BÀI 4

Các điều khiển và cấu trúc điều khiển

Giảng viên: Ts. Chu Thị Hồng Hải

MỤC TIÊU

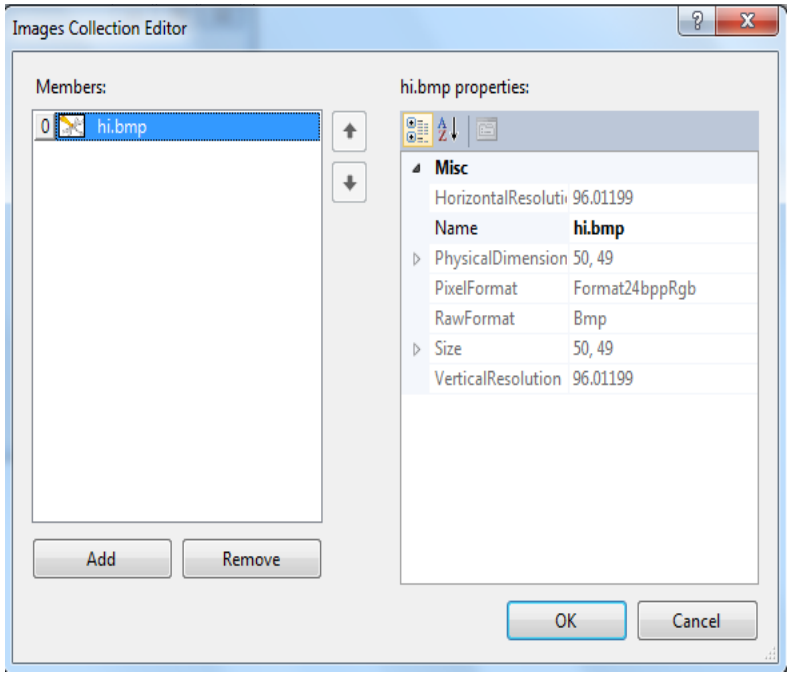
- ❑ Nhận diện được một số điều khiển nâng cao
- ❑ Sử dụng thành thạo các cấu trúc điều khiển cơ bản
- ❑ Ứng dụng được các điều khiển vào các bài toán cụ thể

Nội dung

- Một số điều khiển nâng cao
- Các cấu trúc điều khiển
- Thủ tục sub và hàm function
- Phạm vi của biến
- Bẫy lỗi và sử dụng cấu trúc xử lý lỗi.

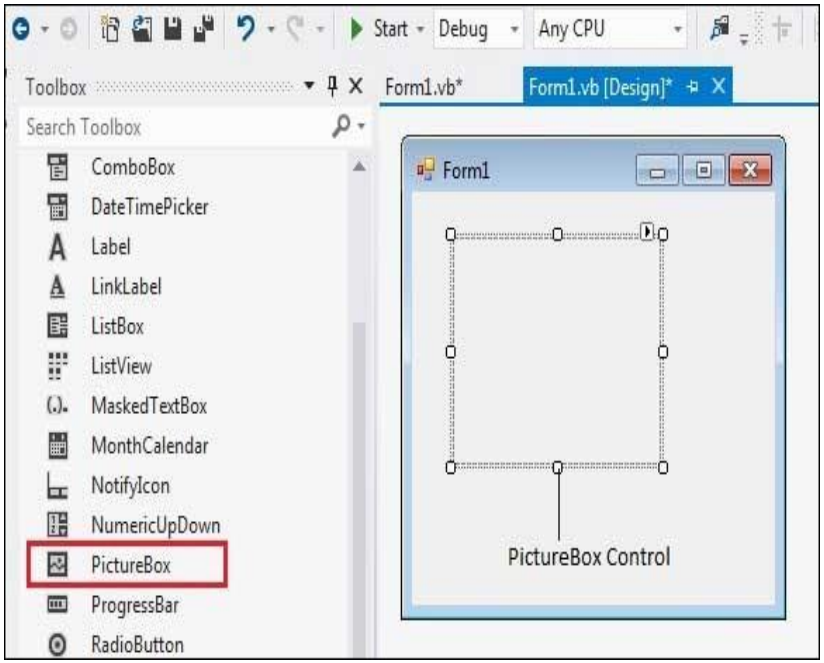
Điều khiển ImageList

- ImageList control có chức năng lưu trữ hình ảnh và liên kết đến các controls khác.
- **Thuộc tính**:
  - ✓ **Images**: tập hợp các ảnh của ImageList.
  - ✓ **ImageSize**: kích thước của các ảnh có trong Image
  - ✓ **TransparentColor**: màu nền của ảnh.
  - ✓ Vào thuộc tính Images của ImageList và nhấp vào để đến cửa sổ Images Collection Editor và thêm hình ảnh:
  - ✓ **Phương thức**:
    - **Draw**: vẽ một ảnh xác định



Điều khiển PictureBox

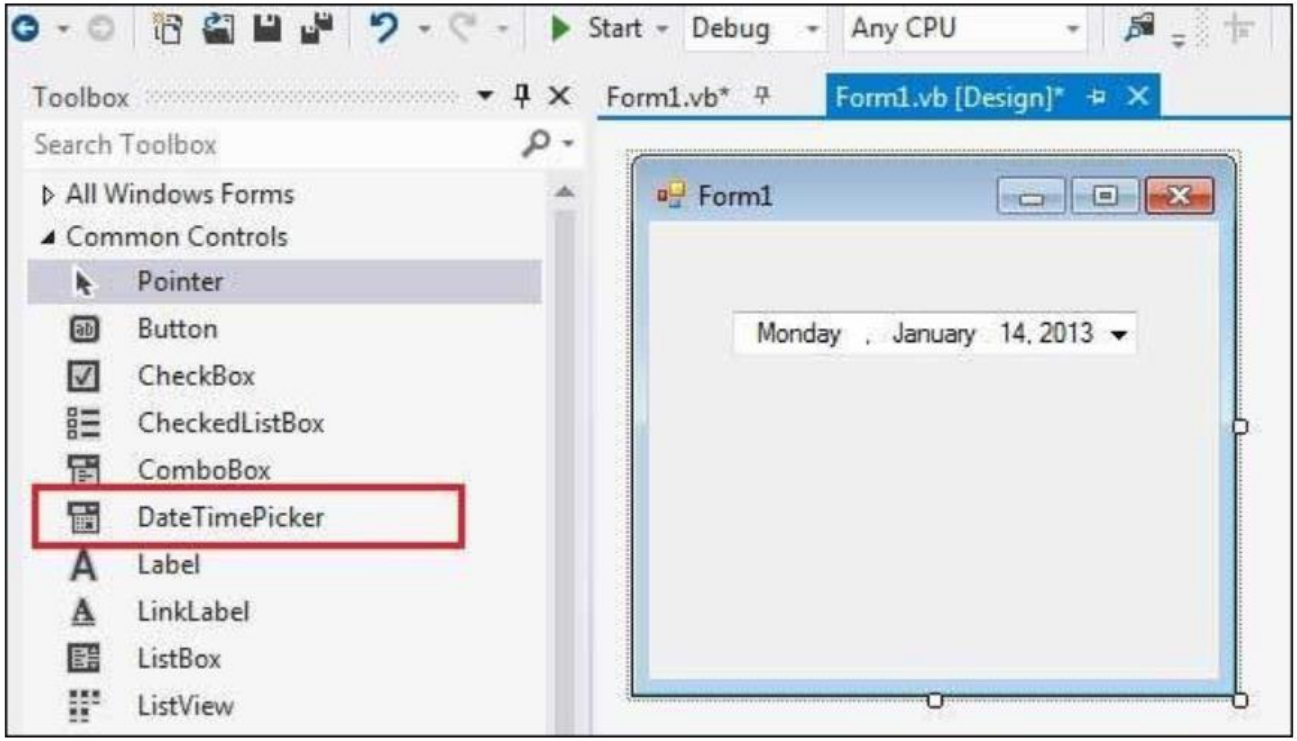
- Điều khiển PictureBox dùng để hiển thị hình ảnh trên biểu mẫu.



- **Thuộc tính**:
  - ✓ **Image**: Xác định hình ảnh cần hiển thị.
  - ✓ **SizeMode**: Xác định cách thức ảnh được hiển thị (AutoSize, CenterImage, Normal, StretImage)
  - ✓ **Sự kiện:**
    - ✓ **SizeModeChange**: xảy ra khi thuộc tính SizeMode thay đổi giá trị.

Điều khiển DateTimePicker

- Điều khiển ngày cho phép người dùng nhập liệu giá trị ngày theo đúng quy cách yêu cầu của chương trình.



Điều khiển DateTimePicker

- **Thuộc tính :**

✓ **CustomFormat**: Quy định các hiển thị ngày tháng năm giờ phút giây trên điều khiển. Thuộc tính này chỉ có tác dụng khi **Format** có trị là **Custom**.

✓ **Format**: Quy định cách hiển thị ngày tháng năm giờ phút giây theo hình thức định sẵn.

- ***Long*** *:* hiển thị ngày tháng năm đầy đủ nhất.
- ***Short*** *:* hiển thị ngày tháng năm ngắn gọn.
- ***Time*** *:* hiển thị giờ phút giây.
- ***Custom*** *:* hiển thị theo định dạng của thuộc tính **CustomFormat**.

✓ **MaxDate**: Quy định giá trị lớn nhất được phép nhập, chọn trên điều khiển.

✓ **MinDate**: Quy định giá trị nhỏ nhất được phép nhập, chọn trên điều khiển.

✓ **ShowUpDown:** Thuộc tính luận lý, mặc định là False, chỉ hiển thị nút sổ xuống để hiển thị lịch tháng cho chọn ngày. Nếu làTrue, sẽ hiển thị bộ nút tăng giảm cho phép chọn thành phần để thay đổi giá trị qua bộ nút này.

Điều khiển DateTimePicker

Ví dụ: ứng dụng DateTimePicker chọn một ngày và in ra thông tin ngày chọn

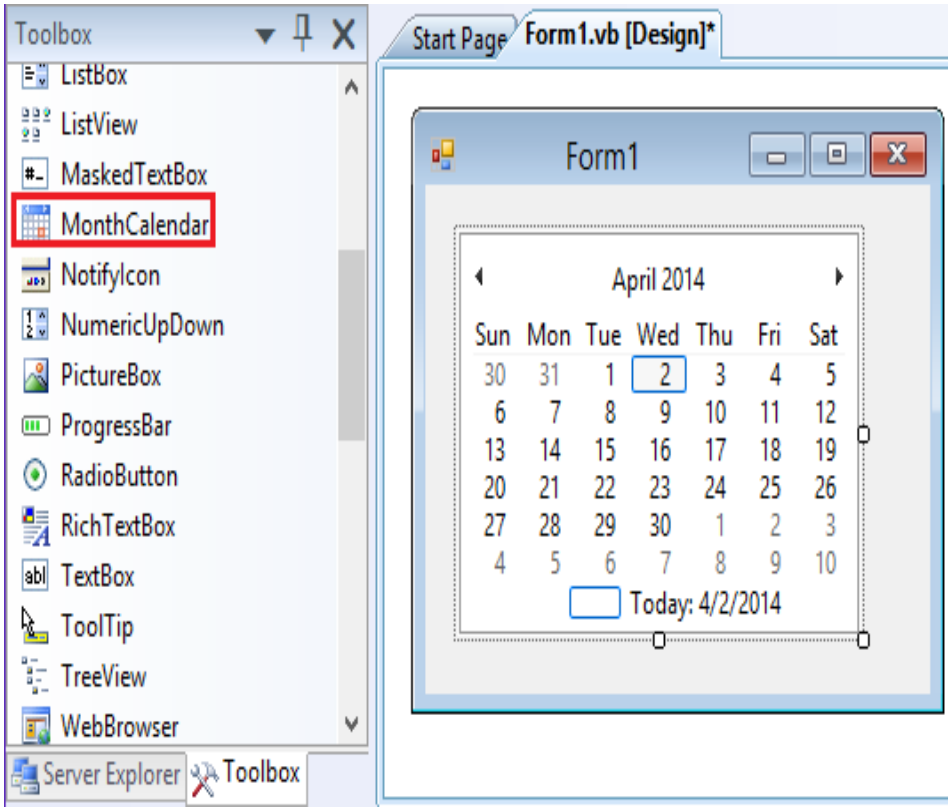
- Thiết kế như hình dưới
- Viết Code như sau:

```
MsgBox("Ngày sinh của bạn là: " & DateTimePicker1.Text)

MsgBox("Ngày trong năm: " & DateTimePicker1.Value.DayOfYear.ToString)

MsgBox("Hôm nay là ngày: " & Now.ToString)
```

Chạy chương trình và quan sát kết quả

Điều khiển MonthCalendar

- **MonthCalendar** là điều khiển được sử dụng để chọn ngày.



- **Thuộc tính:**

✓ **FirstDayOfWeek:** Giá trị Mặc định là Chủ nhật.

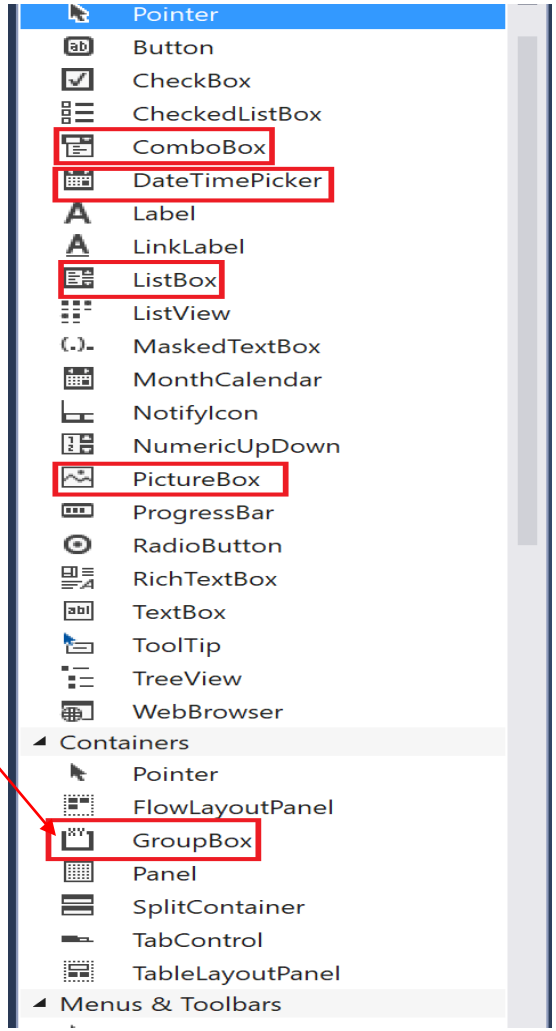
✓ **ShowToday:** Hiển thị ngày hiện tại ở cuối lịch (true) hoặc không (false)

✓ **ShowTodayCircle:** Khoanh tròn vào giá trị được chọn

✓ **ShowWeekNumber:** Mặc định là False. Hiển thị số tuần hiện tại trong năm (true)

Khung (GroupBox)

- **Khái niệm**: Là điều khiển được dùng để nhóm các điều khiển khác.



Khung (GroupBox)

- **Thuộc tính**:

  + **Name**: Đây là một tên xác định một định danh, người lập trình có thể thay đổi tên này theo cách của mình để tiện sử dụng.

  + **Text**: Chuỗi trên tiêu đề của GroupBox.

  + **Font, Fore Color**: Quy định kiểu chữ, kích thước, màu hiển thị.

  + **BackColor**: Quy định màu nền của khung trong trường hợp không trong suốt. Chọn backcolor là Transparent trong tab Web của cửa sổ chọn màu để có màu nền giống với màu đối tượng hoặc đang chứa khung.

Khung (GroupBox)

- **Thuộc tính**:
  - **BackgroundImage**: Chọn hình nền cho khung.
  - **Enable**: có giá trị mặc định là True, cho phép hoặc không cho phép kích hoạt vào khung.
  - **Visible**: có giá trị mặc định là True, dùng để ẩn hiện khung.
  - **TabIndex**: Số thứ tự của đối tượng trên Form. Ta có thể nhấn phím Tab trên bàn phím để chuyển đổi qua lại giữa các đối tượng bằng chỉ số Tabindex. Chỉ số nào được tự động tạo ra khi ta thêm một đối tượng vào Form.
- **Phương thức**:
  - **Location**: di chuyển khung đến tọa độ X,Y: location = new point(X, Y)
  - **Sự kiện:**

    Click, DblClick: xảy ra khi khung nhận được một thao tác nhấp (nhấp đúp) chuột.

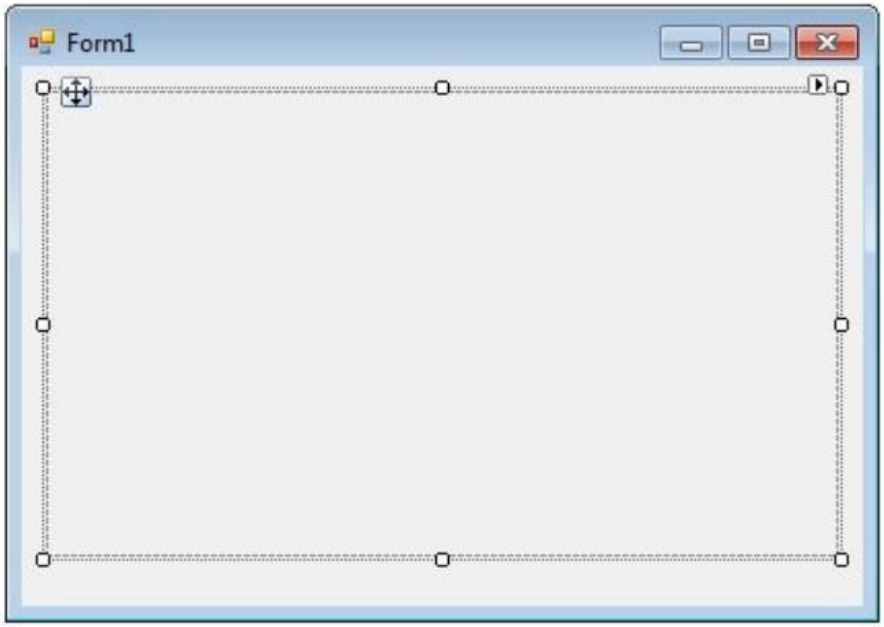
Panel

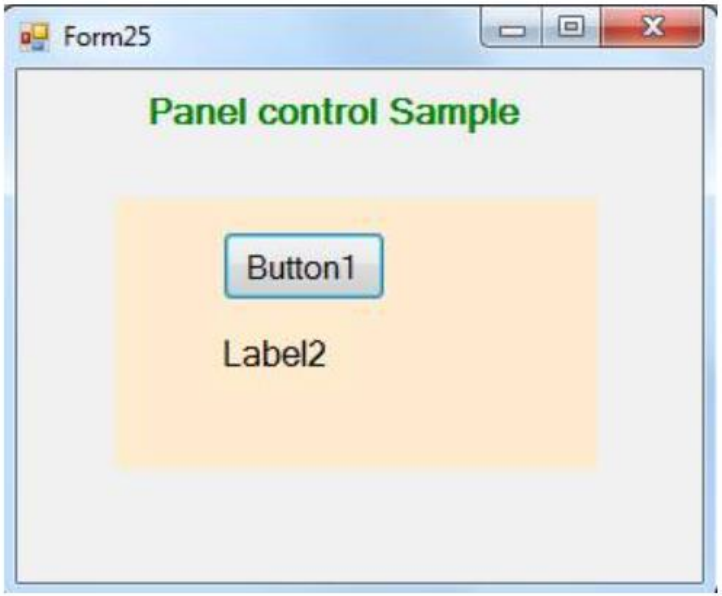
- Là một vùng chứa các điều khiển khác.
- Thuộc tính:

BackColor: màu nền cho điều khiển

BorderStyle: viền cho điều khiển.

Visible: ẩn hoặc hiển thị điều khiển



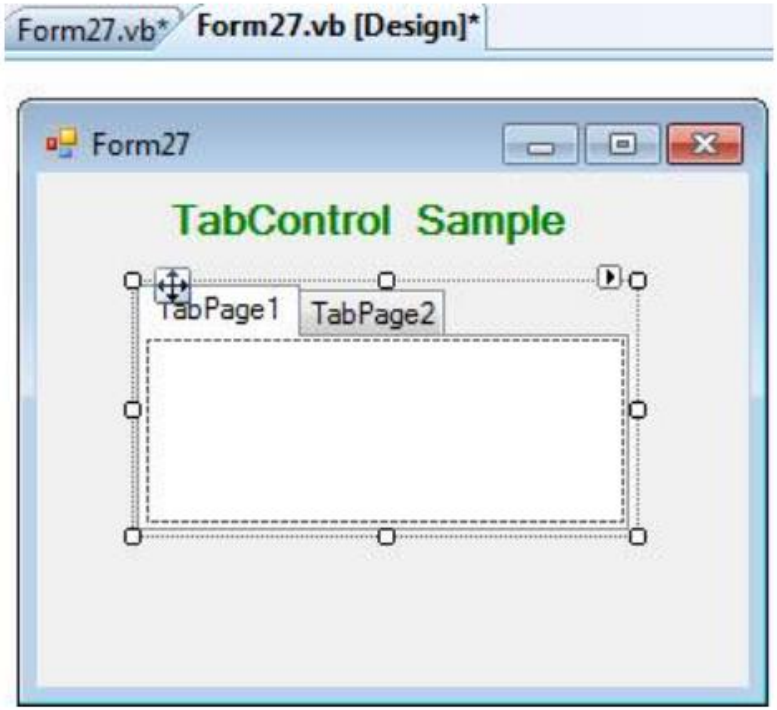


- **Ví dụ:** Thiết kế form có panel như hình trái
- Gợi ý:

Panel1.BorderStyle = BorderStyle.FixedSingle
Panel1.BackColor = Color.BlanchedAlmond

TabControl

- Là một điều khiển vùng chứa cho phép bạn hiển thị nhiều tab trên một biểu mẫu duy nhất và nó cho phép chuyển đổi giữa các tab.



- **Thuộc tính:**
  - ✓ **ItemSize:** kích thước của điều khiển.
  - ✓ **TabStop:** thiết lập giá trị cho phím TAB
  - ✓ **Visible:** cho phép hay không các điều khiển con của nó có được hiển thị hay không.

TabControl

- Thiết kế Tab như hình dưới

- Thực hiện
  - Thiết kế như hình bên
  - Viết code:
- TabPage1.Text = 'Email' 'Header text of TabPage1 will be changed
- TabPage2.Text = 'Language' 'Header text of TabPage2 will be chnaged

Các cấu trúc điều khiển

❑ Dữ liệu và biến

- Các loại dữ liệu và các phép toán
- Biến và hằng

Các loại dữ liệu và các phép toán

- Vb.net có các loại dữ liệu sau:

Kiểu dữ liệu	Kích thước	Phạm vi	Ví dụ
Short	16-bit	-32,678 - 32,767	Dim S as Short S = 12500
Integer	32-bit	-2,147,483,648 đến 2,147,483,647	Dim I as Integer S = 4000
Long	64-bit	-9,233,372,036,854,775,808 đến 9,233,372,036,854,775,807	Dim L as Long L = 3988890343
Single	32-bit (dấu phảy động)	-3.402823E38 đến 3.402823E38	Dim Sg as Single Sg = 899.99
Double	64-bit (dấu phảy động)	-1.797631348623E308 đến 1.797631348623E308	Dim D as Double D=3.1.4159265
Decimal	128-bit	Trong khoảng +/-79,228x10^{24}	Dim Dc as Decimal Dc=7234734.5

Các loại dữ liệu và các phép toán

Kiểu dữ liệu	Kích thước	Phạm vi	Ví dụ
Byte	8-bit	0-255	Dim B as Byte B=12
Char	16-bit	0-65,536	Dim Ch As Char Ch=”L”
String	Nhiều ký tự	Chứa 0 đến 2 tỷ ký tự	Dim St As String St=”Đức Lập”
Boolean	16-bit	Hai giỏ trị True hay False	Dim Bl As Boolean Bl = True
Date	64-bit	Từ 1/1/1 đến 31/12/9999	Dim Da As Date Da=#16/07/1984
Object	32-bit	Bất kỳ kiểu đối tượng nào	Dim Obj As Object

Biến và hằng

❑ Biến

Biến là một đối tượng có giá trị thay đổi trong chương trình. Biến có các đặc tính sau cần quan tâm

- **Name**: Tên của biến. Tên của biến phải là định danh hợp lệ trong VB.Net, nghĩa là phải bắt đầu bằng một chữ cái hoặc ký tự _ và không được trùng với các từ khóa củaVB.Net. Trường hợp muốn dùng từ khóa làm tên biến phải được dùng trong ngoặc vuông như [String], [Boolean], … Tên biến nên có ý nghĩa gợi nhớ đến nội dung trong nó như *Don_gia*, *So_luong_xuat*.
- **Address**: Địa chỉ vùng nhớ nơi lưu giữ giá trị của biến. Trong thời gian sống của chương trình, địa chỉ của biến có thể thay đổi.
- **Type**: Kiểu của biến, còn gọi là kiểu dữ liệu. Mỗi biến phải thuộc về một kiểu dữ liệu trong Common Type System.
- **Value**: Giá trị. Giá trị của biến phải phù hợp với kiểu dữ liệu của biến.
- **Scope**: Phạm vi sử dụng của biến.
- **LifeTime**: Thời gian tồn tại của biến.

Biến và hằng

- **Biến: S**ử dụng để lưu trữ dữ liệu trong quá trình tính toán, trong VB.Net biến cần được khai báo trước khi sử dụng.
- **Cú pháp** khai báo biến thông thường như sau:

Dim|private|public Tên_biến as Kiểu_dữ_liệu

Ví dụ . Khai báo biến a kiểu số nguyên, biến ngaysinh kiểu Date

```
Dim a as interger
Public ngaysinh as date
```

- ***Private**: chỉ ra rằng biến chỉ có phạm vi trong module mà nó được khai báo không cho các thủ tục, hàm ở module khác được phép truy cập vào.*
- ***Public**: chỉ ra rằng biến cho phép không chỉ các thủ tục và hàm trong module mà đó được khai báo sử dụng mà còn cho các hàm, thủ tục ở module khác được phép truy cập vào.*

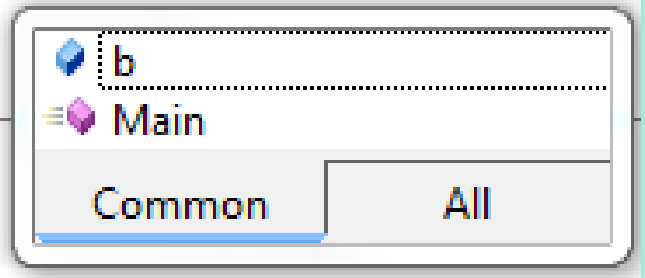
Biến và hằng

- Ví dụ. Biến có phạm vi private và public

Khi đó nếu bạn ở một Module khác mà truy cập đến module1 thì ta chỉ truy cập được đến biến b mà không thể truy cập được biến a

```
Module Module1
    Private a As Integer
    Public b As Integer
    Sub Main()
        a = 100
        b = 200
    End Sub
End Module
```

```
Module Module2

    Public Sub thutuc()
        module1.
    End Sub
End Module
```



Biến và hằng

- Hằng là một vùng nhớ dùng để lưu giữ giá trị, giá trị này sẽ không thay đổi trong suốt thời gian tồn tại hằng.
- Cú pháp khai báo hằng:

Ví dụ . Khai báo hằng số PI, hằng số MAX

const Tên_hằng as Kiểu_dữ_liệu= giá_trị

```
const Pi as double=3.1415
const Max as integer=200
```

Biến và hằng

- **Khai báo biến sử dụng các ký tự hậu tố**

➢ Có thể khai báo biến bằng cách thêm vào sau tên biến một ký tự (hậu tố) xác định kiểu dữ liệu của biến.

➢ Ví dụ: ***Dim x%*** tương ứng với lệnh khai báo ***Dim x As Integer***

Các ký tự hậu tố được chỉ ra trong bảng dưới đây:

Kiểu dữ liệu	Ký tự	Ví dụ
Decimal	@	`Dim decValue@ = 132.24`
Double	#	`Dim dblValue# = .0000001327`
Integer	%	`Dim iCount% = 100`
Long	&	`Dim lLimit& = 1000000`
Single	!	`Dim sngValue! = 3.1417`
String	$	`Dim strInput$ = ""`

CÁC TOÁN TỬ

- Toán tử là ký hiệu chỉ ra phép toán nào được thực hiện trên các toán hạng (có thể là một hoặc hai toán hạng).
- **Toán tử toán học**

Ký hiệu	Mô tả
+	(cộng)
-	(trừ)
*	(nhân)
/	(chia)
\	(chia lấy phần nguyên)
Mod	chia lấy phần dư của số nguyên
^	(lũy thừa)

CÁC TOÁN TỬ

- **Toán tử nối chuỗi**

Toán tử chỉ dành cho toán hạng kiểu String với hai toán tử là & (ampersand) và + (cộng). Kết quả là một trị String gồm các ký tự của toán hạng thứ nhất tiếp theo sau là các ký tự của toán hạng thứ hai.

- **Toán tử gán**

Ký hiệu	Mô tả
=	Gán toán hạng thứ hai cho toán hạng thứ nhất
+=	Cộng hoặc nối chuỗi toán hạng sau vào toán hạng đầu và gán kết quả cho toán hạng đầu
-=	Trừ toán hạng sau khỏi toán hạng đầu và gán hiệu cho toán hạng đầu
*=	Trừ toán hạng sau khỏi toán hạng đầu và gán hiệu cho toán hạng đầu
/=	Chia toán hạng đầu cho toán hạng sau và gán thương cho toán hạng đầu
\=	Thực hiện phép toán \ giữa toán hạng đầu và toán hạng sau và gán kết quả cho toán hạng đầu
^=	Tính lũy thừa toán hạng đầu với số mũ là toán hạng sau và gán kết quả cho toán hạng đầu
&=	Nối chuỗi toán hạng sau vào toán hạng đầu và gán kết quả cho toán hạng đầu

CÁC TOÁN TỬ

- **Toán tử so sánh**

Ký hiệu	Mô tả
=	Bằng
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
>	Lớn hơn
<	Nhỏ hơn
<>	Khác
TypeOf … Is …	So sánh kiểu của biến kiểu tham chiếu thứ nhất có trùng kiểu trên toán hạng thứ hai, nếu trùng trả về True, ngược lại False
Is	Toán tử dành cho toán hạng kiểu tham chiếu, trả về True nếu hai toán hạng cùng tham chiếu đến một đối tượng, ngược lại là False)
Like	Toán tử dành cho toán hạng kiểu String, trả về True nếu toán hạng thứ nhất trùng với mẫu (pattern) của toán hạng thứ hai, ngược lại là False.

CÁC TOÁN TỬ

- **Toán tử luận lý và bitwise**

  Toán tử luận lý trả về giá trị True, False

Ký hiệu	Mô tả
Not	Trả về giá trị ngược lại của toán hạng
And	Trả về True (1) khi và chỉ khi hai toán hạng cùng là True (1)
AndAlso	Trả về giá trị như And nhưng khi toán hạng thứ nhất là False (0) sẽ không kiểm tra toán hạng thứ hai và trả về False
Or	Trả về False (0) khi và chỉ khi hai toán hạng cùng là False (0)
OrElse	Trả về giá trị như Or nhưng khi toán hạng thứ nhất là True (1) sẽ không kiểm tra toán hạng thứ hai và trả về True (1)
Xor	Trả về True (1) khi và chỉ khi có 1 toán hạng là True (1)
Not	Trả về giá trị ngược lại của toán hạng

Các cấu trúc điều khiển

- Lệnh IF
- Lệnh Select case
- Lệnh For…Next
- Lệnh For… each
- Lệnh While
- Lệnh Do….Loop
- Lệnh With… end With

Biểu thức logic

- Tính toán và xử lý các bài toán thực tế không thể thiếu biểu thức logic, giá trị biểu thức logic thường làm điều kiện để thực hiện các công việc cụ thể, biểu thức logic là biểu thức trả về giá trị True hoặc False
- Các toán tử so sánh dùng trong biểu thức logic

Toán tử so sánh	Ý nghĩa
=	Bằng
<>	Khác
<	Nhỏ hơn
>	Lớn hơn
<=	Nhỏ hơn hoặc bằng
>=	Lớn hơn hoặc bằng

Lệnh IF

Dạng 1

- Cú pháp:

If biểu_thức Then Câu_lệnh

- Hoạt động:

Nếu biểu_thức nhận giá trị đúng (True) thì thực hiện Câu_lệnh.

Ví dụ. Cấu trúc if dạng 1

```
Module Module1
    Private a As Integer
    Public b As Integer
    Private Sub Cautrucif_1()
        Console.WriteLine("Nhap vao gia tri a")
        a = Console.ReadLine()
If (a > 0) Then Console.WriteLine("Ban vua nhap
mot so nguyen duong")
    End Sub
    Sub Main()
        Cautrucif_1()
        Console.ReadLine()
    End Sub
End Module
```

Nếu bạn nhập vào số 10 thì sẽ thấy xuất hiện dòng chữ "**Ban vua nhap mot so nguyen duong**" còn nếu nhập số < hoặc = 0 thì không thấy xuất hiện gì.

Lệnh IF

Dạng 2

- Cú pháp:

If biểu_thức Then
Câu_lệnh_với_biểu_thức_đúng
Else
Câu_lệnh_với_biểu_thức_sai
End if

- Hoạt động:

✓ Nếu biểu_thức nhận giá trị đúng (True) thì thực hiện Câu_lệnh_với_biểu_thức_đúng

✓ Ngược lại (biểu thức nhận giá trị sai) thì thực hiện Câu_lệnh_với_biểu_thức_sai

Ví dụ

```
Module Module1
    Private a As Integer
    Public b As Integer
    Private Sub Cautrucif_2()
 Console.WriteLine("Nhap vao gia tri a")
        a = Console.ReadLine()
        If (a > 0) Then
 Console.WriteLine("Ban vua nhap mot so
nguyen duong")
        Else
  Console.WriteLine("Ban vua nhap mot so
nguyen am hoac bang khong")
        End If
    End Sub
   Sub Main()
        Cautrucif_2()
        Console.ReadLine()
    End Sub
End Module
```

Lệnh IF

- Nếu nhập vào số >0 thì dòng chữ xuất hiện “Ban vua nhap mot so nguyen duong”
- Nếu nhập số < hoặc = 0 thì xuất hiện dòng chữ “Ban vua nhap mot so nguyen am hoac bang khong”.
- Lưu ý: Cấu trúc if dạng 2 có thể khuyết toàn bộ phần

❑ Lưu ý: Cấu trúc if dạng 2 có thể khuyết toàn bộ phần

Else

Câu_lệnh_với_biểu_thức_sai

Chỉ còn lại

If biểu_thức Then

Câu_lệnh_với_biểu_thức_đúng

End if

- Khi đó nó hoạt động như cấu trúc if dạng 1
- Ngoài ra lệnh if còn có cấu trúc dạng lồng nhau

Lệnh IF

Ví dụ. Câu lệnh if với biểu thức điều kiện chứa toán tử and

```
Module Module1
  Private Sub Cautrucif_and()
     Dim username As String
     Dim password As String
     Console.WriteLine("Hay nhap vao tai khoan cua ban:")
     username = Console.ReadLine()
     Console.WriteLine("Hay nhap vao matkhau cua ban:")
     password = Console.ReadLine()
     If (username = "HOCVISUALBASIC") And (password = "123") Then
        Console.WriteLine(" Ban dang nhap thanh cong ")
     Else
        Console.WriteLine(" Ban nhap sai ten tai khoan hoac mat khau")
      End If
  End Sub
    Sub Main()
       Cautrucif_and()
       Console.ReadLine()
    End Sub
End Module
```

Lệnh Select case

- Cho phép lựa chọn trường hợp và rẽ nhánh nhanh hiệu quả, dễ hiểu hơn if trong trường hợp có nhiều trường hợp.

- Cú pháp

Select Case BiểuThức

Case DanhSáchGiáTrị

Các CâuLệnh

[Case Else

CácCâuLệnh]

End Select

- Hoạt động:
- BiểuThức sẽ được tính toán và kết quả nếu khớp với các hằng hoặc các biểu thức trong DanhSáchGiáTrị của câu lệnh Case thì CácCâuLệnh sau nó được thực hiện. Nếu không khớp với bất kỳ DanhSáchGiáTrị thì CácCâuLệnh sau Case Else sẽ thực hiện

Lệnh Select case

- Cho phép thực thi các câu lệnh trên cơ sở kết quả của biểu thức.
- Câu lệnh IF .. Then khác câu lệnh Select .. Case
  - IF .. Then tính toán giá trị biểu thức trong mỗi câu lệnh, Select .. Case chỉ tính toán một biểu thức.
  - Biểu thức trong câu lệnh Select .. Case không trả về giá trị kiểu Boolean.

Lệnh Select case

Ví dụ Cấu trúc Select

```
Select Case thang
        Case 1 To 3
           Console.WriteLine(" Ban sinh vao mua xuan")
        Case 4 To 6
           Console.WriteLine(" Ban sinh vao mua ha")
        Case 7 To 9
           Console.WriteLine(" Ban sinh vao mua thu")
        Case 10 To 12
           Console.WriteLine(" Ban sinh vao mua dong")
     Case Else
              Console.WriteLine(" Ban nhap sai thang sinh")
        End Select
      End Sub
```

Khi sử dụng cấu trúc select case có thể sử dụng các toán tử so sánh (<. >, =, >=, <=) trong cấu trúc của nó. Để sử dụng các toán tử so sánh này, ta cần thêm các tử khóa IS hoặc To

Lệnh Select case

Ví dụ Cấu trúc Select với IS

```
Select Case QtyOrdered
   Case Is < 10
      CreditPoints = 10
   Case Is > 20
      CreditPoints = 25
   Case Is <= 20
      CreditPoints = 15
   Case Else
      MessageBox.Show(“Không hợp lệ”)
End Select
```

Lệnh Select case

Ví dụ nhiều hơn một giá trị với Case

```
Select Case Number
  Case 2, 4, 6, 8,10
     MessageBox.Show("Even number")
  Case 1,3,5,7,9
     MessageBox.Show("Odd number")
  Case Else
     MessageBox.Show("Number out of
  range..")
End Select
```

Lệnh For…Next

- Cho phép bạn thực thi lặp lại một nhóm hay nhiều lệnh với một số lần nhất định

- Cú pháp:

For biến=giá_trị_đầu To giá_trị_cuoi [Step bước_nhảy]

Câu Lệnh hoặc các câu lệnh cần lặp

Next [biến]

Lưu ý: Vòng lặp For có thể khuyết Step bước_nhảy **khi đó mặc định bước_nhảy mang giá trị 1**

- ***Hoạt động:***

B1: *Đầ*u biến được gán bằng **giá_trị_đầu**.

B2: Nếu **bước_nhảy**>0 thì sang bước 3 ngược lại sang bước 6

B3: Kiểm tra **giá_trị_đầu** <= **giá_trị_cuối** nếu đúng sang bước 4 ngược lại sang bước 5

B4: Thực hiện **Câu lệnh hoặc các câu lệnh cần lặp** quay lại B2

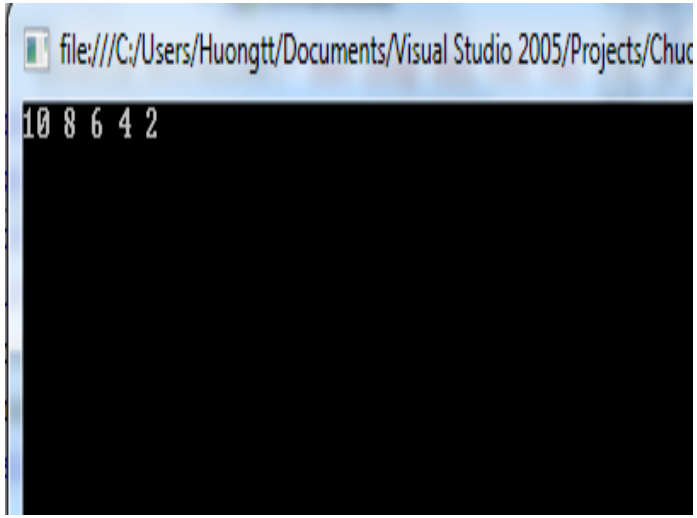
B5: Thoát khởi vòng lặp

B6: Kiểm tra **giá_trị_đầu>= giá_trị_cuối** nếu đúng sang bước 4 ngược lại sang bước 5

Lệnh For…Next

Ví dụ

```
Module Module1
   Private Sub Cautrucfor2()
      Dim i As Integer
      For i = 10 To 1 Step -2
         Console.Write(i & " ")
      Next i
   End Sub
   Sub Main()
      Cautrucfor2()
      Console.ReadLine()
   End Sub
End Module
```



Vòng lặp for lùi với bước nhảy âm

Lệnh For…Next

Ví dụ Sử dụng vòng lặp For in ra màn hình các số từ 1 đến 10

```
Module Module1
   Private Sub Cautrucfor1()
      Dim i As Integer
      For i = 1 To 10 Step 1
         Console.Write(i & " ")
      Next i
   End Sub
   Sub Main()
      Cautrucfor1()
      Console.ReadLine()
   End Sub
End Module
```

```
Module Module1
   Private Sub Cautrucfor1()
      Dim i As Integer
      For i = 1 To 10
         Console.Write(i & " ")
      Next
   End Sub
   Sub Main()
      Cautrucfor1()
      Console.ReadLine()
   End Sub
End Module
```

Lệnh For…Next

- ***Câu lệnh thoát khỏi vòng lặp for***

  Để thoát khỏi vòng lặp For ở vị trí bất kỳ ta sử dụng câu lệnh Exit For

  Xét lại ví dụ: Vòng lặp for lùi với bước nhảy âm

```
Module Module1
   Private Sub Cautrucfor2()
       Dim i As Integer
       For i = 10 To 1 Step -2
           Console.Write(i & " ")
           Exit For
       Next i
    End Sub
    Sub Main()
       Cautrucfor2()
       Console.ReadLine()
    End Sub
End Module
```

Kết quả hiển thị trên màn hình chỉ là số 10 duy nhất.

Lệnh Do….Loop

- Cú pháp 1:

  Do while biểu_thức

  Câu lệnh hoặc các câu lệnh lặp

  Loop

- Hoạt động:
  - ✓ *B1:* kiểm tra biểu_thức, nếu biểu thức đúng thì sang bước 2 ngược lại sang bước 3
  - ✓ *B2*: thực hiện Câu lệnh hoặc các câu lệnh rồi quay lại bước 1
  - ✓ *B3*: Thoát khỏi vòng lặp

- *Nhận xét:* Với vòng lặp dạng này thì nếu biểu_thức sai ngay từ đầu thì vòng lặp sẽ không thực hiện bất cứ câu lệnh hay các câu lệnh lặp một lần nào cả.

Lệnh While

- Cú pháp:

While biểu_thức

Câu lệnh hoặc các câu lệnh lặp

Wend

- Hoạt động:

*B1: K*iểm tra biểu_thức, nếu biểu thức đúng thì sang bước 2 ngược lại sang bước 3

B2: Thực hiện Câu lệnh hoặc các câu lệnh rồi quay lại bước 1

B3: Thoát khỏi vòng lặp

- ***Câu lệnh thoát khỏi vòng lặp không xác định ở vị trí bất kỳ***

Tương tự như vòng lặp For vòng lặp không xác định cũng cung cấp lệnh **Exit do** để thoát khỏi vòng lặp ở vị trí bất kỳ. Khi gặp câu lệnh này máy sẽ ngay tức khắc thoát khỏi vòng lặp mặc dù điều kiện lặp vẫn đúng.

Lệnh Do….Loop

- **Cú pháp 2:**

Do

Câu lệnh hoặc các câu lệnh lặp

Loop while biểu_thức

- **Hoạt động:**

B1: Thực hiện Câu lệnh hoặc các câu lệnh rồi sang bước 2

*B2: K*iểm tra biểu_thức, nếu biểu thức đúng thì quay lại bước 1 ngược lại sang bước 3

B3: Thoát khỏi vòng lặp

- ***Nhận xét***: Với vòng lặp dạng này thì nếu **biểu_thức** sai ngay từ đầu thì vòng lặp sẽ thực hiện một lần **câu lệnh hay các câu lệnh lặp**.

Lệnh Do….Loop

Ví dụ: Sử dụng cấu trúc lặp không xác định để in các số từ 1 đến 10 ra màn hình.

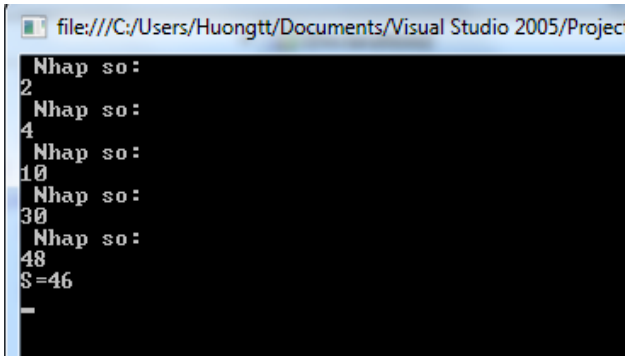
```
Module Module1
   Private Sub Cautrucdowhile1()
       Dim i As Integer
       i = 1
       Do While (i <= 10)
           Console.Write(i & " ")
           i = i + 1
       Loop
    End Sub
    Sub Main()
       Cautrucdowhile1()
       Console.ReadLine()
    End Sub
End Module
```

```
Module Module1
   Private Sub Cautrucdowhile2()
       Dim i As Integer
       i = 1
       Do
           Console.Write(i & " ")
           i = i + 1
       Loop While (i <= 10)
    End Sub
    Sub Main()
       Cautrucdowhile2()
       Console.ReadLine()
    End Sub
End Module
```

Lệnh Do….Loop

```
Module Module1
    Private Sub Cautrucdowhile3()
        Dim s As Integer
        Dim so As Integer
        s = 0
        so = 0
        Do While (s + so < 50)
            s = s + so
            Console.WriteLine(" Nhap so:")
            so = Console.ReadLine()
        Loop
        Console.WriteLine("S=" & s)
    End Sub
```

```
    Sub Main()
        Cautrucdowhile3()
        Console.ReadLine()
    End Sub
End Module
```



Hàm và thủ tục

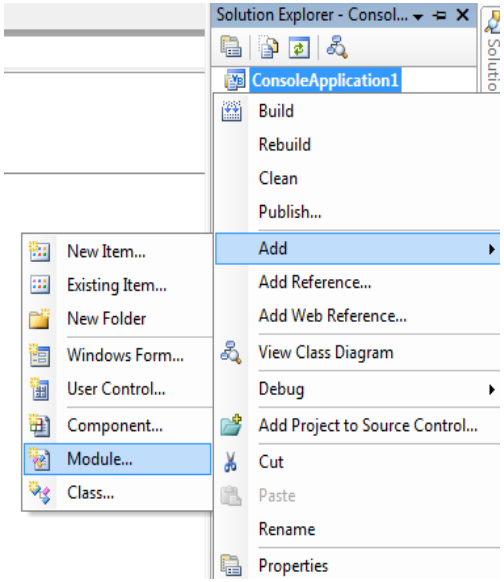
- ***Module***

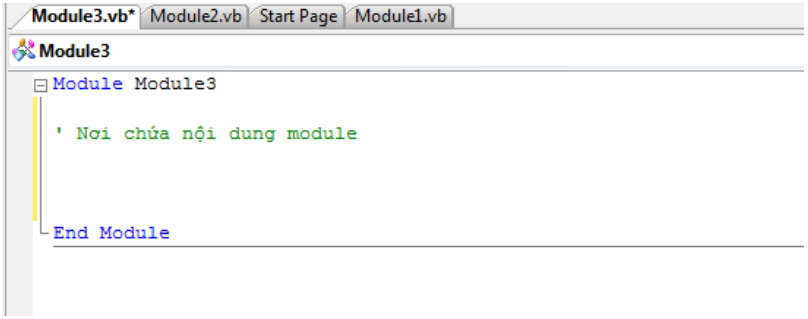
Để chương trình trở nên rõ ràng, sáng sủa thì việc viết chương trình thành các hàm hay thủ tục là việc rất cần thiết.

Module là nơi chứa các biến, hàm, thủ tục có thể triệu gọi từ bất cứ nơi nào trong chương trình. Module là file có phần mở rộng là *.vb chỉ chứa các mã lệnh.

- ***Tạo một Module mới trong Visual Basic.Net***

Tại cửa sổ Solution Explorer kích phải chuột vào tên dự án chọn Add





Thủ tục

- *Cú pháp khai báo*

Public|Private] Sub Tên_thủ_tục ([Tham_đối])

‘ Nội dung thủ tục

End Sub

- Từ khóa Public hoặc Private có thể có hoặc không nếu không có mặc định là Private.
- Public: Thủ tục có thể sử dụng trong phạm vi toàn dự án (có thể sự dụng tại các module khác)
- Private: Thủ tục chỉ có phạm vi trong module hiện tại.
- Tên_thủ_tục: đặt theo quy tắc đặt tên biến, hằng…
- Tham_đối: có thể có hoặc không có, nếu có nhều hơn một tham số thì ngăn cách bởi dấu phẩy. Cách khai báo tham số:

Tên_tham_số As Kiểu_dữ_liệu

Thủ tục

Ví dụ. Xây dựng thủ tục truyền vào tên của một người nào đó, xuất hiện lời chào tương ứng với tên người đó.

```
Module Module1
Private Sub Loichao(ByVal ten As String)
        Console.WriteLine("Xin chao ban " & ten)
        Console.WriteLine("Ban co khoe khong?")
    End Sub
    Sub Main()
        Loichao("Nguyen Hoang Ha")
        Console.ReadLine()
    End Sub
End Module
```

- *Lời gọi thủ tục*

Tên_thủ_tục ([Các tham số])

Như ở ví dụ trên lời gọi thủ tục là **Loichao("Nguyen Hoang Ha")**

- *Thoát khỏi thủ tục tại thời điểm bất kỳ*

Câu lệnh **Exit sub** cho phép ta thoát khỏi thủ tục ở thời điểm bất kỳ dù đang trong câu lệnh if, select case, for, do while…

HÀM

- Cú pháp khai báo

[Public|Private] Function Tên_hàm ([Tham_đối]) As Kiểu_dữ_liệu

‘ Nội dung hàm

Tên_hàm=giá_trị

End Function

- Từ khóa Public hoặc Private có thể có hoặc không nếu không có mặc định là Private.
  - ✓ Public: Hàm có thể sử dụng trong phạm vi toàn dự án (có thể sự dụng tại các module khác)
  - ✓ Private: Hàm chỉ có phạm vi trong module hiện tại.
  - ✓ Tên_hàm: đặt theo quy tắc đặt tên biến, hằng…
  - ✓ Tham_đối: có thể có hoặc không có, nếu có nhều hơn một tham số thì ngăn cách bởi dấu phẩy. Cách khai báo tham số:

Tên_tham_số As Kiểu_dữ_liệu

HÀM

Ví dụ: Xây dựng hàm tính giai thừa của số nguyên n được truyền vào

```
Module Module1
  Public Function Giaithua(ByVal n As Integer) As Single
      Dim s As Single
      Dim i As Integer
      s = 1
      For i = 1 To n
         s = s * i
      Next
      Giaithua = s
   End Function
   Sub Main()
      Dim gt As Single
      gt = Giaithua(5)
      Console.WriteLine("5!=" & gt)
      Console.ReadLine()
    End Sub
End Module
```

- *Lời gọi hàm*

Biến= Tên_hàm ([Các tham số])

Như ở ví dụ trên lời gọi thủ tục là

gt = Giaithua(5)

- *Thoát khỏi hàm tại thời điểm bất kỳ*

Câu lệnh **Exit sub** cho phép ta thoát khỏi hàm ở thời điểm bất kỳ dù đang trong câu lệnh if, select case, for, do while…

Truyền dữ liệu theo tham biến và tham trị

- Việc truyền dữ liệu theo tham biến hay tham trị phụ thuộc vào việc bạn khai báo tham số dạng tham biến hay tham trị
- Nếu truyền theo tham biến bạn sử dụng từ khóa Byval trước khi khai báo tham số, còn truyền theo tham trị bạn sử dụng từ khóa Byref.

```
Module Module1
Public Sub Hoanvi1(ByVal a As Integer, ByVal b As Integer)
    Dim tg As Integer
    tg = a
    a = b
    b = tg
End Sub
```

```
Sub Main()
    Dim m As Integer
    Dim n As Integer
    m = 9
    n = 10
    Hoanvi1(m, n)
Console.WriteLine("Gia tri m la:" & m)
 Console.WriteLine("Gia tri n la:" & n)
    Console.ReadLine()
  End Sub
End Module
```

Sau khi chạy đoạn chương trình trên giá trị in ra của m vẫn là 9 và n vẫn là 10

HÀM

Truyền dữ liệu theo tham biến và tham trị

```
Module Module1
  Public Sub Hoanvi1(Byref a As Integer, Byref b As Integer)
      Dim tg As Integer
      tg = a
      a = b
      b = tg
   End Sub
   Sub Main()
      Dim m As Integer
      Dim n As Integer
      m = 9
      n = 10
      Hoanvi1(m, n)
      Console.WriteLine("Gia tri m la:" & m)
      Console.WriteLine("Gia tri n la:" & n)
      Console.ReadLine()
   End Sub
End Module
```

HÀM

❑ *Phạm vi của biến*

- Các biến được khai báo bên trong thủ tục, hàm, module có phạm vi trong thủ tục module đó
- Tùy thuộc vào việc lựa chọn phạm vi của biến là private, public, dim mà biến đó có thể sử dụng bên ngoài module hay bên ngoài thủ tục
- Các biến được sử dụng bên ngoài module được khai báo với từ khóa publie
- Các biến được khai báo bên trong module sẽ được khai báo với từ khóa private, dim
- Các biến được khai báo bên trong thủ tục, hàm thường được sử dụng với từ khóa dim (có phạm vi private trong thủ tục hay hàm đó)

XỬ LÝ LỖI

Tại mục này các bạn sẽ được tìm hiểu cách xây dựng các khối mã tự xử lý lỗi phát sinh (còn gọi là các ngoại lệ).

Cú pháp Try…Catch

Try

Các phát biểu có thể gây lỗi

Catch

Các phát biểu xử lý nếu có lỗi phát sinh

[Finally

Các phát biểu được gọi ngay cả khi có hay không có lỗi]

End Try

XỬ LÝ LỖI

Ví dụ

```
Module Module1
    Public Sub Xulyloi1()
        Dim a As Integer
        Dim b As Integer
        Dim c As Integer
Console.WriteLine(" Nhap gia tri cho a:")
        a = Console.ReadLine()
Console.WriteLine(" Nhap gia tri cho b:")
        b = Console.ReadLine()
```

```
Try
            c = a / b
        Catch ex As Exception
Console.WriteLine("Loi chia cho so
bang 0")
        End Try
    End Sub
    Sub Main()
        Xulyloi1()
        Console.ReadLine()
    End Sub
End Module
```

TÀI LIỆU THAM KHẢO

1. Kỹ Thuật Lập Trình Ứng Dụng Chuyên Nghiệp Visual Basic .NET, NXB Lao động xã hội, 2004
2. Hướng dẫn lập trình Visual basic.NET, www.vivosoft.com
3. https://www.youtube.com/watch?v=cahAnLyKvu8