, Unload, XrefDatabase
BlockReference	Delete, Document, GetConstantAttributes, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName
Blocks	Delete, Document, GetExtensionDictionary, GetXData, Handle, HasExtensionDictionary, Modified, ObjectID, ObjectName, OwnerID, SetXData
Circle	Circumference, Delete, Diameter, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName
Database *	Blocks, CopyFrom, CopyObjects, Dictionaries, DimStyles, ElevationModelspace, ElevationPaperspace, Groups, HandleToObject, Layers, Layouts, Limits, Linetypes, ModelSpace, ObjectIdToObject, PaperSpace, PlotConfigurations, Preferences, RegisteredApplications, TextStyles, UserCoordinateSystems, Viewports, Views
DatabasePreferences *	AllowLongSymbolNames, Application, ContourLinesPerSurface, DisplaySilhouette, Lineweight, LineWeightDisplay, MaxActiveViewports, ObjectSortByPlotting, ObjectSortByPSOutput, ObjectSortByRedraws, ObjectSortByRegens, ObjectSortBySelection, ObjectSortBySnap, OLELaunch, RenderSmoothness, SegmentPerPolyline, SolidFill, TextFrameDisplay, XRefEdit, XRefLayerVisibility
Dictionaries	Delete, Document, GetExtensionDictionary, GetXData, Handle, HasExtensionDictionary, Modified, ObjectID, ObjectName, OwnerID, SetXData
Dictionary	AddXRecord, Count, Document, GetExtensionDictionary, HasExtensionDictionary, Item, Modified, ObjectID, ObjectName, OwnerID
Dim3PointAngular *	AngleFormat, AngleVertex, Application, ArrayPolar, ArrayRectangular, Arrowhead1Block, Arrowhead1Type, Arrowhead2Block, Arrowhead2Type, ArrowheadSize, Color, Copy, DecimalSeparator, Delete, DimensionLineColor, DimensionLineWeight, DimLine1Suppress, DimLine2Suppress,

## Phương thức, thuộc tính và sự kiện mới kể từ AutoCAD phiên bản 14.01

Đối tượng	Phương thức/ Thuộc tính / Sự kiện
	DimLineInside, Document, ExtensionLineColor, ExtensionLineExtend, ExtensionLineOffset, ExtensionLineWeight, ExtLine1EndPoint, ExtLine1Suppress, ExtLine2EndPoint, ExtLine2Suppress, Fit, ForceLineInside, GetBoundingBox, GetExtensionDictionary, GetXdata, Handle, HasExtensionDictionary, Highlight, HorizontalTextPosition, Hyperlinks, IntersectWith, Layer, Linetype, LinetypeScale, Lineweight, Measurement, Mirror, Mirror3D, Modified, Move, Normal, ObjectID, ObjectName, OwnerID, PlotStyleName, Rotate, Rotate3D, Rotation, ScaleEntity, ScaleFactor, SetXdata, StyleName, SuppressLeadingZeros, SuppressTrailingZeros, TextColor, TextGap, TextHeight, TextInside, TextInsideAlign, TextMovement, TextOutsideAlign, TextOverride, TextPosition, TextPrecision, TextPrefix, TextRotation, TextStyle, TextSuffix, ToleranceDisplay, ToleranceHeightScale, ToleranceJustification, ToleranceLowerLimit, TolerancePrecision, ToleranceSuppressLeadingZeros, ToleranceSuppressTrailingZeros, ToleranceUpperLimit, TransformBy, Update, VerticalTextPosition, Visible
DimAligned	AltRoundDistance, AltSuppressLeadingZeros, AltSuppressTrailingZeros, AltSuppressZeroFeet, AltSuppressZeroInches, AltTextPrefix, AltTextSuffix, AltTolerancePrecision, AltToleranceSuppressLeadingZeros, AltToleranceSuppressTrailingZeros, AltToleranceSuppressZeroFeet, AltToleranceSuppressZeroInches, AltUnits, AltUnitsFormat, AltUnitsPrecision, AltUnitsScale, Arrowhead1Block, Arrowhead1Type, Arrowhead2Block, Arrowhead2Type, ArrowheadSize, DecimalSeparator, Delete, DimensionLineColor, DimensionLineExtend, DimensionLineWeight, DimLine1Suppress, DimLine2Suppress, DimLineInside, Document, ExtensionLineColor, ExtensionLineExtend, ExtensionLineOffset, ExtensionLineWeight, ExtLine1Suppress, ExtLine2Suppress, Fit, ForceLineInside, FractionFormat, GetExtensionDictionary, HasExtensionDictionary, HorizontalTextPosition, Hyperlinks, LinearScaleFactor, Lineweight, Measurement, Modified, ObjectName, OwnerID, PlotStyleName, PrimaryUnitsPrecision, RoundDistance, ScaleFactor, SuppressLeadingZeros, SuppressTrailingZeros, SuppressZeroFeet, SuppressZeroInches, TextColor, TextGap, TextHeight, TextInside, TextInsideAlign, TextMovement, TextOutsideAlign, TextOverride, TextPrefix, TextStyle, TextSuffix, ToleranceDisplay, ToleranceHeightScale, ToleranceJustification, ToleranceLowerLimit, TolerancePrecision, ToleranceSuppressLeadingZeros, ToleranceSuppressTrailingZeros, ToleranceSuppressZeroFeet, ToleranceSuppressZeroInches, ToleranceUpperLimit, UnitsFormat, VerticalTextPosition
DimAngular	AngleFormat, Arrowhead1Block, Arrowhead1Type, Arrowhead2Block, Arrowhead2Type, ArrowheadSize,

## Phương thức, thuộc tính và sự kiện mới kể từ AutoCAD phiên bản 14.01

Đối tượng	Phương thức/ Thuộc tính / Sự kiện
	DecimalSeparator, Delete, DimensionLineColor, DimensionLineWeight, DimLine1Suppress, DimLine2Suppress, DimLineInside, Document, ExtensionLineColor, ExtensionLineExtend, ExtensionLineOffset, ExtensionLineWeight, ExtLine1Suppress, ExtLine2Suppress, Fit, ForceLineInside, GetExtensionDictionary, HasExtensionDictionary, HorizontalTextPosition, Hyperlinks, Lineweight, Measurement, Modified, ObjectName, OwnerID, PlotStyleName, ScaleFactor, SuppressLeadingZeros, SuppressTrailingZeros, TextColor, TextGap, TextHeight, TextInside, TextInsideAlign, TextMovement, TextOutsideAlign, TextOverride, TextPrecision, TextPrefix, TextStyle, TextSuffix, ToleranceDisplay, ToleranceHeightScale, ToleranceJustification, ToleranceLowerLimit, TolerancePrecision, ToleranceSuppressLeadingZeros, ToleranceSuppressTrailingZeros, ToleranceUpperLimit, VerticalTextPosition
DimDiametric	AltRoundDistance, AltSuppressLeadingZeros, AltSuppressTrailingZeros, AltSuppressZeroFeet, AltSuppressZeroInches, AltTextPrefix, AltTextSuffix, AltTolerancePrecision, AltToleranceSuppressLeadingZeros, AltToleranceSuppressTrailingZeros, AltToleranceSuppressZeroFeet, AltToleranceSuppressZeroInches, AltUnits, AltUnitsFormat, AltUnitsPrecision, AltUnitsScale, Arrowhead1Block, Arrowhead1Type, Arrowhead2Block, Arrowhead2Type, ArrowheadSize, CenterMarkSize, CenterType, DecimalSeparator, Delete, DimensionLineColor, DimensionLineWeight, DimLine1Suppress, DimLine2Suppress, Document, Fit, ForceLineInside, FractionFormat, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, LinearScaleFactor, Lineweight, Measurement, Modified, ObjectName, OwnerID, PlotStyleName, PrimaryUnitsPrecision, RoundDistance, ScaleFactor, SuppressLeadingZeros, SuppressTrailingZeros, SuppressZeroFeet, SuppressZeroInches, TextColor, TextGap, TextHeight, TextInside, TextInsideAlign, TextMovement, TextOutsideAlign, TextOverride, TextPrefix, TextStyle, TextSuffix, ToleranceDisplay, ToleranceHeightScale, ToleranceJustification, ToleranceLowerLimit, TolerancePrecision, ToleranceSuppressLeadingZeros, ToleranceSuppressTrailingZeros, ToleranceSuppressZeroFeet, ToleranceSuppressZeroInches, ToleranceUpperLimit, UnitsFormat, VerticalTextPosition
Dimension +	Application, ArrayPolar, ArrayRectangular, Color, Copy, DecimalSeparator, Delete, Document, GetBoundingBox, GetExtensionDictionary, GetXdata, Handle, HasExtensionDictionary, Highlight, Hyperlinks, IntersectWith, Layer, Linetype, LinetypeScale, Lineweight, Mirror, Mirror3D, Modified, Move, Normal, ObjectID, ObjectName, OwnerID,



## Phương thức, thuộc tính và sự kiện mới kể từ AutoCAD phiên bản 14.01

Đối tượng	Phương thức/ Thuộc tính / Sự kiện
	PlotStyleName, Rotate, Rotate3D, Rotation, ScaleEntity, ScaleFactor, SetXdata, StyleName, SuppressLeadingZeros, SuppressTrailingZeros, TextColor, TextGap, TextHeight, TextMovement, TextOverride, TextPosition, TextPrefix, TextRotation, TextStyle, TextSuffix, ToleranceDisplay, ToleranceHeightScale, ToleranceJustification, ToleranceLowerLimit, TolerancePrecision, ToleranceSuppressLeadingZeros, ToleranceSuppressTrailingZeros, ToleranceUpperLimit, TransformBy, Update, VerticalTextPosition, Visible
DimOrdinate	AltRoundDistance, AltSuppressLeadingZeros, AltSuppressTrailingZeros, AltSuppressZeroFeet, AltSuppressZeroInches, AltTextPrefix, AltTextSuffix, AltTolerancePrecision, AltToleranceSuppressLeadingZeros, AltToleranceSuppressTrailingZeros, AltToleranceSuppressZeroFeet, AltToleranceSuppressZeroInches, AltUnits, AltUnitsFormat, AltUnitsPrecision, AltUnitsScale, ArrowheadSize, DecimalSeparator, Delete, Document, ExtensionLineColor, ExtensionLineOffset, ExtensionLineWeight, FractionFormat, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, LinearScaleFactor, Lineweight, Measurement, Modified, ObjectName, OwnerID, PlotStyleName, PrimaryUnitsPrecision, RoundDistance, ScaleFactor, SuppressLeadingZeros, SuppressTrailingZeros, SuppressZeroFeet, SuppressZeroInches, TextColor, TextGap, TextHeight, TextMovement, TextOverride, TextPrefix, TextStyle, TextSuffix, ToleranceDisplay, ToleranceHeightScale, ToleranceJustification, ToleranceLowerLimit, TolerancePrecision, ToleranceSuppressLeadingZeros, ToleranceSuppressTrailingZeros, ToleranceSuppressZeroFeet, ToleranceSuppressZeroInches, ToleranceUpperLimit, UnitsFormat, VerticalTextPosition
DimRadial	AltRoundDistance, AltSuppressLeadingZeros, AltSuppressTrailingZeros, AltSuppressZeroFeet, AltSuppressZeroInches, AltTextPrefix, AltTextSuffix, AltTolerancePrecision, AltToleranceSuppressLeadingZeros, AltToleranceSuppressTrailingZeros, AltToleranceSuppressZeroFeet, AltToleranceSuppressZeroInches, AltUnits, AltUnitsFormat, AltUnitsPrecision, AltUnitsScale, ArrowheadBlock, ArrowheadSize, ArrowheadType, CenterMarkSize, CenterType, DecimalSeparator, Delete, DimensionLineColor, DimensionLineWeight, DimLineSuppress, Document, Fit, ForceLineInside, FractionFormat, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, LinearScaleFactor, Lineweight, Measurement, Modified, ObjectName, OwnerID, PlotStyleName, PrimaryUnitsPrecision, RoundDistance, ScaleFactor, SuppressLeadingZeros, SuppressTrailingZeros, SuppressZeroFeet, SuppressZeroInches, TextColor, TextGap, TextHeight, TextInside, TextInsideAlign,

## Phương thức, thuộc tính và sự kiện mới kể từ AutoCAD phiên bản 14.01

Đối tượng	Phương thức/ Thuộc tính / Sự kiện
	TextMovement, TextOutsideAlign, TextOverride, TextPrefix, TextStyle, TextSuffix, ToleranceDisplay, ToleranceHeightScale, ToleranceJustification, ToleranceLowerLimit, TolerancePrecision, ToleranceSuppressLeadingZeros, ToleranceSuppressTrailingZeros, ToleranceSuppressZeroFeet, ToleranceSuppressZeroInches, ToleranceUpperLimit, UnitsFormat, VerticalTextPosition
DimRotated	AltRoundDistance, AltSuppressLeadingZeros, AltSuppressTrailingZeros, AltSuppressZeroFeet, AltSuppressZeroInches, AltTextPrefix, AltTextSuffix, AltTolerancePrecision, AltToleranceSuppressLeadingZeros, AltToleranceSuppressTrailingZeros, AltToleranceSuppressZeroFeet, AltToleranceSuppressZeroInches, AltUnits, AltUnitsFormat, AltUnitsPrecision, AltUnitsScale, Arrowhead1Block, Arrowhead1Type, Arrowhead2Block, Arrowhead2Type, ArrowheadSize, DecimalSeparator, Delete, DimensionLineColor, DimensionLineExtend, DimensionLineWeight, DimLine1Suppress, DimLine2Suppress, DimLineInside, Document, ExtensionLineColor, ExtensionLineExtend, ExtensionLineOffset, ExtensionLineWeight, ExtLine1Suppress, ExtLine2Suppress, Fit, orceLineInside, FractionFormat, GetExtensionDictionary, HasExtensionDictionary, HorizontalTextPosition, Hyperlinks, LinearScaleFactor, Lineweight, Measurement, Modified, ObjectName, OwnerID, PlotStyleName, PrimaryUnitsPrecision, RoundDistance, ScaleFactor, SuppressLeadingZeros, SuppressTrailingZeros, SuppressZeroFeet, SuppressZeroInches, TextColor, TextGap, TextHeight, TextInside, TextInsideAlign, TextMovement, TextOutsideAlign, TextOverride, TextPrefix, TextStyle, TextSuffix, ToleranceDisplay, ToleranceHeightScale, ToleranceJustification, ToleranceLowerLimit, TolerancePrecision, ToleranceSuppressLeadingZeros, ToleranceSuppressTrailingZeros, ToleranceSuppressZeroFeet, ToleranceSuppressZeroInches, ToleranceUpperLimit, UnitsFormat, VerticalTextPosition
DimStyle	CopyFrom, Document, GetExtensionDictionary, HasExtensionDictionary, Modified, ObjectID, ObjectName, OwnerID
DimStyles	Delete, Document, GetExtensionDictionary, GetXData, Handle, HasExtensionDictionary, Modified, ObjectID, ObjectName, OwnerID, SetXData
Document	Activate, Active, ActiveLayout, BeginClose, BeginDoubleClick, BeginLisp, BeginPlot, BeginRightClick, BeginShortcutMenuCommand, BeginShortcutMenuDefault, BeginShortcutMenuEdit, BeginShortcutMenuGrip, BeginShortcutMenuOsnap, Close, CopyObjects, Database, Deactivate, EndLisp, EndPlot, EndShortcutMenu, EndUndoMark,

## Phương thức, thuộc tính và sự kiện mới kể từ AutoCAD phiên bản 14.01

Đối tượng	Phương thức/ Thuộc tính / Sự kiện
	Height, HWnd, Layouts, LayoutSwitched, LispCancelled, ObjectAdded, ObjectErased, ObjectModified, PickfirstSelectionSet, PlotConfigurations, Preferences, SelectionChanged, SendCommand, StartUndoMark, Width, WindowChanged, WindowMovedOrResized, WindowState, WindowTitle
Documents *	Add, Application, Close, Count, Item, Open
Ellipse	Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, MajorRadius, MinorRadius, Modified, ObjectName, OwnerID, PlotStyleName
Entity +	Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName
ExternalReference *	Application, ArrayPolar, ArrayRectangular, Color, Copy, Delete, Document, Explode, GetAttributes, GetBoundingBox, GetConstantAttributes, GetExtensionDictionary, GetXdata, Handle, HasAttributes, HasExtensionDictionary, Highlight, Hyperlinks, InsertionPoint, IntersectWith, Layer, Linetype, LinetypeScale, Lineweight, Mirror, Mirror3D, Modified, Move, Name, Normal, ObjectID, ObjectName, OwnerID, Path, PlotStyleName, Rotate, Rotate3D, Rotation, ScaleEntity, SetXdata, TransformBy, Update, Visible, XScaleFactor, YScaleFactor, ZScaleFactor
Group	Document, GetExtensionDictionary, HasExtensionDictionary, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName
Groups	Delete, Document, GetExtensionDictionary, GetXData, Handle, HasExtensionDictionary, Modified, ObjectID, ObjectName, OwnerID, SetXData
hatch	Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, ISOPenWidth, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName
Hyperlink *	Application, Delete, URL, URLDescription, URLNamedLocation
Hyperlinks *	Add, Application, Count, Item
IDPair *	Application, IsCloned, IsOwnerXlated, IsPrimary, Key, Value
Layer	Document, GetExtensionDictionary, HasExtensionDictionary, Lineweight, Modified, ObjectID, ObjectName, OwnerID, PlotStyleName, Plottable, ViewportDefault
Layers	Delete, Document, GetExtensionDictionary, GetXData, Handle,

## Phương thức, thuộc tính và sự kiện mới kể từ AutoCAD phiên bản 14.01

Đối tượng	Phương thức/ Thuộc tính / Sự kiện
	HasExtensionDictionary, Modified, ObjectID, ObjectName, OwnerID, SetXData
Layout *	Application, Block, CanonicalMediaName, CenterPlot, ConfigName, CopyFrom, Delete, Document, GetCanonicalMediaNames, GetCustomScale, GetExtensionDictionary, GetLocaleMediaName, GetPaperMargins, GetPaperSize, GetPlotDeviceNames, GetPlotStyleTableNames, GetWindowToPlot, GetXdata, Handle, HasExtensionDictionary, ModelType, Modified, Name, ObjectID, ObjectName, OwnerID, PaperUnits, PlotHidden, PlotOrigin, PlotRotation, PlotType, PlotViewportBorders, PlotViewportsFirst, PlotWithLineweights, PlotWithPlotStyles, RefreshPlotDeviceInfo, ScaleLineweights, SetCustomScale, SetWindowToPlot, SetXdata, ShowPlotStyles, StandardScale, StyleSheet, TabOrder, UseStandardScale, ViewToPlot
Layouts *	Add, Application, Count, Delete, Document, GetExtensionDictionary, GetXdata, Handle, HasExtensionDictionary, Item, Modified, ObjectID, ObjectName, OwnerID, SetXdata
Leader	Annotation, ArrowheadBlock, ArrowheadSize, ArrowheadType, Coordinate, Delete, DimensionLineColor, DimensionLineWeight, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName, ScaleFactor, TextGap, VerticalTextPosition
Line	Angle, Delete, Delta, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Length, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName
LineType	Document, GetExtensionDictionary, HasExtensionDictionary, Modified, ObjectID, ObjectName, OwnerID
LineTypes	Delete, Document, GetExtensionDictionary, GetXData, Handle, HasExtensionDictionary, Modified, ObjectID, ObjectName, OwnerID, SetXData
LWPolyline	ConstantWidth, Coordinate, Delete, Document, Elevation, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, LinetypeGeneration, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName
MenuBar *	Application, Count, Item, Parent
MenuGroup *	Application, MenuFileName, Menus, Name, Parent, Save, SaveAs, Toolbars, Type, Unload
MenuGroups	Application, Count, Item, Load, Parent

## Phương thức, thuộc tính và sự kiện mới kể từ AutoCAD phiên bản 14.01

Đối tượng	Phương thức/ Thuộc tính / Sự kiện
MinsertBlock *	Application, ArrayPolar, ArrayRectangular, Color, Columns, ColumnSpacing, Copy, Delete, Document, Explode, GetAttributes, GetBoundingBox, GetConstantAttributes, GetExtensionDictionary, GetXdata, Handle, HasAttributes, HasExtensionDictionary, Highlight, Hyperlinks, InsertionPoint, IntersectWith, Layer, Linetype, LinetypeScale, Lineweight, Mirror, Mirror3D, Modified, Move, Name, Normal, ObjectID, ObjectName, OwnerID, PlotStyleName, Rotate, Rotate3D, Rotation, Rows, RowSpacing, ScaleEntity, SetXdata, TransformBy, Update, Visible, XScaleFactor, YScaleFactor, ZScaleFactor
Mline *	Application, ArrayPolar, ArrayRectangular, Color, Coordinates, Copy, Delete, Document, GetBoundingBox, GetExtensionDictionary, GetXdata, Handle, HasExtensionDictionary, Highlight, Hyperlinks, IntersectWith, Layer, Linetype, LinetypeScale, Lineweight, Mirror, Mirror3D, Modified, Move, ObjectID, ObjectName, OwnerID, PlotStyleName, Rotate, Rotate3D, ScaleEntity, SetXdata, StyleName, TransformBy, Update, Visible
ModelSpace	AddDim3PointAngular, AddMInsertBlock, AddMLine, AddPolyfaceMesh, AttachExternalReference, Bind, Delete, Detach, Document, GetExtensionDictionary, GetXData, HasExtensionDictionary, IsLayout, IsXRef, Layout, Modified, ObjectID, ObjectName, Origin, OwnerID, Reload, SetXData, Unload, XrefDatabase
Mtext	Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, LineSpacingFactor, LineSpacingStyle, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName
Object +	Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Modified, ObjectName, OwnerID
PaperSpace	AddDim3PointAngular, AddMInsertBlock, AddMLine, AddPolyfaceMesh, AttachExternalReference, Bind, Delete, Detach, Document, GetExtensionDictionary, GetXData, HasExtensionDictionary, IsLayout, IsXRef, Layout, Modified, ObjectID, ObjectName, Origin, OwnerID, Reload, SetXData, Unload, XrefDatabase
Plot	BatchPlotProgress, DisplayPlotPreview, NumberOfCopies, QuietErrorMode, SetLayoutsToPlot, StartBatchMode
PlotConfiguration *	Application, CanonicalMediaName, CenterPlot, ConfigName, CopyFrom, Delete, Document, GetCanonicalMediaNames, GetCustomScale, GetExtensionDictionary, GetLocaleMediaName, GetPaperMargins, GetPaperSize, GetPlotDeviceNames,

## Phương thức, thuộc tính và sự kiện mới kể từ AutoCAD phiên bản 14.01

Đối tượng	Phương thức/ Thuộc tính / Sự kiện
	GetPlotStyleTableNames, GetWindowToPlot, GetXdata, Handle, HasExtensionDictionary, ModelType, Modified, Name, ObjectID, ObjectName, OwnerID, PaperUnits, PlotHidden, PlotOrigin, PlotRotation, PlotType, PlotViewportBorders, PlotViewportsFirst, PlotWithLineweights, PlotWithPlotStyles, RefreshPlotDeviceInfo, ScaleLineweights, SetCustomScale, SetWindowToPlot, SetXdata, ShowPlotStyles, StandardScale, StyleSheet, UseStandardScale, ViewToPlot
PlotConfigurations *	Add, Application, Count, Delete, Document, GetExtensionDictionary, GetXdata, Handle, HasExtensionDictionary, Item, Modified, ObjectID, ObjectName, OwnerID, SetXdata
Point	Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName
PolyfaceMesh *	Application, ArrayPolar, ArrayRectangular, Color, Coordinate, Coordinates, Copy, Delete, Document, GetBoundingBox, GetExtensionDictionary, GetXdata, Handle, HasExtensionDictionary, Highlight, Hyperlinks, IntersectWith, Layer, Linetype, LinetypeScale, Lineweight, Mirror, Mirror3D, Modified, Move, NumberOfFaces, NumberOfVertices, ObjectID, ObjectName, OwnerID, PlotStyleName, Rotate, Rotate3D, ScaleEntity, SetXdata, TransformBy, Update, Visible
PolygonMesh	Coordinate, Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName
Polyline	ConstantWidth, Coordinate, Delete, Document, Elevation, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, LinetypeGeneration, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName
PopupMenu *	AddMenuItem, AddSeparator, AddSubMenu, Application, Count, InsertInMenuBar, Item, Name, NameNoMnemonic, OnMenuBar, Parent, RemoveFromMenuBar, ShortcutMenu, TagString
PopupMenuItem *	Application, Caption, Check, Delete, Enable, HelpString, Index, Label, Macro, Parent, SubMenu, TagString, Type
PopupMenu *	Add, Application, Count, InsertMenuInMenuBar, Item, Parent, RemoveMenuFromMenuBar
Preferences	Display, Drafting, Files, OpenSave, Output, Profiles, Selection, System, User
PreferencesDisplay *	Application, AutoTrackingVecColor, CursorSize,



## Phương thức, thuộc tính và sự kiện mới kể từ AutoCAD phiên bản 14.01

Đối tượng	Phương thức/ Thuộc tính / Sự kiện
	DisplayLayoutTabs, DisplayScreenMenu, DisplayScrollBars, DockedVisibleLines, GraphicsWinLayoutBackgrndColor, GraphicsWinModelBackgrndColor, HistoryLines, ImageFrameHighlight, LayoutCreateViewport, LayoutCrosshairColor, LayoutDisplayMargins, LayoutDisplayPaper, LayoutDisplayPaperShadow, LayoutShowPlotSetup, MaxAutoCADWindow, ModelCrosshairColor, ShowRasterImage, TextFont, TextFontSize, TextFontStyle, TextWinBackgrndColor, TextWinTextColor, TrueColorImages, XrefFadeIntensity
PreferencesDrafting *	AlignmentPointAcquisition, Application, AutoSnapAperture, AutoSnapApertureSize, AutoSnapMagnet, AutoSnapMarker, AutoSnapMarkerColor, AutoSnapMarkerSize, AutoSnapTooltip, AutoTrackTooltip, FullScreenTrackingVector, PolarTrackingVector
PreferencesFile	AltFontFile, AltTabletMenuFile, Application, AutoSavePath, ConfigFile, CustomDictionary, DefaultInternetURL, DriversPath, FontFileMap, GetProjectFilePath, HelpFilePath, LicenseServer, LogFilePath, MainDictionary, MenuFile, ObjectARXPath, PostScriptPrologFile, PrinterConfigPath, PrinterDescPath, PrinterStyleSheetPath, PrintFile, PrintSpoolerPath, PrintSpoolExecutable, SetProjectFilePath, SupportPath, TempFilePath, TemplateDwgPath, TempXrefPath, TextEditor, TextureMapPath, WorkspacePath
PreferencesOpenSave *	Application, AutoAudit, AutoSaveInterval, CreateBackup, DemandLoadArxApp, FullCrcValidation, IncrementalSavePercent, LogFileOn, MRUNumber, ProxyImage, SaveAsType, SavePreviewThumbnail, ShowProxyDialogBox, TempFileExtension, XrefDemandLoad
PreferencesOutput *	Application, DefaultOutputDevice, DefaultPlotStyleForLayer, DefaultPlotStyleForObjects, DefaultPlotStyleTable, OLEQuality, PlotLegacy, PlotPolicy, PrinterPaperSizeAlert, PrinterSpoolAlert, UseLastPlotSettings
PreferencesProfiles *	ActiveProfile, Application, CopyProfile, DeleteProfile, ExportProfile, GetAllProfileNames, ImportProfile, RenameProfile, ResetProfile
PreferencesSelection *	Application, DisplayGrips, DisplayGripsWithinBlocks, GripColorSelected, GripColorUnselected, GripSize, PickAdd, PickAuto, PickBoxSize, PickDrag, PickFirst, PickGroup
PreferencesSystem *	Application, BeepOnError, DisplayOLEScale, EnableStartupDialog, LoadAcadLspInAllDocuments, ShowWarningMessages, SingleDocumentMode, StoreSQLIndex, TablesReadOnly
PreferencesUser *	ADCInsertUnitsDefaultSource, ADCInsertUnitsDefaultTarget, Application, HyperlinkDisplayCursor, HyperlinkDisplayTooltip,



## Phương thức, thuộc tính và sự kiện mới kể từ AutoCAD phiên bản 14.01

Đối tượng	Phương thức/ Thuộc tính / Sự kiện
	KeyboardAccelerator, KeyboardPriority, SCMCommandMode, SCMDDefaultMode, SCMEditMode, ShortCutMenuDisplay
Pviewport	ArcSmoothness, Clipped, CustomScale, Delete, DisplayLocked, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName, StandardScale, StyleSheet, UCSPerViewport, ViewportOn
RasterImage	Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, ImageHeight, ImageWidth, Lineweight, Modified, Name, ObjectName, OwnerID, PlotStyleName, Rotation, ScaleFactor, ShowRotation
Ray	Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName, SecondPoint
Region	Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName
RegisteredApplication	Document, GetExtensionDictionary, HasExtensionDictionary, Modified, ObjectID, ObjectName, OwnerID
RegisteredApplications	Delete, Document, GetExtensionDictionary, GetXData, Handle, HasExtensionDictionary, Modified, ObjectID, ObjectName, OwnerID, SetXData
Shape	Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName
Solid	Coordinate, Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName
Spline	ControlPoints, Delete, Document, FitPoints, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, IsPeriodic, IsPlanar, Knots, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName, Weights
State *	Application, IsQuiescent
Text	Alignment, Backward, Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName, UpsideDown
TextStyle	Document, GetExtensionDictionary, GetFont,

## Phương thức, thuộc tính và sự kiện mới kể từ AutoCAD phiên bản 14.01

<b>Đối tượng</b>	<b>Phương thức/ Thuộc tính / Sự kiện</b>
	HasExtensionDictionary, Modified, ObjectID, ObjectName, OwnerID, SetFont
TextStyles	Delete, Document, GetExtensionDictionary, GetXData, Handle, HasExtensionDictionary, Modified, ObjectID, ObjectName, OwnerID, SetXData
Tolerance	Delete, DimensionLineColor, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName, ScaleFactor, TextColor, TextHeight, TextStyle
Toolbar *	AddSeparator, AddToolbarButton, Application, Count, Delete, Dock, DockStatus, Float, FloatingRows, Height, HelpString, Item, LargeButtons, Left, Name, Parent, TagString, Top, Visible, Width
ToolbarItem *	Application, AttachToolbarToFlyout, Delete, Flyout, GetBitmaps, HelpString, Index, Macro, Name, Parent, SetBitmaps, TagString, Type
Toolbars *	Add, Application, Count, Item, LargeButtons, Parent
Trace	Coordinate, Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName
UCS	Document, GetExtensionDictionary, HasExtensionDictionary, Modified, ObjectID, ObjectName, OwnerID
UCSs	Delete, Document, GetExtensionDictionary, GetXData, Handle, HasExtensionDictionary, Modified, ObjectID, ObjectName, OwnerID, SetXData
Utility	GetRemoteFile, GetSubEntity, IsRemoteFile, IsURL, LaunchBrowserDialog, Prompt, PutRemoteFile
View	Document, GetExtensionDictionary, HasExtensionDictionary, Modified, ObjectID, ObjectName, OwnerID
ViewPort	ArcSmoothness, Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Modified, ObjectID, ObjectName, OwnerID
Viewports	DeleteConfiguration, Document, GetExtensionDictionary, GetXData, Handle, HasExtensionDictionary, Modified, ObjectID, ObjectName, OwnerID, SetXData
Views	Delete, Document, GetExtensionDictionary, GetXData, Handle, HasExtensionDictionary, Modified, ObjectID, ObjectName, OwnerID, SetXData

## Phương thức, thuộc tính và sự kiện mới kể từ AutoCAD phiên bản 14.01

Đối tượng	Phương thức/ Thuộc tính / Sự kiện
Xline	Delete, Document, GetExtensionDictionary, HasExtensionDictionary, Hyperlinks, Lineweight, Modified, ObjectName, OwnerID, PlotStyleName, SecondPoint
Xrecord *	Application, Delete, Document, GetExtensionDictionary, GetXdata, GetXRecordData, Handle, HasExtensionDictionary, Modified, Name, ObjectID, ObjectName, OwnerID, SetXdata, SetXRecordData, TranslateIDs

## 2. Mục đã thay đổi

Phần dưới sẽ liệt kê các mô tả vắn tắt các phương thức và thuộc tính thay đổi so với phiên bản AutoCAD Release 14.01

### Phương thức và thuộc tính đã thay đổi

Tên Phương thức/Thuộc tính ở phiên bản 14	Thể hiện trong AutoCAD 2000	Mô tả thay đổi
ArcSmoothness	object.ArcSmoothness	Chuyển một đối tượng Preferences sang đối tượng PViewport và Viewport
AuditInfo	object.AuditInfoFixError	Xoá tham số Filename
AutoSaveFile	object.AutoSavePath	Tên của thuộc tính này đổi thành và đồng thời cũng chuyển từ đối tượng Preferences sang đối tượng PreferencesFiles object.
EndUndoMark	object.EndUndoMark	Chuyển từ đối tượng Application sang đối tượng Document.
GetUCSMatrix	RetVal = object.GetUCSMatrix()	Thực hiện không chính xác phép chuyển vị. Phương thức này đã được hiệu chỉnh.
InsertBlock	RetVal = object.InsertBlock(Insertion Point, Name, Xscale, Yscale, ZScale, Rotation)	Thêm vào tham số Zscale.
LogFileName	object.LogFilePath	Tên của thuộc tính này đổi thành LogFilePath, đồng thời thuộc tính cũng được chuyển từ đối tượng Preferences sang đối tượng PreferencesFile.

## Phương thức và thuộc tính đã thay đổi

Tên Phương thức/Thuộc tính ở phiên bản 14	Thể hiện trong AutoCAD 2000	Mô tả thay đổi
StartUndoMark	object.StartUndoMark	Chuyển từ đối tượng Application sang đối tượng Document.
TextString (của đối tượng dimension)	object.TextOverride	Tên của thuộc tính này đổi thành TextOverride.
TransformBy	object.TransformBy TransformationMatrix	Thực hiện không chính xác phép chuyển vị. Phương thức này đã được hiệu chỉnh.
TranslateCoordinates	RetVal = TranslateCoordinates (OriginalPoint, From, To, Disp[,OCSNormal])	Thêm tham số OCSNormal cũng như tùy chọn acOCS cho tham số From và To để cho phép chuyển đổi tọa độ từ hoặc sang hệ trục OCS.
ZoomAll	object.ZoomAll ()	Phương thức này chuyển đổi đối tượng PViewport và đối tượng Viewport sang thành đối tượng Application.
ZoomCenter	object.ZoomCenter Center, Magnify	Phương thức này chuyển đổi đối tượng PViewport và đối tượng Viewport sang thành đối tượng Application.
ZoomExtents	object.ZoomExtents	Phương thức này chuyển đổi đối tượng PViewport và đối tượng Viewport sang thành đối tượng Application.
ZoomPickWindow	object.ZoomPickWindow	Phương thức này chuyển đổi đối tượng PViewport và đối tượng Viewport sang thành đối tượng Application.
ZoomScaled	object.ZoomScaled Scale, ScaleType	Phương thức này chuyển đổi đối tượng PViewport và đối tượng Viewport sang thành đối tượng Application.
ZoomWindow	object.ZoomWindow LowerLeft, UpperRight	Phương thức này chuyển đổi đối tượng PViewport và đối tượng Viewport sang thành đối tượng Application.

### 2.1. Đối tượng Preferences

Đối tượng Preferences có một số thay đổi chủ yếu sau:

- Đối tượng Preferences được chia nhỏ hơn. Các đối tượng mới gồm:
  - PreferencesDisplay
  - PreferencesDrafting

- PreferencesFiles
- PreferencesOpenSave
- PreferencesOutput
- PreferencesSelection
- PreferencesSystem
- PreferencesUser
- DatabasePreferences

Tất cả các thuộc tính và phương thức nêu ở phần trước có trong đối tượng Preferences đều được chuyển sang một trong các đối tượng được liệt kê trong danh sách trên. Để tìm một đối tượng mới cho một phương thức hay thuộc tính Preference bất kỳ nào đó, tìm trong “*AutoCAD ActiveX and VBA Reference*”.

- Tất cả các thuộc tính chỉ màu sắc đều được chuyển sang sử dụng hằng số OLE\_COLOR thay vì dùng hằng số màu của AutoCAD. (Chú ý rằng điều này chỉ áp dụng cho các thuộc tính của đối tượng Preferences trong danh sách nêu trên)

### 3. Mục đã loại bỏ

Dưới đây là danh sách mô tả ngắn gọn các phương thức và thuộc tính cũ đã bị loại bỏ kể từ phiên bản 14.01

#### Các phương thức và thuộc tính bị loại bỏ

Tên Phương thức/ Thuộc tính trong phiên bản 14	Mô tả
AdjustAreaFill	Gán tệp định dạng PC3 ( bằng Plotter Configuration Editor).
CrosshairColor	Sử dụng PreferencesDisplay.ModelSpaceCrosshairColor hoặc PreferencesDisplay.LayoutCrosshairColor.
EntityName	Sử dụng thuộc tính ObjectName của ActiveX AutoCAD hoặc trong VB và VBA sử dụng từ khoá TypeOf hoặc hàm TypeName .
EntityType	Xem phần EntityName ở trên.
Erase	Phương thức này được loại bỏ trong tất cả các đối tượng trừ đối tượng SelectionSet. Thay vào đó, ta sẽ sử dụng phương thức Delete.
GraphicsTextBackgrndColor	Chữ đồ họa sử dụng cùng màu nền với đồ họa. Dùng PreferencesDisplay.GraphicsWinLayoutBackgrndColor và PreferencesDisplay.GraphicsWinModelBackgrndColor.
GraphicsTextColor	Chữ đồ họa sử dụng cùng màu nền với con trỏ (crosshair). Dùng PreferencesDisplay.LayoutCrosshairColor Và

## Các phương thức và thuộc tính bị loại bỏ

Tên Phương thức/ Thuộc tính trong phiên bản 14	Mô tả
	PreferencesDisplay.ModelCrosshairColor.
HideLines	Sử dụng AcadLayout.Hidden.
ListADS	Chương trình dạng EXE dựa trên ADSRX không được hỗ trợ trong AutoCAD 2000. Phương thức để truy vấn ứng dụng ARX và DBX là ListARX.
LoadADS	Các ứng dụng ADSRX dựa trên EXE không được hỗ trợ trong AutoCAD 2000. Phương thức tải các ứng dụng ARX and DBX là LoadARX.
LoadPC2	Tệp PC2 không được hỗ trợ trong AutoCAD 2000. Cần chuyển đổi tệp PC2 thành tệp PC3 qua Add-a-Plotter Wizard.
MonochromeVectors	Chức năng không sử dụng nữa.
Origin	Sử dụng AcadLayout.PlotOffset.
PaperSize	Sử dụng AcadLayout.CanonicalMediaName.
PlotExtents	Sử dụng AcadLayout.AreaToPlot.
PlotLimits	Sử dụng AcadLayout.AreaToPlot.
PlotOrientation	Sử dụng AcadLayout.Rotation .
PlotScale	Sử dụng AcadLayout.Scale, StandardScale, CustomScale.
PlotUnits	Sử dụng AcadLayout.PaperUnits.
PlotView	Use AcadLayout.AreaToPlot and ViewToPlot.
PlotWindow	Sử dụng AcadLayout.AreaToPlot
PlotWithConfigTệp	Sử dụng PlotToDevice thay cho tệp PC3 và coi tệp PC3 như một đối số.
Rotation	Sử dụng AcadLayout.Rotation.
SavePC2	Tệp PC2 không được hỗ trợ trong AutoCAD 2000. Cần chuyển đổi tệp PC2 thành tệp PC3 qua Add-a-Plotter Wizard.
UnloadADS	Các ứng dụng ADSRX dựa trên EXE không được hỗ trợ trong AutoCAD 2000. Phương thức dỡ các ứng dụng ARX and DBX là UnLoadARX.