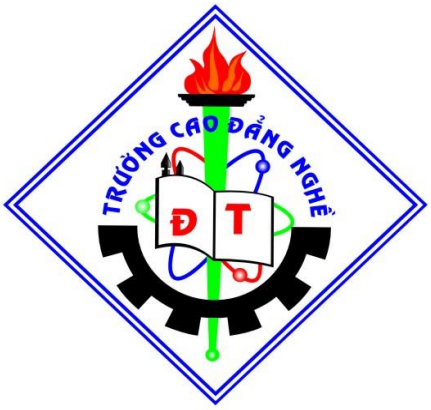
UỶ BAN NHÂN DÂN TỈNH ĐỒNG THÁP
TRƯỜNG CAO ĐẲNG NGHỀ ĐỒNG THÁP



GIÁO TRÌNH

MÔ ĐUN: **QUẢN TRỊ CSDL ACCESS 2**

NGÀNH, NGHỀ: **CÔNG NGHỆ THÔNG TIN (UDPM)**

TRÌNH ĐỘ: **CAO ĐẲNG**

(Ban hành kèm theo Quyết định Số: /QĐ-CĐCĐ-ĐT ngày tháng năm 2017 của Hiệu trưởng Trường Cao đẳng nghề Đồng Tháp)

Đồng Tháp, năm 2017

TUYÊN BỐ BẢN QUYỀN

Bài mở đầu: Tổng quan về ngôn ngữ VBA

-*_*_*-

1. Giới thiệu về VBA

- Ngôn ngữ lập trình VBA (***Visual basic for Application***) một ngôn ngữ lập trình dựa trên nền tảng của Visual Basic, được Microsoft lựa chọn là công cụ tích hợp trong các sản phẩm của mình, ví dụ người sử dụng có thể sử dụng VBA trong Microsoft Word, Microsoft Excel, Microsoft Access, Microsoft Powerpoint, Microsoft Outlook, ngoài ra còn VBA trên Autocad.
- Ngôn ngữ hỗ trợ sử dụng (nhưng không tạo ra), DLL ActiveX (COM), những phiên bản về sau tăng thêm sự hỗ trợ dành cho các module, cho phép sử dụng các công nghệ lập trình hướng đối tượng cơ bản. VBA là một chương trình hữu ích, trong những phiên bản gần đây, nó đã bao gồm những môi trường phát triển tổng hợp (IDE).
- Với VBA, các thao tác thực hiện một cách tự động theo ý muốn người sử dụng.
- Cũng giống như mọi ngôn ngữ lập trình khác, ngôn ngữ VBA cũng có một số quy định về cách khái báo biến, cấu trúc các lệnh, cách khai báo thủ tục, hàm.
- Các cú pháp tương đương Visual Basic 6.0 + Thêm 1 số cú pháp khác.
- Không phân biệt hoa/thường

2. Đặc điểm của VBA

- VBA là một công cụ lập trình cho phép phát triển nhanh phần mềm và được tích hợp vào trong ứng dụng nền. Về thực chất, VBA được xây dựng dựa trên kiến trúc COM1, cho nên người dùng có thể sử dụng các thành phần sẵn có của ứng dụng nền trong việc xây dựng chương trình của mình với VBA.
- Một dự án được xây dựng bằng VBA dựa trên ứng dụng nền nào thì nó phụ thuộc chặt chẽ vào ứng dụng nền đó, bởi theo mặc định, dự án VBA sẽ hoạt động và sử dụng các thành phần trong chính ứng dụng nền đó. Điều này có nghĩa là ta rất khó có thể chuyển đổi một dự án VBA từ loại ứng dụng nền này sang một ứng dụng nền khác cũng như tạo ra một ứng dụng chạy độc lập.
- Sự khác biệt cơ bản nhất của VBA trong các ứng dụng nền (ví dụ giữa VBA trong AutoCAD và VBA trong Excel) là cách thức sử dụng các thành phần (đối tượng) của ứng dụng nền. Cho nên khi xây dựng ứng dụng bằng VBA, việc đầu tiên là phải tìm hiểu mô hình đối tượng của ứng dụng nền và cách sử dụng chúng.
- Xây dựng một dự án VBA, một cách tổng quát, người dùng cần nắm vững hai phần:
    + Ngôn ngữ lập trình Visual Basic và giao diện lập trình VBA IDE
    + Mô hình đối tượng của ứng dụng nền và cách sử dụng chúng

3. Cấu trúc một dự án VBA

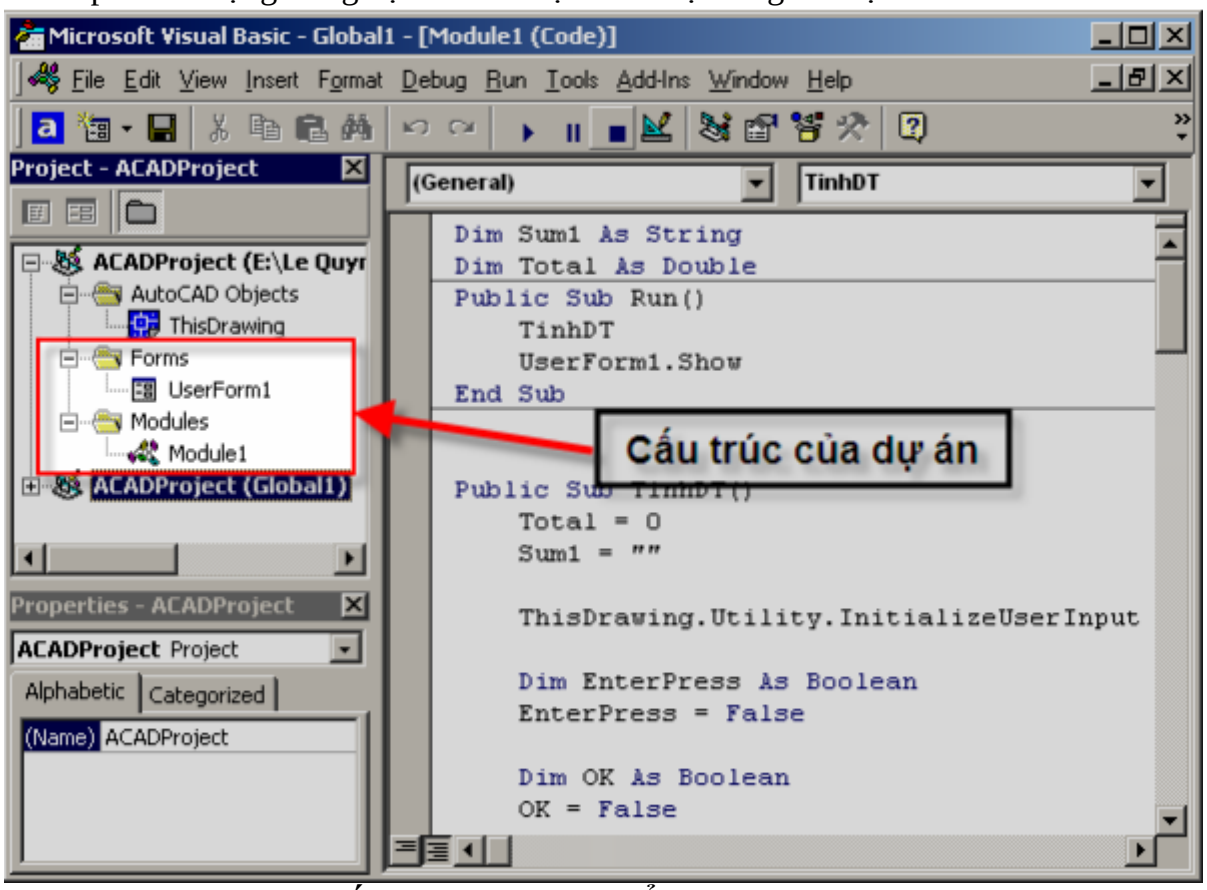
-Khi nói đến các thành phần tạo nên một dự án VBA thì cấu trúc của nó, về tổng quát, như sau:

+ ***Mô-đun chuẩn (Module):*** là nơi chứa các mã lệnh khai báo, các chương trìnhcon (hàm và thủ tục). Việc tạo ra các mô-đun chuẩn thường căn cứ theo các khối chức năng mà người thiết kế hệ thống đặt ra.

+ ***Mô-đun lớp (Class Module):*** là nơi chứa định nghĩa cho các lớp của dự án.

+ ***Userform***: là giao diện dạng hộp thoại giúp cho việc giao tiếp giữa người sử dụng và chương trình được thuận tiện. Thông thường người ta sử dụng Userform để nhập số liệu, xuất kết quả của chương trình. Trong một số dự án, nếu việc nhập số liệu và biểu diễn kết quả được thực hiện trực tiếp trên ứng dụng nền, thì có thể không cần sử dụng Userform.

-Những thành phần này là bộ khung để người dùng xây dựng chương trình của mình lên trên đó, ví dụ như viết mã lệnh hay thiết kế giao diện cho chương trình. Mô-đun lớp và UserForm là hai thành phần có thể xuất hiện hoặc không thùy thuộc vào từng dự án và tất cả những thành phần sử dụng trong dự án đều được hiển thị trên giao diện của VBA IDE.



Cấu trúc của dự án thể hiện trên vba IDE

- Tuy nhiên, khi xây dựng chương trình (viết mã lệnh) cụ thể thì khái niệm cấu trúc của một chương trình là sự bố trí, sắp xếp các câu lệnh trong chương trình đó. Như vậy khái niệm cấu trúc này phụ thuộc vào từng loại ngôn ngữ lập trình. Đối với ngôn ngữ lập trình Visual Basic (VB), cấu trúc của nó chỉ tập trung vào chương trình con (hàm và thủ tục) chứ không có một quy định về cấu trúc nào đối với chương trình chính. Chi tiết của cấu trúc của chương trình con sẽ được đề cập đến trong các phần sau.

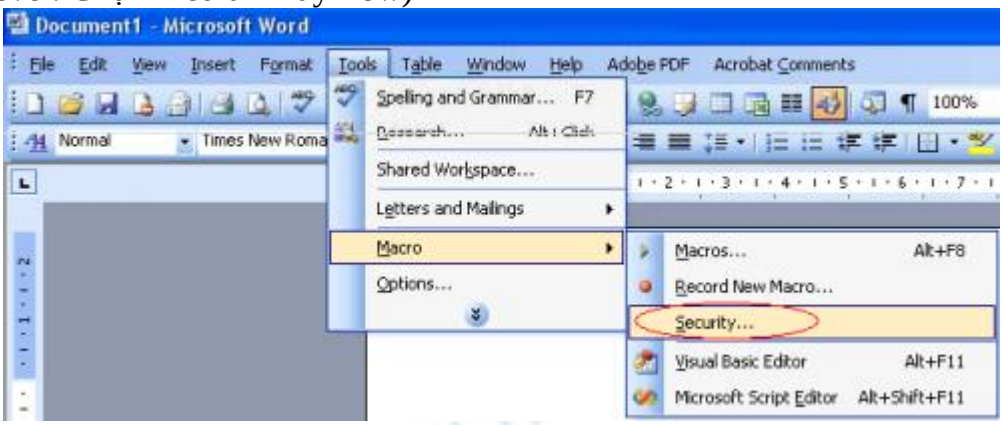
4. Môi trường phát triển tích hợp VBA IDE

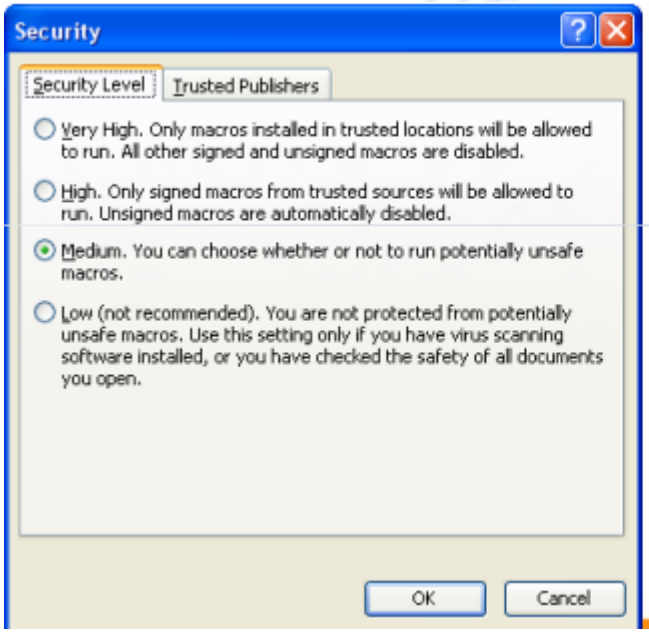
- Trong mỗi công cụ lập trình trên ứng dụng nền, luôn có một môi trường lập trình nhằm hỗ trợ người dùng có thể xây dựng, thử nghiệm và hoàn thiện chương trình của mình. Trong AutoCAD và Microsoft Office, khi sử dụng VBA để lập trình, môi trường lập trình được gọi là Môi trường phát triển tích hợp (viết tắt là VBA IDE). Trên tất cả các ứng dụng nền, VBA IDE có cấu trúc và hoạt động tương đương nhau với giao diện cơ bản và cách gọi giao diện VBA IDE từ ứng dụng nền như sau:

+ Phím tắt: từ giao diện chính của ứng dụng nền, nhấn tổ hợp phím Alt+F11.

+ Menu:

* WORD, EXCEL, PowerPoint, Outlook, Access 2003: Tool/Macro/Security (Security Level: Chọn Medium hay Low)



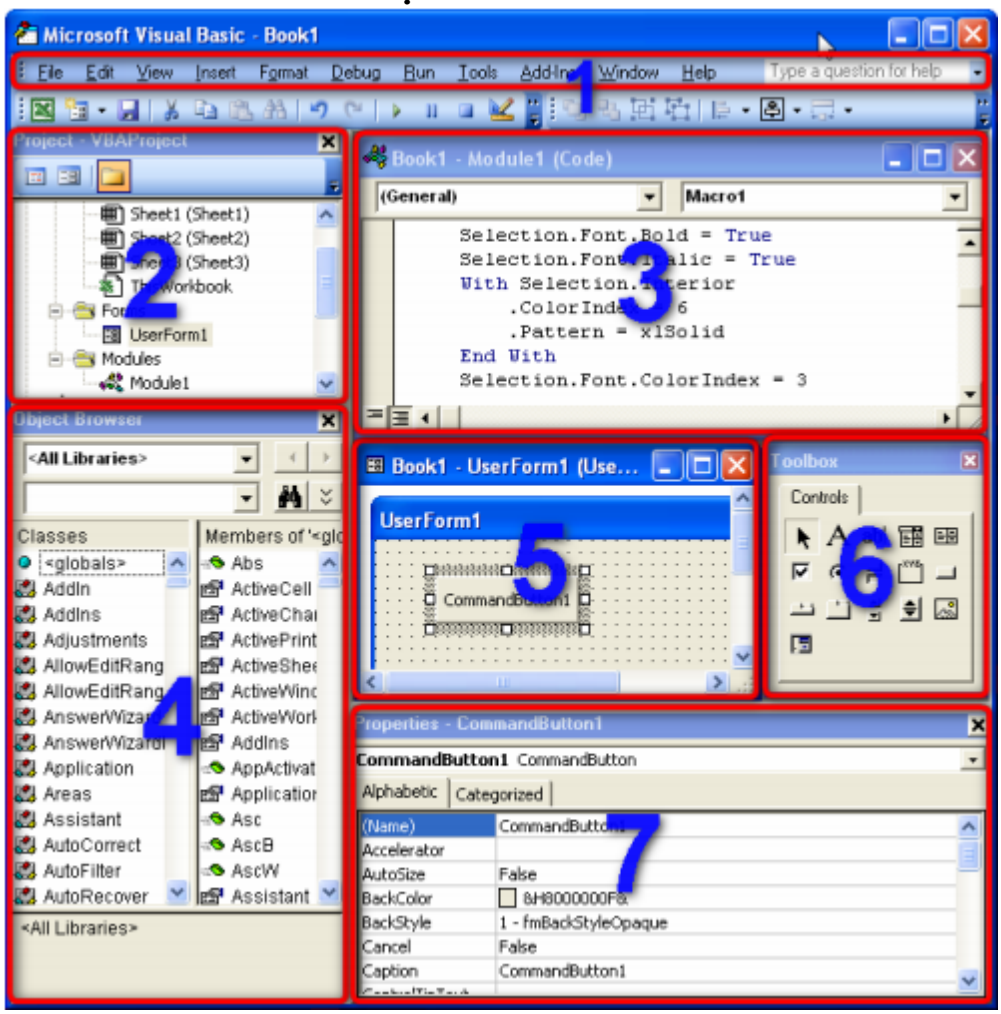


•**Medium**: Hỏi

•**Low**: Luôn chạy

* WORD, EXCEL, PowerPoint, Outlook, Access 2007: Show Developer tab in Ribbon

Giao diện chính của VBA IDE



1/Thanh trình đơn (***Menu bar***): chứa tất cả các lựa chọn cần thiết để thao tác với VBA IDE

2/Cửa sổ dự án (***Project Explorer Window***): liệt kê dưới dạng cây phân cấp các dự án hiện đang được mở trong VBA IDE và các thành phần có trong từng dự án như các tài liệu thành phần, các mô-đun chứa chương trình con, các mô-đun lớp, các cửa sổ do người dùng tạo.

3/Cửa sổ mã lệnh (***Code Window***): mỗi thành phần được liệt kê trong cửa sổ dự án đều có một cửa sổ mã lệnh riêng, chứa mã lệnh cho thành phần đó. Người dùng có thể hiệu chỉnh mã lệnh, tạo ra mã lệnh mới trong cửa sổ mã lệnh.

4/Cửa sổ tra cứu đối tượng (***Object Browser Window***): hiển thị các lớp, phương thức, thuộc tính, sự kiện và hằng số có trong thư viện đối tượng và trong dự án mà người dùng vừa tạo. Ta có thể sử dụng cửa sổ này để tìm kiếm, tra cứu tất cả các đối tượng mà ta vừa tạo ra cũng như các đối tượng trong các chương trình khác.

5/ Cửa sổ đối tượng trực quan (***Visual Object Window):*** khi người dùng tạo các đối tượng trực quan thì cửa sổ này sẽ cho phép người dùng thao tác trên các điều khiển một cách dễ dàng và thuận tiện.

6/Hộp công cụ chứa điều khiển ***(Tool Box***): chứa các thanh công cụ giúp người dùng có thể chèn các điều khiển vào cửa sổ người dùng (UserForm).

7/Cửa sổ thuộc tính (***Properties Window):*** cửa sổ này liệt kê tất cả các thuộc tính của đối tượng, qua đó người dùng có thể tham khảo và thay đổi các thuộc tính khi cần như màu chữ, tên đối tượng…

Bài 1: Cơ bản về ngôn ngữ lập trình VBA

-*.*.*-

1. Những qui định về cú pháp

Cú pháp được hiểu là một một tập hợp bao gồm các quy tắc, luật lệ về trật tự và hình thức viết của một câu lệnh hay một cấu trúc lệnh.

Trong ngôn ngữ lập trình Visual Basic (VB), cũng như các ngôn ngữ lập trình khác, đều có những quy định về cú pháp cho việc viết mã lệnh và người lập trình cần phải tuân theo các quy tắc này để trình biên dịch có thể dịch mã lệnh mà không phát sinh lỗi. Sau đây là các quy định cơ bản về cú pháp của VB:

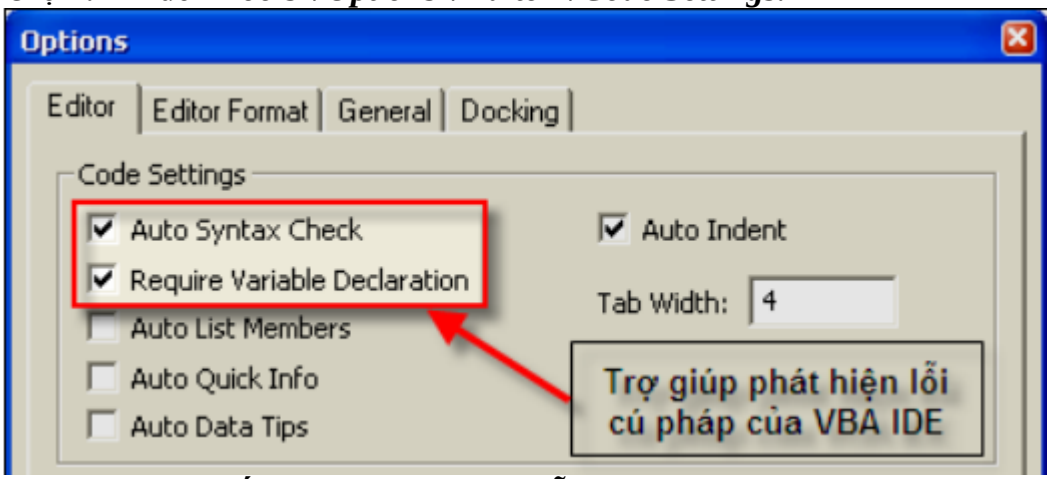
+ Các câu lệnh phải là các dòng riêng biệt. Nếu có nhiều lệnh trên cùng một dòng thì giữa các lệnh ngăn cách nhau bằng dấu hai chấm (**:**). Nếu dòng lệnh quá dài, muốn ngắt lệnh thành hai dòng thì sử dựng dấu cách và dấu gạch dưới (**_**).

+ Nếu muốn chèn thêm ghi chú, phải bắt đầu dòng chú thích bằng dấu nháy đơn (**'**).

+ Qui ước khi đặt tên: phải bắt đầu bằng kí tự kiểu chữ cái thông thường; không chứa dấu chấm, dấu cách hay các ký tự đặc biệt khác; không quá 255 kí tự; không trùng với các từ khoá; các biến có cùng một phạm vi thì không được đặt tên trùng nhau.

2. Các trợ giúp về cú pháp trong quá trình viết mã lệnh

-Các quy tắc về cú pháp thường khó nhớ đối với những người mới học lập trình hay mới sử dụng ngôn ngữ lập trình mới, cho nên, để thuận tiện cho người lập trình, VBA IDE cung cấp tính năng tự động phát hiện lỗi cú pháp trong quá trình viết mã lệnh. Tuy nhiên việc kiểm tra tự động này có thể gây khó chịu cho những lập trình viên chuyên nghiệp, những người rất hiếm khi mắc lỗi cú pháp khi lập trình, cho nên chức năng này chỉ hoạt động khi được kích hoạt, bằng cách chọn trình đơn ***Tools \ Options \ Editor \ Code Settings.***

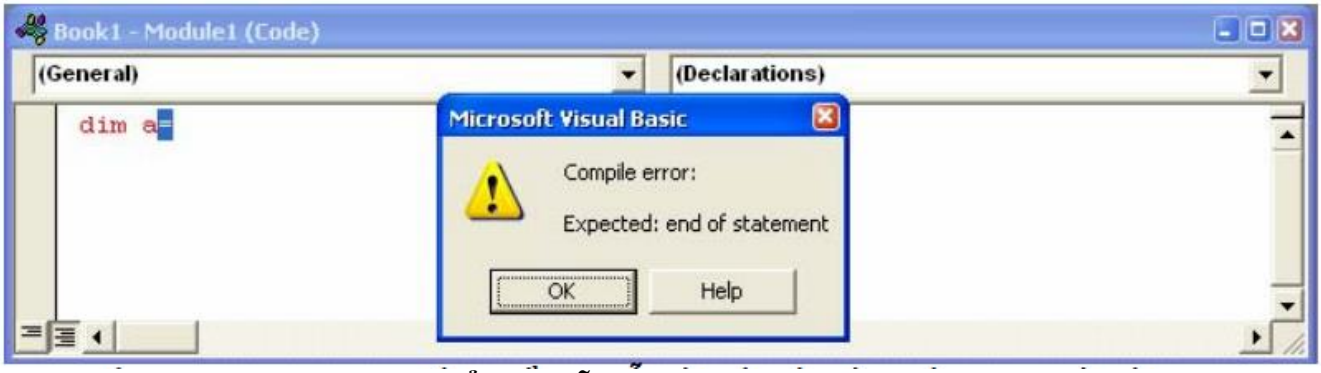


Bật /Tắt trợ giúp phát hiện lỗi cú pháp của VBA IDE

-Ý nghĩa của hai tùy chọn này như sau:

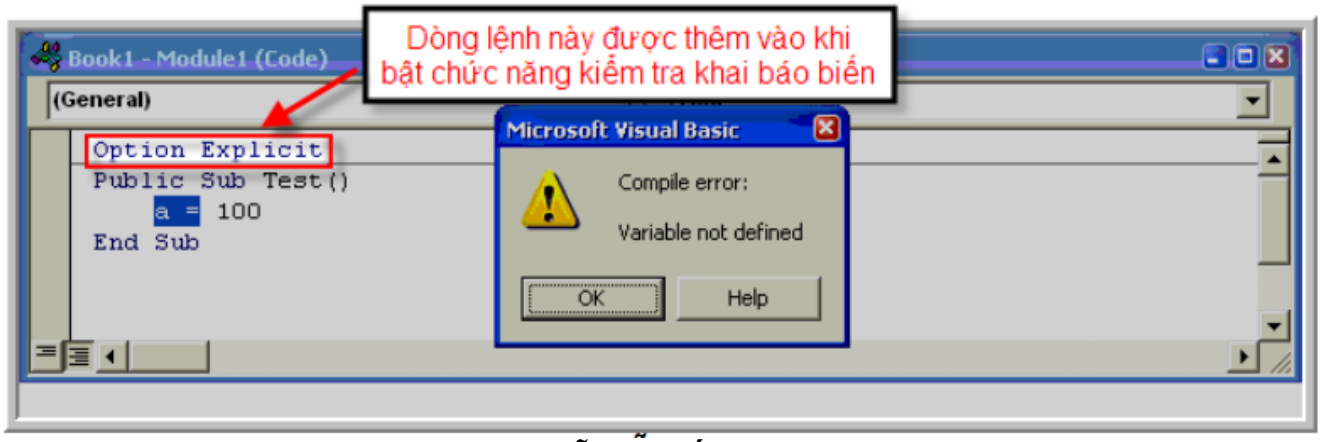
+Tự động kiểm tra lỗi cú pháp (Auto Systax Check): Tùy chọn này cho phép VBA IDE tự động phát hiện lỗi cú pháp ngay sau khi người dùng kết thúc dòng lệnh (xuống dòng mới), một hộp thoại (như hình dưới đây) sẽ thông báo vị trí gây lỗi cũng

như nguyên nhân gây lỗi. Nếu người dùng bỏ qua không sửa ngay thì dòng lệnh có lỗi sẽ được đánh dấu.



VBA IDE tự động kiểm tra lỗi cú pháp và thông báo cho người dùng

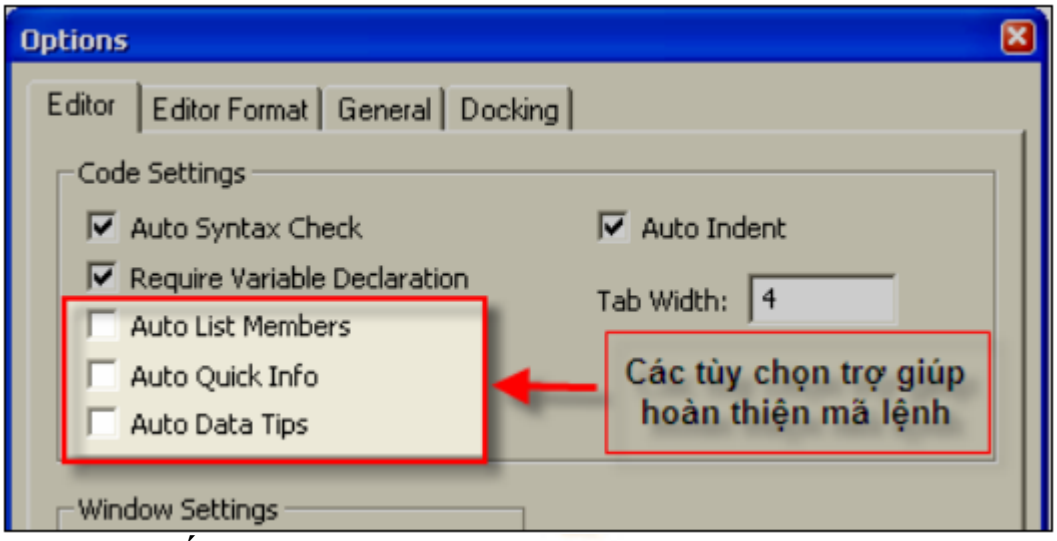
+Kiểm tra các biến (Require Variable Declaration): Trong VB, người dùng có thể sử dụng một biến mà không cần khai báo. Trong trường hợp này biến sẽ được khởi tạo và nhận một giá trị mặc định. Tuy nhiên, nếu lạm dụng điều này, rất có thể sẽ làm cho chương trình khó quản lý và dễ nhầm lẫn, vì thế VBA IDE cung cấp tùy chọn này để cho phép người dùng thiết lập tính năng kiểm soát quá trình khai báo biến. Khi tùy chọn này được kích hoạt, tất cả các biến đều phải khai báo trước khi sử dụng và VBA IDE sẽ tự động thêm vào đầu của mỗi mô-đun dòng lệnh "Option Explicit".



VBA IDE tự động thông báo lỗi khi biến được sử dụng mà chưa khai báo

3. Tính năng gợi nhớ và tự hoàn thiện mã lệnh

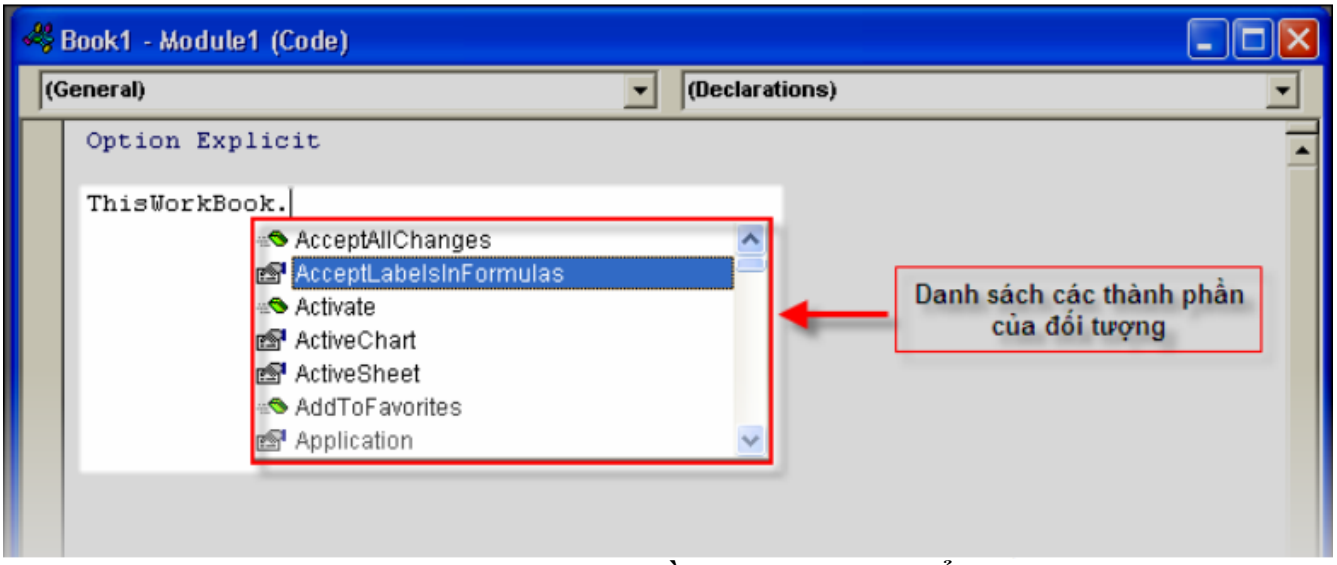
- Mã lệnh, thông thường là một tập hợp bao gồm các từ khóa, câu lệnh, tên biến hay toán tử được sắp xếp theo một trật tự nhất định. Tên của các thành phần này có thể khó nhớ chính xác hoặc quá dài, cho nên VBA IDE đưa ra tính năng này bằng cách hiển thị những thành phần có thể phù hợp với vị trí dòng lệnh đang soạn thảo trong một danh sách và sẽ tự động điền vàochương trình theo lựa chọn của người dùng (bấm phím Tab). Để kích hoạt tính năng này, trong VBAIDE, chọn trình đơn Tools \ Options \ Editor.



Bật / tắt trợ giúp hoàn thiện mã lệnh tự động trong VBA IDE

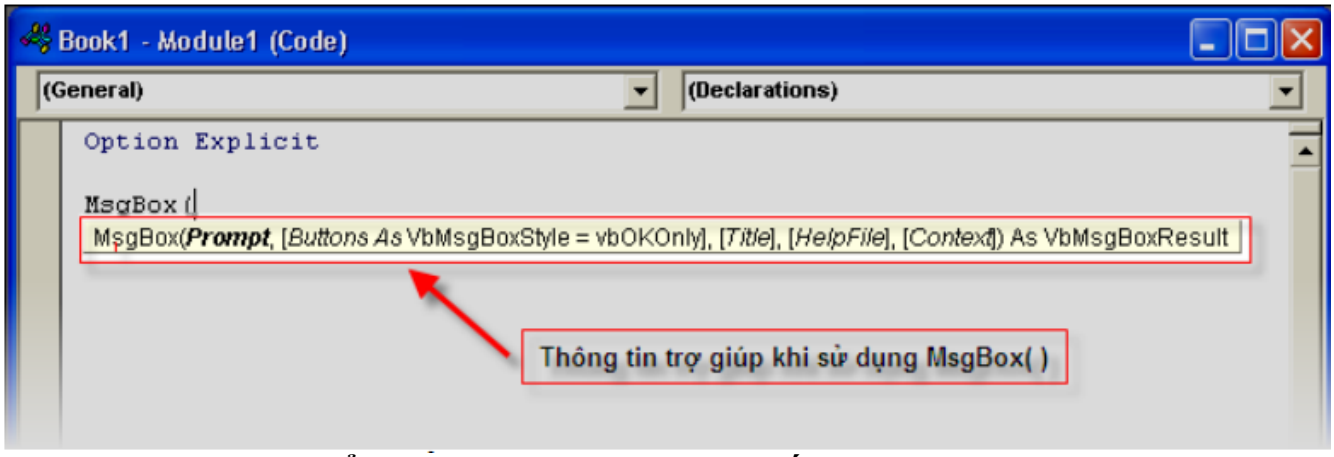
-Ý nghĩa của các tùy chọn này như sau:

+ Tự động hiển thị danh sách các thành phần của đối tượng (Auto List Member): Với tùy chọn này, khi một đối tượng của ứng dụng nền hay của chương trình được gọi ra để sử dụng thì một danh sách các thành phần của nó (bao gồm các phương thức và thuộc tính) sẽ được tự động hiển thị để người dùng chọn, sau khi bấm phím Tab, tên của thành phần này sẽ được tự động điền vào vị trí thích hợp trong dòng lệnh.



Danh sách các thành phần được tự động hiển thị.

+ Tự động hiển thị cú pháp cho chương trình con (Auto Quick Info): Với tùy chọn này, VBA IDE sẽ hiển thị những thông tin về tham số của một hàm hay thủ tục (đã được xây dựng từ trước) khi người dùng sử dụng nó. Các thông tin này bao gồm tên của tham số cùng với kiểu của nó.



Tự động hiển thị thông tin của các tham số trong chương trình con.

+ Tự động hiển thị giá trị của biến (Auto Data Tips): Với tùy chọn này, trong chế độ gỡ rối (Break mode), giá trị của biến (được gán trong quá trình chạy của chương trình) sẽ được hiển thị khi người dùng đặt chuột tại vị trí biến.

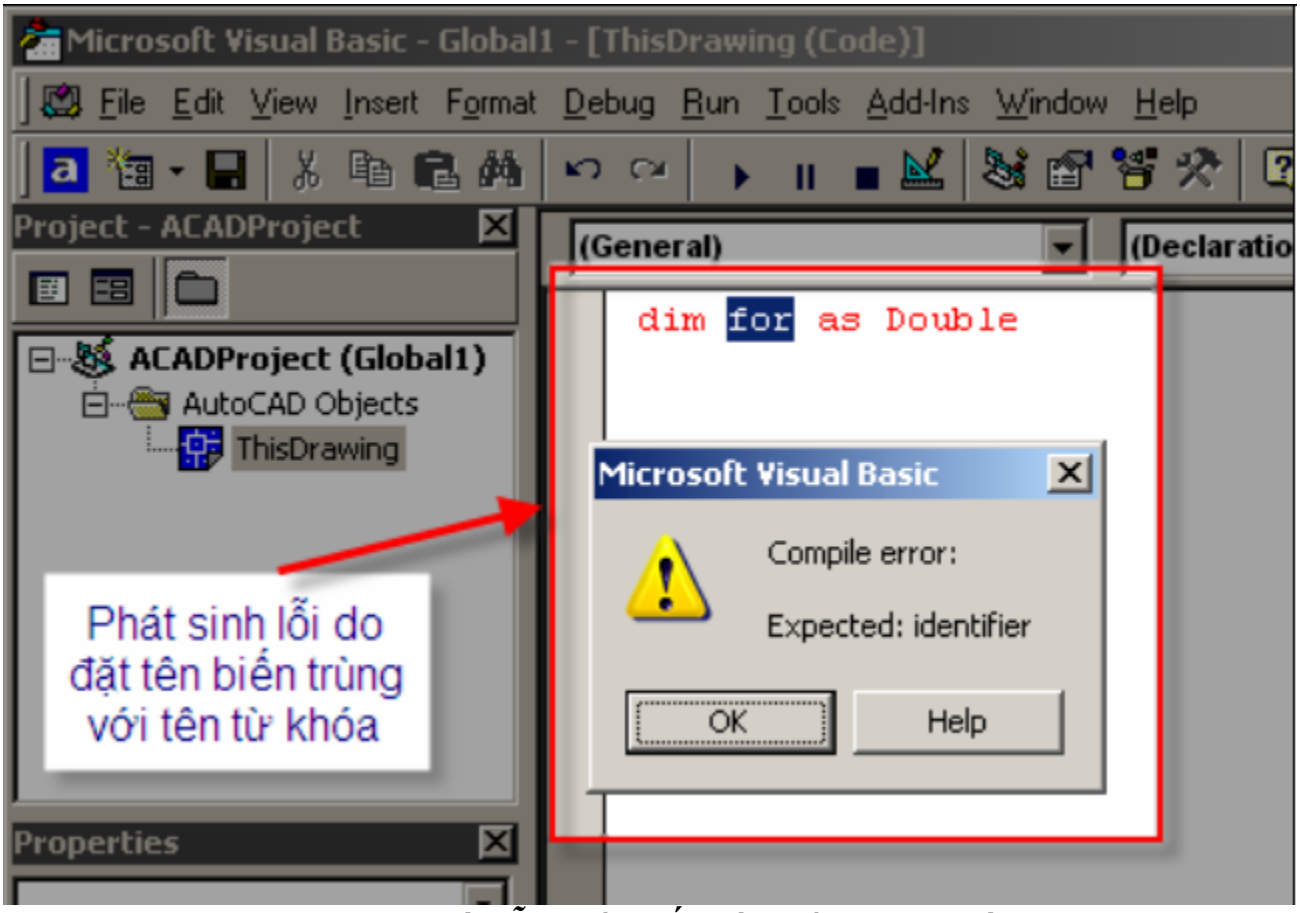
- Ngoài ra, nếu những tính năng trợ giúp trên chưa được kích hoạt, trong quá trình viết mã lệnh, người dùng có thể kích hoạt tạm thời chúng bằng cách nhấn tổ hợp phím Ctrl + Space. Cần chú ý rằng, khi danh sách trợ giúp hiện ra, người dùng có thể sử dụng chuột hoặc phím mũi tên để lựa chọn mục cần sử dụng trong danh sách đó rồi bấm phím Tab để xác nhận.

4. Từ khoá trong VBA

-Từ khoá là tập hợp các từ cấu thành một ngôn ngữ lập trình. Mỗi ngôn ngữ lập trình đều có một bộ từ khoá riêng, dưới đây là danh sách các từ khoá trong ngôn ngữ lập trình VB:

As	For	Mid	Print	String
Binary	Friend	New	**Private**	Then
ByRef	Get	Next	Property	Time
ByVal	Input	Nothing	**Public**	To
Date	Is	Null	Resume	True
Else	Len	On	Seek	WithEvents
Empty	Let	Option	Set	
Error	Lock	Optional	**Static**	
False	Me	ParamArray	Step	

- Các từ khóa là những từ được dùng riêng cho những chức năng khác nhau trong ngôn ngữ lập trình, ví dụ từ khóa “Private” hạn chế phạm vi sử dụng của biến hay chương trình con. Do đó việc đặt tên (biến, chương trình con) bắt buộc phải khác so với các từ khóa, nếu không sẽ phát sinh lỗi cú pháp.



VBA IDE báo lỗi do tên biến trùng tên với từ khóa

5. Các kiểu dữ liệu cơ bản

Kiểu dữ liệu là loại giá trị mà một biến có thể nhận, nói cách khác, khi một biến được khai báo thì ta buộc phải gán cho nó một kiểu dữ liệu nhất định. Về tổng thể có thể chia các kiểu dữ liệu trong VB ra làm hai loại:

+ Các kiểu dữ liệu được định nghĩa sẵn trong VB: là những kiểu dữ liệu cơ bản và thường gặp như kiểu số thực (Double), số nguyên (Integer), Chuỗi (String)...

+ Các kiểu dữ liệu do người dùng tự định nghĩa: là kiểu dữ liệu được tự xây dựng dựa trên những thành phần dữ liệu cơ bản trong VB. Cách xây dựng kiểu dữ liệu này được đề cập trong phần dưới.

5.1. Kiểu logic (boolean)

Chỉ chứa hai giá trị TRUE và FALSE (đúng và sai). Khi chuyển từ các dữ liệu dạng số sang kiểu logic, 0 sẽ được chuyển thành FALSE còn giá trị khác sẽ được chuyển thành TRUE. Khi chuyển từ kiểu logic sang kiểu số, giá trị FALSE sẽ được chuyển thành 0 còn giá trị TRUE sẽ được chuyển thành -1.

Ví dụ : ‘Khai báo biến A là kiểu logic

Dim phai **As Boolean**

(Biến phai lúc này chỉ có thể nhận cặp giá trị: True hay False.)

5.2. Kiểu số nguyên

Dùng để chứa các giá trị là số nguyên và có vài loại dữ liệu kiểu này. Sự khác nhau của những loại dữ liệu này là giới hạn giá trị (lớn nhất và nhỏ nhất) mà biến có thể nhận được(tham khảo bảng dưới).

Kiểu số nguyên	Kích thước	Phạm vi
Byte	1 byte	0 đến 255
Interger	2 bytes	-32,768 đến 32,767
Long	4 bytes	-2,147,483,648 đến 2,147,483,647

5.3. Kiểu số thực

Dùng để chứa các giá trị là số thực. Các kiểu số thực thường dùng được trình bày trong bảng dưới đây:

Kiểu số thực	Kích thước	Phạm vi
Single	4 byte	Từ -3.402823E38 đến -1.401298E-45 và từ 1.401298E-45 đến 3.402823E38
Double	8 bytes	-1.79769313486231E308 đến -4.94065645841247E-324 và từ 4.94065645841247E-324 đến 1.79769313486232E308
Currency	8bytes	Từ -922,337,203,685,477.5808 đến 922,337,203,685,477.5807

Ví dụ : Dim diem As Single

5.4. Kiểu mảng (array)

Khi gặp trường hợp phải sử lý một loạt các biến tương tự như nhau:

Ví dụ 1: các phần tử của một ma trận, nếu ta phải đặt tên khác nhau cho tất cả các biến này thì rất bất tiện, thay vào đó ta có thể dùng kiểu mảng để đặt tên chung cho cả nhóm các phần tử đó và khi nào cần sử dụng từng phần tử ta sẽ gọi tên theo chỉ số của chúng trong mảng.

```
'Khai báo mảng
Dim Matrix_1(10) As Double
```

Ví dụ 2: sau khai báo và sử dụng (gán giá trị cho phần tử) một ma trận 3 hàng 5 cột

```
'Khai báo mảng (3x5)
    Dim Matrix_3(1 To 3, 1 To 5) As Double
'Gán giá trị 100 cho phần tử tại hàng thứ 2 cột thứ 3
    Matrix_3(2,3)=100
```

5.5. Kiểu chuỗi (String)

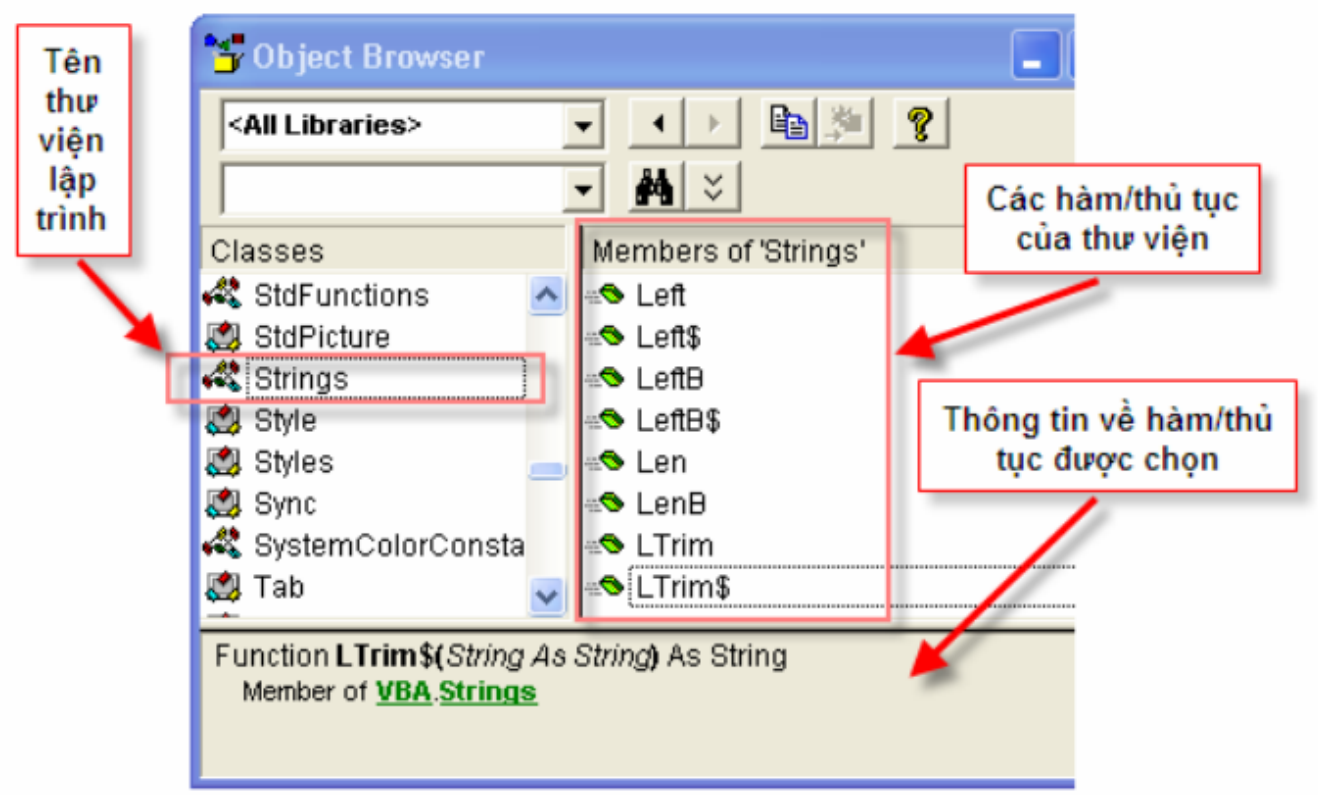
- Chuỗi là một hàng bao gồm các ký tự liên tục nhau, các ký tự ở đây rất đa dạng: có thể là chữ số, chữ cái, dấu cách (space), ký hiệu. Số lượng ký tự trong một chuỗi là rất lớn (216 ký tự). Mặc định trong VB, các biến hay tham số kiểu chuỗi có chiều dài thay đổi tùy theo giá trị dữ liệu được gán cho nó.

Ví dụ:

```
Dim S As String, ten As String
S="ABCD 1234 @#$%"
```

- Để tạo điều kiện thuận lợi cho người dùng, bên trong VB có sẵn một số hàm liên quan đến xử lý chuỗi, ví dụ như cắt chuỗi, tách chuỗi, ghép chuỗi, tìm kiếm, … Các hàm cơ bản này

được trình bày ở phần sau trong giáo trình này hoặc có thể tra cứu toàn bộ các hàm liên quan trong MSDN (Microsoft Developer Network) hoặc Object Browser (thư viện Strings) bằng cách nhấn phím F2 trong giao diện lập trình VBA IDE



Thông tin về các hàm trong thư viện lập trình của VBA được hiển thị trong Object Browser

5.6. Kiểu thời gian (Date)

- Dùng để lưu trữ và thao tác trên các giá trị thời gian (ngày và giờ). Định dạng ngày và giờ phụ thuộc vào các thiết lập về hiển thị trong hệ thống của người dùng. Khi chuyển từ các dữ liệu kiểu số sang kiểu ngày tháng, các giá trị ở bên trái dấu phẩy chuyển thành thông tin về ngày còn giá trị ở bên phải dấu phẩy sẽ được chuyển thành thông tin về giờ.

Ví dụ : Hiển thị thông tin về thời gian (ngày – tháng – năm) trong cửa sổ Immediate của VBA IDE.

```
Dim D As Date
Dim S As String
D = Now()
S = "Ngay: " & Day(D) & " - Thang: " & Month(D) & " - Nam: " & Year(D)
Debug.Print (S)
```

5.7. Kiểu Variant (ký tự)

- Kiểu Variant là một kiểu dữ liệu đặc biệt có thể chứa tất cả các loại dữ liệu, ngoại trừ kiểu chuỗi có chiều dài cố định. Kiểu Variant cũng có thể chứa các giá trị đặc biệt như Empty, Error, Nothing và Null.
- Tuy kiểu dữ liệu Variant có vẻ tiện dụng nhưng khi sử dụng một cách quá thoải mái thì nguy cơ gây lỗi của loại biến này là rất lớn, đặc biệt khi thao tác với các toán tử.

Ví dụ :

```
Dim V As Variant
‘ Gán biến V với một chuỗi
```

```
        V = “String”
‘ Gán biến V với một số
        V = 16
‘ Gán biến V với giá trị kiểu logic
        V = True
‘ Gán biến V với một dữ liệu kiểu thời gian
        V = #01/06/2007#
```

-Sử dụng hàm VarType(vVariant) sẽ cho ta mã của kiểu dữ liệu hiện đang lưu trữ trong biến Variant.

Giá trị VarType	Chú thích
0-vbEmpty	Không có gì trong variant
1-vbNull	Không có dữ liệu hợp lệ trong variant
2-vbInteger	Variant chứa Integer
4-vbSingle	Variant chứa Single
7-vbDate	Variant chứa Date/Time
8-vbString	Variant chứa String
9-vbObject	Variant chứa một Object
11-vbBoolean	Variant chứa Boolean

5.8. Kiểu lớp (Class)

- Kiểu lớp (Class) là một mở rộng của kiểu dữ liệu tự định nghĩa, sự khác biệt cơ bản ở đây là trong kiểu lớp còn có những đoạn chương trình dùng để xử lý chính những dữ liệu trong nó.Dữ liệu bên trong lớp thường được gọi là các thuộc tính (Properties), còn những đoạn chương trình trong lớp để xử lý dữ liệu này thực chất là các Hàm / Thủ tục (Function / Sub) được định nghĩa bên trong lớp và thường được gọi là các Phương thức (Methods). Một biến có kiểu dữ liệu là lớp được gọi là một đối tượng (Object) và cách sử dụng các Properties, Methods của đối tượng này tương tự như cách sử dụng các thành phần của kiểu dữ liệu tự định nghĩa.

- Lớp cần được xây dựng trong Class Module hoặc ta có thể sử dụng lại các lớp sẵn có từ các thư viện lập trình.

- Như vậy lớp có thể gồm các thành phần sau:

+ Các thuộc tính (Property): là các dữ liệu mô tả trạng thái của bản thân đối tượng hoặc các quan hệ của nó với các đối tượng khác.Về bản chất, thuộc tính là các biến được khai báo trong lớp đó. Kiểu dữ liệu của các thuộc tính có thể là các kiểu dữ liệu cơ bản hoặc có thể là một lớp khác (kiểu Class).

+ Các phương thức (Method): mô tả hành vi, chức năng của đối tượng. Về bản chất, phương thức là các chương trình con được xây dựng bên trong lớp và chúng có nhiệm vụ xử lý các dữ liệu của chính lớp đó.

+ Các sự kiện (Event): Sự kiện giúp cho lớp có khả năng giao tiếp với các lớp khác hoặc với môi trường ngoài.

6. Khai báo biến trong VBA

6.1. Khai báo biến:

- Cú pháp: Public/Dim/Private <tên biến> [as<kiểu>]

Ví dụ:

Dim **x** as **Integer**

Dim **y**
Public **d** as **Date**

- Nên khai báo đầy đủ kiểu để giảm số lỗi.
- Quy tắc đặt tên biến:
    + Bắt đầu với ký tự chữ
    + Không vượt quá 255 ký tự
    + Không trùng từ khóa VBA
    + Không chứa khoảng trắng, . , !, # , $, %, &, @
    + Phạm vi sử dụng của biến

Tên biến	Ghi chú
Byte	Số nguyên 0 - 255
Interger	Số nguyên (-32,768 đến 32,767)
Long	Số ngyên dài (-2,147,483,648 đến 2,147,483,647)
Single	Số thực (dài 4 byte)
Double	Số thực (dài 8 byte)
Boolean(kiểu logic)	True/ false (1/0)
Date	Ngày (dài 8 byte)
String	Chuỗi (khả năng lưu trữ từ 1 – 65,400 ký tự)

6.2 Khai báo hằng:

- Cú pháp: [Public/ Private] CONST <tên hằng> [as <kiểu>] = trị

Ví dụ:

Const x = 5
Const y = 1
Const Pi=3.14159 (có thể sử dụng giá trị 3.14159 bất cứ chỗ nào trong chương trình với cái tên dễ nhớ hơn là Pi)

7. Các toán tử và hàm thông dụng

7.1. Các toán tử

Toán tử được sử dụng cho mục đích xử lý dữ liệu. Ta sử dụng các toán tử để thực hiện tính toán, so sánh, gán và thực hiện nhiều thao tác khác. Dưới đây là danh sách và ý nghĩa của một số toán tử thông dụng:

+ **Toán tử gán** : = (Gán giá trị cho biến hoặc thuộc tính)
+ **Toán tử toán học**: +(cộng) - (trừ) *(nhân) / (chia) ^(lũy thừa) Mod (chia lấy phần dư) \ (chia lấy phần nguyên)
+ **Toán tử logic** :
    Not: Trả về giá trị phủ định với giá trị biểu thức. Not(TRUE)=FALSE
    And: Nối logic hai biểu thức. (TRUE And TRUE)=TRUE; các trường hợp khác cho kết quả bằng FALSE
    Or: (FALSE or FALSE)=FALSE; các trường hợp khác cho kết quả là TRUE
    Xor: Cho kết quả TRUE nếu hai đối số có cùng giá trị; ngược lại cho kết quả là FALSE
    Eqv: So sánh hai giá trị logic; cách thức xử lý tương tự như toán tử Xor
+ **Toán tử so sánh :**
    = : so sánh bằng

<> : Khác nhau
>: Lớn hơn
>=: Lớn hơn hoặc bằng
<: Nhỏ hơn
<=: nhỏ hơn hoặc bằng

Ví dụ 1: phép gán

ten = "Visual Basic"
phai = True
ngay = #10/18/2011#
diem = 6.5
diem = diem + 0.5
x = y^2 + (z mod y)

Ví dụ 2: viết hàm trả về thương số của 1 phép chia

```
Public Function ThuongSo(SoBiChia As Integer, SoChia As Integer)
   ThuongSo = SoBiChia \ SoChia
End Function
```

7.2. Các hàm toán học

Các hàm toán học được chứa trong thư viện Math (có thể tra cứu thư viện này bằng Object Browser) và có nhiệm vụ thực hiện các phép toán thông thường hay gặp. Sau đây là một số hàm thông dụng:

Abs(x) : Lấy giá trị tuyệt đối
Exp(x): Lấy mũ cơ số tự nhiên
Log(x): Logarit cơ số tự nhiên
Sqr(x): Lấy bình phương
Cos(x), Sin(x), Tan(x) : Hàm lượng giác
Atn(x): Hàm lượng giác ngược
Fix(x): Lấy phần nguyên (trước dấu phẩy). Fix(3.7)=3
Int(x): Lấy phần nguyên đã được làm tròn. Int(3.7)=4
Round(x,num) : Làm tròn số thực <x> đến <num> chữ số sau dấu phẩy
Val(str): Chuyển đổi chuỗi <str> thành giá trị kiểu số

7.3. Các hàm chuyển đổi dữ liệu

- Trong lập trình, cần chuyển đổi qua lại giữa các kiểu dữ liệu.
- Các hàm chuyển đổi kiểu là một phần không thể thiếu của các ngôn ngữ lập trình.
- VBA cung cấp các hàm chuyển đổi kiểu khá tiện. như sau:

+ CBool(Expression) : Chuyển đổi dữ liệu sang kiểu logic (Boolean)

+ CByte(Expression): Chuyển đổi tên_biến thành kiểu **Byte.** tên_biến có thể chứa dữ liệu số bất kỳ haydữ liệu chuỗi có thể chuyển đổi thành số, trong khoảng 0 đến 255. Nếu tên_biếnngoài khoảng đó, VBA báo lỗi Overflow. Nếu tên_biến là dấu chấm động, nó đượclàm tròn thành số nguyên gần nhất trước khi chuyển đổi kiểu.

+ Cdec: Chuyển đổi tên_biến thành kiểu phụ Decimal. Hàm này chấp nhận dữ liệu số haychuỗi có thể chuyển thành số, trong giới hạn của kiểu Decimal. Đây là hàm cung cấpphương thức duy nhất để tạo kiểu phụ Decimal.

+ Cdate: Chuyển đổi tên_biến thành kiểu Date. CDate chấp nhận số và chuỗi biểu diễn ngày và chuyển đổi thành dạng được mô tả bởi thông tin địa phương trên máy tính. Thí dụ máy tính được cài đặt dạng ngày của Mỹ mm/dd/yy, nếu ta nhập ngày dạng Anhdd/mm/yy và dùng hàm Cdate thì ngày được sẽ chuyễn đổi sang dạng Mỹ.

+ Ccur: Chuyển đổi tên_biến thành kiểu Currency. CCur chấp nhận dữ liệu số hay chuỗi bấtkỳ biểu diễn giá trị tiền tệ. Hàm này nhận dạng thập phân hay dấu phân cách hàngngàn theo thông tin địa phương của máy tính. Chỉ dùng cho VBA.

+ CInt(Expression): Chuyển đổi dữ liệu sang kiểu nguyên (Integer)

+ CLng(Expression): Chuyển đổi dữ liệu sang kiểu nguyên (Long)

+ CDbl(Expression): Chuyển đổi dữ liệu sang kiểu thực (Double**)**

+ CSng(Expression): Chuyển đổi dữ liệu sang kiểu thực (Single)

+ CStr(Expression): Chuyển đổi dữ liệu sang kiểu xâu (String)

+ Str(Number): Chuyển đổi dữ liệu số sang kiểu xâu (String)

+ Val(String As String): Chuyển đổi dữ liệu từ String sang Double

Ví dụ 1:

```
Public Sub Test ()
    Dim StrA as String
    Dim A as Double
    StrA=”1234”
    A=Val(StrA) ‘ Kết quả A=1234
    Debug.print A
    A=4567
    StrA=Str(A) ‘ Kết quả StrA=”4567”
    Debug.Print StrA
End Sub
```

Ví dụ 2: Val(chuoi) dạng số đầu tiên tìm thấy trong chuỗi

Val(“30something”) □ => 30

CInt(“30something”) => lỗi

7.4. Các hàm xử lý chuỗi

Các hàm loại này được chứa trong thư viện Strings (có thể tra cứu thư viện này bằng Object Browser). Sau đây là một số hàm thông dụng:

+ Asc(str): Trả về mã ASCII của ký tự đầu trong một chuỗi

+ **Chr(mumchar)** : Chuyển đổi từ mã ASCII sang một ký tự

+ CStr(exp): Chuyển biểu thức sang dạng chuỗi

+ LCase(str): Chuyển sang chữ thường, dạng variant (ký tự)

+ LTrim(str): Cắt các khoảng trắng đầu str, dạng variant

+ RTrim(str): Bỏ các khoảng trắng bên phải, dạng variant

+ Trim(str): Bỏ tất cả các khoảng trắng dư, dạng variant

+ Str(num): Đổi numchuwx chuỗi dạng variant

+ **Left(String, Length as Long)** : Trích dữ liệu bên trái của một chuỗi

+ **Mid(String, Start As Long, [Length]):** Trích dữ liệu phần giữa của một chuỗi

+ **Right(String, Length As Long)** : Trích dữ liệu phần bên phải của một chuỗi

+ **Split(String):** Tách một chuỗi dài thành một mảng gồm nhiều chuỗi nhỏ hơn

+ Joint(StringArray): Gộp một mảng các chuỗi thành một chuỗi duy nhất

+ **Len(String):** Trả về độ dài của chuỗi (số lượng ký tự trong chuỗi bao gồm cả ký tự trống)

+ **Ucase(String):** Hàm thực hiện đổi tất cả các ký tự trong chuỗi thành chữ HOA.

+ **InStr([start,]string1, string2[, compare]) :** Trả về vị trí bắt đầu của chuỗi String2 trong chuỗi String1.

Ví dụ: Lưu ý là dấu "_" trong ví dụ trên có thể thay thế bằng bất cứ ký tự nào.

```
Public Sub Test()
    Dim StrArDes() As String
    ' Mảng các chuỗi được khai báo dạng mảng động
    Dim StrScr As String 'Chuỗi ban đầu
    StrScr = "Point1_23.5_4.5_44.8"
    StrArDes = Split(StrScr, "_")
    ' Tách chuỗi StrScr thành một mảng các chuỗi và đưa vào StrArDe
    ' kí tự ngăn cách là "_"
    ' Khi đó StrArDes(0)="Point1", StrArDes(1)="23.5"
    ' StrArDes(2)="4.5", StrArDes(3)="44.8"
    Debug.Print StrArDes(0), StrArDes(1), StrArDes(2), StrArDes(3)
End Sub
```

Kết quả

8.Các cấu trúc điều khiển

8.1. Cấu trúc điều kiện

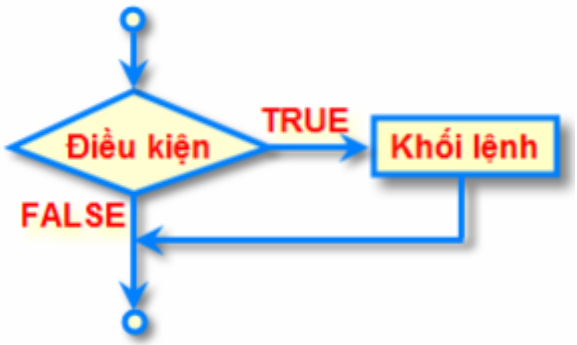
Các từ khóa: If, Then, Else, ElseIf, End If

Cú pháp: **If <biểu_thức_điều_kiện> then**

Khối_lệnh

End If

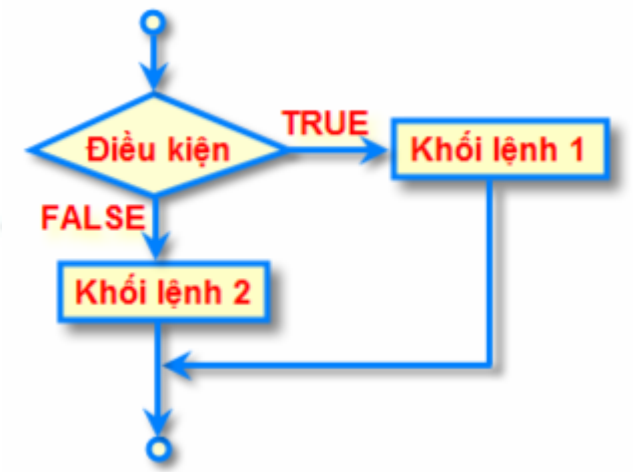
Diễn giải tiến trình của cấu trúc điều kiện như sau: nếu <biểu_thức_điều_kiện> là đúng thì chương trình sẽ thực hiện <khối_lệnh>, nếu sai thì chương trình sẽ thoát khỏi cấu trúc lệnh này. Sơ đồ khối của cấu trúc lệnh kiểu này có thể được biểu diễn như sau:



Ngoài cấu trúc cơ bản và trường hợp riêng ở trên, trong nhiều trường hợp, ta buộc phải xử lý khi <Biểu_thức_điều_kiện> trả về giá trị False (sai). Để giải quyết tình huống này ta sử dụng cấu trúc điều kiện mở rộng như sau:

If <biểu_thức_điều_kiện>
Khối_lệnh_1
Else
Khối_lệnh_2
End If

Diễn giải tiến trình của cấu trúc lệnh này như sau: nếu <biểu_thức_điều_kiện> là đúng thìchương trình sẽ thực hiện <khối_lệnh_1>, còn nếu không đúng thì chương trình sẽ thực hiện <khối_lệnh_2>.
Sơ đồ khối của cấu trúc lệnh kiểu này có thể được biểu diễn như sau:



<u>Ví dụ1</u> : viết hàm nhận vào 1 số nguyên, cho biết số đó chẵn hay lẻ.

```
Public Function ChanLe(So As Integer)
        If (So mod 2) = 0 then
                ChanLe = "Số chẵn"
         Else
                 ChanLe = "Số lẻ"
        End If
End Function
```

Các cấu trúc lệnh điều kiện có thể được lồng nhau để thể hiện những thao tác phức tạp hơnbằng cách sử dụng thêm từ khoá ElseIf. Như vậy, cấu trúc điều kiện có cú pháp tổng quát như sau:

```
If <điều_kiện_1> Then
        [Khối_lệnh_1]
[ElseIf <điều_kiện_n> The
          [khối_lệnh_n]
…
[Else
        [Khối_lệnh_2]]
End If
```

Ví dụ 2: viết hàm nhận vào 1 số, cho biết số đó âm, dương, hay bằng 0.

```
Public Function Dau(So As Double)
    If (So > 0) then
        Dau = "so duong"
     Elseif (So < 0) then
        Dau = "so le"
     Else
        Dau = "so 0"
    End If
End Function
```

8.2. Cấu trúc lựa chọn

Cấu trúc này sử dụng khi ta muốn thực hiện một số lệnh nào đấy tương ứng với từng giá trị của biểu thức kiểm tra.

Các từ khoá sử dụng trong cấu trúc này: Select Case, Case, Case Else, End Select.

Cú pháp của cấu trúc lựa chọn:

```
Select Case <biểu_thức_kiểm_tra>
[Case điều_kiện_1
    [khối_lệnh_1]]
 ...
[Case điều_kiện_n
    [khối_lệnh_n]]
[Case Else
    [khối_lệnh_else]]
End Select
```

Diễn giải tiến trình của cấu trúc lựa chọn như sau: Giá trị của <biểu_thức_kiểm_tra> sẽ được so sánh với các <điều_kiện_i> nếu giá trị của <biểu_thức_kiểm_tra> thoả mãn <điều_kiện_i> thì <khối_lệnh_i> tương ứng sẽ được thực hiện, sau đó chương trình sẽ thoát khỏi cấu trúc lựa chọn. Trong trường hợp giá trị của <biểu_thức_kiểm_tra> không thoả mãn tất cả các điều kiện thì <khối_lệnh_else> sẽ được thực hiện nếu có từ khoá Case Else, còn nếu không có từ khoá Case Else thì chương trình sẽ thoát khỏi khối lệnh lựa chọn này mà không thực hiện gì cả.

Ví dụ : viết hàm nhận vào tháng và năm, cho biết tháng đó có bao nhiêu ngày?

```
Public Function SoNgay(Thang As Byte, Nam As Integer)
    Dim sn As Byte, tmp As Integer
    Select Case Thang
        Case 1, 3, 5, 7, 8, 10, 12: sn = 31
        Case 4, 6, 9, 11: sn = 30
        Case 2
            tmp = Nam Mod 100
            If tmp = 0 Then tmp = Nam \ 100
            If (tmp Mod 4) = 0 Then sn = 29 Else sn = 28
    End Select
    SoNgay = sn
End Function
```