



ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT

BÀI GIẢNG

LẬP TRÌNH TRỰC QUAN

Tài liệu lưu hành nội bộ - dành cho sinh viên
Ngành CÔNG NGHỆ THÔNG TIN



Biên soạn:
TS. HOÀNG THỊ MỸ LỆ



Khoa ĐIỆN-ĐIỆN TỬ


LẬP TRÌNH TRỰC QUAN

- **Mã học phần 5505173**
- **Mục tiêu của học phần**
 - + Vận dụng được phương pháp lập trình trực quan trong VB.NET
 - + Thông hiểu được các thành phần cơ bản của giao diện đồ họa
 - + Lập trình cơ sở dữ liệu ADO.NET
 - + Xây dựng được một phần mềm ứng dụng

- **Mô tả tóm tắt về học phần**

Học phần trang bị kiến thức và kỹ năng phát triển ứng dụng bằng kỹ thuật lập trình trực quan, hướng cho sinh viên tiếp cận với môi trường phát triển của Visual studio.net các ứng dụng đồ họa qua giao diện đồ họa, xử lý các sự kiện tương tác người dùng. Phát triển ứng dụng tích hợp cơ sở dữ liệu. Khai thác các tiện ích đóng gói, cài đặt và tạo giao diện trợ giúp.

MỤC LỤC

▪ Mã học phần 5505173.....	2
▪ Mục tiêu của học phần	2
▪ Mô tả tóm tắt về học phần	2
Chương I: GIỚI THIỆU VỀ LẬP TRÌNH TRỰC QUAN-VISUAL BASIC.NET (VB.NET).....	7
I. Giới thiệu	7
II. Giao diện VB.NET	7
1) Khởi động VB.NET	8
2) Thoát khỏi VB.NET	9
III. Các thao tác thường dùng trong VB.NET	9
IV. Các khái niệm thường dùng.....	10
Chương II: CÁC THÀNH PHẦN CƠ BẢN CỦA GIAO DIỆN ĐỒ HOẠ (GUI- Graphical User Interface)	11
I. Điều khiển BUTTON ()	11
II. Điều khiển LABEL, LINK LABEL.....	11
III. Điều khiển TEXTBOX, RICHTEXTBOX.....	12
IV. Điều khiển CHECKBOX, CHECKEDLISTBOX, RADIOBUTTON	13
V. Điều khiển LISTBOX, COMBOBOX, LISTVIEW, TREEVIEW	14
VI. Một số điều khiển đặc biệt.....	19
VI. Điều khiển xây dựng Menu	21
VII. Bài tập.....	21
Chương III: LẬP TRÌNH TRONG VB.NET	22

I. CÁC KIỂU DỮ LIỆU.....	22
1. Kiểu dữ liệu cơ bản.....	22
2. Kiểu con trỏ.....	22
II. MỘT SỐ HÀM THƯỜNG DÙNG	23
1. Hàm chuyển đổi giữa các kiểu dữ liệu.....	23
2. Hàm xử lý số (System.Math)	23
3. Hàm xử lý chuỗi.....	23
4. Các hàm ngày tháng.....	24
5. Hàm kiểm tra điều kiện.....	24
II. CÁC PHÉP TOÁN	25
1. Các phép toán cơ bản.....	25
2. Các phép toán số học viết tắt	25
3. Các phép toán so sánh.....	25
4. Các phép toán logic.....	25
5. Các phép toán trên chuỗi	25
III. KHAI BÁO DỮ LIỆU	25
1. Khai báo biến	25
2. Khai báo hằng	26
3. Khai báo mảng	26
4. Khai báo dữ liệu kiểu bản ghi	27
IV. CHƯƠNG TRÌNH CON.....	28
1. Khái niệm.....	28
2. Khai báo chương trình con.....	28
3. Sử dụng chương trình con (lời gọi).....	29
4. Khai báo các biến hình thức.....	29
5. Module chương trình	29

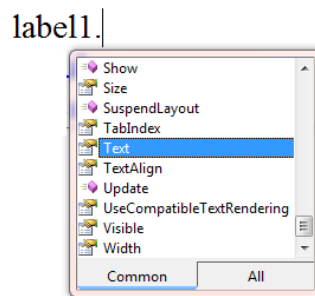
V. HÀM INPUTBOX	30
VI. MSGBOX	30
1. Msgbox có thể như một thủ tục	30
2. Msgbox có thể như một hàm	30
VII. CÁC CẤU TRÚC ĐIỀU KHIỂN.....	31
1. Cấu rẽ nhánh.....	31
2. Cấu trúc lặp.....	32
VII. NHÃN	34
IX. QUI ƯỚC KHI VIẾT CHƯƠNG TRÌNH.....	34
1. Qui ước khi đặt tên	34
2. Lập trình theo cấu trúc.....	34
Chương IV: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG	35
I. PHƯƠNG PHÁP LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG	35
II. LỚP ĐỐI TƯỢNG	35
1. Khái niệm	35
2. Khai báo lớp	35
3. Tạo thuộc tính của lớp	36
III. ĐỐI TƯỢNG	37
1. Khái niệm	37
2. Khai báo đối tượng	37
IV. PHƯƠNG THỨC THIẾT LẬP	37
1. Mục đích.....	37
2. Các đặc điểm của phương thức thiết lập	37
3. Phân loại phương thức thiết lập.....	37
4. Ví dụ	38

V. TÍNH KẾ THỪA TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG	40
1. Khái niệm.....	40
2. Một số chú ý của tính kế thừa.....	40
3. Ví dụ.....	41
VI. ĐỊNH NGHĨA TOÁN TỬ TRÊN LỚP	42
1. Cú pháp	42
2. Ví dụ.....	42
VII. BÀI TẬP	44
Chương V: LẬP TRÌNH CƠ SỞ DỮ LIỆU	45
I. CÁC KHÁI NIỆM LIÊN QUAN ĐẾN CƠ SỞ DỮ LIỆU	45
1. Bộ máy cơ sở dữ liệu là gì	45
2. Thiết kế cơ sở dữ liệu.....	45
3. DataSet là gì.....	45
II/ KẾT NỐI CƠ SỞ DỮ LIỆU VỚI ADO.NET.....	45
1. Thiết lập kết nối CSDL	45
2. Tạo bộ điều phối dữ liệu DataAdapter.....	46
3. Đối tượng điều khiển OleDbDataAdapter	46
4. Tạo đối tượng DataSet	46
5. Trình bày dữ liệu từ CSDL lên Form.....	46
II. LẬP TRÌNH KẾT NỐI CƠ SỞ DỮ LIỆU.....	49
1. Kết nối cơ sở dữ liệu	49
2. Truy vấn dữ liệu từ các bảng trong CSDL.....	50
3. Đọc dữ liệu.....	50
4. Khởi tạo liên kết với dataTable.....	51
TÀI LIỆU THAM KHẢO	64

Chương I: GIỚI THIỆU VỀ LẬP TRÌNH TRỰC QUAN-VISUAL BASIC.NET (VB.NET)

I. LẬP TRÌNH TRỰC QUAN

- Người lập trình có thể nhìn thấy ngay những yếu tố do mình đã thiết kế, không phải đợi đến lúc chạy chương trình như nhiều ngôn ngữ lập trình khác.
- Khi viết một đối tượng, vừa viết dấu chấm để chuẩn bị truy nhập đến thuộc tính hoặc phương thức thì .NET đã liệt kê danh sách các thuộc tính và phương thức đó, ta chỉ việc chọn và nhấn phím *space bar* hay phím *Tab* để đẩy nội dung phương thức đó vào chương trình (không cần phải gõ nội dung phương thức vào).



- Khi viết đúng tên một hàm hay một thuộc tính thì cũng được hiển thị ngay thanh chỉ dẫn đi kèm để nhắc nhở về các đối và giá trị của nó.

```
x1 = system.Math.Round(  
1 of 8 Round (a As Double) As Double  
a: A double-precision floating-point number to be rounded.
```

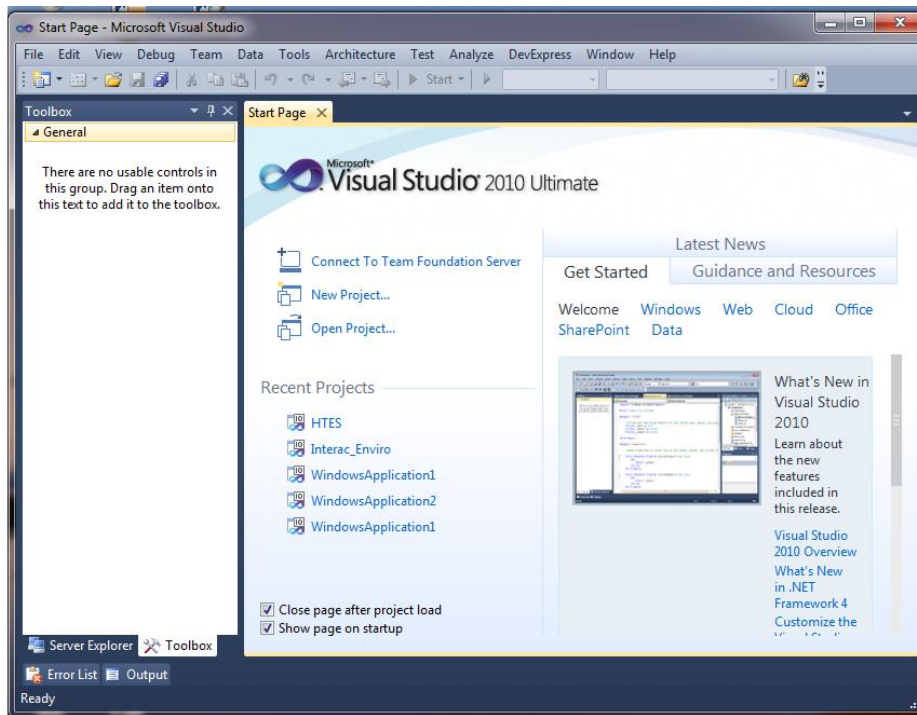
- Được hỗ trợ các phức tạp khi lập trình trên nền Windows. Do đó, khi triển khai xây dựng ứng dụng ta chỉ tập trung vào các vấn đề liên quan đến dự án, công việc hay doanh nghiệp.

II. VISUAL BASIC.NET

1) Giới thiệu

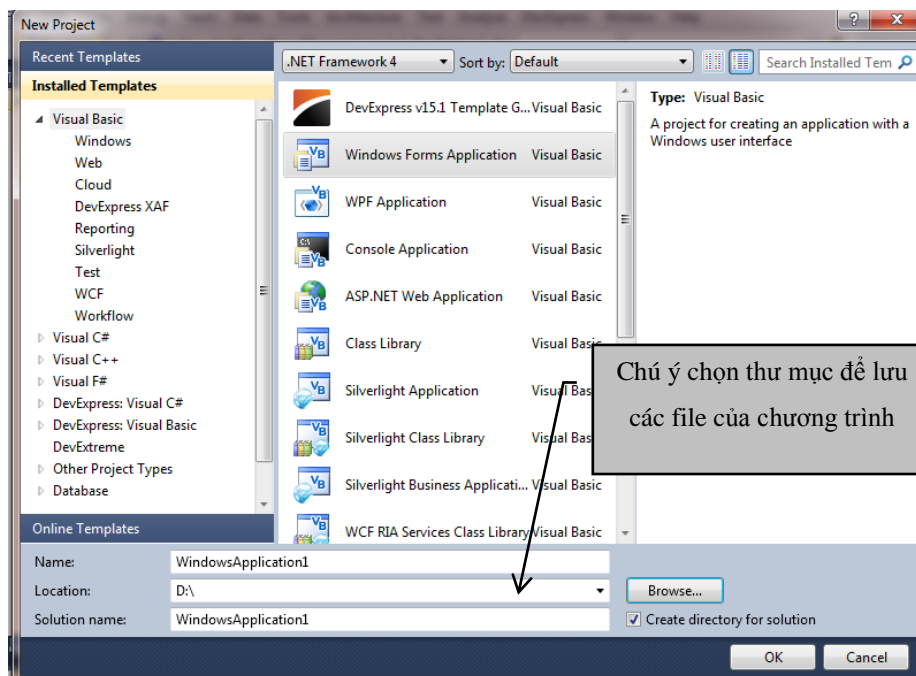
- VB.NET là ngôn ngữ lập trình hướng đối tượng (Object Oriented Programming Language) của Microsoft
- Các chương trình đang triển khai và phát triển được gọi là dự án (Project) hoặc giải pháp (Solution) bởi chúng chứa rất nhiều file và do nhiều thành phần, đối tượng riêng lẻ hợp lại. Một chương trình VB.NET bao gồm một file giải pháp và một file dự án hợp lại. File dự án chứa thông tin đặc biệt liên quan đến một tác vụ lập trình đơn lẻ. File giải pháp lại chứa thông tin về một hay nhiều dự án.

2) Khởi động VB.NET

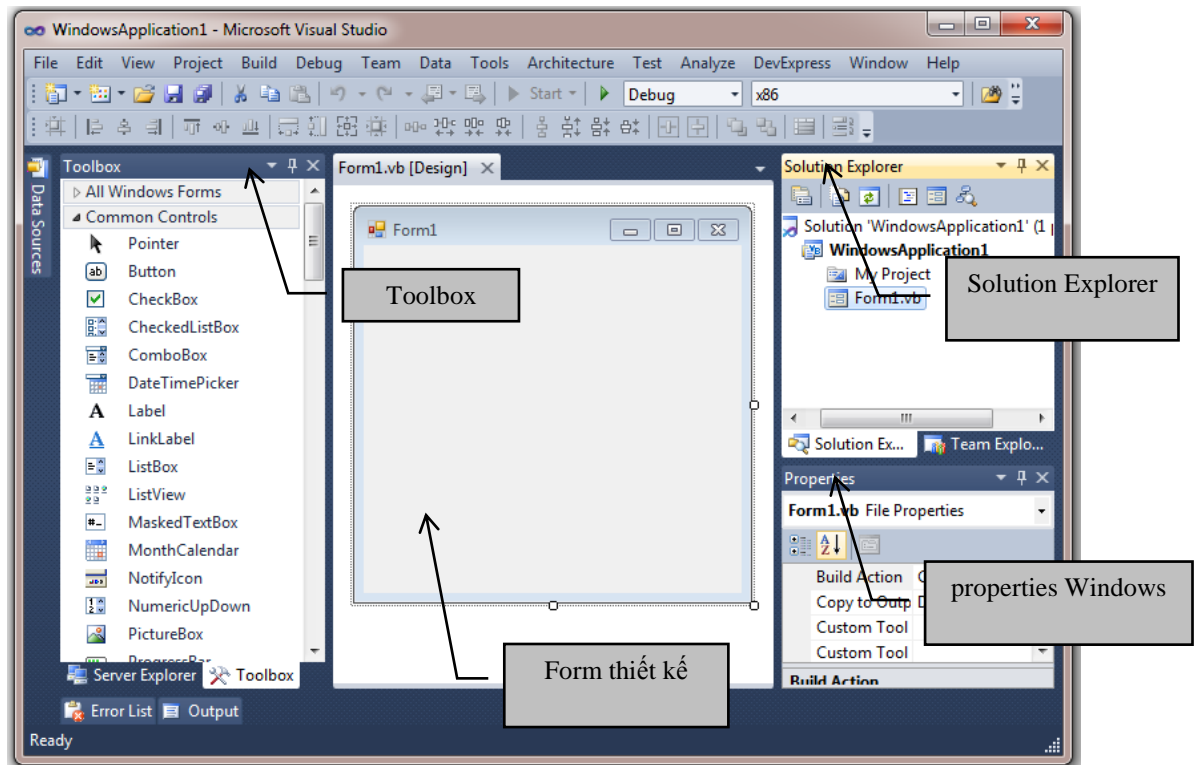


Mở một dự án đã có chọn **Open Project**, xuất hiện hộp hội thoại để chọn tên tập tin cần mở.

Mở một dự án mới ta chọn **New Project**, xuất hiện hộp hội thoại.




Chọn **Windows Application** ta có giao diện:





3) Thoát khỏi VB.NET

- Lưu lại những gì chúng ta làm bằng cách chọn *File / Save all*.
- Chọn *File / Exit* để thoát.

III. MỘT SỐ THAO TÁC TRONG VB.NET

- Lưu project: **File/Save all**
- Lưu một Form: **File/Save Form**
- Lưu một chương trình: Chương trình basic bao gồm các thành phần:
 - + Một tập tin dự án *.SLN
 - + Một hoặc nhiều tập tin Form hay module *.VB
- Thêm một Form: **Project/Add Windows Form**
- Thêm một Module: **Project/Add Module**
- Xác định Form trong một Project chạy đầu tiên: **Project/Properties/chọn form** cần chạy trong **Startup form**
- Chạy chương trình: **Debug/Start Debugging** (nhấn F5 hay biểu tượng )
- Soạn một chương trình mới: **File/New/Project**
- Mở cửa sổ Properties: **View/Properties Windows** (nhấn F4 hay biểu tượng



- Mở cửa sổ Solution Explorer: **View / Solution Explorer** (nhấn CTRL R hay biểu tượng )
- Hiện thị thanh Toolbox: **View / Toolbox** (nhấn vào biểu tượng )
- Bật cửa sổ code: **View / Code** (Nhấn F7 hay nhấp đúp lên đối tượng)
- File X.EXE nằm trong thư mục\bin\Debug

IV. CÁC KHÁI NIỆM THƯỜNG DÙNG

1. Đối tượng (**Object**): là một tập hợp bao gồm chương trình và dữ liệu liên quan với nhau tạo thành một đơn vị xử lý độc lập. Trong VB.NET các đối tượng chính là biểu mẫu (form) và các công cụ điều khiển (control)
2. Hộp điều khiển (**Control Box**): là một đối tượng đặt trên Form, mỗi hộp điều khiển sẽ ứng với một chức năng nào đó sẽ được thực hiện.
3. Phương thức (**Method**): mỗi đối tượng có thể có nhiều dãy hoạt động, các hoạt động này gọi là phương thức của đối tượng, theo cú pháp:

<tên đối tượng>.<tên phương thức>[tham số]

Ví dụ: Form1.show (hiển thị Form có tên là Form1)

4. Thủ tục tình huống (**Event Procedure**): là một dãy các chỉ thị lệnh và được tự động thực hiện khi xảy ra tình huống tác động lên đối tượng.

(Mỗi đối tượng có nhiều thuộc tính. Mỗi đối tượng có các thủ tục tình huống và phương thức riêng)

Chương II: CÁC THÀNH PHẦN CƠ BẢN CỦA GIAO DIỆN ĐỒ HOẠ

(GUI-Graphical User Interface)


I. ĐIỀU KHIỂN BUTTON (Button)

Là nút lệnh cho phép người sử dụng (NSD) kích hoạt, ngắt hoặc kết thúc chương trình.
Các thuộc tính:

- Name: đặt tên cho nút lệnh (do NSD đặt nếu không muốn đặt lại thì lấy tên được đặt sẵn).
- Enabled: nút lệnh có hiệu lực hay không có hiệu lực
Enabled = true (nút lệnh có hiệu lực)
Enabled = false (nút lệnh không có hiệu lực)
- Text: nội dung hiển thị trên nút lệnh.
- Visible: hiển thị hay không hiển thị nút lệnh trên Form
Visible = True (hiển thị); = False (không hiển thị)


Muốn viết chương trình cho sự kiện nút lệnh: nhấp đúp vào nút lệnh.

II. ĐIỀU KHIỂN LABEL, LINK LABEL

1. **Label** ( Label): dùng để trình bày thông tin chuỗi trên Form hay trên Report

Các thuộc tính:

- Name: đặt tên cho nhãn
- Text: nội dung hiển thị trên nhãn.
- Font, Fore Color, BackColor: tạo phong chữ, màu sắc cho nhãn.

2. **Link Label** ( LinkLabel): dùng để mở trình duyệt khi được kích chuột


Các thuộc tính:

- Name: đặt tên cho link label
- Text: nội dung hiển thị LinkLabel.

Muốn liên kết được với trình duyệt, kích đúp chuột vào điều khiển LinkLabel và viết đoạn lệnh sau:

System.Diagnostics.Process.Start("địa chỉ URL")


III. ĐIỀU KHIỂN TEXTBOX, RICHTEXTBOX

1. **Textbox** ( **TextBox**): dùng để tạo giao diện cho phép NSD nhập dữ liệu, có thể dùng làm cái chứa dữ liệu trong các trường kể cả trường memo của cơ sở dữ liệu (CSDL)

Các thuộc tính:

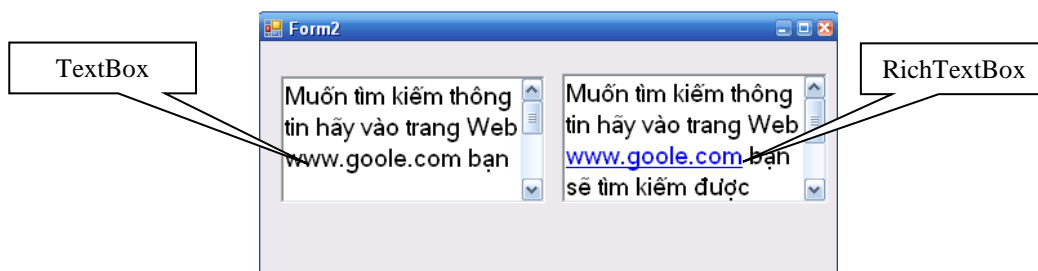
- Name: đặt tên cho text box
- Text: nội dung hiển thị bên trong hộp text box (ta có thể xóa để hộp trống)
- Multi Line = False trong hộp chỉ chứa 1 dòng; = True chứa được nhiều dòng
- Font, Fore Color, BackColor: tạo phong chữ, màu sắc cho text box.
- Maxlength: qui định chiều dài tối đa được nhập cho text box.

(Giá trị nhận được trên text box nằm trong biến có tên là **têntextbox.Text** và kiểu dữ liệu luôn là kiểu String, muốn chuyển sang kiểu dữ liệu khác ta phải dùng các hàm biến đổi để chuyển đổi sang các kiểu dữ liệu khác)

2. **RichTextBox** ( **RichTextBox**): dùng để nhập đoạn văn bản với nhiều định dạng khác nhau.

Các thuộc tính:

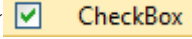
- AcceptsTab: cho phép sử dụng phím Tab khi trình bày văn bản.
- MaxLength: dùng để khai báo chiều dài lớn nhất có thể nhập.
- MultiLine: cho phép nhập hay trình bày dữ liệu nhiều hàng.
- ReadOnly: không cho thay đổi dữ liệu trên RichTextBox
- DetectURLs: cho phép dò tìm địa chỉ URL để NSD kích hoạt.



Muốn liên kết được với trình duyệt, kích đúp chuột vào điều khiển RichTextBox và viết đoạn mã lệnh sau:

```
Private Sub RichTextBox1_LinkClicked(ByVal sender As System.Object, ByVal e
    As System.Windows.Forms.LinkClickedEventArgs)
    System.Diagnostics.Process.Start("Iexplore.exe", e.LinkText)
End Sub.
```

IV. ĐIỀU KHIỂN CHECKBOX, CHECKEDLISTBOX, RADIOBUTTON

1. **CheckBox** (): dùng để tạo giao diện cho phép chọn hay không chọn một thông tin nào đó.

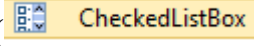
Các thuộc tính:

- Name: đặt tên cho CheckBox
- Text: dòng chú giải bên cạnh điều khiển
- Checked = true: được chọn
= false: không được chọn

Nếu lập trình thì biến có tên:

tênCheckBox.checked có giá trị = true/ false

(Trên 1 Form có thể có nhiều nút vuông, chọn hay không chọn 1 nút thì tùy các nút không loại trừ nhau)

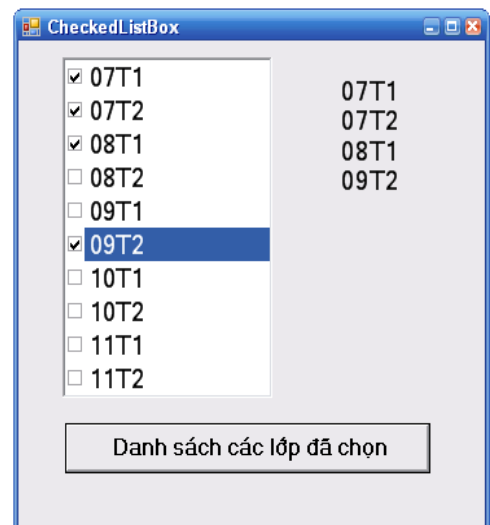
2. **CheckedListBox** ()

Các thuộc tính:

- MultiColumn: cho phép trình bày nhiều cột.
- Itemn: tập các phần tử trong điều khiển, ta dùng phương thức **Add** hay **Remove** để thêm hay loại bỏ phần tử trên điều khiển.

Sau khi chọn vào các CheckBox, ta dùng thuộc tính **SelectedItemn** (có chọn) và **CheckedItemns** (trả về danh sách các phần tử chọn)

Ví dụ: Tạo một CheckedListBox như hình bên:



```
Private Sub CheckedListBox_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
```

```
    Dim i, lop As Integer
```

```
    lop = 7
```

```
    For i = 1 To 5
```

```
        CLB.Items.Add(Microsoft.VisualBasic.Right("0" & lop, 2) & "T1")
```

```
        CLB.Items.Add(Microsoft.VisualBasic.Right("0" & lop, 2) & "T2")
```

```
        lop = lop + 1
```

```
    Next
```

End Sub

```
-----  
  
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles Button1.Click  
  
    If CLB.SelectedItem <> " " Then  
  
        Dim st As String = ""  
  
        For Each ptu As Object In CLB.CheckedItems  
  
            st = st + ptu.ToString() + vbNewLine  
  
        Next  
  
        Label1.Text = st  
  
    End If  
  
End Sub
```

3. **RadioButton**: dùng để tạo giao diện trong một số các tùy chọn loại trừ nhau.

Các thuộc tính:

- Name: đặt tên cho RadioButton
- Checked: nút được chọn hay không.
- Text: dòng chú giải bên cạnh điều khiển




Nếu lập trình thì biến có tên **Tên nút.Checked** có giá trị = True/ False

Để nhóm các điều khiển ta sử dụng điều khiển **GroupBox**

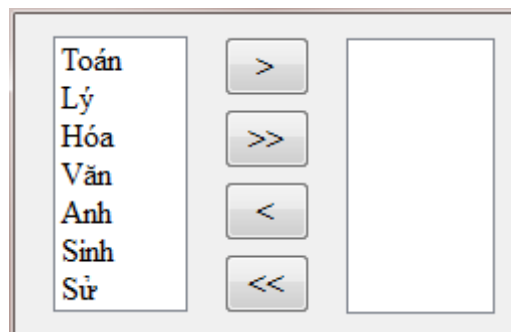
Ví dụ: Tạo một RadioButton như hình trên

V. ĐIỀU KHIỂN LISTBOX, COMBOBOX, LISTVIEW, TREEVIEW

1. **ListBox** ( **ListBox**): là một mảng các chuỗi ký tự để NSD chọn. Trong Listbox chứa một loạt thanh biểu tượng (Item), mỗi thanh là một chuỗi (một hàng trong hộp danh sách).

Các thuộc tính và phương thức:

- Name: đặt tên cho ListBox
- Sorted: dữ liệu được trình bày trong ListBox được sắp xếp tăng dần.
- SelectionMode: cho phép chọn một phần tử (**one**) hay nhiều phần tử (**MultiSimple**).



- MultiColumn: trình bày nhiều cột trong ListBox.
- Items: nhập trực tiếp nội dung vào ListBox
- SelectedItem: trả về giá trị phần tử đã chọn
- SelectedItems: trả về tập các giá trị của phần tử đã chọn

Nếu lập trình:

- Tên_ListBox.Items.Add <xâu kí tự>: Thêm một xâu kí tự vào danh sách
- Tên_ListBox.Items.Remove <xâu kí tự>: Xóa xâu kí tự khỏi danh sách
- Tên_ListBox.Items.Count: Tổng số hàng hiện có trong danh sách


Muốn làm việc với các giá trị của phần tử đã chọn trong SelectedItems:

For Each biến as object in tenListbox.SelectedItems

//biến sẽ lần lượt nhận các giá trị trong tập hợp các giá trị đã chọn

//sử dụng biến để thực hiện công việc nào đó


Next

2. **ComboBox** ( **ComboBox**): phối hợp các đặc tính của TextBox và ListBox, cho phép NSD tự nhập dữ liệu vào hộp hoặc chọn nó có trong danh sách.

Các thuộc tính và phương thức:

Tương tự như ListBox có thêm một số thuộc tính:

- MaxDropDownItems: Cho trình bày số phần tử trong điều khiển mỗi khi NSD chọn nút thả xuống, (mặc định là 8 phần tử)
- DropDownStyle
 - + DropDown: cho phép chọn và nhập vào giá trị (được mặc định)
 - + DropDownList: cho phép chọn giá trị trong điều khiển.
 - + Simple: trình bày như TextBox.

3. **ListView** ( **ListView**): dùng để trình bày dữ liệu dạng danh sách.

Các thuộc tính:

- AllowColumnReorder: thay đổi vị trí của cột trên ListView
- CheckBoxes: mỗi phần tử trong ListView xuất hiện 1 CheckBox bên cạnh.
- Columns: gán tiêu đề cho từng cột trong ListView

Stt	Lớp	Phòng
1	09T1	A309
2	09T2	A310
3	10T1	A310
4	10T2	A311
5	11T1	A311

TênListView.Columns.Add("Tiêu đề cột", Độ rộng, Canh lề tiêu đề trong cột)

- FullRowSelect: cho phép chọn toàn bộ hàng dữ liệu mỗi khi chọn từng phần tử trên hàng.
- Items: cho phép điền dữ liệu từng đối tượng ListViewItem. Trong mỗi đối tượng ListViewItem bao gồm các SubItem.
- MultiSelect: cho phép chọn nhiều hàng dữ liệu tại một thời điểm.
- SubItems: trả về giá trị tương ứng với cột

Nếu thuộc tính MultiSelect = true. Muốn lấy giá trị của các hàng đã chọn, ta khai báo biến kiểu ListViewItem và dùng vòng lặp For Each duyệt từng phần tử đã được chọn trong ListView.

For each biến in tênListViev.SelectedItems

//sử dụng biến để thực hiện công việc nào đó

Next

Để lấy giá trị trên các cột của hàng: tênListView.SubItems(số thứ tự cột).Text

Cột đầu tiên có giá trị là 0.

Ví dụ: Tạo ListView dạng danh sách gồm các cột, stt, lớp, phòng gồm 10 hàng

```
Private Sub Form3_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

    lv.Items.Clear()

    lv.Columns.Add("Stt", 40, HorizontalAlignment.Center)

    lv.Columns.Add("Lớp", 100, HorizontalAlignment.Center)

    lv.Columns.Add("Phòng", 100, HorizontalAlignment.Center)

    lv.View = View.Details

    Dim ptu As ListViewItem

    Dim i, lop As Integer

    lop = 9

    For i = 1 To 10

        ptu = New ListViewItem(i)

        If i Mod 2 <> 0 Then

            ptu.SubItems.Add(Microsoft.VisualBasic.Right("0" & lop, 2) & "T1")

        Else
```



```

        ptu.SubItems.Add(Microsoft.VisualBasic.Right("0" & lop, 2) & "T2")

        lop = lop + 1

    End If

    ptu.SubItems.Add("A3" & Microsoft.VisualBasic.Right("0" & lop, 2))

    lv.Items.Add(ptu)

Next

End Sub

```

Lấy giá trị của các hàng đã chọn.

```

Private Sub Button1_Click_1(sender As System.Object, e As System.EventArgs) Handles
    Button1.Click

    Dim ptu As ListViewItem

    Label1.Text = ""


    For Each ptu In lv.SelectedItems

        Label1.Text = Label1.Text & ptu.SubItems(0).Text & " " & ptu.SubItems(1).Text

    Next

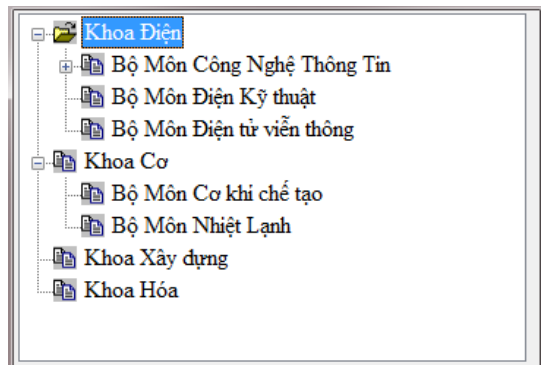
End Sub

```

4. **TreeView** ( **TreeView**): dùng để trình bày dữ liệu dạng phân cấp dạng cây

Các thuộc tính:

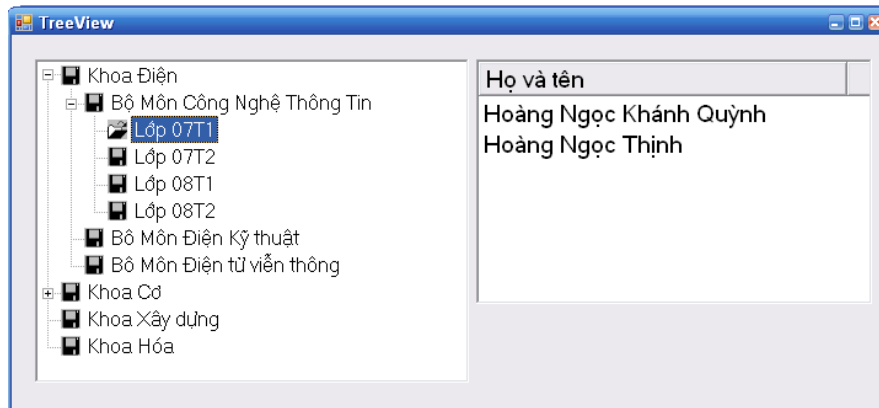
- ImageList: cho phép người chọn ảnh có trong điều khiển ImageList tương ứng với ImageIndex hay SelectedImageIndex.
- Nodes: Khai báo danh sách các Node tương ứng với từng Node.
- ImageIndex: Ảnh tương ứng với từng Node.
- SelectedImageIndex: Ảnh tương ứng với từng Node mỗi khi Node được chọn.
- Sorted: Sắp xếp dữ liệu trình bày trên TreeView.



Trường hợp gán ảnh (biểu tượng) cho từng node, ta sử dụng thuộc tính ImageIndex và SelectImageIndex ứng với từng node theo cú pháp:

TênTreeView.Node(stt node).Node(stt node)....ImageIndex = số thứ tự ảnh trong điều khiển ImageList.

Ví dụ: Xây dựng TreeView như hình và sau khi chọn ta được



```
Private Sub TreeView_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
```

```
    TV.Nodes.Clear()

    TV.ImageList = Me.anh

    TV.ImageIndex = 1

    TV.SelectedIndex = 0

    TV.Nodes.Add("Khoa Điện")

    TV.Nodes(0).Nodes.Add("Bộ Môn Công Nghệ Thông Tin")

    TV.Nodes(0).Nodes.Add("Bộ Môn Điện Kỹ thuật")

    TV.Nodes(0).Nodes.Add("Bộ Môn Điện tử viễn thông")

    TV.Nodes(0).Nodes(0).Nodes.Add("Lớp 07T1")

    TV.Nodes(0).Nodes(0).Nodes.Add("Lớp 07T2")

    TV.Nodes(0).Nodes(0).Nodes.Add("Lớp 08T1")

    TV.Nodes(0).Nodes(0).Nodes.Add("Lớp 08T2")

    TV.Nodes.Add("Khoa Cơ")

    TV.Nodes(1).Nodes.Add("Bộ Môn Cơ khí chế tạo")

    TV.Nodes(1).Nodes.Add("Bộ Môn Nhiệt Lạnh")

    TV.Nodes.Add("Khoa Xây dựng")

    TV.Nodes.Add("Khoa Hóa")

    lv1.Visible = False
```

```
End Sub
```

Nhấp đúp vào điều khiển TreeView ta thủ tục để lấy giá trị của node đang chọn.

```
Private Sub TV_AfterSelect(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.TreeViewEventArgs) Handles TV.AfterSelect
```

```
    If e.Node.Text = "Lớp 07T1" Then
```

```

lv1.Items.Clear()

lv1.Columns.Add("Họ và tên", 300, HorizontalAlignment.Center)

lv1.View = View.Details

Dim ptu As ListViewItem

ptu = New ListViewItem("Hoàng Ngọc Khánh Quỳnh")

lv1.Items.Add(ptu)

ptu = New ListViewItem("Hoàng Ngọc Thịnh")

lv1.Items.Add(ptu)

lv1.Visible = True

Else

lv1.Items.Clear()


lv1.Visible = False

End If

End Sub

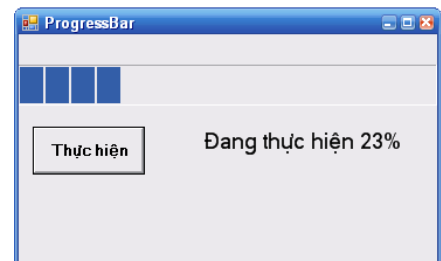
```

VI. MỘT SỐ ĐIỀU KHIỂN ĐẶC BIỆT

1. **ProgressBar** ( **ProgressBar**): dùng để trình bày giá trị đạt được trong giới hạn. Ví dụ như phần trăm công việc đã hoàn thành.

Các thuộc tính:

- Maximum: giá trị giới hạn trên.
- Minimum: giá trị giới hạn dưới.
- Value: trình bày phần trăm thực hiện được.



Ví dụ: Trình bày phần trăm thực hiện công việc như hình trên:

```

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles Button1.Click

    Dim max As Integer = 50000

    Dim i As Integer

    Me.pb.Maximum = max

    Dim st As String

    For i = 0 To max

        pb.Value = i


        st = Math.Floor(((i / max)) * 100).ToString & "%"

        Application.DoEvents()
    
```

```
Label1.Text = "Đang thực hiện " & st
```

```
Next
```

```
End Sub
```

2. **HelpProvider** ( **HelpProvider**): dùng để định nghĩa các tham số gọi đến tập tin .HTML hay tập tin trợ giúp .CHM khi người sử dụng nhấn phím F1 trong lúc làm việc với các điều khiển hoặc Form.

Các thuộc tính:

- HelpNamespace: khai báo tập tin cần hiển thị khi nhấn F1

Cần hiển thị Help khi nhấn F1 ở Form nào thì chọn **Find** ở thuộc tính **HelpNavigator on tênHelpProvider** của Form đó.

3. **Timer** ( **Timer**): dùng để định nghĩa thời gian dựa vào thời gian của hệ thống.

Các thuộc tính:

- Interval: cài đặt thời gian của điều khiển bằng với thời gian thực ta gán thuộc tính này với giá trị là 200. Để lấy giá trị của Timer

DateTime.Now.TimeOfDay.ToString

DateTime.Now.Date.ToString


Để hiển thị đồng hồ và ngày giờ hiện hành, chọn thuộc tính **Enable = True** và nhấp đúp vào biểu tượng nhập vào đoạn mã sau:

```
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.Tick
```

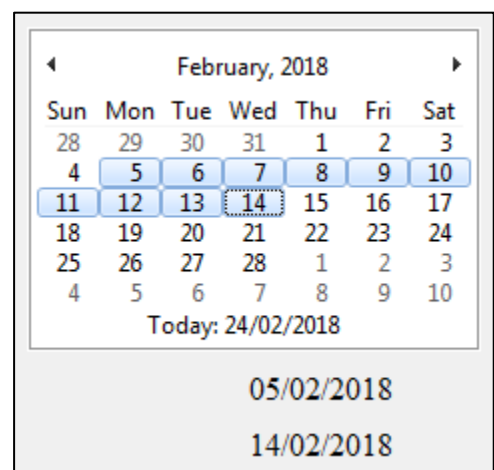
```
Label1.Text = DateTime.Now.Date.ToShortDateString
```

```
Label2.Text = DateTime.Now.TimeOfDay.ToString
```

```
End Sub
```

4. **MonthCalendar** ( **MonthCalendar**): dùng để định dạng kiểu thời gian do NSD nhập hay chọn.

- MaxSelectionCount: Cho phép chọn tối đa 7 ngày (mặc định), ta có thể thay đổi lại số ngày chọn.
- ShowToday: Hiển thị ngày tháng năm hiện hành
- ShowTodayCircle: Ngày hiện hành được vòng



màu đỏ.


Để lấy giá trị được chọn trên lịch ta dùng thuộc tính **Start** và **End** của đối số e trong sự kiện DateChanged

```
Private Sub MonthCalendar1_DateChanged(ByVal sender As System.Object, ByVal e As System.Windows.Forms.DateRangeEventArgs) Handles MonthCalendar1.DateChanged

    Dim bd As DateTime = e.Start.ToLongDateString

    Dim kt As DateTime = e.End.ToLongDateString


End Sub
```

5. **DateTimePicker** ( **DateTimePicker**): Điều khiển này xuất hiện trên Form như một ComboBox. Khi chọn vào mới xuất hiện MonthCalendar cho chọn ngày.

Các thuộc tính


- Format: Định dạng kiểu ngày
- Value: Giá trị của ngày giờ được chọn trên MonthCalendar

VI. ĐIỀU KHIỂN XÂY DỰNG MENU

1. **MenuStrip** ( **MenuStrip**): dùng để xây dựng menu chính trên Form.

Các thuộc tính:

- Enabled: Cho phép hay vô hiệu hóa Menu
- Shortcut: Tạo tổ hợp phím
- Image: Thêm hình ảnh vào trước các mục của menu
- Text: Chuỗi tương ứng với Menu

2. **ToolStrip** ( **ToolStrip**): dùng để tạo ToolBar trên Form.

VII. BÀI TẬP

Tạo Form để minh họa các điều khiển trên.

Chương III: LẬP TRÌNH TRONG VB.NET

I. CÁC KIỂU DỮ LIỆU

1. Kiểu dữ liệu cơ bản

Tên kiểu	Lớp kế thừa	Kích thước	Giá trị mặc định	Phạm vi
Boolean	System.Boolean	4 byte	False	True/False
Byte	System.Byte	1 byte	0	0 ... 255
Char	System.Char	2 byte	Char(0)	0 ... 65535
Date	System.DateTime	8 byte		
Decimal	System.Decimal	12 byte		7.9...E+28
Double	System.Double	8 byte	0.0	1.7...E+308
Integer	System.Int32	4 byte	0	4.2...E+9
Long	System.Int64	8 byte	0	1.8...E+19
Short	System.Int16	2 byte	0	6.5..E+4
Single	System.Single	4 byte	0.0	3.4...E+38
Structure	System.ValueType			

Dữ liệu kiểu ngày:

Ngày - Ngày = khoảng cách đại số giữa 2 ngày

Ngày \pm số nguyên = ngày quá khứ (-) hay ngày tương lai (+)

Hằng kiểu ngày được đặt trong cặp dấu ##

Dữ liệu kiểu structure:

Kiểu dữ liệu Structure được xem như là một đơn thể

Structure không được kế thừa từ một kiểu dữ liệu khác.

Một lớp đối tượng không được kế thừa từ Structure.

2. Kiểu con trỏ

Tên kiểu	Lớp kế thừa	Kích thước	Giá trị mặc định	Phạm vi
Object	System.Object	4 byte		
String	System.String	10 byte		2 tỷ kí tự

II. MỘT SỐ HÀM THƯỜNG DÙNG

1. Hàm chuyển đổi giữa các kiểu dữ liệu

Hàm chuyển đổi	Đổi sang kiểu	Hàm chuyển đổi	Đổi sang kiểu
CBool	Boolean	CLng	Long
CByte	Byte	CSng	Single
CDate	Date	CVar	Variant
CDbl	Double	CInt	Integer

(Giá trị truyền phải hợp lệ, nghĩa là phải thuộc khoảng của kiểu kết quả)

2. Hàm xử lý số (*System.Math*)

Abs(x)	→ Trị tuyệt đối của x
ATan(x)	→ arctg của x (x radian)
Tan(x)	→ tgx (x radian)
Cos(x)	→ cosx (x radian)
Sin(x)	→ sinx (x radian)
Exp(x)	→ e^x
Round(số, [n])	→ làm tròn số thực đến n số lẻ
Fix(số)	→ bỏ phần thập phân để đổi thành số nguyên
Rnd[(số)]	→ tạo một số ngẫu nhiên bất kỳ.

Giá trị trả về là một số thực ≥ 0 và < 1 .

Muốn lấy giá trị ngẫu nhiên từ a đến b ta dùng công thức:

$\text{FIX}(a+b*\text{RND})$

Sgn(số)	→ -1 (nếu số âm); 1 (nếu số dương)
Sqrt(x)	→ \sqrt{x}

3. Hàm xử lý chuỗi

Len(st)	→ độ dài chuỗi st
Mid(st, n, [m])	→ trích trong chuỗi st tại vị trí n lấy m kí tự (nếu không có giá trị của m thì lấy đến hết chuỗi).
Left(st,n)	→ lấy phần bên trái st qua phải n kí tự.

Right(st,n)	→ lấy phần bên phải st qua trái n kí tự.
Ltrim(st)	→ cắt các kí tự trắng bên trái của st.
Rtrim(st)	→ cắt các kí tự trắng bên phải của st.
Trim(st)	→ cắt các kí tự trắng cả hai bên của st.
String(n, ch)	→ lặp lại n lần kí tự ch.
Lcase(st)	→ đổi chuỗi sang chữ thường
Ucase(st)	→ đổi chuỗi sang chữ hoa
Chr(số)	→ kí tự của số đó trong bảng mã ASCII
Asc(kí tự)	→ số của kí tự đó trong bảng mã ASCII
Val(chuỗi)	→ chuyển chuỗi dạng số sang số.
Instr(st1, st2)	→ vị trí của st2 trong st1

4. Các hàm ngày tháng

d : là biến kiểu ngày (gồm có ngày, tháng, năm)

Day(d)	→ ngày của d
Month(d)	→ tháng của d
Year(d)	→ năm của d
Now	→ cho ngày, giờ của hệ thống.
Time	→ cho giờ hệ thống
Date	→ cho ngày hệ thống.
Weekday(d)	→ thứ của ngày trong tuần.

5. Hàm kiểm tra điều kiện

IFF(điềukiện, bt1, bt2) →

trả về giá trị bt1 khi bthức đkiện đúng, còn lại trả về giá trị của bt2

II. CÁC PHÉP TOÁN

1. Các phép toán cơ bản

+, -, *

/ chia (ví dụ: $5/3 = 1.66667$)

\ chia lấy phần nguyên (ví dụ: $5\backslash 3 = 1$)

mod chia lấy phần dư (ví dụ: $5 \bmod 3 = 2$)

^ lũy thừa (ví dụ: $4^2 = 16$)

2. Các phép toán số học viết tắt

+= cộng bằng

-= trừ bằng

*= nhân bằng

/= chia bằng

/= chia bằng (nguyên)

^= lũy thừa bằng

3. Các phép toán so sánh

>

>=

<

<=

=

< >

4. Các phép toán logic

And

Or

Not

Xor

5. Các phép toán trên chuỗi

+ cộng 2 xâu kí tự

& ghép nối 2 xâu kí tự

Sự khác nhau giữa phép + và phép &:

- Phép +: có kiểm tra kiểu dữ liệu, nếu 2 xâu cộng không cùng kiểu dữ liệu thì máy thông báo lỗi.
- Phép &: không kiểm tra dữ liệu khi ghép.

III. KHAI BÁO DỮ LIỆU

1. Khai báo biến

Cách 1: khai báo biến theo kiểu dữ liệu.

DIM <tên biến> AS <kiểu dữ liệu>

Phạm vi: Lệnh dim chỉ có thể xuất hiện ở cấp đơn thể, cấp biểu mẫu hoặc cấp thủ tục.
Nếu ta khai báo dim ở cấp nào thì có hiệu lực trong cấp đó.

Ví dụ: Dim hten as string

 Dim hovaten as string * 25

 Dim ngaysinh as date

Cách 2: Khai báo biến theo kiểu ghi trực tiếp kí hiệu hậu tố vào sau tên biến để định kiểu dữ liệu. **DIM <tên biến><kí tự hậu tố>**

Các kí tự hậu tố:

Kí tự hậu tố	Kiểu dữ liệu
%	Integer
&	Long
!	Single
#	Double
@	Currency
\$	String

Ví dụ: Dim hten\$ → khai báo biến hten kiểu String

 Dim tuoi% → khai báo biến tuoi kiểu Integer

2. Khai báo hằng

CONST <Tên hằng> [AS <kiểu dữ liệu>] = <biểu thức>

Ví dụ: khai báo hằng g có giá trị 9.8

Const g as double = 9.8

hay Const g = 9.8

Ta cũng có thể gán kí tự hậu tố để định kiểu cho hằng.

Ví dụ: Const g# = 3.14

3. Khai báo mảng

Mảng là tập hợp các phần tử có cùng chung 1 kiểu dữ liệu và cùng chung 1 tên.
Các phần tử của mảng phân biệt nhau qua chỉ số của nó trong mảng.

a) Khai báo dữ liệu kiểu mảng

- Mảng 1 chiều: **DIM <tên mảng>(số phần tử) AS <kiểu dữ liệu>**

hoặc: DIM <tên mảng><hệ số kiểu>(số phần tử)

Ví dụ: khai báo một mảng gồm 50 phần tử để chứa số nguyên

Dim a(50) as integer hay Dim a%(50)

→ với khai báo trên ta có 50 phần tử a(0), a(1), ..., a(49)

- Mảng nhiều chiều: **DIM <tên mảng>(số pt1, số pt2, ..., số ptn) AS <kiểu dữ liệu>**

Ví dụ: khai báo một mảng 2 chiều 4x5

Dim b(4,5) as integer

b) Sử dụng mảng

Để làm việc với từng phần tử của mảng ta gọi qua tên mảng và chỉ số của nó trong mảng theo cú pháp:

Tênmảng(chỉ số) → mảng 1 chiều

Tênmảng(chỉ số1, chỉ số2, ..., chỉ số n) → mảng n chiều

4. Khai báo dữ liệu kiểu bản ghi

Bản ghi là tập hợp các phần tử có cùng kiểu dữ liệu hoặc khác kiểu. Đặc trưng của bản ghi gồm tên bản ghi và các phần tử của nó (mỗi phần tử là một trường (Field)), mỗi trường gồm có tên trường và kiểu dữ liệu của nó.

a) Khai báo: dữ liệu kiểu bản ghi khai báo phải đặt trong phần khai báo của đơn thể (module) chương trình, không thể đặt ở cấp biểu mẫu hay thủ tục. Biến bản ghi phải là biến toàn cục.

Cú pháp:

STRUCTURE <Tên bản ghi>

<tên trường 1> AS <kiểu dữ liệu 1>

<tên trường 2> AS <kiểu dữ liệu 2>

.....

<tên trường n> AS <kiểu dữ liệu n>

END STRUCTURE

b) Sử dụng biến bản ghi

Khai báo biến bản ghi: **DIM** <Tên biến> **AS** <Tên bản ghi>

Để làm việc với các trường của bản ghi ta gọi qua tên bản ghi và tên trường theo cú pháp: <Tên bản ghi>.<Tên trường>

IV. CHƯƠNG TRÌNH CON

1. Khái niệm

Chương trình con là 1 đoạn chương trình viết một lần và được dùng nhiều lần trong chương trình. Chương trình con có 2 loại:

- Hàm (Function)
- Thủ tục (Sub)

2. Khai báo chương trình con

a) Khai báo hàm

[Private | Public] FUNCTION <Tên hàm>[(khai báo các biến hình thức)] **[AS kiểu dữ liệu trả về]**

[khai báo các biến và các hàm cục bộ]

các câu lệnh

(Chú ý: phải có ít nhất một câu lệnh gán **tên hàm=biểu thức**

hay **return biểu thức**)

[Exit Function]

END FUNCTION

Public: chỉ định hàm có khả năng hoạt động trong tất cả các thủ tục và các module của chương trình.

Private: chỉ rằng hàm chỉ hoạt động trong module mà nó được khai báo.

b) Khai báo thủ tục

[Private | Public] SUB <Tên thủ tục>[(khai báo các biến hình thức)]

[khai báo các biến và các hàm cục bộ]

Các câu lệnh

[Exit Sub]

END SUB

Public: chỉ định thủ tục có khả năng hoạt động trong tất cả các thủ tục và các module của chương trình.

Private: chỉ rằng thủ tục chỉ hoạt động trong module mà nó được khai báo.

3. Sử dụng chương trình con (lời gọi)

a) Lời gọi hàm

Tênhàm(danh sách các biến thực)

Hàm được dùng như một toán hạng trong biểu thức

b) Lời gọi thủ tục

Call TênThủtục(danh sách các biến thực)

Thủ tục được dùng như một câu lệnh.

4. Khai báo các biến hình thức

Có hai dạng khai báo: Tham trị và Tham biến.

a) Tham trị

Byval Tênbiến As Kiểu dữ liệu

Biến khai báo byval chỉ nhận bản sao → bản gốc không bị thay đổi

b) Tham biến

Byref Tênbiến As Kiểu dữ liệu

Chú ý: Khai báo đối đầu ra phải là kiểu tham biến

Biến khai báo byref nhận bản gốc → (bản gốc có thể bị thay đổi nếu biến thay đổi)

5. Module chương trình

Trong chương trình có lúc ta cần định nghĩa nhiều hàm, thủ tục để dùng chung cho các công việc nào đó, ta nên tạo các chương trình con này thành một tập tin riêng.

Thêm module vào chương trình:

Project / Add Module → xuất hiện cửa sổ để viết chương trình con.

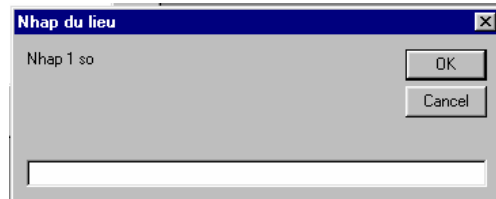
Các chương trình con trong module thường được dùng trong bất kỳ phần nào của Project, vì vậy khi khai báo ta không cần dùng từ khoá Private hay Public. Các Biến muốn được dùng chung phải khai báo là Global.

V. HÀM INPUTBOX

Cho hiển thị một hộp nhập dữ liệu cho phép NSD nhập dữ liệu và hàm sẽ trả về dữ liệu vừa nhập vào.

InputBox(Lời nhắc[,tiêu đề],[default])

Ví dụ: Inputbox(“nhập 1 số”, “Nhập du lieu”) khi chạy ta được như hình:



VI. MSGBOX

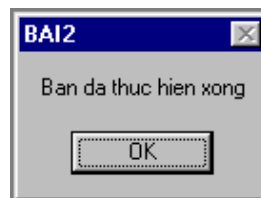
Msgbox được sử dụng như một hàm hay như một thủ tục.

1. MsgBox có thể như một thủ tục

Công dụng: đưa dòng thông báo ra màn hình trong một cửa sổ riêng. NSD chỉ xem xong rồi nhấn nút OK.

Msgbox <dòng thông báo>

Ví dụ: MsgBox “Bạn dữ liệu thực hiện xong”, khi chạy ta được như hình:



2. MsgBox có thể như một hàm

Công dụng: đưa thông báo lên màn hình đồng thời tạo điều kiện cho NSD ấn nút nào đó để chọn sự lựa chọn.

Msgbox(Nội dung, kiểu, “tiêu đề”)

- Kiểu: là các số nguyên dùng để xác định các nút sẽ hiển thị. Bảng liệt kê các giá trị:

Hằng		Giá trị ý nghĩa
VbOkOnly	0	Hiển thị duy nhất nút OK
VbOkCancel	1	Hiển thị 2 nút OK và Cancel
VbAbortRetryIgnore	2	Hiển thị 3 Nút Abort, Retry và Ignore

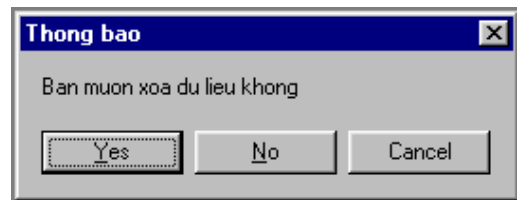
VbYesNoCancel	3	Hiện thị 3 nút Yes, No và Cancel
VbYesNo	4	Hiện thị 2 nút Yes và No
VbRetryCancel	5	Hiện thị 2 nút Retry và Cancel

Hàm MsgBox trả về một giá trị tương ứng với nút NSD đã chọn. Các giá trị qui định cho các nút như sau:

Nút được chọn	Giá trị trả về
OK	1
Cancel	2
Abort	3
Retry	4
Ignore	5
Yes	6
No	7

Ví dụ: t1 = MsgBox("Ban muon xoa du lieu khong", vbYesNoCancel, "Thong bao")

Khi chạy có hình và biến t1 sẽ nhận một giá trị tương ứng với nút được chọn.



VII. CÁC CẤU TRÚC ĐIỀU KHIỂN

1. Cấu rẽ nhánh

a) Cấu trúc IF ... THEN dạng khuyết

IF <bthức điều kiện> THEN

Các câu lệnh

END IF

b) Cấu trúc IF ... THEN dạng đủ

IF <bthức điều kiện> THEN

Các câu lệnh

[ELSEIF <bthức điều kiện> THEN

Các câu lệnh]

ELSE

Các câu lệnh

END IF

c) Cấu trúc SELECT CASE

SELECT CASE <bthức>

CASE <danh sách các giá trị 1>

Các câu lệnh 1

CASE <danh sách các giá trị 2>

Các câu lệnh 2

.....

CASE <danh sách các giá trị n>

Các câu lệnh n

[CASE ELSE

Các câu lệnh n+1]

END SELECT

Danh sách các giá trị ngăn cách nhau bằng dấu phẩy (1, 2, 3,...). Ngoài ra Select Case còn kiểm tra theo dãy số (case 1 to 5 nghĩa là trong khoảng 1-5)

2. Cấu trúc lặp

a) Lặp với số lần lặp biết trước

- Lệnh lặp For

FOR <biến đếm> = <điểm đầu> TO <điểm cuối> [STEP <bước nhảy>]

Các câu lệnh

[EXIT FOR]

NEXT [<biến đếm>]

- Lệnh lặp For Each

FOR EACH biến IN GROUP

Các câu lệnh

[EXIT FOR]

NEXT

b) Lặp với số lần lặp không biết trước

- Lệnh kiểm tra điều kiện trước khi lặp

Trong lúc biểu thức điều kiện đúng thì thực hiện các câu lệnh

DO WHILE <biểu thức điều kiện>

Các câu lệnh

[EXIT DO]

LOOP

WHILE <biểu thức điều kiện>

Các câu lệnh

[EXIT DO]

END WHILE

Trong lúc biểu thức điều kiện sai thì thực hiện các câu lệnh

DO UNTIL <biểu thức điều kiện>

Các câu lệnh

[EXIT DO]

LOOP

- Lệnh kiểm tra điều kiện sau khi lặp

Thực hiện các câu lệnh lặp đi lặp lại cho đến khi biểu thức điều kiện sai thì dừng.

DO

Các câu lệnh

[EXIT DO]

LOOP WHILE <biểu thức điều kiện>

Thực hiện các câu lệnh lặp đi lặp lại cho đến khi biểu thức điều kiện đúng thì dừng.

DO

Các câu lệnh

[EXIT DO]

LOOP UNTIL <biểu thức điều kiện>

VII. NHÃN

Nhãn là một đoạn chỉ thị lệnh bất kỳ trong chương trình được gán một tên xác định. Khi cần thực hiện đoạn chỉ thị lệnh này ta chỉ việc nhảy về nhãn đó. Mỗi nhãn được dùng trong biểu mẫu hoặc đơn thể phải là duy nhất. Cách viết:

Tên nhãn:

Các câu lệnh

Để chuyển đến thực hiện ở một đoạn chương trình hoặc một dòng lệnh bằng cách dùng lệnh nhảy đến nhãn theo cú pháp:

GO TO <Tên nhãn>

IX. QUI ƯỚC KHI VIẾT CHƯƠNG TRÌNH

1. Qui ước khi đặt tên

- Bắt đầu bằng kí tự chữ cái
- Không chứa dấu chấm hay các kí tự đặc biệt trong khai báo dữ liệu.
- Không quá 255 kí tự. Tên của điều khiển, biểu mẫu, lớp và module không quá 40 kí tự.
- Không đặt tên trùng với từ khóa hay tên chuẩn.
- *Nên đặt tên ngắn gọn, dễ gọi nhớ.*

2. Lập trình theo cấu trúc

- Chương trình được viết phải canh thẳng hàng để chương trình sáng sủa dễ đọc.
- Lời chú thích trong chương trình phải đặt trước dấu nháy đơn '.
- Kết hợp nhiều dòng lệnh trên 1 dòng dùng dấu hai chấm :
- Khi chuỗi dài ta dùng dấu gạch dưới _ để tiếp tục nối qua dòng.

Chương IV: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

I. PHƯƠNG PHÁP LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Trong lập trình hướng đối tượng, chương trình là một hệ thống các đối tượng, mỗi đối tượng là sự bao bọc bên trong nó 2 thành phần: Dữ liệu và Hành động.

- Dữ liệu: Là các thông tin chính về đối tượng. (còn gọi là thuộc tính)
- Hành động: Là khả năng mà đối tượng có thể thực hiện. (còn gọi phương thức, hàm thành phần, hành vi).

Mỗi đối tượng được cài đặt vào trong chương trình với dạng đơn thể chứa dữ liệu. Tính chất kế thừa của lập trình hướng đối tượng cho phép ta xây dựng đối tượng mới dựa trên đối tượng đã có.

II. LỚP ĐỐI TƯỢNG

1. Khái niệm

Theo phương pháp lập trình hướng đối tượng, chương trình được thiết kế một phần code đại diện cho một vật tượng đương ngoài đời gọi là lớp.

Lớp là sự tích hợp giữa hai thành phần: dữ liệu và xử lý dữ liệu. Khi đặt tên cho lớp là những danh từ.

2. Khai báo lớp

CLASS tênlớp

Khai báo các thuộc tính của lớp

Khai báo các phương thức

END CLASS

- Trong một file có nhiều lớp, code của mỗi lớp nằm trong Class ... End Class.
- Namespaces để sắp xếp các lớp cho thứ tự theo nhóm, loại.
- Khai báo Namespace:

NAMESPACE tênNamespace

Khai báo các lớp

END NAMESPACE

- Muốn dùng các lớp được khai báo bên trong Namespace theo cú pháp:

tênNamespace.tênlớp

- Phương thức trong VB.NET có hai loại: Sub và Function.
- Phân biệt tham số **byval** và **byref**
 - + byval: lấy bản sao → (bản gốc không bị thay đổi)
 - + byref: lấy bản gốc → (bản gốc có thể bị thay đổi)
- Phạm vi hoạt động của các từ khóa: Private, Friend, Public, Protected
 - + Private: cho phép dùng trong lớp được gọi.
 - + Public: cho phép ai dùng cũng được.
 - + Friend: cho phép dùng trong cùng một Project được gọi.
 - + Protected: cho phép dùng trong lớp con, cháu được gọi.

*Các thuộc tính của lớp khai báo với từ khóa **Private** cho biết biến này chỉ dùng để truy xuất trong phạm vi khai báo nó. Muốn thay đổi hay sử dụng giá trị của thuộc tính ta phải tạo thuộc tính lớp cho các thuộc tính*

3. Tạo thuộc tính của lớp

PUBLIC PROPERTY tên thuộc tính lớp() **AS** Kiểu dữ liệu

GET

END GET

SET (ByVal value As Kiểu dữ liệu)

END SET

END PROPERTY

Từ khóa **Get** sẽ trả về giá trị của thuộc tính và **Set** sẽ gán giá trị khi muốn gán giá trị cho thuộc tính. Thêm vào mã cài đặt đầy đủ cho thuộc tính:

PUBLIC PROPERTY tên thuộc tính lớp () **AS** Kiểu dữ liệu

GET

Return tên thuộc tính

END GET

SET (ByVal value As Kiểu dữ liệu)

tên thuộc tính= value

END SET

END PROPERTY

Khi đọc thông tin từ thuộc tính của đối tượng thì thuộc tính trả về giá trị biến, còn khi gán giá trị thì nó sẽ gán giá trị của biến bằng Value trong phần Set. Trong các thuộc tính phức tạp thì ta có thể cài thêm các câu lệnh xử lý trong khối lệnh Get...End Get, Set...End Set. Việc trả về giá trị và gán giá trị là bắt buộc phải có.

III. ĐỐI TƯỢNG

1. Khái niệm

Đối tượng là sự thể hiện của một lớp. Một lớp có thể có nhiều sự thể hiện khác nhau.

Đối tượng = dữ liệu + phương thức

2. Khai báo đối tượng

Dim tênbiến As New Tênlớp

IV. PHƯƠNG THỨC THIẾT LẬP

1. Mục đích

Các phương thức thiết lập của một lớp có nhiệm vụ thiết lập thông tin ban đầu cho các đối tượng thuộc về lớp ngay khi đối tượng được khai báo.

2. Các đặc điểm của phương thức thiết lập

- Phương thức thiết lập của lớp được định nghĩa thông qua toán tử new.
- Không có giá trị trả về.
- Được tự động gọi ngay khi đối tượng được khai báo.
- Có thể có nhiều phương thức thiết lập trong một lớp.
- Trong một quá trình sống của đối tượng thì chỉ có một lần duy nhất phương thức thiết lập được gọi thực hiện đó là khi đối tượng được khai báo.
- Các phương thức thiết lập của lớp thuộc nhóm các phương thức khởi tạo.

3. Phân loại phương thức thiết lập

Phương thức thiết lập của một lớp gồm 3 nhóm:

- Phương thức thiết lập mặc định.
- Phương thức thiết lập sao chép.
- Phương thức thiết lập nhận tham số đầu vào.

4. Ví dụ

Tạo lớp đối tượng Sinh viên.

Lớp Sinh viên:

- Dữ liệu (thuộc tính):
Mã sinh viên, họ tên, mã lớp, điểm trung bình
- Phương thức:
 - + Khởi tạo Sinh viên
 - + Tính học bổng

```
Public Class SINHVIEN

    Private masv As String

    Private hten As String

    Private malop As String

    Private dtb As Double

    Public Sub New()

        masv = ""

        hten = ""

        malop = ""

        dtb = 0

    End Sub

    Public Sub New(masv1 As String, hten1 As String, malop1 As String, dtb1 As Double)

        If dtb1 > 10 Or dtb1 < 0 Then

            MsgBox("DDTB sai")

        Else

            masv = masv1

            hten = hten1

            malop = malop1

            dtb = dtb1

        End If

    End Sub

    Public Sub New(sv As SINHVIEN)
```

```
masv = sv.masv  
  
hten = sv.hten  
  
malop = sv.malop  
  
dtb = sv.dtb
```

End Sub

Public Property S_masv() As String

Get

```
Return masv
```

End Get

Set(value As String)

```
masv = value
```

End Set

End Property

Public Property S_hten() As String

Get

```
Return hten
```

End Get

Set(value As String)

```
hten = value
```

End Set

End Property

Public Property S_malop() As String

Get

```
Return malop
```

End Get

Set(value As String)

```
malop = value
```

End Set

End Property

Public Property S_DTB() As Double

Get

```

        Return dtb

    End Get

    Set(value As Double)

        dtb = value

    End Set

End Property

Overridable Function HBong() As Integer

    If dtb >= 8 Then

        Return 100

    ElseIf dtb >= 7 Then

        Return 50

    Else

        Return 0

    End If

End Function

End Class

```

V. TÍNH KẾ THỪA TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

1. Khái niệm

Kế thừa là một trong các nguyên tắc cơ bản của LTHĐT. Đặc biệt ở đây là cơ sở cho việc nâng cao khả năng sử dụng lại các bộ phận của chương trình. Kế thừa cho phép ta định nghĩa một lớp mới, gọi là lớp dẫn xuất (lớp con), từ một lớp đã có gọi là lớp cơ sở (lớp cha).

2. Một số chú ý của tính kế thừa

- Các trường dữ liệu của lớp con không trùng tên với các trường dữ liệu đã khai báo trong lớp cha.
- Lớp con kế thừa tất cả các trường dữ liệu của lớp cha.
- Lớp con kế thừa các phương thức của lớp cha không trùng tên với các phương thức đã khai báo ở lớp con. Với các phương thức trùng tên, nó sẽ có ý nghĩa trong lớp con. Như vậy để hủy phương thức từ lớp cha chỉ cần tạo một phương thức trùng tên trong lớp con.
- Khi xây dựng các phương thức của lớp con có thể và cần sử dụng các phương thức của lớp cha để khỏi viết lại các câu lệnh trùng lặp.

- Trong 1 module có thể sử dụng đồng thời các đối tượng của lớp cha và đối tượng của lớp con.
- Khi một thông điệp được gửi tới một đối tượng của lớp con thì quá trình phản hồi diễn ra như sau:
 - + Đầu tiên hệ thống đi tìm một phương thức tương ứng với lớp con. Nếu tìm thấy phương thức này sẽ được thực hiện.
 - + Nếu việc tìm phương thức trong lớp con thất bại, hệ thống sẽ xét đến các phương thức của lớp cha.

3. Ví dụ

Tạo một lớp Sinh viên kế thừa lớp Sinh viên đã tạo trên. Bổ sung thêm thuộc tính điểm rèn luyện và tính lại học bổng.

```
Public Class SVIEN_KETHUA
    Inherits SINHVIEN

    Private dr1 As Double

    Public Property S_dr1() As Double
        Get
            Return dr1
        End Get
        Set(value As Double)
            dr1 = value
        End Set
    End Property

    Public Sub New(masv1 As String, hten1 As String, malop1 As String, dtb1 As Double, dr11 As Double)
        S_masv = masv1
        S_hten = hten1
        S_malop = malop1
        S_DTB = dtb1
        dr1 = dr11
    End Sub

    Overrides Function HBong() As Integer
        If S_DTB + dr1 >= 8 Then
```

```

        Return 500

    ElseIf S_DTB + dr1 >= 7 Then

        Return 300

    Else

        Return 0

    End If

End Function

End Class

```

VI. ĐỊNH NGHĨA TOÁN TỬ TRÊN LỚP

1. Cú pháp

PUBLIC SHARED OPERATOR kí hiệu toán tử (khai báo các đối) As kiểu dữ liệu

.....

Return biểu thức

END OPERATOR

2. Ví dụ

Định nghĩa toán tử cộng, trừ, so sánh hai phân số.

```

Public Class PHANSO

    Private tu As Integer

    Private mau As Integer

    Public Sub New()

        Return

    End Sub

    Public Sub New(t As Integer, m As Integer)

        tu = t

        mau = m

    End Sub

    Public Property S_tu() As Integer

        Get

            Return tu

        End Get

        Set(value As Integer)

```

```

        tu = value

    End Set

End Property

Public Property S_mau() As Integer

    Get

        Return mau

    End Get

    Set(value As Integer)

        mau = value

    End Set

End Property

Public Shared Operator +(ByVal x As PHANSO, ByVal y As PHANSO) As PHANSO

    Dim ps As New PHANSO

    ps.tu = x.tu * y.mau + y.tu * x.mau

    ps.mau = x.mau * y.mau

    Return ps

End Operator

Public Shared Operator -(ByVal x As PHANSO, ByVal y As PHANSO) As PHANSO

    Dim ps As New PHANSO

    ps.tu = x.tu * y.mau - y.tu * x.mau

    ps.mau = x.mau * y.mau

    Return ps

End Operator

Public Shared Operator >(ByVal x As PHANSO, ByVal y As PHANSO) As Boolean

    Dim ps As New PHANSO

    ps = x - y

    Return (ps.tu * ps.mau > 0)

End Operator

Public Shared Operator <(ByVal x As PHANSO, ByVal y As PHANSO) As Boolean

    Dim ps As New PHANSO

    ps = x - y

    Return (ps.tu * ps.mau < 0)

```

End Operator

Public Sub rutgon()

Dim a, b As Integer

a = Math.Abs(tu)

b = Math.Abs(mau)

While (a <> b) And (a <> 0)

If (a > b) Then

a = a - b

Else

b = b - a

End If

End While

tu = tu \ b

mau = mau \ b

End Sub

End Class

VII. BÀI TẬP

- 1) Tạo lớp đối tượng phân số
- 2) Tạo lớp đối tượng tam giác
- 3) Tạo lớp đối tượng điểm trong mặt phẳng
- 5) Viết chương trình nhập tọa độ 3 đỉnh A, B, C của một tam giác trong mặt phẳng Oxy. Tính diện tích tam giác và in kết quả.
- 6) Mỗi nhóm xây dựng bài tập cho ví dụ về tính kế thừa.

Chương V: LẬP TRÌNH CƠ SỞ DỮ LIỆU

I. CÁC KHÁI NIỆM LIÊN QUAN ĐẾN CƠ SỞ DỮ LIỆU

1. Bộ máy cơ sở dữ liệu là gì

Chức năng cơ bản của một cơ sở dữ liệu được cung cấp bởi một bộ máy cơ sở dữ liệu, là hệ thống chương trình quản lý cách thức chứa và trả về dữ liệu.

Bộ máy cơ sở dữ liệu là **Microsoft Jet**, đây là một hệ thống con được nhiều ứng dụng của Microsoft sử dụng.

2. Thiết kế cơ sở dữ liệu

Để tạo một cơ sở dữ liệu, trước hết trước hết ta phải xác định các thông tin liên quan. Sau đó, ta thiết kế cơ sở dữ liệu, tạo bảng chứa các trường định nghĩa kiểu dữ liệu của trường đó. Sau khi tạo ra cấu trúc cơ sở dữ liệu, cơ sở dữ liệu có thể chứa dữ liệu dưới dạng các bản ghi (mẫu tin – record).

3. DataSet là gì


DataSet là một bản sao của tập hợp con các mẫu tin từ bộ điều phối (Data Adapter) truy vấn dữ liệu từ cơ sở dữ liệu sau khi đã kết nối với CSDL.

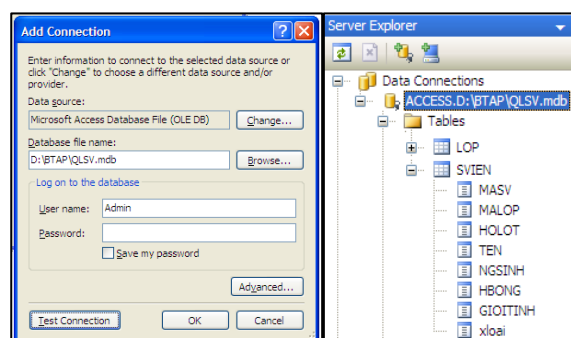
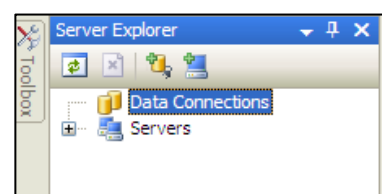
ADO.NET ta có thể truy xuất đến mọi hệ CSDL theo cùng cách thức và mã chương trình như nhau.

II. KẾT NỐI CƠ SỞ DỮ LIỆU VỚI ADO.NET

1. Thiết lập kết nối CSDL

Sử dụng Server Explorer để thiết lập kết nối đến CSDL

- View/Server Explorer (Ctrl Alt S)
- Xuất hiện cửa sổ **Server Explorer**
 - + Chọn nút Connect DataBase  để kết nối CSDL
 - + Xuất hiện hộp hội thoại **Choose Data Source**, chọn nguồn CSDL cần kết nối.
 - + Xuất hiện hộp hội thoại **Add Connection**
 - + Chọn tập tin CSDL cần kết nối
 - + Nhấn OK để thêm kết nối vào Server Explorer



2. Tạo bộ điều phối dữ liệu *DataAdapter*



- Tạo bộ điều phối *DataAdapter* để truy vấn dữ liệu từ các bảng trong CSDL.
- *Data Adapter* là nền tảng để tạo *DataSet*

Dùng đối tượng **OleDbDataAdapter** để tạo bộ điều phối

3. Đối tượng điều khiển *OleDbDataAdapter*

- Chọn đối tượng *OleDbDataAdapter*  trong Tab **Data** trên ToolBox.

(Nếu không có biểu tượng *OleDbDataAdapter* trên ToolBox, ta đưa vào bằng cách: nhấp chuột phải vào Tab Data chọn Choose Item.... , trong Tab .NET Framework Components / Chọn vào mục *OleDbDataAdapter*)

- Xuất hiện hộp hội thoại **Data Adapter Configuration Wizard** / Chọn Next / Chọn Yes / chọn Next trong hộp hội thoại
- Chọn Query Builder... để thực hiện tạo câu lệnh SQL (sau khi tạo câu lệnh SQL ta có thể chọn nút Execute Query để xem kết quả truy vấn) / Chọn OK / Chọn Next / Chọn Finish. Hai đối tượng  *OleDbDataAdapter1*  *OleDbConnection1* được tạo trên khay hệ thống.

4. Tạo đối tượng *DataSet*

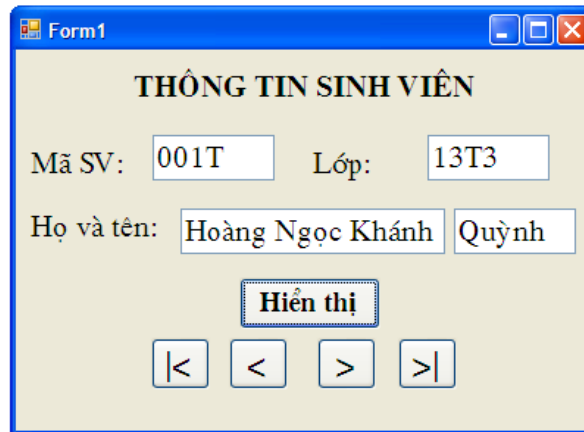
DataSet là đối tượng chứa dữ liệu cho người dùng thao tác. Nó là bản sao của *DataAdapter* (là ảnh của CSDL), vì vậy mọi thao tác của người dùng ảnh hưởng đến CSDL khi có yêu cầu **Cập nhật** dữ liệu (Thêm, Sửa, Xóa).

Tạo *DataSet*: Chọn menu **Data/Generate DataSet** xuất hiện hộp hội thoại/ Đặt tên cho *DataSet* trong hộp **New** / Chọn **OK** và *DataSet* được tạo trên khay hệ thống.

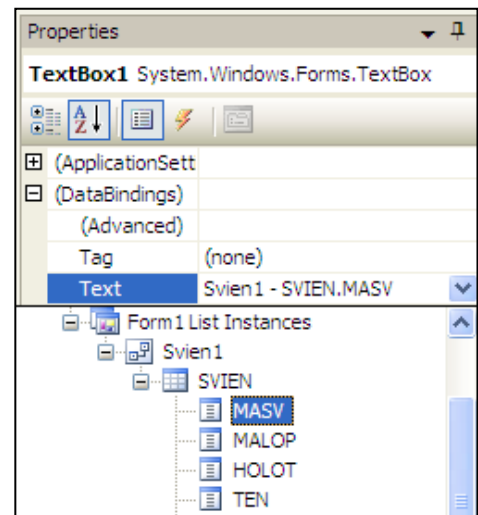
5. Trình bày dữ liệu từ CSDL lên Form

Để các điều khiển như Textbox, Label, Listbox, Combobox, RadioButon, DataGrid hiển thị dữ liệu từ CSDL lên Form ta phải thực hiện ràng buộc dữ liệu (data binding). Dữ liệu hiển thị lên trong các điều khiển phụ thuộc vào nguồn dữ liệu có trong *DataSet* hay *DataAdapter*.

Ví dụ 1: Hiển thị dữ liệu lên Textbox trong Form như hình ta thực hiện



- Kết nối CSDL với tập tin QLSV.MDB
- Tạo bộ điều phối OleDbDataAdapter có tên OleDbDataAdapter1 với bảng SVIEN trong QLSV.MDB
- Tạo đối tượng DataSet có tên SVIEN1
- Đưa các Textbox vào Form như hình
- Ràng buộc các textbox vào các trường tương ứng trong bảng SVIEN: Chọn Properties của Textbox cần ràng buộc
- Viết code với nút **Hiển thị** để đưa dữ liệu từ DataSet lên Form với lệnh

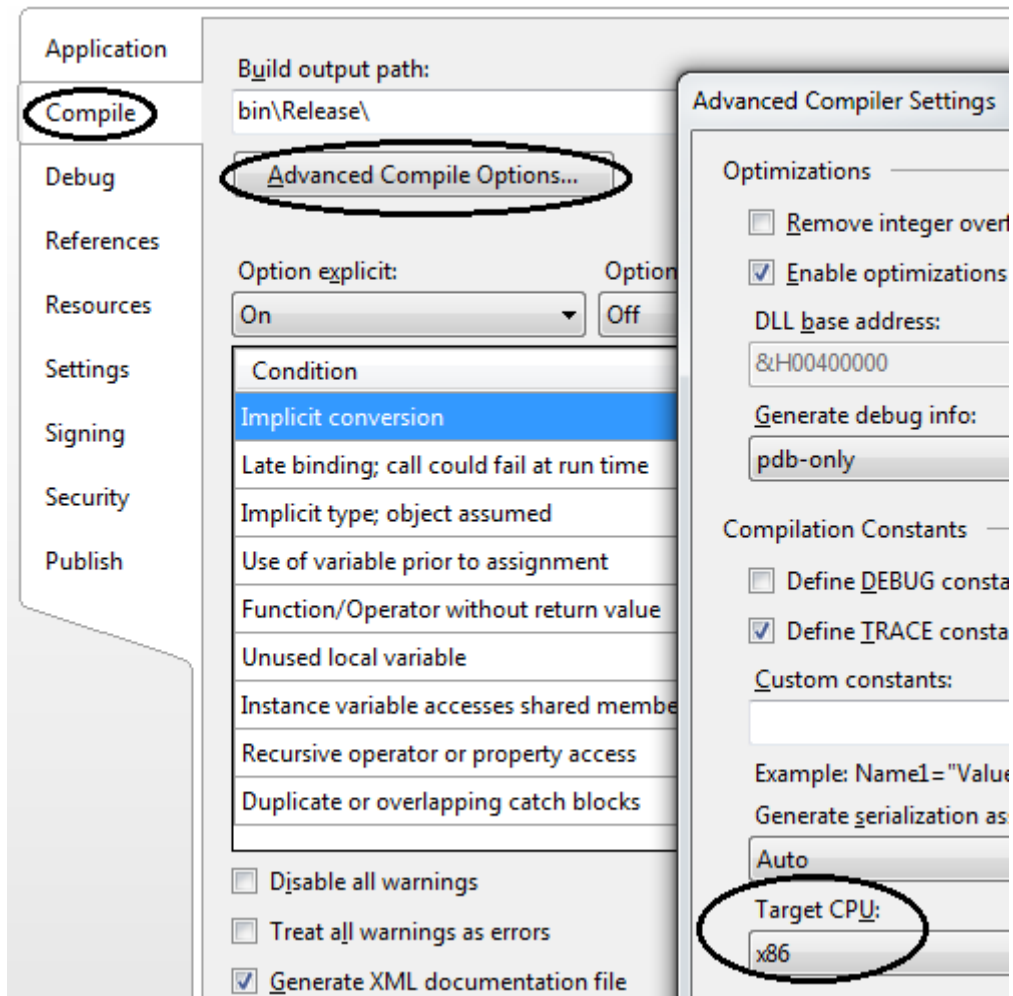


Tên_DataSet.clear() xoá sạch dữ liệu mà DataSet đã nắm giữ trước nếu có

Tên_OleDbDataAdapter.Fill(Tên_DataSet) điều khiển bộ điều phối điền dữ liệu vào đối tượng DataSet

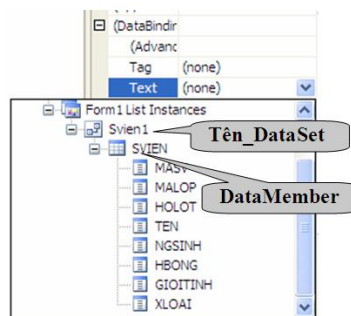
Nếu báo lỗi **“Microsoft.Jet.OLEDB.4.0’ provider is not registered on the local machine”** ta thực hiện:

- Tại cửa sổ **Solution explorer** kích phải chuột vào tên Solution/ **Properties**
- Chọn Tab **Compile** / chọn nút lệnh **Advanced Compile Options...** Xuất hiện hộp thoại
- Chọn **Target CPU** là x86



- Viết code cho 4 nút điều khiển: Trong ADO.NET không có phương thức MoveNext, MovePrevious, MoveLast, MoveFirst. Muốn thực hiện duyệt từng bản ghi ADO.NET cho phép quản lý và duyệt các bản ghi bằng đối tượng CurrentManager. Với đối tượng này ta biết được vị trí của bản ghi hiện hành. Mỗi DataSet đều có sẵn một CurrentManager và mỗi đối tượng Form đều có thuộc tính BindingContext theo dõi tất cả các CurrentManager trên Form.


Các phương thức liên quan công việc duyệt các bản ghi



BindingContext(tên_DataSet, "DataMember").Count → tổng số các bản ghi

BindingContext(tên_DataSet, "DataMember").position= Số thứ tự của bản ghi
 → di chuyển đến bản ghi thứ Stt - 1 (bản ghi đầu tiên có STT = 0)

Ví dụ 2: Hiển thị dữ liệu lên DataGrid như hình ta thực hiện



MASV	MALOP	HOLOT	TEN	NGSINH	HBONG	GIOITINH	XLOAI
001T	13T3	Hoàng Ngọc Khánh	Quỳnh	01/01/1994	1000	<input type="checkbox"/>	co
002T	09T2	Vô Thị Thuỳ	Trang	11/01/1992	800	<input type="checkbox"/>	co
003T	12T1	Hoàng Ngọc	Khánh	13/06/1992	900	<input type="checkbox"/>	co
010T	12T1	Lê Trần Thảo	nhi	01/01/1993	200	<input type="checkbox"/>	co
004T	11T2	Hoàng Ngọc	Thịnh	15/12/1994	1200	<input type="checkbox"/>	co
005T	12T1	Nguyễn Hoàng	Tuấn	(null)	1000	<input type="checkbox"/>	co
006T	11T1	Lê Thị	Vân Anh	01/01/1999	1200	<input type="checkbox"/>	(null)
007T	10T1	Lê Trung	Chinh	(null)	500	<input type="checkbox"/>	co
008T	10T2	Trần Thị	Lan	07/07/1994	400	<input type="checkbox"/>	co
009T	11T2	Nguyễn Phạm	Thân	03/02/1991	600	<input type="checkbox"/>	co

Tương tự như ví dụ 1

- Kết nối CSDL với tập tin QLSV.MDB
- Tạo bộ điều phối OleDbDataAdapter có tên OleDbDataAdapter1 với bảng SVIEN trong QLSV.MDB
- Tạo đối tượng DataSet
- Đưa đối tượng DataGrid vào Form (Nếu trên thanh toolbox không có biểu tượng DataGrid, ta bổ sung vào như đã bổ sung OleDbDataAdapter như trên). Ràng buộc thuộc tính **DataSoure** và **DataMember**
- Viết Code cho nút **Hiện thị** như trong ví dụ 1.

II. LẬP TRÌNH KẾT NỐI CƠ SỞ DỮ LIỆU

1. Kết nối cơ sở dữ liệu

Khai báo chuỗi kết nối

Dim tên chuỗi kết nối As String = "Provider= bộ máy CSDL;" _

& "Data Source = " & Application.StartupPath & "\tên CSDL;"

Biến kết nối = New OleDbConnection(tên chuỗi kết nối)

Biến kết nối được khai báo kiểu *OleDbConnection*

Ví dụ:

Dim constring As String = "Provider= Microsoft.Jet.OLEDB.4.0;" _

& "Data Source = " & Application.StartupPath & "\QLSV.mdb;"

Dim con as OleDbConnection

con = New OleDbConnection(constring)

2. Truy vấn dữ liệu từ các bảng trong CSDL

- Khai báo câu lệnh truy vấn dùng để đọc bảng trong CSDL

Biến lệnh truy vấn = Chuỗi câu lệnh SQL

Ví dụ:

Dim lenh as string

lenh = "select * from svien"

- Khai báo biến đối tượng Command dùng để thực hiện câu lệnh truy vấn

Dim biến thực hiện câu truy vấn As New OleDbCommand(Biến lệnh truy vấn, Biến kết nối)

Ví dụ: Dim cmd As New OleDbCommand(lenh, con)

- Mở kết nối ra để đọc dữ liệu từ bảng

Biến kết nối.Open

Ví dụ: Con.open

3. Đọc dữ liệu

- Sử dụng phương thức ExecuteReader để đọc và trả về cho đối tượng DataReader

Dim biến chứa dữ liệu đọc từ câu truy vấn As OleDbDataReader = biến thực hiện câu truy vấn.ExecuteReader

Ví dụ: Dim bang_doc As OleDbDataReader = cmd.ExecuteReader

- Khai báo biến kiểu DataTable để nhận kết quả là các dòng đọc được từ bảng truy vấn từ câu lệnh SQL

Dim biến dataTable As New DataTable("Bảng")

Ví dụ: Dim dttable As New DataTable("svien")

- Sử dụng phương thức Load và gởi vào DataReader để bắt đầu đọc các dòng ra DataTable

biến dataTable.Load(biến chứa dữ liệu đọc từ câu truy vấn, LoadOption.OverwriteChanges)

Ví dụ: `dttable.Load(bang_doc, LoadOption.OverwriteChanges)`

- Đóng kết nối: sau khi đọc xong cần đóng kết nối lại

Biến kết nối.Close

Ví dụ: `Con.close`

4. Khởi tạo liên kết với dataTable

- Khai báo biến để làm việc với các bản ghi trong dataTable (khai báo toàn cục)

PRIVATE WITHEVENTS biến As BindingManagerBase

Biến = Me.BindingContext(Biến dataTable)

Ví dụ:

Private WithEvents danhsach As BindingManagerBase

DataGrid1.DataSource = dttable → đưa dataTable vào DataGrid

danhsach = Me.BindingContext(dttable)

Ví dụ 1: Chức năng cập nhật dữ liệu vào bảng SINHVIEN

MASV	MALOP HOLOT	TÊN	NGSIN	HBONC
001T	12T2	Hoàng Ngọc Khánh	Quỳnh	01/01/1 1000
002T	09T2	Võ Thị Thuý	Trang	11/01/1 800
003T	12T1	Hoàng Ngọc Khánh	Khánh	13/06/1 900
010T	12T1	Lê Trần Thảo	Nhi	01/01/1 200
004T	11T2	Hoàng Ngọc	Thịnh	15/12/1 1200

Imports System.Data.OleDb

Public Class Form1

Private con As OleDbConnection

Private WithEvents danhsach As BindingManagerBase

Public lenh As String

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

'Kết nối cơ sở dữ liệu

'Khai báo chuỗi kết nối

```
Dim constring As String = "Provider= Microsoft.Jet.OLEDB.4.0;" _  
    & "Data Source = " & Application.StartupPath & "\QLSV.mdb;"  
con = New OleDbConnection(constring)  
Xuat_sinhvien()  
danh_sach_moi(sender, e)
```

End Sub

```
Private Sub Xuat_sinhvien()
```

```
    Dim lenh As String
```

```
    'Khai báo câu lệnh truy vấn dùng để đọc bảng SinhVien
```

```
    lenh = "select * from svien"
```

```
    'Khai báo đối tượng Command dùng để thực hiện câu lệnh truy vấn
```

```
    Dim cmd As New OleDbCommand(lenh, con)
```

```
    'Trước khi đọc cần mở kết nối ra
```

```
    con.Open()
```

```
    'Sử dụng phương thức ExecuteReader để đọc và trả về cho đối tượng DataReader
```

```
    Dim bang_doc As OleDbDataReader = cmd.ExecuteReader
```

```
    'Khai báo DataTable để nhận kết quả là các dòng đọc được
```

```
    Dim dttable As New DataTable("svien")
```

```
    'Sử dụng phương thức Load và gửi vào DataReader để bắt đầu đọc các dòng ra  
    DataTable
```

```
    dttable.Load(bang_doc, LoadOption.OverwriteChanges)
```

```
    'Sau khi đọc cần đóng kết nối lại
```

```
    con.Close()
```

```
    DataGrid1.DataSource = dttable
```

```
    'khởi tạo liên kết với datatable.
```

```
    Danh_sach = Me.BindingContext(dttable)
```

End Sub

```
Private Sub xuat()
```

```
    T_masv.Text = danh_sach.Current("masv")
```

```
    T_holot.Text = danh_sach.Current("holot")
```

```
    T_ten.Text = danh_sach.Current("ten")
```

```
    T_malop.Text = danh_sach.Current("malop")
```

End Sub

```
Private Sub danh_sach_moi(ByVal sender As Object, ByVal e As System.EventArgs)
    Handles danh_sach.PositionChanged

    xuat()

```

End Sub

```
Private Sub bt_Them_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles bt_them.Click

    T_masv.Text = ""
    T_holot.Text = ""
    T_ten.Text = ""
    T_malop.Text = ""

```

End Sub

```
Private Sub bt_Luu_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles bt_luu.Click

    If MsgBox("Bạn có muốn lưu không?", MsgBoxStyle.Question + MsgBoxStyle.YesNo,
        "Lưu") = MsgBoxResult.Yes Then

        If T_masv.Text = "" Or T_holot.Text = "" Or T_ten.Text = "" Or
            T_malop.Text = "" Then

            MsgBox("Chưa nhập giá trị!!!")

        Else

            lenh = "insert into svien(Masv,holot,ten,malop) values('" & _
                T_masv.Text & "', '" & T_holot.Text & "', '" & T_ten.Text & _
                "', '" & T_malop.Text & "')"

            Dim cmd As New OleDbCommand(lenh, con)

            con.Open()

            cmd.ExecuteNonQuery()

            con.Close()

            Xuat_sinhvien()

            MsgBox("Đã thêm!")

        End If

    End If

End Sub
```

```
Private Sub bt_Sua_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles bt_sua.Click

```

```

If MsgBox("Bạn có muốn sửa không ?", MsgBoxStyle.Question +
        MsgBoxStyle.YesNo, "Sửa ") = MsgBoxResult.Yes Then

    If T_masv.Text = "" Or T_holot.Text = "" Or T_ten.Text = "" Or _
        T_malop.Text = "" Then

        MsgBox(" Nhập Giá Trị Cần Sửa !!! ,,")

    Else

        lenh = "Update svien set masv='" & T_masv.Text & "', holot= '" & _
            T_holot.Text & "', ten= '" & T_ten.Text & "', malop= '" & _
            T_malop.Text & "' where Masv = '" & Trim(T_masv.Text) & "'"

        Dim cmd As New OleDbCommand(lenh, con)

        con.Open()

        cmd.ExecuteNonQuery()

        con.Close()

        Xuat_sinhvien()

        MsgBox("Sửa thành công")

    End If

End If

End Sub

```

```

Private Sub bt_Xoa_Click(ByVal sender As System.Object, ByVal e As
        System.EventArgs) Handles bt_xoa.Click

    If MsgBox("Bạn có muốn xóa không ?", MsgBoxStyle.Question + MsgBoxStyle.YesNo,
        "Xóa") = MsgBoxResult.Yes Then

        If T_masv.Text = "" Then

            MsgBox(" Phải Chọn Giá Trị Cần Xóa !!! ")

        Else

            lenh = " delete * from svien where Masv = '" & T_masv.Text & "'
                where Masv = '" & Trim(T_masv.Text) & "'"

            Dim cmd As New OleDbCommand(lenh, con)

            con.Open()

            cmd.ExecuteNonQuery()

            con.Close()

            Xuat_sinhvien()

            MsgBox("Xóa thành công")

        End If

    End If

```

End If

End Sub

```
Private Sub bt_Vedau_Click(ByVal sender As System.Object, ByVal e As  
                           System.EventArgs) Handles bt_vedau.Click
```

```
    If danh_sach.Position = 0 Then
```

```
        MsgBox("Đã về Đầu")
```

```
    Else
```

```
        danh_sach.Position = 0
```

```
    End If
```

End Sub

```
Private Sub bt_Vetruoc_Click(ByVal sender As System.Object, ByVal e As  
                             System.EventArgs) Handles bt_vetruoc.Click
```

```
    If danh_sach.Position = 0 Then
```

```
        MsgBox("Đã về đầu")
```

```
    Else
```

```
        danh_sach.Position -= 1
```

```
    End If
```

End Sub

```
Private Sub bt_Vesau_Click(ByVal sender As System.Object, ByVal e As  
                           System.EventArgs) Handles bt_vesau.Click
```

```
    If danh_sach.Position = danh_sach.Count - 1 Then
```

```
        MsgBox("Đã về cuối")
```

```
    Else
```

```
        danh_sach.Position += 1
```

```
    End If
```

End Sub

```
Private Sub bt_Vecuoi_Click(ByVal sender As System.Object, ByVal e As  
                            System.EventArgs) Handles bt_vecuoi.Click
```

```
    If danh_sach.Position = danh_sach.Count - 1 Then
```

```
        MsgBox("Đã về cuối")
```

```
    Else
```

```
        danh_sach.Position = danh_sach.Count - 1
```

```
    End If
```

End Sub

End Class

Ví dụ 2: Chức năng cập nhật dữ liệu vào bảng HÀNG HÓA

MAHG	TENHG	GIABAN	SOLG
001	Gạo	5000	51
002	Sắn	1000	13
003	Bắp	3000	20
005	Đậu xanh	3000	20
006	Đậu phụng	1700	37
004	Ngô	1500	20

```
Public Class HÀNG
```

```
    Private ma$
```

```
    Private ten$
```

```
    Private gia%
```

```
    Private sl%
```

```
    Public Property p_ma$()
```

```
        Get
```

```
            Return ma
```

```
        End Get
```

```
        Set(value$)
```

```
            ma = value
```

```
        End Set
```

```
    End Property
```

```
    Public Property p_ten$()
```

```
        Get
```

```
            Return ten
```

```
        End Get
```

```
        Set(value$)
```



```
        ten = value

    End Set

End Property
```

```
Public Property p_gia$()

    Get

        Return gia

    End Get

    Set(value$)

        gia = value

    End Set

End Property
```

```
Public Property p_sl$()

    Get

        Return sl

    End Get

    Set(value$)

        sl = value

    End Set

End Property
```

```
Public Sub New(mah$, tenh$, slh%, giah%)

    ma = mah

    ten = tenh

    gia = giah

    sl = slh

End Sub
```

```
Public Sub Luu(ByRef lenh$)

    lenh = "insert into Hang(mahg, tenhg, solg, giaban) values('" & ma & "',''" & _
        ten & "',''" & sl & "','" & gia & "')"

End Sub
```

End Sub

Public Sub xoa(ByRef lenh\$)

lenh = "delete * from Hang where Mahg = '" & ma.Trim & "'"

End Sub

Public Sub sua(ByRef lenh\$)

lenh = "Update hang set mahg='" & ma & "',tenhg= '" & ten & "', solg =" & sl_
& ", giaban=" & gia & " where mahg='" & ma.Trim & "'"

End Sub

End Class

Imports System.Data.OleDb

Public Class Form1

Private con As OleDbConnection

Private WithEvents danh_sach As BindingManagerBase

Public lenh As String

Private Sub Form1_Load(sender As System.Object, e As System.EventArgs) Handles
MyBase.Load

Dim constring As String = "Provider= Microsoft.Jet.OLEDB.4.0;" _
& "Data Source = " & Application.StartupPath & "\QLSV.mdb;"

con = New OleDbConnection(constring)

Xuat_hang()

danh_sach_moi(sender, e)

End Sub

Private Sub Xuat_hang()

Dim lenh As String

lenh = "select * from hang"

Dim cmd As New OleDbCommand(lenh, con)

con.Open()

Dim bang_doc As OleDbDataReader = cmd.ExecuteReader

Dim dttable As New DataTable("hang")

```
dttable.Load(bang_doc, LoadOption.OverwriteChanges)

con.Close()

DataGrid1.DataSource = dttable

danh_sach = Me.BindingContext(dttable)
```

```
End Sub
```

```
Private Sub xuat()

    T_mahg.Text = danh_sach.Current("mahg")

    T_Tenhg.Text = danh_sach.Current("tenhg")

    T_giaban.Text = danh_sach.Current("giaban")

    T_Solhg.Text = danh_sach.Current("solhg")
```

```
End Sub
```

```
Private Sub danh_sach_moi(ByVal sender As Object, ByVal e As System.EventArgs)
    Handles danh_sach.PositionChanged
```

```
    xuat()
```

```
End Sub
```

```
Private Sub Them_Click(sender As System.Object, e As System.EventArgs) Handles
    Them.Click
```

```
    T_mahg.Text = ""
    T_Tenhg.Text = ""
    T_Solhg.Text = ""
    T_giaban.Text = ""
    Them.Enabled = False
    Luu.Enabled = True
    Sua.Enabled = False
    Xoa.Enabled = False
```

```
End Sub
```

```
Private Sub Luu_Click(sender As System.Object, e As System.EventArgs) Handles
    Luu.Click
```

```
    If MsgBox("Bạn có muốn lưu không? ", vbYesNo, "Luu") = MsgBoxResult.Yes Then
```

```

If T_mahg.Text = "" Or T_Tenhg.Text = "" Or T_giaban.Text = "" Or
    T_Solg.Text = "" Then

    MsgBox("Chưa nhập đủ")

Else

    Dim hag As New HANG(T_mahg.Text, T_Tenhg.Text, T_Solg.Text,
        T_giaban.Text)

    hag.Luu(lenh)

    Dim cmd As New OleDbCommand(lenh, con)

    con.Open()

    cmd.ExecuteNonQuery()

    con.Close()

    Xuat_hang()

    MsgBox("Đã thêm!")

End If

End If

Them.Enabled = True

Luu.Enabled = False

Sua.Enabled = True

Xoa.Enabled = True

End Sub

```

```

Private Sub Sua_Click(sender As System.Object, e As System.EventArgs) Handles
    Sua.Click

    If MsgBox("Bạn có muốn sửa không? ", vbYesNo, "Sửa") = MsgBoxResult.Yes Then

        If T_mahg.Text = "" Or T_Tenhg.Text = "" Or T_giaban.Text = "" Or
            T_Solg.Text = "" Then

            MsgBox("Chưa nhập đủ")

        Else

            Dim hag As New HANG(T_mahg.Text, T_Tenhg.Text, T_Solg.Text,
                T_giaban.Text)

            hag.sua(lenh)

            Dim cmd As New OleDbCommand(lenh, con)

            con.Open()

```

```
        cmd.ExecuteNonQuery()

        con.Close()

        Xuat_hang()

        MsgBox("Sửa thành công")

    End If

End If

End Sub
```

```
Private Sub Xoa_Click(sender As System.Object, e As System.EventArgs) Handles Xoa.Click

    If MsgBox("Bạn có muốn xóa không? ", vbYesNo, "Xóa") = MsgBoxResult.Yes Then

        Dim hag As New HANG(T_mahg.Text, T_Tenhg.Text, T_Solg.Text,
                                T_giaban.Text)

        hag.xoa(lenh)

        Dim cmd As New OleDbCommand(lenh, con)

        con.Open()

        cmd.ExecuteNonQuery()

        con.Close()

        Xuat_hang()

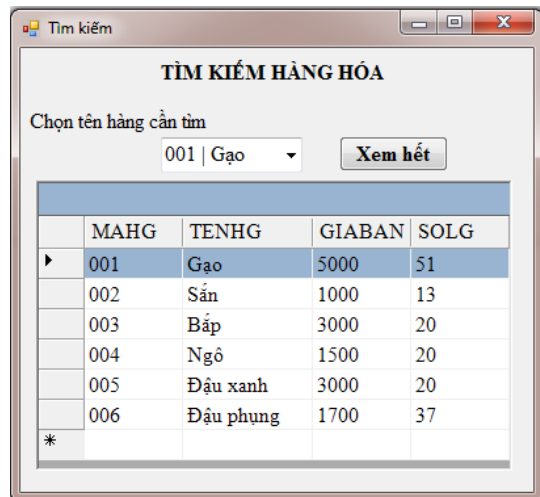
        MsgBox("Xóa thành công")

    End If

End Sub

End Class
```

Ví dụ 3: Chức năng tìm kiếm hàng hóa trong bảng HÀNG HÓA



`Imports System.Data.OleDb`

`Public Class TIMKIEM`

`Private con As OleDbConnection`

`Private WithEvents danh_sach As BindingManagerBase`

`Public lenh As String`

`Private Sub TIMKIEM_Load(sender As System.Object, e As System.EventArgs) Handles MyBase.Load`

`Dim constring As String = "Provider= Microsoft.Jet.OLEDB.4.0;" _`

`& "Data Source = " & Application.StartupPath & "\QLSV.mdb;"`

`con = New OleDbConnection(constring)`

`Xuat_hang()`

`CB.Text = danh_sach.Current("mahg") & " | " & danh_sach.Current("tenhg")`

`'Lấy dữ liệu từ bảng đưa vào Combobox`

`For I = 0 To danh_sach.Count - 1`

`danh_sach.Position = I`

`CB.Items.Add(danh_sach.Current("mahg") & " | " & _
danh_sach.Current("tenhg"))`

`Next`

`End Sub`

`Private Sub Xuất_hang()`

```

lenh = "select * from hang order by mahg"

Dim cmd As New OleDbCommand(lenh, con)

con.Open()

Dim bang_doc As OleDbDataReader = cmd.ExecuteReader

Dim dttable As New DataTable("hang")

dttable.Load(bang_doc, LoadOption.OverwriteChanges)

con.Close()

danh_sach = Me.BindingContext(dttable)

DataGrid1.DataSource = dttable

End Sub

```

```

Private Sub CB_SelectedIndexChanged(sender As System.Object, e As
                                   System.EventArgs) Handles CB.SelectedIndexChanged

    Dim tach As String

    tach = CB.Text.Substring(0, InStr(CB.Text, " |")).Trim

    lenh = "select * from hang where mahg = '" & tach & "'"

    Dim cmd As New OleDbCommand(lenh, con)

    con.Open()

    Dim bang_doc As OleDbDataReader = cmd.ExecuteReader

    Dim dttable As New DataTable("hang")

    dttable.Load(bang_doc, LoadOption.OverwriteChanges)

    con.Close()

    danh_sach = Me.BindingContext(dttable)

    DataGrid1.DataSource = dttable

End Sub

```

```

Private Sub TKIEM_Click(sender As System.Object, e As System.EventArgs) Handles
                                   TKIEM.Click

    Xuat_hang()

End Sub

End Class

```

TÀI LIỆU THAM KHẢO

- [1]. Phạm Hữu Khang chủ biên; Phương Lan, Hoàng Đức Hải, *Kỹ thuật lập trình ứng dụng chuyên nghiệp Visual Basic.net*, Lao động xã hội - Hà Nội, 2004.
- [2]. Phương Lan, *VisualBassic.Net*, Nhà xuất bản Lao động - Xã hội, 2002.
- [3]. Nguyễn Ngọc Tuấn - Hồng Phú, *VisualBassic.Net*, Nhà xuất bản Thống kê, 2004.