

Thực hành với Visual Basic

Biên tập bởi:

Khoa CNTT ĐHSP KT Hưng Yên

Thực hành với Visual Basic

Biên tập bởi:

Khoa CNTT ĐHSP KT Hưng Yên

Các tác giả:

Khoa CNTT ĐHSP KT Hưng Yên

Phiên bản trực tuyến:

<http://voer.edu.vn/c/4942cd3c>

MỤC LỤC

1. Bài thực hành số 1: Cài đặt Visual Basic và môi trường làm việc của VB
 - 1.1. Cài đặt phần mềm Visual Basic
 - 1.2. Chạy chương trình Visual Basic 6.0
 - 1.3. Thiết lập môi trường làm việc
 - 1.4. Lưu dự án (Project) ra đĩa
 - 1.5. Chạy và đóng chương trình Visual Basic (VB)
 - 1.6. Viết lệnh (Code) cho Form để hiển thị lời chào “Hello World”
 - 1.7. Sử dụng câu lệnh Debug.Print
 - 1.8. Sử dụng Câu lệnh InputBox
2. Bài thực hành số 2: Biến, mảng, hàm, thủ tục và các cấu trúc điều khiển
 - 2.1. Khai báo, gán và hiển thị giá trị của các loại biến cơ bản
 - 2.2. Khai báo và sử dụng biến mảng
 - 2.3. Định nghĩa và sử dụng kiểu dữ liệu mới - Kiểu bản ghi
 - 2.4. Định nghĩa Hàm (function) trong Visual Basic
 - 2.5. Định nghĩa thủ tục trong Visual Basic
 - 2.6. Truyền tham trị cho chương trình con
 - 2.7. Truyền tham chiếu cho chương trình con
 - 2.8. Cấu trúc rẽ nhánh If...Then và If ... ElseIf...Then
 - 2.9. Cấu trúc đa rẽ nhánh Select Case
 - 2.10. Cấu trúc lặp For
 - 2.11. Cấu trúc lặp Do ... Loop While | Do ... Loop Until
3. Bài thực hành số 3: Sử dụng các điều khiển cơ bản trong Visual Basic
 - 3.1. Sử dụng TextBox, Label kết hợp với Command Button
 - 3.2. Sử dụng điều khiển Option
 - 3.3. Sử dụng điều khiển CheckBox
 - 3.4. Sử dụng điều khiển ListBox
 - 3.5. Sử dụng điều khiển PictureBox
 - 3.6. Sử dụng điều khiển Image
 - 3.7. Sử dụng HscrollBar (Thanh cuộn ngang)
 - 3.8. Sử dụng điều khiển Timer, Drive, Dir và File
4. Bài thực hành số 4: Sử dụng các hộp thoại
 - 4.1. Sử dụng các hộp thoại
5. Bài thực hành số 5: Sử dụng Menu và các thanh công cụ

- 5.1. Tạo menu có nhiều cấp
 - 5.2. Tạo một Menu ngang có nhiều mục
 - 5.3. Tạo một Menu ngang (Menu bar) đơn giản.
 - 5.4. Viết lệnh cho các mục của menu
 - 5.5. Tạo thanh công cụ Toolbar
 - 5.6. Viết lệnh cho các nút trên thanh công cụ
 - 5.7. Xây dựng chương trình soạn thảo văn bản đơn giản
 - 5.8. Xây dựng chương trình nghe nhạc đơn giản
6. Bài thực hành số 6: Tạo, thao tác với cơ sở dữ liệu và sử dụng các đối tượng
- 6.1. Tạo một bảng CSDL trong Microsoft Access 2000
 - 6.2. Kết nối đến CSDL sử dụng đối tượng ADO Data Control
 - 6.3. Hiển thị bảng CSDL trong Data Grid
 - 6.4. Thêm một bản ghi vào bảng CSDL
 - 6.5. Sửa đổi nội dung của bản ghi
 - 6.6. Tìm kiếm một bản ghi trong bảng
 - 6.7. Loại bỏ (Xoá) một bản ghi khỏi bảng CSDL
 - 6.8. Sử dụng các phương thức của đối tượng RecordSet để duyệt các bản ghi

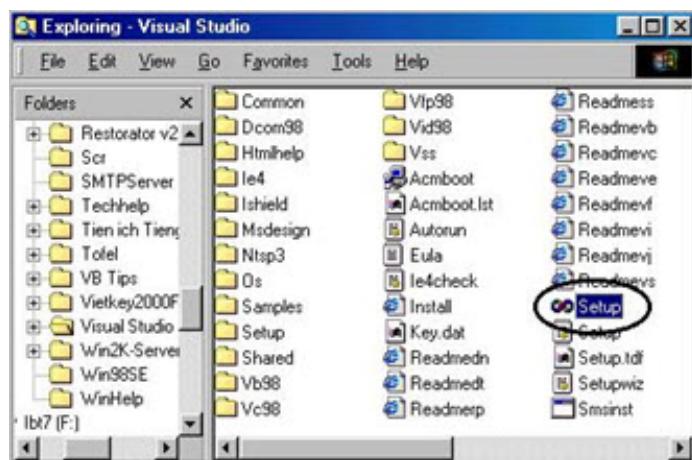
Tham gia đóng góp

Bài thực hành số 1: Cài đặt Visual Basic và môi trường làm việc của VB

Cài đặt phần mềm Visual Basic

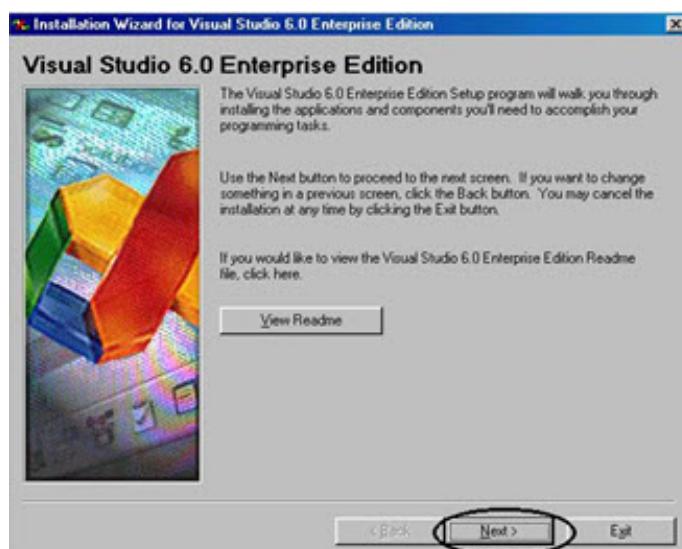
Để cài đặt Visual Basic 6.0 chúng ta cần có bộ Visual Studio 6.0 (hoặc đĩa cài VB riêng) lưu trong đĩa cứng hoặc đĩa CD-ROM. Với bộ Visual Studio cài đặt trên đĩa cứng, Các bước thực hiện như sau :

B1: Tìm và chạy file Setup.exe



Chạy file Setup

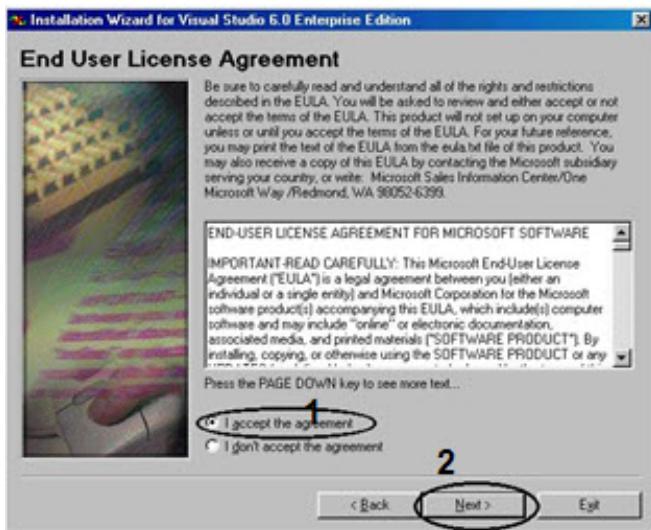
B2: Chọn Next



Chọn Next

B3: Chọn “I Accept the Agreement”, sau đó chọn (click) **Next**

12



Các điều khoản về bản quyền

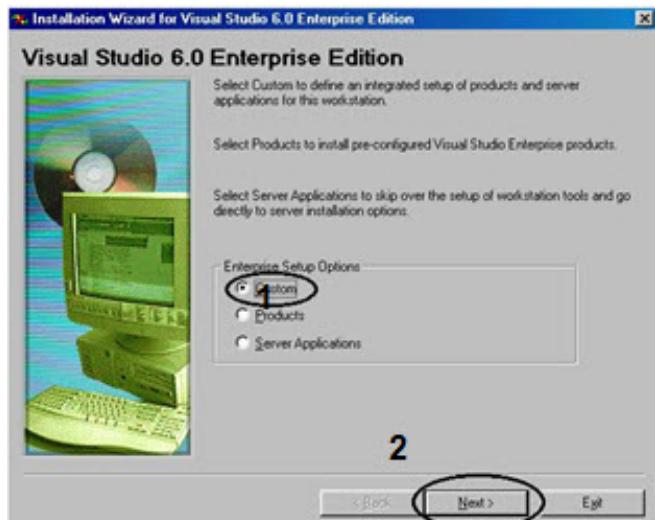
B4: Chọn **Next**



Nhập thông tin đăng ký

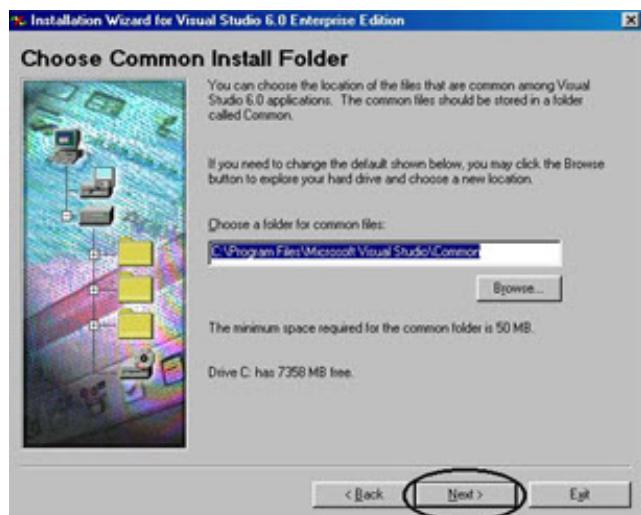
B5: Chọn “Custom” và chọn **Next** (Hoặc có thể chọn Products để cài đặt các sản phẩm riêng biệt – Đây là cách đơn giản nhất)

12



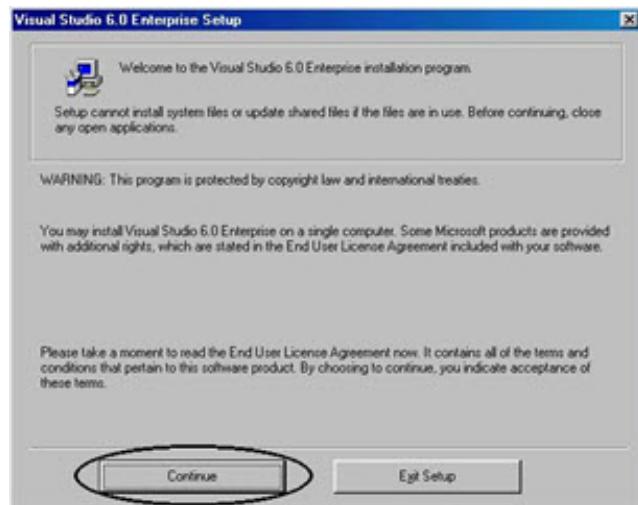
Lựa chọn sản phẩm cần cài đặt

B6: Nếu muốn cài Visual Studio vào thư mục khác, click chọn Browse. Tiếp theo chọn Next



Chọn đường dẫn để cài đặt

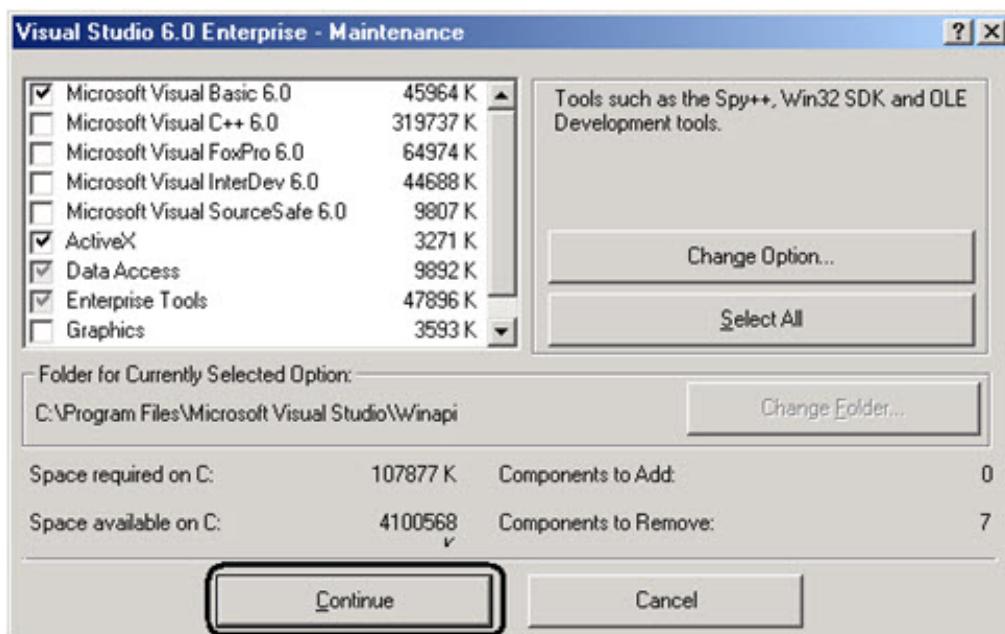
B7: Chọn Continue, Bước tiếp theo chọn OK



Tiến hành cài đặt

B8: Để khôi tồn dung lượng đĩa cứng, nên bỏ các thành phần không cần thiết (bỏ dấu kiểm tra đối với mục không muốn cài đặt) như hình 8 dưới đây:

Sau đó chọn **Continue** và chờ cho quá trình cài đặt kết thúc (Finish).

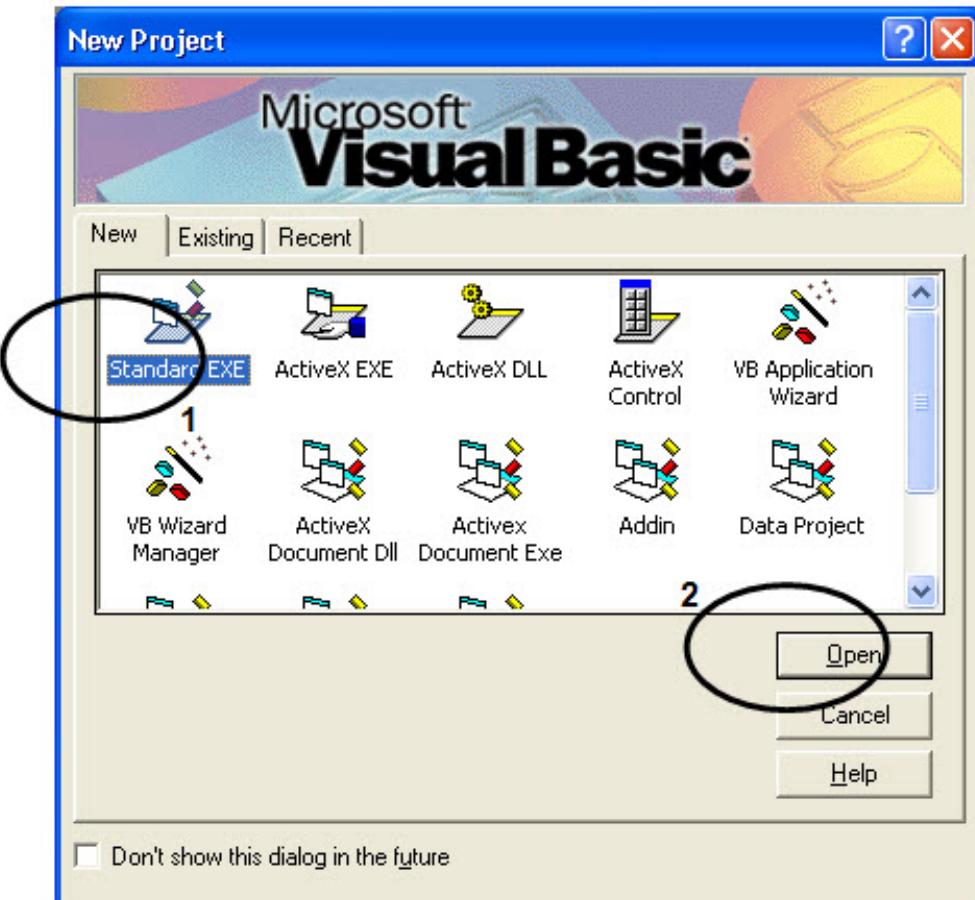


Chọn các thành phần cần cài đặt

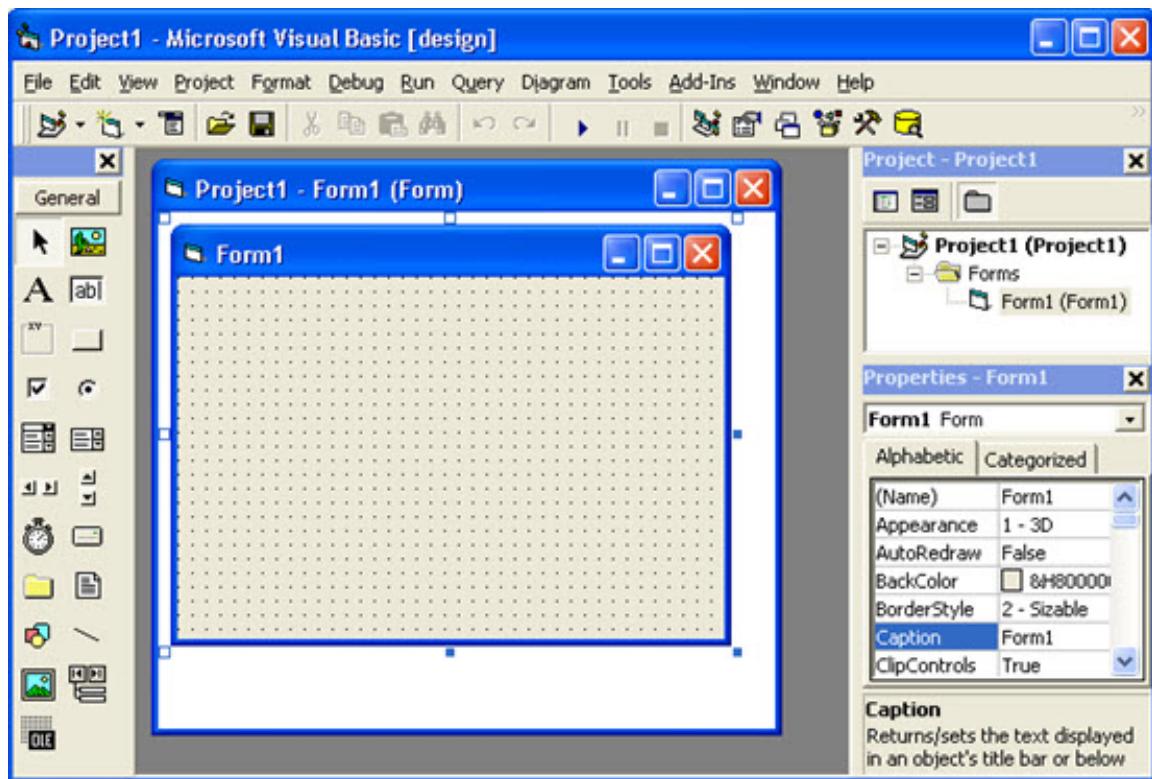
Chạy chương trình Visual Basic 6.0

Click chọn Start→ Programs→ MS Visual Studio 6.0→ MS Visual Basic 6.0. Sau đó chọn kiểu dự án là Standard EXE khi có hộp thoại hiện ra:

21



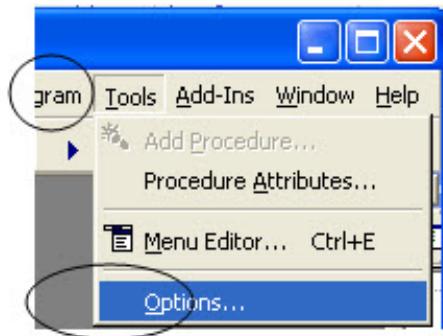
Sau khi nhấn nút Open, thì VB sẽ tạo sẵn cho chúng ta một Project, có giao diện như hình dưới đây :



Cửa sổ giao diện chính của Visual Basic

Thiết lập môi trường làm việc

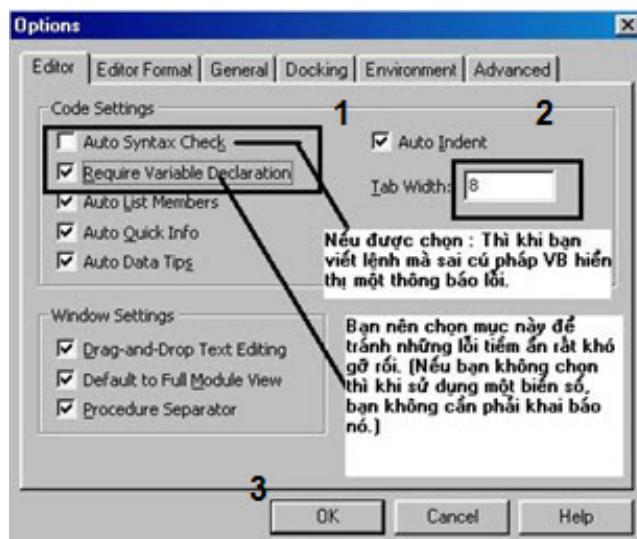
Từ cửa sổ chính, click chọn thực đơn (Menu) Tools, và chọn mục **options**



Đặt các tùy chọn cho môi trường làm việc

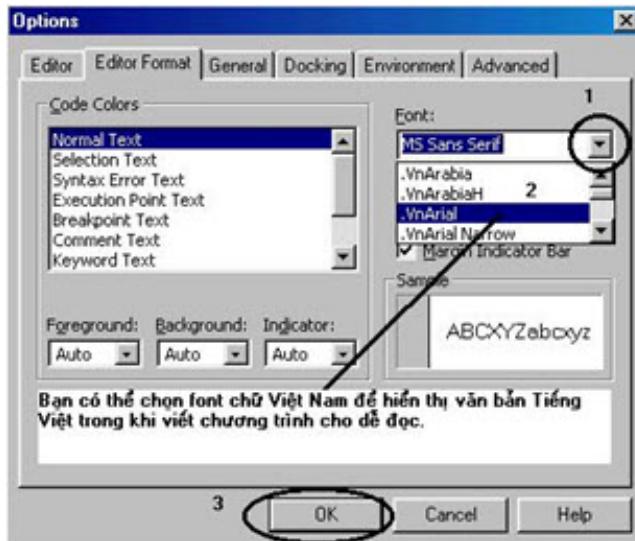
Huỷ lựa chọn tự động kiểm tra cú pháp và đặt độ rộng phím TAB = 8 (hoặc 6 v.v...)

1 2 3



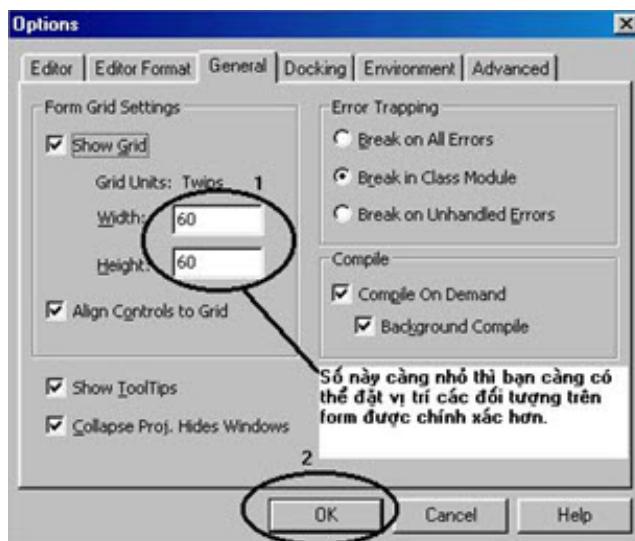
Đặt chế độ kiểm tra cú pháp và yêu cầu khai báo biến

Chọn Font chữ hiển thị cho văn bản chương trình nguồn. Hãy chọn font chữ là font vntime hoặc VK Sans serif.



Chọn font và màu chữ cho văn bản chương trình nguồn

Đặt độ rộng của lưới trên Form :



Đặt độ rộng cho lưới

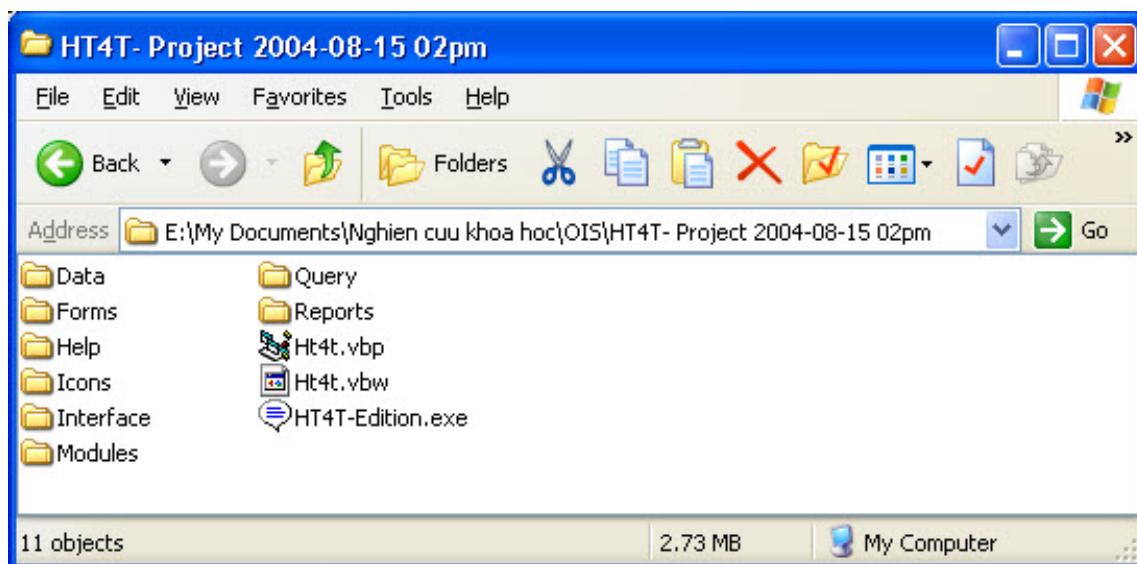
Ngoài ra còn nhiều thiết lập khác nữa, yêu cầu sinh viên tự thực hành !

Lưu dự án (Project) ra đĩa

Để lưu dự án ra đĩa, chọn menu File → Save Project. Hoặc nhấn biểu tượng đĩa mềm trên thanh công cụ.

Lưu ý khi lưu Project: Một project có thể chứa các **Form**, report, module, image, class v.v... Thì mỗi đối tượng này nên lưu vào một thư mục riêng tương ứng như thư mục **Forms**, reports, modules, images, class v.v... Còn riêng file *.vbp (Visual Basic Project) thì được lưu ở thư mục cha.

Dưới đây là một hình ảnh của việc lưu trữ các thành phần của một Project:



Việc tổ chức lưu trữ các thành phần của một dự án

Chạy và đóng chương trình Visual Basic (VB)

Để chạy chương trình, chúng ta có thể vào menu Run → Start (F5) hoặc Run→Start with Full compile (Ctrl + F5)

- **Run → Start**: Chạy chương trình nhưng không dịch toàn bộ chương trình (tức là chương trình chạy đến đâu thì máy dịch tới đó)
- **Run → Start with full compile** : Dịch toàn bộ chương trình trước khi chạy, như vậy nếu có xuất hiện lỗi ở bất cứ đâu trong chương trình thì máy sẽ dừng lại và thông báo lỗi.

Viết lệnh (Code) cho Form để hiển thị lời chào “Hello World”

Hiển thị lời chào trong cửa sổ trung gian (Intermediate Window)

Bước 1: Tạo một Project : Vào menu **Project** → **New Project**, sau đó chọn loại Project là Standard EXE như **Hình 9**.

Bước 2 : Mở cửa sổ soạn thảo lệnh: Vào menu **View** → **Code**

Bước 3: Viết lệnh như sau:



```
Project1 - Form1 (Code)
Form Load
Option Explicit

Sub Form_Load()
    Debug.Print "Hello world"
End Sub
```

Viết lệnh trong cửa sổ Code

Bước 4: Chạy chương trình : Nhấn phím F5 hoặc tổ hợp phím Ctrl-F5 và quan sát kết quả, ta sẽ thấy xuất hiện dòng chữ “Hello world” trong một cửa sổ có tên là Immediate. Có thể hiện cửa sổ này bằng tổ hợp phím Ctrl-G.

Như vậy, lệnh **Debug.print** có chức năng hiển thị kết quả ra màn hình, nó tương tự như lệnh Writeln trong PAscal, printf trong C hay ? trong Foxpro...

Hiển thị lời chào trong hộp thoại - MsgBox.

Các bước thực hiện giống như phần a) nhưng viết lệnh sau thay vì lệnh print:

The screenshot shows the Microsoft Visual Basic IDE with the title bar "Project1 - Form1 (Code)". The main window displays the following VBA code:

```
Option Explicit

Sub Form_Load()
    MsgBox "Hello world"
End Sub
```

Hiển thị lời chào bằng lệnh MsgBox

Nhấn F5 để chạy chương trình, ta có kết quả:



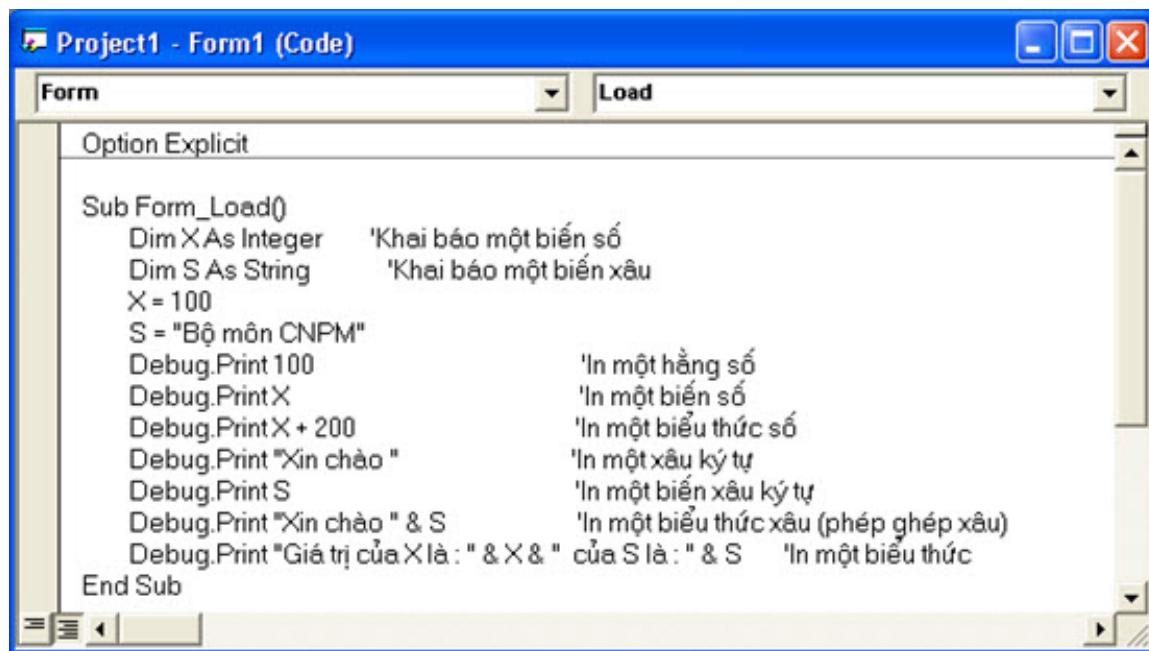
Kết quả chạy chương trình

Như vậy, lệnh **MsgBox** cũng có chức năng hiển thị kết quả ra màn hình giống như `Debug.Print` nhưng trên một cửa sổ (hộp thoại) riêng.

Sử dụng câu lệnh Debug.Print

Lệnh Print dùng để in một biểu thức ra cửa sổ tạm thời (Immediate).

Chương trình sau đây sẽ hiển thị một xâu, một số, một biến số, một xâu với một biến, một biểu thức bất kỳ bằng lệnh Debug.Print.

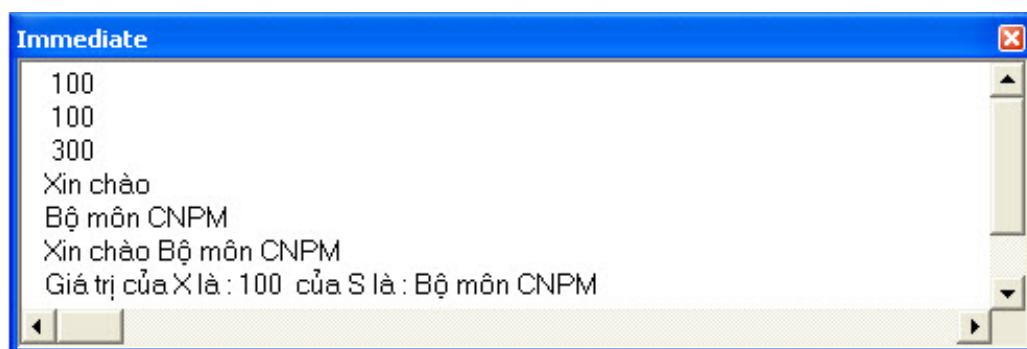


```
Project1 - Form1 (Code)
Form | Load
Option Explicit

Sub Form_Load()
    Dim X As Integer      'Khai báo một biến số
    Dim S As String        'Khai báo một biến xâu
    X = 100
    S = "Bộ môn CNPM"
    Debug.Print 100          'In một hằng số
    Debug.Print X            'In một biến số
    Debug.Print X + 200       'In một biểu thức số
    Debug.Print "Xin chào "
    Debug.Print S            'In một biến xâu ký tự
    Debug.Print S & "         " 'In một biến xâu ký tự
    Debug.Print "Xin chào " & S 'In một biểu thức xâu (phép ghép xâu)
    Debug.Print "Giá trị của X là: " & X & " của S là: " & S 'In một biểu thức
End Sub
```

Các cách sử dụng lệnh Print

Nhấn F5 để chạy chương trình. Cửa sổ Immediate cho ta kết quả như sau:



```
Immediate
100
100
300
Xin chào
Bộ môn CNPM
Xin chào Bộ môn CNPM
Giá trị của X là: 100 của S là: Bộ môn CNPM
```

Kết quả chạy chương trình

Lưu ý: Dấu “&” dùng để ghép các giá trị cần in.

Sử dụng Câu lệnh InputBox

Lệnh (hàm) InputBox có chức năng nhập dữ liệu từ người dùng, tương tự như Readln trong PAscal, scanf trong C, Accept trong Foxpro ... Hàm **InputBox** có thể nhận một **trong 3 tham số quan trọng là: Dòng nhắc “Prompt”, Tiêu đề của hộp thoại “Title” và giá trị mặc định “Default”**. Hàm này trả về giá trị mà người dùng vừa nhập.

- *Thực hành 1* : nhập họ tên của người dùng bằng hàm InputBox

Gõ đoạn lệnh sau vào trong thủ tục **Form_Load** :

SubForm_Load

Dim HoTen **As** String

HoTen = InputBox("Nhập họ tên")

Msgbox "Bạn vừa nhập xâu là : " &HoTen

End Sub

- *Thực hành 2*: Nhập Họ tên và tuổi, sau đó thông báo ra màn hình bằng MsgBox

SubForm_Load

Dim HoTen **As** String

Dim Tuoi **As** Integer

HoTen = InputBox("Nhập họ tên", "Tiêu đề: Nhập thông tin")

Tuoi = InputBox("Tuổi của bạn : ", "Nhập thông tin", 20)

Msgbox "Bạn vừa nhập xâu là : " &HoTen

MsgBox "Tuổi là : " &Tuoi

End Sub

Một số nhận xét:

- Thủ tục **SubForm_Load** tương tự như hàm main trong C hay **Begin ... End**. trong PAscal sẽ được gọi đầu tiên khi chương trình chạy.

- Hàm **MsgBox**, **Debug.Print** và **InputBox** được sử dụng như những lệnh nhập xuất dữ liệu đơn giản nhất trong Visual Basic.

Bài thực hành số 2: Biến, mảng, hàm, thủ tục và các cấu trúc điều khiển

Khai báo, gán và hiển thị giá trị của các loại biến cơ bản

Khai báo biến

- *Thực hành:* Khai báo các biến tương ứng với các kiểu dữ liệu cơ bản trong VB, sau đó gán giá trị và hiển thị giá trị của các biến ra màn hình bằng hàm MsgBox.
- *Hướng dẫn:* Các kiểu dữ liệu cơ bản trong VB bao gồm Byte, Integer, Long, Single, Double, String, Variant...
- *Viết lệnh:* Tạo một Project mới và gõ đoạn lệnh sau trong Form

Option Explicit

Dim ToanCucTrongFormAs Integer

Public ToanCucTrongUngDung As Integer

Private Sub Form_Load()

Dim b As Byte

Dim Bool As Boolean

Dim i As Integer

Dim L As Long

Dim F As Single

Dim D As Double

Dim S As String

Dim S1 As String * 30

Dim V As Variant

App.Title = "Khai báo biến trong Visual Basic"

MsgBox "Biến b, chiếm 1 byte, phạm vi biểu diễn 0-255"

MsgBox "Biến Bool, 2 byte, biểu diễn giá trị True và False"

MsgBox "Biến i, 2 byte, phạm vi: -32768 ... +32767"

MsgBox "Biến L, chiếm 4 byte, phạm vi: -2,147,483,648 đến 2,147,483,647 "

MsgBox "Biến F, chiếm 4 byte, biểu diễn số thực âm và dương"

MsgBox "Biến D, chiếm 8 byte, biểu diễn số thực âm và dương rất lớn"

MsgBox "Biến S, chiếm 10+độ dài của xâu. có thể lưu tối 2 tý ký tự"

MsgBox "Biến S1 là biến xâu có độ dài cố định (trường hợp này là 30)." & _

"có thể lưu tối đa khoảng 65400 ký tự"

MsgBox "Biến V là biến Variant, chiếm 16 byte. Nó có thể lưu bất kỳ loại giá trị nào"

End sub

Giải thích thêm:

- Biến *ToanCucTrongForm* (Toàn cục trong **Form**) được khai báo với từ khoá **Dim** là biến có thể sử dụng ở bất kỳ đâu trong chính **Form** nó được khai báo.
- Biến *ToanCucTrongUngDung* (tất cả các biến trong toàn ứng dụng) được khai báo với từ khoá **Public** có thể sử dụng ở bất kỳ **Form** nào trong toàn ứng dụng.
- Các biến khai báo trong **SubForm_Load** được gọi là các biến cục bộ trong thủ tục, chúng chỉ được sử dụng bên trong chính thủ tục đó mà thôi.
- Đối với các biến thuộc kiểu Variant thì khi khai báo chỉ cần viết, ví dụ: **Dim V**
- Khai báo **Dim V, S As String** tương đương với : **Dim V As Variant, S As String** (V sẽ có kiểu là Variant chứ không phải là string !!!).
- Biến kiểu Variant không được hỗ trợ trong phiên bản VB.NET !.
- Dấu & _ (Có 1 dấu cách giữa dấu & và dấu _) cho phép ngắt câu lệnh trên nhiều dòng.

Gán và hiển thị giá trị của các biến

Thực hành: Gán giá trị cho các biến và hiển thị ra màn hình

Viết lệnh: Gõ đoạn code sau vào trong **Form**

Option Explicit

```
Dim ToanCucTrongFormAsInteger
Public ToanCucTrongUngDung AsInteger
PrivateSubForm_Load()
Dim b As Byte
Dim Bool As Boolean
Dim i AsInteger
Dim L As Long
Dim F As Single
Dim D As Double
Dim S As String
Dim S1 As String * 30
Dim V As Variant
App.Title = "Khai báo biến trong Visual Basic"
b = 100
Bool = True
i = 30000
L = 500000
F = 123.456
D = 1.5E+30
S = "Khoa Công nghệ Thông tin"
S1 = "Bộ môn Công nghệ phần mềm"
V = 10000
```

ToanCucTrong**Form** = 10

ToanCucTrongUngDung = 1000

MsgBox "b=" & b

MsgBox "Bool=" & Bool

MsgBox "i=" & i

MsgBox "L = " & L

MsgBox "F=" & F

MsgBox "D=" & D

MsgBox "S=" & S

MsgBox "S1=" & S1

MsgBox "V=" & V

MsgBox "ToanCucTrong**Form**=" & ToanCucTrong**Form**

MsgBox "ToanCucTrongUngDung=" & ToanCucTrongUngDung

End sub

Khai báo và sử dụng biến mảng

Khai báo hai biến mảng để lưu danh sách họ tên và điểm của 100 SV.

Viết lệnh: Hãy gõ đoạn code sau vào trong **Form**:

Option Explicit

Dim HT(100) As String '/// Mảng chứa được 101 phần tử từ 0 đến 100

Dim Diem(1 To 100) As Single '/// Mảng chứa được 100 phần tử từ 1 đến 100

Dim MaTran1(4, 4) As Single '/// Ma trận (mảng 2 chiều) có 5 hàng 5 cột

Dim MaTran2(1 To 4, 1 To 4) As Single '/// Mảng 2 chiều có 4 hàng, 4 cột

Private Sub Form_Load()

HT(0) = "Bill"

HT(1) = "John"

HT(2) = "Gorge"

Diem(1) = 7

Diem(2) = 9

Diem(3) = 10

MaTran1(0, 0) = 5

MaTran1(0, 1) = 6

MaTran1(0, 4) = 8

MaTran2(1, 1) = 8

MaTran2(1, 2) = 9

MaTran2(4, 4) = 10

MsgBox "Giá trị của phần tử đầu tiên của mảng HT là :" & HT(0)

```
MsgBox "Diem(2) = " & Diem(2)
```

```
MsgBox "MaTran1(0,4)=" & MaTran1(0, 4)
```

```
MsgBox "MaTran2(4,4) = " & MaTran2(4, 4)
```

```
End sub
```

- Lưu ý:
- Nếu khi khai báo mảng mà không chỉ rõ cận dưới (không có từ khoá **To**) thì mặc định VB sẽ lấy cả phần tử có chỉ số là 0.
- Để truy cập đến một phần tử của mảng thì viết tên mảng kèm thêm chỉ số đặt trong cặp ngoặc đơn.
- **Lbound(M)** (**LBound** = Lower Bound = Cận dưới) cho biết chỉ số dưới của mảng M. Ví dụ **LBound(Diem)** cho ta 1. **LBound(MaTran1)** cho ta 0.
- **Ubound(M)** (**UBound** = Upper Bound = Cận trên) cho ta chỉ số trên của mảng M. **UBound(Diem)** → 100.
- Khi khai báo một biến mảng mà không chỉ rõ số phần tử, ví dụ: **Dim D() AsInteger** thì D được gọi là một mảng động (Dynamic array).
- Đối với mảng động, ta có thể thay đổi lại số phần tử của mảng bằng câu lệnh **Redim**. Ví dụ, xin 50 phần tử lưu trữ cho mảng D bằng cách viết : **Redim D(50)**.

Định nghĩa và sử dụng kiểu dữ liệu mới - Kiểu bản ghi

Thực hành: Định nghĩa kiểu dữ liệu mới để biến thuộc kiểu dữ liệu này có thể lưu trữ được các thông tin về một cuốn sách (Tên sách, Tên tác giả, năm xuất bản, giá).

Hướng dẫn: Kiểu dữ liệu mới nên định nghĩa trong Module, còn nếu định nghĩa trong **Form** thì chỉ có thể ở dạng **Private** (tức chỉ sử dụng cục bộ trong **Form**) mà không thể ở dạng **Public** (Sử dụng trong mọi **Form**).

Mình họa: Tạo module mới: Vào menu **Project** → **Add Module**. Lưu module này với tên : **modDataTypes.bas**

Gõ đoạn lệnh dưới đây vào trong module vừa tạo:

Option Explicit

'// Định nghĩa kiểu dữ liệu mới : KieuSach

Public Type KieuSach

TenSach **As String**

TacGia **As String**

NamXB **As Integer**

Gia **As Single**

End Type

Gõ đoạn lệnh dưới đây trong thủ tục **Form_Load**:

Option Explicit

Dim Sach **As KieuSach**

Dim KhoSach(100) **As KieuSach**

Private Sub Form_Load()

Sach.TenSach = "Lập trình VB thật là đơn giản"

Sach.TacGia = "Software Team - UTEHY"

```
Sach.NamXB = 2006
```

```
Sach.Gia = 45000
```

```
'// Gán một số giá trị cho phần tử có chỉ số là 1 cho mảng
```

```
KhoSach(1).TenSach = "Bài tập Visual Basic"
```

```
KhoSach(1).TacGia = "Software Team - UTEHY"
```

```
KhoSach(1).NamXB = 2006
```

```
KhoSach(1).Gia = 34500
```

```
MsgBox Sach.TenSach & " giá : " & Sach.Gia
```

```
MsgBox KhoSach(1).TenSach & " Giá : " & KhoSach(1).Gia
```

```
End sub
```

- **Ghi chú:**

- Đoạn chương trình trên định nghĩa kiểu bản ghi (tương tự như **Record** trong Pascal hay **struct** trong C/C++) bằng câu lệnh **Type**, từ khoá **Public** đứng trước để chỉ ra rằng kiểu dữ liệu này có thể được dùng trong mọi **Form**, mọi module. Còn nếu sử dụng từ khoá **Private** thay vì **Public** thì kiểu dữ liệu mới này chỉ được sử dụng trong chính module đó mà thôi.
- Đoạn code tiếp theo khai báo 2 biến thuộc kiểu dữ liệu vừa định nghĩa. Một là biến thông thường, biến thứ hai là một mảng.

? Kiểu dữ liệu mảng thường được thao tác kết hợp với vòng lặp. Các ví dụ thêm về mảng kết hợp với vòng lặp sẽ được đề cập ở các phần tiếp sau.

Định nghĩa Hàm (function) trong Visual Basic

Hàm và thủ tục được gọi là những chương trình con, giúp cho chương trình dễ bảo trì, dễ hiểu và tránh phải viết lại những đoạn lệnh tương tự nhau.

Thực hành: Định nghĩa hàm tính tổng của 2 số nguyên, kết quả được trả lại (gán) về cho hàm.

Hướng dẫn: Vì hàm cần tính tổng của 2 số nguyên nên số tham số đầu vào là 2, kiểu của tham số đầu vào là **Integer**, và vì chỉ cần lấy giá trị của tham số vào mà **không có nhu cầu thay đổi giá trị** của nó do vậy ta sẽ khai báo 2 tham số của hàm **thuộc dạng tham trị**.

Minh họa:

Option Explicit

'// Hàm tính tổng của hai số nguyên, 2 tham số truyền vào dưới dạng tham trị

Function Tong(byVal a **AsInteger**, byVal b **AsInteger**) **As Long**

Dim S **As Long**

S = a + b

Tong = S '//' Gán kết quả cho hàm

End Function

'// Sử dụng hàm vừa tạo

PrivateSubForm_Load()

Dim X **AsInteger**, Y **AsInteger**, Z **As Long**

X = 5

Y = 10

Tong 10,20 '//' Gọi hàm **Tong** độc lập

Z = Tong(X,Y) '// Gọi hàm **Tong** và gán KQ cho Z

```
MsgBox "Tổng là : " & Tong(10, 20) '// Gọi hàm Tong
```

```
End sub
```

Chú ý:

- Khi định nghĩa hàm, nếu trước các tham số hình thức (tham số a, b ở trên) mà không có từ khóa byVal thì VB sẽ hiểu là tham số đó ở dạng tham chiếu (tham biến) mà ta sẽ nói sau.
- Hàm thì có thể gọi độc lập (ví dụ : Tong 10,20), khi đó các tham số không được đặt trong cặp ngoặc đơn. Còn nếu hàm tham gia vào biểu thức hay câu lệnh khác (2 cách gọi còn lại ở trên) thì các tham số phải được đặt trong cặp ngoặc đơn.
- Việc gán kết quả cho tên hàm được gọi là trả kết quả về cho hàm.

Định nghĩa thủ tục trong Visual Basic

Khi tính tổng của 2 số nguyên như phần trên thì ta cần lấy kết quả trả về là tổng của chúng, khi tính sin(x) thì ta cần kết quả trả về là sin của số x... lúc đó ta cần phải định nghĩa hàm. Tuy nhiên, trong một số trường hợp khi viết các chương trình con, nếu không cần phải kết quả trả về từ chương trình con, lúc đó ta nên định nghĩa chương trình con đó ở dạng thủ tục (**Sub**).

Thực hành: Hiển thị ngày tháng năm, giờ phút giây hiện tại trong máy tính.

Hướng dẫn: Hàm Now cho ta biết thông tin về ngày/tháng/năm và giờ/phút/giây hiện tại trong máy tính. Muốn trích riêng ngày, tháng, năm, giờ, phút, giây thì dùng các hàm tương ứng là Year, Month, day, hour, minute, second....

Viết lệnh:

Option Explicit

'// Thủ tục hiển thị thời gian hiện tại trong máy tính

Private Sub ThoiGian()

Dim D As Date // Khai báo một biến kiểu Date/Time

D = Now // Lấy ngày, tháng, năm, giờ, phút, giây hiện hành trong máy tính

MsgBox "Hôm nay là ngày " & Day(D) & " tháng " & Month(D) & " năm " & Year(D)

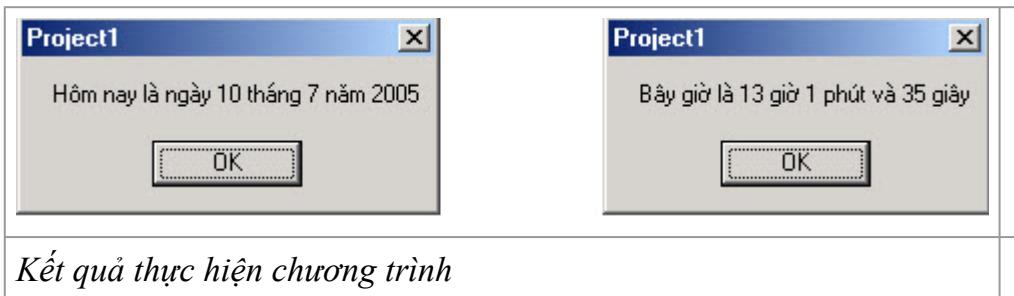
MsgBox "Bây giờ là " & Hour(D) & " giờ " & Minute(D) & " phút và " & Second(D) & " giây"

End sub

Private Sub Form_Load()

Call ThoiGian // Gọi thủ tục ThoiGian (Có thể bỏ qua từ khoá Call)

End sub



Kết quả thực hiện chương trình

Chú thích:

- Vì công việc ở trên chỉ đơn thuần là hiển thị thời gian hiện có trong máy tính mà không cần lấy giá trị trả về từ chương trình con, do vậy chương trình con được định nghĩa ở đây nên là dạng thủ tục (**Sub**).
- Việc gọi độc lập hàm hay thủ tục có thể kèm thêm từ khoá Call, nhưng không bắt buộc (Từ khoá Call được VB giữ lại từ phiên bản Basic **for Dos**). Tuy nhiên khi chúng ta gọi chương trình con với từ khoá Call thì các tham số (nếu có) bắt buộc phải đặt trong cặp ngoặc đơn.
- Câu lệnh Date = <Thời gian mới> và Time = <Giờ phút mới> để đặt lại thời gian của máy tính. Ví dụ: Date = #February 12, 1985# và Time = #4:35:17 PM# để thay đổi thời gian trong máy tính.

Truyền tham trị cho chương trình con

Khi ta có một biến số và truyền biến số này cho một chương trình con nhưng ta không muốn giá trị trong biến số này bị thay đổi khi gọi chương trình con đó thì lúc định nghĩa chương trình con, tham số hình thức tương ứng phải ở dạng tham trị (có từ khoá byVal đúng trước).

Thực hành: Viết chương trình tăng giá trị của một số lên 1 đơn vị và hiển thị.

Hướng dẫn: Hãy gõ đoạn lệnh dưới đây và chạy, bạn sẽ quan sát thấy rằng, giá trị của biến số X sẽ không bị thay đổi giá trị sau khi kết thúc gọi thủ tục Tang, mặc dù trong chương trình có làm thay đổi giá trị của tham số truyền vào. Tham số truyền vào chỉ bị thay đổi tạm thời trong thủ tục đó mà thôi, bởi vì tham số ta khai báo ở dạng THAM TRỊ.

Viết lệnh:

Option Explicit

'// Khai báo thủ tục với tham số ở dạng tham trị (Có từ khoá byVal)

PrivateSub Tang(**ByVal** a **AsInteger**)

a = a + 1

MsgBox "Giá trị của tham số trong thủ tục là :" & a

End sub

PrivateSubForm_Load()

Dim X **AsInteger**

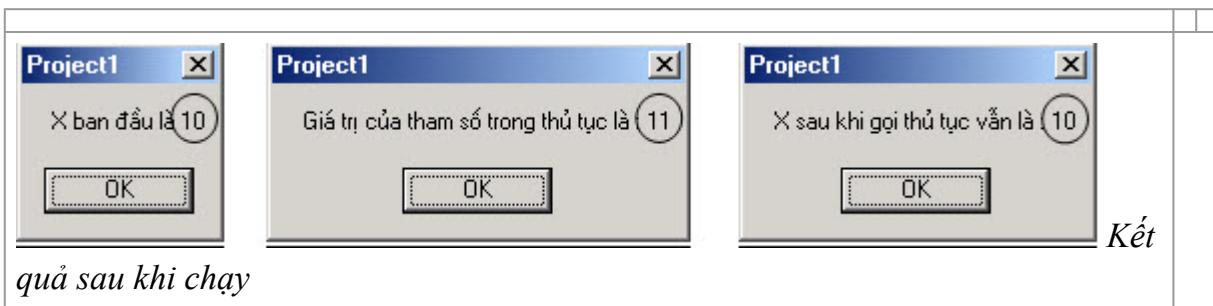
X = 10

MsgBox "X ban đầu là " & X

Tang X

MsgBox "X sau khi gọi thủ tục vẫn là :" & X

End sub



Kết luận:

- Khi tham số hình thức khai báo ở dạng tham trị thì tham số thực sự truyền vào sẽ không bị thay đổi bởi chương trình con đó (cho dù trong chương trình con có lệnh làm thay đổi tham số truyền vào).
- Khi khai báo tham số ở dạng tham trị thì tham số thực sự truyền vào có thể là hằng số, biến số hay biểu thức....

Truyền tham chiếu cho chương trình con

Không giống như truyền theo tham trị ở phần 5, nếu một chương trình con có làm thay đổi giá trị của tham số thực sự truyền vào và ta **muốn giữ lại sự thay đổi** này thì cần khai báo tham số hình thức ở dạng **tham chiếu**.

Thực hành: Viết chương trình tăng giá trị của một số lên 1 đơn vị và hiển thị.

Hướng dẫn: Viết chương trình giống như phần 5, nhưng thay byVal bởi byRef (hoặc có thể xoá từ khoá byVal đứng trước)

Option Explicit

'/// Khai báo thủ tục với tham số ở dạng tham trị (Có từ khoá byVal)

PrivateSub Tang(**ByRef** a **AsInteger**)

a = a + 1

MsgBox "Giá trị của tham số trong thủ tục là : " & a

End sub

PrivateSubForm_Load()

Dim X AsInteger

X = 10

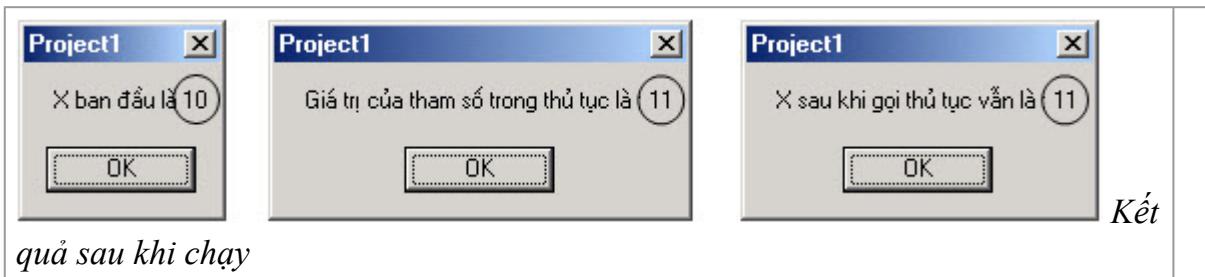
MsgBox "Giá trị của biến số X ban đầu là : " & X

Tang X

MsgBox "Giá trị của biến số X sau khi gọi thủ tục vẫn là : " & X

End sub





- *Qui tắc chung khi khai báo các tham số và chương trình con:*
- Nếu có sự thay đổi giá trị của các tham số truyền vào chương trình con thì tham số hình thức được khai báo sẽ ở dạng tham trị (có từ khoá byVal đúng trước).
- Nếu có nhu cầu thay đổi tham số truyền vào cho chương trình con thì tham số truyền vào sẽ ở dạng tham chiếu (có từ khoá byRef đúng trước, mặc định không có từ khoá byVal hay byRef thì được coi là byRef).
- Chương trình con cần bao nhiêu Dữ kiện đầu vào thì mới giải quyết được Thực hành đặt ra thì sẽ khai báo bấy nhiêu tham số tương ứng.
- Các tham số có thể khai báo là các tham số ngầm định (tức khi gọi chương trình con thì có thể bỏ qua tham số này).

Ví dụ:

Sub ThongBao(Optional TB **As** String = "Hôm nay không có gì để Thông báo, nếu có gì cần thông báo, chúng tôi sẽ ...")

MsgBox TB

End sub

- Hàm thì có thể tham gia vào biểu thức tính toán, thủ tục thì không.

Cấu trúc rẽ nhánh If...Then và If ... ElseIf...Then

Cấu trúc **IF...Then** dùng để thực hiện việc thay đổi luồng thực thi của chương trình tùy vào điều kiện đang được xét.

Thực hành: Giải phương trình bậc hai sử dụng cấu trúc **IF**.

Hướng dẫn: Chương trình giải phương trình bậc hai có thể viết trực tiếp ngay bên trong thủ tục **Form_load** hoặc có thể viết trong một chương trình con riêng. Ở đây ta viết trong một thủ tục và thủ tục này sẽ được gọi trong thủ tục chính **Form_Load**.

Chú thích:

- Toán tử \wedge sử dụng để tính số mũ của một số. Ví dụ: x^5 , $a \wedge b$
- Thủ tục GiaiPT ở trên chưa phải là thủ tục viết tối ưu nhất (nhằm mục đích cho đơn giản). Chương trình con giải phương trình bậc 2 đúng nhất nếu viết theo thủ tục thì phải khai báo là: **Sub** GiaiPT(ByVal a **As** Single, byVal b **As** Single, byVal c **As** Single, x1 **As** Single, x2 **As** Single, CóNghiệmHayKhông **As Boolean**). Còn nếu viết theo hàm thì phải khai báo là: **Function** GiaiPT(ByVal a **As** Single, byVal b **As** Single, byVal c **As** Single, x1 **As** Single, x2 **As** Single) **As Boolean**. Ở đây ta giảm đi một tham số vì dấu hiệu có nghiệm hay không được trả về từ hàm. 2 cách khai báo này coi như là một bài tập về nhà để bạn đọc tự làm !.
- Có thể viết câu lệnh **If** theo dạng : **If** <Đkiện> **Then** <Câu_Lệnh>

Cấu trúc đa rẽ nhánh Select Case

Thực hành: Viết chương trình hiển thị thứ trong tuần ứng với mỗi số người dùng nhập từ bàn phím.

Viết lệnh:

PrivateSubForm_Load()

Dim Thu As Integer

Thu = InputBox("Nhập vào một con số : ", "Sử dụng cấu trúc SelectCase", 2)

Select Case Thu

Case 2

MsgBox "Thứ 2"

Case 3

MsgBox "Thứ 3"

Case 4: MsgBox "Thứ 4"

Case 5: MsgBox "Thứ 5"

Case 6: MsgBox "Thứ 6"

Case 7: MsgBox "Thứ 7"

Case 8: MsgBox "Chủ nhật. Chúc bạn một ngày vui vẻ !"

Case Else

MsgBox "Bạn phải nhập đúng (2--> 8)", vbCritical, "Khuyến cáo !"

End Select

End Sub

Ghi chú:

- Biểu thức để kiểm tra (đứng sau **SelectCase**) có thể là một biểu thức **số thực**, **số nguyên**, **ký tự**, **xâu ký tự** v.v...
- Có thể viết nhiều câu lệnh trên cùng một dòng bằng cách thêm vào dấu hai chấm ":" giữa các câu lệnh (Như các phần **Case** ở trên)

Thực hành: Viết chương trình tra cứu từ điển tin học Anh-Việt. Người dùng gõ từ Tiếng Anh, chương trình sẽ hiển thị nghĩa tiếng Việt của từ đó.

Viết lệnh:

PrivateSubForm_Load()

Dim TiengAnh As String

TiengAnh = InputBox("Nhập từ tiếng Anh: ", "Sử dụng cấu trúc Select", "computer")

SelectCase TiengAnh

Case "computer"

MsgBox "Nghĩa của computer là Máy tính"

Case "ram"

MsgBox "Nghĩa của ram là Bộ nhớ trong"

Case "cpu", "CPU"

MsgBox "Nghĩa của cpu là: Bộ xử lý trung tâm"

Case Else

MsgBox "Xin lỗi, từ này chưa có trong từ điển", vbInformation

End Select

End Sub

c) Sử dụng từ khoá **is** để kiểm tra biểu thức thuộc một miền giá trị. Đây là tính năng rất mạnh của cấu trúc **SelectCase** so với các ngôn ngữ khác.

Thực hành: Giải phương trình bậc hai sử dụng cấu trúc **SelectCase** (Đối với các ngôn ngữ khác như PAscal hay C/C++ thì không hỗ trợ cách thức này)

Option Explicit

```
' ///////////////////Chương trình con giải PT bậc 2           ///////////////////////////////
Private Sub GiaiPT()
    Dim A As Single, B As Single, C As Single, Delta As Single
    Dim X1 As Single, X2 As Single

    A = InputBox("Nhập hệ số A (a<>0) ")
    B = InputBox("Nhập hệ số B")
    C = InputBox("Nhập hệ số C")
    Delta = B ^ 2 - 4 * A * C

    Select Case Delta
        Case Is < 0
            MsgBox "Phương trình vô nghiệm !"
        Case 0
            MsgBox "Phương trình có nghiệm kép là X1 = X2 = " & (-B / 2 / A)
        Case Else
            X1 = (-B + Sqr(Delta)) / 2 / A
            X2 = (-B - Sqr(Delta)) / 2 / A
            MsgBox "Phương trình có 2 nghiệm là X1=" & X1 & " X2 =" & X2
    End Select
End Sub

Sub Form_Load
    GiaiPT          '/// Gọi thủ tục GiaiPT
End Sub
```

Cấu trúc lặp For

Vòng lặp **for** dùng để lặp đi lặp lại một khối lệnh với số lần lặp xác định.

Thực hành: Viết chương trình tính tổng của N số tự nhiên đầu tiên, số N nhập từ bàn phím.

Hướng dẫn: Sử dụng hàm InputBox để nhập số N, Sau đó cộng dồn các số i (i chạy từ 1 đến N) vào tổng S. Vì việc cộng lặp lại với số lần lặp cố định là N do vậy ta sử dụng vòng lặp **For**.

Viết lệnh :

Option Explicit

PrivateSubForm_Load()

Dim i AsInteger, N As Integer, S As Long

N = InputBox("Nhập vào số N : ")

S = 0

For i = 1 To N

S = S + i

Next

MsgBox "Tổng của " & N & " số tự nhiên đầu tiên là : " & S

End Sub

Thực hành: Tính tổng của Các số chẵn trong khoảng từ 1 đến N. N nhập từ bàn phím.

Hướng dẫn: Có nhiều cách tính, nhưng ở đây cần tận dụng thêm điều khoản step trong cấu trúc lặp **for** như sau:

Viết lệnh:

Option Explicit

PrivateSubForm_Load()

Dim i AsInteger, N As Integer, S As Long

N = InputBox("Nhập vào số N : ")

S = 0

For i = 2 To N Step 2

S = S + i

Next

MsgBox "Tổng của các số chẵn trong khoảng từ 2 đến " & N & " là : " & S

End Sub

Chú thích

- Khi không có điều khoản step thì mặc định sau mỗi vòng lặp biến chạy sẽ tự động được tăng lên 1 đơn vị.
- Muốn biến chạy bị giảm đi 1 đơn vị (giống như **downto** trong PASCAL) thì ta thêm điều khoản **Step -1** sau lệnh **for**.
- Cận trên và cận dưới của biến chạy có thể là số nguyên, số thực.

Cấu trúc lặp Do ... Loop While | Do ... Loop Until

Trái với cấu trúc lặp Do while là kiểm tra điều kiện trước khi thực hiện công việc, Cấu trúc lặp **Do ... Loop While** và **Do ... Loop Until** lại thực hiện công việc sau đó mới kiểm tra điều kiện lặp (Cấu trúc Do...Loop While giống như cấu trúc **Repeat ... Until** trong PAscal).

Thực hành: Viết chương trình nhập vào một danh sách gồm Họ tên và điểm thi của sinh viên trong một lớp. Việc nhập kết thúc nếu họ tên nhập vào là một xâu rỗng.

Hướng dẫn:

- Công việc nhập lặp đi lặp lại do vậy cần sử dụng vòng lặp
- Số lần lặp là không xác định nên không thể sử dụng vòng lặp **for**
- Việc kiểm tra được tiến hành sau khi người dùng nhập họ tên, do vậy cần dùng vòng lặp **Do ... Loop While**.

Viết lệnh (sử dụng cấu trúc Do ... Loop While):

Option Explicit

'/// Định nghĩa kiểu bản ghi (Kiểu dữ liệu mới)

'/// *Kiểu bản ghi định nghĩa trong Form phải có thuộc tính là Private (không được là public)*

PrivateType Kieu_SinhVien

HoTen **As** String

Diem **As** Integer

End Type

Dim N **As** Integer '/// Dùng để lưu số lượng phần tử đã nhập vào

PrivateSubForm_Load()

Dim SinhVien(100) **As** Kieu_SinhVien

N = 0

Do

N = N + 1

SinhVien(N).HoTen = InputBox("Họ tên của sinh viên thứ " & N)

SinhVien(N).Diem = InputBox("Điểm của sinh viên thứ " & N)

Loop While SinhVien(N).HoTen <> ""

'/// Hiển thị kết quả vừa nhập

Dim i **As** Integer

For i = 1 To N - 1

MsgBox "Họ tên : " & SinhVien(i).HoTen & " Điểm: " & SinhVien(i).Diem

Next

End Sub

Viết lệnh (sử dụng cấu trúc Do ... Loop Until):

Option Explicit

'// Định nghĩa kiểu bản ghi (Kiểu dữ liệu mới)

'// Kiểu bản ghi định nghĩa trong Form phải có thuộc tính là Private (không được là public)

PrivateType Kieu_SinhVien

HoTen **As** String

Diem **As** Integer

End Type

Dim N **As** Integer // Dùng để lưu số lượng phần tử đã nhập vào

PrivateSubForm_Load()

Dim SinhVien(100) **As** Kieu_SinhVien

N = 0

Do

N = N + 1

SinhVien(N).HoTen = InputBox("Họ tên của sinh viên thứ " & N)

SinhVien(N).Diem = InputBox("Điểm của sinh viên thứ " & N)

Loop Until SinhVien(N).HoTen = ""

'/// Hiển thị kết quả vừa nhập

Dim i As Integer

For i = 1 To N - 1

Debug.Print "Họ tên : " & SinhVien(i).HoTen & " Điểm: " & SinhVien(i).Diem

Next

End Sub

Nhận xét:

- Cấu trúc Do ... Loop While và cấu trúc Do ... Loop Until đều thực hiện công việc lặp sau đó mới kiểm tra điều kiện nhưng trong cấu trúc lặp Do ... Loop While thì việc lặp chỉ kết thúc nếu điều kiện là sai, còn Do ... Loop Until thì kết thúc nếu điều kiện lặp là đúng (giống với Repeat until trong PASCAL)

Bài thực hành số 3: Sử dụng các điều khiển cơ bản trong Visual Basic

Sử dụng TextBox, Label kết hợp với Command Button

- TextBox là một điều khiển cho phép hiển thị thông tin đồng thời cho phép sửa đổi thông tin trực tiếp bởi người sử dụng.
- Label là điều khiển chỉ cho phép hiển thị thông tin mà người dùng không thể sửa đổi hay tương tác. Nó thường được sử dụng mang tính chất chú thích cho các điều khiển khác đối với người dùng.
- Command button là một điều khiển thường dùng để thực thi một công việc khi người dùng click chuột. (Các điều khiển khác cũng làm được như vậy nhưng nút lệnh command thì trực quan và chuẩn tắc hơn).

Thực hành: Thiết kế chương trình như hình vẽ với các yêu cầu:

- Khi click vào nút "Hiển thị" thì hiển thị nội dung trong textbox
- Khi click vào nút "Xoá" thì xoá nội dung trong textbox
- Khi click vào nút "Cấm" thì cấm người dùng soạn thảo trong textbox
- Khi click vào nút "Soạn thảo" thì được phép soạn trên textbox
- Khi click vào nút "Ẩn" thì ẩn Textbox, Click vào nút "Hiện" thì hiện text
- Click vào nút "Thoát" để kết thúc chương trình.

Thiết kế giao diện: Như hình dưới



Giao diện chương trình

Đặt tên các điều khiển.

Loại điều khiển	Tên điều khiển (Name)	Caption (nhãn)
Form	frmMain	Sử dụng các điều khiển cơ bản
Label	lblHuongDan	Hãy gõ nội dung...

Textbox	txtMsg	txtMsg (Thuộc tính Text)
Command button	cmdHienThi	&Hiển thị
Command button	cmdCam	&Cấm
Command button	cmdSoanThao	&Soạn
Command button	cmdAn	Ấn
Command button	cmdHien	Hiện
Command button	cmdThoat	Thoát

Viết lệnh trong Form như sau:

Option Explicit

Private Sub cmdAn_Click()

txtMsg.Visible = False '/// Ấn hộp textbox

End Sub

Private Sub cmdCam_Click()

txtMsg.Enabled = False '/// Cấm người dùng tương tác (copy, sửa...) với textbox

End Sub

Private Sub cmdHien_Click()

txtMsg.Visible = True '/// Cho hiện hộp textbox

End Sub

Private Sub cmdHienThi_Click()

MsgBox "Nội dung trong textbox : " & txtMsg.Text

End Sub

Private Sub cmdSoan_Click()

txtMsg.Enabled = True '/// Cho phép người dùng tương tác với textbox

End Sub

Private Sub cmdXoa_Click()

txtMsg.Text = ""'/// Gán nội dung là xâu rỗng, tương đương với việc xoá

End Sub

Private Sub cmdThoat_Click()

End'/// Thoát khỏi chương trình

End Sub

Ghi chú:

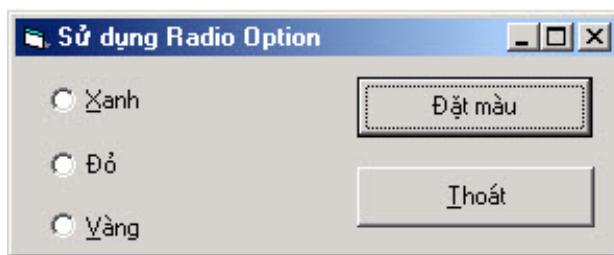
- Thuộc tính name của điều khiển không được chứa dấu cách, ký tự đặc biệt và không được trùng với từ khoá.
- Thuộc tính name nên thêm vào tiền tố gồm 3 ký tự, là viết tắt của loại điều khiển đó, ví dụ : cmd, lbl, txt, lst, cbo, pic, img ...
- Thuộc tính Enable qui định rằng người dùng có được phép tương tác với điều khiển hay không, giá trị true là có được phép
- Thuộc tính Visible qui định tính chất ẩn hay hiện, True là hiện.
- Thuộc tính Text cho biết nội dung văn bản hiện thời chưa trong textbox. Có thể lấy về hoặc thay đổi nội dung textbox thông qua thuộc tính này.
- Câu lệnh **End** dùng để thoát khỏi chương trình.
- Có thể vào menu **Format** để căn lề và đặt độ rộng, độ giãn cách bằng nhau cho các điều khiển.

Sử dụng điều khiển Option

Điều khiển Option button chỉ cho phép người dùng thực hiện lựa chọn 1 trong số nhiều lựa chọn được đưa ra.

Thực hành: Viết chương trình thực hiện việc thiết lập màu của **Form** là Xanh, Đỏ và Vàng tương ứng khi người dùng chọn nút radio Xanh, Đỏ, Vàng.

Thiết kế giao diện:



Giao diện chương trình

Đặt tên cho các điều khiển

Điều khiển	Tên (Name)	Caption
Option Button	optXanh	&Xanh
Option Button	optDo	ĐỎ
Option Button	optVang	&Vàng
Command button	cmdDatMau	Đặt màu
Command button	cmdThoat	&Thoát

Option Explicit

'/// Hiển thị thông báo khi người dùng click chọn nút Đỏ

```
Private Sub optDo_Click()
```

```
Me.Caption = "Nút ĐỎ đã được chọn"
```

```
End Sub
```

'/// Hiển thị thông báo khi người dùng click chọn nút Xanh

```
Private Sub optXanh_Click()
```

```
    Me.Caption = "Nút Xanh đã được chọn"
```

```
End Sub
```

```
'/// Hiển thị thông báo khi người dùng click chọn nút Vàng
```

```
Private Sub optVang_Click()
```

```
    Me.Caption = "Nút Vàng đã được chọn"
```

```
End Sub
```

```
'/// Đặt lại màu nền của Form ứng với lựa chọn của người dùng
```

```
Private Sub cmdDatMau_Click()
```

```
If optXanh.Value = True Then Me.BackColor = &HFF0000 '/// nền Màu Xanh
```

```
If optDo.Value = True Then Me.BackColor = &HFF& '// Nền Màu Đỏ
```

```
If optVang.Value = True Then Me.BackColor = &HC0FFFF '// Nền Màu Vàng
```

```
End Sub
```

```
Private Sub cmdThoat_Click()
```

```
End '/// Thoát
```

```
End Sub
```

Ghi chú:

- Sự kiện Click của Option button xảy ra bất cứ khi nào người dùng click chuột, lúc đó thuộc tính value của nút được click tự động có giá trị là true, còn của các option khác thì có giá trị là false.
- Thuộc tính value của option cho ta biết là nút đó có được click hay không ?. True có nghĩa là được click và false có nghĩa là không.
- Tại một thời điểm, nếu có nhiều nút option trên **Form** thì chỉ có nhiều nhất là 1 option được click (chọn). Trong một số trường hợp nếu có nhiều nhóm chọn thì các option tương ứng phải được đặt thành các nhóm (bằng cách đặt chúng trong các frame) như H3.
- Me.Caption hay viết **Form1.Caption** để đặt tiêu đề cho **Form**.



Nhiều nhóm lựa chọn

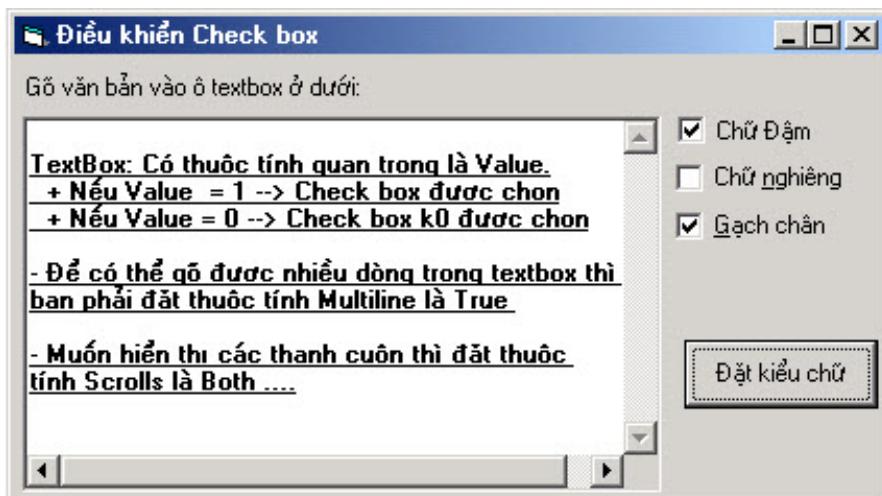
Sử dụng điều khiển CheckBox

Điều khiển CheckBox (Hộp kiểm tra) được sử dụng để yêu cầu người sử dụng lựa chọn một hoặc nhiều lựa chọn (Cũng có thể là 0 có lựa chọn nào). Thuộc tính **Value** sẽ cho biết là check box có được chọn hay là không.

Thực hành: Đặt kiểu chữ cho hộp văn bản (Đậm, nghiêng, gạch chân) khi người dùng click vào các lựa chọn tương ứng, sau đó click nút "Đặt kiểu chữ".

Hướng dẫn: Vì kiểu chữ của văn bản có thể là **Đậm**, **Nghiêng** và **gach chân** đồng thời, do vậy ta không thể đưa ra cho người dùng lựa chọn theo kiểu Radio option, mà ở đây phải đưa điều khiển Checkbox vào để người dùng có thể thực hiện nhiều lựa chọn.

Thiết kế giao diện:



Giao diện và kết quả của chương trình

Đặt các thuộc tính

Điều khiển	Thuộc tính : Giá trị
Label	+ Name: lblThongBao+ Caption: Gõ văn bản vào ô textbox ở dưới
TextBox	+ Name: txtThongBao+ MultiLine: True+ Scrolls : Both+ Text: để trống
CheckBox	+ Name: chkBold+ Value: 0 – Unchecked
CheckBox	+ Name: chkItalic+ Value: 0 – Unchecked
CheckBox	+ Name: chkUnderline+ Value: 0 – Unchecked

Command button	+ Name: cmdDatKieuChu+ Caption: Đặt kiểu chữ
----------------	--

Viết lệnh:

Option Explicit

'/// Thủ tục này xuất hiện khi người dùng click chọn hộp kiểm tra "Chữ đậm"

Private Sub chkBold_Click()

Me.Caption = "Hộp kiểm tra ""Đậm"" đã được chọn"

End Sub

Private Sub chkItalic_Click()

MsgBox "Hộp kiểm tra ""Nghiêng"" đã được chọn"

End Sub

Private Sub chkUnderline_Click()

MsgBox "Hộp kiểm tra ""Gạch chân"" đã được chọn"

End Sub

Private Sub cmdDatKieuChu_Click()

'/// Kiểm tra nếu người dùng chọn hộp kiểm tra nào thì đặt kiểu font tương ứng

If chkBold.Value = 1 **Then** '/// Người dùng đã chọn hộp "Chữ đậm"

txtThongBao.Font.Bold = True

Else '/// Người dùng đã không chọn hộp "Chữ đậm"

txtThongBao.Font.Bold = False '/// Đặt lại thành chữ thường (Không đậm)

End If

If chkItalic.Value = 1 **Then**

txtThongBao.Font.Italic = True

Else

txtThongBao.Font.Italic = False

End If

If chkUnderline.Value = 1 **Then**

txtThongBao.Font.Underline = True

Else

txtThongBao.Font.Underline = False

End If

End Sub

Ghi chú:

- Ngoài 2 giá trị thường dùng của thuộc tính Value là 1-Checked và 0-UnChecked, còn một giá trị thứ 3 dùng trong một số trường hợp là 2-Grayed để ám chỉ rằng bên trong nó có nhiều lựa chọn con khác và một số đã được chọn và một số thì không.
- Có thể viết lệnh ngay ở bên trong sự kiện click chuột của CheckBox.
- Thuộc tính Visible và Enable của CheckBox cũng được dùng để ẩn/hiện và cấm/cho phép tương tác với checkbox.
- Đối tượng con **Font** của điều khiển TextBox dùng để thay đổi kiểu chữ và kiểu font của hộp văn bản.

Sử dụng điều khiển ListBox

Điều khiển ListBox được dùng để lưu một danh sách các mục (Danh sách các số, các xâu ký tự). ListBox có sẵn các thuộc tính và phương thức để thao tác với danh sách này như thêm, bớt, xoá các mục v.v...

Thực tế khi lập trình, điều khiển ListBox và ComboBox rất hay được dùng, do vậy bài thực hành này sẽ hướng dẫn sử dụng các thuộc tính, phương thức và sự kiện của các điều khiển này tương đối chi tiết.

Thực hành sử dụng phương thức Clear và AddItem:

Yêu cầu: Thêm các mục vào danh sách (ListBox) khi người dùng click vào nút "Thêm" và xoá tất cả các mục khi người dùng click vào nút "Xoá".

Thiết kế giao diện:



Giao diện và kết quả khi chạy chương trình

Đặt giá trị cho các thuộc tính:

Điều khiển	Thuộc tính và giá trị tương ứng
Label	Name: lblThongBaoCaption: Gõ một dòng vào đây và click nút thêm
TextBox	Name: txtNewItemText: Để trống
Command button	Name: cmdXoaCaption: &Xoá
Command button	Name: cmdThemCaption: &Thêm

Command button	Name: cmdThoatCaption: Thoát
ListBox	Name: lstDanhSach

Viết lệnh:

Option Explicit

'/// Thực hiện thêm nội dung trong textbox vào Listbox khi người dùng

'/// Click chuột vào nút "Thêm"

Private Sub cmdThem_Click()

lstDanhSach.AddItem txtNewItem.Text, 0 '/// 0 --> Để thêm vào đầu danh sách

End Sub

'/// Gọi Phương thức Clear để xoá toàn bộ các mục trong ListBox

Private Sub cmdXoa_Click()

lstDanhSach.Clear

End Sub

Private Sub cmdThoat_Click()

End '/// Thoát khỏi chương trình

End Sub

Ghi chú:

- Có thể thực hiện sắp xếp các mục trong Listbox bằng cách đặt thuộc tính Sorted = True trong lúc thiết kế.
- Có thể thêm ngay một số mục vào trong ListBox tại cửa sổ Properties bằng cách chọn thuộc tính List. Tại đây, sau khi nhập xong một mục, nhấn tổ hợp phím Ctrl-Enter để thêm mục mới.
- Mặc định khi thêm một mục mới vào trong ListBox bằng phương thức AddItem, mục đó sẽ được thêm vào cuối danh sách, nhưng cũng có thể thêm vào một vị trí bất kỳ bằng cách đưa thêm vị trí (chỉ số) cho phương thức này.

Nếu chỉ số đưa vào là 0 (như VD trên) thì mục mới sẽ được thêm vào đầu danh sách.

Thực hành đọc một số thuộc tính quan trọng của ListBox

Trong phần thực hành này, chúng ta sẽ đọc một số thuộc tính rất hay dùng của ListBox và hiển thị ra màn hình.

Thiết kế giao diện: Như phần trước nhưng thêm nút lệnh (đặt Name là cmdThongTin, Caption là "&Các thông tin thêm về ListBox" như hình 6) :



Giao diện chương trình khi chạy

Viết lệnh:

Option Explicit

'/// Thực hiện thêm nội dung trong textbox vào Listbox khi người dùng

'/// Click chuột vào nút "Thêm"

Private Sub cmdThem_Click()

IstDanhSach.AddItem txtNewItem.Text, 0 '/// 0 --> Thêm vào đầu danh sách

End Sub

// Hiển thị giá trị một số thuộc tính quan trọng của ListBox

Private Sub cmdThongTin_Click()

Debug.Print "Tổng số mục trong listbox là " & IstDanhSach.ListCount

```
Debug.Print "Chỉ số của mục đang chọn là (lstDanhsach.ListIndex) : " & lstDanhSach.ListIndex
```

```
Debug.Print "Nội dung của mục đang được chọn: lstDanhsach.Text = " & lstDanhSach.Text
```

```
Debug.Print "Nội dung (ND) của mục thứ 3 là : lstDanhsach.List(2) = " & lstDanhSach.List(2)
```

```
Debug.Print "Mục đầu tiên là : lstDanhsach.List(0) = " & lstDanhSach.List(0)
```

```
Debug.Print "Mục cuối là : lstDanhsach.List(lstDanhsach.ListCount-1) = " & _  
lstDanhSach.List(lstDanhsach.ListCount - 1)
```

```
End Sub
```

```
'// Gọi Phương thức Clear để xoá toàn bộ các mục trong ListBox
```

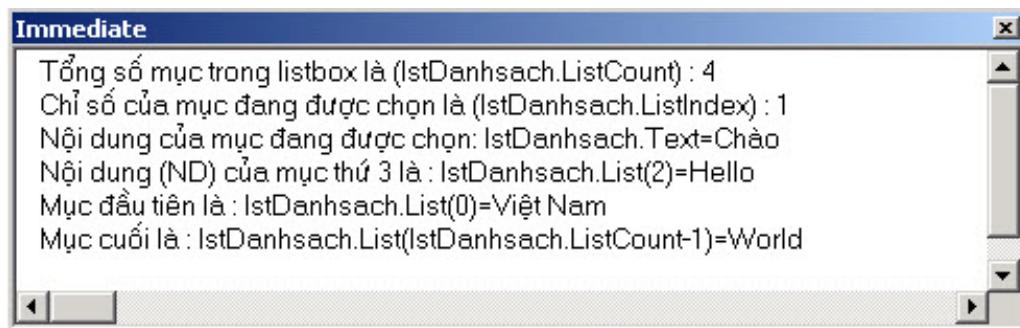
```
Private Sub cmdXoa_Click()
```

```
lstDanhSach.Clear
```

```
End Sub
```

```
Private Sub cmdThoat_Click()
```

Kết quả khi chạy chương trình ứng với các lựa chọn ở hình 6:



Kết quả khi chạy chương trình

Ghi chú:

- Thuộc tính ListCount cho ta biết số mục hiện có trong ListBox, nếu không có mục nào thì thuộc tính này có giá trị là 0.

- Thuộc tính ListIndex cho ta biết chỉ số (lưu ý chỉ số được tính từ 0) của mục hiện đang được chọn (Mục có thanh sáng). Nếu không có mục nào được chọn thì nó có giá trị là -1.
- Thuộc tính Text cho ta nội dung của mục hiện đang được chọn. nếu không có mục nào được chọn (không có thanh sáng) thì thuộc tính này sẽ có giá trị là một xâu rỗng.
- Thuộc tính List(i) cho ta nội dung của mục có chỉ số là i. Như vậy muốn lấy giá trị của mục đang được chọn thì ngoài việc sử dụng thuộc tính Text, ta còn có thể viết như sau: lstDanhSach.List(DanhSach.ListIndex)
- Mục đầu tiên của Listbox có chỉ số là 0 và Mục cuối cùng có chỉ số là lstDanhSach.ListCount – 1.
- Trong trường hợp listbox có thể không chứa mục nào (Listbox rỗng) thì để tránh bị lỗi, trước khi thao tác, chúng ta nên kiểm tra thuộc tính ListCount xem có > 0 hay không (tức không rỗng hay không?).

Thực hành sử dụng đặc tính đa lựa chọn của Listbox

Listbox có thể đặt ở chế độ cho phép người dùng chọn đồng thời nhiều mục bằng cách click và di chuột hoặc click chuột kết hợp với nhấn phím Shift/ Ctrl để chọn các mục liền nhau/ rời rạc nhau, như 2 hình dưới đây:



Muốn hiển thị Listbox theo kiểu này (Kiểu CheckBox), thì cần đặt thuộc tính Style là CheckBox

Hiển thị kiểu CheckBox



Muốn hiển thị Listbox theo kiểu này (Kiểu Standard), thì cần đặt thuộc tính Style là Standard và thuộc tính MultiSelect là 2-Extended.

Cho phép chọn nhiều mục kiểu Ext

Thiết kế giao diện: Lấy ví dụ như hình trên

Đặt giá trị cho các điều khiển, ngoài ra cần thêm một nút lệnh có tên và caption tương ứng là: cmdHienThi, "Hiển thị các mục đang được chọn"

Viết lệnh:

Option Explicit

'/// Hiển thị các mục hiện đang được chọn.

'// Nếu một mục có chỉ số i đang được chọn thì lstDanhSach.Selected(i) sẽ có giá trị True

'// Giá trị của mục có chỉ số i sẽ là : lstDanhSach.List(i)

Private Sub cmdHienThi_Click()

Dim i **As** Integer

For i = 0 To lstDanhSach.ListCount – 1 '/// Kiểm tra từng mục trong ListBox

If lstDanhSach.Selected(i) = True **Then** '/// Nếu mục i này được chọn (=True)

MsgBox lstDanhSach.List(i) '/// Thị hiển thị ra màn hình

End If

Next

End Sub

'/// Thực hiện thêm nội dung trong textbox vào Listbox khi người dùng

'/// Click chuột vào nút "Thêm"

Private Sub cmdThem_Click()

lstDanhSach.AddItem txtNewItem.Text, 0 '/// 0 --> Thêm vào đầu danh sách

End Sub

'/// Gọi Phương thức Clear để xoá toàn bộ các mục trong ListBox

Private Sub cmdXoa_Click()

```
IstDanhSach.Clear
```

```
End Sub
```

Ghi chú:

- Khi duyệt tất cả các phần tử trong ListBox, ta thường sử dụng vòng lặp dạng **For i = 0 to listbox1.ListCount – 1**
- Khi muốn lấy các mục đang được chọn (ở chế độ MultiSelect) thì cần kiểm tra thuộc tính Selected(i) = True ? để biết mục i có được chọn hay không.

Thực hành tìm kiếm và loại bỏ một mục khỏi danh sách

Yêu cầu (Giao diện như hình 10): Khi người dùng nhập giá trị vào trong hộp textbox và nhấn vào nút "Xoá" thì chương trình sẽ thực hiện tìm kiếm và xoá mục này khỏi Listbox, hoặc thông báo là không tìm thấy nếu mục này không có trong ListBox.

Thiết kế giao diện:



Giao diện chương trình khi chạy

Thiết lập giá trị cho các điều khiển:

Điều khiển	Thuộc tính / giá trị
Label	Name: lblThongBaoCaption: Gõ vào nội dung từ cần xoá
TextBox	Name: txtTuCanXoaText: Để trống
Command button	Name: cmdXoaCaption: &Xoá mục này khỏi ListBox
ListBox	Name: lstDanhSach

Viết lệnh:

Option Explicit

```
'// Thêm một số mục ban đầu vào ListBox
```

```

Private SubForm_Load()

    lstDanhSach.AddItem "Hello"

    lstDanhSach.AddItem "World"

    lstDanhSach.AddItem "welcome"

    End Sub

'//Thực hiện tìm kiếm và xoá mục trong danh sách mà người dùng vừa nhập trong
textbox

Private Sub cmdXoa_Click()

Dim i As Integer

Dim TimThay As Boolean

TimThay = False

For i = 0 To lstDanhSach.ListCount - 1 '/// Duyệt và so với tất cả các phần tử

If lstDanhSach.List(i) = txtTuCanXoa.Text Then

    TimThay = True

    Exit For ' /// Thoát khỏi vòng lặp for

End If

Next

If TimThay = True Then'/// Đã tìm thấy → Xoá

    MsgBox "Vị trí tìm thấy là " & i & " Mục này đã được loại bỏ khỏi ListBox"

    lstDanhSach.RemoveItem i

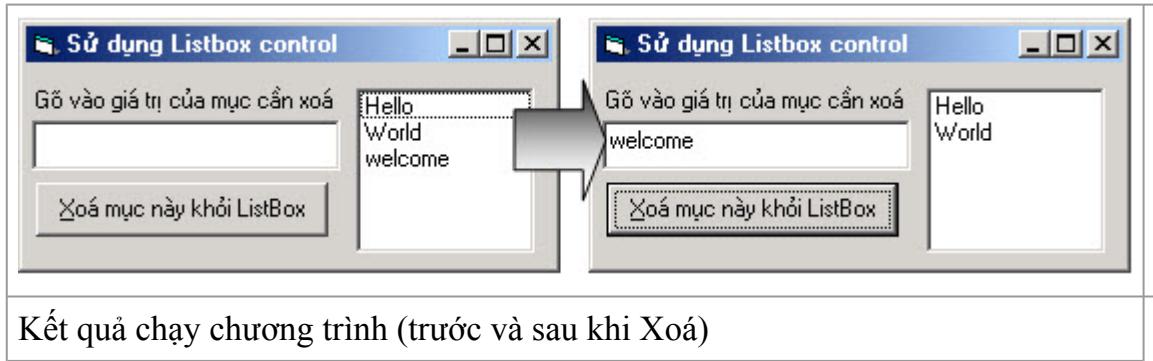
Else

    MsgBox "Không có mục này trong ListBox", vbInformation, "Thông báo"

End If

```

End Sub



Ghi chú:

- Khi muốn loại bỏ một mục khỏi Listbox, ta sử dụng phương thức RemoveItem. Phương thức này đòi hỏi một tham số là chỉ số của mục cần xoá.
- Khi xoá, cần lưu ý là chỉ số i của mục cần xoá phải thỏa mãn : $0 \leq i \leq \text{ListCount} - 1$.
- Khi so sánh các xâu ký tự, có thể sử dụng thêm các hàm Trim, Ucase, Lcase. Ví dụ: **If Ucase(txtTuCanXoa) = Ucase(...) Then**

Sử dụng điều khiển PictureBox

Điều khiển PictureBox cho phép hiển thị (Load) các file ảnh dạng BMP, ICO, WMF, JPEG, GIF cũng như có các phương thức cho phép ta thực hiện công việc xử lý đồ họa trên đó như vẽ đường thẳng, vẽ đường tròn v.v...

Thực hành: Hiển thị file ảnh và vẽ đường thẳng trên PictureBox.

Yêu cầu (Giao diện như hình vẽ): Khi người dùng nhập đường dẫn của file ảnh và click vào nút "Hiển thị" thì hiển thị file ảnh đó trên PictureBox. Còn khi người dùng click vào nút "Vẽ đường thẳng" thì vẽ một đường thẳng.

Giao diện chương trình:



Giao diện và kết quả khi chạy chương trình

Thiết lập giá trị cho các điều khiển:

Điều khiển	Thuộc tính / Giá trị
PictureBox	Name: picViewer AutoRedraw: True
TextBox	Name: txtTenFile
Command button	Name: cmdHienThi Caption: &Hiển thị
Command button	Name: cmdVeDuongThang Caption: &Vẽ đường thẳng

Viết lệnh:

```
Form1.frm
'/' Nạp file ảnh và hiển thị trong Picturebox
Private Sub cmdHienThi_Click() On Error GoTo errpicViewer.Picture = LoadPicture(txtTenFile.Text)
Suberr:MsgBox "File này không tồn tại hoặc có lỗi !", vbCritical, "Lỗi"
End Sub'/'
```

Vẽ một đường chéo: cú pháp Line (Cột 1, Hàng 1) – (Cột 2, Hàng 2)

```
Private Sub cmdVeDuongThang_Click()
    picViewer.Line (0, 0)-(picViewer.Width, picViewer.Height)
End Sub
```

Ghi chú:

- Thuộc tính AutoRedraw của Picture đặt là true để đảm bảo những gì vẽ trên đó không bị mất đi khi **Form** bị vẽ lại.
- Phương thức LoadPicture(<Tên file>) trả về cho ta một đối tượng ảnh, đối tượng này có thể gán trả lại cho thuộc tính Picture của điều khiển PictureBox.
- Cần phải có cơ chế bắt lỗi như ví dụ trên vì khi nạp ảnh rất có thể có trường hợp xuất hiện lỗi.
- Có thể đặt thuộc tính AutoResize của Picturebox là True nếu bạn muốn kích thước của Picturebox luôn tự động co giãn bằng với kích thước của ảnh.
- Có thể vẽ (copy) ảnh trong Picturebox vào **Form** bằng lệnh: *Me.PaintPicture picViewer.Picture,0,0* (Hiểu là: Vẽ ảnh trong picturebox vào **Form**, bắt đầu từ toạ độ 0,0).

Sử dụng điều khiển Image

Điều khiển Image tương tự như điều khiển Picture (chính xác là thực hiện được một số chức năng của Picturebox) nhưng có ưu điểm là tốc độ nhanh hơn, chiếm ít tài nguyên hơn Picture. Ngoài ra nó còn có thuộc tính Stretch để co giãn kích thước của ảnh, trong khi Picturebox không có.

Thực hành: Hiển thị file ảnh trong điều khiển Image

Thiết kế giao diện:



Giao diện chương trình

Thiết lập giá trị cho các thuộc tính:

Điều khiển	Thuộc tính / Giá trị
Image	Name: imgViewerStretch: True
TextBox	Name: txtTenFile
Command button	Name: cmdHienThiCaption: &Hiển thị

Viết lệnh:

```
Form1.frm
Option Explicit'/' Nạp file ảnh và hiển thị trong điều khiển ImagePrivate Sub
cmdHienThi_Click()On Error GoTo errImgViewer.Picture =
LoadPicture(txtTenFile.Text)Exit Suberr:MsgBox "File này không tồn tại hoặc có
lỗi !", vbCritical, "Lỗi"End Sub
```



Stretch = True



Stretch = False

Kết quả khi thay đổi thuộc tính Stretch

Sử dụng HscrollBar (Thanh cuộn ngang)

HscrollBar là một điều khiển thường được sử dụng để cho người dùng thay đổi một giá trị nào đó một cách trực quan thay vì phải nhập các con số. VscrollBar làm việc hoàn toàn giống với HscrollBar, tuy rằng hiển thị theo chiều dọc, Do vậy ta sẽ không giới thiệu ở đây.

Thực hành: Thay đổi màu nền và kích thước font chữ trong textbox sử dụng thanh cuộn ngang (HscrollBar) – Giao diện như hình dưới.



Giao diện chương trình và kết quả khi chạy

Thiết lập giá trị cho các thuộc tính:

Điều khiển	Thuộc tính / Giá trị
TextBox	Name: txtThongBaoText: Thay đổi các thanh cuộn và quan sát !
Label	Caption: Thay đổi kích thước font chữ của textbox
Label	Caption: Thay đổi màu nền của TextBox
HScrollBar	Name: hscKichThuocMin: 10Max : 50SmallChange: 1LargeChange: 10
HscrollBar	Name: hscMauNenMin: 0Max : 255SmallChange: 1LargeChange: 20

Viết lệnh:

Form1.frm

```
Option Explicit'// Sự kiện này kích hoạt khi giá trị trên hscKichThuoc thay đổi'//
nên ta sẽ thực hiện cập nhật lại kích thước font tại đâyPrivate Sub
hscKichThuoc_Change()txtThongBao.Font.Size= hscKichThuoc.Value '/ Đặt
fontsize bằng giá trị của hscRollbarEnd Sub'// Sự kiện này kích hoạt khi giá trị trên
```

hscMauNen thay đổi'// nên ta sẽ thực hiện cập nhật lại giá trị của màu nền tại đây**Private Sub** hscMauNen_Change()*txtThongBao.BackColor = hscMauNen.Value'//* Đặt màu bằng với hscMauNen.Value*End Sub*

Ghi chú:

- Có thể đặt "Thanh gạt" của thanh cuộn tại vị trí bất kỳ bằng cách thay đổi thuộc tính Value thích hợp (trong lúc thiết kế hoặc khi chương trình chạy).

Sử dụng điều khiển Timer, Drive, Dir và File

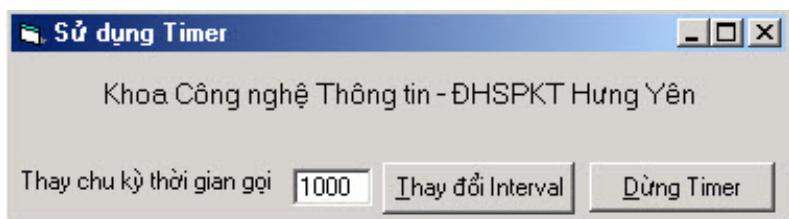
Sử dụng điều khiển Timer

Điều khiển Timer có chức năng tự động kích hoạt thủ tục Timer sau mỗi khoảng thời gian đã được ấn định trước mà không phụ thuộc vào công việc đang được thực hiện là gì.

Thực hành: Viết chương trình thực hiện nhấp nháy một dòng chữ trên **Form**. Trong đó khoảng thời gian nhấp nháy có thể thay đổi bằng việc nhập giá trị vào một textbox.

Hướng dẫn: Nhấp nháy dòng chữ (đặt trong label) thực chất là việc chúng ta cho ẩn rồi lại hiện dòng chữ đó sau mỗi khoảng thời gian nhất định (ví dụ là 1 s). Như vậy, giải pháp là ta sẽ cập nhật thuộc tính Visible sau mỗi khoảng thời gian 1 s, nếu đang là hiện (Visible = true) thì cho ẩn đi (đặt visible là false) và ngược lại, hay nói cách khác là sau 1 giây thì ta đặt thuộc tính Visible = not Visible.

Thiết kế giao diện như sau:



Đặt giá trị cho các thuộc tính:

Điều khiển	Thuộc tính / Giá trị
Label	Name: lblThongBaoCaption: Khoa Công nghệ Thông tin....
Label	Caption: Thay chờ kỳ thời gian gọi
TextBox	Name: txtIntervalText: 1000
Command button	Name: cmdThayDoiIntervalCaption: &Thay đổi Interval
Command button	Name : cmdDungTimerCaption: &Dừng Timer
Timer	tmrNhapNhay

Viết lệnh:

Form1.frm

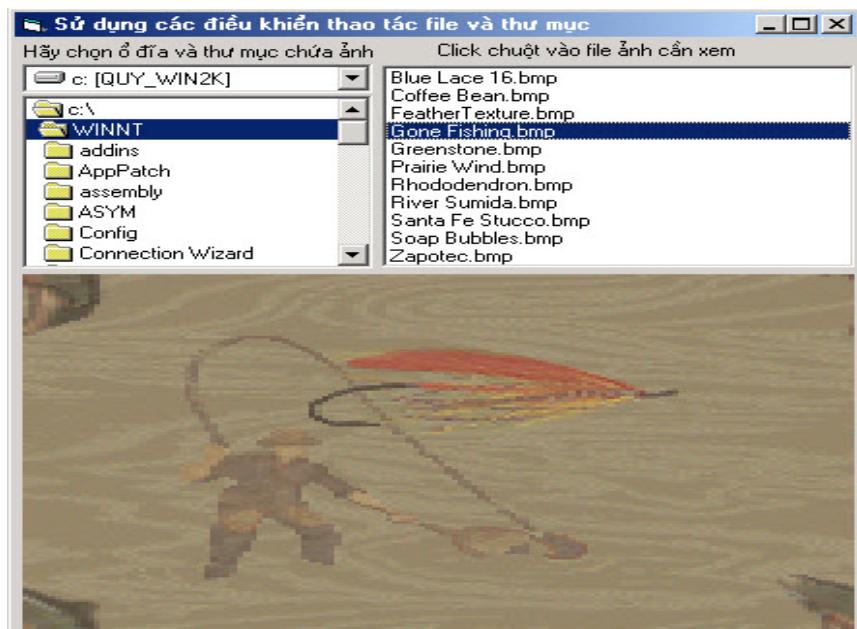
```
Option Explicit'// Khởi tạo một số thuộc tínhPrivate
SubForm_Load()tmrNhapNhay.Interval = 1000tmrNhapNhay.Enabled =
TruelblThongBao.Visible = TrueEnd Sub'// Thủ tục này sẽ được tự động gọi sau
khoảng thời gian là 1 giây (1000 ms)Private Sub
tmrNhapNhay_Timer()lblThongBao.Visible = Not lblThongBao.VisibleEnd Sub'//
Cập nhật lại chu kỳ thời gian tự động kích hoạt sự kiện TimerPrivate Sub
cmdThayDoiInterval_Click()tmrNhapNhay.Interval = txtInterval.TextEnd Sub'//
Dừng đồng hồ. Có thể dừng bằng cách đặt Interval = 0Private Sub
cmdDungTimer_Click()tmrNhapNhay.Enabled = False End Sub
```

Sử dụng điều khiển Drive, Dir và File

Điều khiển Drive được dùng để cho ta chọn một ổ đĩa trên máy tính, điều khiển Dir để hiển thị các thư mục của một ổ đĩa nào đó theo dạng phân cấp. Còn điều khiển File dùng để hiển thị tất cả các file trong một thư mục nhất định. 3 Điều khiển này hoàn toàn có thể sử dụng độc lập nhưng thường được sử dụng kết hợp với nhau.

Thực hành: Viết chương trình xem các file ảnh trong máy tính.

Giao diện chương trình:



Giao diện và kết quả khi chạy chương trình

Thiết lập các thuộc tính:

Điều khiển	Thuộc tính /giá trị
Label	Caption: Hãy chọn ổ đĩa và thư mục chứa ảnh
Label	Click vào file ảnh cần xem
Drive	Name: drvPhoto
Dir	Name: dirPhotoPattern: *.BMP;*.JPG;*.GIF (lưu ý không có dấu cách giữa các dấu ";")
File	Name: filPhoto
Image	Name: imgPhotoStretch : True

Viết lệnh:

Form1.frm

```

Option Explicit'/// Đặt thuộc tính Pattern để điều khiển File chỉ hiển thị những file ảnh'/// Nếu muốn hiển thị tất cả các file thì đặt pattern là *.*Private Sub
Sub Form_Load()filPhoto.Pattern = "*.BMP;*.JPG;*.GIF" '/// Chỉ hiển thị các file ảnh.End Sub'/// Khi người dùng chọn thư mục thì ta hiển thị các file của thư mục này'/// trong điều khiển file bằng cách thay đổi thuộc tính Path của điều khiển file'///
bằng với đường dẫn của thư mục vừa chọn.Private Sub
dirPhoto_Change()filPhoto.Path = dirPhoto.PathEnd Sub'/// Khi người dùng chọn ổ đĩa khác thì ta lấy ổ đó gán cho điều khiển'/// dirPhoto để điều khiển dirPhoto hiển thị nội dung của ổ đĩa vừa chọnPrivate Sub drvPhoto_Change()dirPhoto.Path = drvPhoto.Drive 'hoặc có thể viết dirPhoto.Path = "c:\"
v.v...End Sub'/// Khi người dùng click vào một file trong điều khiển File thì ta sẽ hiển thị file này'/// trong điều khiển Image. Đường dẫn của file này có thể lấy từ thuộc tính'/// filePhoto.Path và filPhoto.FileNamePrivate Sub filPhoto_Click()imgPhoto.Picture =
LoadPicture(filPhoto.Path & "\" & filPhoto.FileName)End Sub

```

Giải thích câu lệnh:

- 3 điều khiển ở trên không nhất thiết là phải đi cùng với nhau, ví dụ chỉ cần điều khiển filPhoto, ta có thể dùng hiển thị toàn bộ các file trong thư mục C:\My documents bằng cách viết : filPhoto.Path = "C:\My documents" hay ta có thể toàn bộ thư mục trong ổ đĩa E trong điều khiển dirPhoto bằng cách viết dirPhoto.Path = "E:\\"
- drvPhoto.Drive sẽ trả về cho ta tên ổ đĩa (ví dụ c:, d: hay e:) hiện đang được chọn trong điều khiển drvPhoto.

- filPhoto.Path trả về đường dẫn (không bao gồm tên) của file đang được chọn trong điều khiển File, filPhoto.FileName mới trả về tên file (nhưng không có đường dẫn) của file đang chọn.

Bài thực hành số 4: Sử dụng các hộp thoại

Sử dụng các hộp thoại

Hiển thị các loại hộp thoại OPEN

Hộp thoại Open (Open dialog) là một hộp thoại chuẩn cho phép người dùng chọn một file hay một thư mục.

Thực hành: Viết chương trình mở một file (Text hoặc rtf) và hiển thị trong điều khiển Richtext Box.

Các bước thực hiện:

Bước 1: Tạo một Project mới

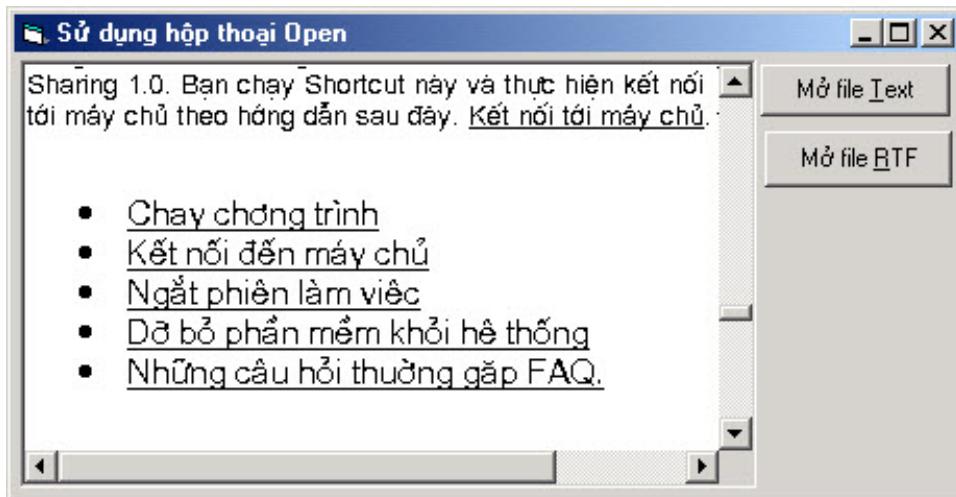
Bước 2: Vào menu Project → Components và tham chiếu đến 2 file OCX là Rich Textbox và Common Dialog như hình 1

Bước 3: Kéo điều khiển Dialog và Richtext vào Form



Các điều khiển dùng trong chương trình

Thiết kế giao diện:



Giao diện và kết quả khi chạy chương trình

Đặt giá trị cho các thuộc tính:

Điều khiển	Thuộc tính / Giá trị
Form	Name : frmMainCaption: Sử dụng hộp thoại OpenStartup Position: 2- CenterScreen
Rich Textbox Control	Name : rtfEditorAutoVerbMenu: TrueScrollBars : 3 - Both
CommonDialog	Name: dlgHopThoai (Hộp thoại)
Command button	Name : cmdOpenTextFileCaption: Mở file &Text
Command Button	Name : cmdOpenRTFFileCaption: Mở file &RTF

Viết lệnh:

```
frmMain.frm

Option Explicit'// Mở các file RichText Format (các file có phần mở rộng là RTF)Private sub cmdOpenRTFFile_Click()dlgHopThoai.DialogTitle = "Mở file Text" '/// Tiêu đề của hộp thoạidlgHopThoai.Filter = "Cac file RichText (*.RTF)|*.rtf" '/// Chỉ hiển thị các file RTFdlgHopThoai.Showopen '/// Hiển thị hộp thoạirtfEditor.LoadFile dlgHopThoai.FileName, rtfRTF '/// Mở file vừa chọn trong RTFEndSub'/// Mở các file Text (các file có phần mở rộng là TXT)Private sub cmdOpenTextFile_Click()dlgHopThoai.DialogTitle = "Mở file Text"dlgHopThoai.Filter = "Cac file text"
```

```
(*.txt)|*.txt|"dlgHopThoai.ShowopenrtfEditor.LoadFile dlgHopThoai.FileName,  
rtfTextEndSub
```

Ghi chú:

- Phương thức **LoadFile** của điều khiển RichText dùng để mở một file. Nó đòi hỏi 2 tham số, tham số thứ nhất là đường dẫn đến file cần mở, tham số thứ hai là loại file cần mở (Tham số này là tùy chọn và có thể bỏ qua).
- Thuộc tính **CenterScreen** dùng để đặt **Form** vào giữa màn hình khi hiển thị.
- Thuộc tính **AutoVerbMenu = True** của điều khiển RichText để cho phép hiển thị menu (Copy, cut, Paste, Undo) khi người dùng click chuột phải lên RichText control.
- Trong trường hợp, khi hộp thoại mở ra, nếu người dùng không chọn file nào (Chọn nút Cancel) thì thuộc tính **FileName** sẽ có giá trị ở lần mở trước đó hoặc sẽ có giá trị rỗng, và như vậy có thể gây lỗi nạp file. Để giải quyết trường hợp này, có thể viết lại đoạn chương trình trên như sau:

Private sub cmdOpenRTFFile_Click()

```
dlgHopThoai.DialogTitle = "Mo file Text" dlgHopThoai.Filter = "Cac file RichText  
(*.RTF)|*.rtf|" dlgHopThoai.FileName = ""'// Khởi tạo trước
```

dlgHopThoai.Showopen

```
If dlgHopThoai.FileName <> "" Then '// Đã chọn file
```

```
rtfEditor.LoadFile dlgHopThoai.FileName, rtfRTF
```

```
End If
```

End Sub

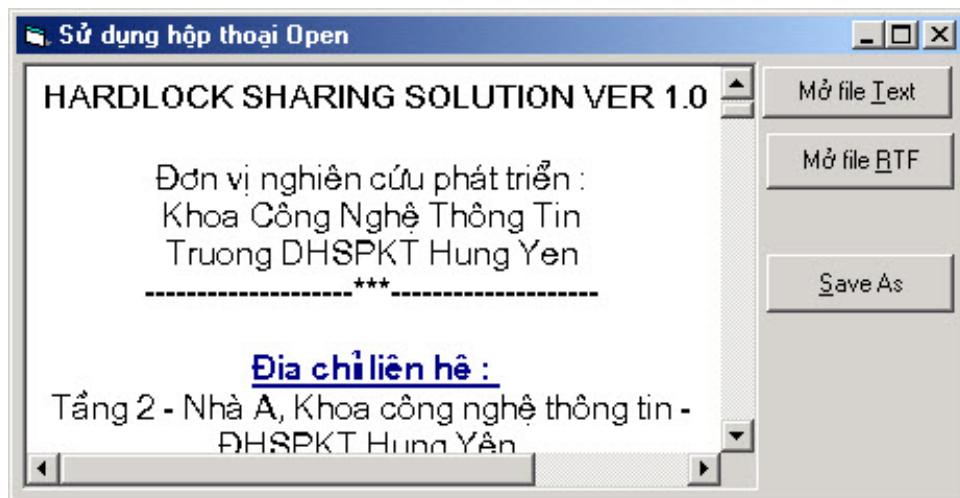
Hiển thị các loại hộp thoại SAVE - SAVE AS

Hộp thoại Save (hay Save As) về thực chất cũng chính là hộp thoại Open (Cùng sử dụng điều khiển Dialog) nhưng có điểm khác là tiêu đề của hộp thoại này mặc định bây giờ là "Save As" thay vì "Open" để người dùng có "cảm giác" thân thiện hơn. Tiêu đề của hộp thoại luôn luôn có thể thay đổi được bằng cách thay đổi thuộc tính **DialogTitle**.

Thực hành: Xây dựng chương trình giống như phần 1 nhưng có thêm nút "Save As" để khi người dùng click vào nút này thì nội dung đang soạn thảo sẽ được lưu ra đĩa với một tên file mới.

Các bước thực hiện: Giống như phần 1

Thiết kế giao diện: Thêm nút Save (Name: cmdSaveAs, Caption: &Save As)



Giao diện và kết quả chạy chương trình

Viết lệnh:

```
frmMain.frm
Option Explicit'/'' Mở các file Richtext Format (các file có phần mở rộng là RTF)Private sub cmdOpenRTFFile_Click()dlgHopThoai.DialogTitle = "Mở file Text" '/// Đặt lại tiêu đềdlgHopThoai.Filter = "Các file Richtext (*.RTF)|*.rtf|" '///
Chỉ hiển thị file dạng RTFdlgHopThoai.Showopen'/'' Mở hộp thoại chọn filertfEditor.LoadFile dlgHopThoai.FileName, rtfRTF '/// Nạp file vào RichTextboxEndSub'/'' Mở các file Text (các file có phần mở rộng là TXT)Private sub cmdOpenTextFile_Click()dlgHopThoai.DialogTitle = "Mở file Text"dlgHopThoai.Filter = "Các file text (*.txt)|*.txt|"dlgHopThoai.ShowopenrtfEditor.LoadFile dlgHopThoai.FileName, rtfTextEndSub'/'' Thực hiện mở hộp thoại để cho người dùng gõ một tên file mới'/'' và lưu nội dung đang soạn thảo ra file nàyPrivate sub cmdSaveAs_Click()dlgHopThoai.Filter = "Tất cả các file (*.*)|*.*|"dlgHopThoai.FileName = ""dlgHopThoai.Showsave'/'' Mở hộp thoại "Save As"IfdlgHopThoai.FileName <> "" ThenrtfEditor.SaveFile dlgHopThoai.FileName, rtfRTF'/'' rtfRTF Hoặc rtfText nếu muốn lưu sang dạng file Text (TXT)EndIfEndSub
```

- Chú ý:
- Giống như hộp thoại Open, hộp thoại "Save As" cũng "Không tự động lưu nội dung soạn thảo ra đĩa" mà ta phải tự làm (lệnh rtfEditor.SaveFile

dlgHopThoai.FileName, rtfRTF) Nó chỉ cho ta biết là người dùng đã gõ vào tên file gì mà thôi (thông qua thuộc tính FileName)

- Nếu muốn lưu nội dung vào chính file đó (chứ không phải ra file mới) thì không cần phải hiển thị hộp thoại Save As mà chỉ cần viết câu lệnh rtfEditor.SaveFile **dlgHopThoai.FileName**.

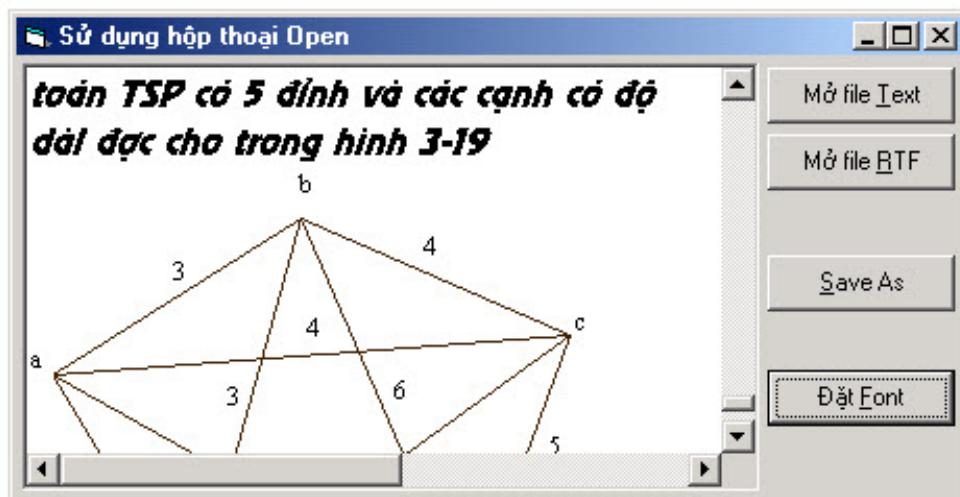
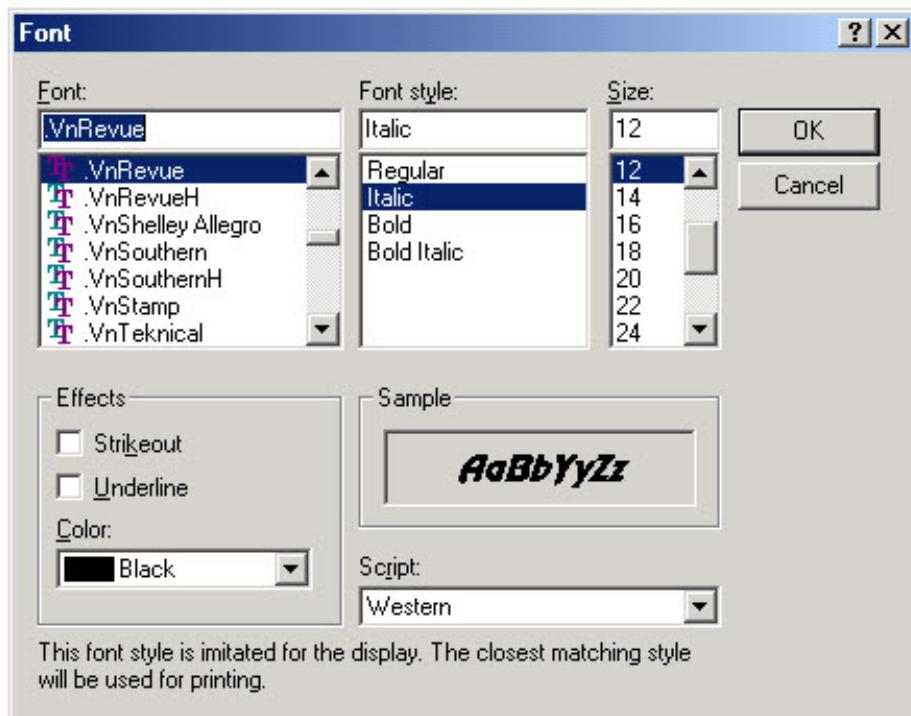
Hiển thị hộp thoại FONT

Hộp thoại chọn font cũng sử dụng điều khiển Dialog, nó hiển thị và cho phép người dùng chọn các thông số liên quan đến font chữ như tên font, kích thước, kiểu chữ v.v... Các giá trị này khi chọn sẽ được lưu trong các thuộc tính tương ứng của điều khiển Dialog.

Thực hành: Xây dựng chương trình giống như phần 2, nhưng có thêm nút "Đặt font" để khi người dùng click vào nút này thì chương trình sẽ hiển thị hộp thoại chọn font, sau đó đặt font chữ đã chọn cho nội dung chứa trong RichTextBox **Format**.

Các bước thực hiện: Giống như phần 2

Thiết kế giao diện : Giống phần 2 và thêm nút "Set Font" (Name: cmdSetFont, Caption : Set &Font)



Kết quả khi chạy chương trình

Viết lệnh:

frmMain.frm

```
Option Explicit/// Mở các file Richtext Format (các file có phần mở rộng là RTF)Private sub cmdOpenRTFFile_Click()dlgHopThoai.DialogTitle = "Mở file Text"dlgHopThoai.Filter = "Cac file Richtext (*.RTF)|*.rtf|dlgHopThoai.ShowopenrtfEditor.LoadFiledlgHopThoai.FileName, rtfRTFEndSub'// Mở các file Text (các file có phần mở rộng là TXT)Private sub cmdOpenTextFile_Click()dlgHopThoai.DialogTitle =
```

```

"Mo file Text"dlgHopThoai.Filter = "Cac file text
(*.txt)|*.txt|"dlgHopThoai.ShowopenrtfEditor.LoadFile dlgHopThoai.FileName,
rtfTextEndSub'/// Thực hiện mở hộp thoại để cho người dùng gõ một tên file
mới'/// và lưu nội dung đang soạn thảo ra file nàyPrivate sub
cmdSaveAs_Click()dlgHopThoai.Filter = "Tất cả các file
(*.*|*.*|"dlgHopThoai.FileName =
"""dlgHopThoai.ShowsaveIfdlgHopThoai.FileName <> "" ThenrtfEditor.SaveFile
dlgHopThoai.FileName, rtfRTF'/// rtfRTF Hoặc rtfText nếu muốn lưu sang dạng
file TextEndIfEndSub'/// Đặt font chữ cho toàn bộ văn bản trong Rich
TextBoxPrivate sub cmdSetFont_Click()dlgHopThoai.Flags = cdICFBBoth Or
cdICFEffects'/// Phải có dòng nàydlgHopThoai.Showfont'/// Hiển thị hộp thoại
chọn font rtfEditor.Font.Name = dlgHopThoai.FontName'/// Đặt font chữ trong
Rich textboxrtfEditor.Font.Size = dlgHopThoai.FontSize'/// bằng với giá trị vừa
chọnrtfEditor.Font.Bold = dlgHopThoai.FontBoldrtfEditor.Font.Italic =
dlgHopThoai.FontItalicrtfEditor.Font.Underline =
dlgHopThoai.FontUnderlinertfEditor.Font.Strikethrough =
dlgHopThoai.FontStrikethruEndSub

```

Ghi chú:

- Hộp thoại chọn font *không tự đặt font chữ* cho bất kỳ văn bản nào, việc đó hoàn toàn do lập trình viên quyết định. Nó chỉ cho ta biết được là người dùng đã chọn font có tên là gì, kích thước, kiểu cách v.v... thông qua các thuộc tính tương ứng.
 - Có thể *đặt font chữ* chỉ cho riêng phần văn bản được bôi đen trong RichText thông qua các thuộc tính có tiền tố là Sel..., ví dụ:
-

rtfEditor.SelBold = **dlgHopThoai**.FontBold

rtfEditor.SelItalic = **dlgHopThoai**.FontItalic

rtfEditor.SelFontSize = **dlgHopThoai**.FontSize

rtfEditor.SelFontName = **dlgHopThoai**.FontName

.....

Hiển thị hộp thoại COLOR

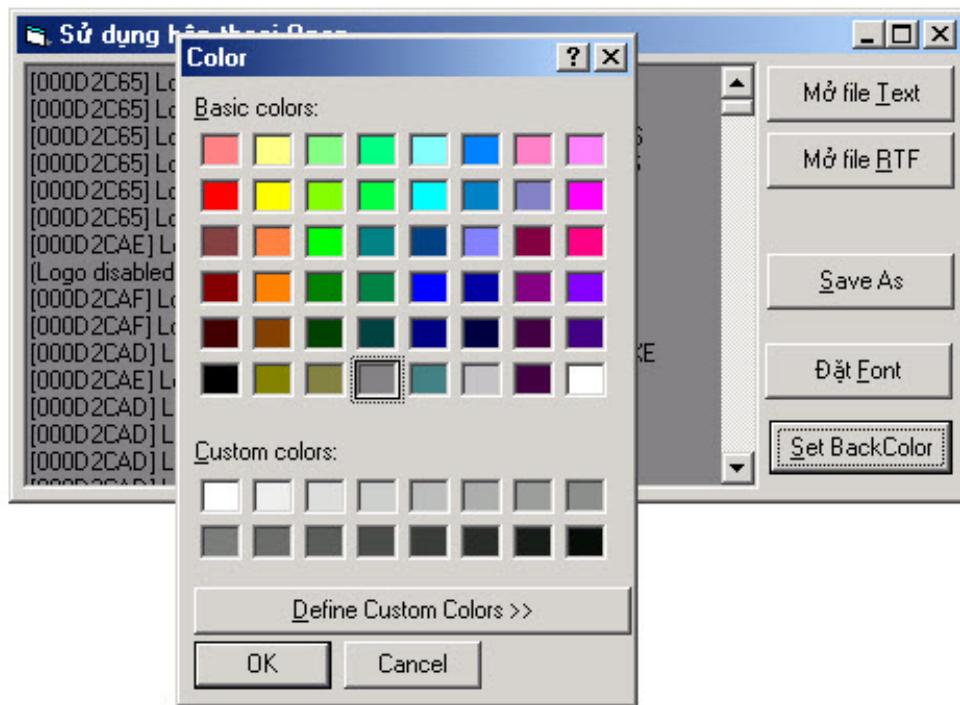
Hộp thoại hiển thị bảng màu (Color) cũng nằm trong điều khiển Dialog. Sau khi hiển thị, hộp thoại này sẽ cho ta biết là người dùng đã chọn màu có giá trị là bao nhiêu. Giá

trị này sẽ tương ứng với một màu xác định và thường được sử dụng cho mục đích đặt màu chữ, màu nền v.v... cho các thành phần khác.

Thực hành: Giống phần 3 nhưng thêm khả năng đặt màu nền cho RichTextbox khi người dùng click vào nút "Set BackColor".

Các bước thực hiện: Như phần 3

Thiết kế giao diện: Như phần 3 và thêm nút lệnh: Name: cmdSetBackColor, Caption: Set &BackColor"



Hộp thoại chọn Font

Viết lệnh:

```
frmMain.frm
Option Explicit/// Mở các file Richtext Format (các file có phần mở rộng là RTF)Private sub cmdOpenRTFFile_Click()dlgHopThoai.DialogTitle = "Mo file Text"dlgHopThoai.Filter = "Cac file Richtext (*.RTF)|*.rtf|"dlgHopThoai.ShowopenrtfEditor.LoadFile dlgHopThoai.FileName, rtfRTFEndSub'/// Mở các file Text (các file có phần mở rộng là TXT)Private sub cmdOpenTextFile_Click()dlgHopThoai.DialogTitle = "Mo file Text"dlgHopThoai.Filter = "Cac file text (*.txt)|*.txt|"dlgHopThoai.ShowopenrtfEditor.LoadFile dlgHopThoai.FileName,
```

```

rtfTextEndSub'// Thực hiện mở hộp thoại để cho người dùng gõ một tên file
mới'// và lưu nội dung đang soạn thảo ra file nàyPrivate sub
cmdSaveAs_Click()dlgHopThoai.Filter = "Tất cả các file
(*.*|*.*)"|"dlgHopThoai.FileName =
"""dlgHopThoai.ShowSaveIfdlgHopThoai.FileName <> "" ThenrtfEditor.SaveFile
dlgHopThoai.FileName, rtfRTF'// rtfRTF Hoặc rtfText nếu muốn lưu sang dạng
file TextEndIfEndSub'// Hiển thị hộp thoại chọn font và thiết lập font chữ cho văn
bản trong FTFPrivate sub cmdSetFont_Click()dlgHopThoai.Flags = cdlCFBoth
Or cdlCFFEffectsdlgHopThoai.ShowFontrtfEditor.Font.Name =
dlgHopThoai.FontNamertfEditor.Font.Size =
dlgHopThoai.FontSizertfEditor.Font.Bold =
dlgHopThoai.FontBoldrtfEditor.Font.Italic =
dlgHopThoai.FontItalicrtfEditor.Font.Underline =
dlgHopThoai.FontUnderlinertfEditor.Font.Strikethrough =
dlgHopThoai.FontStrikeThroughEndSub'// Hiển thị hộp thoại chọn màu sau đó đặt
màu vừa chọn cho nền của RTFPrivate sub
cmdSetBackColor_Click()dlgHopThoai.ShowColor'// Hiển thị hộp thoại chọn
màu rtfEditor.BackColor = dlgHopThoai.Color '// Đặt màu nền bằng màu vừa
chọnEndSub

```

Hiển thị hộp thoại Printer

Hộp thoại Print cho phép người dùng chọn tên máy in, số trang, số bản sao v.v... Hộp
thoại này cũng KHÔNG TỰ IN ĐƯỢC TÀI LIỆU.

Thực hành: Thêm một nút lệnh "Print" vào **Form** và đặt Name = cmdPrint, Caption =
"&Print". Khi người dùng click nút này thì hiển thị hộp thoại máy in.

Các bước thực hiện và giao diện giống như phần 4.

Viết thêm đoạn code xử lý sự kiện click nút cmdPrint như sau:

Private sub cmdPrint_Click()

dlgHopThoai.ShowPrinter

MsgBox "Số bản in :" & **dlgHopThoai.Copies**

End Sub

Hiển thị hộp thoại HELP

Hộp thoại Help có nhiệm vụ thực thi một file trợ giúp dạng Window Help (Các file help cũ có phần mở rộng là hlp, các file mới hiện nay có phần mở rộng là CHM).

Thực hành: Mở file trợ giúp c:\winnt\winhelp.hlp.

Các bước thực hiện: Thêm một nút vào **Form**, đặt name = cmdHelp, Caption = &Help và viết lệnh cho sự kiện click chuột cho nút này như sau:

```
Private sub cmdHelp_Click()  
  
dlgHopThoai.HelpFile = "c:\winnt\winhelp.HLP"  
  
dlgHopThoai.HelpCommand = cdIHelpContents  
  
dlgHopThoai.Showhelp  
  
End Sub
```

Toàn bộ giao diện của bài thực hành

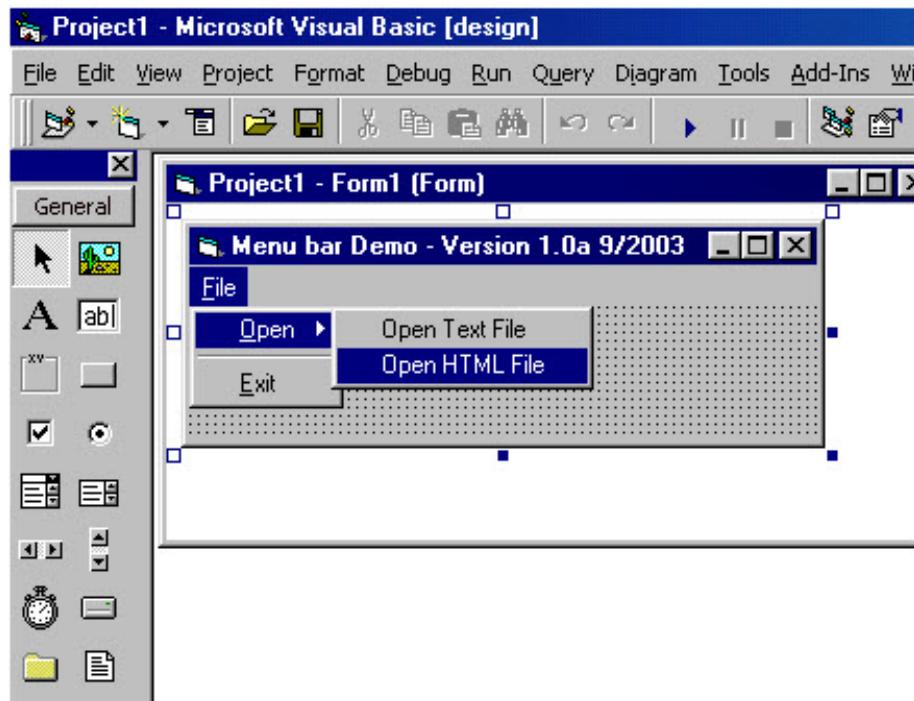


Giao diện chương trình hoàn chỉnh

Bài thực hành số 5: Sử dụng Menu và các thanh công cụ

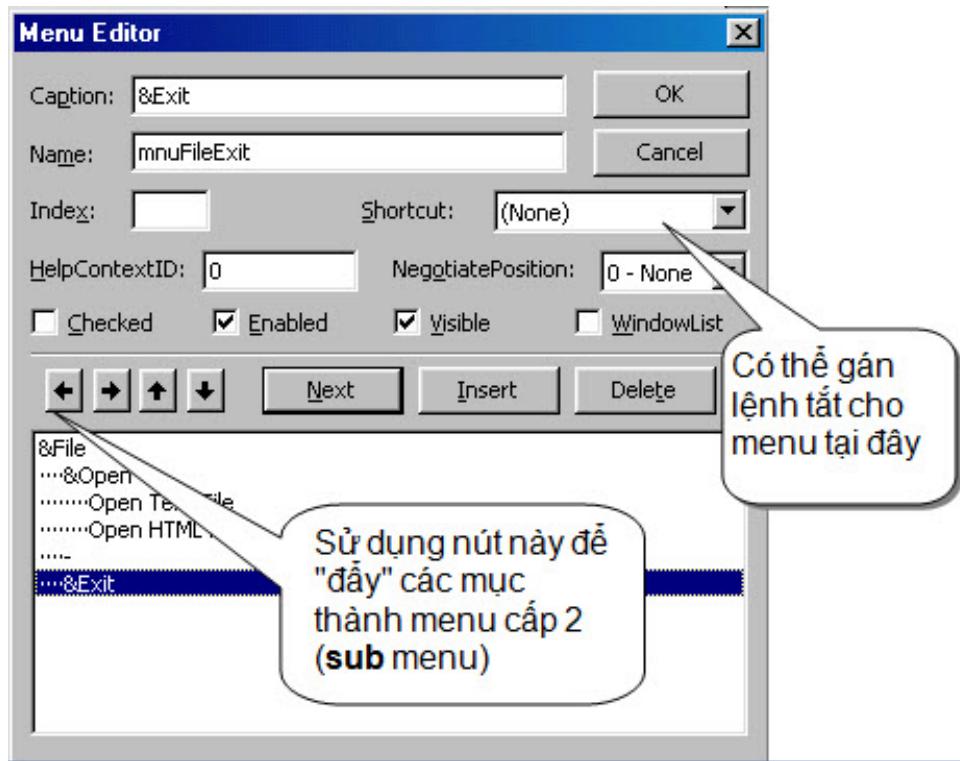
Tạo menu có nhiều cấp

Thực hành: Tạo hệ thống Menu như hình dưới



Menu có nhiều cấp

Các bước thực hiện: Tương tự như phần 1 và 2 (Tham khảo thêm hình).



Soạn thảo các mục của menu có 2 cấp

Ghi chú:

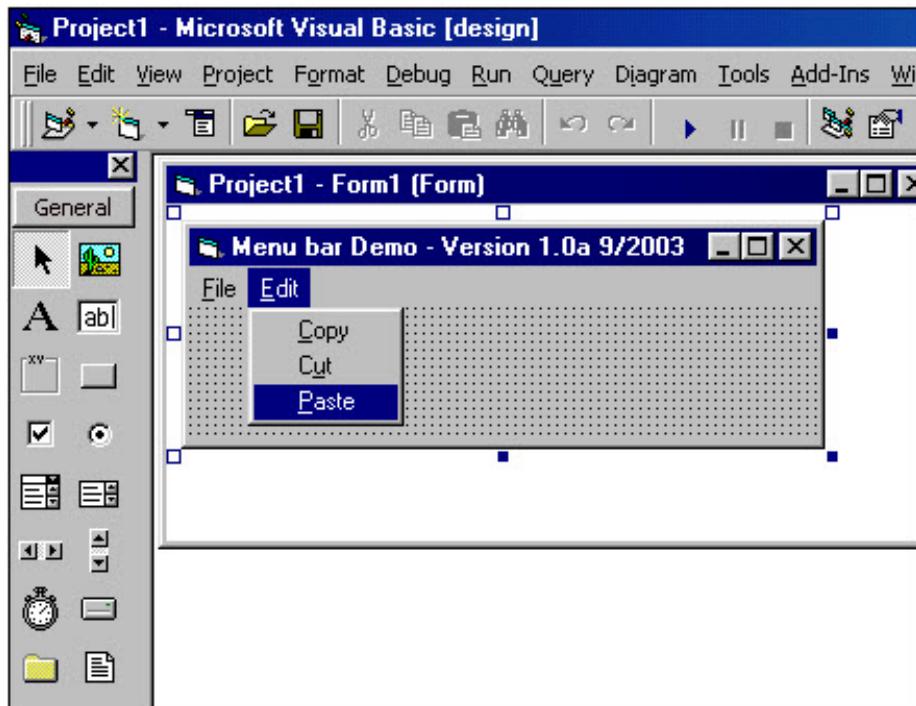
- Hệ thống menu trong VB có tối đa là 4 mức con (**Sub** menu)
- Có thể ẩn / hiện các mục menu cha và menu con bằng cách đặt thuộc tính **Visible** là true và False.
- Có thể thêm các dấu kiểm tra (Check mark) vào mỗi mục menu bằng cách đặt thuộc tính **Checked = True** hay false. Hay thường thêm câu lệnh sau vào sự kiện Click để ẩn/hiện menu, ví dụ:

`mnuOpen.Checked = Not mnuOpen.Checked`

- Có thể gán phím tắt cho các mục menu, ví dụ Ctrl-O, Ctrl-X,... bằng cách chọn hộp Shortcut (Như hình trên)

Tạo một Menu ngang có nhiều mục

Thực hành: Tạo hệ thống Menu như Hình 3 sau đây



Menu ngang có nhiều mục

Các bước thực hiện :

Bước 1: Tạo menu File và các mục con của nó → Giống như phần 1 ở trên. Sau khi soạn xong, đừng click chọn **OK**. Thay vào đó bạn click vào nút **Next** để thêm menu **Edit** và các mục con của **Edit**. Lưu ý: khi bạn click **Next** và gõ &Edit vào ô caption, mnuEdit vào ô Name thì VB sẽ hiểu Edit là một mục con của Menu File. Ở đây ta muốn menu Edit ngang cấp với menu File (như trên màn hình), do vậy sau khi gõ &Edit và mnuEdit xong, bạn cần click chọn vào biểu tượng để Edit trở thành menu cấp 0.

Bước 2: Tạo các mục

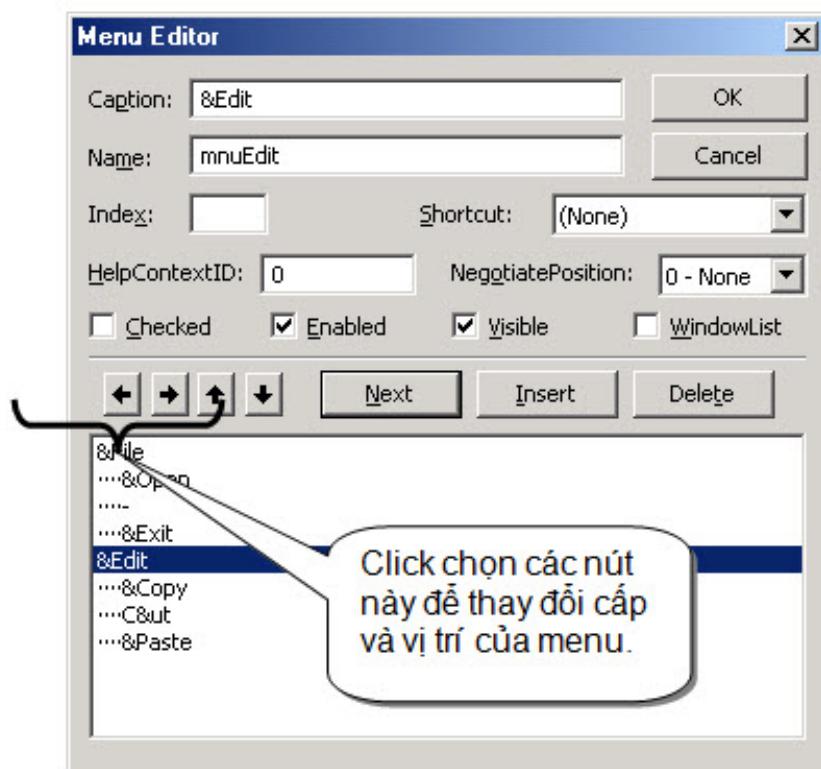


con cho menu Edit

- Tiếp theo các bước trước, bạn click vào nút **Next** và soạn các mục **Copy**, **Cut**, **Paste**, lưu ý là bạn phải đặt các mục menu này thuộc mục con của menu **Edit** bằng cách sử dụng các nút :



- Click vào nút **OK**. (Xem hình minh họa)

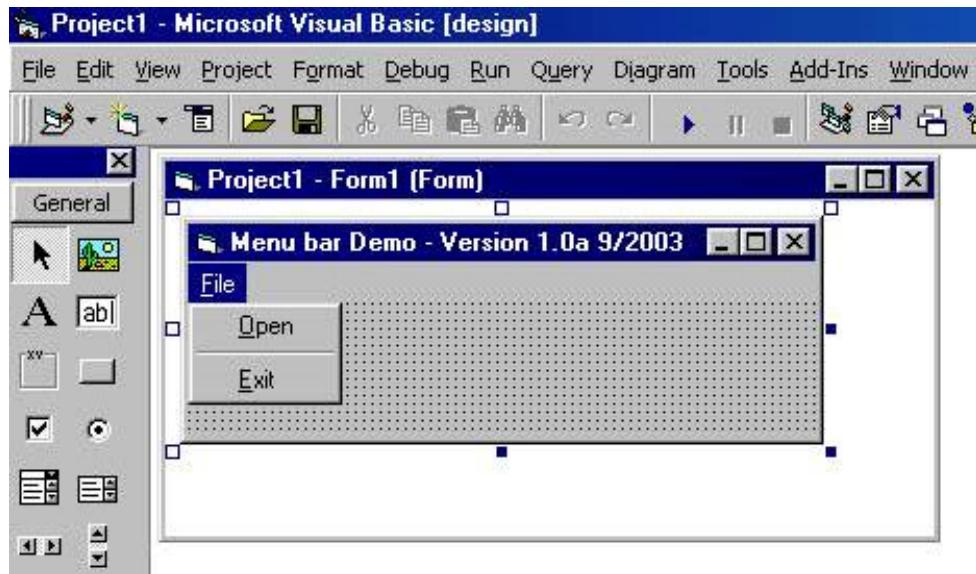


Soạn các mục menu Edit

Bước 3: nhấn F5 để chạy và kiểm tra chương trình.

Tạo một Menu ngang (Menu bar) đơn giản.

Yêu cầu: Xây dựng một menu ngang như *Hình 1* sau đây:



Hệ thống menu ngang 1 cấp đơn giản

Các bước thực hiện:

Bước 1: Vào menu Tools->Menu Editor (Hoặc ấn Ctrl+E, hoặc chọn biểu tượng



) để hiển thị hộp thoại soạn thảo menu.

Bước 2: Tạo menu File và các mục con của nó:

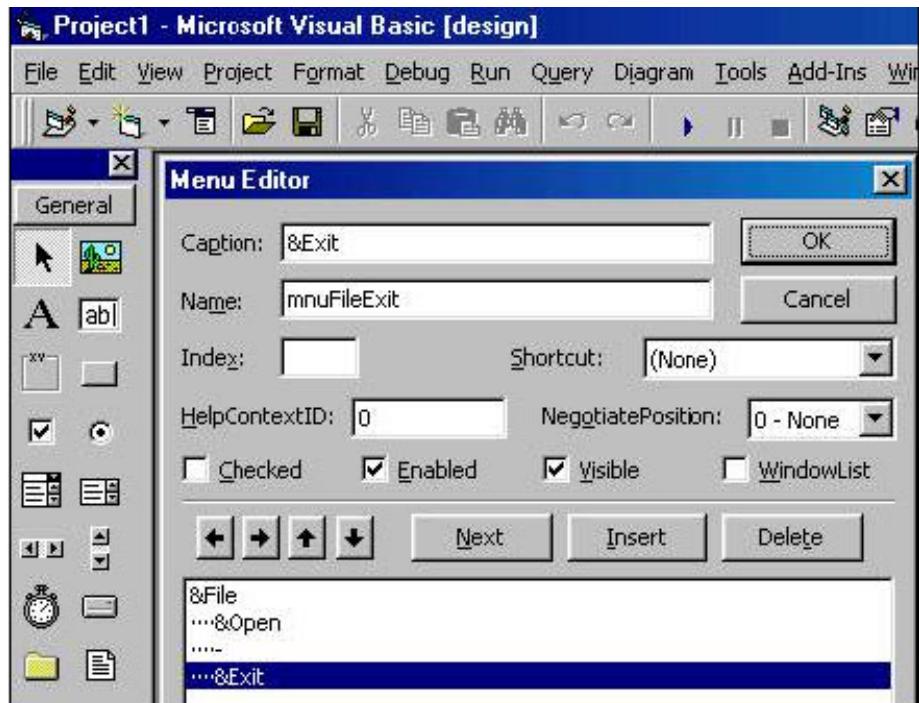
- Soạn trong ô Caption: &**File** (Menu này gọi là menu mức 0)
- Soạn trong ô Name: **mnuFile**, sau đó ấn Enter để thêm mục **Open**.
- Soạn trong ô Caption: &**Open**
- Soạn trong ô Name: **mnuFileOpen** (sau khi soạn xong đừng Enter vội !!!)
- Click vào biểu tượng



để cho VB biết rằng, mục menu Open là một mục con của menu File. (Khi đó menu Open được gọi là menu cấp 1 của menu File)

- Tiếp theo click **Next** để thêm thanh phân cách.
- Soạn trong ô Caption **một dấu gạch ngang** để tạo thanh phân cách: -
- Soạn trong ô Name: **mnuThanhphancach1**, sau đó click **Next** để thêm mục **Exit**.

- Soạn trong ô Caption: &Exit
- Soạn trong ô Name: mnuFileExit, sau đó click OK để kết thúc



Hộp thoại soạn các mục Menu

Bước 3: Nhấn F5 để chạy chương trình. Lưu ý là khi chạy chương trình nếu bạn chọn các mục Open và Exit thì chương trình vẫn không thực hiện bất kỳ một công việc nào bởi vì ta chưa viết lệnh cho nó.

Viết lệnh cho các mục của menu

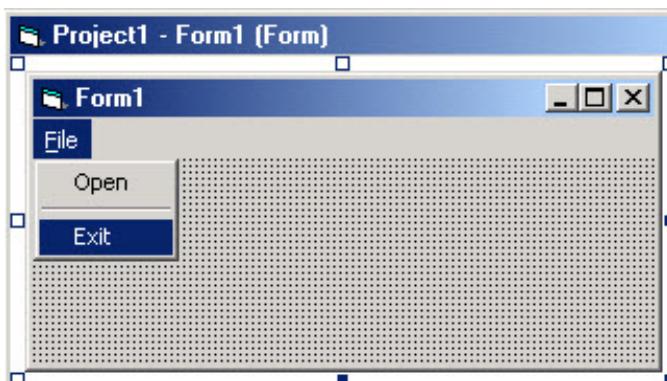
Ở các phần thực hành trước, khi chạy chương trình thì các mục menu sẽ không thực hiện bất cứ một câu lệnh gì khi người dùng click chọn. Phần thực hành này sẽ minh họa việc gắn kết các mục chọn với việc thực thi chương trình tương ứng với mục mà người dùng lựa chọn.

Thực hành: Thực hiện thoát khỏi chương trình khi người dùng chọn mục Exit.

Thiết kế menu: Như phần 1

Viết lệnh:

Để yêu cầu máy tính thực hiện một công việc nào đó khi người dùng click chọn một mục Menu, Tại thời điểm thiết kế, Bạn hãy click chọn mục menu này, sau đó một thủ tục (gọi là thủ tục sự kiện click) sẽ tự sinh ra để viết lệnh. Trong trường hợp này, ta sẽ thoát khỏi chương trình khi người dùng click chọn menu Exit. Như vậy, hãy chọn mục Exit trong hình vẽ và viết code như sau:



Option Explicit

'/// Thủ tục này sẽ được gọi khi người dùng click menu Exit

Private Sub mnuExit_Click()

End '/// Thoát khỏi chương trình

End Sub

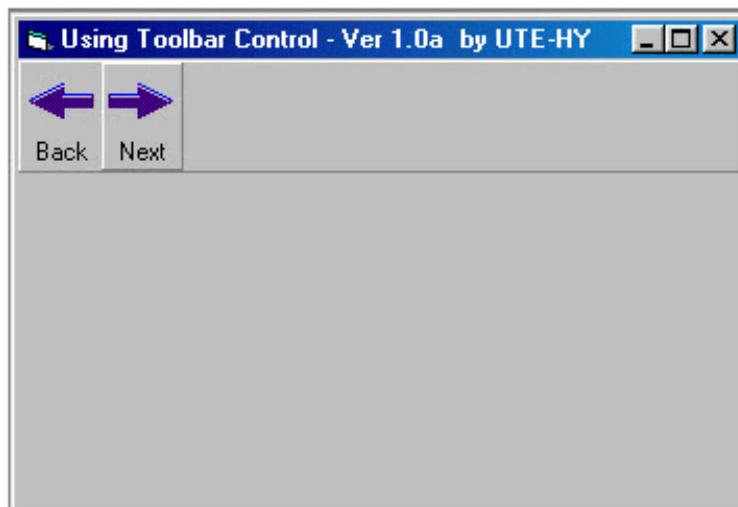
Tạo thanh công cụ Toolbar

Thanh công cụ giúp chúng ta truy cập đến các chức năng của chương trình được nhanh hơn. Mặt khác nó còn nâng cao tính thẩm mỹ cho chương trình. Điều khiển này không phải là điều khiển nội tại của VB mà nó được gói trong tệp tin có tên là MSCOMCTL.OCX. Khi sử dụng thanh Toolbar chúng ta thường phải thêm một control khác là ImageList (control này cùng nằm trong tệp MSCOMCTL.OCX với Toolbar), ImageList có khả năng lưu trữ các hình ảnh và biểu tượng để Toolbar sử dụng vào việc hiển thị trên các nút.

Để xây dựng một thanh Toolbar, chúng ta cần phải theo mấy bước chính như sau:

- Kéo đối tượng Toolbar và ImageList vào **Form**
- Chèn các icon (biểu tượng) vào ImageList để nó lưu trữ
- Gắn ImageList với Toolbar
- Chèn các nút trong Toolbar và chỉ định icon cho nó

Thực hành: xây dựng toolbar theo như hình dưới đây



Giao diện chương trình

Các bước thực hiện

- Vào menu → Project → Components (Hoặc ấn tổ hợp phím Ctrl+T)
- Đánh dấu chọn () vào mục “Microsoft Windows Common Control 6.0 (SP4)”, sau đó click nút OK. Trong hộp ToolBox sẽ xuất hiện các control vừa mới thêm.
- Vẽ 2 control ToolBar ()

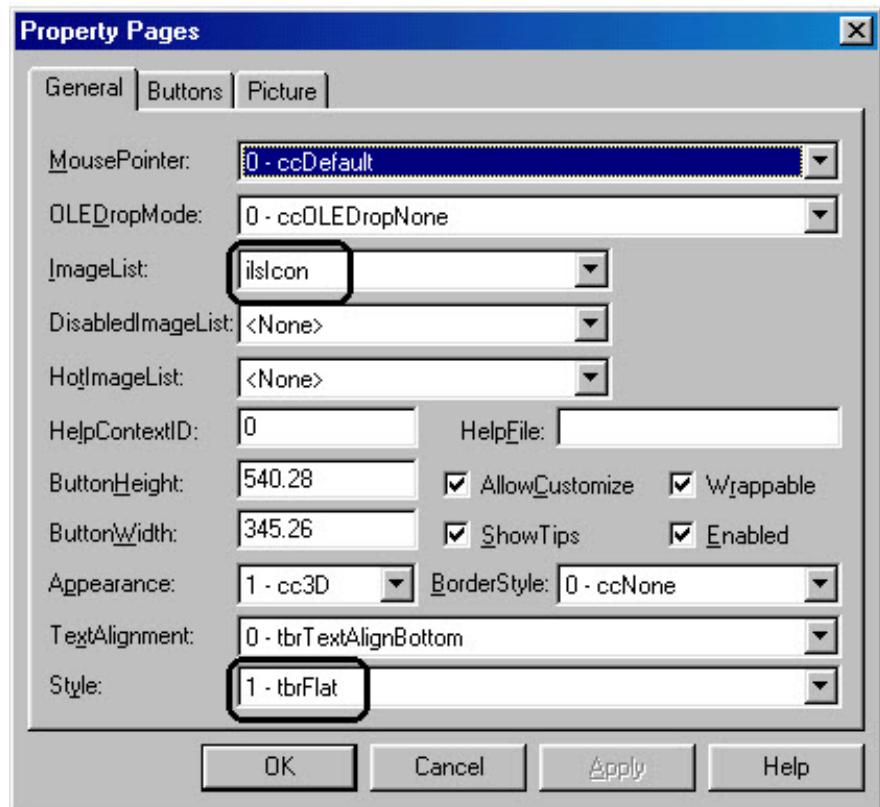
) và ImageList (



) vào **Form**.

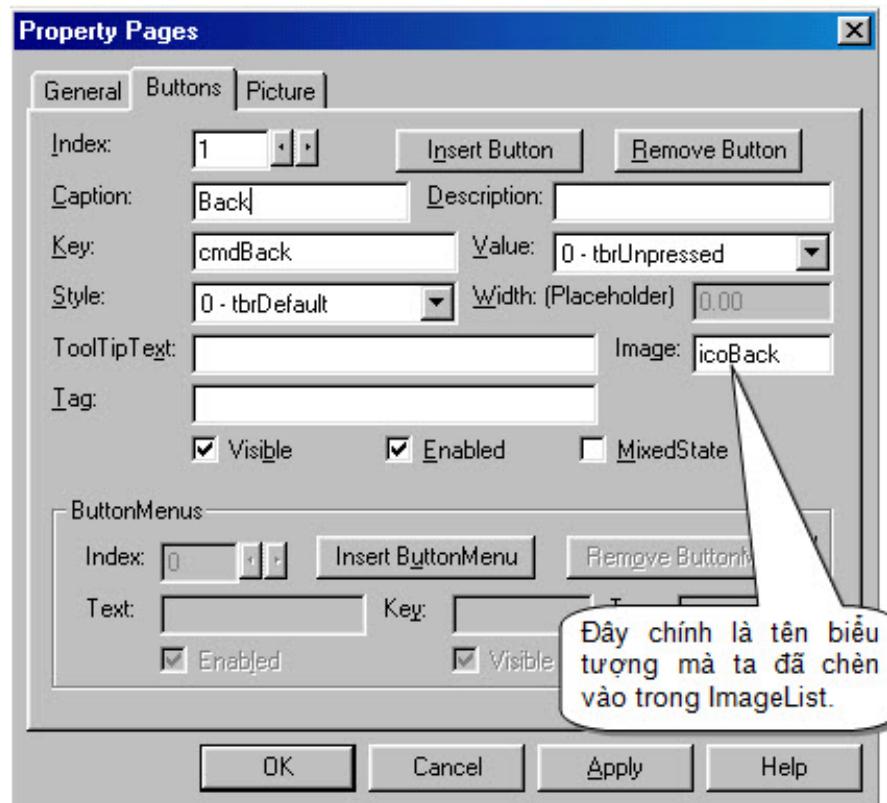
- Đặt tên: **tbrMenu** cho Toolbar và **imlIcon** cho ImageList
- Chèn các icon vào ImageList (Các icon này sẽ gắn vào các nút của toolbar):
 - Trong cửa sổ Properties, chọn đối tượng **imlIcon**, sau đó click đúp lên thuộc tính Custom.

 - (Hoặc click lên nút có biểu tượng 3 dấu chấm bên phải) để mở hộp thoại chèn icon.
 - Trong tab General đầu tiên, chọn kích thước khi hiển thị icon là 16x16
 - Sau đó chọn tab (trang/thẻ) Images
 - Click vào nút Insert Picture để thêm biểu tượng thứ nhất: bạn hãy trỏ đến file icon theo đường dẫn sau: C:\Program Files\Microsoft Visual Studio\Common\Graphics\Icons\Arrows\Arw05lt.ico. Sau đó click vào nút Open để thêm vào **imlIcon**. Đặt thuộc tính Key cho icon này là: **icoBack** (ico → tiền tố cho icon). Thuộc tính Key này rất quan trọng, nó là một tên để phân biệt icon này với icon khác và để chúng ta tham chiếu khi đặt icon cho các nút. Đừng quên đặt Key cho các icon khi thêm !
 - Tương tự bước trên để thêm icon thứ 2, là Key là **icoNext**. Hãy ghi nhớ thuộc tính Key của các icon này và lưu ý là có phân biệt giữa chữ thường và chữ HOA : **icoNext** ≠ **iconext**. Tiếp theo chọn OK để đóng hộp thoại chèn Icon lại.
- Đặt thuộc tính cho Toolbar và chèn các nút
 - Chọn đối tượng **tbrMenu** và chọn Custom trong cửa sổ Properties, hộp thoại sau sẽ hiện ra:

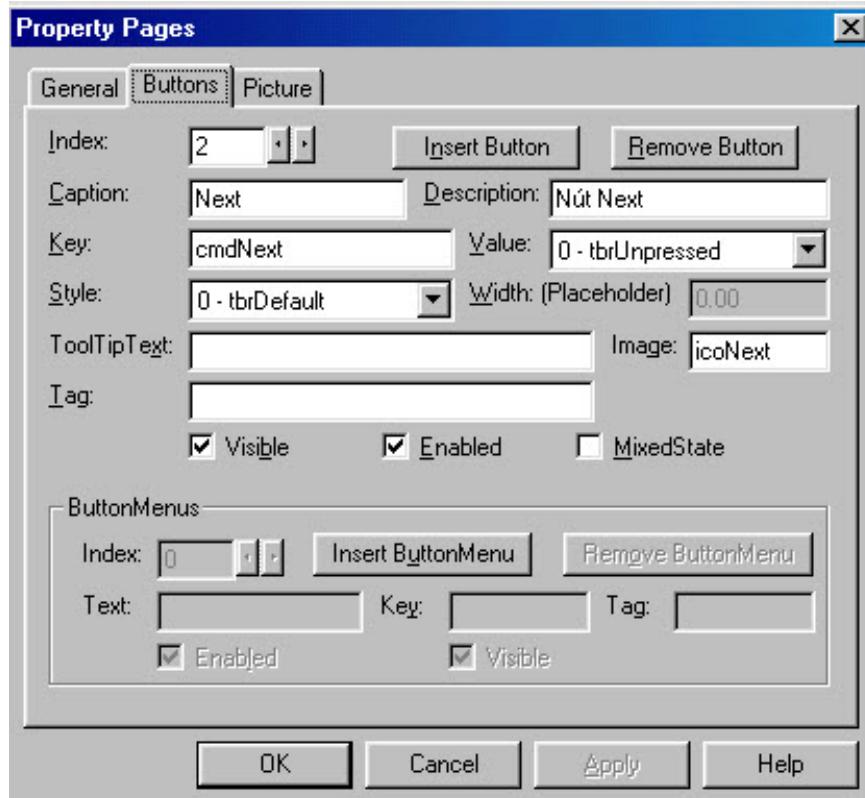


Gắn kết giữa ImageList vào Toolbar

- Trong trang General, đặt thuộc tính ImageList là imlIcon (Điều này nói lên rằng: Các icon sử dụng cho các nút của toolbar sẽ được lấy trong điều khiển imlIcon)
- Đặt thuộc tính Style là Flat (Bạn có thể không đặt thuộc tính này)
- Chọn sang trang (thẻ) Buttons. Click vào nút **Insert** để chèn nút và điền vào các thuộc tính như trong hình.
- Sau khi điền xong, click vào nút Insert để chèn nút thứ hai, và soạn các mục như hình kê tiếp
- Click vào nút OK để kết thúc
- Ấn F5 để chạy chương trình nếu bạn thực hiện đúng các bước như trên.
- Hãy thay đổi các thuộc tính khác như ToolTip Text, Visible... và tự kiểm tra kết quả.



Tạo các nút của Toolbar



Tạo các nút của Toolbar

- Ghi chú:

Có thể thêm vào ToolBar các thanh phân cách, ví dụ Thêm thanh phân cách giữa các nút của toolbar như sau:

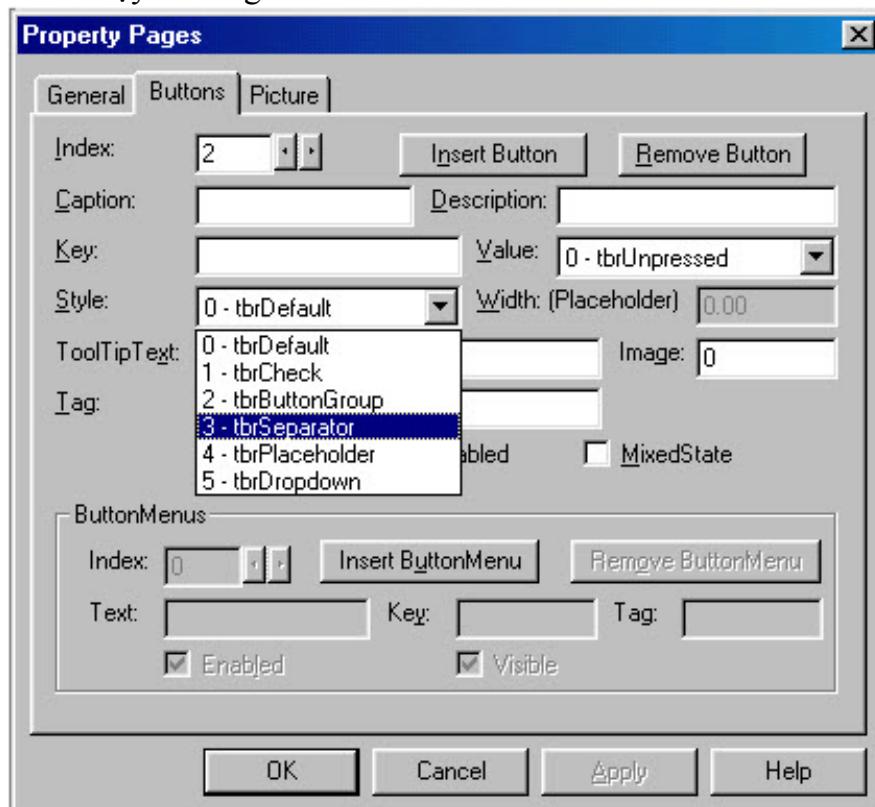
- Mở hộp thoại **Property page** của **tbrMenu** như ở phần trước
- Di chuyển đến mục số 1 (mục Back) bằng cách dùng các nút



- Click vào nút Insert để chèn một nút mới (Nút này sẽ là thanh phân cách)
- Đặt thuộc tính Style là 3-tbrSeparator như trong sau (



-)
- Ấn F5 để chạy chương trình.



Tạo thêm thanh ngăn cách giữa các nút

Viết lệnh cho các nút trên thanh công cụ

Thực hành: Thực hiện lệnh thoát khỏi chương trình khi người dùng click vào nút Back và hiển thị thông báo là "Bạn đã click vào nút Next" khi người dùng click vào nút Next.

Thiết kế giao diện: như phần 5

Viết lệnh: Mở cửa sổ code Editor, viết lệnh cho thủ tục sự kiện ButtonClick của Toolbar Option Explicit

/// Thủ tục này được kích hoạt khi người dùng click chuột lên bất kỳ nút nào của Toolbar

Private Sub tbrMenu_ButtonClick(ByVal Button As MSComctlLib.Button)

SelectCase Button.Key

Case "cmdBack" */// Nếu người dùng click vào nút Back thì thoát chương trình*

End

Case "cmdNext"

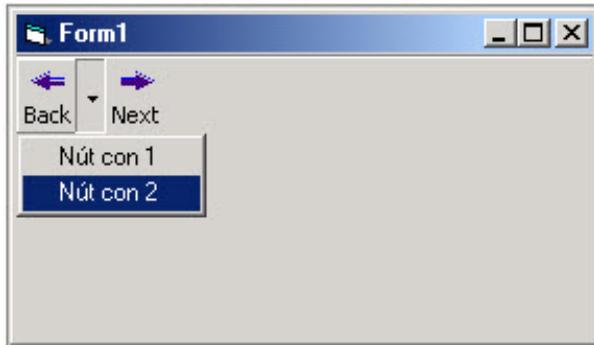
MsgBox "Bạn đã click vào nút Next trên toolbar", vbInformation

End Select

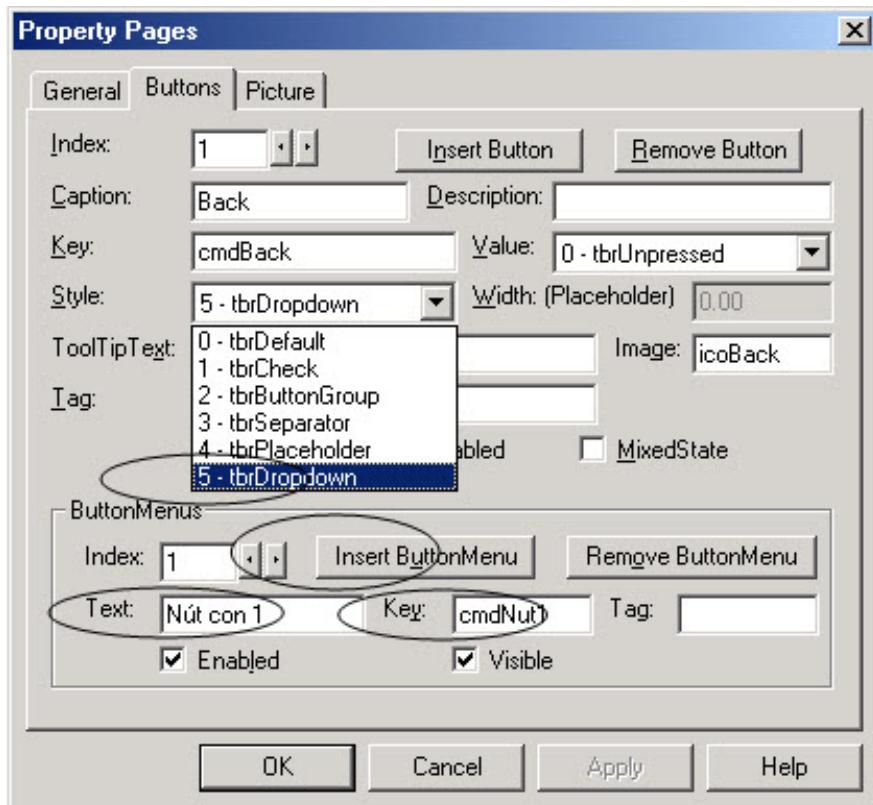
End Sub

Ghi chú:

- Sự kiện ButtonClick được kích hoạt khi người dùng click chọn một mục bất kỳ trên thanh công cụ, khi đó thuộc tính Button.Key sẽ trả về Tên (Name) của nút vừa bị click. Thuộc tính này được sử dụng để viết lệnh tương ứng (như ví dụ trên)
- Mỗi nút trên thanh công cụ có thể lại chứa một danh sách các nút khác bằng cách đặt thuộc tính style là : 5 – Dropdown



Mỗi nút chứa một menu xổ xuống (Dropdown)



Tạo các mục menu cho nút Back

- Để viết lệnh ứng với khi người dùng click chọn các mục menu con, ta viết trong thủ tục sự kiện: ButtonMenuClick, như ví dụ sau:

```
Private Sub tbrMenu_ButtonMenuClick(ByVal ButtonMenu As
MSComctlLib.ButtonMenu)
```

```
SelectCase ButtonMenu.Key
```

```
Case "cmdNut1" : MsgBox "Bạn đã click mục menu 1 của nút Back"
```

```
Case "cmdNut2": MsgBox "Bạn đã click mục menu 2 của nút Back"
```

'.....

End Select

End Sub

Xây dựng chương trình soạn thảo văn bản đơn giản

Trong bài thực hành này, để minh họa việc đưa hệ thống menu vào chương trình, ta sẽ đi xây dựng một ứng dụng soạn thảo văn bản đơn giản.

Thực hành: Xây dựng chương trình soạn thảo văn bản có các tính năng

- Mở file, Soạn thảo và lưu file dạng TXT hoặc RTF
- Định dạng kiểu chữ cho văn bản.
- Giao diện như hình.



Giao diện chương trình.

Các bước thực hiện:

Bước 1: Đưa các điều khiển vào Form:

Vào menu Project → Components, và Click chọn 3 OCX sau:



Thêm các file OCX cần cho ứng dụng

Bước 2: Kéo các điều khiển Dialog

, RichTextbox



vào Form

Đặt giá trị cho các thuộc tính của mỗi điều khiển.

Điều khiển	Thuộc tính / Giá trị
 Menu File	<u>File</u> : mnuFile/&File (Theo thứ tự Tên / Caption) <u>Mở file</u> : mnuOpen/ Mở file <u>File Text (*.txt)</u> : mnuFileTXT/ File Text <u>File Text (*.rtf)</u> : mnuFileRTF/ File Text <u>Lưu file</u> : mnuSave/ Lưu file <u>Thoát</u> : mnuExit/Thoát
 Menu Font chữ	<u>Font chữ</u> : mnuFont / F&ont chữ <u>Chữ đậm</u> : mnuBold / Chữ đậm <u>Chữ nghiêng</u> : mnultalic / Chữ nghiêng <u>Chữ thường</u> : mnuNormal / Chữ thường
Dialog Command control	Name : dlgChonFile (Chọn file)
Richtextbox control	Name : rtfEditor

Viết lệnh:

? Khi người dùng mở file Text (Click chọn mục Mở file → File Text (*.txt)):

'/// Mở hộp thoại chọn file, sau đó nạp file vừa chọn vào trong richtext box để soạn thảo

Private Sub mnuFileText_Click()

dlgChonFile.DialogTitle = "Chọn file cần mở"

dlgChonFile.Filter = "Các file text (*.txt)" '/// Chỉ hiện những file có phần mở rộng txt

dlgChonFile.ShowOpen '/// Mở hộp thoại để người dùng chọn file

rtfEditor.LoadFile dlgChonFile.FileName '/// Nạp file vừa chọn vào Richtextbox

End Sub

? Khi người dùng mở file RTF (Click chọn mục Mở file → File RTF (*.rtf))

'/// Mở hộp thoại chọn file, sau đó nạp file vừa chọn vào trong richtext box để soạn thảo

Private Sub mnuFileRTF_Click()

dlgChonFile.DialogTitle = "Chọn file cần mở"

dlgChonFile.Filter = "Các file RTF |*.rtf" '/// Chỉ hiện những file có phần mở rộng rtf

dlgChonFile.ShowOpen '/// Mở hộp thoại để người dùng chọn file

rtfEditor.LoadFile dlgChonFile.FileName '/// Nạp file vừa chọn vào RichTextbox

End Sub

? Khi người dùng chọn menu Save (Click chọn mục File → Save)

'/// Lưu file đang soạn hiện tại vào đĩa. Lưu ý, tên và đường dẫn của file này vẫn còn
'/// trong thuộc tính dlgChonFile.FileName

Private Sub mnuSave_Click()

rtfEditor.SaveFile dlgChonFile.FileName '/// Gọi SaveFile để Lưu ra đĩa

End Sub

? Khi người dùng chọn menu Chữ đậm :

Private Sub mnuBold_Click()

rtfEditor.Font.Bold = True '/// Đặt font chữ văn bản trong richtextbox là đậm

End Sub

Tương tự cho các menu khác.

Dưới đây là toàn bộ chương trình nguồn :

Option Explicit

'/// Mở file RTF để soạn thảo

Private Sub mnuFileRTF_Click()

dlgChonFile.DialogTitle = "Chọn file cần mở"

```
dlgChonFile.Filter = "Các file RTF |*.rtf|" '/// Chỉ hiện thị các file RTF  
dlgChonFile.ShowOpen '/// Hiển thị hộp thoại chọn file  
rtfEditor.LoadFile dlgChonFile.FileName '/// Nạp file vừa chọn vào RichTextbox
```

End Sub

'/// Mở file TEXT để soạn thảo

```
Private Sub mnuFileText_Click()
```

```
dlgChonFile.DialogTitle = "Chọn file cần mở"
```

```
dlgChonFile.Filter = "Các file text |*.txt|" '/// Chỉ hiển thị các file Text
```

```
dlgChonFile.ShowOpen
```

```
rtfEditor.LoadFile dlgChonFile.FileName
```

End Sub

'/// Định dạng văn bản ở dạng chữ đậm.

```
Private Sub mnuBold_Click()
```

```
rtfEditor.Font.Bold = True
```

End Sub

'/// Định dạng văn bản ở dạng chữ nghiêng

```
Private Sub mnuItalic_Click()
```

```
rtfEditor.Font.Italic = True
```

End Sub

'/// Đặt văn bản trở về chữ thường

```
Private Sub mnuNormal_Click()
```

```
rtfEditor.Font.Italic = False
```

```
rtfEditor.Font.Bold = False
```

End Sub

'/// Lưu nội dung của file đang soạn ra đĩa. Bạn cũng có thể chỉ định lưu ra file khác

Private Sub mnuSave_Click()

```
rtfEditor.SaveFile dlgChonFile.FileName
```

End Sub

'/// Thoát khỏi chương trình

Private Sub mnuExit_Click()

End

End Sub

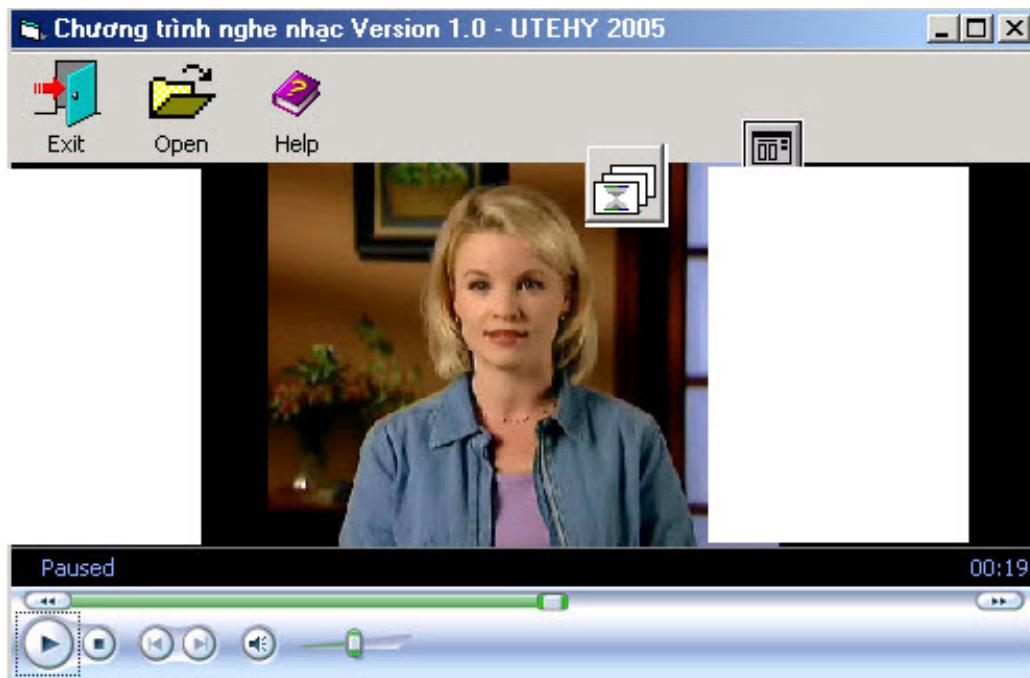
Ghi chú:

- Hộp thoại mở file (ShowOpen) chỉ trả về cho ta tên và đường dẫn của file mà người dùng chọn chứ không thể TỰ ĐỘNG MỞ file đó được. Ở đây chúng ta phải viết lệnh để mở file đó (Ví dụ dùng phương thức LoadFile của đối tượng RichTextbox để mở)
- Bạn có thể định dạng văn bản chỉ trong phạm vi bị bôi đen bằng cách thiết lập giá trị cho các thuộc tính tương ứng, ví dụ: rtfEditor.SelBold = True để đặt phần văn bản bị bôi đen trở thành đậm (phần văn bản khác không bị ảnh hưởng gì) v.v...

Xây dựng chương trình nghe nhạc đơn giản

Trong bài thực hành này, ta sẽ minh họa việc sử dụng Thanh công cụ (Toolbar) bằng cách xây dựng một ứng dụng nghe nhạc đơn giản.

Thực hành: Sử dụng điều khiển Window Media Player, viết chương trình nghe những file nhạc tiếng, nhạc hình được hỗ trợ bởi điều khiển này. Giao diện như hình 18.



Giao diện chương trình

Các bước thực hiện:

Bước 1: Đưa các file OCX vào dự án.

Vào menu Projects --> Components và click chọn các file OCX sau:



Thêm các OCX vào Form

Bước 2: Kéo 4 điều khiển Window Media Player

, Toolbar



, ImageList



và CommonDialog



vào Form.

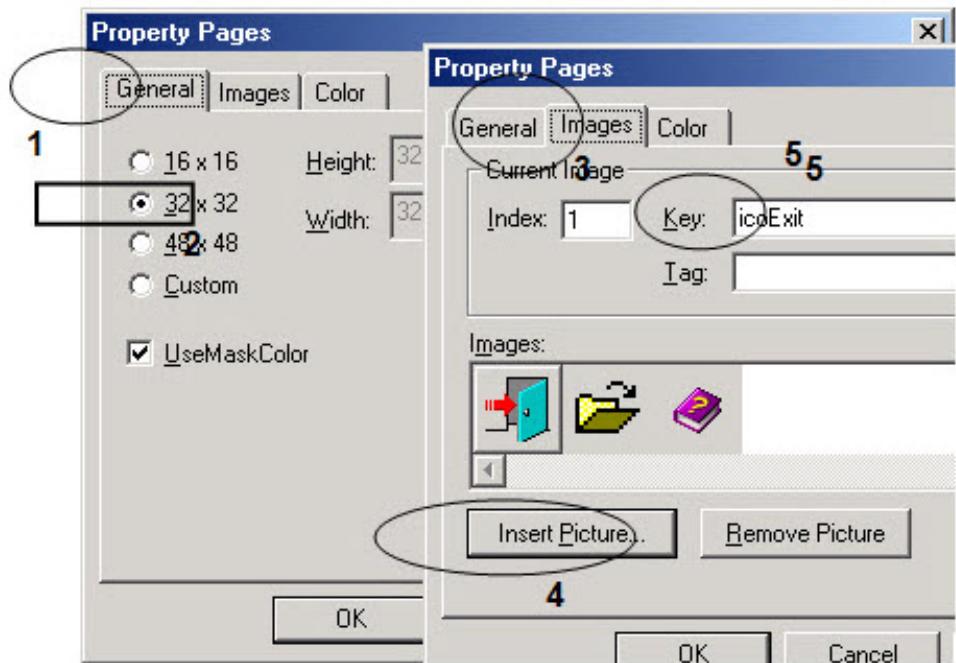
Bước 3: Đặt giá trị cho các thuộc tính

Điều khiển	Tên thuộc tính /Giá trị
Toolbar	Name : tbrMainStyle : Flat
ImageList	Name: imlIcons
CommonDialog	Name : dlgChonFile
WindowMediaPlayer	Name: wmpMain

Bước 4: Thêm các icons vào ImageList (Xin xem lại phần 5)

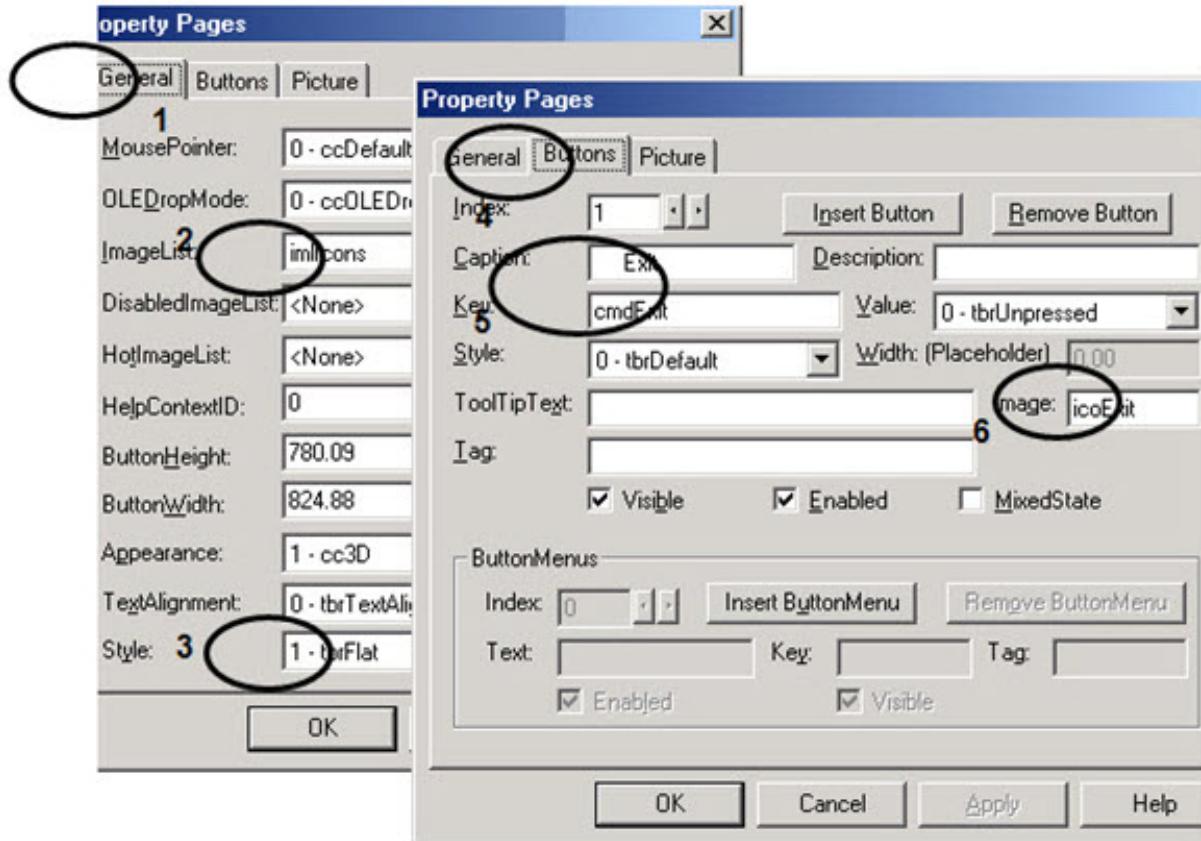
Thêm 3 icon vào imlIcon và đặt tên lần lượt là icoExit, icoOpen và icoHelp (Với mục đích minh họa, bạn có thể chọn 3 icon bất kỳ). Đặt thông số theo như hình 20:

5



Đặt các thông số cho ImageList

Bước 5: Thiết lập thông số cho Toolbar: tbrMain (Tham khảo phần 5) như hình dưới:



Thiết lập thông số choToolBar

Viết lệnh:

Option Explicit

'// Thủ tục này được gọi mỗi khi kích thước của Form bị thay đổi

'// Khi đó ta kéo giãn Window Media Player bằng với kích thước của Form . (có thể bỏ qua phần này)

Private Sub Form_Resize()

wmpMain.Width = Me.ScaleWidth

wmpMain.Height = Me.ScaleHeight - tbrMain.Height

End Sub

'// Thủ tục này được gọi mỗi khi người dùng click chọn một nút trên thanh công cụ

'/// Tại đây ta sẽ kiểm tra xem người dùng chọn nút nào và viết lệnh xử lý tương ứng

Private Sub tbrMain_ButtonClick(ByVal Button **As** MSComctlLib.Button)

SelectCase Button.Key

Case "cmdOpen" '/// Người dùng click nút Open

dlgChonFile.DialogTitle = "Chọn file để nghe" '/// Tiêu đề của hộp thoại

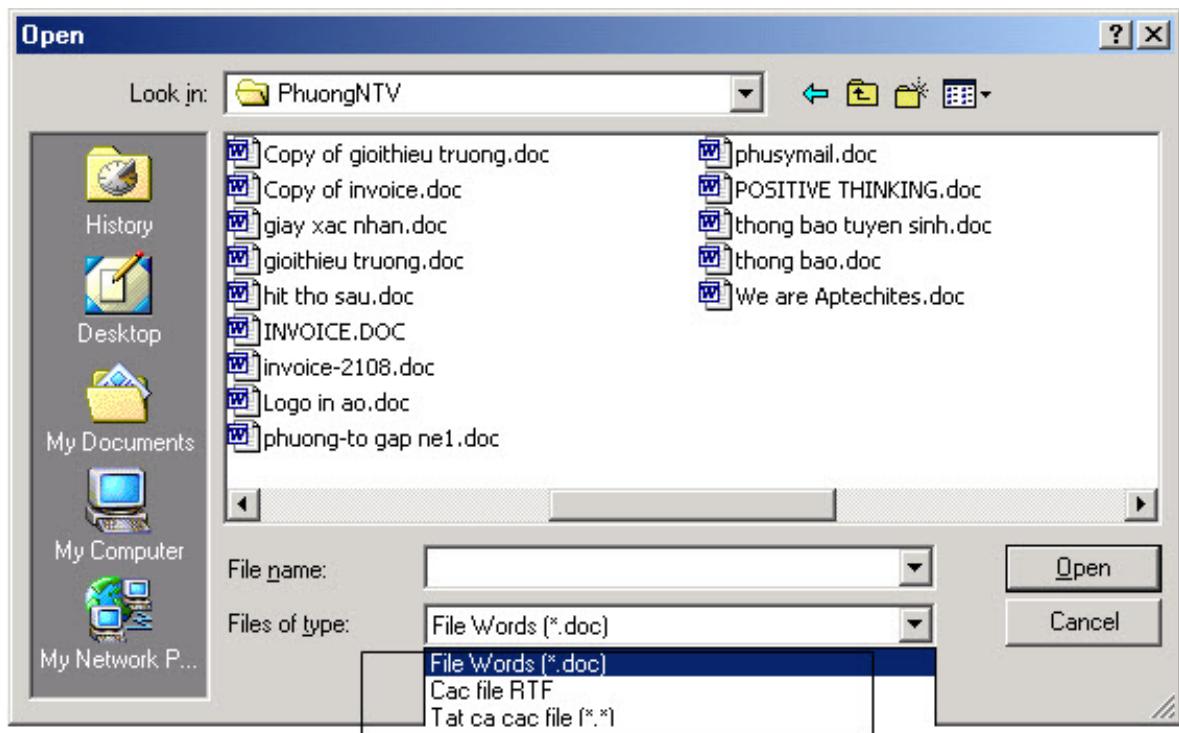
dlgChonFile.Filter = "Tất cả các file |*.*|"

- Nhấn F5 để chạy chương trình !

Ghi chú:

- Thuộc tính DialogTitle và Filter có thể bỏ qua.
- Trong trường hợp khi hộp thoại mở ra, nếu người dùng không chọn file nào (nhấn nút Cancel) thì thuộc tính FileName sẽ có giá trị của lần mở trước đó.
- Luôn luôn ghi nhớ rằng, Hộp thoại "Open file" **không tự động mở file** mà chỉ trả về thông tin duy nhất là tên file mà người dùng vừa chọn.
- Muốn có một hộp thoại cho người dùng chọn lựa các kiểu file khác nhau, cần đặt thuộc tính Filter như sau (*Thành phần (*.doc| *.rtf|... → rất quan trọng*):

dlgChonFile.Filter = "File Words (*.doc) (*.doc|Cac file RTF|*.rtf|Tat ca cac file (*.*)|*.*|"



Tùy biến hiển thị các kiểu file khác nhau

Bài thực hành số 6: Tạo, thao tác với cơ sở dữ liệu và sử dụng các đối tượng

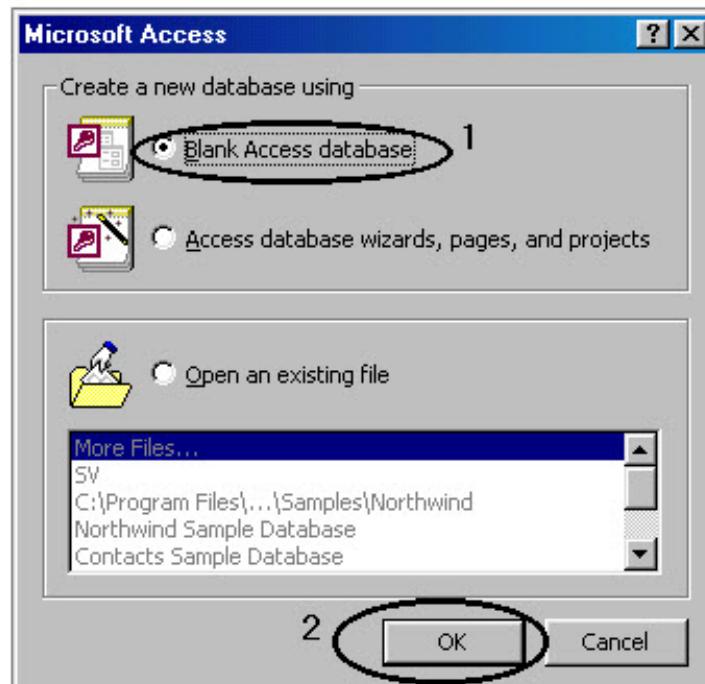
Tạo một bảng CSDL trong Microsoft Access 2000

Thực hành: Tạo một bảng CSDL gồm các trường như sau :

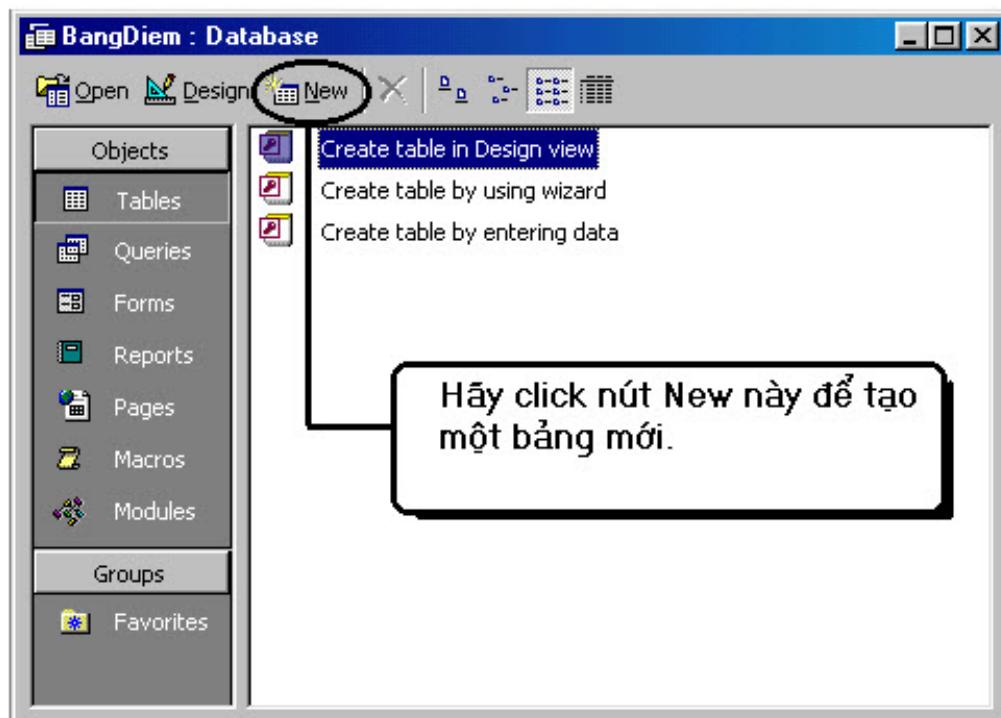
Tên trường	Kiểu của trường	Miêu tả
SoTT	AutoNumber	Số thứ tự
HoTen	Text	Họ và tên
DiemToan	Number	Điểm toán
DiemLy	Number	Điểm lý
DiemHoa	Number	Điểm hóa

Các bước thực hiện:

- Khởi động Access 2000: Chọn Start->Programs->Microsoft Access 2000. Một hộp thoại sẽ hiển thị và thông báo là mở file CSDL đã có hay tạo file mới. Hãy chọn mục 1 và 2 như H.1 để tạo một file mới.
- Đặt tên file: Tiếp theo Access yêu cầu nhập tên file, bạn hãy gõ: **DiemThi** và nhấn **Enter**. DiemThi chính là tên tập CSDL. Trong Access mỗi tập CSDL có thể chứa nhiều hơn một bảng (Table). Ở đây chúng ta chỉ cần tạo một bảng.



Tạo cơ sở dữ liệu mới



Tạo bảng cơ sở dữ liệu

- Click chọn mục **Create table in design view** và chọn **New** như hình 2 để tạo một bảng CSDL mới.
- Chọn chế độ thiết kế là **Design View** và click **OK** như hình 3.



Thiết kế bảng ở chế độ Design view

Soạn thảo các trường như hình số 4. *Ghi chú* khi tạo 3 trường số là DiemToan, DiemLy, DiemHoa. Ở đây ta phải xác định xem 3 trường này là trường số thuộc dạng gì (Số nguyên-Integer; Số nguyên dài – LongInteger hay Số thực-Single...) ? Vì điểm có thể có phần thập phân do vậy ta chọn các trường này có kiểu là Single như trong hình 4.

Field Name	Data Type	Description
SoTT	AutoNumber	Số thứ tự
Hoten	Text	Họ và tên
DiemToan	Number	Điểm toán
DiemLy	Number	Điểm lý
DiemHoa	Number	Điểm hoá

Lựa chọn kiểu cho trường loại số

- Click chọn trường **AutoNumber** và click vào biểu tượng chiếc chìa khoá



như hình số 5. Điều này để nói lên rằng: “Trường SoTT là một trường khoá” tức là một giá trị duy nhất và mỗi người khác nhau sẽ có SoTT khác nhau.

	Field Name	Data Type	Description
▶	SoTT	AutoNumber	Số thứ tự
	Hoten	Text	Họ và tên
	DiemToan	Number	Điểm toán
	DiemLy	Number	Điểm lý
	DiemHoa	Number	Điểm hoá

Đặt khoá chính cho bảng

- Click vào biểu tượng



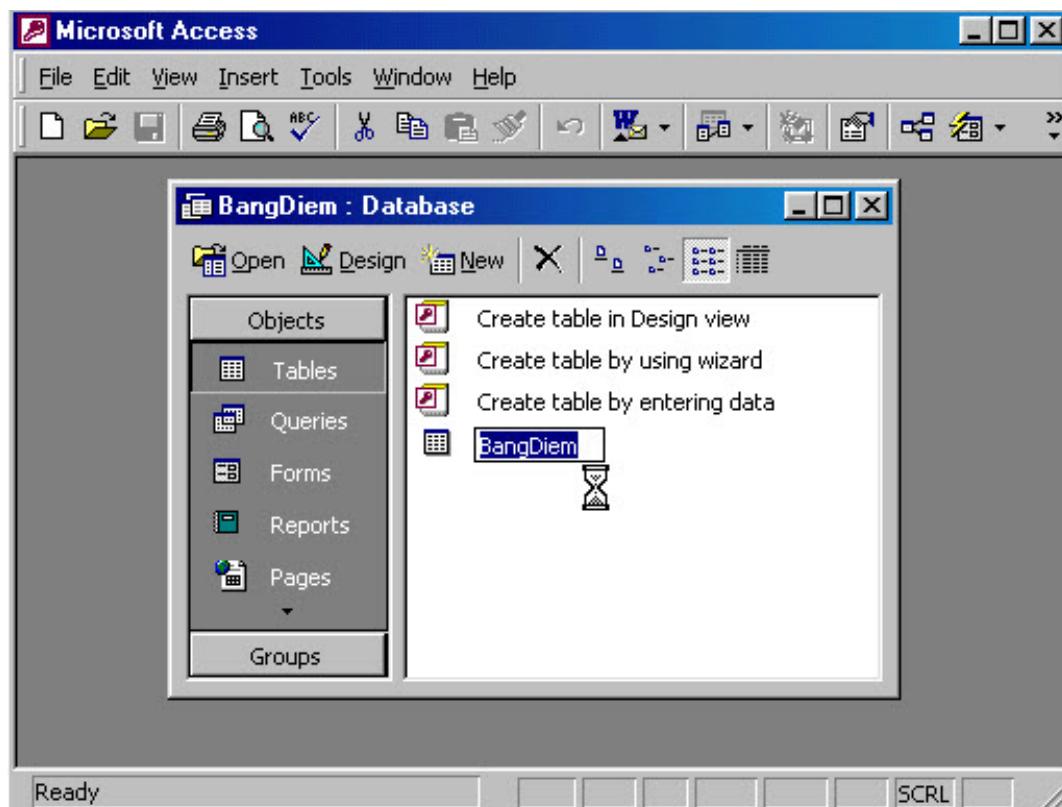
để lưu CSDL và gõ tên bảng là **BangDiem**



Đặt tên cho bảng

Nhập dữ liệu (Bản ghi) cho bảng:

- Click đúp lên bảng **BangDiem** như H.7 để nhập dữ liệu, nhưng *Ghi chú* là không phải nhập trường AutoNumber, nó sẽ tự động tăng lên. Xem H.8



Mở bảng để nhập dữ liệu

H.7 - Mở bảng để nhập dữ liệu

The screenshot shows the Microsoft Access application window titled "Microsoft Access". The menu bar includes File, Edit, View, Insert, Format, Records, Tools, Window, and Help. The toolbar contains icons for file operations and record navigation. The main window title is "BangDiem : Table". The table structure has columns: SoTT, Hoten, DiemToan, DiemLy, and DiemHoa. The data rows are:

SoTT	Hoten	DiemToan	DiemLy	DiemHoa
1	Nguyễn Tiến Anh	8	8	8
2	Nguyễn Trọng Đặng	8	9	7
3	Đinh Quang Đức	10	8	6
[AutoNumber]		0	0	0

Record: [navigation buttons] 4 [next button] of 4

Nhập dữ liệu cho bảng

Kết nối đến CSDL sử dụng đối tượng ADO Data Control

Đưa điều khiển ADO Data Control vào Form

- Vào menu Project → Components (Hoặc nhấn tổ hợp phím Ctrl+T)
- Click chọn mục Microsoft ADO Data Control và click OK
- Kéo (Drag) điều khiển ADO Data Control từ ControlBox vào from, Đặt tên (Name) cho điều khiển này là **adoDiemThi**.

Thiết lập thông số kết nối cho ADO Data Control để thực hiện kết nối đến CSDL

Có 2 cách thiết lập thông số kết nối cho ADO Data control :

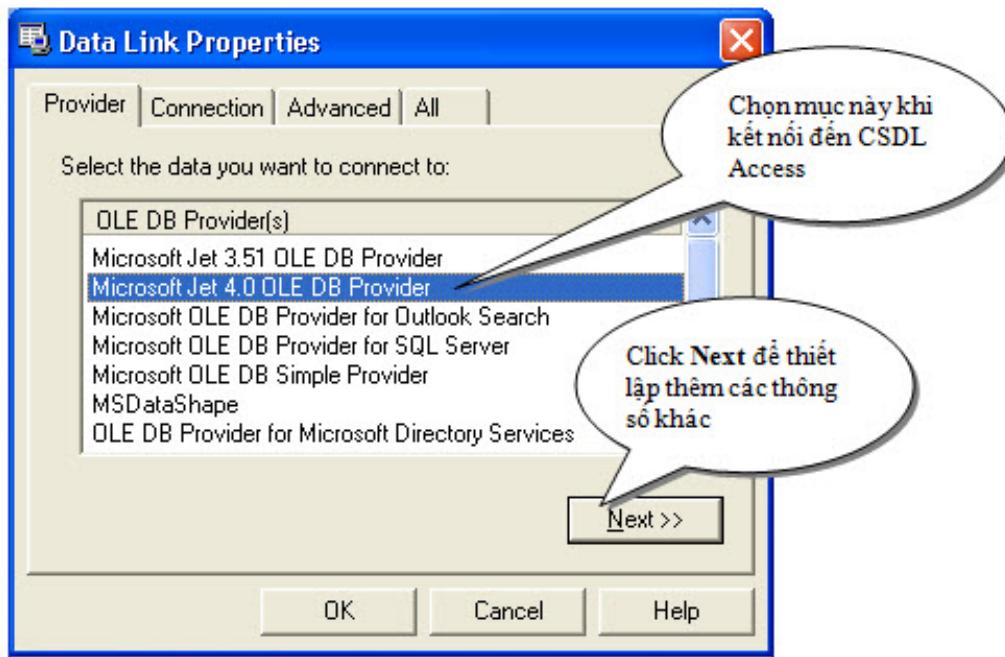
Đặt thông số kết nối trong chế độ thiết kế (Design time)

Click chuột vào điều khiển **adoDiemThi** và chọn thuộc tính Custom trong cửa sổ Properties. Một cửa sổ sau sẽ hiện ra như hình dưới đây:



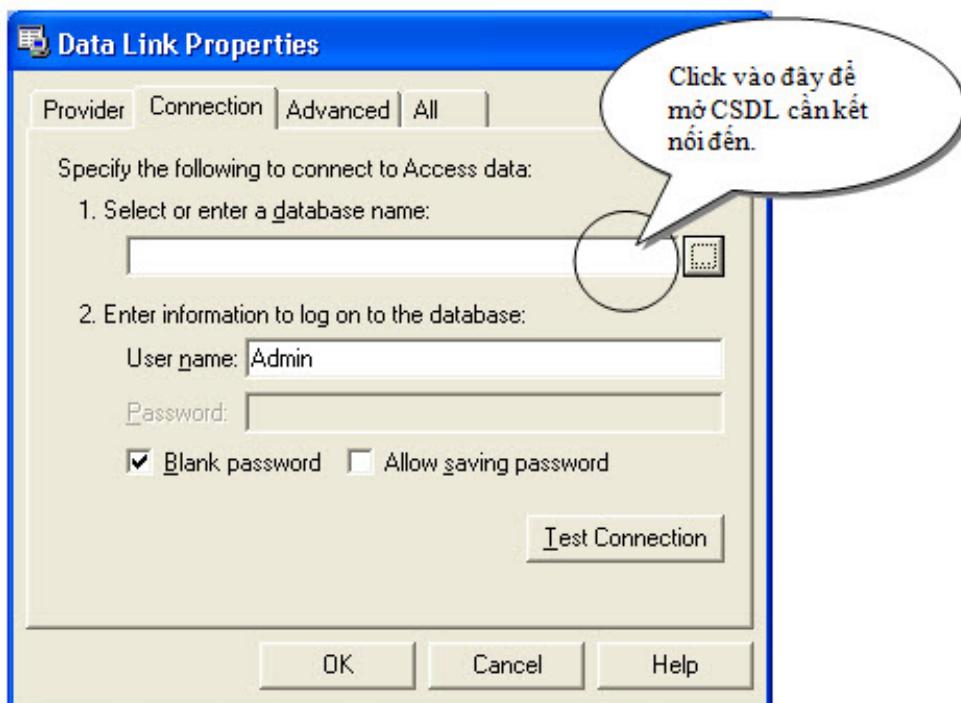
Cửa sổ thiết lập thông số kết nối cho ADO Data Control

- Chọn nút Build...

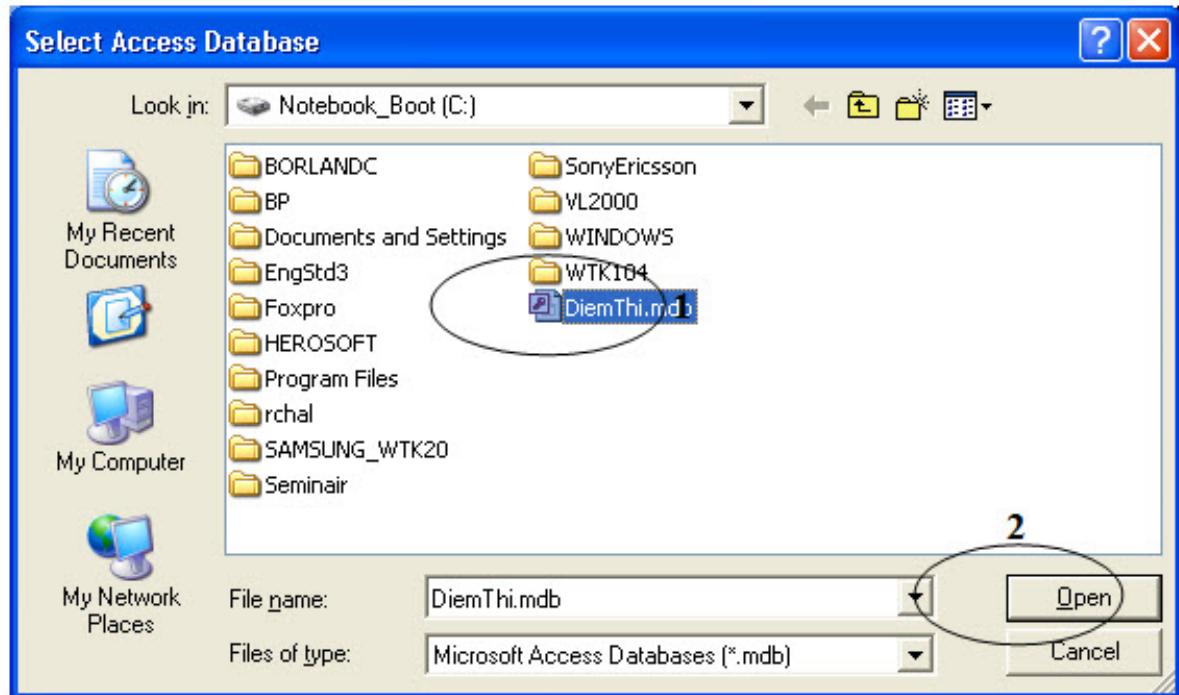


Lựa chọn trình điều khiển truy cập CSDL

- Chọn nút chú thích trong hình



Chọn tệp CSDL Access cần kết nối

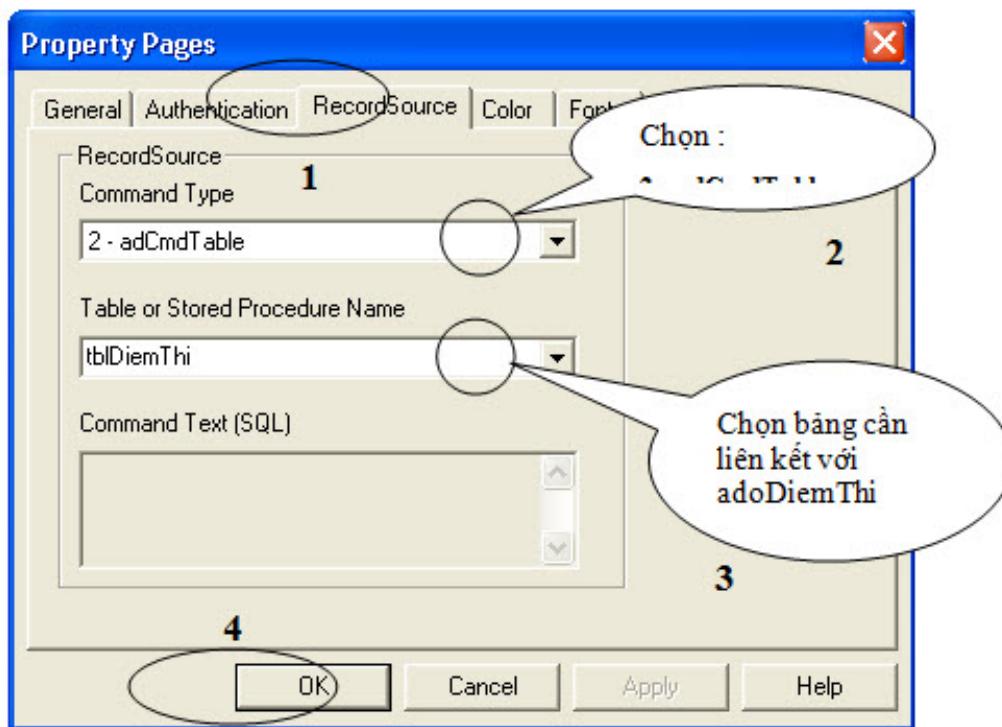


Chọn tệp CSDL cần kết nối đến

- Chọn nút nút **Open** sau đó chọn nút OK để dừng lại ở hộp thoại hình **hình**.

Vì tại một thời điểm, mỗi ADO Data Control chỉ có thể làm việc với một bảng duy nhất, do vậy, bước tiếp theo cần chỉ ra adoDiemThi gắn với bảng dữ liệu nào !?

- Trong hình, Chọn thẻ (Tab) RecordSource và chọn các giá trị như sau:



Liên kết bảng với ADO Data Control

```

Private SubForm_Load() adoDiemThi.ConnectionString = 
"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=C:\DiemThi.mdb" adoDiemThi.RecordSource = "Select * from
tblDiemThi" adoDiemThi.RefreshEnd Sub

```

- Thực hiện các bước theo như trình tự đã đánh số trong hình H.13

Thiết lập thông số kết nối bằng mã lệnh (Trong quá trình Runtime)

Hãy gõ đoạn lệnh sau trong thủ tục sự kiện **Form_Load**

Trong đó:

- + Thuộc tính ConnectionString cho biết trình điều khiển và tệp CSDL là gì ?
- + Thuộc tính RecordSource: Cho biết dữ liệu cần lấy ra là 1 bảng hay 1 truy vấn.

Ở ví dụ trên, dữ liệu cần lấy ra là tập các bản ghi trong bảng *tblDiemThi*. Nếu muốn lấy dữ liệu của toàn bảng mà không cần viết câu lệnh SQL thì phải viết lại như sau:

Private SubForm_Load()

```
adoDiemThi.ConnectionString      =      "Provider=Microsoft.Jet.OLEDB.4.0;Data  
Source=C:\DiemThi.mdb"
```

adoDiemThi.CommandType=adCmdTable ‘ *Dữ liệu cần lấy là một bảng*

adoDiemThi.RecordSource = “tblDiemThi” ‘ *Bảng cần lấy là bảng tblDiemThi*

adoDiemThi.Refresh ‘ *Thực thi*

End Sub

+ Thuộc tính Refresh : Thực hiện việc kết nối và thực thi các truy vấn.

≈ *Ghi chú :*

+ Cách 1 thường được sử dụng khi chúng ta cần thực hiện kết nối để kiểm tra, thử nghiệm. Cách 2 là cách được ưu dùng hơn trong lập trình.

+ Tại một thời điểm chỉ sử dụng một trong 2 cách đã nêu để thiết lập thông số kết nối.

Hiển thị bảng CSDL trong Data Grid

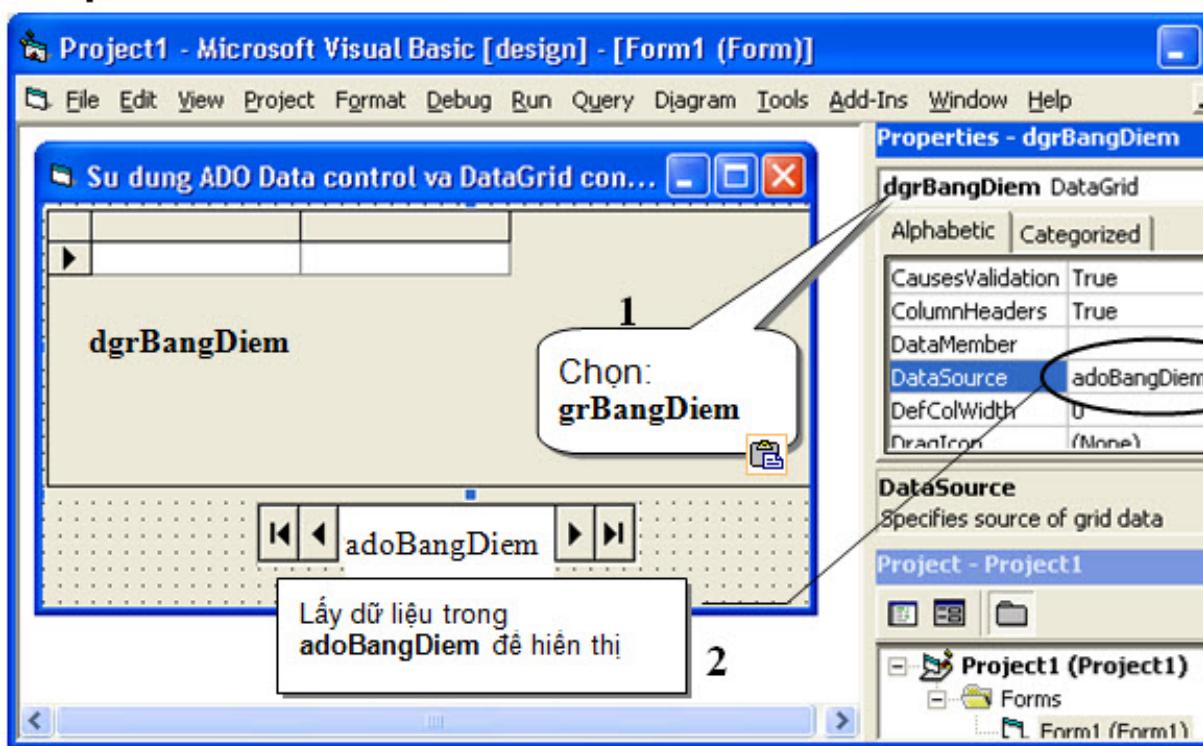
Khi đã thực hiện được mối liên kết giữa ADO Data control với một bảng trong tệp CSDL thì có thể hiển thị dữ liệu trong bảng đó bằng một đối tượng hiển thị khác là điều khiển Data Grid. Các bước tiến hành hiển thị bảng **tblBangDiem** trong Data Grid.

Đưa điều khiển Data Grid Control : Vào menu Project → Components Và chọn mục Microsoft DataGrid Control 6.0 (OLEDB). Sau đó kéo điều khiển vào **Form** và đặt tên là **dgrBangDiem**.

Gắn kết DataGrid với ADO Data Control để hiển thị dữ liệu.

Có 2 cách để gắn kết DataGrid với ADO Data Control:

Thiết lập trong lúc thiết kế (Design time): Cần đặt duy nhất 1 thuộc tính cho DataGrid là DatAsource.



Gắn kết DataGrid với ADO Data Control

Thiết lập trong khi chương trình chạy (Runtime), còn gọi là liên kết động:

Có thể đặt mã lệnh trong bất kỳ đâu trong chương trình, Ví dụ đặt khi chương trình bắt đầu chạy (Trong thủ tục **Form_Load**) như sau (với điều kiện adoBangDiem được kết nối với CSDL theo một trong 2 cách như đã trình bày ở phần trước) :

Private

```
SubForm_Load()adoDiemThi.ConnectionString="Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\DiemThi.mdb" adoDiemThi.RecordSource = "Select * from tblDiemThi" adoDiemThi.RefreshSet dgrBangDiem.DataSource = adoBangDiem.Recordset End Sub
```

- Sau khi thiết lập theo 1 trong 2 cách, ấn F5 để chạy chương trình, ta có kết quả như sau:

SoBD	HoVaTen	DiemToan	DiemLy	DiemHoa
1	Ngô Thành Hoa	6	5	7
2	Nguyễn Văn Tiến	5	7	8
3	Bùi Quang Duy	6	5	7
4	Nguyễn Thị Chính	6	8	10
5	Phạm Văn Tuấn	8	8	8
6	Đinh Thành Hưng	7	7	10

Hiển thị bảng dữ liệu trong Data Grid Control

Thêm một bản ghi vào bảng CSDL

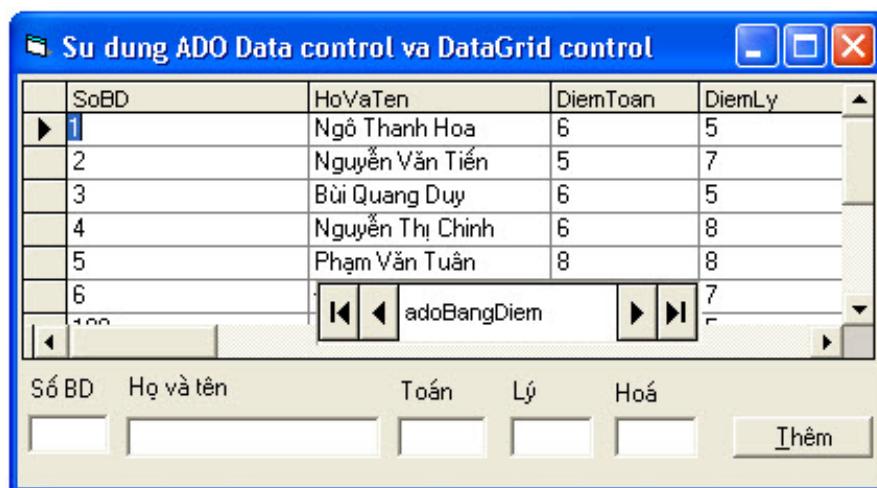
Các bước sau đây sẽ hướng dẫn cách để thêm một bản ghi vào trong bảng CSDL:

Mỗi bản ghi trong bảng có 5 trường (SoBD, HoVaTen, DiemToan, DiemLy, DiemHoa). Do vậy, ý tưởng tiến hành thêm của chúng ta là tạo ra 5 textbox để cho người dùng nhập 5 thông tin trên và có một nút nhấn (Command) để khi người dùng click vào thì sẽ thực hiện thêm một bản ghi có nội dung trong 5 textbox vào bảng tblBangDiem. Ngoài ra có thể thêm các nhãn (Label) để chú thích cho mỗi textbox.

Thêm vào **Form** 5 textbox và 1 command button

Đặt tên và caption cho các phần tử này:

Điều khiển	Đặt thuộc tính Name mới	Thuộc tính Caption mới
Textbox1	txtSoBD	
Textbox2	txtHoVaTen	
Textbox3	txtDiemToan	
Textbox4	txtDiemLy	
Textbox5	txtDiemHoa	
Command1	cmdThem	&Thêm



Giao diện chương trình

- Viết lệnh: Vì việc thêm bản ghi được thực hiện khi người dùng click vào nút cmdThem, do vậy các lệnh thêm bản ghi sẽ được đặt trong thủ tục sự kiện click:

The screenshot shows the Microsoft Visual Basic IDE in design mode. The title bar reads "Project1 - Microsoft Visual Basic [design] - [Form1 (Code)]". The menu bar includes File, Edit, View, Project, Format, Debug, Run, Query, Diagram, Tools, Add-Ins, Window, and Help. A toolbar with icons for Save, Open, and Run is visible. The code editor window has a dropdown menu showing "cmdThem" and a tab labeled "Click". The code itself is written in VBScript:

```
Option Explicit

Private Sub Form_Load()
    adoBangDiem.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\DiemThi.mdb"
    adoBangDiem.CommandType = adCmdTable
    adoBangDiem.RecordSource = "tblDiemThi"
    adoBangDiem.Refresh
    Set dgrBangDiem.DataSource = adoBangDiem.Recordset
End Sub

'/// Thêm bản ghi vào bảng khi người dùng click vào nút cmdThem
Private Sub cmdThem_Click()
    adoBangDiem.Recordset.AddNew

    adoBangDiem.Recordset.Fields("SoBD").Value = txtSoBD.Text
    adoBangDiem.Recordset.Fields("HoVaTen").Value = txtHoVaTen.Text
    adoBangDiem.Recordset.Fields("DiemToan").Value = txtToan.Text
    adoBangDiem.Recordset.Fields("DiemLy").Value = txtLy.Text
    adoBangDiem.Recordset.Fields("DiemHoa").Value = txtHoa.Text

    adoBangDiem.Recordset.Update
End Sub
```

Toàn bộ code chương trình thêm bản ghi vào bảng CSDL

- Nhấn F5 để chạy chương trình.

Giải thích các câu lệnh ở trên:

- Câu lệnh: **adoBangDiem.Recordset.AddNew** → Thêm một bản ghi trống
- Câu lệnh: **adoBangDiem.Recordset.Fields("SoBD").Value = txtSoBD.Text**
→ Điền nội dung chứa trong txtSoBD vào trường SoBD của bản ghi trống vừa tạo.
- **adoBangDiem.Recordset.Update** → Thực sự tiến hành ghi nội dung vào bảng.

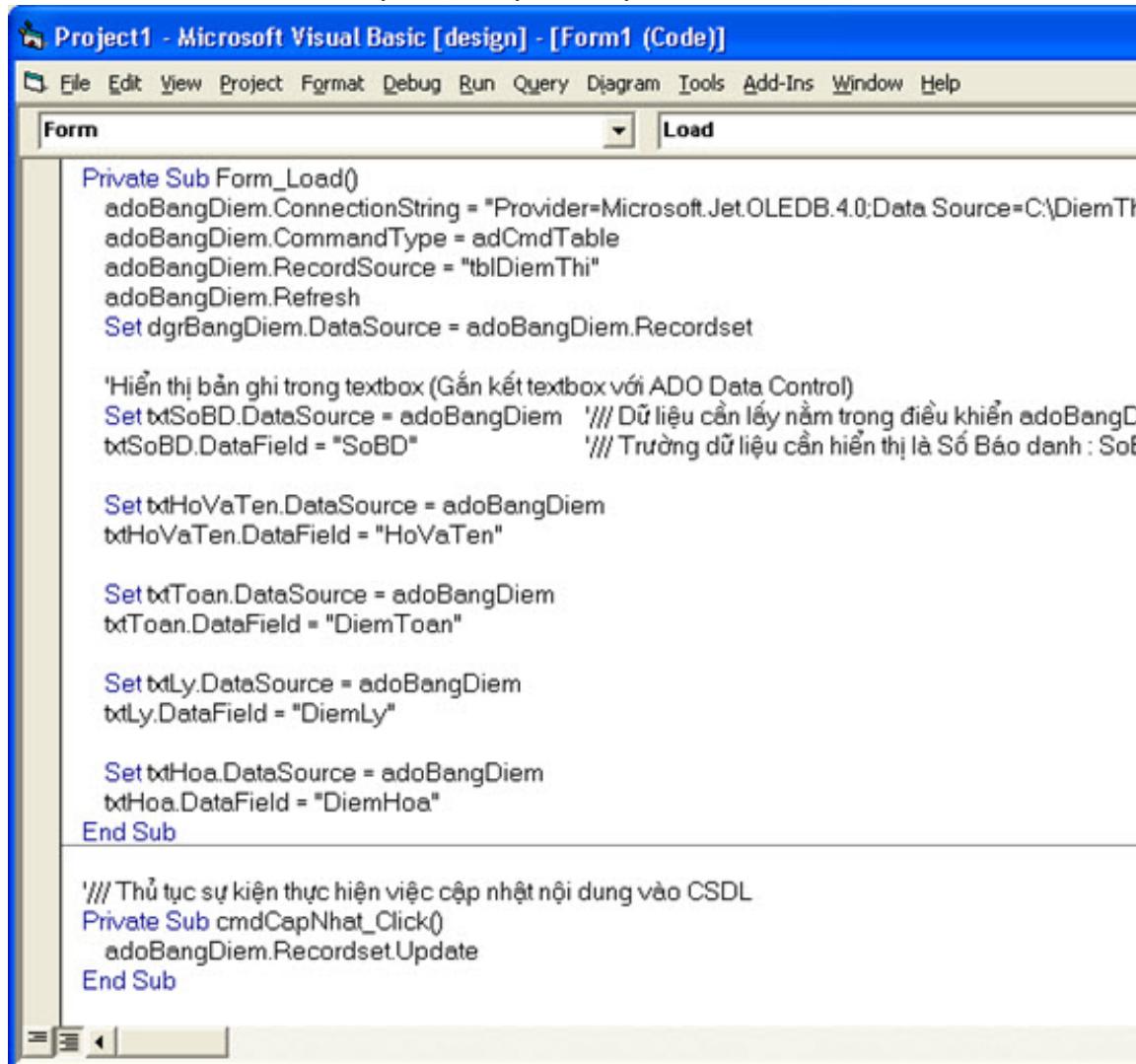
Ghi chú: Nếu chúng ta thực hiện kết nối ADO Data Control với CSDL và gắn kết giữa ADO Data Control với DataGrid trong khi thiết kế (Design time) thì có thể bỏ qua (Xoá) các câu lệnh nằm trong thủ tục sự kiện **Form_Load** ở trên.

Sửa đổi nội dung của bản ghi

Trong một số trường hợp, dữ liệu nhập vào bảng có thể bị sai lệch hoặc không đúng và cần phải sửa đổi lại. Phần này sẽ hướng dẫn cách sửa đổi nội dung nằm trong bảng.

Để sửa đổi dữ liệu trong bảng, có rất nhiều cách, tuy nhiên ở đây sẽ trình bày cách sửa đổi đơn giản nhất thông qua các điều khiển được gọi là: “Data Bound Control”.

- **Thiết kế giao diện** như hình vẽ (Giữ nguyên như phần 4, chỉ thay nút **Thêm** bằng nút **Cập nhật**, Đặt thuộc tính **Name** cho nút cập nhật là **cmdCapNhat**). Để cập nhật (Sửa đổi) người sử dụng sẽ gõ nội dung trong các Textbox tương ứng và click nút **cmdCapNhat**.
- **Viết Code:** Toàn bộ code được viết lại như H.18



```
Project1 - Microsoft Visual Basic [design] - [Form1 (Code)]
File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help
Form Load
Private Sub Form_Load()
    adoBangDiem.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\DiemThi.mdb"
    adoBangDiem.CommandType = adCmdTable
    adoBangDiem.RecordSource = "tblDiemThi"
    adoBangDiem.Refresh
    Set dgrBangDiem.DataSource = adoBangDiem.Recordset

    'Hiển thị bản ghi trong textbox (Gắn kết textbox với ADO Data Control)
    Set txtSoBD.DataSource = adoBangDiem    '/// Dữ liệu cần lấy nằm trong điều khiển adoBangDiem
    txtSoBD.DataField = "SoBD"                '/// Trường dữ liệu cần hiển thị là Số Báo danh : SoBD

    Set txtHoVaTen.DataSource = adoBangDiem
    txtHoVaTen.DataField = "HoVaTen"

    Set txtToan.DataSource = adoBangDiem
    txtToan.DataField = "DiemToan"

    Set txtLy.DataSource = adoBangDiem
    txtLy.DataField = "DiemLy"

    Set txtHoa.DataSource = adoBangDiem
    txtHoa.DataField = "DiemHoa"
End Sub

'/// Thủ tục sự kiện thực hiện việc cập nhật nội dung vào CSDL
Private Sub cmdCapNhat_Click()
    adoBangDiem.Recordset.Update
End Sub
```

Toàn bộ Code chương trình

Click vào nút này để di chuyển giữa các bản ghi.Sửa thông tin ở đây và click vào nút **Cập nhật**



Kết quả khi chạy chương trình

*** Một số lưu ý:

- Các điều khiển (Như textbox ở trên) có khả năng gắn kết với ADO Data Control để hiển thị dữ liệu được gọi là các : “Data Bound Control”.
- DataGridView control ở trên đưa vào chỉ có tác dụng hiển thị dữ liệu hiện thời trong bảng để ta quan sát chứ không nhất thiết cần phải đưa vào.
- Có thể chèn thêm hoặc sửa đổi nội dung trong bảng CSDL trực tiếp trong DataGridView bằng cách chọn Custom và đặt thuộc tính AllowAddNew và AllowUpdate:

Tìm kiếm một bản ghi trong bảng

Thực tế ta thường phải trả lời câu hỏi: “Trong bảng CSDL có bản ghi R hay không?”

Đối tượng con Recorset của đối tượng ADO Data Control có một phương thức tên là Find sẽ giúp ta trả lời được câu hỏi đó.

Thực hành: Cho người dùng nhập vào họ tên của một người trong một textbox, Chương trình sẽ thông báo là “Tìm thấy” cùng với điểm của môn Toán hoặc thông báo là “Không tìm thấy” người này trong bảng CSDL.

Các bước tiến hành:

- Thiết kế giao diện

Như phần 5, nhưng xoá các textbox và command button, Thay bằng một textbox có tên là **txtHoVaTen** và một command button có tên là **cmdTim**, Caption là **&Tìm**



Giao diện tìm kiếm

- **Viết code** (Như hình dưới đây)

```

Project1 - Microsoft Visual Basic [design] - [Form1 (Code)]
File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help
cmdTim Click
Private Sub Form_Load()
    adoBangDiem.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\DiemThi.mdb"
    adoBangDiem.CommandType = adCmdTable
    adoBangDiem.RecordSource = "tblDiemThi"
    adoBangDiem.Refresh
    Set dgrBangDiem.DataSource = adoBangDiem.Recordset
End Sub

'/// Thủ tục sự kiện thực hiện việc tìm kiếm người có họ tên như trong txtHoVaTen
Private Sub cmdTim_Click()
    adoBangDiem.Recordset.MoveFirst

    adoBangDiem.Recordset.Find "HoVaTen = ' " & txtHoVaTen.Text & " '"

    If adoBangDiem.Recordset.EOF = False Then
        MsgBox "Đã tìm thấy ! Điểm toán là: " & adoBangDiem.Recordset.Fields("DiemToan").Value
    Else
        MsgBox "Không có người này trong CSDL !"
    End If
End Sub

```

Cần phải thêm dấu nháy đơn khi so sánh với trường xâu ký tự

Toàn bộ code chương trình
`adoBangDiem.Recordset.Find "HoVaTen = ' " & txtHoVaTen.Text & " '"`

- Nhấn F5 để chạy chương trình.

Giải thích các câu lệnh:

- `adoBangDiem.Recordset.MoveFirst` → Chuyển về bản ghi đầu tiên (Bắt đầu tìm từ bản ghi đầu tiên)
- `adoBangDiem.Recordset.Find "HoVaTen = ' " & txtHoVaTen.Text & " '"` → Thực hiện tìm kiếm trong bảng CSDL có Họ và tên bằng với giá trị nằm trong `txtHoVaTen`. *Ghi chú* là do trường HoVaTen là trường kiểu Text nên giá trị cần so sánh phải thêm vào dấu nháy đơn ở 2 đầu như trên. Còn nếu so sánh với trường số thì không cần.
- `If adoBangDiem.Recordset.EOF = False Then ...` → Sau khi thực thi phương thức Find thì có thể tìm thấy hoặc không. Nếu thấy thì thuộc tính EOF (**E**nd **O**f **F**ile = hết tệp) sẽ có giá trị là False, trái lại sẽ có giá trị là True. Như vậy, để biết là có tìm thấy hay không ta chỉ việc kiểm tra thuộc tính này sau phương thức Find.

Loại bỏ (Xoá) một bản ghi khỏi bảng CSDL

Để xoá một bản ghi khỏi bảng CSDL, có thể sử dụng phương thức Delete của đối tượng con Recordset trong điều khiển ADO Data Control. Khi gọi phương thức này thì bản ghi hiện hành sẽ bị xoá. Về trực quan, bản ghi hiện hành có biểu tượng mũi tên đang chỉ tới.

Có thể xoá một hoặc nhiều bản ghi bằng cách đặt câu lệnh SQL dạng như “Delete...” cho thuộc tính RecordSource của điều khiển ADO Data Control, tuy nhiên để đơn giản, phần hướng dẫn sau đây sẽ trình bày cách xoá một bản ghi thoả mãn một điều kiện nào đó.

Thực hành: Cho người dùng nhập vào số báo danh của một người trong một textbox và chương trình sẽ thực hiện xoá bản ghi có SoBD bằng với Số báo danh này.

Các bước tiến hành:

- Thiết kế giao diện:

SoBD	HoVaTen	DiemToan	DiemLy	DiemHoa
1	Nguyễn Duy Đông	6	5	7
2	Nguyễn Văn Tiến	5	7	8
3	Bùi Quang Duy	6	5	7
4	Nguyễn Thị Chinh	6	8	8
5	Phạm Văn Tuân	8	8	8
6	Đinh Thành Hưng	7	7	7

Nhập vào số báo danh cần xoá Xoá

adoBangDiem

Giao diện chương trình và tên các control

- **Viết code :** Viết code trong thủ tục **Form_Load** và nút cmdXoa

Project1 - Microsoft Visual Basic [design] - [frmDelete (Code)]

File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help

Form Load

```

'///
'/// Trong thủ tục này, cần thực hiện kết nối ADO Data control với bảng tblDiemThi trong CSDL
'/// lệnh Set dgrBangDiem.DataSource = adoBangDiem.Recordset để hiển thị bảng trong DataGridView
'/// Trong trường hợp không cần hiển thị trong DataGridView thì có thể bỏ qua dòng lệnh này!
'///

Private Sub Form_Load()
    adoBangDiem.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\DiemThi.mdb"
    adoBangDiem.CommandType = adCmdTable
    adoBangDiem.RecordSource = "tblDiemThi"
    adoBangDiem.Refresh
    Set dgrBangDiem.DataSource = adoBangDiem.Recordset      '///Đòng này có thể không cần
End Sub

'///
'/// Thủ tục sự kiện thực hiện việc xoá một bản ghi có SoBD bằng số BD trong txtSoBD
'///

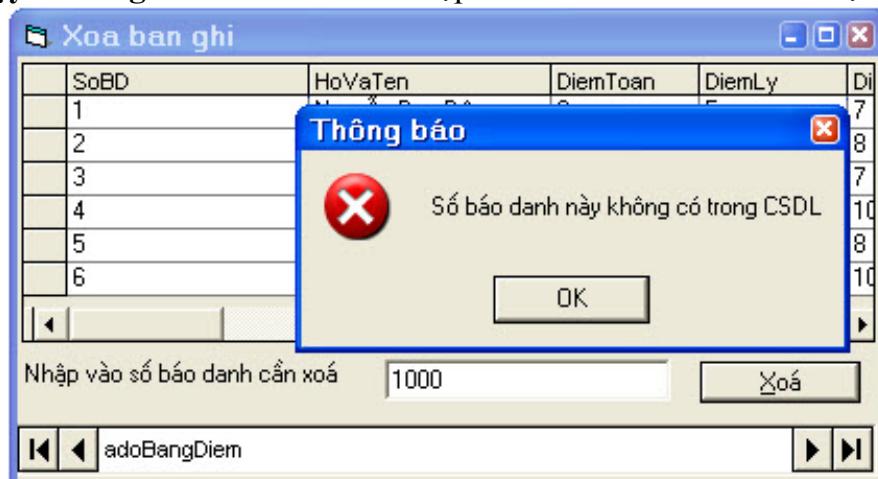
Private Sub cmdXoa_Click()
    '/// Bước 1: Thực hiện tìm đến bản ghi cần xoá (Đưa con trỏ hiện hành về bản ghi cần xoá)
    adoBangDiem.Recordset.MoveFirst
    adoBangDiem.Recordset.Find "SoBD = " & txtSoBD.Text & ""

    '/// Nếu tìm thấy (Khi đó adoBangDiem.Recordset.EOF = False) thì mới xoá
    If adoBangDiem.Recordset.EOF = False Then
        adoBangDiem.Recordset.Delete
    Else
        MsgBox "Số báo danh này không có trong CSDL", vbCritical, "Thông báo"
    End If
End Sub

```

Toàn bộ code chương trình

- Chạy chương trình : Nhấn F5. Nhập số báo danh cần xoá và chọn nút “Xoá”



Xoá bản ghi trong bảng CSDL

Một số Ghi chú:

- Trường SoBD trong bảng CSDL khi thiết kế ta đặt kiểu là Text (Xâu ký tự), do vậy trong phương thức tìm kiếm Find ta cũng phải thêm dấu nháy đơn ở hai đầu.
- Vì phương thức Delete của đối tượng con Recordset sẽ thực hiện xoá ngay bản ghi hiện hành, do vậy để xoá bản ghi cần xoá thì trước hết ta phải di chuyển con trỏ bản ghi về bản ghi cần xoá đó bằng phương thức **find** (Tìm kiếm). Khi dùng phương thức find, nếu có bản ghi phù hợp thì phương thức này sẽ dừng lại và bản ghi đó trở thành bản ghi hiện hành.
- Nói chung, việc xoá một bản ghi trong CSDL thường dựa vào trường khoá, bởi vì tính duy nhất của nó. Còn nếu xoá bản ghi dựa vào trường không phải là khoá (Ví dụ trường DiemToan) thì rất có thể xoá phải bản ghi không đúng như yêu cầu, mặc dù vẫn có thể tiến hành tìm kiếm và xoá bình thường.

Sử dụng các phương thức của đối tượng RecordSet để duyệt các bản ghi

Như đã giới thiệu ở phần 5, các textbox có thể hiển thị một trường dữ liệu của một bản ghi và khi con trỏ bản ghi thay đổi thì dữ liệu trong các textbox cũng tự động được cập nhật theo. Trong phần này sẽ hướng dẫn cách kết hợp với các phương thức **MoveFirst**, **MoveNext**, **MovePrevious**, **MoveLast** của đối tượng RecordSet để xem lần lượt tất cả các bản ghi trong bảng CSDL mà không cần đến đối tượng DataGridView.

Các bước tiến hành:

- **Thiết kế giao diện:** Thêm các điều khiển và bố trí (Layout) như hình H.24



Giao diện trước khi đặt tên và caption

≈ **Mẹo nhỏ:** Có thể đóng thẳng hàng và đặt kích thước các điều khiển cho bằng nhau bằng cách chọn một nhóm các điều khiển và chọn Menu **Format→Align, Make same...**

Đặt các thuộc tính cho các control như sau:

Control	Thuộc tính name mới	Caption/ text	ToolTips
Label1			
Để nguyên, không cần đặt vì ta không có nhu cầu tham chiếu đến.			

Số BD			
Họ và tên			
Toán			
Lý			
Hoá			
Textbox1	txtSoBD		
Textbox2	txtHoVaTen		
Textbox3	txtToan		
Textbox4	txtLy		
Textbox5	txtHoa		
Command1	cmdFirst	<<	Về bản ghi đầu tiên
Command2	cmdPrevious	<<	Về bản ghi trước
Command3	cmdNext	>>	Bản ghi tiếp
Command4	cmdLast	>>	Bản ghi cuối cùng

Sử dụng các phương thức duyệt bản ghi

Số BD	Họ và tên	Toán	Lý	Hoá
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value=" <<"/>	<input type="button" value="<<"/>	<input type="button" value=">>"/>	<input type="button" value=">> "/>	
<input type="button" value=" <"/> <input type="button" value="< "/> adoBangDiem <input type="button" value=" >"/> <input type="button" value="> "/>				

Giao diện sau khi đặt tên (Name) và Caption

- Viết lệnh : Xem trang sau.

Project1 - Microsoft Visual Basic [design] - [frmDelete (Code)]

File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help

cmdPrevious Click

```

'////// Kết nối đến CSDL và bind (Gắn) các Textbox với ADO Data Control để hiển thị dữ liệu -///
Private Sub Form_Load()
    adoBangDiem.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\DiemThi.mdf"
    adoBangDiem.CommandType = adCmdTable
    adoBangDiem.RecordSource = "tblDiemThi"
    adoBangDiem.Refresh

    txtSoBD.DataField = "SoBD"           '/// Trường số BD sẽ hiển thị trong txtSoBD
    Set txtSoBD.DataSource = adoBangDiem '/// Dữ liệu lấy trong adoBangDiem

    txtHoVaTen.DataField = "Hovaten"     '/// Trường họ tên sẽ hiển thị trong txtHovaten
    Set txtHoVaTen.DataSource = adoBangDiem

    txtToan.DataField = "DiemToan"       '/// Trường DiemToan hiển thị trong txtToan
    Set txtToan.DataSource = adoBangDiem

    txtLy.DataField = "DiemLy"
    Set txtLy.DataSource = adoBangDiem

    txtHoa.DataField = "DiemHoa"
    Set txtHoa.DataSource = adoBangDiem
End Sub

'///////// Các thủ tục xử lý sự kiện khi người dùng click vào các nút tương ứng ————— ///////////////
Private Sub cmdFirst_Click()
    adoBangDiem.Recordset.MoveFirst          '/// Chuyển về bản ghi đầu tiên
End Sub

Private Sub cmdLast_Click()
    adoBangDiem.Recordset.MoveLast          '/// Chuyển về bản ghi cuối cùng
End Sub

Private Sub cmdNext_Click()
    adoBangDiem.Recordset.MoveNext          '/// Chuyển tới bản ghi tiếp theo
End Sub

Private Sub cmdPrevious_Click()
    adoBangDiem.Recordset.MovePrevious      '/// Chuyển về bản ghi trước
End Sub

```

Toàn bộ chương trình nguồn

- **Chạy chương trình:** Nhấn F5 và click vào các nút trên màn hình.



Kết quả chạy chương trình

Một số chú ý:

- Các phương thức MoveFirst, MoveLast, MoveNext, MovePrevious chỉ làm việc tốt nếu con trỏ bản ghi không ở vị trí BOF và EOF, do vậy để tránh các lỗi xảy ra thì trước khi di chuyển cần phải kiểm tra, Ví dụ : Trước khi di chuyển về bản ghi tiếp theo (Next) thì ta cần viết câu lệnh sau:

If adoBangDiem.Recordset.EOF= False Then adoBangDiem.Recordset.MoveNext

- Có thể sửa đổi nội dung trong các textbox, sau đó di chuyển sang bản ghi khác thì nội dung vừa sửa đổi sẽ được cập nhật ngay vào trong bảng CSDL.

Tham gia đóng góp

Tài liệu: Thực hành với Visual Basic

Biên tập bởi: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://voer.edu.vn/c/4942cd3c>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Cài đặt phần mềm Visual Basic

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/0fcd210d>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Chạy chương trình Visual Basic 6.0

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/0e9da800>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Thiết lập môi trường làm việc

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/aacf7fc7>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Lưu dự án (Project) ra đĩa

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/402e707e>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Chạy và đóng chương trình Visual Basic (VB)

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/aa6ded9f>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Viết lệnh (Code) cho Form để hiển thị lời chào “Hello World”

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/6023a6dc>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Sử dụng câu lệnh Debug.Print

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/06299e4a>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Sử dụng Câu lệnh InputBox

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/77778963>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Khai báo, gán và hiển thị giá trị của các loại biến cơ bản

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/a221c22c>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Khai báo và sử dụng biến mảng

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/dc4d0fcb>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Định nghĩa và sử dụng kiểu dữ liệu mới - Kiểu bản ghi

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/8d8092bd>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Định nghĩa Hàm (function) trong Visual Basic

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/9127fffb>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Định nghĩa thủ tục trong Visual Basic

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/41c66b77>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Truyền tham trị cho chương trình con

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/bc9865cf>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Truyền tham chiếu cho chương trình con

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/4bfda28e>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Cấu trúc rẽ nhánh If...Then và If ... ElseIf...Then

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/00299be4>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Cấu trúc đa rẽ nhánh Select Case

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/f4cd6edd>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Cấu trúc lặp For

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/df096178>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Cấu trúc lặp Do ... Loop While | Do ... Loop Until

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/0b18fc36>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Sử dụng TextBox, Label kết hợp với Command Button

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/54df848d>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Sử dụng điều khiển Option

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/c2e4dc64>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Sử dụng điều khiển CheckBox

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/a7986343>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Sử dụng điều khiển ListBox

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/ac6bf144>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Sử dụng điều khiển PictureBox

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/db426910>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Sử dụng điều khiển Image

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/9f8c771b>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Sử dụng HscrollBar (Thanh cuộn ngang)

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/7a0eed59>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Sử dụng điều khiển Timer, Drive, Dir và File

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/3a4750b1>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Sử dụng các hộp thoại

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/f9e21e0d>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Tạo menu có nhiều cấp

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/eae2d78c>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Tạo một Menu ngang có nhiều mục

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/fa694038>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Tạo một Menu ngang (Menu bar) đơn giản.

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/08c5d0ff>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Viết lệnh cho các mục của menu

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/82d994c3>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Tạo thanh công cụ Toolbar

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/48020dcb>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Viết lệnh cho các nút trên thanh công cụ

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/77485c95>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Xây dựng chương trình soạn thảo văn bản đơn giản

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/2cdd3dd8>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Xây dựng chương trình nghe nhạc đơn giản

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/26d529f6>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Tạo một bảng CSDL trong Microsoft Access 2000

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/251ee931>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Kết nối đến CSDL sử dụng đối tượng ADO Data Control

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/f04be987>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Hiển thị bảng CSDL trong Data Grid

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/2d84a07c>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Thêm một bản ghi vào bảng CSDL

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/2e45aab4>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Sửa đổi nội dung của bản ghi

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/e134e148>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Tìm kiếm một bản ghi trong bảng

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/247cf8d7>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Loại bỏ (Xoá) một bản ghi khỏi bảng CSDL

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/34a982ad>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Sử dụng các phương thức của đối tượng RecordSet để duyệt các bản ghi

Các tác giả: Khoa CNTT ĐHSP KT Hưng Yên

URL: <http://www.voer.edu.vn/m/ffdb6ad2>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Chương trình Thư viện Học liệu Mở Việt Nam

Chương trình Thư viện Học liệu Mở Việt Nam (Vietnam Open Educational Resources – VOER) được hỗ trợ bởi Quỹ Việt Nam. Mục tiêu của chương trình là xây dựng kho Tài nguyên giáo dục Mở miễn phí của người Việt và cho người Việt, có nội dung phong phú. Các nội dung đều tuân thủ Giấy phép Creative Commons Attribution (CC-by) 4.0 do đó các nội dung đều có thể được sử dụng, tái sử dụng và truy nhập miễn phí trước hết trong môi trường giảng dạy, học tập và nghiên cứu sau đó cho toàn xã hội.

Với sự hỗ trợ của Quỹ Việt Nam, Thư viện Học liệu Mở Việt Nam (VOER) đã trở thành một cổng thông tin chính cho các sinh viên và giảng viên trong và ngoài Việt Nam. Mỗi ngày có hàng chục nghìn lượt truy cập VOER (www.voer.edu.vn) để nghiên cứu, học tập và tải tài liệu giảng dạy về. Với hàng chục nghìn module kiến thức từ hàng nghìn tác giả khác nhau đóng góp, Thư Viện Học liệu Mở Việt Nam là một kho tàng tài liệu khổng lồ, nội dung phong phú phục vụ cho tất cả các nhu cầu học tập, nghiên cứu của độc giả.

Nguồn tài liệu mở phong phú có trên VOER có được là do sự chia sẻ tự nguyện của các tác giả trong và ngoài nước. Quá trình chia sẻ tài liệu trên VOER trở lên dễ dàng như đếm 1, 2, 3 nhờ vào sức mạnh của nền tảng Hanoi Spring.

Hanoi Spring là một nền tảng công nghệ tiên tiến được thiết kế cho phép công chúng dễ dàng chia sẻ tài liệu giảng dạy, học tập cũng như chủ động phát triển chương trình giảng dạy dựa trên khái niệm về học liệu mở (OCW) và tài nguyên giáo dục mở (OER). Khái niệm chia sẻ tri thức có tính cách mạng đã được khởi xướng và phát triển tiên phong bởi Đại học MIT và Đại học Rice Hoa Kỳ trong vòng một thập kỷ qua. Kể từ đó, phong trào Tài nguyên Giáo dục Mở đã phát triển nhanh chóng, được UNESCO hỗ trợ và được chấp nhận như một chương trình chính thức ở nhiều nước trên thế giới.