# MỤC LỤC[1]



[1] Daipv78@gmail.com

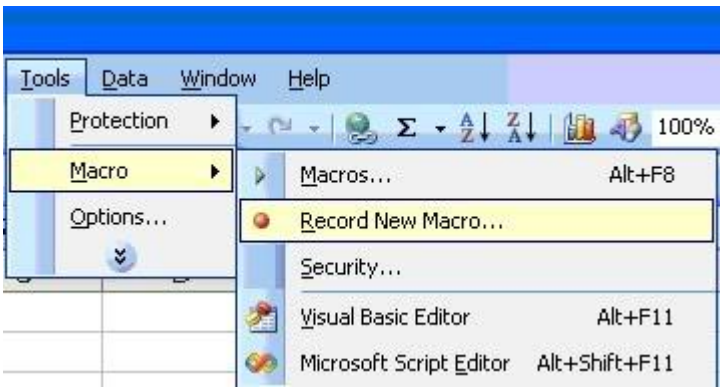
## 1. Ghi và thực hiện macro

Macro là gì?

Macro là tập hợp một số các dòng lệnh.

Bạn sử dụng chức năng Macro Recorder là một ý tưởng hay để từng bước thực hiện các công việc, nhất là lúc đầu tìm hiểu về macro. Excel đã hỗ trợ ghi lại (recorder) các công việc bạn đã thực hiện và chỉ không ghi lại khi bạn dừng ghi.

Ví dụ, một ô (cell) được chọn (selected) ở hiện tại sẽ không được ghi cho đến khi bạn thực hiện công việc trong ô đó. Ngoài ra, Excel cũng không ghi lại các công việc khi đang sử dụng bảng điều khiển (dialog box) cho đến khi bạn ấn nút OK trên bảng điều khiển đó.




*Hình 1: Thực hiện ghi macro*


*Hình 2: Cửa sổ Record Macro*

Trong suốt thời gian ghi, macro đã được lưu lại với tên xác định trong module, module được tạo ra trong quá trình ghi và là một phần của Workbook. Marco được ghi lại có thể được lưu trong This Workbook (Workbook hiện hành), New Workbook (Workbook mới) hoặc trong Personal Macro Workbook (những macro sở hữu riêng). Những lệnh (code) được lưu trong Personal.xls, những macro sở hữu riêng đều sử dụng được khi bạn mở Excel ra. Các macro trong các Workbook khác nhau có thể sử dụng bất cứ lúc nào khi các Workbook đang mở (kể cả sử dụng chúng từ Workbook khác).

Điều kiện để có thể tiến hành ghi macro:

1. Bảng tính Excel hiện hành (Activate Excel).

2. Sử dụng Workbook mới.

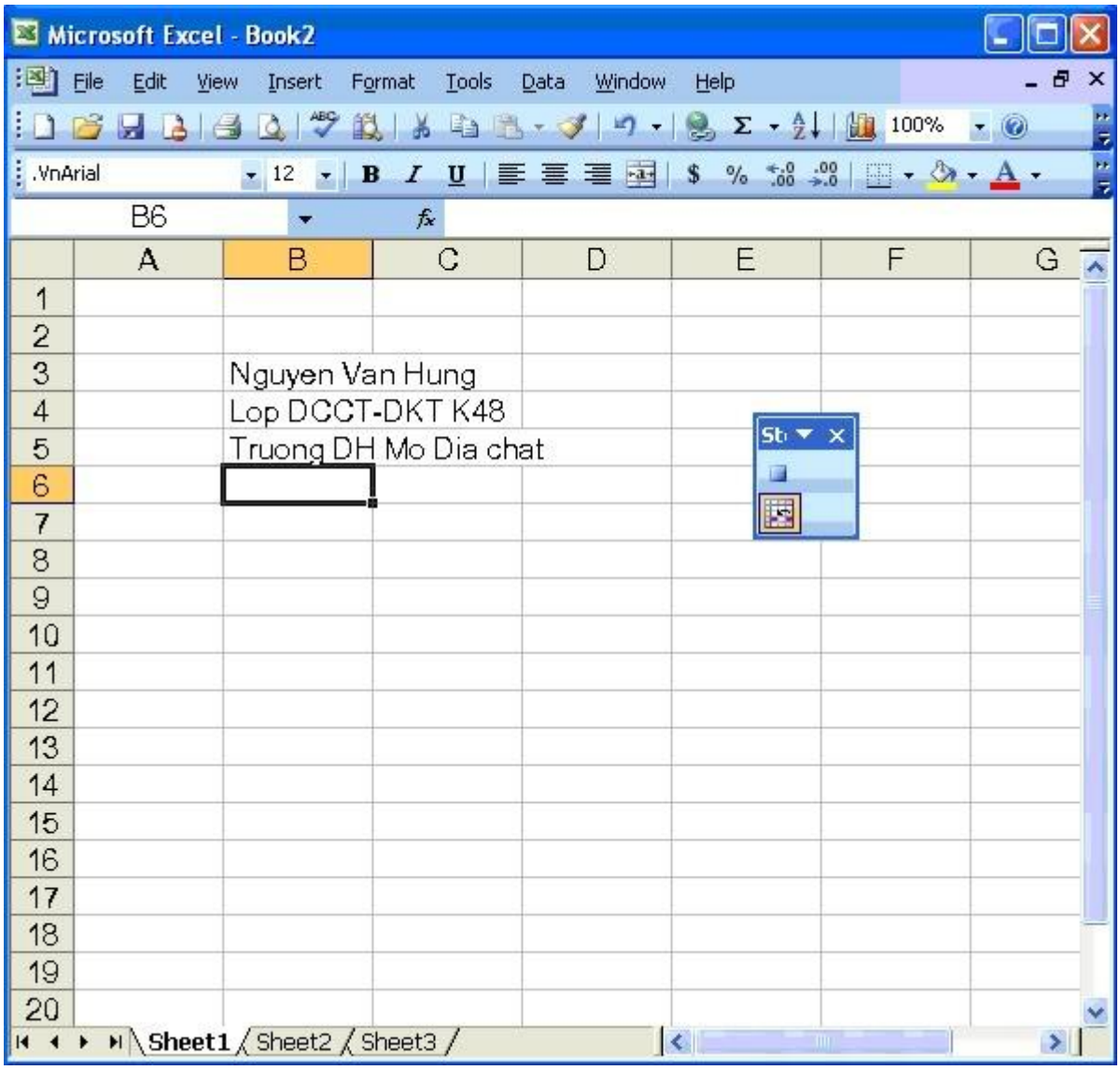
### *1.1. Ghi macro trong trường hợp sử dụng tham chiếu địa chỉ ô tuyệt đối*

Bạn hãy ghi lại macro trình bày tên bạn và địa chỉ như sau:

*1. Trong Tools/Macro, chọn Record New Macro (hình 1).*

*2.Trong Macro name: gõ Address_abs để đặt tên macro đó (hình 2).*

Đặc điểm là ký tự đầu tiên là của tên macro phải là chữ. Còn các ký tự khác có thể là chữ, số hoặc ký tự gạch dưới (ký tự _). Các ký tự đặc biệt như khoảng trống (Space), @, %, $, #, &, ... không được chấp nhận, bạn có thể dùng ký tự _ để tách tên trong macro.




*Hình 3: Quá trình ghi*

*3. Chuyển sang Shortcut key: để trống (sẽ thực hiện sau).*

*4. Trong Store macro in: để mặc định là This Workbook.*

*5. Trong Description: bạn gõ nội dung sau*

*Enter address starting in cell B3*

*6. Bấm OK.*

*7. Thanh Stop Recording sẽ xuất hiện. Bạn có thể di chuyển nó đến vị trí khác nếu thấy cần thiết.*

*8. Trong thanh Stop Recording, ấn vào nút Relative Reference cho mờ đi (không tác dụng- hình 3).*

*9. Trong Sheet1, bấm vào B3 và gõ tên bạn. Ô ở dưới gõ tên lớp, tiếp theo là tên trường.*

*10. Cho toàn bộ các chữ đậm và nghiêng.*

*11. Bấm vào ô B6.*

*12. Trong Stop Recording, bấm vào nút Stop Recording.*

Như vậy, macro có tên Address_abs đã được ghi lại. Những ô mà bạn đã sử dụng trong quá trình ghi được thể hiện dưới dạng địa chỉ tuyệt đối. Vì vậy, những ô trong Worksheet đó sẽ thực hiện khi bạn cho chạy macro, tên, lớp và tên trường sẽ được tạo ra đúng vị trí trong Worksheet.

**Ghi chú:** Bạn có thể lựa chọn tham chiếu tương đối trong suốt quá trình ghi macro. Vấn đề này sẽ được đề cập ở mục 1.3.

### *1.2. Chạy macro khi sử dụng bảng điều khiển macro (Macro dialog box)*

Bạn cho chạy macro trên từ Sheet2 như sau:

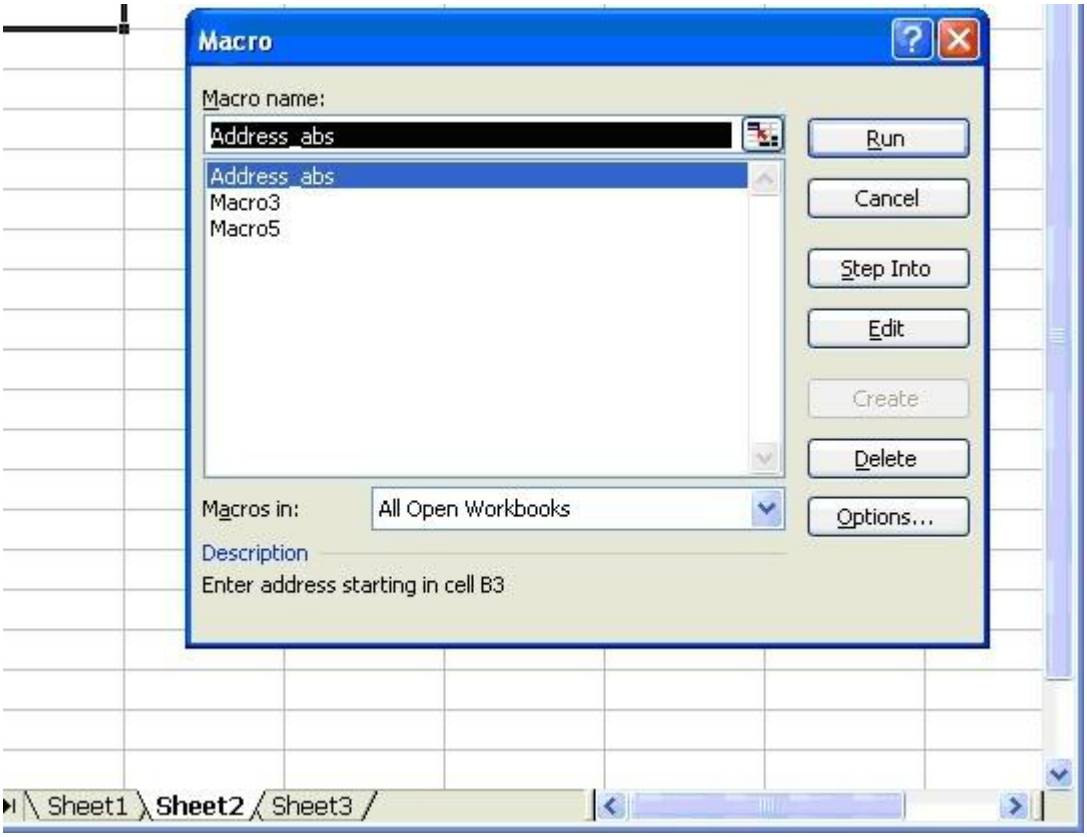
*1. Chọn sang Sheet2 và bấm vào ô nào đó ngoài ô B3.*

*2. Trong menu Tools/Macro, chọn Macros (hình 1).*

*3. Bấm vào macro có tên Address_abs trong danh sách macro (hình 4).*

*4. Bấm vào nút Run.*

Sau đó bạn sẽ thấy nội dung ở Sheet2 giống như ở Sheet1.




*Hình 4: Chạy macro ở Sheet2*

**Ghi chú:** Nếu bạn muốn huỷ quá trình chạy macro trước khi kết thúc, ấn vào nút Esc.

### *1.3. Ghi macro trong trường hợp sử dụng tham chiếu địa chỉ ô tương đối*

Macro Address_abs sử dụng địa chỉ ô tuyệt đối. Tiếp theo bạn sẽ tạo một macro cũng giống như trên. Macro trước đã chọn các ô (select cells) có quan hệ với vị trí của ô hoạt động (active) trong quá trình chạy, macro sẽ ghi lại quan hệ tham chiếu ô tương đối.

*1. Chọn Sheet1.*

*2. Bấm vào ô B11.*

*3. Trong menu Tools/Macro, chọn Record New Macros (hình 1).*

*4. Trong Macro name: gõ Address_Ref để đặt tên macro đó (hình 2).*

*5. Trong Shortcut key: Gõ chữ A, như vậy phím tắt sẽ là Ctrl+Shift+A (Nếu phím tắt bị trùng với phím có sẵn thì Excel tự động bổ sung thêm phím Shift như trường hợp này).*

*6. Trong Store macro in: để mặc định là This Workbook.*

*7. Trong Description: bạn gõ nội dung sau*

Enter address starting in activate cell position

*8. Bấm OK.*

*9. Thanh Stop Recording sẽ xuất hiện. Bạn có thể di chuyển nó đến vị trí khác nếu thấy cần thiết.*

*10. Trong thanh Stop Recording, ấn vào nút Relative Reference cho mờ đi (không tác dụng).*

Microsoft Excel sẽ tiếp tục ghi macro với quan hệ tương đối cho đến khi nào thoát khỏi Microsoft Excel hoặc bạn ấn lại vào nút Relative Reference.

*11. Gõ tên bạn, lớp, tên trường và địa chỉ trong các ô B11, B12, B13 và B14. Nội dung thể hiện như sau:*

Nguyen Van Hung

Lop DCCT-DKT K48

Truong DH Mo Dia chat

xa Dong Ngac, Tu Liem, Ha Noi

*12. Cho toàn bộ các chữ đậm.*

*13. Bấm vào ô B15.*

*14. Trong Stop Recording, bấm vào nút Stop Recording.*

***Ghi chú:*** Nếu bạn muốn macro chọn ô đặc biệt, đầu tiên chọn ô đầu (active cell), sau đó chọn ô có quan hệ với ô đầu, bạn có thể lựa chọn hỗn hợp địa chỉ tuyệt đối và tương đối trong quá trình ghi macro.

Để sử dụng tham chiếu tương đối trong suốt quá trình ghi macro, nút Relative Reference luôn sáng (có tác dụng).

Để sử dụng tham chiếu tuyệt đối trong suốt quá trình ghi macro, nút Relative Reference luôn tối (không tác dụng).

### *1.4. Dùng phím tắt để thực hiện một macro (shortcut key)*

Macro **Address_Ref** có thể thực hiện như mô tả trong mục 2.2. Lúc trước phím tắt đã được ấn định để thực hiện công việc đó, hãy sử dụng phương pháp thay thế này:

*1. Tại Sheet2 bạn chọn vào 1 ô (ví dụ ô H14).*

*2. Ấn tổ hợp phím Ctrl+Shift+A. Khi đó tên và địa chỉ sẽ xuất hiện dưới ô đó.*

*3. Bạn hãy thử thực hiện lại macro đó tại các vị trí khác trong Sheet2.*

## 2. Cách thực hiện một macro đơn giản

Dưới đây là các phương thức để thực hiện macro, bạn có thể cho thực hiện macro từ các đối tượng sau:

- *Đối tượng đồ hoạ trong worksheet hoặc biểu đồ*
- *Nút (button) trong thanh công cụ (Toolbar)*
- *Dòng lệnh (command) trong menu của Excel*

## *2.1. Thực hiện macro từ một đối tượng đồ hoạ trong worksheet*

Bạn có thể dùng đối tượng đồ hoạ trong worksheet để thực hiện một macro.

*1. Chọn Sheet3, nơi mà còn trống.*

*2. Vào menu View/Toolbars và bạn chọn Drawing (trừ trường hợp thanh công cụ Toolbar đã có trên màn hình).*

*3. Chọn đối tượng đồ hoạ như hình Oval và vẽ hình oval đó.*

*4. Gõ chữ vào hình oval đó bằng cách ấn phải chuột vào rồi chọn Add Text từ thực đơn tắt (hình 5).*

*5. Gõ nội dung Address rồi bấm ra ngoài để thoát.*

*6. Bạn có thể thay đổi kích thước hình oval cho phù hợp để thể hiện đủ nội dung chữ ở trong và tính mỹ thuật.*

*7. Ấn phải chuột vào hình oval đó, chọn Assign Macro.*

*8. Trong bảng Assign Macro, chọn macro có tên Address_Ref.*

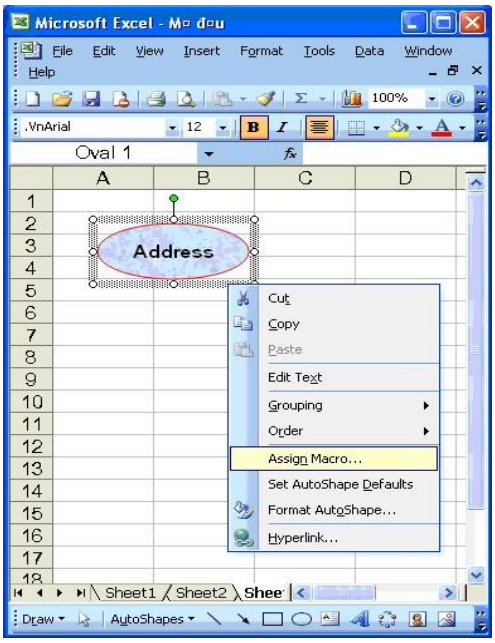
*9. Sau đó ấn OK.*

Sau đó, bạn cho thực hiện thử macro:

*1. Chọn 1 ô nào đó (ví dụ như ô J13).*

*2. Bấm vào hình oval trên, macro sẽ thực hiện.*

***Ghi chú:*** Nếu bạn muốn di chuyển đối tượng đồ hoạ (có macro) ra khỏi chỗ khác trong worksheet, sử dụng phải chuột để di chuyển (vì bấm trái chuột thì macro sẽ chạy). Còn nếu bạn muốn thay đổi macro khác thì bạn bấm phải chuột trên đối tượng, chọn Assign Macro và lựa chọn macro nào bạn muốn.




*Hình 5: Gán macro vào hình oval*

### *2.2. Chạy macro từ nút lệnh trên thanh công cụ*

Ngoài ra, có thể chạy macro từ nút lệnh (button) trong các các thanh công cụ tự tạo (custom toolbar). Ví dụ như có thể ấn định macro Address_abs trong nút hình mặt cười (Smiley Face) như sau:

*1. Di chuyển chuột đến một điểm nào đó trong các thanh toolbar.*

*2. Ấn phải chuột, trong thực đơn tắt chọn Customize.*

*3. Trong bảng Customize, chọn tab Toolbars (hình 6).*

*4. Chọn nút New.*

*5. Sau đó bảng New Toolbar xuất hiện và bạn gõ tên vào (Macro1) rồi OK.*

*6. Trong bảng Customize, chọn tab Commands (hình 7).*

*7. Trong hộp Categories, chọn AutoShapes.*

*8. Trong Commands, cuộn xuống cho đến khi bạn chọn được hình ưng ý (Smiley Face).*

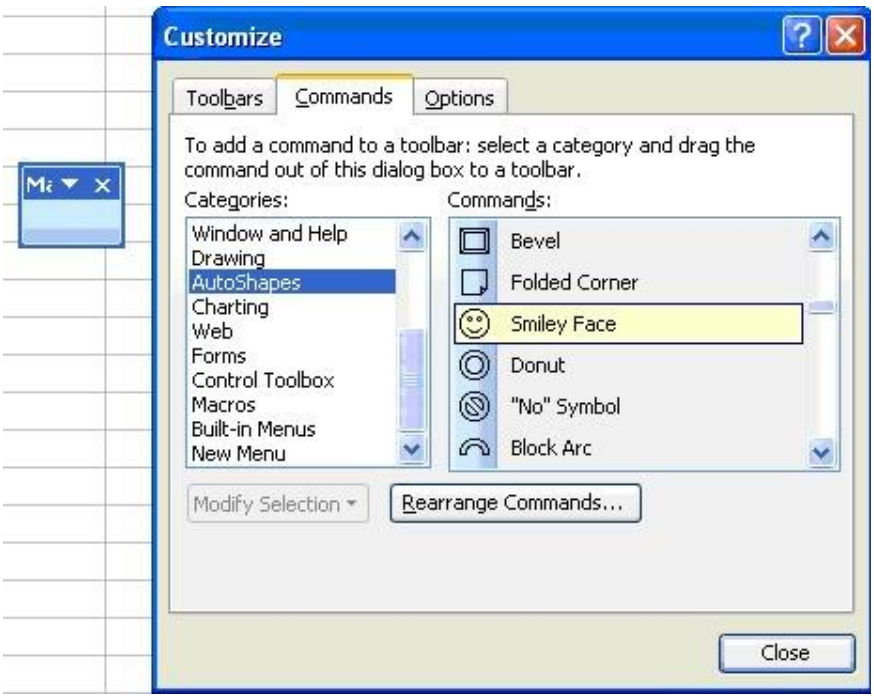
*9. Tại hình Smiley Face, giữ trái và kéo chuột vào trong thanh công cụ Macro1 (hình 8). Bạn có thể chọn thêm các biểu tượng khác nếu cần.*

*10. Bấm phải chuột vào nút Smiley Face, thực đơn tắt sẽ hiện ra. Bạn có thể sửa hay xoá hình đó và thay bằng các hình khác.*

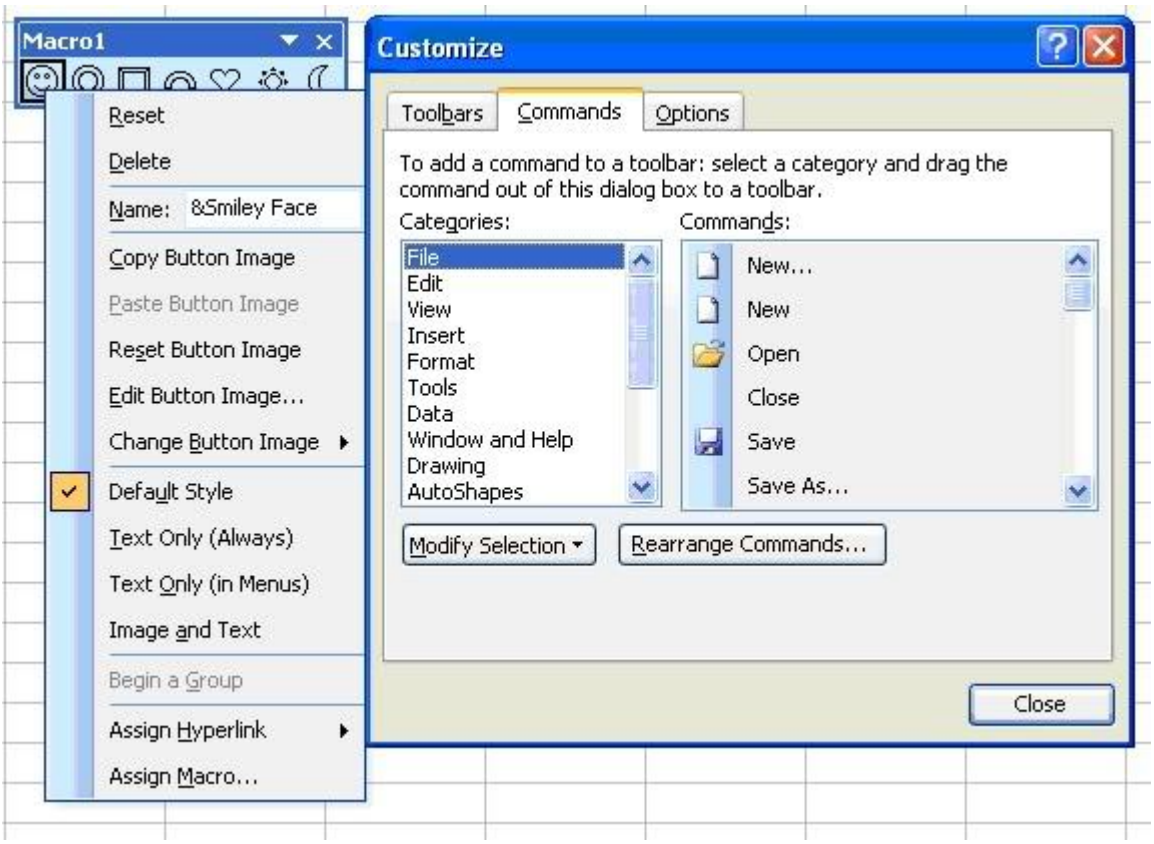
*11. Chọn Assign Macro trong thực đơn tắt, chọn macro Address_abs và ấn OK.*

*12. Đóng bảng Customize vào.*


*Hình 6: Tạo thanh công cụ mới*




*Hình 7: Gán hình vào nút lệnh mới*




*Hình 8: Tạo các nút lệnh trong thanh Macro1 và gán Assign Macro vào.*

Ghi chú: Thanh công cụ tự tạo thuộc sở hữu của workbook mà nó được tạo ra.

Bạn hãy thử sử dụng nút lệnh vừa tạo ra để thực hiện công việc như sau:

Code:

*1. Xoá sạch nội dung của Sheet2.*

*2. Bấm chuột vào nút Smiley Face trong thanh công cụ Macro1.*

Kết quả sẽ thể hiện trên Sheet2.

### *2.3. Chạy macro từ lệnh trong menu của Excel*

Từ menu của Excel bạn có thể thêm các menu mới mà khi lựa chọn chúng thì macro sẽ chạy. Ví dụ: Tạo menu mới có tên là Work Address có thể chứa menu con Macro2 trên thanh tiêu chuẩn như sau:

*1. Phải đảm bảo rằng workbook đang chứa macro của bạn đang hoạt động.*

*2. Thêm một worksheet mới bằng cách vào menu Insert/Worksheet (đặt là Sheet4)*

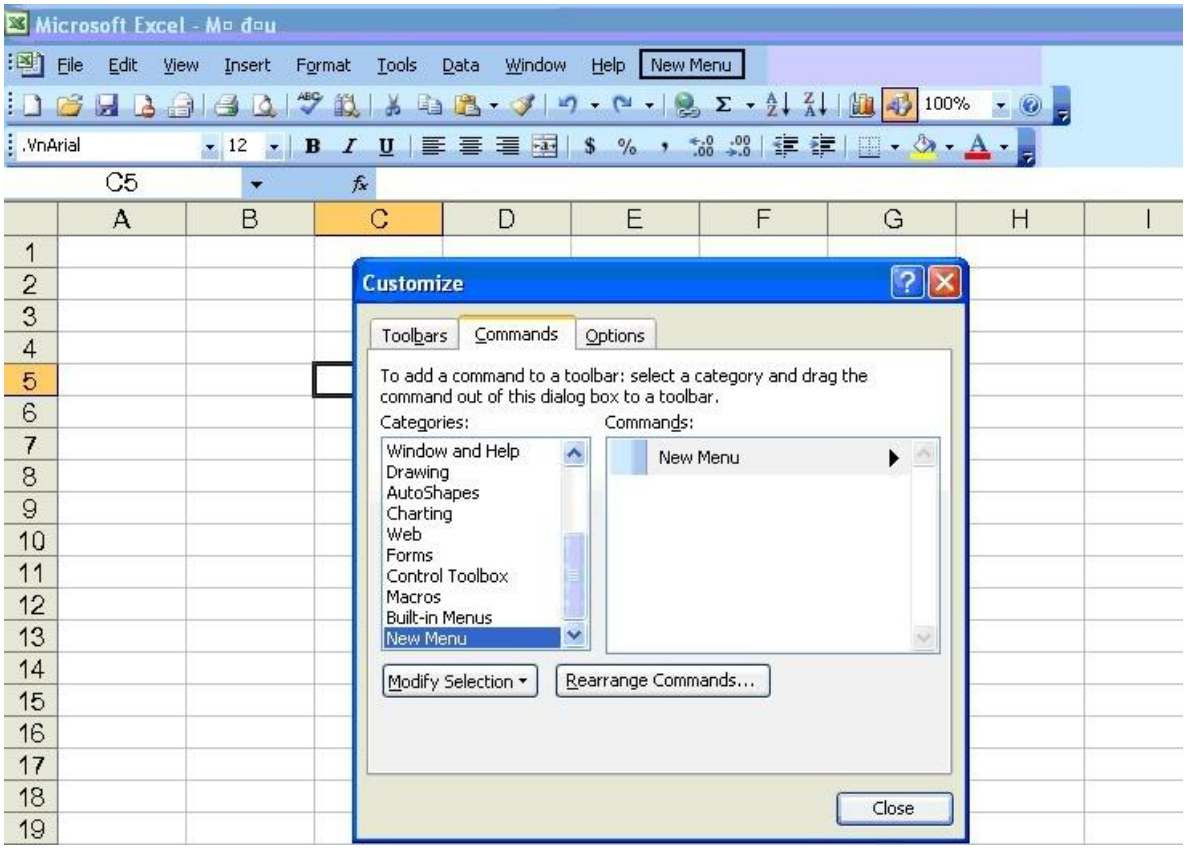
*3. Trong menu Tools/Customize, chọn tab Commands trong bảng Customize.*

*4. Cuộn xuống dòng cuối cùng và chọn New Menu trong Categories (hình 9).*

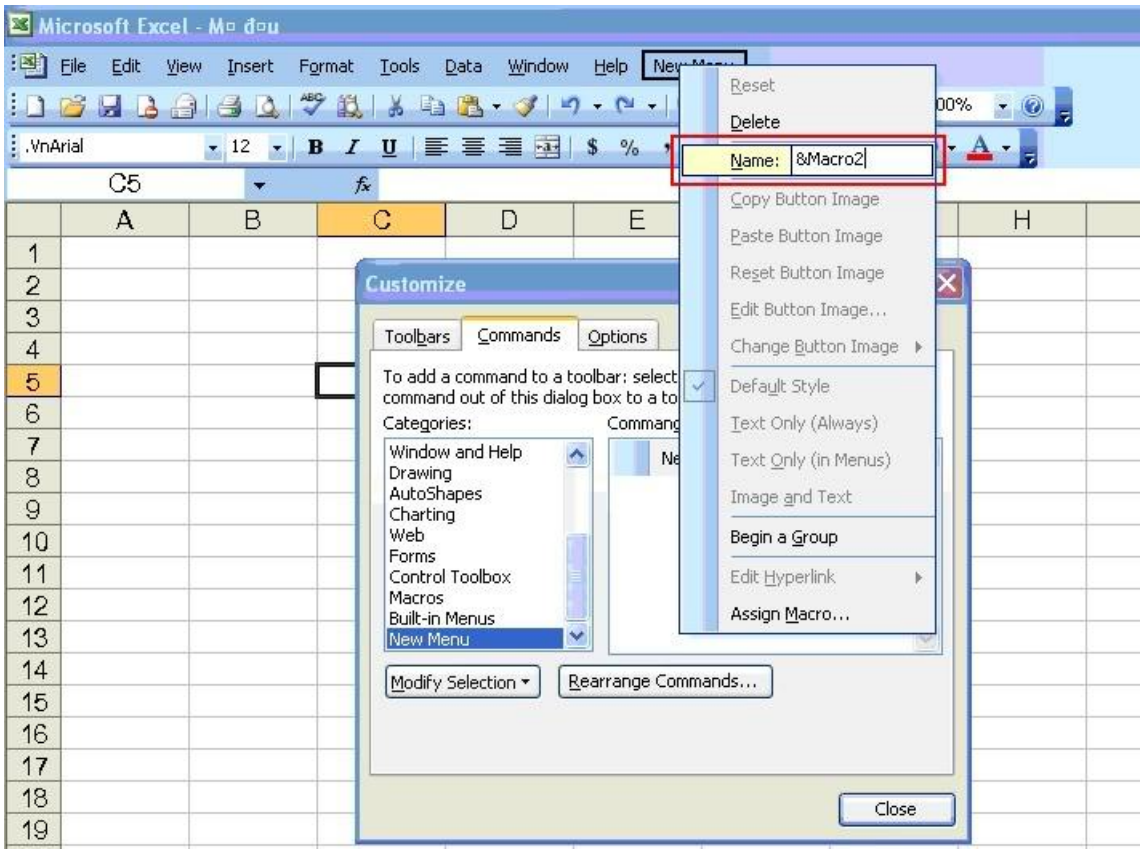
*5. Giữ trái chuột ở New Menu trong Commands và kéo vào dòng menu cạnh Help.*

*6. Ấn phải chuột vào New Menu trong menu của Excel, thực đơn tắt hiện ra.*

*7. Thay tên mới trong Name là &Macro2. Nếu chỉ cần 1 menu này thì bấm vào Assign Macro để chọn (hình 10). Còn nếu cần thêm các menu con (menu item) thì không cần.*




*Hình 9: Tạo New Menu trong menu của Excel*



*Hình 10: Tạo menu Maro2 trong menu của Excel*

Ký tự và (&) trước M sẽ gạch chân chữ M trong menu Macro2 (trở thành Macro2), đó chính là phím tắt để chạy macro Macro2 (chỉ cần ấn Alt+M).

Tiếp theo ta tiến hành tạo menu con trong Macro2:

*1. Trong Categories (trong trường hợp bảng Customize vẫn đang mở), chọn Macros.*

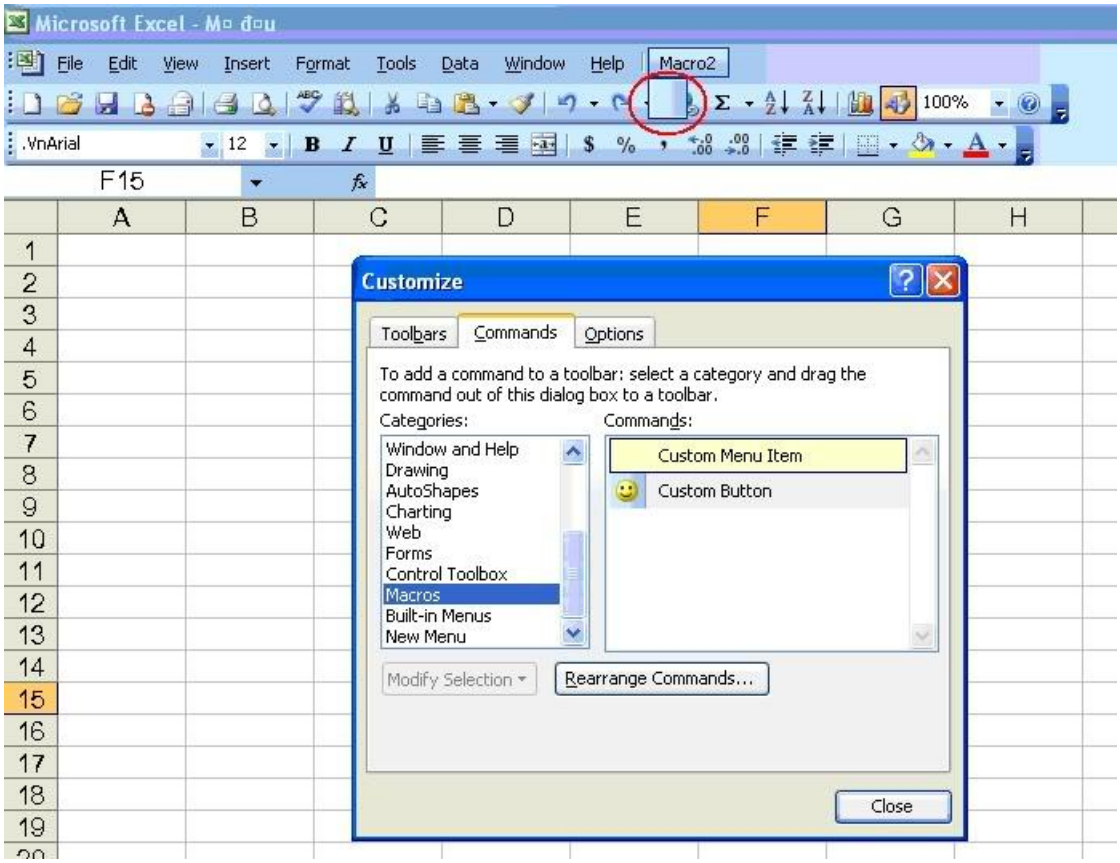
*2. Tại Commands, chọn Custom Menu Item (hình 11), giữ trái và kéo chuột đến phần trống ở dưới Macro2 (vùng được khoanh đỏ).*

*3. Bấm phải chuột vào Custom Menu Item trong menu mẹ Macro2.*

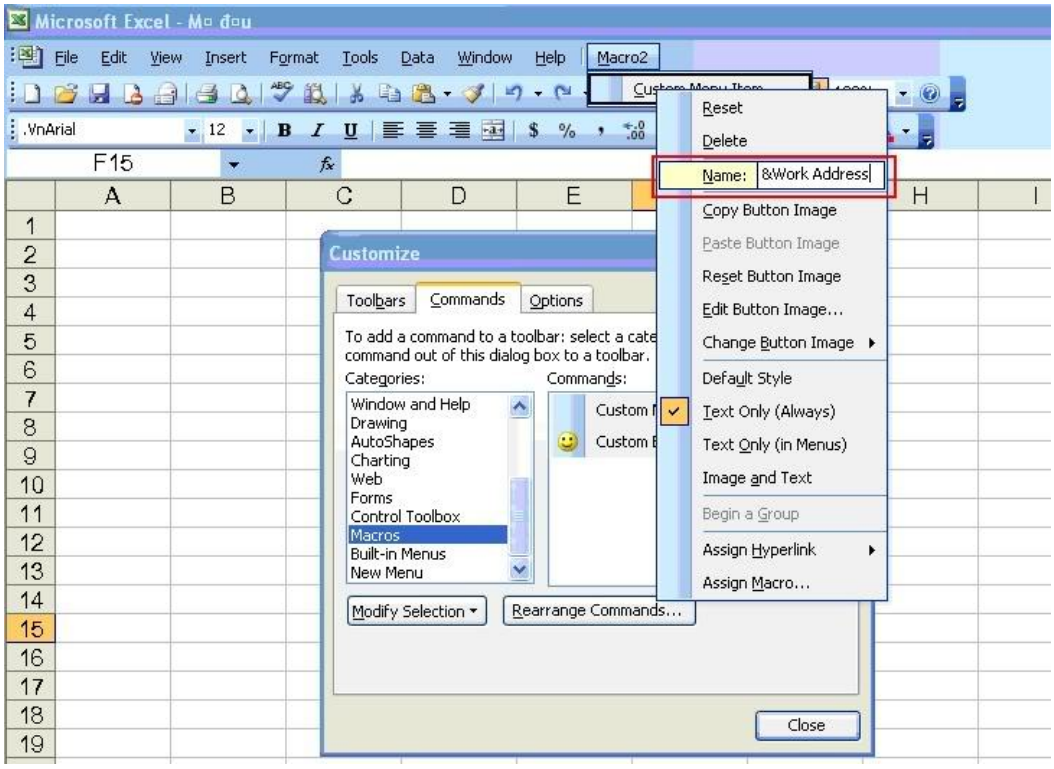
*4. Tại thực đơn tắt, đổi tên trong Name thành &Work Address (hình 12).*

*5. Sau đó vào Assign Macro để chọn macro chạy.*

*6. Cuối cùng là đóng bảng Customize.*




*Hình 11: Tạo các menu con*




*Hình 12: Đổi tên menu con và gán Assign Macro cho nó.*

Menu mới tạo được lưu giữ trong workbook đó. Kể cả bạn đã đóng workbook nhưng khi bạn bấm vào menu thì workbook chứa menu đó tự động mở ra và thực hiện lệnh luôn.

### *2.4. Thay đổi lựa chọn trong macro*

Nếu bạn muốn thay đổi các lực chọn chi tiết trong macro, bước đầu tiên bạn vào menu Tools/Macro và chọn Macros. Sau đó chọn tên macro mà bạn muốn thay đổi và bấm vào nút Option. Bạn có thể thay đổi phím tắt và mô tả lại công việc macro trong Description.

## 3. Sửa macro

Khi bạn ghi macro đầu tiên, Excel tạo ra molule trong workbook đó. Module đó chứa các lệnh (code) được viết trong VBA. Các bước thực hiện để nhìn thấy module:

*[code/1. Từ menu Tools/Macro chọn Macros.*

*2. Chọn macro Address_abs và bấm vào nút Edit.[/code]*

Cửa sổ Microsoft Visual Basic hiện ra như hình 13. Bạn có thể thấy rõ được các dòng code từng macro khi cuộn xuống.

### *3.1. Dạng form chung (General form)*

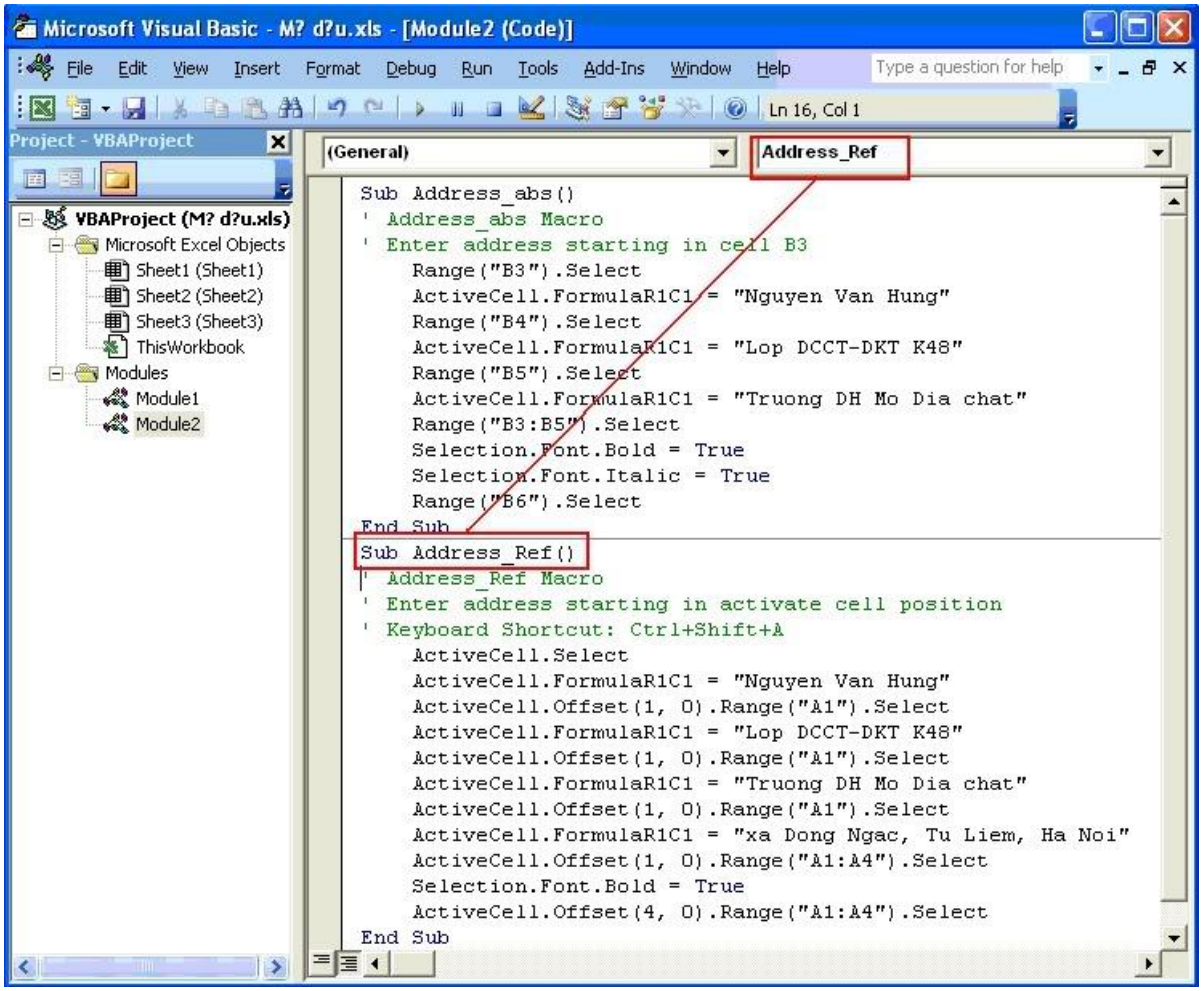
Từ khoá (keywords) là số hạng đặc biệt trong VB, được thể hiện bằng màu xanh lá cây. Tất cả các macro đều bắt đầu với Sub và kết thúc bởi End Sub (còn gọi là thủ tục).

Dòng màu xanh đó với dấu ' ở đầu dòng được gọi là chú thích (comments). Lời chú thích không anh hưởng đến macro và bạn có thể thay đổi nội dung của nó. Tên của macro và lời mô tả sử dụng (description) trong quá trình ghi macro xuất hiện dưới dạng chú thích. Bạn có thể dùng comments để chú thích trong quá trình xây dựng macro. Khi đó bạn sẽ dễ dàng hiểu được các bước cũng như nội dung thực hiện macro.

Đường đen liền có ý nghĩa phân chia các macro, function (hàm) trong module.

Đường gạch dưới (_) thỉnh thoảng gặp ở cuối dòng code. Khi code quá dài thì dùng (_) để xuống dòng, nhưng được hiểu là code vẫn liên tục.

Khi bạn ghi macro phức tạp hơn, bạn có thể gặp một số code không phải là bản chất của nó (essential). Excel ghi lại tất cả những gì bạn thực hiện một cách cụ thể nhất, kể cả những đối số (arguments) cài đặt mặc định trong Excel đã sử dụng. Khi bạn di chuyển chuột đến macro nào thì tên của macro đó hiện ở phần khoanh đỏ như hình 13 (Addres_Ref).




*Hình 13: Cửa sổ Microsoft Visual Basic.*

### *3.2. Tạo ra những thay đổi*

Trong cửa sổ Visual Basic Editor(VBE) (hình 13) có các module. Có thể coi module là nơi lưu trữ các thủ tục (sub) và hàm (function). Đây cũng là nơi khai báo các hằng số, biến số, kiểu dữ liệu người dùng. Mỗi module có thể chứa một hay nhiều Sub hoặc Function. Phần cửa sổ chính hiện nội dung code trông gần giống như Word, bạn có thể dễ dàng tạo những thay đổi trong đó, như bổ sung hay bớt đi nội dung nếu thấy cần thiết.

Ví dụ, bạn có thể thay đổi tên macro Address_abs thành Dia_chi chẳng hạn, chỉ cần gõ nội dung Dia_chi thay thế Address_abs trong Sub Address_abs(). Khi con chuột nằm trong macro Dia_chi, bạn vào Run và chọn Run Sub/UserForm (phím tắt F5). Để xem kết quả như thế nào thì bạn vào View/Microsoft Excel (phím tắt Alt+F11). Lúc đó của sổ VBE vẫn hiện hữu trong Task bar.

Để đóng cửa sổ VBE và trở về Excel, bạn vào menu File, sau đó chọn Close and Return to Microsoft Excel (phím tắt Alt + Q).

Ghi macro và xem lại những gì nó thực hiện là cách học rất hay, giúp các bạn có thể học hỏi thêm nhiều lệnh, nhiều đối tượng và các thuộc tính của nó, hiểu rõ trình tự các bước thực hiện. Nhưng đến một lúc nào đó, bạn muốn viết một macro cho riêng mình hoặc bổ sung thêm một vài code trong macro hiện tại để thực hiện các bài toán phức tạp hơn. Khi đó việc sử dụng ghi macro trở nên không hữu dụng nữa.

Macro không thể thực hiện được các tác vụ sau:

_Các kiểu vòng lặp.

_Các kiểu hành động theo điều kiện (sử dụng If-Then)

_Gán giá trị cho biến.

_Các kiểu dữ liệu đặc biệt.

_Hiện các thông báo (pop-up messages)

_Hiện các hộp thoại (dialog boxes)

Trong chương dưới đây, bạn có thể tìm được nhiều thông tin về VBA.

thay đổi nội dung bởi: **PhanTuHuong**, 14-04-07 lúc 10:12 PM

## 4. Ngữ pháp VB (Visual Basic Grammar)

### *4.1. Các đối tượng (Objects)*

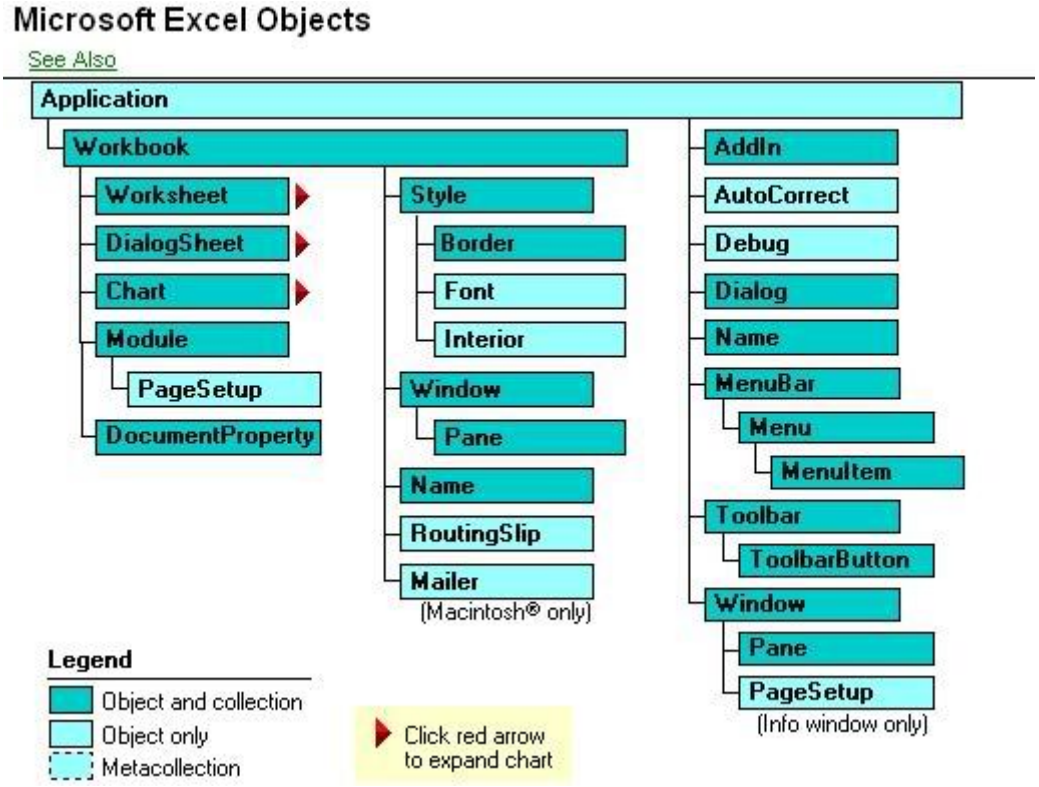
Visual Basic là ngôn ngữ lập trình hướng đối tượng (object-oriented). Điều đó có nghĩa là các thành phần trong Excel có thể coi là các đối tượng. Excel có hơn 100 đối tượng. Để cho các bạn dễ hình dung chúng ta có thể lấy một ví dụ như sau: Ta có một chiếc xe máy của Honda, đó có thể xem là một đối tượng. Honda có nhiều chủng loại xe máy như Future, Future II, Future neo;

Super Dream; Wave anh-pha... Vậy ta có thể xem Xe máy của hảng Honda là một tập hợp, trong tập hợp này có các đối tượng cùng nằm trong một nhóm như Future, Future II, Future neo.

Ví dụ dưới đây là những đối tượng trong Excel:

Code:

- the Excel application (là ứng dụng trong Excel- đối tượng lớn nhất- hình 14)
- a **workbook** (chính là file excel)
- a **worksheet** (là các sheet trong workbook)
- a **range** (là vùng)
- a **chart** (là biểu đồ)




*Hình 14: Các đối tượng trong Excel*

Bạn có thể coi những đối tượng trên như là danh từ (ví dụ: cái bánh là danh từ). Trong macro bạn lập, mà Range("B3") chính là đối tượng.

Đối tượng này có thể chứa các đối tượng khác ở trong nó. Đối tượng Application ở bậc cao nhất (đối tượng mẹ), bao gồm toàn bộ đối tượng trong Excel. Những thay đổi xảy ra trong đối tượng Application ảnh hưởng đến toàn bộ nội dung trong nó. Đối tượng Application có chứa đối tượng cũng lớn như Workbooks. Ví dụ như sau:

Application.Workbooks đề cập (refer) đến tất cả workbook đang mở trong Excel.

Workbooks.Item(1) đề cập đến workbook đầu tiên và thường được gọi tắt là Workbooks(1).

Workbooks("Seles.xls") sẽ đề cập đến workbook tên đó.

Trong workbook thường chứa các worksheet, trong mỗi worksheet đó chứa nhiều ô (cell). Bạn có thể đề cập đến ô B3 như sau

Workbooks("Seles.xls").Worksheets("Sheet1" ).Range("B3")

Trong lúc workbook đang làm việc thì nó được gọi là active workbook (workbook hiện hành), worksheet nào đang hiển thị thì được gọi là active worksheet. Nếu bạn có vài worksheet đang hiển thị, worksheet nào đang có trỏ (cursor) ở trong nó thì được gọi là avtive. Nếu bạn có vài workbook đang hiển thị, workbook nào đang chứa active worksheet ở trong nó thì được gọi là avtive workbook.

Nếu bạn không muốn thực hiện riêng trên workbook hay worksheet nào, VBA sẽ thực hiện trên active workbook hay active worksheet (mặc định). Còn nếu bạn thực hiện theo ý muốn, thì cần thực hiện như ở trên (Range("B3")).

Còn Sheets lựa chọn toàn bộ sheet trong workbook, kể cả chart sheets (biểu đồ) và worksheets.

Sheet("Year2006") sẽ tham chiếu đến sheet có tên là Year2006.

Chart(1) sẽ tham chiếu đến chart sheet theo thứ tự tab.

### *4.2. Các phương thức (Methods)*

Các đối tượng có các phương thức mà có thể thực hiện các hành động trong nó.

Nếu ta xét đến đối tượng là Range, ví dụ dưới đây là các phương thức có thể thực hiện:

Code:

- `Activate (Hoạt động hay hiện hành)`
- `Clear (Xoá)`
- `Copy (Sao chép)`
- `Cut (Cắt bỏ đi)`
- `Delete (Xoá nội dung trong Range)`
- `Select (Lựa chọn)`

Các phương thức có thể được coi là động từ (ví dụ: bake là động từ).

Cú pháp của câu lệnh trong VB như sau:

Object.Method (Cake.Bake)

Trong macro bạn lập như sau:

Range(“B3”).Select

### *4.3. Các thuộc tính (Properties)*

Mỗi đối tượng đều có các đặc điểm riêng. Thông thường thuộc tính điều khiển hình dáng xuất hiện của đối tượng.

Đối với đối tượng Range, các thuộc tính đặc trưng như sau:

Code:

- `ColumnWidth`
- `Font`
- `Formula`
- `Text`
- `Value`

Thuộc tính có thể được coi gần như là tính từ. Nó được thiết lập sử dụng trong câu lệnh như sau:

Object.Property = Value hay Noun.Adjective = Value

Với macro trên:

ActiveCell.FormulaR1C1 = “Nguyen Van Hung”

Tất cả các đối tượng đều được thiết lập các phương pháp (methods) và những thuộc tính (Properties) trong chúng.

Câu lệnh như

Range(“C3”).ColumnWidth = 14

sẽ thiết lập chiều rộng của cột chứa ô C3 rộng 14. Excel mặc định chiều rộng của cột là 8.43 điểm (point).

### *4.4. Các biến (Variables)*

Cũng như các ngôn ngữ lập trình khác, bạn có thể sử dụng các biến trong việc tính toán. Bình thường, VBA không yêu cầu khai báo (declare) những biến. VBA luôn tự động lưu giữ đối với những biến vào lần đầu tiên bạn sử dụng. Những biến được tạo ra tự động là các dạng của biến thể (Variant) và có thể là những kiểu dữ liệu như các chuỗi (strings), số (numbers), giá trị Boolean, các lỗi (errors), các mảng (arrays) hoặc những đối tượng (objects).

Ví dụ dưới đây là khai báo ấn định là số 34 đối với biến X.

X = 34

Trong ví dụ dưới đây, biến số Number1 và Number2 được đưa ra ở giá trị ban đầu và sử dụng chúng trong tính toán (vì chúng là số).

Number1 = 3

```
Number2 = 9
Mynumber = Number*Number2
```

### *4.4.1. Kiểu dữ liệu trong VBA*

Mỗi ứng dụng thường xử lý nhiều dữ liệu, ta dùng khái niệm biến để lưu trữ dữ liệu trong bộ nhớ máy tính, mỗi biến lưu trữ 1 dữ liệu của chương trình. Mặc dù VBA không đòi hỏi, nhưng ta nên định nghĩa rõ ràng từng biến trước khi truy xuất nó để code của chương trình được trong sáng, dễ hiểu, dễ bảo trì và phát triển. Nếu bạn cần những số liệu có đặc trưng riêng (như số nguyên, thập phân, chuỗi, mảng,...) để sử dụng trong macro, bạn có thể khai báo biến đó.

Cũng như quy định đặt tên của macro, cách đặt tên cho biến như sau:

• Tên biến có thể dài đến 255 ký tự.

• Ký tự đầu tiên phải là một ký tự chữ (letter), các ký tự tiếp theo có thể là các ký tự chữ (letter), ký số (digit), dấu gạch dưới (_).

• Tên biến không được chứa các ký tự đặc biệt như các ký tự : ^, &, ),(,%, $, #, @, !, ~, +, -, *, …

• VBA không phân biệt chữ HOA hay chữ thường trong tên biến.

• Nên chọn tên biến ngắn gọn nhưng thể hiện rõ ý nghĩa.

• Khi viết tên biến ta nên viết hoa chữ đầu tiên của một từ có ý nghĩa.

• Không được dùng tên biến trùng với các từ khoá như : Print, Sub, End…(từ khóa là những từ mà ngôn ngữ VBA đã dùng cho những thành phần xác định của ngôn ng

### *4.4.2. Khai báo kiểu dữ liệu*

Cách khai báo biến số:

Dim variable_name As data_type

Có các kiểu dữ liệu (data_type) được trình bày như sau:

| Kiểu dữ liệu | Mô tả |
|---|---|
| Byte | Lưu giữ số nguyên dương nhỏ, từ 0 đến 255 |
| Boolean | Lưu giữ kết quả logic, True hoặc False |
| Integer | Lưu giữ giá trị nguyên, từ -32,768 đến 32,767 |
| Long | Lưu giữ giá trị nguyên, từ -2,147,483,648 to 2,147,483,647 |
| Single | Lưu giữ giá trị số, từ -3.402823E38 đến -1.401298E-45 ; từ 1.401298E-45 đến 3.402823E38 |
| Currency | Lưu giữ số liệu kiểu tiền tệ, áp dụng cho lĩnh vực tài chính, kế toán. Từ - 922337203685477.5808 đến 922337203685477.5807 |
| Date | Lưu giữ số liệu kiểu thời gian, từ January 1, 100 tới December 31, 9999. Dữ liệu kiểu Date phải có dấu # ở hai đầu. |
| Object | Chứa tham khảo đến bất kỳ đối tượng nào |
| String | Lưu giữ dưới dạng chuỗi, dài của chuỗi từ 0 đến 65535 ký tự, giá trị chuỗi được đặt trong dấu " " |
| Variant | Lưu mọi dữ liệu thuộc kiểu định sẵn như Date, String, Double, Integer... Nếu không khai báo kiểu rõ ràng cho 1 biến thì biến này sẽ được hiểu là thuộc kiểu này. VBA sẽ chuyển đổi dữ liệu thuộc kiểu Variant thành một kiểu dữ liệu khác cho phù hợp (khi gán dữ liệu,...). |

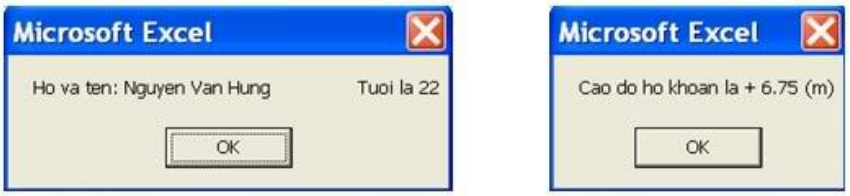
Khai báo biến số là thủ tục tác động đến quy trình xử lý và không bị thay đổi bởi thủ tục khác. Những biến số mà vượt quá vùng của loại dữ liệu quy định (trong bảng trên) thì biến số đó bị lỗi Overflow (tràn bộ nhớ).

Ví dụ về sử dụng Dim trong khai báo biến số:

```
Sub Kieudulieu()
Dim Tuoi As Integer ‘ Tuổi là số nguyên
Dim Caodo As Single ‘ Cao độ là số
Dim Ten As String ‘ Tên người là chuỗi
Tuoi = 22 ‘ Khai báo từng giá trị Tuoi, Caodo, Ten
Caodo = 6.75
Ten = "Nguyen Van Hung"
MsgBox "Ho va ten: " & Ten & vbTab & vbTab & "Tuoi la " & Tuoi
MsgBox "Cao do ho khoan la + " & Caodo & " (m)"
```

End Sub

Kết quả thể hiện ở hình vẽ dưới đây:




*Hình vẽ 15: Kết quả thể hiện kiểu dữ liệu*

Với những giá trị không thay đổi thì nên thiết lập như những hằng số (constant). Điều đó ngăn cản chúng bị biến đổi do nhầm lẫn.

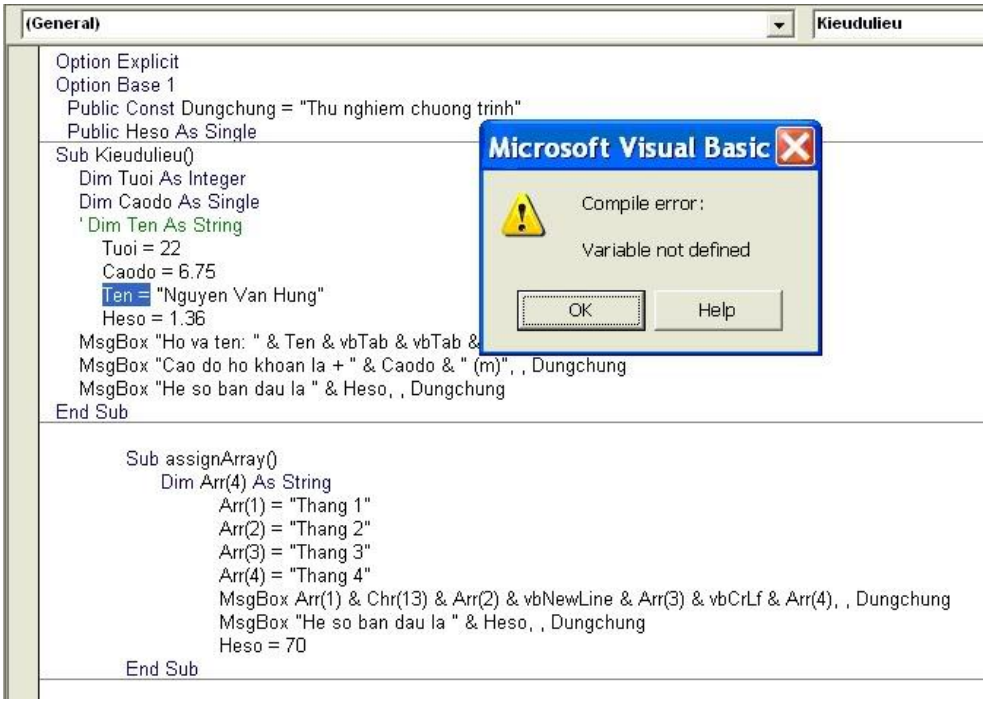
Ví dụ:

Const Pi = 3.14159

Dientich = Pi*2

Nếu bạn muốn thủ tục (Sub) khác truy cập những biến số đó, hãy khai báo chúng ở dòng đầu tiên của Module, trên cả câu lệnh Sub (hình 16). Trường hợp này hay sử dụng khi bạn có một biến số dùng chung cho chương trình.

Ngoài ra, bạn có thể yêu cầu phải khai báo toàn bộ biến số bằng cách sử dụng Option Explicit. Nếu có biến nào chưa được khai báo, VBA sẽ báo lỗi ngay (hình 16)




*Hình vẽ 16: Khai báo Option Explicit và biến dùng chung ở trên cùng*

Trong ví dụ tiếp theo, biến số đã khai báo ở giá trị ban đầu (bằng 0) và sau khi sử dụng phương pháp đếm các ô trong vùng B1:B10 thoả mãn điều kiện giá trị (value) trong ô đó nhỏ hơn 40. Biến số D sẽ bị thay đổi.

Sub VD_Bienso()

Dim Marks As Range

Dim C, D As Integer

Set Marks = Range("B1:B10")

D = 0

For Each C in Marks

If C.value < 40 then

D = D + 1

End If

Next C

MsgBox "Gia tri moi cua bien so D la " & D

End Sub

thay đổi nội dung bởi: **PhanTuHuong**, 24-11-06 lúc 10:28 AM Lý do: sửa lỗi

### *4.5. Sử dụng mảng (Array)*

Mảng là kiểu dữ liệu đặc biệt và hay được ứng dụng trong việc thống kê, tính toán,... nên được trình bày ở mục riêng. Các mảng (Arrays) chứa các biến số được sắp xếp theo trình tự quy định. Mỗi biến số được gọi là phần tử của mảng. Mảng có biên trên và biên dưới, các phần tử trong mảng là liên tục. Ví dụ như danh sách học sinh trong một lớp, giá trị chỉ tiêu đơn lẻ trong đối với một chỉ tiêu trong mẫu. Có hai loại biến mảng: mảng có chiều dài cố định và mảng động.

#### *4.5.1. Mảng có chiều dài cố định*

Thủ tục Dim có thể sử dụng để khai báo trong mảng có chiều dài cố định mà không cần đưa giá trị nào vào.

Ví dụ:

Code:

```
Dim Arr(4)
Dim Myfriends(1 to 30) As String
Dim Noisuy(1 to 20, 1 to 30) As Single
```

Mảng Arr(4) tạo ra mảng 1 chiều chứa 5 phần tử. Với kiểu khai báo này (4), phần tử đầu tiên (biên dưới) là Arr(0). Để phần tử đầu tiên bắt đầu từ 1 thì bạn phải khai báo Option Base 1 trên đầu của thủ tục (Sub).

Mảng Myfriends tạo ra mảng 1 chiều chứa được 30 chuỗi (là tên người).

Mảng Noisuy tạo ra mảng 2 chiều với kích thước cạnh 20 x 30 (tương ứng 600 giá trị là số).

Hàm số có tên là Array có thể tạo nên mảng từ các biến số trong nó.

Dim Array(“Michael”, “David”, “Peter”, “Jackson”)

Khi sử dụng hàm Array, những biến số mặc định là kiểu biến Variant.

Để xác định thông số của hàm Array, phổ biến dùng 2 hàm sau:

• Hàm UBound trả về phần tử cuối cùng của mảng.

• Hàm LBound trả về phần tử đầu tiên của mảng.

Ví dụ: Hình 17 là kết quả của Sub dưới đây

```
Option Base 1
Sub assignArray()
Dim Arr(4) As String
Arr(1) = "Thang 1"
Arr(2) = "Thang 2"
Arr(3) = "Thang 3"
Arr(4) = "Thang 4"
MsgBox Arr(1) & Chr(13) & Arr(2) & vbNewLine & Arr(3) & vbCrLf & Arr(4)
End Sub
```


*Hình 17: Các phần tử trong mảng*

Hàm Msgbox sẽ cho hiện hộp thông báo như bên cạnh, các bạn sẽ học ở mục 11.1.

Ngoài ra các bạn còn thấy hàm Chr(13), vbNewLine, vbCrLf có cùng tác dụng là ngắt dòng trong hộp thoại (giống như phím Enter ngắt dòng trong Word).

### *4.6. Sử dụng With - End With*

With - End With dùng để thực hiện nhiều thao tác đối với đối tượng đơn lẻ. Phương pháp này được sử dụng đối với đối tượng nào có nhiều thuộc tính. Để hiểu được cách sử dụng With - End With trong công việc, ví dụ dưới đây thể hiện quy trình thực hiện. Chương trình con này sẽ làm thay đổi 5 thuộc tính của vùng định dạng.

Code:

```
Sub ChangeFont1()
  Selection.Font.Name = "Times New Roman"
  Selection.Font.FontStyle = "Bold Italic"
  Selection.Font.Size = 12
  Selection.Font.Underline = xlUnderlineStyleSingle
  Selection.Font.ColorIndex = 5
End Sub
```

Trong thủ tục trên, bạn thấy đoạn Selection.Font. được lặp lại nhiều lần và bạn có thể viết lại khi sử dụng With - End With. Dưới đây là thủ tục đã sửa lại:

Code:

```
Sub ChangeFont2()
  With Selection.Font
    .Name = "Times New Roman"
    .FontStyle = "Bold Italic"
    .Size = 12
    .Underline = xlUnderlineStyleSingle
    .ColorIndex = 5
  End With
End Sub
```

Bạn sẽ thấy khi sử dụng With- End With, việc quản lý các đối tượng và thuộc tính của chúng dễ dàng hơn.

## 5. Sử dụng giúp đỡ Help

Trong quá trình viết macro, chắc chắn bạn phải cần đến trợ giúp. Không có sách nào có thể viết được hết về VBA nói riêng và các ngôn ngữ lập trình nói chung, vì những kiến thức trong đó rất rộng lớn. Vì vậy bạn nên sử dụng tính

năng Help của VBA. Điều cơ bản nhất để sử dụng Help là bạn phải biết tiếng Anh để đọc và hiểu được các hướng dẫn đó. Những người có trình độ về lập trình cao như tôi biết đều chủ yếu sử dụng sách tiếng Anh và đọc trong Help. Nội dung trình bày dưới đây sẽ cho các bạn hiểu được mức độ tiện dụng của Help như thế nào.

### *5.1. Tại thời điểm đang viết code*

Trong quá trình viết macro tại cửa sổ Microsoft Visual Basic, bạn có thể truy cập vào help tại những mục chọn chi tiết (ví dụ như Select) như sau:

- Chọn mục cụ thể (di chuyển chuột vào chữ Select).

- Sau đó ấn phím F1. Khi đó hiện cửa sổ Microsoft Visual Basic Help như hình 19.

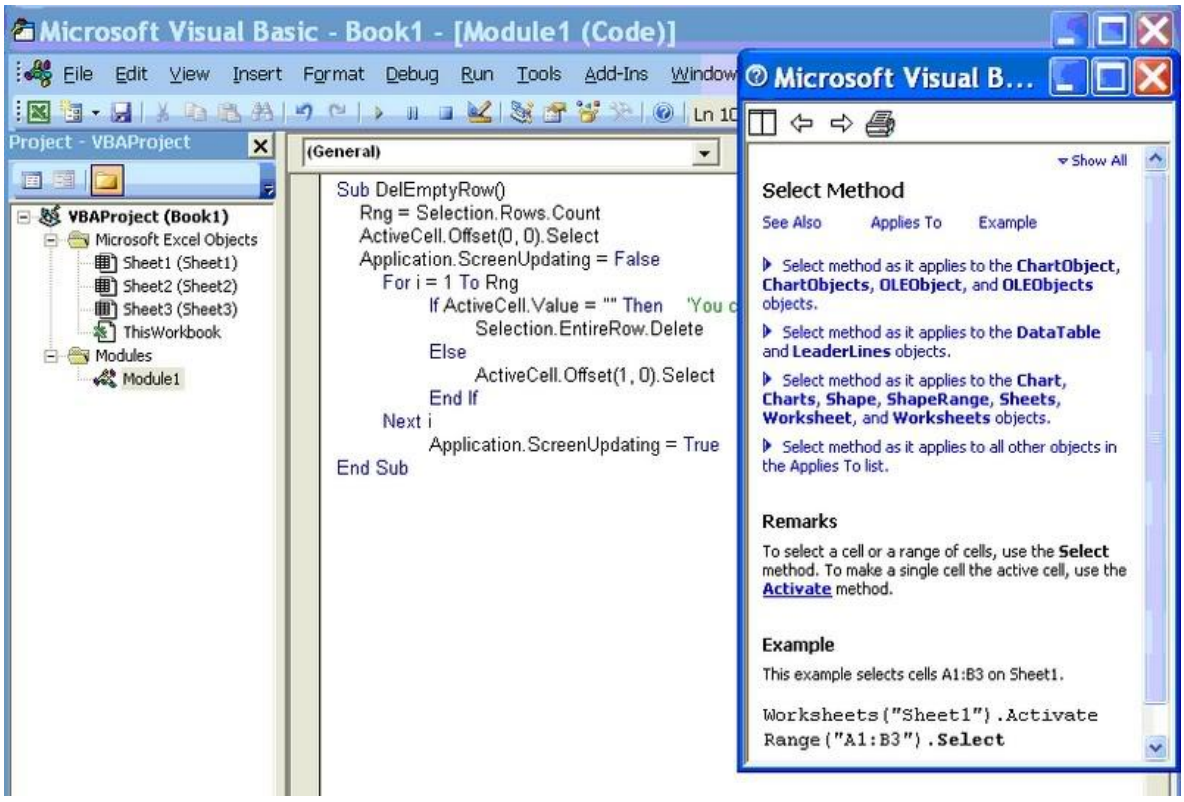
### *5.2. Sử dụng hộp thoại giúp đỡ với chủ đề cụ thể*

Để sử dụng hộp thoại giúp đỡ với chủ đề (topic) cụ thể, bạn thực hiện các bước sau:

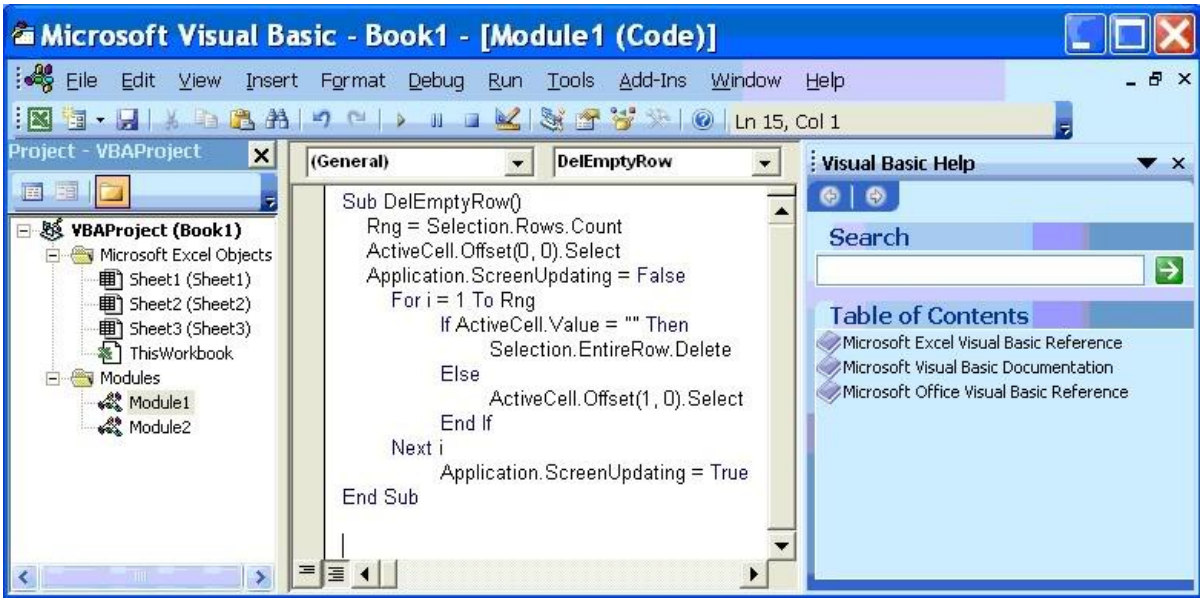
- Vào cửa sổ Microsoft Visual Basic đang mở (nếu chưa mở thì bạn vào menu Tools/Maros/Visual Basic Editer hoặc phím tắt Alt + F11).

- Từ menu Help, bạn chọn Microsoft Visual Basic Help.

- Bạn có thể thực hiện bằng cách sử dụng chức năng Search (ví dụ gõ nội dung "commandbar", rồi Enter) hoặc có thể chọn chủ đề mà bạn đang cần tìm trong danh mục.



*Hình 19: Cửa sổ Microsoft Visual Basic Help*




*Hình 20: Sử dụng Visual Basic Help*

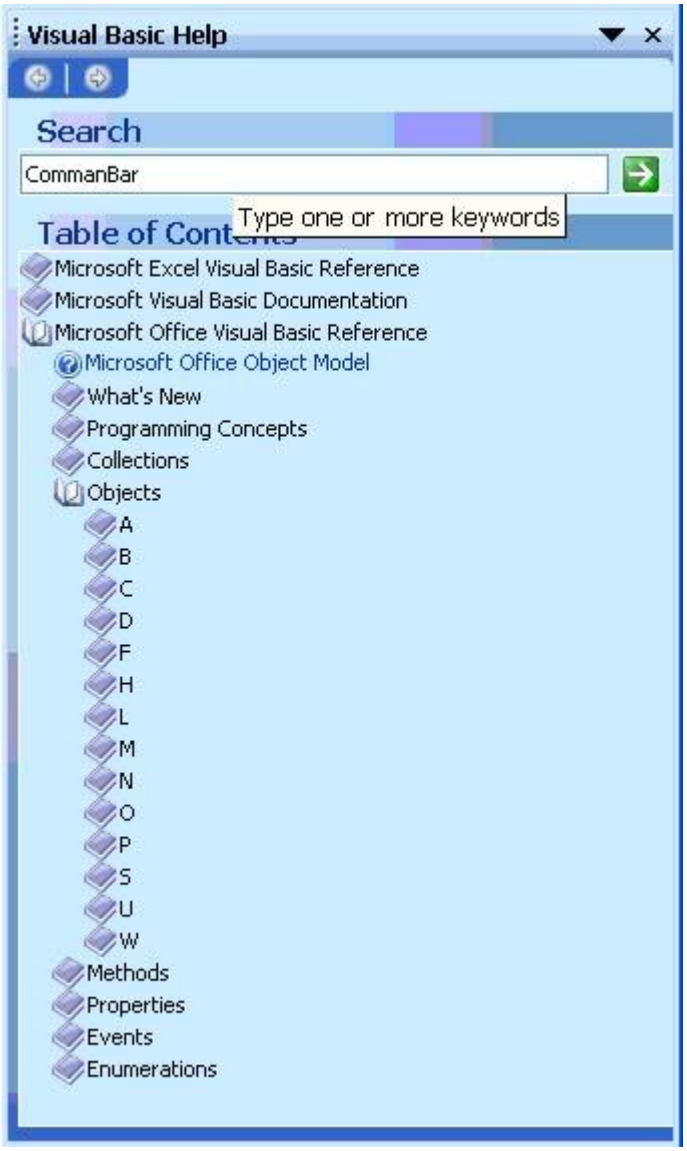
## *5.3. Trình duyệt đối tượng*

Phương thức trình duyệt đối tượng (Object Browser) được sử dụng để xem các đối tượng, các phương pháp và những thuộc tính trong việc bổ sung thêm các hàm số (functions) và các lệnh (statements) được xây dựng trong Visual Basic for Excel.

Code:

1. Vào cửa sổ Microsoft Visual Basic đang mở.

2. Từ menu View, bạn chọn Object Browser (hoặc ấn phím F2).




*Hình 21: Chọn chủ đề cụ thể bằng search hoặc trong Table of contents*

Sau đó cửa sổ hiện ra ở giữa chứa danh mục các nhóm - lớp (classes) khác nhau của đối tượng.

Nhóm (class) chính là phần mô tả các dạng của đối tượng (ví dụ như particular chart thuộc về nhóm Chart) . Nhóm thuộc dự án (project) hay thư viện (library).

Code:

```
3. Bấm vào hình tam giác đi xuống bên cạnh <All Libraries>
và chọn Excel (hình 19). Khi đó các nhóm thuộc Excel sẽ xuất
hiện.

4. Trong vùng Classes, ban cuộn xuống và chọn Range.

5. Trong Members of 'Range' bạn bấm vào ClearContents.

6. Chỉ dẫn ở bên cạnh ClearContents mà có ký hiệu màu xanh
cho biết đối tượng đó là phương pháp (method).
```

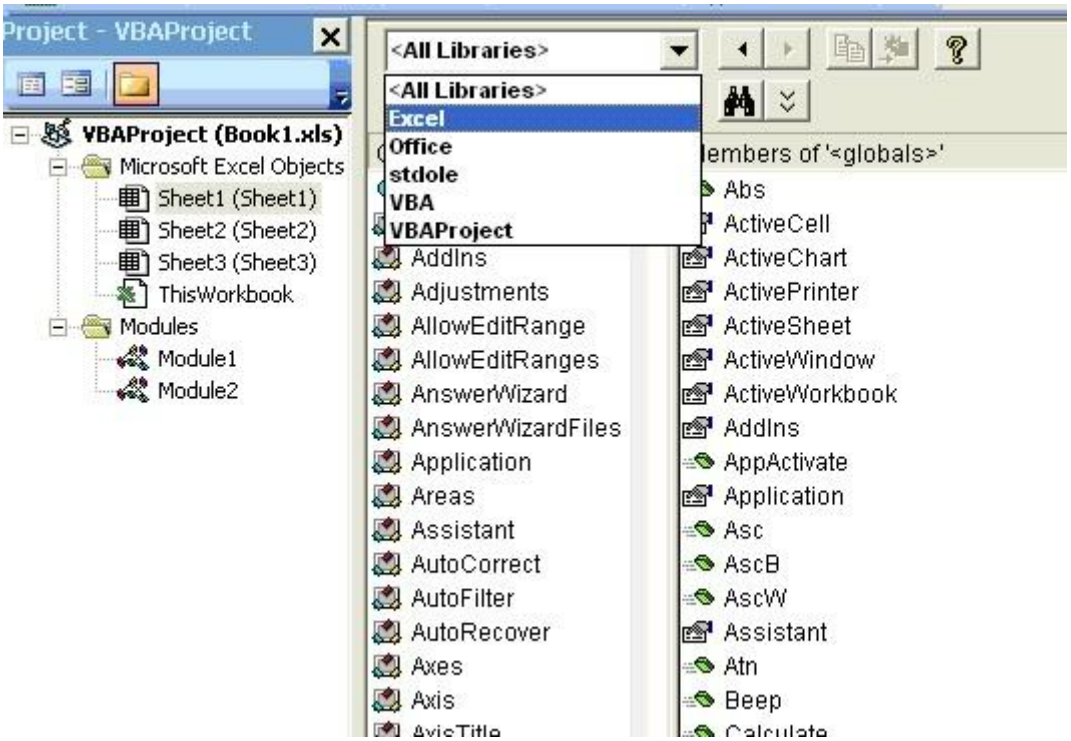
7. Nếu muốn biết thêm thông tin về ClearContents, bạn bấm vào nút Help (hình dấu ? màu vàng).

8. Để xem ví dụ, bạn ấn vào Example màu xanh (hình 20).

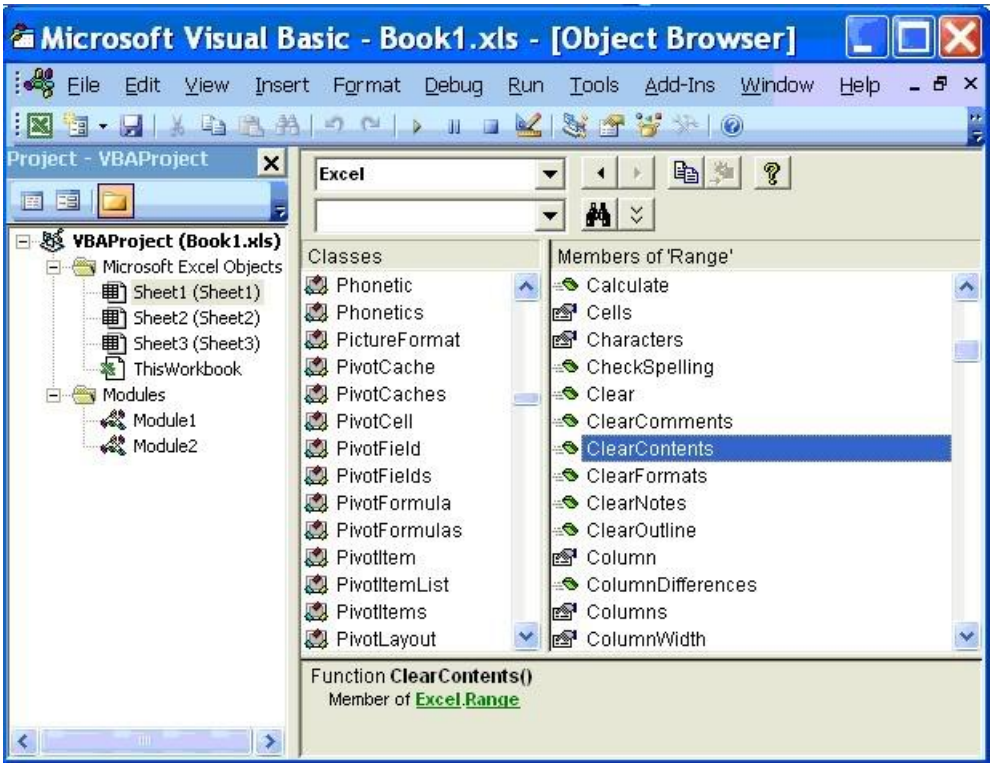
9. Đóng cửa sổ hướng dẫn sử dụng ClearContents và tiếp tục cuộn để tìm các thành phần khác trong Members of 'Range'.

10. Chỉ dẫn mà có biểu tượng khác ở bên cạnh (hình bàn tay chỉ) cho biết hàm đó là thuộc tính (property).

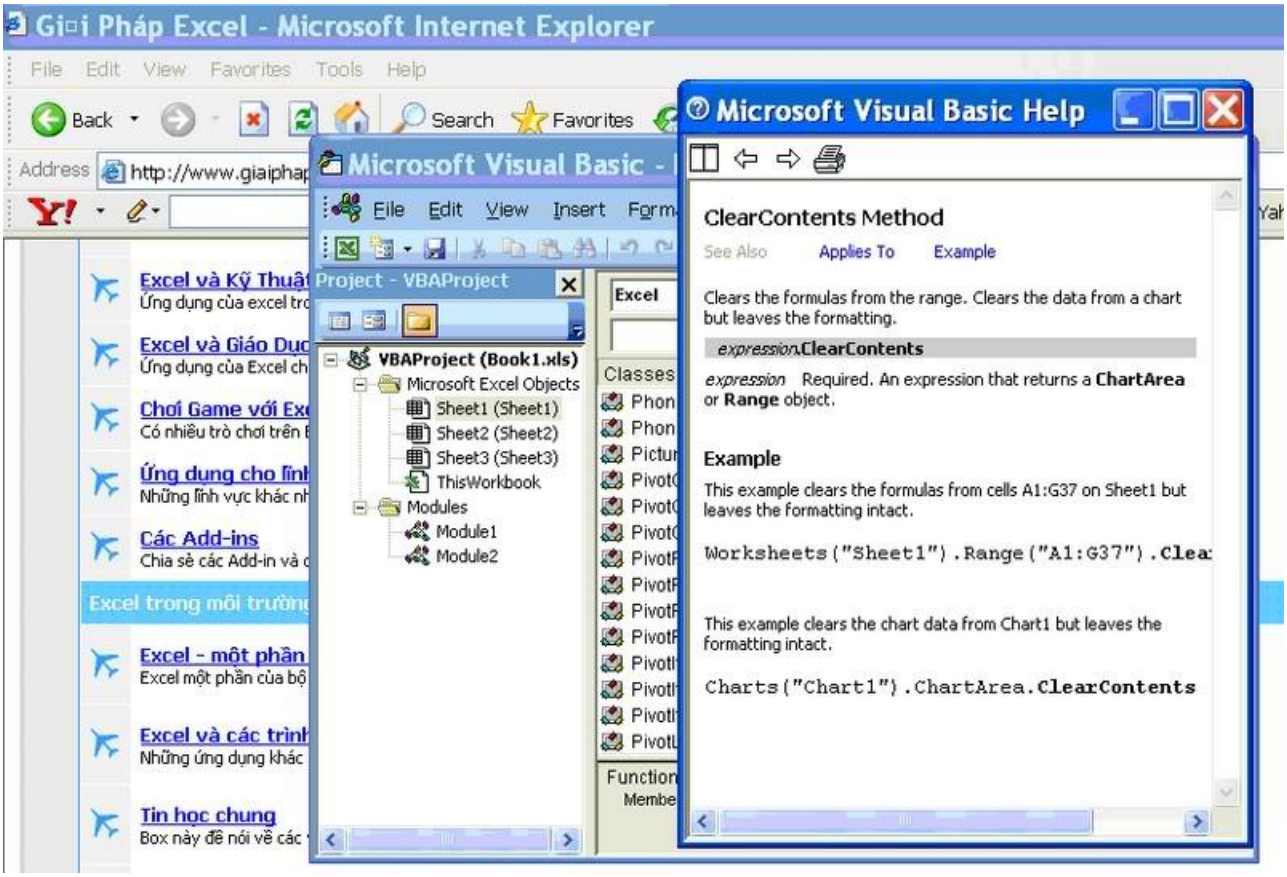
11. Đóng cửa sổ Visual Basic Object Browser vào.




*Hình 22: Cửa sổ Object Browser*




*Hình 23: Các nhóm thuộc đối tượng Excel*



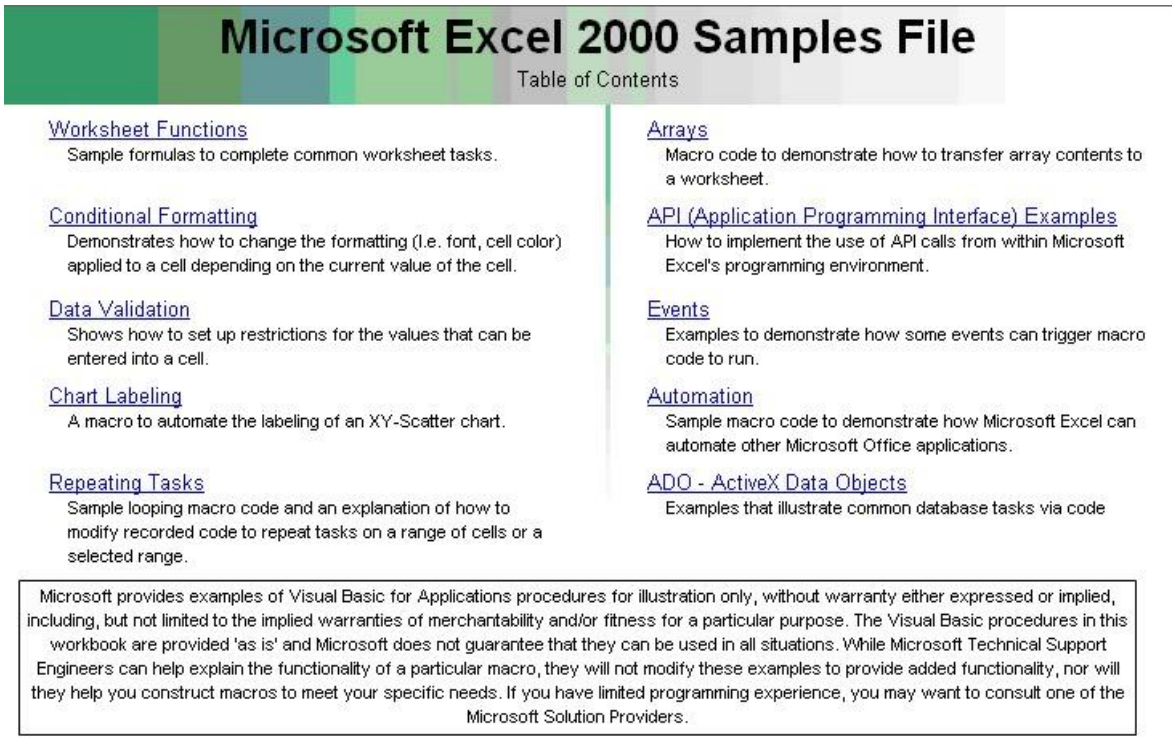

*Hình 24: Cửa sổ Help đối với các đối tượng trong Excel.*

## *5.4. Các file ví dụ*

Excel đưa ra một số file ví dụ có tên là Samples.xls. Hầu hết đối với mỗi phiên bản Excel khác nhau thì đường dẫn đến file này cũng khác nhau. Trong Excel 2003, bạn có thể tìm thấy file này tại đường dẫn sau:

***C:\Program Files\Microsoft Office\Office10\Samples.xls ???***

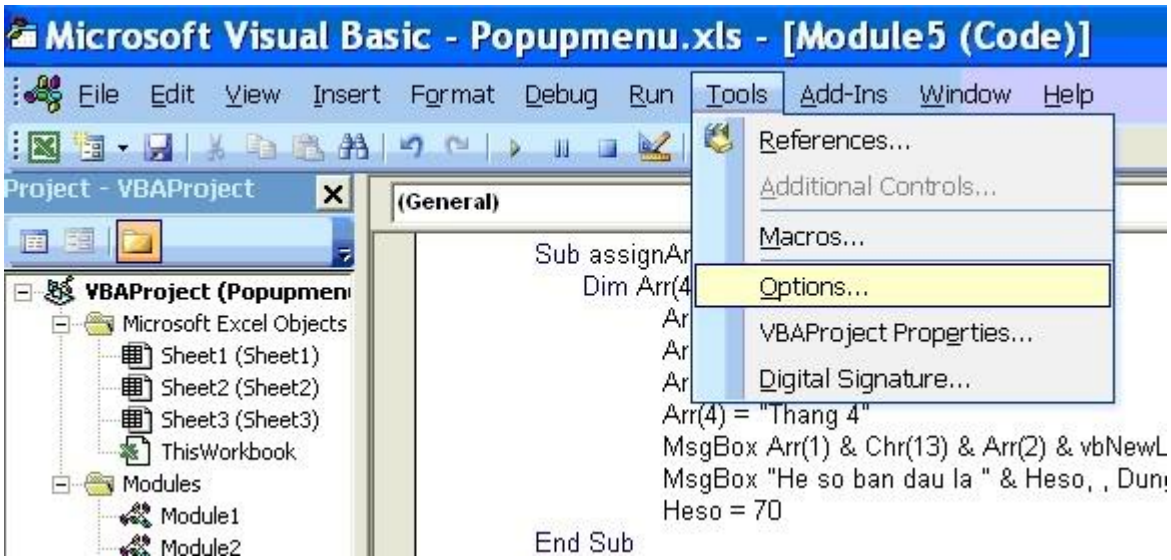
Samples.xls đề cập nhiều ứng dụng của Excel. Mỗi sheet hướng dẫn một phần công việc. Nội dung có liên quan đến VBA là Chart Labeling, Repeating Tasks, Arrays, API examples, Events, Automation, ADO,... Hình ảnh về file Samples. xls trong Excel 2000.



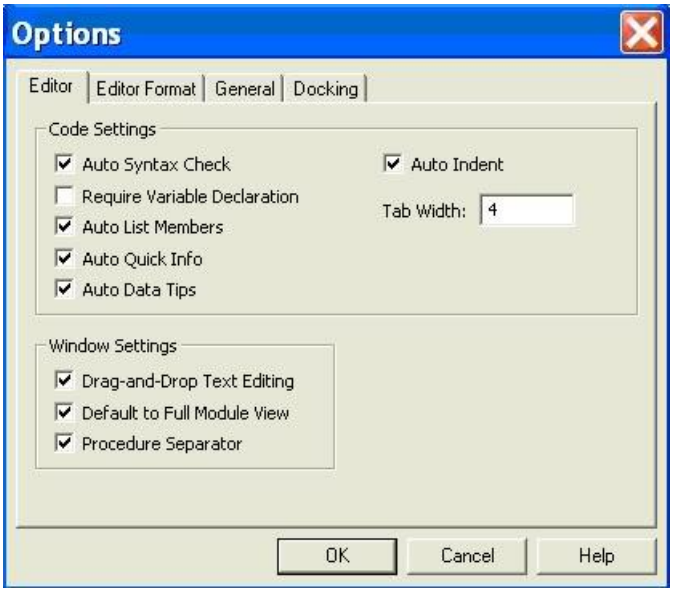

*Hình 25: Nội dung File ví dụ Samples. xls*

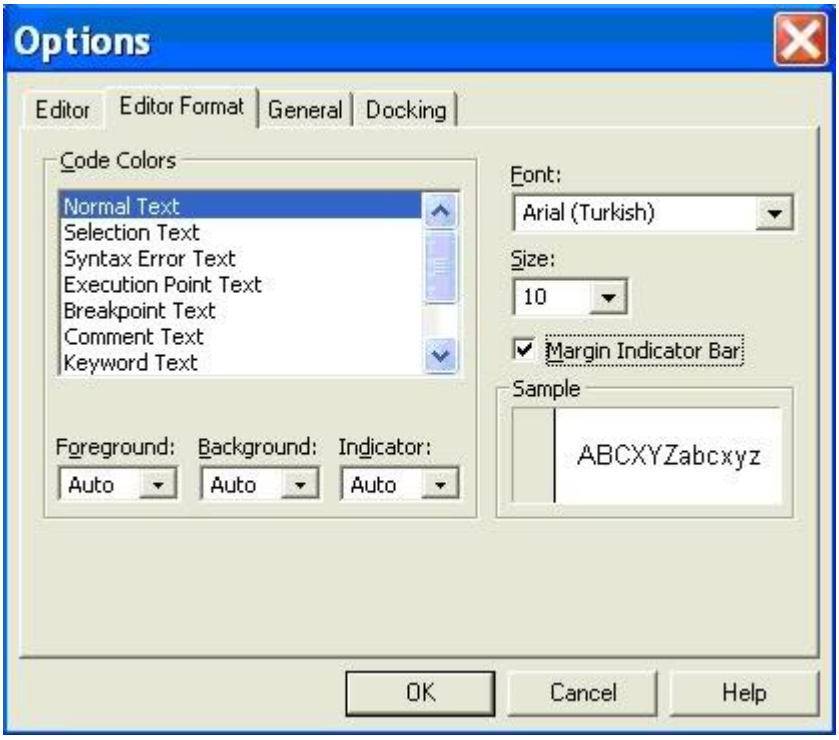
## 6. Một số chức năng điều khiển trong VBA

Cũng như VB, VBA có những tính năng điều khiển trong quá trình viết code rất thuận lợi. Bạn có thể tuỳ biến thay đổi những thông báo hoặc giao diện của cửa sổ soạn code Microsoft Visual Basic (MSB).




*Hình 26: Sử dụng Options trong menu Tools của MVS*





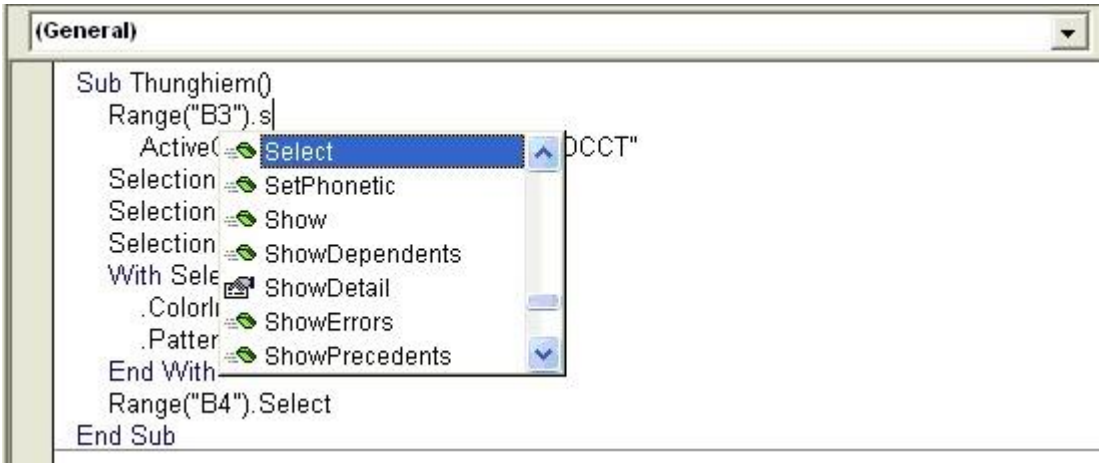

*Hình vẽ 27: Cửa sổ Editor và Editor Format trong Options*

### *6.1. Sử dụng Options*

Bạn vào menu Tools, chọn Option (hình 26), cửa sổ Options hiện ra như hình 27. Trong Editor có các lựa chọn chính sau:

• **Auto Syntax Check:** Trong trường hợp không chọn, mà khi bạn thực hiện sai thì dòng đó có màu đỏ, không xuất hiện hộp thông báo như hình 37.

• **Require Variable Declaration:** Khi được chọn, dòng Option Explicit luôn xuất hiện ở đầu Module.



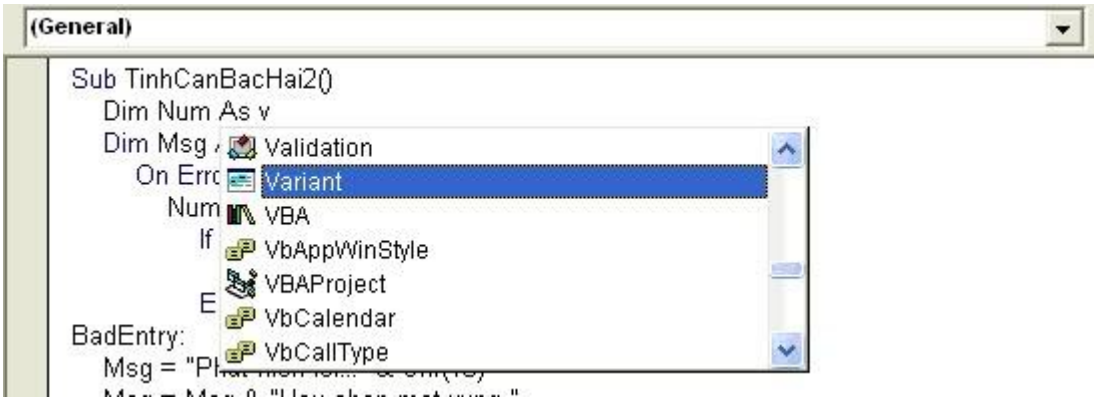

*Hình 28: Cửa sổ Auto List Members*

• **Auto List Members:** Khi được chọn, VBA sẽ tự động cho hiện danh sách các thuộc tính và phương thức của một điều khiển hay một lớp, khi ta gõ vào tên của điều khiển đó (hình 28).

• **Auto Quick Info:** Tương tự như trên, nhưng nó hiển thị cú pháp của 1 hàm hay thủ tục, tham số đầu tiên được in đậm.


*Hình 29: Cửa sổ Auto Quick Info*

• **Auto Data Tip:** Hiển thị danh sách các dữ liệu khi khai báo biến.




*Hình 30: Cửa sổ Auto Data Tip*

Trong cửa sổ Editor Format có các lựa chọn chính sau:

• **Code Color:** Bạn có thể lựa chọn màu chữ, màu nền của từng loại code, mặc định là Auto.

• **Font:** Chọn loại font chữ tuỳ ý thích, mặc định là font Courier New.

• **Size:** Chọn kích cỡ font chữ, phù hợp với người mắt kém 😊😊.

**Ngoài ra còn một số lựa chọn khác, các bạn tự tìm hiểu.**

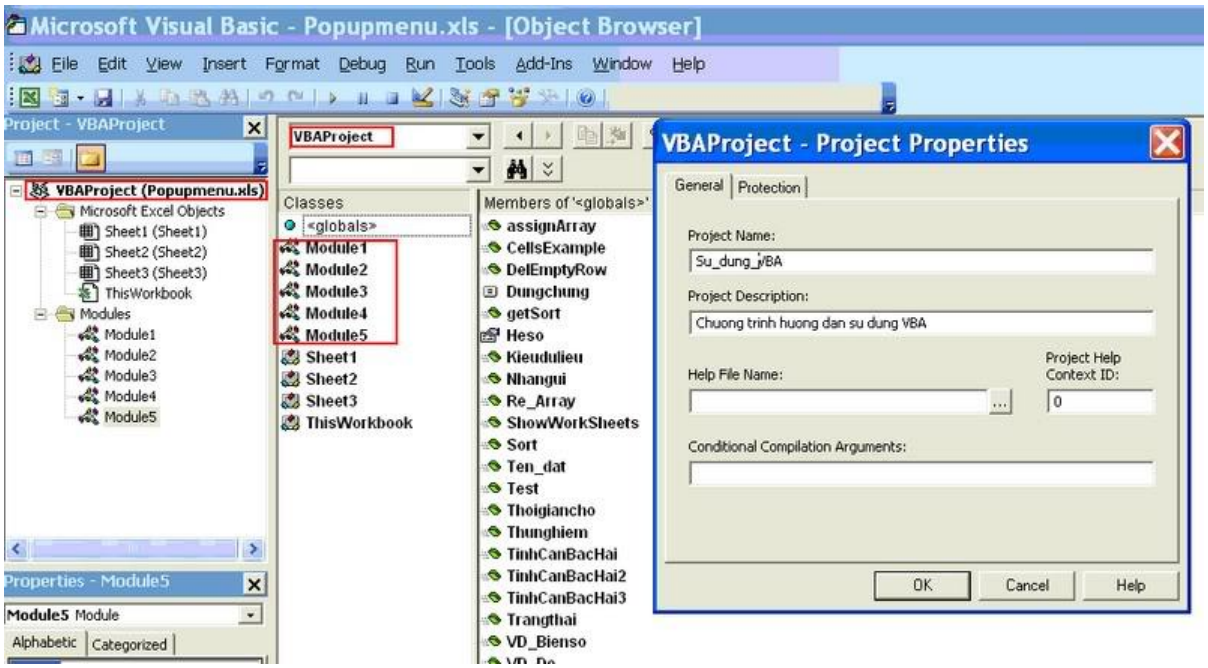
**Ghi chú:** VBA có rất nhiều đối tượng, phương thức và thuộc tính. Bạn không thể nào biết được hết hoặc sẽ bị quên. VBA cung cấp cho bạn những tính năng gợi nhớ trên giúp các bạn có thể khai thác tốt hơn VBA.

### *6.2. Sử dụng VBAProject*

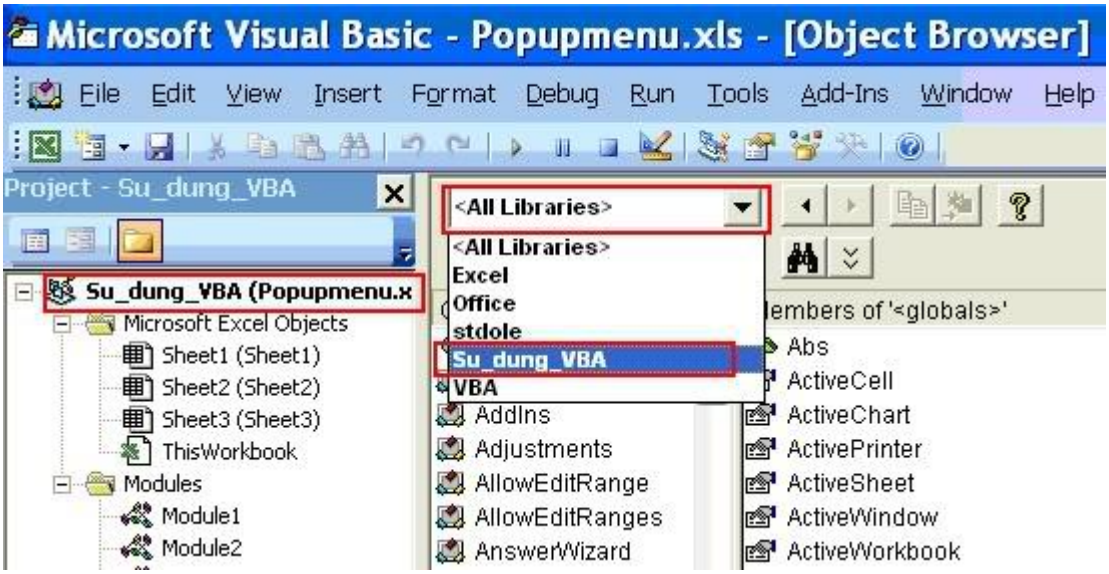
Trong menu Tools, chọn VBAProject Properties, cửa sổ VBAProject hiện ra như hình 31.

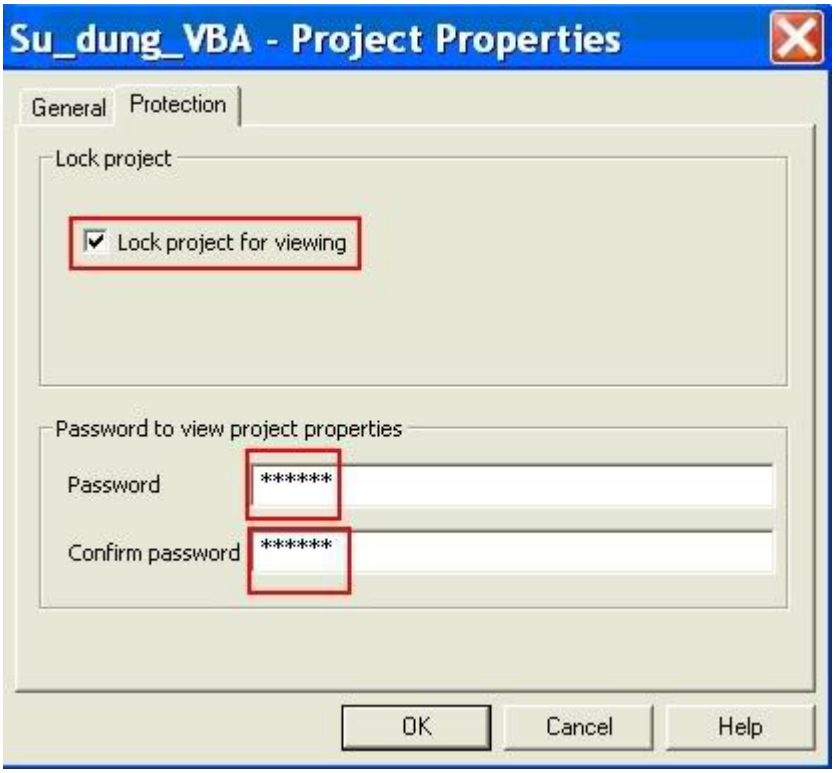
• **VBProject:** Để nhận dạng dự án của bạn trong Window Registry (khai báo trong Window) và trong Object Browser. Điều quan trọng là nó có tên duy nhất.

• **Project Description:** Mô tả tên của dự án của bạn trong Type Library. Thư viện Type Library chứa toàn bộ những mô tả về đối tượng và giao diện của dự án của bạn.




*Hình 31: Sử dụng VBAProject Properties*






*Hình 32: Dự án Su_dung_VBA trong <All Librairies> và bảo vệ code trong Protection*

Có những dự án (project) của bạn lập ra mà không muốn người khác xem code, bạn có thể khoá lại. Để thực hiện công việc này, trong tab Protection bạn lựa chọn như sau:

• **Lock project:** Khoá code trong module, không cho nhìn thấy và không cho sửa chữa. Bạn phải chọn mục Lock project for viewing.

• **Password to view project properties:** Bạn phải gõ nội dung mã khoá trong hộp Password, nội dung mã khoá biến thành dấu sao *. Sau đó, bạn phải xác nhận nội dung mã khoá trong Confirm password bằng cách gõ lại nội dung mã khoá vừa vào. Nếu bạn gõ không đúng nội dung, VBA sẽ báo lỗi và bạn phải gõ lại cho đúng. Số ký tự tối đa là 24 ký tự, có thể là số, chữ và các ký tự đặc biệt.

Sau đó, mỗi khi mở file trên, để có thể xem được code, bạn vào menu Tools/Macro, chọn Visual Basic Editer (hoặc ấn Alt + F11). Cửa sổ Microsoft Visual Basic hiện ra, tuy nhiên toàn bộ nội dung code đều không hiện ra (hình vẽ 33). Để xem được nội dung code, bạn nháy kép vào Su_dung_VBA Project, cửa sổ Su_dung_VBA Password hiện ra. Bạn phải khai báo đúng Password thì nội dung code mới hiện ra.


*Hình 33: Hộp thoại hỏi mã khoá khi bạn mở Project bị khoá.*

Ghi chú: Lưu ý khi sử dụng mã khoá, nếu bạn quên thì sẽ không thể mở được project. Vì vậy, bạn phải nhớ nội dung mã khoá và nên chọn nội dung nào dễ nhớ. Nếu muốn đổi mã khoá thì bạn vào cửa sổ Password để thay đổi.

### *6.3. Sử dụng chức năng Security*

Mấy năm gần đây, do virus macro phát triển nên Microsoft đã bổ sung thêm chế độ an toàn trong các ứng dụng. Chức năng Security điều khiển sự làm việc của macro, tức là có thể cho hoạt động hoặc không.

1. Bạn vào menu Tools/Macro và chọn Security (hình 34), cửa sổ Security hiện ra.

2. Trong Security, tại tab Security Level có 4 trường hợp chọn như sau:

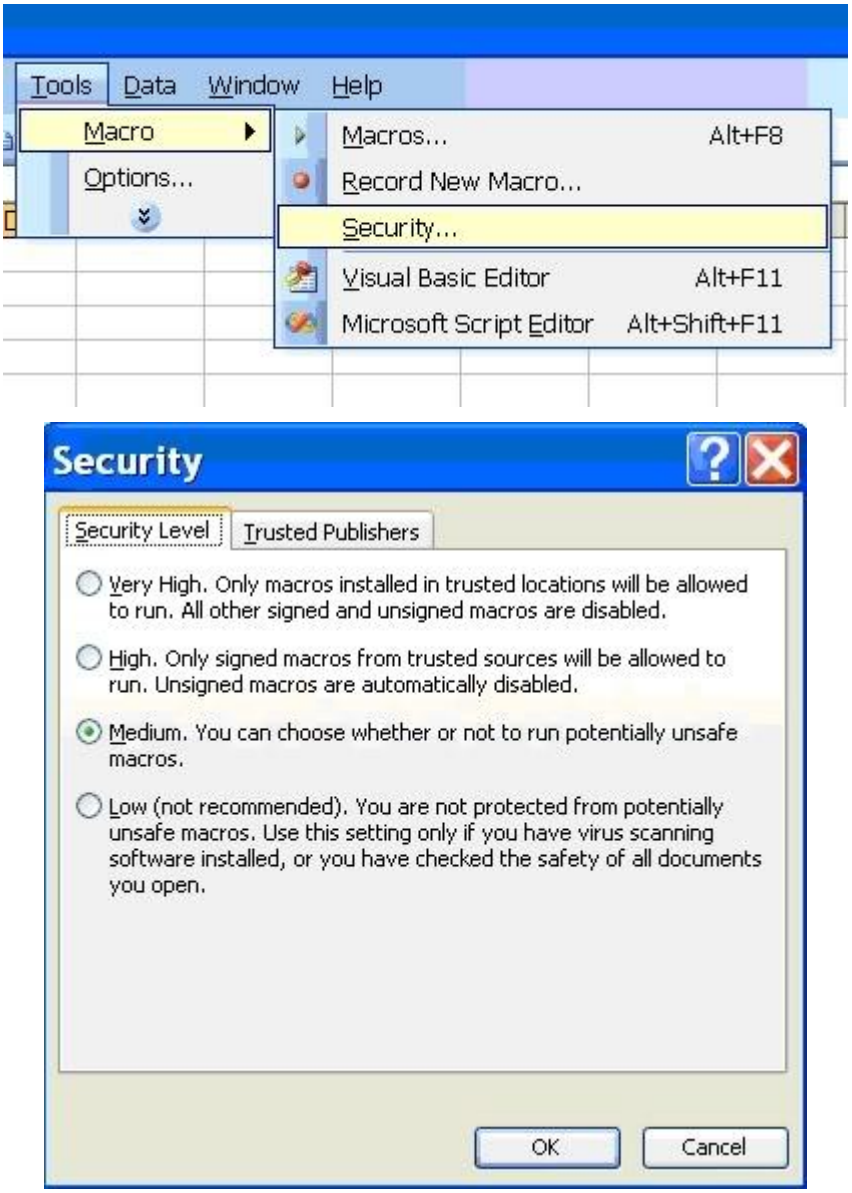
- Very High: Đặt chế độ an toàn rất cao, các macro không thể chạy được, chỉ trừ macro của Office.

- High: Chỉ những macro được xác nhận mới có thể chạy, các macro khác cũng bị vô hiệu hóa. Để có macro được xác nhận, bạn phải đăng ký trong menu Tools\Option\Securitry\More Macro\Trusted Publisher.

- Medium: Đặt chế độ an toàn trung bình. Khi chọn trường hợp này, nếu bạn mở file có chứa macro thì nó sẽ cảnh báo như hình 35. Bạn có thể lựa chọn Enable Macros để cho macro hoạt động hoặc Disable Macros để macro không hoạt động. Trong trường hợp file của bạn không sử dụng macro (thủ tục hay hàm tự tạo) mà khi mở Excel cảnh báo như hình 31 thì file của bạn bị nhiễm virus macro.

- Low: Không đặt chế độ an toàn, tức là Excel không cảnh báo bất cứ vấn đề gì cả.

3. Như vậy, khi bạn sử dụng VBA thì nên đặt Security Level ở mức độ Medium hoặc Low. Khi đó các thủ tục, hoặc hàm mới hoạt động được.




*Hình 34: Vào menu Security và cửa sổ Security*


*Hình 35: Cảnh báo macro chứa trong file*

## 7. Viết macro

Khi bạn tiến hành ghi (record) macro, Excel sẽ tự động tạo module và bổ sung nó vào trong workbook và viết lại những hành động bạn đã ghi thuộc về module đó.

Khi bạn muốn viết mã (code) trong workbook, bạn có thể bổ sung module trong workbook đó. Sự ghép nối cho phát triển macro được gọi là Visual Basic Integrated Development Evironment (IDE). Macro có trong module được hiện ra trong IDE thay thế cho bảng tính trong workbook (như Excel đời trước 97).

### *7.1. Viết macro*

Trước tiên chuyển sang workbook mới (nhưng cho phép workbook cũ đó vẫn mở) như sau:

*1. Tiếp theo bấm chuột vào nút New trong thanh công cụ (toolbar), hoặc vào menu File rồi chọn New.*

*2. Bấm chuột phải tại tên của Sheet1 và chọn Rename trong menu tắt.*

*3. Gõ nội dung Text rồi ấn Enter.*

*Viết macro:*

*1. Từ menu Tools/Macros bạn chọn Visual Basic Editor.*

*2. Trong cửa sổ Microsoft Visual Basic bạn vào menu Insert và chọn Module (hình 36).*

*3. Nếu cần, bạn có thể thay đổi tên của module theo ý muốn. Trong cửa sổ Properties, bên cạnh (Name) bạn chọn Module1 và sửa thành Chuongtrinh.*

*4. Bấm vào vùng trống của cửa sổ Chuongtrinh (phần code).*

*5. Gõ Sub MyFirst rồi bấm Enter. Khi đó Excel sẽ tự động điền () và End Sub, thể hiện như hình 36.*

*6. Gõ các lệnh từng bước một theo sự mô tả ở dưới. Bạn có thể có được những giúp đỡ trong Sub Address_abs() tại mục 4 và hình 13.*

Trước đó, macro của bạn chứa các lệnh đơn giản.

- Bước 1: Chọn sheet có tên Text (dùng Sheets("Text").Select)

- Bước 2: Gõ đoạn **I can write macros!** trong ô B2 trong sheet đó.

- Bước 3: Bôi đậm chữ.

Cuối cùng, bạn kiểm tra (test) lại macro Text:

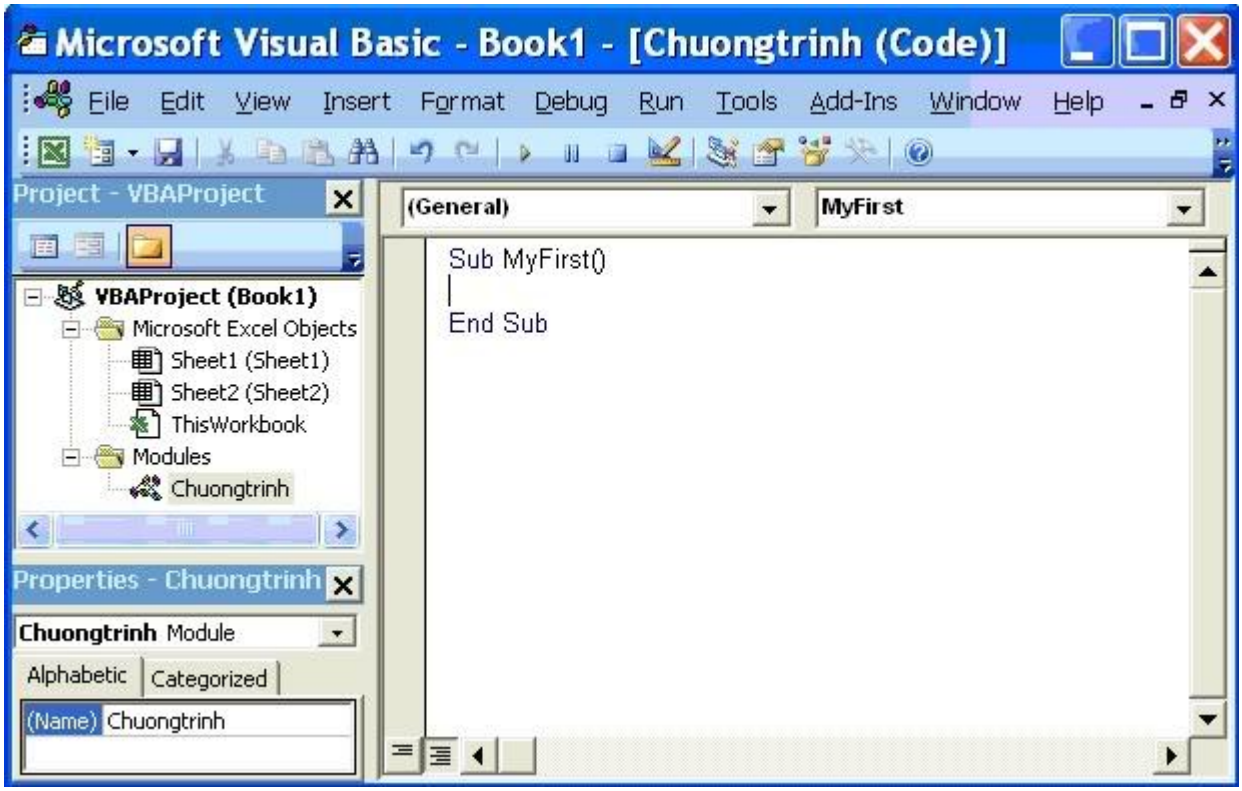
*1. Quay trở về sheet Text.*

*2. Từ menu Tools/Macros chọn Macros.*

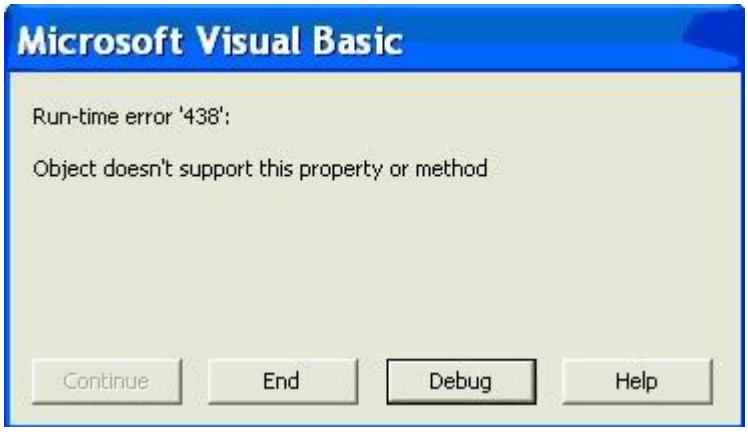
*3. Trong cửa sổ Macros, bạn chọn macro có tên là MyFirst và chọn Run.*

Mọi việc sẽ tốt đẹp, đoạn chữ đậm **I can write macros!** sẽ được nhập vào ô B2.

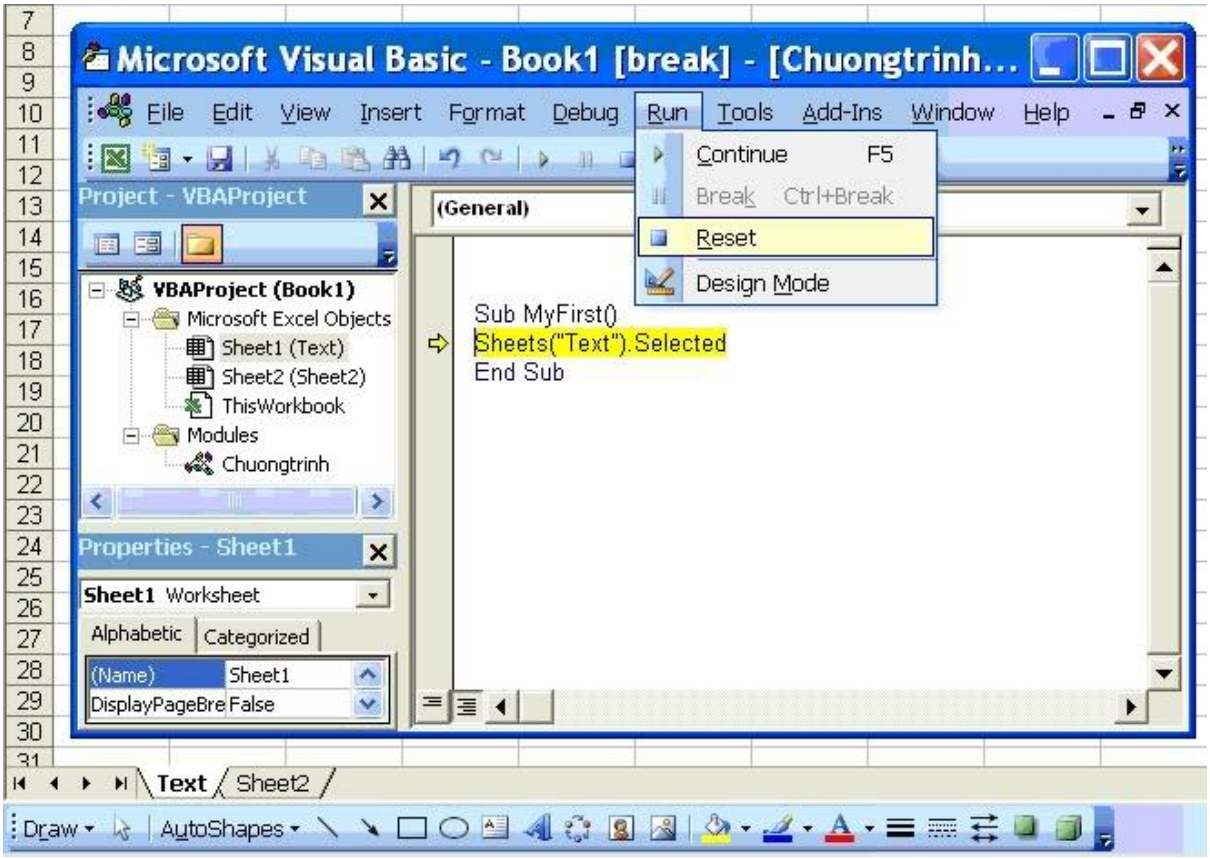
Khi code bị lỗi thì sẽ có bảng thông báo lỗi, ví dụ như hình 37.




*Hình 36: Tạo Module và Sub trong workbook*




*Hình 37: Lỗi gặp phải trong việc xây dựng macro*




*Hình 38: Sửa lỗi gặp phải khi viết code*

Khi gặp lỗi, bạn tiến hành theo các bước sau đây:

*1. Bấm vào nút Debug và tìm kiếm lỗi để sửa lại. Lỗi của câu lệnh đầu tiên sẽ được bôi nền màu vàng (hình 38).*

*2. Sửa những câu lệnh sai trong phần được bôi vàng đó.*

*3. Mũi tên vàng ở lề sẽ cho biết rằng macro đang ở chế độ dừng (break mode).*

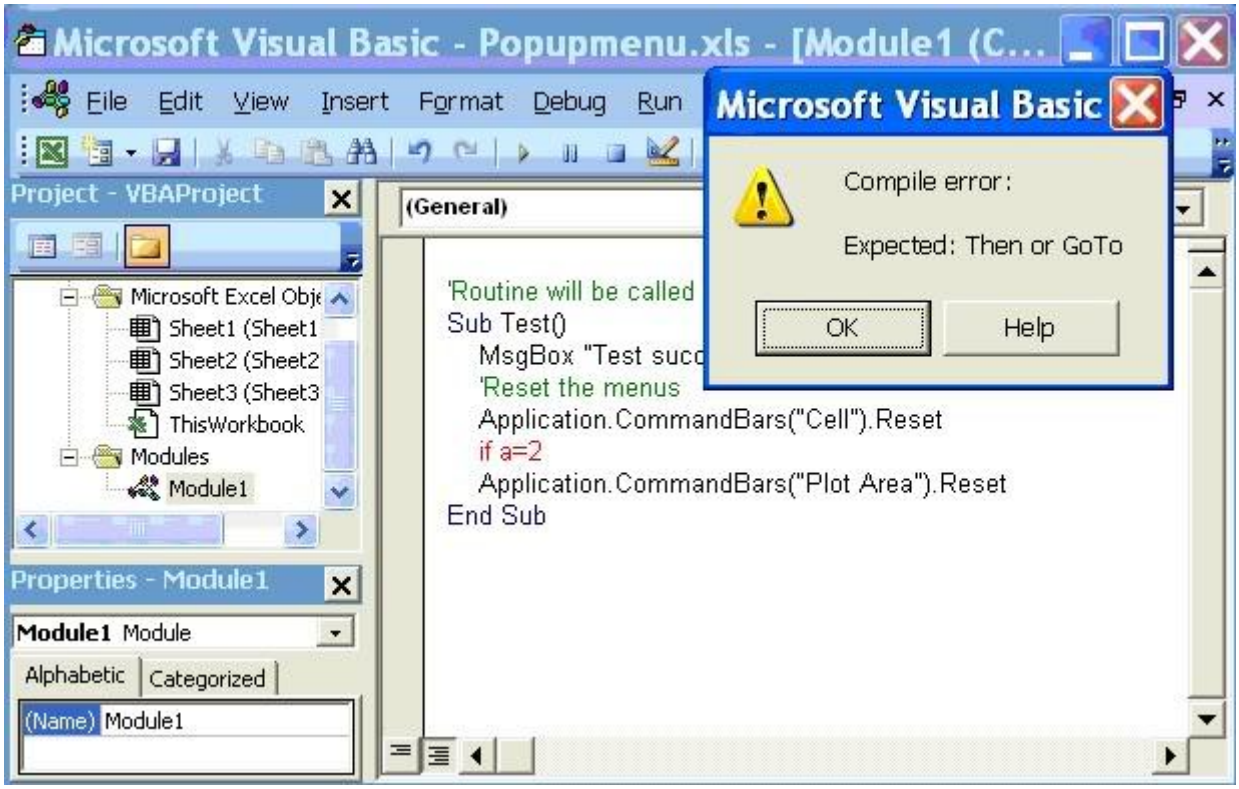
*4. Ngoài ra bạn có thể bấm vào Run, sau đó chọn Reset để xác lập lại (hình 38) hoặc chọn Design Mode để xác lập chế độ thiết kế. Còn nếu muốn macro chạy tiếp thì chọn Continue (hoặc ấn phím F5).*

*5. Quay trở về sheet Text và xem macro làm việc có chính xác không.*

## *7.2. Sửa chữa lỗi*

Khi bạn gõ một dòng code trong macro và gõ Enter, Excel sẽ kiểm tra dòng đó. Nếu nó tìm được số hạng mà hiểu được, ví dụ như range, thì sẽ trở thành Range (chữ r tự động chuyển thành chữ hoa R ở đầu).

Nếu code đó thiếu hoặc tìm ra lỗi, Excel sẽ biến nội dung đó thành màu đỏ và hiện ra bảng thông báo lỗi (hình 39). Có nhiều loại lỗi khác nhau, tuỳ vào lỗi cụ thể mà có từng kiểu nội dung bảng thông báo.




*Hình 39: Báo lỗi code*

Nếu bạn muốn biết thêm thông tin về lỗi đó thì bấm vào nút Help. Để sửa chữa lỗi đó, bạn bấm OK và sửa nội dung dòng có màu đỏ cho đúng.

Có những trường hợp gặp phải những lỗi mà không được thông báo cho đến khi Visual Basic biên dịch nó trước khi chạy. Trường hợp mà bạn gặp đó là lỗi compile-time. Visual Basic sẽ cho biết vị trí của lỗi đó và sẽ gửi cho bạn thông báo về lỗi đó.

Còn các lỗi khác chỉ xuất hiện khi macro chạy thật sự. Đó được gọi là lỗi run-time. Để sửa chữa lỗi này thì bạn bấm vào Goto rồi sửa đoạn code đó.

Một số "lỗi" gặp phải không hẳn là lỗi, nó chỉ xuất hiện khi macro chạy. Ví dụ như chia một số cho không (zero) có thể xảy ra ngoài ý muốn. Dựa vào hoàn cảnh đó mà bạn có thể sử dụng câu lệnh On Error để "bẫy lỗi" (xem ở mục 13).

## 8. Tham chiếu đến ô và vùng

Bạn có thể sử dụng macro để tham chiếu đến các ô hoặc vùng trong worksheet. Nếu bạn muốn gán dữ liệu vào worksheet, bạn sẽ phải sử dụng đến đối tượng Range. Đối tượng Range được sử dụng vào loại nhiều nhất trong Excel để tham chiếu đến ô riêng lẻ (a cell) hoặc vùng (range). Có vài cách cho giá trị đối tượng Range đã được mô tả phía dưới đây.

### *8.1. Tham chiếu kiểu A1*

Dưới đây là bảng ví dụ các dạng tham chiếu đến ô, vùng của ô theo kiểu A1 khi sử dụng phương thức Range.(ĐỀ CẬP ĐẾN THAY BẰNG THAM CHIẾU ĐẾN)

| Tham chiếu | Đề cập đến |
|---|---|
| Range("B1") | ô B1 |
| Range("B1:B6") | vùng từ B1 đến B6 |
| Range("B2:B7, F4: K30") | 2 vùng B2 đến B7 và F4 đến K30 |
| Range("C:C") | Cột C |
| Range("7:7") | Hàng 7 |
| Range("D:G") | Cột D đến cột G |
| Range("2:6") | Hàng 2 đến cột 6 |
| Range("2:2, 5:5, 8:8") | Hàng 2, cột 5, cột 8 |
| Range("B:B, D:D, G:G") | Cột B, D và G |

Ví dụ 1:

Range("A1:A3").Select

thì vùng A1:A3 sẽ được chọn (bôi đen)

Ví dụ 2:

Với workbook có tên Popupmenu, trong worksheet Sheet1, bạn gán nội dung Bo mon DCCT vào ô B3. Sau đó cho nội dung chữ đó đậm, nghiêng, màu đỏ và nền màu vàng.

Code:

```
Sub Thunghiem()
  Workbook("Popupmenu").Sheets("Sheet1").Range("B3").Select
   ActiveCell.FormulaR1C1 = "Bo mon DCCT"
     Selection.Font.Bold = True
     Selection.Font.Italic = True
     Selection.Font.ColorIndex = 3
      With Selection.Interior
```

```
            .ColorIndex = 6
            .Pattern = xlSolid
        End With
    Range("B4").Select
End Sub
```

*Hướng dẫn chọn vùng tắt:*

Bạn có thể dùng ngoặc vuông [ ] để chọn vùng ô thay vì (). So sánh với ví dụ như sau:

[A1:A3].Select là cách chọn vùng giống như Range("A1:A3").Select

thay đổi nội dung bởi: **PhanTuHuong**, 05-12-06 lúc 12:49 PM

### *8.2. Số chỉ mục (Index numbers)*

Thuộc tính Cells có thể sử dụng để trả về đối tượng mảng là ô đơn. Số chỉ mục hàng và cột của ô cung cấp cho Cells(row_no,col_no). Nếu mà không có số hàng và cột thì Cells() sẽ trả về đối tượng là toàn bộ ô trong sheet (giống như phím tắt Ctr + A).

Ví dụ 1:

Cells(4,1) trả về ô A4

Cells() trả về toàn bộ ô trong sheet

Ví dụ 2:

Worksheets("Sheet2").Cells(3,2).Value = 2000

trả về số 2000 trong ô B3 tại Sheet2, trong workbook hiện hành.

Ghi chú: Thuộc tính Cells được ứng dụng nhiều khi viết các vòng lặp giữa các ô.

### *8.3. Số hàng và số cột (Rows and Columns)*

Đây là một cặp thuộc tính được gọi là Rows và Columns, chúng giúp bạn có thể làm việc với toàn bộ dòng hoặc cột.

Code:

```
Tham chiếu      Đề cập đến
_____________________________________________________
Rows(4)          Hàng số 4
Rows             Toàn bộ dòng trong sheet hiện hành
Columns(4)       Cột D (cột thứ 4)
Columns("D")     Cột D
Columns          Toàn bộ cột trong sheet hiện hành
```

Ví dụ:

Worksheets(“Week4”).Rows(2).Font.Bold = True

cho kết quả là toàn bộ hàng 2 trong sheet Week4 chữ đậm của workbook hiện hành.

Ghi chú: Bạn có thể thực hiện đối với nhiều hàng và cột khi sử dụng phương thức Union.

Ví dụ về sự hợp nhất giữa hai vùng Range1 và Range2 khi sử dụng phương thức Union được điền đầy bởi công thức =RAND()

```
Worksheets("Sheet1").Activate
Set Vung = Application.Union(Range("Range1"), Range("Range2"))
Vung.Formula = "=RAND()"
```

## *8.4. Đặt tên cho vùng (Named ranges)*

Với một số trường hợp bạn phân chia vùng các ô ra với tên xác định để dễ truy cập và nghiên cứu. Công việc này gần giống như khi bạn sử dụng chức năng đặt tên cho vùng ô trong Excel (xem trong menu Insert/Name/Define...). Khi bạn chọn tên những vùng đó thì Excel sẽ truy cập đến vùng mà bạn lựa chọn. Bạn phải đặt tên những vùng đó trước khi viết macro hay dùng chính macro để tạo tên của vùng.

### *8.4.1. Tên được tạo ra ngoài macro*

Để đặt tên cho vùng, đầu tiên bạn chọn chọn những ô đó bằng cách bôi đen, sau đó bạn bấm vào phần Name Box (phần góc trên bên trái, cùng hàng với thanh công thức). Sau đó đặt tên của vùng đó rồi Enter.

Giả thiết rằng bạn đặt tên Congty cho các ô C2:C8 trong sheet Danhsach của workbook Quanly (hình 40).

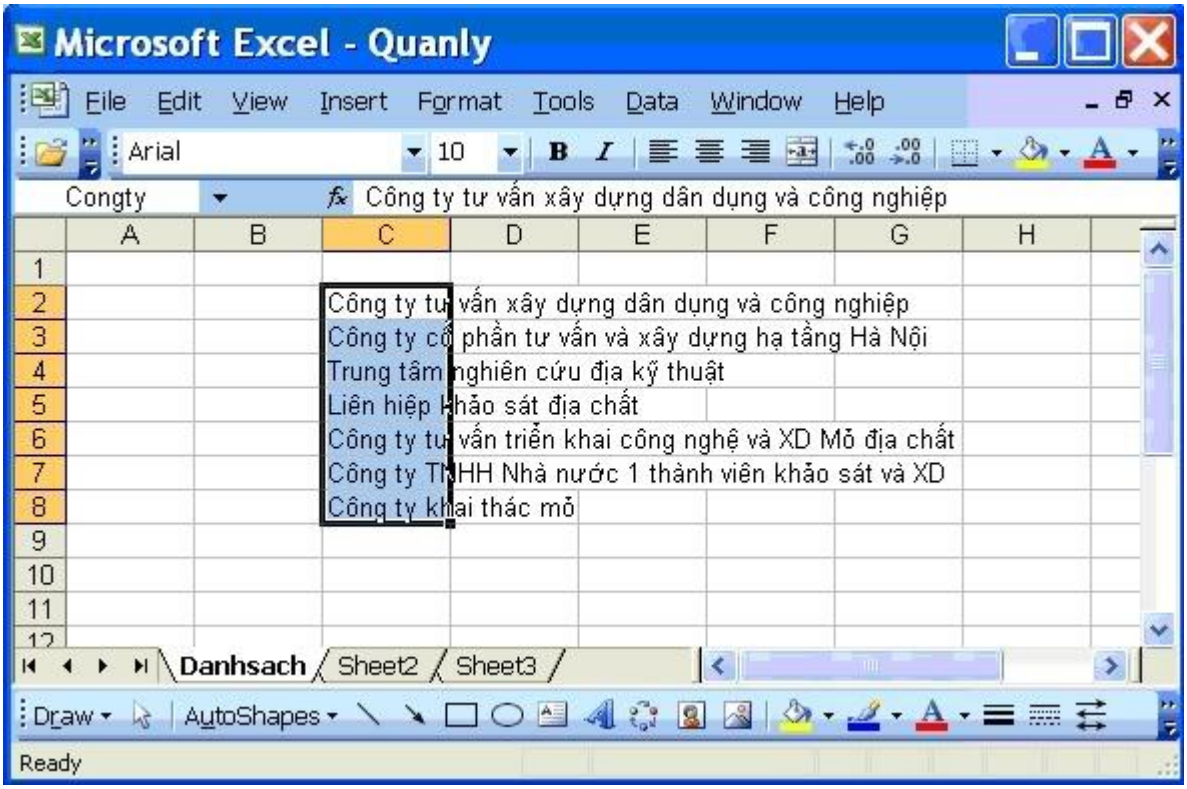
Ví dụ 1: sẽ làm các ô trong vùng C2:C8 đậm lên.

```
Range(“[Quanly.xls]Danhsach!Congty”).Font.Bold = True
```

Ví dụ 2: Nếu workbook Quanly và worksheet Danhsach đang hiện hành, thì

```
Range(“Congty”).Font.Bold = False
```

sẽ làm các ô trong vùng C2:C8 mất đậm (chữ bình thường).




*Hình 40: Tạo tên của vùng*

### *8.4.2. Tên được tạo ra trong macro*

Tên vùng có thể được ấn định khi sử dụng macro để lập, ví dụ dưới đây:

*Workbooks("Congty.xls").Names.Add Name:="Congty", _*

*RefersTo:="=Danhsach!D1:D10"*

*Range("Congty").Font.Italic = True*

Kết quả là các ô trong vùng D1:D10 sẽ bị nghiêng.

## *8.5. Nhiều vùng (Multiple ranges)*

Trường hợp này hay được sử dụng để tham chiếu đến nhiều vùng trong macro, có thể xoá sạch nội dung trong các ô đó.

*Worksheets("Bang").Range("A1:C3,H4:L8,P14:Z3 4").ClearContents*

sẽ xoá sạch nội dung những ô đã chỉ định trong worksheet Bang.

Còn đối với những tên vùng bạn đặt (như ở trên), có thể thực hiện như sau:

*Range("Danhsach1, Danhsach2, Danhsach3").ClearContents*

Ghi chú: Trong macro, các vùng có thể được xác định, đặt tên và được phối hợp khi sử dụng phương thức Union. Xem mục 9.3.

## *8.6. Offset cells*

Thuộc tính Offset thường được sử dụng để tham chiếu đến ô khác mà có quan hệ với ô đang hoạt động.

**Công thức dạng tổng quát:**

**Offset(no_rows_down, no_cols_to_right)**

- no_rows_down

là số nguyên và được hiểu là xuống dưới bao nhiêu dòng.

- no_cols_to_right

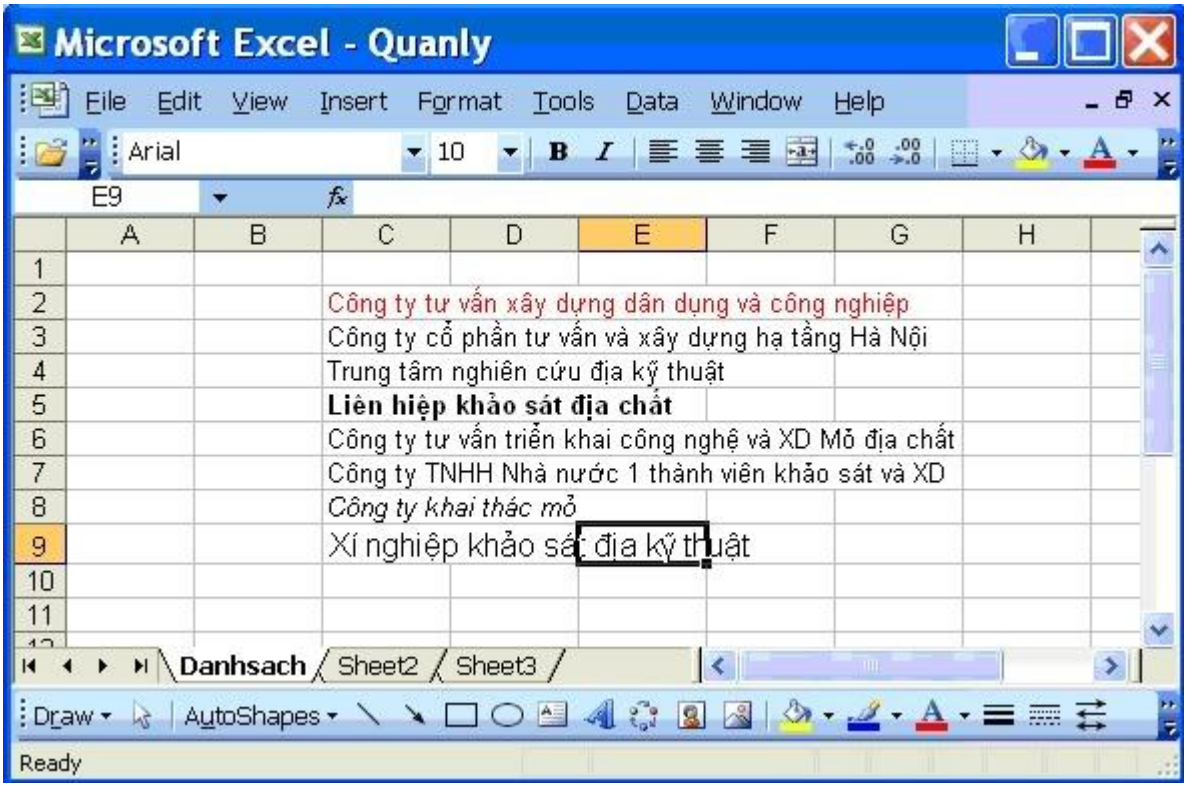
là số nguyên và được hiểu chuyển sang phải bao nhiêu cột.

Ví dụ 1:

Như ở hình 41, giả thiết ô B1 là ô hiện hành. Bây giờ bạn dùng Offset để chữ trong ô C2 có màu đỏ, C5 đậm, C8 nghiêng, C9 có nội dung "Xí nghiệp khảo sát địa kỹ thuật".

```
Sub Offset()
Range("B1").Activate
ActiveCell.Offset(1, 1).Font.ColorIndex = 3
ActiveCell.Offset(4, 1).Font.Bold = True
ActiveCell.Offset(8, 1).Value = "Xí nghiệp khảo sát địa kỹ thuật"
ActiveCell.Offset(8, 1).Font.Size = 12
Range("E9").Activate
ActiveCell.Offset(-1, -2).Font.Italic = True
End Sub
```

Kết quả thể hiện ở hình 41.




*Hình 41: Sử dụng Offset để tham chiếu đến các ô*

Ghi chú: Khi giá trị no_rows_down hoặc no_cols_to_right có giá trị âm thì sẽ có hướng ngược lại.

### *8.7. Kiểu tham chiếu R1C1*

Khi sử dụng kiểu R1C1, Excel sẽ tham chiếu đến ô mà được xác định bởi số hàng và cột. Ví dụ ô tham chiếu R4C2 sẽ truy cập đến ô B4.

Khi sử dụng kiểu tham chiếu này, mối quan hệ giữa các ô trong tính toán sẽ được thể hiện trong công thức.

R[m]C[n] sẽ tham chiếu đến (truy cập đến) ô có m dòng phía dưới và n cột phía bên phải so với ô hoạt động (hiện hành). Giá trị m,n có thể là số âm, khi đó hướng sẽ ngược lại, lên trên và sang bên trái.

Ví dụ 1: Nhập vào công thức Sum("B2:B4") trong ô B5.

Ô B5 có địa chỉ hàng 5 và cột B và có giá trị là tổng các ô của 3 hàng trước đó, gồm dòng thứ 2 đến dòng thứ tư nhưng cùng cột.

Từ đó xây dựng macro như sau:

```
Range("B5").Select
ActiveCell.FormulaR1C1 = "=Sum(R[-3]C:R[-1]C)"
```

Ví dụ 2: Nhập vào công thức = F2-F4 trong ô D5.

Nội dung R1C1 trong FormulaR1C1 có thể không cần dùng đến và nếu bạn muốn, 2 dòng trên có thể ghép thành 1 như sau:

*Range("B5").Formula = "=R[-3]C[2]-R[-1]C[2]"*

Ví dụ 3:

Thay đổi công thức thành giá trị kết quả.

Ô G6 có công thức là =G5*G4. Ví dụ ô G5 có giá trị là 2, ô G4 có giá trị là 3, như vậy giá trị nhận được của ô G6 là 6. Ta sẽ thay nội dung hàm thành giá trị là 6.

*Range("G6").Select*

*ActiveCell.FormulaR1C1 = "=R[-1]C:R[-2]C"*

*Selection.Copy*

*Selection.PasteSpecial Paste:=xlValues*

*Application.CutCopyMode = False*

Dòng lệnh cuối cùng là lệnh huỷ bỏ chế độ trạng thái Cut/Copy (đường gạch nhấp nháy bao quanh ô đã chọn).

## 9. Cấu trúc điều khiển

Một số trường hợp, bạn phải sử dụng macro để kiểm tra những điều kiện đặc biệt trong worksheet và điều khiển chúng để đáp ứng được yêu cầu đề ra. Với mỗi điều kiện khác nhau thì macro sẽ thực hiện công việc khác nhau.

Với cấu trúc điều khiển như vậy không thể tự động ghi được, bạn phải viết chúng trong Visual Basic.

### *9.1. Câu lệnh IF*

Đây là kiểu đơn giản nhất, mẫu của câu lệnh IF như sau:

**If <điều kiện> Then <dòng lệnh 1> [Else <dòng lệnh 2>]**

Trong chỉ dẫn trên, các thông số trong [ ] là tuỳ chọn, có thể bỏ qua nếu thấy không cần thiết.

Nếu <điều kiện> được toại nguyện (đúng - True) thì <dòng lệnh 1> được thực hiện, còn nếu không được toại nguyện (sai - False) thì <dòng lệnh 2> được thực hiện.

Thông thường, bạn hay sử dụng câu lệnh If ... then ... Else mà không cần phải giới hạn số dòng lệnh. Mẫu như sau:

*If <điều kiện1> Then*

*<Khối lệnh 1 thực hiện>*

*[ElseIf <điều kiện2>*

*<Khối lệnh 2 thực hiện>]*

*[ElseIf <điều kiện3>*

*<Khối lệnh 3 thực hiện>]*

*[Else*

*<Khối lệnh 4 thực hiện>]*

*End If*

Trong mẫu tổng quát ở trên, từ khoá ElseIf và Else là tuỳ chọn (như biểu thị trong dấu ngoặc vuông). Đầu tiên VB kiểm tra điều kiện thứ nhất, nếu sai thì sẽ chuyển sang điều kiện thứ 2,... cho đến khi điều kiện đúng. VB thi hành khối lệnh tương ứng và sau đó, thi hành dòng chương trình ngay sau End If.

Ví dụ:

Macro dưới đây tìm kiếm giá trị tại ô A1 (là điểm trung bình môn học).

*Nếu 10 > A1 ≥ 8.0 : "Học lực giỏi";*

*Nếu 8 > A1 ≥ 6.5 : "Học lực khá";*

*Nếu 6.5 > A1 ≥ 5.0 : "Học lực trung bình";*

*Nếu 5 > A1 ≥0 : "Học lực kém".*

Ô B2 thể hiện kết quả học lực.

```
Sub Hocluc()
Sheets("Sheet1").Select
Range("A1").Select
If ActiveCell >= 8 Then
Range("B2").Value = "Học lực giỏi"
ElseIf ActiveCell >= 6.5 Then
Range("B2").Value = "Học lực khá"
ElseIf ActiveCell >= 5 Then
Range("B2").Value = "Học lực trung bình"
Else
Range("B2").Value = "Học lực kém"
End If
End Sub
```

Ghi chú: Bạn có thể bỏ qua dòng Range("A1").Select và thay bằng If Range("A1").Value >= 8 Then.

Ngoài ra, bạn cũng có thể sử dụng If để kết thúc macro, câu lệnh như sau có thể sử dụng để kết thúc macro.

If ActiveCell = "" Then End Sub

(nếu ô hiện hành mà trống thì sẽ kết thúc Sub, không cần phải có End If)

Ví dụ:

Giả sử bạn tìm kiếm giá trị của một ô và bạn muốn kết quả như sau

• Dừng macro khi ô đó trống.

• Nhập giá trị "Tốt" vào ngay ô bên phải ô đó nếu có giá trị lớn hơn 40.

• Nhập giá trị "Kém" vào ngay ô bên phải ô đó nếu có giá trị nhỏ hơn 40.

```
Sub user_If()
If ActiveCell.Value = "" Then Exit Sub
If ActiveCell.Value >= 40 Then
ActiveCell.Offset(0, 1).Value = "Tốt"
Else
ActiveCell.Offset(0, 1).Value = "Xấu"
End If
End Sub
```

### *9.2. Sử dụng Select Case*

Select Case là một dạng của If ... Then ... Else, được sử dụng khi có nhiều điều kiện chọn lọc giá trị. Câu lệnh như sau:

**Select Case <biểu thức kiểm tra>**

**[Case <biểu thức 1>**

**<khối lệnh 1>]**

**[Case <biểu thức 2>**

**<khối lệnh 2>]**

**[Case <biểu thức 3>**

**<khối lệnh 3>]**

**....**

**[Case Else <biểu thức n>**

**<khối lệnh n>]**

**End Select**

Mỗi danh sách biểu thức có 1 hay nhiều giá trị. Các giá trị cách nhau bằng dấu phẩy (,). Còn giá trị biến đổi trong vùng thì bạn sử dụng từ khoá To. Mỗi khối lệnh có thể chứa 0 hay nhiều dòng lệnh. Nếu biểu thức nào thoả mãn điều kiện thì khối lệnh tương ứng sẽ thực hiện. Case Else không nhất thiết phải có, dùng trong trường hợp còn lại của các Case trước.

Ví dụ:

Ô B2 chứa giá trị độ sệt của đất, ô C2 sẽ thể hiện trạng thái của nó.

```
Sub Trangthai()
Sheets("Sheet1").Select
Doset = Cells(2,2).Value
Select Case Doset
Case 1, 1 to 10
Cells(2,3).Value= "Chảy"
Case 0.75 to 1
Cells(2,3).Value= "Dẻo chảy"
Case 0.5 to 0.75
Cells(2,3).Value= "Dẻo mềm"
Case 0.25 to 0.5
Cells(2,3).Value= "Dẻo cứng"
Case 0 to 0.25
Cells(2,3).Value= "Nửa cứng"
Case < 0
Cells(2,3).Value= "Cứng"
End Select
End Sub
```

## 9.3. Xây dựng các điều kiện

Trong nhiều trường hợp, điều kiện lọc dữ liệu đã trở nên khá phức tạp. Nếu chỉ sử dụng If hay Select Case thì công việc sẽ rất cồng kềnh, rắc rối. Trong hoàn cảnh đó, And và Or giúp bạn thực hiện công việc đó, giúp chương trình sáng sủa và dễ đọc.

### 9.3.1. Sử dụng And

Câu lệnh như sau:

```
If <điều kiện 1> And <điều kiện 2> Then
<khối lệnh 1>
Else
<khối lệnh 2>
End If
```

`

<khối lệnh 1> chỉ thực hiện khi cả hai điều kiện 1 và 2 đều đúng. Chỉ 1 trong 2 điều kiện sai thì <khối lệnh 2> sẽ thực hiện.

### *9.3.2. Sử dụng Or*

Câu lệnh như sau:

**If <điều kiện 1> Or <điều kiện 2> Then**

**<khối lệnh 1>**

**Else**

**<khối lệnh 2>**

**End If**

<khối lệnh 1> thực hiện khi một trong hai điều kiện 1 và 2 đúng. Cả 2 điều kiện sai thì <khối lệnh 2> sẽ thực hiện.

### *9.3.3. Sử dụng nhiều And và Or*

Câu lệnh như dưới đây:

**If <điều kiện 1> And <điều kiện 2> And <điều kiện 3> Then**

**<khối lệnh 1>**

**Else**

**<khối lệnh 2>**

**End If**

<khối lệnh 1> chỉ thực hiện khi cả ba điều kiện đều đúng. Chỉ 1 trong 3 điều kiện sai thì <khối lệnh 2> sẽ thực hiện.

Tương tự đối với Or.

Ví dụ:

Bạn có thể xác định tên đất dựa vào hệ số rỗng tự nhiên, chỉ số dẻo, độ sệt.

```
Sub Ten_dat()
Dim Hsr, Chisodeo, Doset As Single
Hsr = InputBox("Vao gia tri he so rong:")
Chisodeo = InputBox("Vao gia tri chi so deo:")
Doset = InputBox("Vao gia tri do set:")
If Hsr > 1.5 And Chisodeo >= 17 And Doset > 1 Then
MsgBox "Day la dat BUN SET!"
ElseIf Hsr > 1.0 And Chisodeo >= 7 And Doset > 1 Then
MsgBox "Day la dat BUN SET PHA!"
```

```
ElseIf Hsr > 0.9 And Chisodeo >= 1 And Doset > 1 Then
MsgBox "Day la dat BUN CAT PHA!"
Else
MsgBox "Chua ro ten dat!!!!"
End If
End Sub
```

## 10. Hộp thoại trong VBA

Hộp thoại (Dialog) là một trong những cách thức để Windows giao tiếp với người sử dụng. Dưới đây là 2 loại hộp thoại mà bạn dễ dàng tạo ra để điều khiển trong suốt quá trình chạy macro (MsgBox và InputBox).

### *10.1. Hộp thông báo (Message box)*

Câu lệnh MsgBox sẽ cho hiện lên trên màn hình một hộp thông báo, giá trị nhận được là biến số (variable) trong macro (như hình 18). Sử dụng MsgBox giúp bạn rất hiệu quả trong việc gỡ rối (hoặc tìm chỗ sai, giá trị trung gian,...) khi xây dựng chương trình.

Hàm MsgBox ở dạng tổng quát

**MsgBox (prompt [, buttons] [, title] [, helpfile, context])**

Trên màn hình sẽ hiện hộp thông báo và đợi bạn bấm chuột vào nút chọn và trở về giá trị nguyên nào khi bạn chọn loại nút.

- prompt là nội dung lời nhắc của hộp thông báo.
- buttons là tuỳ chọn loại nút điều khiển (như Yes, No, OK)
- title là tuỳ chọn nội dung chữ trên đầu hộp thông báo
- helpfile là tuỳ chọn và điều khiển file trợ giúp nào để sử dụng.

context là tuỳ chọn và là số thứ tự tình huống trong helpfile. Nếu helpfile có thì mục context cũng phải có.

#### *10.1.1. Các loại thông điệp trong buttons*

| Hằng số | Giá trị | Mô tả |
|---|---|---|
| vbOKOnly | 0 | Hiện nút OK |
| vbOKCancel | 1 | Hiện nút OK và Cancel |
| vbAbortRetryIgnore | 2 | Hiện nút Abort, Retry và Ignore. |
| vbYesNoCancel | 3 | Hiện nút Yes, No và Cancel. |
| vbYesNo | 4 | Hiện nút Yes và No. |
| vbRetryCancel | 5 | Hiện nút Retry và Cancel. |

### 10.1.2. Mô tả thông số các nút

| Hằng số | Giá trị | Mô tả |
|---|---|---|
| vbOK | 1 | Chọn nút OK |
| vbCancel | 2 | Chọn nút Cancel |
| vbAbort | 3 | Chọn nút Abort |
| vbRetry | 4 | Chọn nút Retry |
| vbIgnore | 5 | Chọn nút Ignore |
| vbYes | 6 | Chọn nút Yes |
| vbNo | 7 | Chọn nút No |

### 10.1.3. Các biểu tượng thông điệp

| Hằng số | Thể hiện | Giải thích |
|---|---|---|
| vbCritical |  | Dùng cho những thông báo lỗi thất bại khi thực hiện công việc nào đó. |
| vbQuestion |  | Dùng cho những câu hỏi yêu cầu người sử dụng chọn lựa. |
| vbExclamation |  | Dùng cho các thông báo của chương trình. |
| vbInformation |  | Dùng cho các thông báo cung cấp thêm thông tin cho người sử dụng. |
| vbDefaultButton1 | 0 | Mặc định nút lệnh thứ nhất |
| vbDefaultButton2 | 256 | Mặc định nút lệnh thứ hai |
| vbDefaultButton3 | 512 | Mặc định nút lệnh thứ ba |

Ghi chú: Tại mỗi kiểu thông điệp, âm thanh báo khi hiển thị thông điệp đi kèm theo sẽ khác nhau.

### 10.1.4. Xây dựng tham số cho MsgBox

Để sử dụng tuỳ biến hộp thông báo, bạn phải biết phối hợp các thông số và nút lệnh. Việc sử dụng hộp MsgBox có ý nghĩa rất quan trọng trong việc điều khiển chương trình. Để hiểu chi tiết, các bạn xem ví dụ dưới đây.

```
Sub Nhangui()
Dim Truonghop As Integer
Truonghop = MsgBox("Ban co muon thoat khoi chuong trinh khong", vbYesNoCancel + vbQuestion + vbDefaultButton1,"Chuong trinh tinh lun")
```

*If Truonghop = vbYes Then*

*MsgBox "Ban vua chon nut Yes.", vbInformation*

*ElseIf Truonghop = vbNo Then*

*MsgBox "Ban vua chon nut No.", vbCritical*

*ElseIf Truonghop = vbCancel Then*

*MsgBox "Ban vua bam nut Cancel.", vbExclamation*

*End If*

*End Sub*

Hình vẽ dưới thể hiện kết quả chạy Sub trên và hộp thông báo khi bạn chọn nút No. Trong Sub trên, bạn có thể thay

ElseIf Truonghop = vbNo Then

bằng

ElseIf Truonghop = 7 Then


*Hình 42: Ví dụ về cách tạo MsgBox trong VB và khi chọn nút No*

## *10.2. Phương thức InputBox (Inputbox Method)*

Nhằm thể hiện hộp thoại để người sử dụng nhập dữ liệu vào.

Khi sử dụng phương thức này, một hộp thoại sẽ cho hiện ra để bạn vào dữ liệu, chờ cho người dùng nhập dữ liệu vào hoặc là bấm vào nút OK hoặc Cancel, giá trị nhận được được coi là chuỗi (string). Đây là một cách để vào giá trị đơn lẻ hoặc địa chỉ của các ô trong quá trình chạy macro. Bạn không thể gán được

lệnh khi chọn nút OK hay Cancel như trong MsgBox. Đó chính là hạn chế của hàm này nên ít được ứng dụng khi đầu vào nhiều số liệu.

Phương thức InputBox ở dạng tổng quát

*expression*.**InputBox** (prompt [, title] [, default], [, left], [, top] [helpfile, context] [, type])

Expression: một biểu thức trả về đối tượng Application.

Trong đó:

- prompt là nội dung lời nhắc của hộp vào dữ liệu.
- title là tuỳ chọn nội dung chữ trên đầu hộp vào dữ liệu.
- left là tuỳ chọn khoảng cách từ góc bên trái hộp thoại đến góc bên trái màn hình (mặc định là hộp thoại nằm giữa màn hình). Đơn vị tính là là điểm (point), một điểm bằng 1/72 inch hay khoảng 1/28 cm. Chức năng này ít sử dụng.
- top là tuỳ chọn khoảng cách từ đỉnh hộp thoại đến đỉnh màn hình (mặc định là hộp thoại nằm giữa màn hình). Đơn vị tính là là điểm.
- helpfile là tuỳ chọn và điều khiển file trợ giúp nào để sử dụng.
- context là tuỳ chọn và là số thứ tự tình huống trong helpfile. Nếu helpfile có thì mục context cũng phải có.
- type là tuỳ chọn biến số đầu vào. Trong trường hợp bỏ qua, giá trị đầu vào coi như là chuỗi.

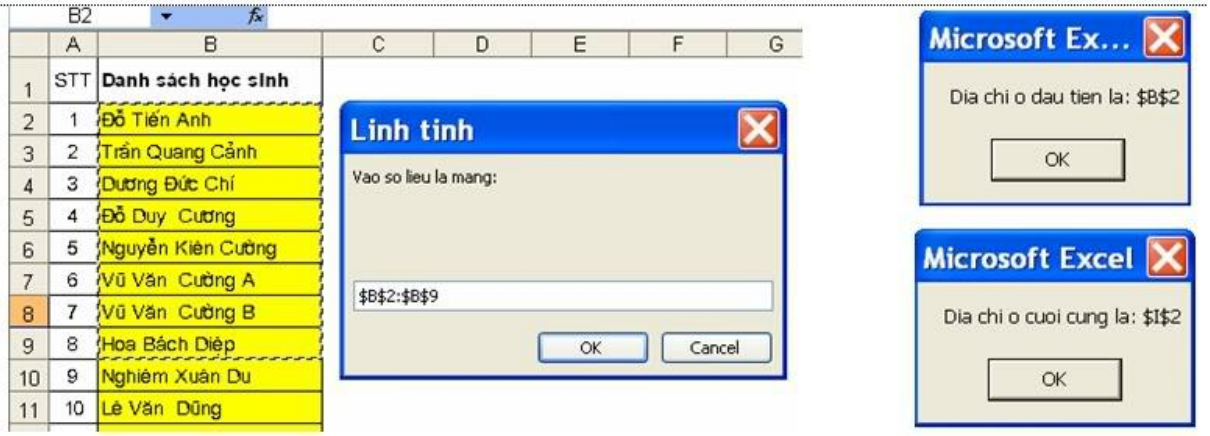
| Giá trị | Nghĩa là |
|---|---|
| 0 | Là công thức |
| 1 | Là số |
| 2 | Là chuỗi (a string) |
| 4 | Là giá trị logic (**True** hay **False**) |
| 8 | Là ô tham chiếu, ví dụ như đối tượng Range |
| 16 | Là giá trị lỗi, ví dụ #N/A |
| 64 | Là mảng của các giá trị (array) |

Ví dụ:

```
Sub VD_Input()
Dim Dangmang
Dim Cot, Hang As Integer
```

```
Set Mang = Application.InputBox("Vao mang:", "Linh tinh", Type:=8)
Cot = Dangmang.Columns.Count ' Tính số cột chọn
Hàng = Dangmang.Rows.Count ' Tính số hàng chọn
MsgBox "So cot la: " & Cot
MsgBox "So hang la: " & Hang
MsgBox "Dia chi o dau la: " & Dangmang.Cells(1, 1).Address
MsgBox "Dia chi o cuoi la: " & Dangmang.Cells(Cot, Hang).Address
' Address là thông tin địa chỉ ô
End Sub
```

Kết quả vào dữ liệu là mảng dưới đây. Ngoài ra bạn còn thu được một số thông tin về mảng đó như số hàng, số cột, địa chỉ ô,...




**Hình 43: Ví dụ về sử dụng InputBox**

## 11. Hành động lặp (Loop)

Hành động lặp cho phép bạn thực hiện một đoạn chương trình nhiều lần. Chức năng này hết sức có ý nghĩa khi bạn xử lý các đối tượng là mảng. Bạn có thể điều khiển hành động lặp theo quy định đặt ra. Có các kiểu hành động lặp như sau:

### *11.1. Do ... Loop*

Thực hiện một khối lệnh với số lần lặp xác định. Trong đó, một biểu thức điều kiện dùng so sánh để quyết định vòng lặp tiếp tục hay không. Điều kiện phải quy về False (0) hoặc True (khác 0). Mẫu tổng quát:

```
Do
<khối lệnh>
Loop
```

Ví dụ:

```
Sub VD_Do()
m = 4 ‘ m nhận giá trị ban đầu là 4
Do ‘ bắt đầu vòng lặp
m = m + 1 ‘ đặt giá trị m tăng (+ 1)
MsgBox m ‘ hộp thông báo giá trị m
If m > 10 Then Exit Do ‘ nếu m > 10 thì sẽ thoát khỏi Do
Loop ‘ Tiếp tục lặp
End Sub
```

### 11.2. Do While ... Loop

Thực hiện khối lệnh khi điều kiện True. Hành động sẽ lặp với điều kiện True, cho đến khi điều kiện False thì sẽ thoát ra. Mẫu tổng quát:

**Do While <điều kiện>**

**<khối lệnh>**

**Loop**

Ví dụ:

```
Sub VD_DoW_Loop()
i = 1 ‘ Đặt i lúc đầu bằng 1
Do While i <= 10 ‘ Đặt giới hạn cho i, nếu False thì thoát
Cells(i,1) = i ‘ Gán i vào ô
i = i + 1 ‘ Cho giá trị i tăng dần
MsgBox i ‘ Hộp thông báo giá trị i
Loop ‘ Tiếp tục lặp
End Sub
```

### 11.3. Do ... Loop While

Tương tự như Do While ... Loop, thực hiện khối lệnh khi điều kiện True. Hành động sẽ lặp với điều kiện True, cho đến khi điều kiện False thì sẽ thoát ra. Mẫu tổng quát:

**Do**

**<khối lệnh>**

**Loop While <điều kiện>**

Ví dụ:

```
Sub VD_Do_LoopW()
i = 1
Do
Cells(i,3) = i
i = i + 1
Msgbox i
Loop While i <= 10
End Sub
```

### 11.4. Do Until ... Loop

Bạn có thể thực hiện các khối lệnh từ đầu vòng lặp cho đến khi điều kiện vẫn True. Đến khi điều kiện False thì sẽ thoát ra. Phương thức này giống như vòng lặp For ... Next. Mẫu tổng quát:

**Do Until <điều kiện>**

**<khối lệnh>**

**Loop**

Ví dụ:

```
Sub VD_DoU_Loop()
i = 1
Do Until i = 10
Cells(i,5) = i
i = i + 1
MsgBox i
Loop
End Sub
```

Tương tự đối với Do ... Loop Until.

### 11.5. For ... Next

Bạn có thể lặp hành động với số lần biết trước. Ta dùng biến đếm tăng dần hoặc giảm dần trong vòng lặp.

**For <biến đếm> = <điểm đầu> To <điểm cuối> [Step <bước nhảy>]**

**<khối lệnh>**

**Next [<biến đếm>]**

Biến đếm, điểm đầu, điểm cuối, bước nhảy là những giá trị số. Bước nhảy có thể là giá trị dương (tăng) hoặc âm (giảm). Nếu Step không được chỉ định ra, mặc định bước nhảy là 1.

Ví dụ 1: Không dùng Step

```
Sub VD_ForNext()
For i = 1 To 5
Cells(10, i) = i
MsgBox i
Next
End Sub
```

Ví dụ 2: Dùng Step

```
Sub VD_ForNext_Step()
For i = 1 To 7 Step 2
Cells(12, i) = i
MsgBox i
Next
End Sub
```

Trong ví dụ này, giá trị i tăng từng bước 1, 3, 5, 7.

### 11.6. For Each ... Next

Tương tự như vòng lặp For ... Next, nhưng nó lặp khối lệnh theo số phần tử của một tập hợp đối tượng hay một mảng, thay vì theo số lần lặp xác định. Vòng lặp này rất tiện lợi khi ta chưa biết chính xác bao nhiêu phần tử trong tập hợp.

**For Each <phần tử> In <nhóm>**

**<khối lệnh>**

**Next <phần tử>**

Để xác định tên và số lượng sheet trong workbook thì bạn dùng thủ tục sau:

```
Sub ShowWorkSheets()
Dim mySheet As Worksheet
Dim i As Integer : i = 1
For Each mySheet In Worksheets
MsgBox mySheet.Name
i = i + 1
```

```
Next mySheet
MsgBox "So sheet trong workbook la " & i
End Sub
```

### 11.7. Lệnh thoát (Exit)

Trong một số trường hợp, bạn có thể thoát khỏi công việc nào đó khi đã thoả mãn yêu cầu công việc. Bạn có thể sử dụng thủ tục Exit như Exit Do (thoát khỏi vòng lặp Do ... Loop), Exit For (thoát khỏi vòng For ... Next), Exit Function (thoát khỏi hàm), Exit Sub (thoát khỏi chương trình), Exit Property (thoát khỏi thuộc tính đang làm việc).

Ví dụ:

```
Sub ExitStatementDemo()
Dim I, MyNum
Do ' Đặt vòng lặp Do Loop
For I = 1 To 1000 ' Lặp 1000 lần
MyNum = Int(Rnd * 1000) ' Tạo số nguyên ngẫu nhiên
Select Case MyNum ' Tính toán với số nguyên trên
Case 7: Exit For ' Nếu là 7, thoát khỏi For...Next
Case 29: Exit Do ' Nếu là 29, thoát khỏi Do...Loop
Case 54: Exit Sub ' Nếu là 54, thoát khỏi vòng Sub
End Select
Next I
Loop
End Sub
```

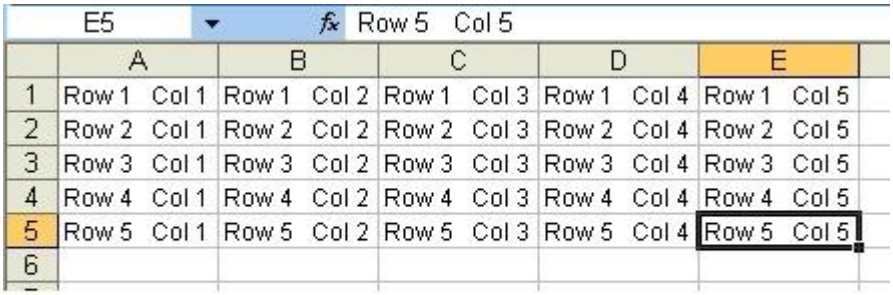
### 11.8. Vòng lặp lồng

Vòng lặp có thể được lồng vào nhau. Ứng dụng này rất có hiệu quả khi bạn tính toán với mảng hay đối với bảng tính nhiều chiều.

Ví dụ:

```
SubCellsExample()
For i = 1 To 5
For j = 1 To 5
Cells(i, j) = "Row " & i & " Col " & j
Next j
Next i
End Sub
```

Kết quả thể hiện ở hình vẽ dưới đây:



| | E5 | fx Row 5 Col 5 | | | |
|---|---|---|---|---|---|
| | A | B | C | D | E |
| 1 | Row 1 Col 1 | Row 1 Col 2 | Row 1 Col 3 | Row 1 Col 4 | Row 1 Col 5 |
| 2 | Row 2 Col 1 | Row 2 Col 2 | Row 2 Col 3 | Row 2 Col 4 | Row 2 Col 5 |
| 3 | Row 3 Col 1 | Row 3 Col 2 | Row 3 Col 3 | Row 3 Col 4 | Row 3 Col 5 |
| 4 | Row 4 Col 1 | Row 4 Col 2 | Row 4 Col 3 | Row 4 Col 4 | Row 4 Col 5 |
| 5 | Row 5 Col 1 | Row 5 Col 2 | Row 5 Col 3 | Row 5 Col 4 | Row 5 Col 5 |
| 6 | | | | | |

*Hình 44: Sản phẩm tạo ra khi dùng vòng lặp lồng.*