HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN I

BÀI GIẢNG

TIN HỌC QUẢN LÝ

NGƯỜI VIẾT: Ths. TRỊNH THỊ VÂN ANH

HÀ NỘI - 2013

CHƯƠNG 4: CẤU TRÚC LỆNH

4.1. Lệnh điều kiện

Quyết định kiểu **Conditional Logic** trong nguồn mã gồm 2 loại:

- Loại để tìm hiểu ta đang xử lý phần nào trong giải thuật hoặc đối phó với các trở ngại định trước hay bất ngờ. Thí dụ như: ta muốn mở 10 tập tin để đọc, trước hết ta cần kiểm tra xem tập tin có hiện diện hay không? Nếu không, ta sẽ phải làm gì? Nếu có, ta cần kiểm tra xem khi nào tập tin đó được đọc hết? Sau đó, ta sẽ lập lại các bước kiểm tra đó với tập tin kế tiếp.
- Loại để thi hành các phần khác nhau trong giải thuật dựa trên dữ kiện nào đó. Thí dụ như: ta muốn gởi email đến 10 khách hàng trong danh sách với điều kiện khách hàng đó có máy vi tính, nếu không ta phải điện thoại hay in thư gởi đến khách hàng.

Các cú pháp (syntax) khi dùng lệnh if như sau:

Cú pháp 1:

Cách đơn giản nhất khi làm một quyết định.

```
If (điều kiện) Then
    (mã thi hành nếu điều kiện thỏa mãn, nghĩa là = True)
End If
```

Thí dụ:

```
'Tuyên bố biến số và giá trị cho intYourAge
  Dim intYourAge
  intYourAge = 70
  'Quyết định và công bố kết quả
  If intYourAge => 65 Then
      MessageBox.Show ("You should retire")
  End If
```

Cú pháp 2:

Quyết định với 2 tình trạng trái ngược nhau.

```
If (điều kiện) Then
   (mã thi hành nếu điều kiện thỏa mãn, nghĩa là = True)
Else
    (mã thi hành nếu điều kiện không được thỏa mãn, nghĩa
là = False)
End If
```

Thí dụ:

```
    'Tuyên bố biến số và giá trị cho intYourAge
    Dim intYourAge
    intYourAge = 40
    'Quyết định và công bố kết quả
    If intYourAge => 65 Then
        MessageBox.Show ("You should retire.")
    Else
       MessageBox.Show ("You are too young to retire.")
```

```
    End If
```

Cú pháp 3:

Quyết định với nhiều tình trạng thay đổi hay khác nhau.

```
If (điều kiện 1) Then
     (mã thi hành nếu điều kiện 1 thỏa mãn, nghĩa là điều
kiện 1 = True)
ElseIf (điều kiện 2) Then
    (mã thi hành nếu điều kiện 2 được thỏa mãn, nghĩa là
điều kiện 2 = True)
Else
    (mã thi hành khi không điều kiện nào thỏa mãn, nghĩa
là điều kiện 1 và điều kiện 2 đều = False)
End If
```

Thí dụ:

```
'Tuyên bố biến số và giá trị cho intYourAge
Dim intYourAge
intYourAge = 50
'Quyết định và công bố kết quả
If intYourAge => 65 Then
  MessageBox.Show ("You should retire.")
  ElseIf intYourAge => 45 Then
  MessageBox.Show ("You should pay more to your current
scheme.")
 Else
    MessageBox.Show ("You are too young to worry about.")
 End If
```

Cú pháp 4:

Quyết định trong quyết định (**Nested If**).

```
If (điều kiện 1) Then
     (mã thi hành nếu điều kiện 1 thỏa mãn, nghĩa là điều
kiện 1 = True)
     If (điều kiện 2) Then
     (mã thi hành nếu điều kiện 1 và 2 được thỏa mãn,
nghĩa là điều kiện 1 = true và điều kiện 2 = True)
     End If
End If
```

Thí dụ:

```
    'Tuyên bố biến số và giá trị cho intYourAge
    Dim intYourAge
    intYourAge = 65
    'Quyết định và công bố kết quả
    If intYourAge > 55 Then
        MessageBox.Show ("You should retire.")
```

```
            If intYourAge = 65 Then
                MessageBox.Show ("You must retire.")
            End If
        End If
```

Chú thích:

Nếu ta trên 55 tuổi, MessageBox sẽ hiển thị hàng chữ 'You should retire', nhưng nếu ta đúng 65, MessageBox sẽ hiển thị hàng chữ 'You must retire'. Sở dỉ có điều kiện trong điều kiện trên 55 tuổi như vậy là vì tuổi ta có thể là 60 hay 70 miễn sao trên 55 là được, do đó ta kèm thêm một điều kiện nữa để có thể hiển thị hàng chữ 'You must retire' khi nào số tuổi vừa đúng 65.

Cú pháp 5:

Dùng If chỉ trong một hàng mã có thể gọn gàng hơn nhưng không cung cấp cấu trúc giải thuật rõ ràng và thường khó đọc hơn.

```
If (điều kiện) Then (mã thi hành nếu điều kiện = True)
Else (mã thi hành nếu điều kiện = False)
```

hoặc

```
If (điều kiện) Then
  (nếu điều kiện =True, thi hành mã 1: mã 2 : mã 3: ...)
```

Lưu ý: kiểu cú pháp 5 không cần phải có **End If** ở cuối hàng.

Thí dụ:

```
       Dim count As Integer
       If count = 8 Then MessageBox.Show ("Count = 8")
          Else MessageBox.Show ("Count is not 8")
       If count = 8 Then
         MessageBox.Show ("Count = 8") : count = count + 1 :
MessageBox.Show ("Count is now " & count)
```

Chú thích:

Ở hàng If đầu tiên, ta tuyên bố biến số (variables) count , so sánh với giá trị 8 và hiển thị kết quả.

Hàng If thứ nhì, ta hiển thị kết quả, sau đó cộng thêm 1 vào cùng 1 biến số (variables) count, như vậy count (cuối cùng) = count (với giá trị là 8) + 1 sẽ bằng 9 và hiển thị kết quả **Count is now 9**.

Cách viết mã từ trên xuống dưới, ta thấy mã dễ đọc, dễ hiểu hơn:

```
       Dim count As Integer
      If count = 8 Then
         MessageBox.Show ("Count = 8")
         count = count + 1
         MessageBox.Show ("Count is now " & count)
      End If
```

4.2. Toán tử so sánh

Khi dùng If để kiểm tra các diều kiện, không những ta chỉ so sánh **bằng (=)** không thôi, mà còn so sánh nhiều kiểu khác nhau nữa. Gọi chung là các dấu so sánh (**Comparision Operators**) gồm có:

- < > : là dấu không bằng (Not Equal To)
- < : dấu nhỏ hơn
- > : dấu lớn hơn
- < = : dấu nhỏ hơn hoặc bằng
- > = : dấu lớn hơn hoặc bằng
- AND : dùng kiểm tra hơn 1 điều kiện
- OR : dùng kiểm tra điều kiện này hoặc điều kiện nọ

Ví dụ:

```
        If yourSurName < > "Vu" Then
           MessageBox.Show("You are not my relative")
        End If
        If yourAge < 18 Then
           MessageBox.Show("You can not drive a car")
        End If
        If yourSurName > 21 Then
           MessageBox.Show("You can marry")
        End If
        If yourAge <= 64 Then
           MessageBox.Show("You can not retire")
        End If
        If yourAge >= 65 Then
           MessageBox.Show("You must retire")
        End If
        If yourAge >= 18 AND yourHeigth >= 1.60 Then
           MessageBox.Show("You can be a movie star")
        End If
       If yourPreferredDrink="Rượu" OR yourPreferredDrink="Bia"
Then
           MessageBox.Show("You are ... dân nhậu")
        End If
```

Chú thích:

Nhắc thêm ở đây về cách dùng **AND** và **OR** trong điều kiện. Ta biết khi điều kiện được kiểm tra sẽ cho biết giá trị là True (hay 1, hay 'ON', hay là Tắt hoặc Đóng) hoặc False (hay 0, hay 'OFF', hay là Mở hoặc Khoá). Để dễ nhớ, ta thiết lập bảng sau:

Bảng 4.1: Toán tử AND

Điều kiện A	Điều kiện B	Kết quả	Chú thích
0	0	0	Nếu A False và B False, kết quả sau cùng là False
0	1	0	Nếu A False và B True, kết quả sau cùng là False
1	0	0	Nếu A True và B False, kết quả sau cùng là False
1	**1**	**1**	Nếu A True và B True, kết quả sau cùng là True. Ta

			thấy chỉ có một trường hợp bằng True với dấu AND khi nào cả 2 đều True.

Bảng 4.2: Toán tử OR

Điều kiện A	Điều kiện B	Kết quả	Chú thích
0	**0**	**0**	Nếu A Fasle và B False, kết quả sau cùng là False. Ta thấy chỉ có một trường hợp bằng False với dấu OR khi nào cả 2 đều False.
0	1	1	Nếu A False và B True, kết quả sau cùng là True
1	0	1	Nếu A True và B False, kết quả sau cùng là True
1	1	1	Nếu A True và B True, kết quả sau cùng là True.

4.3. So sánh xâu

Khi so sánh chữ hay câu, ta thường gặp trở ngại khi không lưu ý đến các **chữ thường hay chữ Hoa** (**case sensitive**). Nhớ là đối với máy vi tính, khi so sánh như vậy, chữ **a** thường khác với chữ **A** Hoa vì chúng có giá trị khác nhau.

Thí dụ 1:

```
Dim mySociety As String
mySociety = "VOVISOFT"
If mySociety = "Vovisoft" Then
   MessageBox.Show("You are a Vovisoft's member")
Else
   MessageBox.Show("You are not a Vovisoft's member")
End IF
```

Thí dụ 2: dùng **Compare** method của **String** object so sánh 2 chữ hay câu như sau:

```
Dim mySociety As String
mySociety = "VOVISOFT"
If String.Compare (mySociety, "Vovisoft", True) = 0 Then
   MessageBox.Show("You are a Vovisoft's member")
Else
   MessageBox.Show("You are not a Vovisoft's member")
End IF
```

Chú thích:

String.Compare dùng để so sánh 2 giá trị của String và trả về 1 số nguyên (Integer) sau khi so sánh. Nếu method trả về số 0, nghĩa là 2 chữ hay câu giống nhau về giá trị, ngoài ra sẽ trả về số khác số 0.

4.4. Lệnh lựa chọn

Đây là loại thứ hai trong **Conditional Logic** để thi hành các phần khác nhau trong giải thuật dựa trên những điều kiện khác nhau nào đó. Thí dụ như:

- Nếu là khách hàng A, gởi email đến địa chỉ khách hàng A.
- Nếu là khách hàng B, gởi email đến địa chỉ khách hàng B.

- Nếu là khách hàng C, gởi email đến địa chỉ khách hàng C.
- Nếu là khách hàng D, gởi email đến địa chỉ khách hàng D.
- Nếu là khách hàng E, gởi email đến địa chỉ khách hàng E.

Ta có thể dùng If ... Then ... ElseIf ... End If như sau:

```
     If khách hàng = "A" Then
    gởi email đến địa chỉ khách hàng A
 ElseIf khách hàng = "B" Then
    gởi email đến địa chỉ khách hàng B
 ElseIf khách hàng = "C" Then
    gởi email đến địa chỉ khách hàng C
 ElseIf khách hàng = "D" Then
    gởi email đến địa chỉ khách hàng D
 ElseIf khách hàng = "E" Then
    gởi email đến địa chỉ khách hàng E
 End If
```

Tuy nhiên, nếu ta muốn đổi **khách hàng** thành **công ty** chẳng hạn, ta phải thay đổi chữ **khách hàng** ở từng câu If một, như vậy quả là phiền phức và không đạt năng suất cao như cách dùng cú pháp **Select Case** :

Cú pháp 1 (Syntax 1):

```
Select Case công ty
Case "A"
    gởi email đến địa chỉ công ty A
Case "B"
    gởi email đến địa chỉ công ty B
Case "C"
    gởi email đến địa chỉ công ty C
Case "D"
    gởi email đến địa chỉ công ty D
Case "E"
    gởi email đến địa chỉ công ty E
End Select
```

Lưu ý:

Khi dùng Select Case, có phân biệt chữ thường và chữ Hoa, ví dụ: công ty **A** khác với công ty **a**.

Cú pháp 2:

Dùng Select Case để chọn trường hợp gồm nhiều điều kiện có một giải đáp chung:

```
Select Case strMyContactName
  Case "A", "B", "E"
   MessageBox.Show ("Chào các bạn học!!", "Greeting")
  Case "C", "D"
    MessageBox.Show("Hay tham gia nhe!", "Greeting")
End Select
```

Trong đó, ta thấy trường hợp A, B và E có chung một giải đáp nhưng khác với trường hợp C và D.

Cú pháp 3:

Dùng Select Case cho các trường hợp ngoại lệ **Case Else**:

```
Select Case strMyContactName
    Case "A", "B", "E", "C", D"
        MessageBox.Show ("Chào các bạn học!", "Greeting")
    Case Else
        MessageBox.Show ("Hay tham gia!", "Greeting")
End Select
```

4.5. Vòng lặp

Looping Logic dùng trong trường hợp cần lập đi lập lại nhiều lần (hay đúng hơn nữa, một số lần nhất định) việc thi hành một công tác nào đó, ví dụ: cộng thêm 10 sản phẩm vào bảng liệt kê sản phẩm của công ty, hiển thị (display) 5 CD nhạc tuyệt phẩm hàng đầu trong năm.

2 loại cơ bản của Looping Logic - **For** loop và **Do** loops bao gồm:

- `For ... Next`
- `For Each ... In ... Next`
- `Do Until ... Loop`
- `Do While ... Loop`
- Các trường hợp đặc biệt

Cú pháp 1:

```
For số lần đếm từ số ... đến số ...
    (thi hành công việc nào đó)
Next
```

Thí dụ 1:

```
'Tuyên bố biến số dùng làm counter
Dim intCounter
For intCounter = 1 To 10
 MessageBox.Show ("Vovisoft", "Greeting")
Next
```

Thí dụ 2:

```
'Tuyên bố biến số dùng làm counter
Dim intCounter
For intCounter = 10 To 100 Step 10
 MessageBox.Show ("Vovisoft", "Greeting")
Next
```

Chú thích:

Thí dụ 1, tạo biến số (variables) intCounter để đếm từ 1 đến 10, mỗi lần đếm như vậy trong For ... Next loop, ta hiển thị 1 cửa sổ với hàng chữ Vovisoft.

Thí dụ 2, mỗi lần đếm ta nhảy 10 bước (hay cộng thêm 10 vào số lần đếm) bắt đầu với inCounter = 10 là lần đầu tiên, kế là 20, 30, ... đến 100.

Cú pháp 2:

```
For Each ... In ...
    (thi hành công việc nào đó)
Next
```

Thí dụ: Liệt kê tất cả các thư mục phụ (subfolders) trong dĩa C (root directory trong drive C)

```
'Biến array dùng lưu trữ các ngăn chứa phụ (subfolders)
Dim subFolders( ) As DirectoryInfo
subFolders = New DirectoryInfo("C:\").GetDirectories
'Loop liệt kê các ngăn chứa phụ (subfolders) trong dĩa C
Dim subFolder As DirectoryInfo
For Each subFolder In subFolders
     lstData.Items.Add (subFolder.FullName)
Next
```

Chú thích:

Khai báo và tạo biến số (variables) loại Array trực thuộc đối tượng **DirectoryInfo**. Dùng method **GetDirectories** của object DirectoryInfo để lấy và lưu trữ các thư mục phụ trong dĩa C.

Sau đó, dùng **For Each ... Next** loop kiểm tra từng phần một trong array subFolder**s** và cộng tên của phần vào bảng liệt kê tên **lstData**.

Cú pháp 3:

```
Do Until (điều kiện)
     (thi hành công việc nào đó)
Loop
```

Thí dụ: Liệt kê từng số ngẫu nhiên và chấm dứt loop khi nào số đó là số 10

```
'Bố trí object tạo số ngẫu nhiên
Dim random As New Random( )
'Khai baos 1 biến số chứa số ngẫu nhiên mặc định là 0
Dim intRandomNumber As Integer = 0
'Loop cho đến khi nào số intRandomNumber = 10
Do Until intRandomNumber = 10
    'Tạo 1 số ngẫu nhiên
    intRandomNumber = random.Next (25)
    'cộng vào bảng liệt kê tên lstData
    lstData.Items.Add (intRandomNumber)
Loop
```

Chú thích:

Ta dùng random là 1 thể hiện (instance) của object Random để tạo số ngẫu nhiên (random number generator) trong Do Until ... Loop và lưu trữ giá trị đó vào biến số (variables) intRandomNumber. Khi nào giá trị số này bằng 10, ta chấm dứt việc cộng số vào bảng liệt kê tên lstData.

Cú pháp 4:

Ngược lại với Do Until ... Loop là Do While ... Loop. **Do While ... Loop chỉ thi hành khi nào điều kiện bằng True, ngược lại với Do Until ... Loop sẽ chấm dứt khi nào điều kiện bằng True.**

```
Do While (điều kiện)
    (thi hành công việc nào đó)
Loop
```

Thí dụ: Liệt kê từng số ngẫu nhiên và chấm dứt loop khi nào số đó = 10 hay lớn hơn 10

```
'Bố trí object tạo số ngẫu nhiên
Dim random As New Random( )
'Khai bao 1 biến số chứa số ngẫu nhiên mặc định là 0
Dim intRandomNumber As Integer = 0
'Loop khi số intRandomNumber < 10
Do While intRandomNumber < 10
    'Tạo 1 số ngẫu nhiên
    intRandomNumber = random.Next (25)
    'cộng vào bảng liệt kê tên lstData
    lstData.Items.Add (intRandomNumber)
Loop
```

Chú thích:

Ta dùng random là 1 thể hiện của object Random để tạo số ngẫu nhiên (random number generator) trong Do While ... Loop và lưu trữ giá trị đó vào biến số (variables) intRandomNumber. Khi nào giá trị số này nhỏ hơn 10, ta cộng số đó vào bảng liệt kê tên lstData, nếu không, ta chấm dứt loop.

Cú pháp 5:

Đây là phiên bản khác của Do Until và Do While:

```
Do
    (thi hành công việc nào đó)
Loop While (điều kiện)
Do
    (thi hành công việc nào đó)
Loop Until (điều kiện)
```

Lưu ý:

Phiên bản này khác phiên bản trước ở chỗ:

- **Thi hành công việc trước**
- Sau đó mới kiểm tra điều kiện để tiếp tục hay chấm dứt loop, như vậy tối thiểu, công việc được thi hành 1 lần.

Các trường hợp đặc biệt:

1. Nested Loops:

Nhiều trường hợp cần đến 2 hay nhiều loop trong algorithm, tỷ như:

```
'Tuyên bố và bố trí hàng và cột
Dim intRow, intColume As Integer
'Loop hàng từ hàng thứ nhất đến hàng 10
For intRow = 1 To 10
```

```
        'Mỗi hàng, loop từ cột thứ nhất đến cột 5
        For intColume = 1 To 5
         'hiển thị hàng và cột bảng liệt kê tên lstData
         lstData.Items.Add ("Hàng" & intRow & " và cột " &
           intColume)
        Next
    Next
```

Chú thích:

Ta dùng 2 lần For ... Next (nested loop) để hiển thị hàng trước cột sau trong bảng liệt kê tên lstData theo thứ tự sau:

Hàng	**1**	và	cột	1
Hàng	1	và	cột	2
Hàng	1	và	cột	3
Hàng	1	và	cột	4
Hàng	1	và	cột	5
Hàng	**2**	và	cột	1
Hàng	2	và	cột	2
Hàng	2	và	cột	3
Hàng	2	và	cột	4
Hàng	2	và	cột	5
...				
...				
Hàng	**10**	và	cột	1
Hàng	10	và	cột	2
Hàng	10	và	cột	3
Hàng	10	và	cột	4
Hàng	10	và	cột	5

Dùng **Exit For** hay **Exit Do** để chấm dứt loop (quit) vô điều kiện.

Chú ý khi tạo và loop, ta có thể gặp vòng lặp không thoát, vú dụ như:

```
    'Tuyên bố biến số
     Dim counter As Integer = 0
     'Loop bế tắc không lối thoát
     Do
         counter += 1
     Loop Until counter = 0
```

Chú thích:

Mặc dù trước khi vào loop, counter = 0 nhưng với **counter += 1** được thi hành trước khi kiểm tra điều kiện counter = 0, ở đây có nghĩa là counter (hiện tại) bằng counter (trước đó là 0) cộng thêm 1 như vậy giá trị của counter bây giờ bằng 1 (vì 0 + 1 = 1).

Đến khi kiểm tra điều kiện counter = 0 ở câu Loop Until counter = 0, điều kiện này sẽ là False, do đó loop bắt đầu lập lại với counter = 2, 3, 4,… không thoát.

Cách giải quyết khi Infinite Loop:

- Trường hợp chạy ứng dụng (application) trong MS Visual Studio.NET, chọn **Debug | Stop Debugging** để chấm dứt.
- Trường hợp chạy ứng dụng (application) bên ngoài MS Visual Studio.NET, nhấp các nút **Ctrl + Alt + Delete** và chọn **Task Manager**, sau đó chọn ứng dụng (application) có hàng chữ kèm '**Not Responding**' trong phần mục **Status** và nhấp nút **End** để chấm dứt.

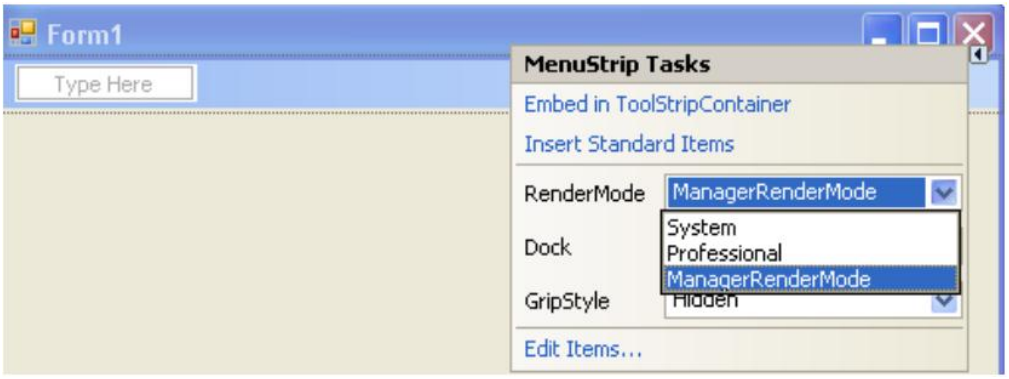
Cuối cùng, kiểm tra và điều chỉnh lại điều kiện để chấm dứt loop.

CHƯƠNG 5: THIẾT KẾ BIỂU MẪU DÙNG CÁC ĐIỀU KHIỂN

5.1. Cửa sổ form

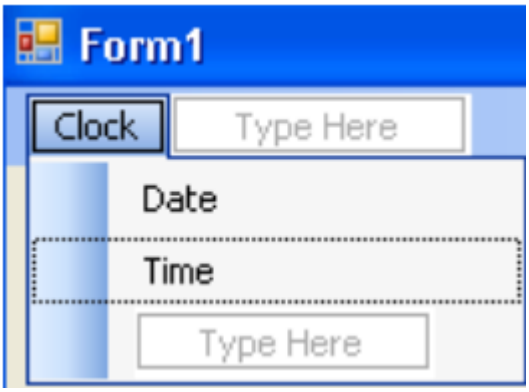
Tạo mới một giải pháp mang tên MyMenu và thêm vào đó một dự án mới cùng tên như đã biết trong các ví dụ trước. Tại giao diện thiết kế, ta đưa điều khiển MenuStrip vào trong Form bằng cách tích đúp chuột hay kéo thả như đã biết.

Chúng ta không cần quan tâm đến vị trí của menu trên form vì Visual Studio sẽ tự động đặt nó sao cho phù hợp. Chúng ta có thể thay đổi các thuộc tính sao cho phù hợp bằng cách chọn mở Smart Tags (nút mũi tên tam giác màu đen bên góc phải điều khiển Menu).



Hình 5.1. Tạo menu

Khi được đặt vào form thì điều khiển menu sẽ được đặt tại một vùng như trên hình gọi là khay công cụ - Component tray và VS sẽ hiển thị trực quan menu trên đầu cửa sổ Form. Chuỗi Type Here là nơi bạn có thể chọn và nhập vào các mục chọn cho menu. Chúng ta sẽ tạo ra menu ngay sau đây. Nhắp chuột vào chuỗi Type Here và gõ vào chuỗi "Clock" và ấn enter. Nhắp chuột vào chuỗi Type Here con ở dưới rồi gõ Date, Time như hình 5.2.



Hình 5.2. Menu con

Để đóng phần thiết kế menu, ta chọn một vùng nào đó trên form, để hiển thị ta lại chọn vào menu Clock như trên. Bây giờ chúng ta sẽ tạo một số tùy biến cho Menu. Trong một số phần mềm hay ngay trình duyệt Windows Explorer của hệ điều hành chúng ta có thể ấn tổ hợp Alt + phím tắt để mở nhanh một thực đơn nào đó. Các phím tắt ấy được gọi là phím truy cập (Access Key). Phím này có dấu gạch chân ở dưới. Trong VS, để tạo phím này ở ta chỉ việc gõ thêm dấu „&" trước ký tự nào muốn hiển thị gạch chân trong phần Type Here. Tạo ra các phím tắt cho các mục chọn của menu Clock như hình 5.3.

Hình 5.3. Phím tắt

Việc thay đổi thứ tự các mục chọn bằng cách mở chế độ thiết kế menu rồi nhắp chọn mục chọn nào đó, kéo nó đến vị trí mong muốn. ví dụ ta kéo mục chọn Time lên thay cho vị trí mục chọn Date. Bây giờ chúng ta tạo ra sự kiện kích chuột cho các mục chọn của menu. Khi bạn tích chuột vào Date hay Time thì một nhãn Label sẽ xuất hiện và hiển thị thông tin ngày hay giờ tương ứng. Để làm được điều này, trước hết ta tạo ra một Label vào trong form.Tạo thuộc tính cho đối tượng Label1: BorderStyle – FixedSingle; Font – Bold 14; Text – rỗng; TextAlign – MiddleCenter. Cài đặt thủ tục sự kiện cho mục chọn menu. Bây giờ chúng ta sẽ tạo sự kiện tích chuột cho các mục con trong menu Clock.

Nhắp vào menu Clock trên form1 để hiển thị menu con. Nhắp đôi chuột vào mục chọn Time để mở cửa sổ Code Editor và tạo ra một thủ tục có tên TimeToolStripMenuItem_Click. Trong VS.NET 2005 thì khi bạn gõ tên mục chọn là gì thì mặc định khi tích đúp chuột để viết mã thì VS sẽ tạo ra một thủ tục có phần đầu tên trùng với tên mục chọn (phần tên chưa có dấu cách trống phân cách tên mục chọn) menu (ở trên là TimeToolStripMenuItem_Click). Tất nhiên đây là mặc định, ta có thể thay đổi tên nhờ thuộc tính Name ở cửa sổ Properties.

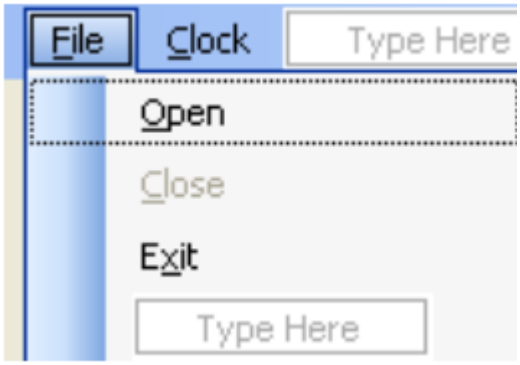
Nhập dòng mã sau:

```
Label1.Text = TimeString
Tương tự với thủ tục DateToolStripMenuItem_Click của mục chọn Date
Label1.Text = DateString
```

Thêm mục File vào menu chương trình

Tạo thêm một mục con Color vào trong menu Clock. Mục này sẽ kích hoạt hộp thoại ColorDialog1 chọn màu cho Label1.

Tạo một Menu File bên cạnh menu Clock như hình 5.4. Đồng thời tạo thêm các mục con Open, Close, Exit trong menu này.



Hình 5.4. Menu file

Tiếp theo ta thay đổi tên bằng thuộc tính Name trong cửa sổ Properties cho các mục chọn: mục Open thành mnuOpenItem, Close thành mnuCloseItem, Exit thành mnuExitItem. Ta cũng đặt thuộc tính Enable của mục Close (giờ là mnuCloseItem) thành False. Thuộc tính này vô hiệu hóa hay làm mờ mục Close như hình. Nó chỉ được sáng lên để người dùng chọn khi mã thực thi chương trình cho phép.

Viết mã chương trình

Ttạo thủ tục mnuOpenItem_Click bằng cách nhấp đúp chuột vào mục Open trên menu File và nhập đoạn mã sau:

```
OpenFileDialog1.Filter = "Bitmaps (*.bmp) | *.bmp"
If OpenFileDialog1.ShowDialog()=Windows.Forms.DialogResult.OK
    Then
       PictureBox1.Image = System.Drawing.Image.FromFile _
       (OpenFileDialog1.FileName)
       mnuCloseItem.Enabled = True
End If
```

Chú thích mã:

- Đoạn mã thứ nhất giúp lọc ra loại file để mở là file ảnh dạng Bitmap (*.bmp). Bạn có thể mở nhiều loại file bằng câu lệnh:

```
OpenFileDialog1.Filter = _ "Bitmaps(*.bmp)|*.bmp|JPEG(*.jpg)|
    *.jpg|All Files(*.*)|*.*"
```

- Phương thức ShowDialog() là phương thức mới trong VS.NET, nó có thể dùng được với mọi hộp thoại và cửa sổ Windows Forms. Phương thức này trả về kết quả mang tên DialogResult cho biết người dùng đã chọn vào hộp thoại. Và nếu nút OK được chọn thì kết quả trả về sẽ bằng với DialogResult.OK.

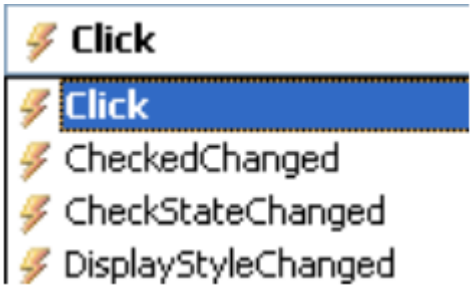
- Khi nút Open được nhấn, nếu hợp lệ thì thuộc tính FileName của OpenFileDialog sẽ mang đầy đủ đường dẫn và tên file của file đã mở vì thế mà dòng mã thứ 3 sẽ nạp chính xác ảnh vào PictureBox1.

Tương tự ta cũng nhấn đúp chuột vào mục Close để tạo thủ tục chọn cho nó và nhập đoạn mã sau:

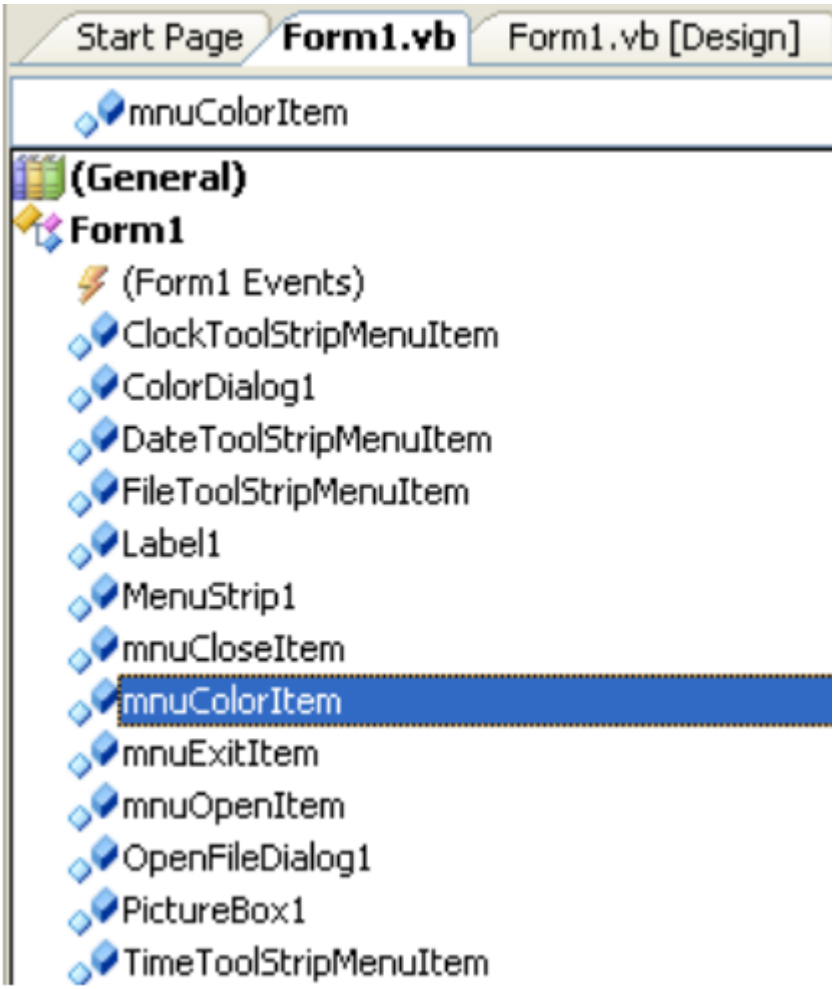
```
PictureBox1.Image = Nothing
        mnuCloseItem.Enabled = False
```

Khi mở ảnh rồi thì mục Close sáng lên, khi chọn vào mục này thì PictureBox1 không còn ảnh nữa và mục này lại bị vô hiệu hóa. Nhắp đôi vào mục Exit và nhập dòng mã: End

Tạo thủ tục mnuColorItem_Click bằng cách nhấn đúp hay chọn từ danh sách xổ xuống như hình 5.5.



Hình 5.5. Sự kiện tích chuột



Hình 5.6. Sự kiện

Nhập vào đoạn mã:

```
ColorDialog1.ShowDialog()
        Label1.ForeColor = ColorDialog1.Color
```

Chú thích mã:

- Phát biểu đầu tiên gọi ShowDialog() để hiển thị hộp thoại ColorDialog.

- Phát biểu thứ hai nhận giá trị màu trả về từ hộp thoại ColorDialog và gán cho màu chữ Text – ForeColor của điều khiển Label1. Ta có thể gán màu cho bất cứ thuộc tính nào như BackColor. Ngoài ra, ta cũng có thể thêm các thuộc tính khác cho hộp thoại ColorDialog trước khi gọi đến phương thức ShowDialog(). Một số thuộc tính và cách gọi được liệt kê như sau:

'ColorDialog1.FullOpen = True :Hiển thị khung tùy biến màu mở rộng

'ColorDialog1.AllowFullOpen = True: hiển thị nút định nghĩa màu tùy biến

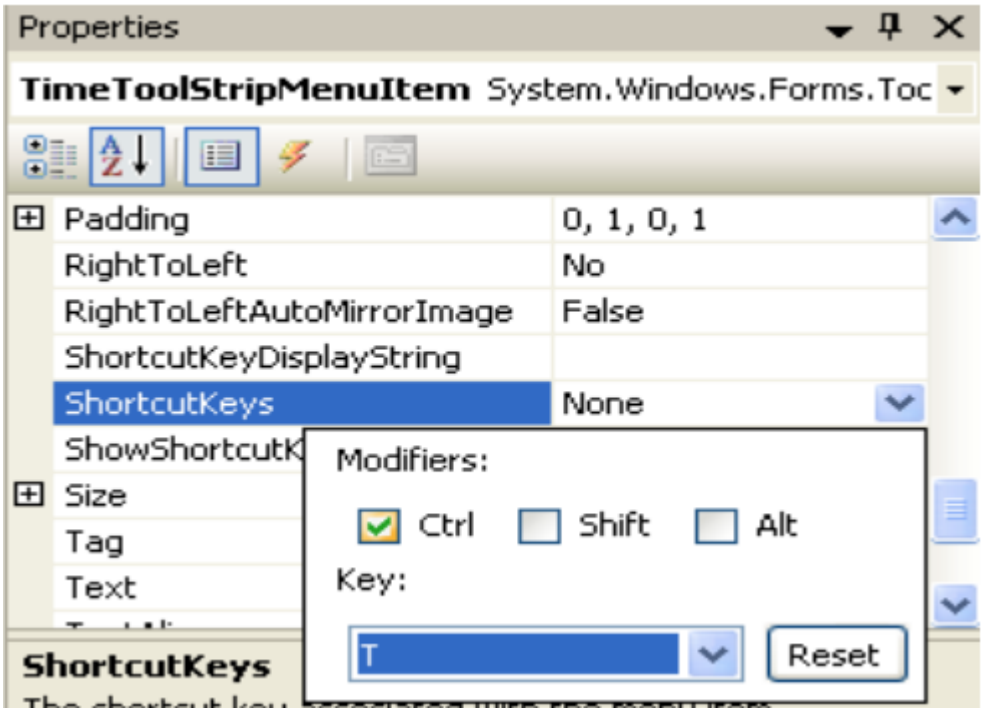
'ColorDialog1.AnyColor = True: cho phép chọn tất cả các loại màu

'ColorDialog1.ShowHelp = True: Hiển thị nút nhấn trợ giúp

'ColorDialog1.SolidColorOnly = True: Hiển thị chỉ những màu đặc

Khi chạy chương trình bằng cách nhấn phím F5 hay Start trên Standard Bar và thử tất cả các tính năng của chương trình. Phím tắt cho phép ta ấn tổ hợp phím để thực hiện lệnh mà không cần chọn menu. Ví dụ như Ctrl+C để sap chép một đoạn text trong Word. Chúng ta chọn gán các phím tắt cho menu trong chương trình MyMenu.

Trước hết mở giải pháp MyMenu ở chế độ thiết kế tích vào menu Clock trên Form, chọn mục Time và chọn chuột phải, chọn Properties. Thiết lập thuộc tính ShortCutKeys như hình 5.7.



Hình 5.7. đặt phím tắt

Tương tự chúng ta chọn các mục còn lại theo ý thích sao cho các phím nóng không trùng nhau.

5.2. Các thành phần giao diện cơ bản trong window

Giao diện:

Hình 5.8. Giao diện các phép toán cơ sở

Chương trình gồm hai textbox cho phép nhập hai giá trị để gán cho hai biến value1 và value2, bốn radiobutton cho phép chọn bốn toán tử khác nhau, khi đã nhập đầy đủ hai giá trị thì có thể thực hiện tính bằng cách nhấp chọn nút „thực hiện tính" và kết quả hiển thị trong ô textbox3 – kết quả. Xây dựng giao diện:

Tạo một giải pháp và thêm một dự án cùng tên BasicMath đồng thời thiết kế giao diện như hình 5.8. Viết mã:

- Khai báo biến: ta khai báo 2 biến value1, value2 ở đầu lớp form1 như sau:

Dim value1, value2 As Double

- Tạo thủ tục Button1_Click bằng cách tích đúp chuột vào nút „thực hiện tính‟ và nhập đoạn mã sau:

```
If TextBox1.Text = "" Or TextBox2.Text = "" Then
          MsgBox("Bạn cần nhập đầy đủ hai giá trị")
      Else
          value1 = CDbl(TextBox1.Text)
          value2 = CDbl(TextBox2.Text)
          If RadioButton1.Checked = True Then
              TextBox3.Text = value1 + value2
          End If
          If RadioButton2.Checked = True Then
              TextBox3.Text = value1 - value2
          End If
          If RadioButton3.Checked = True Then
              TextBox3.Text = value1 * value2
          End If
          If RadioButton4.Checked = True Then
              TextBox3.Text = value1 / value2
          End If
      End If
```

Chú thích mã:

- Hàm CDbl là hàm chuyển kiểu sang kiểu Double. Chọn phím F5 hay nút start để chạy chương trình.

CHƯƠNG 6: MẢNG

6.1. Làm việc với mảng

Mảng giúp quản lý các dữ liệu lớn hết sức dễ dàng. Việc truy cập các phần tử của mảng thông qua chỉ số. Việc khai báo mảng tương tự như khai báo biến. Việc khai báo thường chứa các thông tin như:

- Tên mảng: Tên đại diện cho mảng, việc truy cập một phần tử mảng gồm tên mảng và chỉ số mảng.
- Kiểu dữ liệu: Tất cả các phần tử trong mảng phải có cùng kiểu.
- Kích thước mảng: Là số chiều của mảng.
- Số phần tử của mảng: Số phần tử tối đa của mảng

Cú pháp chung khai báo mảng có kích thước là:

Dim ArrayName(Dim1Index, Dim2Index) As DataType

Trong đó:

- ArrayName: tên mảng
- Dim1Index và Dim2Index: là hai chiều của mảng
- Datatype: kiểu dữ liệu của mảng. Khi chưa xác định kiểu cụ thể, có thể dùng kiểu Object.

Ví dụ:

Khai báo Dim Employee(4) As String: khai báo mảng một chiều chứa 5 phần tử có tên là Employee có kiểu String.

Ta cũng có thể khai báo mảng một cách toàn cục trong module bằng từ khóa Public như sau: Public Employee(4) As String. Để khai báo mảng hai chiều mang tên ScoreBoard ta có thể khai báo như sau:

```
Dim ScoreBoard(1, 4) As Short
```

Mảng này gồm 2*5 = 10 phần tử tương ứng với 10 ô vuông gồm hai dòng và 5 cột đánh số từ 0.

6.2. Làm việc với các phần tử trong mảng

Sau khi khai báo, bạn có thể sử dụng mảng. Việc truy cập vào một phần tử của mảng nhờ tên mảng và chỉ số của mảng đặt trong ngoặc đơn, chỉ số là số nguyên, là biến nguyên hay biểu thức có giá trị. Để duyệt qua tất cả các phần tử trong mảng, dùng vòng lặp For…Next.

Ví dụ:

```
employee(3) = "Thanh Van"
```

Khai báo trên gán cho phần tử có chỉ số thứ 3 (tại ô thứ 4) tên là “Thanh Van”.

```
ScoreBoard(0, 2) = 12
```

Khai báo trên gán cho phần tử ở dòng 0, cột 2 giá trị là 12.

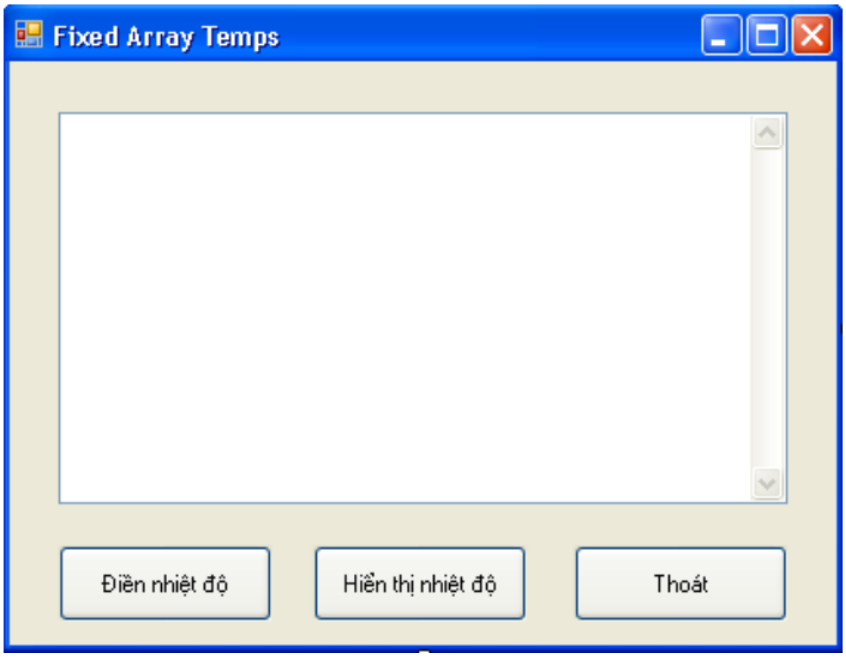
6.3. Sử dụng mảng có kích thước cố định

Bây giờ ta tạo ví dụ MyFixedArray sử dụng mảng một chiều có tên nhietdo để ghi lại giá trị nhiệt độ cao thấp hàng ngày trong tuần. Mảng này được khai báo ở đầu form và được

gán giá trị bằng hàm InputBox nhờ vòng lặp For…Next. Toàn bộ nội dung của mảng sau đó lại được hiển thị lại vào một textbox cũng nhờ vòng lặp For…Next.

Thiết kế giao diện:

Tạo mới một giải pháp và thêm vào một dự án có cùng tên là MyFixedArray. Thiết kế giao diện như hình 6.1.



Hình 6.1. Màn hình thiết kế

Trong đó: nút button1 có text là "Điền nhiệt độ", button2 là "Hiển thị nhiệt độ", button3 là "Thoát". Viết mã:

Trước hết ta khai báo mảng nhietdo ở ngay dưới dòng Public Class Form1 như sau:

```
Dim nhietdo(6) As Single
```

Khai báo như trên nghĩa là tất cả các thủ tục, các hàm đều có thể sử dụng mảng này. Tiếp theo ta tạo ra sự kiện nhập vào các giá trị nhiệt độ trong tuần bằng cách tạo thủ tục Button1_Click và nhập mã như sau:

```
Private Sub Button1_Click(ByVal sender As Object, _
  ByVal e As System.EventArgs) Handles Button1.Click
      Dim Prompt, tieude As String
      Dim i As Short
      Prompt = "Điền vào nhiệt độ của ngày."
      For i = 0 To UBound(nhietdo)
          tieude = "Ngày " & (i + 1)
          nhietdo(i) = CInt(InputBox(Prompt, tieude))
      Next
  End Sub
```

Trong đó, hàm Ubound(nhietdo) là hàm lấy về chỉ số trên của mảng nhietdo, trong trường hợp này là 6. Sau đó ta cho hiển thị các giá trị nhiệt độ trong bảy ngày trong tuần cũng như giá trị nhiệt độ trung bình bằng thủ tục Button2_Click khi người dùng click vào nút "Hiển thị nhiệt độ" như sau:

```
Private Sub Button2_Click(ByVal sender As Object, _
  ByVal e As System.EventArgs) Handles Button2.Click
```

```
        Dim ketqua As String
        Dim i As Short
        Dim tong As Single = 0
        ketqua = "Nhiệt độ của tuần: " & vbCrLf & vbCrLf
        For i = 0 To UBound(nhietdo)
            ketqua = ketqua & "Ngày " & (i + 1) & _
            vbTab & nhietdo(i) & vbCrLf
            tong = tong + nhietdo(i)
        Next
        ketqua = ketqua & vbCrLf & _
        "Nhiệt độ trung bình: " & _
        Format(tong / 7, "0.0")
        TextBox1.Text = ketqua
    End Sub
```

Thủ tục này lại sử dụng vòng lặp For…Next để duyệt lại các phần tử trong mảng sau khi đã được gán giá trị ở thủ tục button1_Click. Biến ketqua được dùng để làm chuỗi kết xuất gộp các giá trị phần tử mảng. Sau mỗi lần gộp ta sử dụng hằng số vbCrLf điều khiển dấu ngắt dòng và dấu về đầu dòng (tương đương với hai hàm Chr(13) và Chr(10)). Hằng vbTab để phân cách giữa phần ghi ngày và ghi nhiệt độ.

Ta tạo thủ thục Button3_Click và nhập phát biểu End để kết thúc chương trình. Chạy chương trình: Khi chạy chương trình và nhập đủ giá trị nhiệt độ cho 7 ngày.

6.4. Tạo mảng động

Việc dùng mảng là rất thuận tiện. Tuy nhiên khi ta chưa biết chính xác số phần tử của mảng là bao nhiêu thì như thế nào? Ví dụ khi ta muốn để người dùng nhập vào bao nhiêu nhiệt độ tùy thích, nhập càng nhiều thì độ chính xác càng cao. VB giải quyết việc này bằng mảng động. Kích thước mảng động chỉ được chỉ định khi chương trình thực thi chứ không định trong lúc viết mã. Việc khai báo trước kích thước mảng là không cần thiết nhưng cũng cần dành chỗ trước cho mảng đó. Các bước tạo mảng động:

- Chỉ định tên và kiểu cho mảng khi thiết kế form, ví dụ Dim nhietdo() As Single
- Thêm mã xác định kích thước mảng khi chương trình thực thi. Ví dụ khi chương trình chạy ta hỏi xem người dùng muốn nhập bao nhiêu ngày, ví dụ:

```
Dim songay As Integer
songay = InputBox("Ban muon nhap bao nhieu ngay?", "Tao mang dong")
```

- Dùng biến songay để định lại kích thước mảng (trừ đi 1 vì mảng tính từ 0). Ví dụ

```
If songay > 0 Then ReDim nhietdo(songay - 1)
```

- Tiếp theo ta dùng hàm Ubound(nhietdo) để xác định số phần tử của mảng.

Bây giờ chúng ta sẽ làm lại ví dụ trên sử dụng mảng động:

- Trước hết, ta khai báo lại mảng động và khai báo biến songay chứa số ngày người dùng muốn nhập bằng đoạn mã ngay dưới dòng khai báo lớp form1:

```
Dim nhietdo() As Single
Dim songay As Integer
```

- Sau đó sửa lại mã của thủ tục Button1_Click như sau:

```
Dim Prompt, tieude As String
    Dim i As Short
    Prompt = "Điền vào nhiệt độ của ngày."
    'Nhap so ngay muon ghi nhiet do
    songay = InputBox("Ban muon nhap bao so ngay?","")
    If songay > 0 Then ReDim nhietdo(songay - 1)
    For i = 0 To UBound(nhietdo)
        tieude = "Ngày " & (i + 1)
        nhietdo(i) = CInt(InputBox(Prompt, tieude))
Next
```

- Tiếp theo thay số 7 trong thủ tục Button2_Click bằng biến songay:

```
ketqua = ketqua & vbCrLf & _
   "Nhiệt độ trung bình: " & _
Format(tong / songay, "0.0")
```

- Ta có thể dùng phát biểu Try…Catch để bắt lỗi nếu người dùng nhập vào một số nhỏ hơn 0.
- Chạy lại chương trình và kết quả rõ ràng linh động hơn.

CHƯƠNG 7: VB.NET KẾT NỐI CƠ SỞ DỮ LIỆU DÙNG ADO.NET

7.1 ADO.NET là gì

ActiveX Data Object.NET (ADO.NET) là một thư viện phần mềm .NET Framework. Bao gồm các thành phần phần mềm cung cấp dịch vụ truy cập dữ liệu. ADO.NET được thiết kế để cho phép các nhà phát triển viết mã được quản lý cho việc tiếp cận các nguồn dữ liệu bị ngắt kết nối, có thể có quan hệ hoặc không quan hệ (như XML hoặc dữ liệu ứng dụng). Tính năng này của ADO.NET giúp ứng dụng có thể chia sẻ dữ liệu hay các ứng dụng phân tán.

ADO.NET cung cấp kết nối truy cập với CSDL bằng cách sử dụng .NET Managed Provinder (*mặc định .Net Framework có hai Managed Providers là SQL Managed Provider và OleDB Managed Provider*) và truy cập bị ngắt kết nối bằng cách sử dụng **Datasets**, đó là các ứng dụng sử dụng kết nối cơ sở dữ liệu chỉ trong thời gian truy xuất dữ liệu hoặc cập nhật dữ liệu. Datasets là thành phần giúp đỡ để lưu trữ các dữ liệu liên tục trong bộ nhớ để cung cấp truy cập khi bị ngắt kết nối sử dụng các nguồn tài nguyên cơ sở dữ liệu một cách hiệu quả và khả năng mở rộng tốt hơn.

ADO.NET được phát triển từ ADO, cũng là một công nghệ tương tự như ADO.NET với một vài thay đổi cấu trúc cơ bản. Mặc dù có một vài trường hợp để làm việc trong chế độ bị ngắt kết nối bằng cách sử dụng ADO, nhưng dữ liệu được chuyển đến CSDL trong ADO.NET hiệu quả hơn bằng cách sử dụng **Data Adapters**. Các đại diện trong bộ nhớ của dữ liệu giữa ADO và ADO.NET là khác nhau. ADO.NET có thể giữ các dữ liệu trong một bảng kết quả duy nhất, nhưng ADO giữ nhiều bảng cùng với các chi tiết của mối quan hệ. Không giống như ADO, việc truyền dữ liệu giữa các ứng dụng bằng cách sử dụng ADO.NET không sử dụng COM (Component Object Model) để sắp xếp mà dùng datasets, nó truyền dữ liệu như là một dòng XML.

Kiến trúc ADO.NET dựa trên hai yếu tố chính là : Dataset và .NET Framework data provider

Data provides gồm các thành phần sau :

Datasets

- Một tập hợp đầy đủ của CSDL bao gồm các bảng có liên quan , các ràng buộc và mối quan hệ của chúng
- Chức năng giống như truy cập dữ liệu từ xa từ dịch vụ Web XML
- Thao tác với dữ liệu động
- Xử lý dữ liệu theo kiểu không kết nối
- Cung cấp cho xem thứ bậc XML của dữ liệu quan hệ
- Xử dụng các công cụ XSLT và XPath Query để hoạt động trên dữ liệu

.NET Framework Data Provider bao gồm các thành phần sau đây để thao tác dữ liệu :

- Connection: Cung cấp kết nối với nguồn dữ liệu (database). Hay nói cách khác nó là đối tượng có nhiệm vụ thực hiện kết nối tới CSDL
- Command: Thực hiện các thao tác cần thiết với CSDL để lấy dữ liệu, sửa đổi dữ liệu hoặc thực hiện các thủ tục được lưu trữ. Hiểu đơn giản là nó đưa ra các mệnh lệnh đối với CSDL

- DataReader: Cung cấp việc đọc dữ liệu và chỉ đọc 1 dòng dữ liệu tại một thời điểm và nó đọc theo chiều tiến từ đầu đến cuối.

- DataApdater: Nó đóng vài trò như là cầu nối giữa Dataset và CSDL, tải dữ liệu lên dataset hoặc đồng bộ các thay đổi ở dataset về lại CSDL

- Datasets: Để lưu trữ, truy cập từ xa và lập trình với dữ liệu phẳng (Flat), dữ liệu XML và dữ liệu quan hệ

ADO.NET cho phép:

- Thao tác với CSDL trong cả hai môi trường là Connected data & Disconnected data.
- Làm việc tốt với XML
- Tương tác được với nhiều nguồn dữ liệu
- Làm việc trên môi trường internet

7.2 Lập trình với ADO.NET

Cơ sở dữ liệu rất quan trọng trong việc lưu trữ thông tin. Dữ liệu có rất nhiều nguồn và đa dạng. VB.NET được thiết kế với mục đích truy xuất, hiển thị, phân tích cơ sở dữ liệu. ADO.NET là mô hình lập trình truy xuất dữ liệu chung cho tất cả các ngôn ngữ và chương trình Windows. Với ADO.NET, chúng ta có thể truy xuất đến mọi hệ cơ sở dữ liệu theo cùng cách thức và mã chương trình như nhau.

7.2.1 Thuật ngữ về cơ sở dữ liệu

Một số thuật ngữ về cơ sở dữ liệu:

- Cơ sở dữ liệu là một file tổ chức thông tin thành các bảng gọi là bảng (Table).
- Mỗi bảng bao gồm nhiều hàng và cột, cột thường được gọi là trường (field) và dòng được gọi là bản tin (record).

Mô hình truy xuất cơ sở dữ liệu trong ADO.NET: thiết lập kết nối đến cơ sở dữ liệu. Tiếp theo đối tượng điều phối (data adapter) được tạo ra để truy vấn dl từ các bảng. Sau đó tạo các đối tượng DataSet chứa bảng dữ liệu mà chúng ta muốn trích dữ liệu.

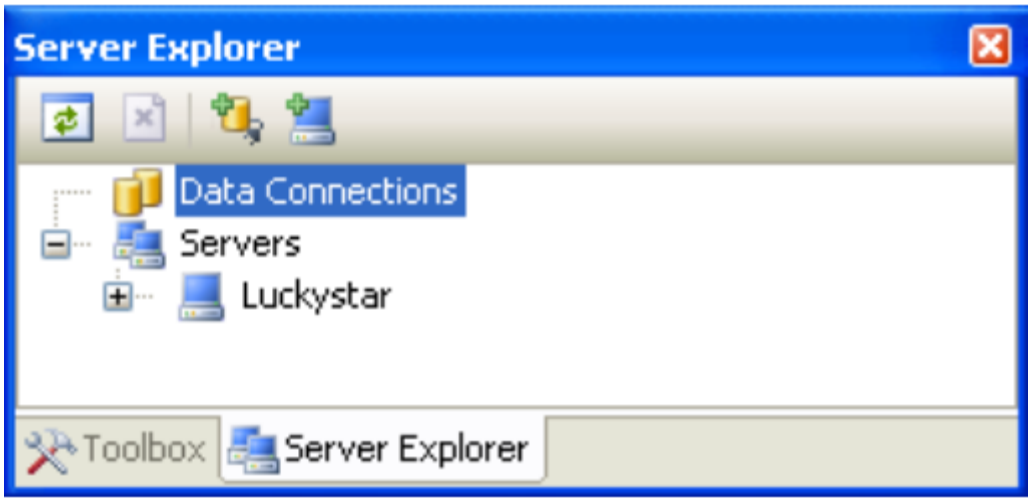
DataSet chỉ tạo bản sao của bảng dữ liệu mà thôi. Cuối cùng là gán thông tin trong DataSet vào các đối tượng hiển thị trên Form như TextBox, Label, Button, DataGrid,…

7.2.2 Làm việc với cơ sở dữ liệu Access

MS Access có tên Students.mdb. Sau khi đã biết cách kết nối và đưa dữ liệu vào dataset, chúng ta xây dựng và tích hợp chúng vào giao diện của form.

Chúng ta tạo mới một Solution có tên MyADOForm và thêm vào một dự án cùng tên.

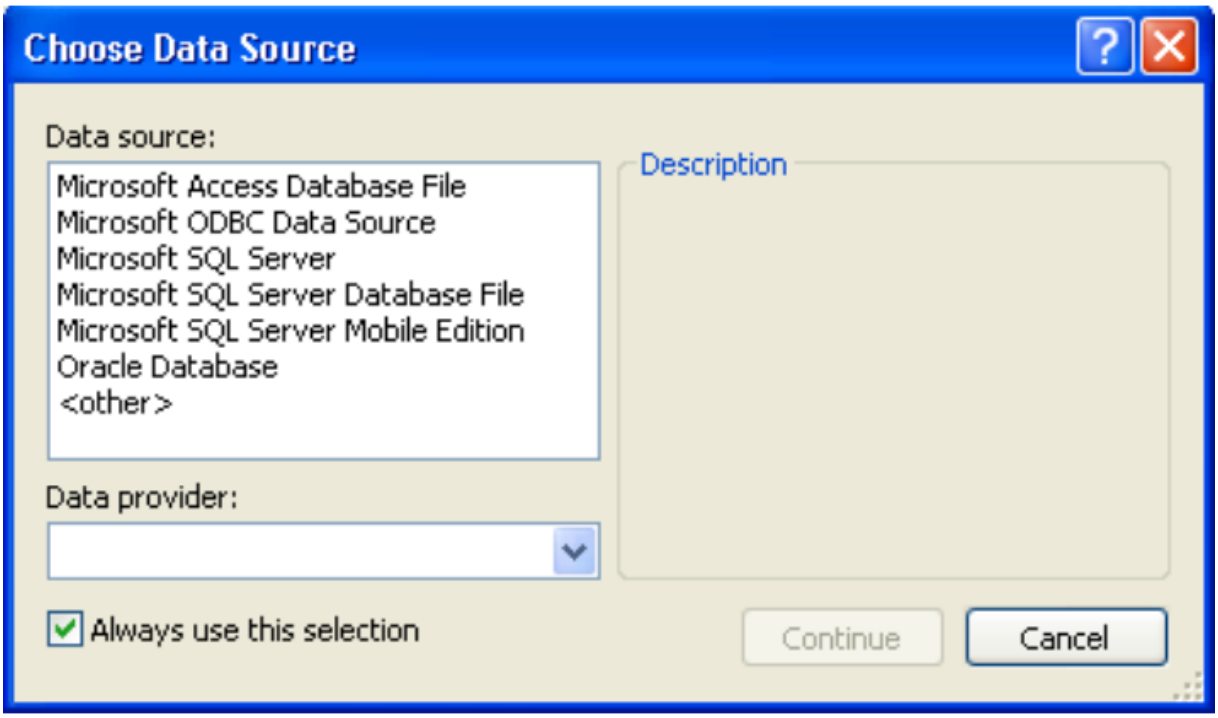
Chọn View | Server Explorer từ menu để hiện cửa sổ Server Explorer như hình sau:



Hình 7.1 Giao diện kết nối cơ sở dữ liệu cục bộ

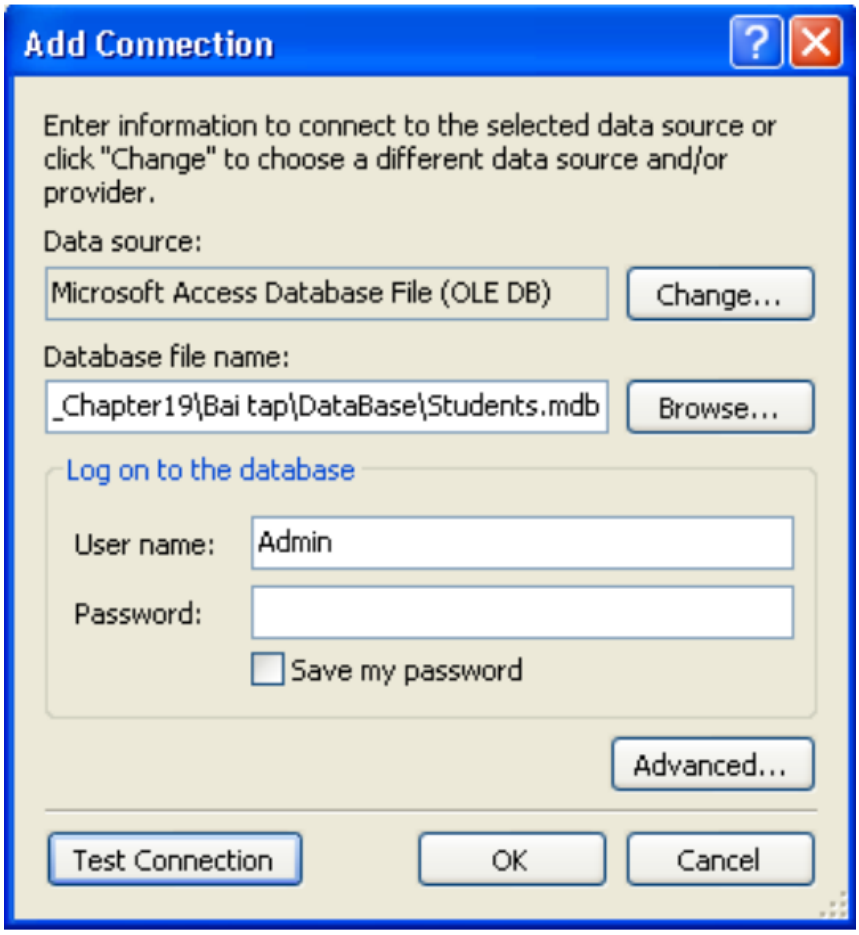
Đây là công cụ đồ họa cho phép kết nối đến cơ sở dữ liệu cục bộ, trên server theo mô hình client server. Ta có thể dùng nó để xem cấu trúc trong cơ sở dữ liệu, xem thuộc tính của bảng, kiểu dữ liệu của trường và bản tin trong cơ sở dữ liệu. Chúng ta có thể kéo các kết nối và bảng dữ liệu trong cửa sổ này để tạo ra đối tượng dữ liệu cho chương trình.

Chúng ta có thể dùng kéo thả, bằng cách tích chuột vào nút Connect To DataBase trong cửa sổ Server Explorer. Một hộp thoại Choose Data Source hiện ra cho phép ta chọn nguồn dữ liệu.



Hình 7.2 Giao diện chọn hệ cơ sở dữ liệu

Chúng ta chọn Microsoft Access DataBase File và chọn nút Continue, xuất hiện hộp thoại Add Connection như hình sau:



Hình 7.3 Giao diện chọn đường dẫn tệp cơ sở dữ liệu

Ta chọn đường dẫn đến cơ sở dữ liệu bằng cách chọn nút Browse, sau đó chọn cơ sở dữ liệu Students.mdb. Chúng ta có thể kiểm tra xem kết nối có thành công không bằng cách chọn nút Test Connection, và cũng có thể tùy chỉnh kết nối bằng cách chọn nút Advanced.

Chúng ta thấy dòng mã kết nối ở ô cuối cùng, dòng mã có nội dung như sau:

"Provider=Microsoft.Jet.OLEDB.4.0;DataSource="D:\Data\Studying\VS.Net05\Bai tap\DataBase\Students.mdb""

Nhấn OK để thêm kết nối vào Server Explorer.

7.2.3 Tạo bộ điều phối dữ liệu Data Adapter

Bước hai trong thao tác cơ sở dữ liệu là tạo bộ điều phối Data Adapter. Data Adapter sẽ định nghĩa chính xác những thông tin mà ta muốn lấy trong cơ sở dữ liệu, là nền tảng để tạo DataSet.

VB.NET cung cấp rất nhiều cách tạo bộ điều phối. Cách đơn giản nhất là ta kéo các biểu tượng bảng trong Server Explorer vào cửa sổ form trong chế độ thiết kế. Ta cũng có cách thứ hai dùng công cụ Data Adapter Configuration Winzard. Ta gọi đến công cụ này bằng cách chọn đối tượng OledbDataAdapter trên tab Data của ToolBox và đặt nó lên form.

7.2.4 Sử dụng đối tượng điều khiển OleDbDataAdapter

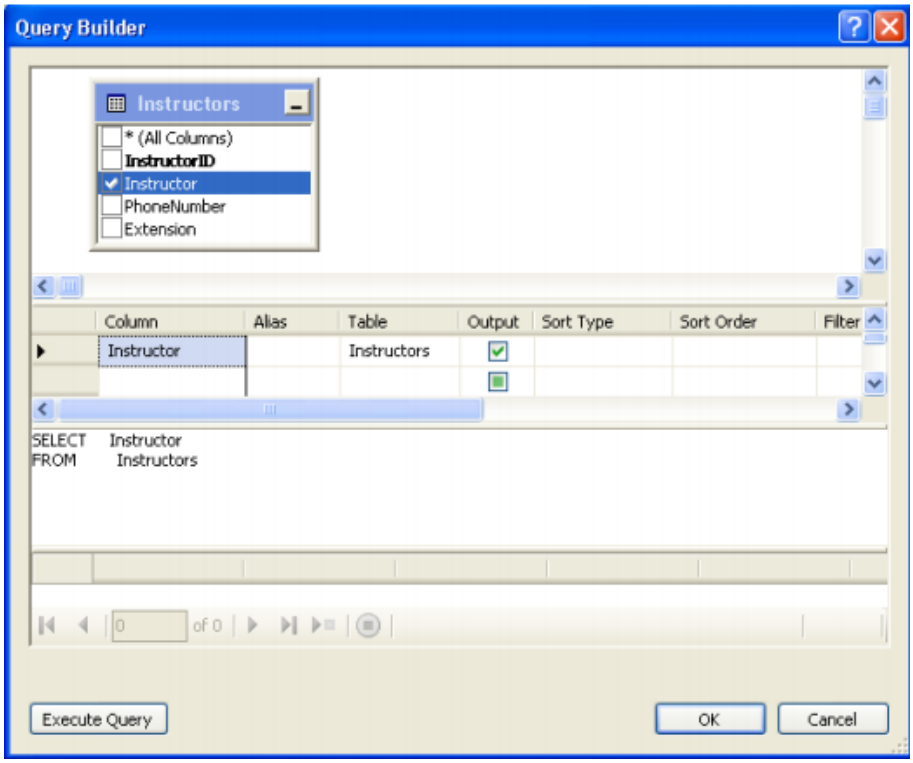
Chọn tab Data trong cửa sổ ToolBox. Tab này chứa các điều khiển để thao tác với cơ sở dữ liệu. Trong tab này có hai đối tượng OleDbConnection và sqlConnection đều cho phép tạo kết nối đến cơ sở dữ liệu. Nhưng chúng ta đã kết nối bằng Server Explorer nên không cần hai đối tượng này nữa.

Kéo đối tượng OleDbDataAdapter vào trong form. Nếu đối tượng này không xuất hiện, ta có thể thêm nó vào bằng cách tích chuột phải vào tab Data chọn Choose Item, xuất hiện cửa sổ Choose ToolBox Items. Chọn tab .Net Framework Components và chọn OleDbAdapter. Chọn OK để hoàn thiện việc thêm Item này cho ToolBox. Chúng ta cũng có thể làm tương tự với các đối tượng khác.

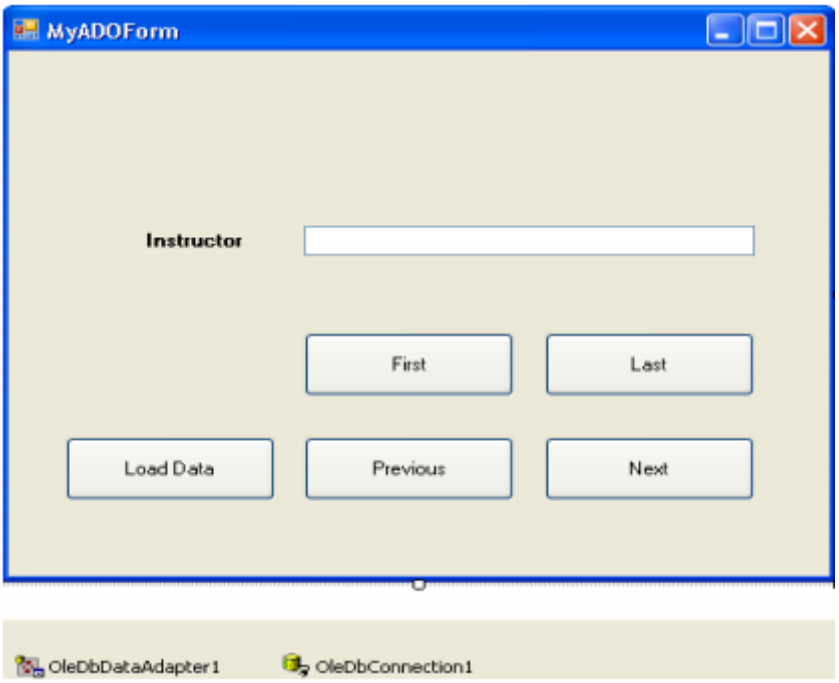
OleDbAdapter được thiết kế để kết nối đến cơ sở dữ liệu Access. Khi kéo thả đối tượng này vào form thì VS.NET sẽ tao trình Data Adapter Configuration Winzard. Một màn hình khởi đầu, chọn Next để chuyển sang màn hình thứ hai.

Ta chọn Next hai lần để xuất hiện màn hình soạn thảo câu lệnh SQL. Nếu chúng takhông dùng câu lệnh SQL, có thể nhấn vào nút Query Builder, để dùng kéo thả mà Visual Basic hỗ trợ. Sau đó chọn bảng Instructors, chọn Add, chọn Close để đóng cửa sổ này lại.

Ta thấy trong bảng Instructors có các ô CheckBox tương ứng với các trường, Query sẽ tạo câu lệnh tương ứng để rút thông tin của bảng. Trong ví dụ sau chúng ta chỉ rút thông tin từ một cột trong bảng, ta chọn cột Instructor như hình H.7.4, chọn OK. (câu lệnh SQL để trích rút dữ liệu).



Hình 7.4 Chọn trường để xây dựng câu lệnh SQL



Hình 7.5 Giao diện Form

Sau khi nhấn OK, một cửa sổ Generate The SQL Statement hiện ra hiển thị câu lệnh SQL ta vừa tạo. Chọn Finish đê hoàn thành việc tạo đối tượng điều phối.

7.2.5 Làm việc với DataSet

Chúng ta tạo ra đối tượng trình diễn dữ liệu cho người dùng thao tác, đối tượng DataSet. Nó là ảnh của cơ sở dữ liệu nên mọi thao tác của người dùng sẽ chưa ảnh hưởng đến cơ sở dữ liệu cho đến khi có yêu cầu cập nhật. Trong ví dụ ta tạo đối tượng DataSet trình diễn thông tin trong cột Instructor của bảng Instructors trong cơ sở dữ liệu Students.mdb.

Ta chọn phải chọn form1 nếu không chọn thì các lệnh tạo DataSet sẽ không hiển thị trên menu. Chọn Data | Generate DataSet từ menu để làm xuất hiện hộp thoại Generate DataSet. Chúng ta đặt tên nào tùy thích tại ô New, ví dụ là DsInstructors. Chọn ô checkBox Add this dataset to the designer để Visual Basic đưa dataser vào khay công cụ. Chọn OK và đối tượng DataSet DsINstructors được tạo ra trên khay công cụ. VB.NET sẽ tự thêm vào một file có tên DsInstructors.xsd trong cửa sổ Solution Explorer. File này chứa các thông tin về dữ liệu theo khuôn dạng XML.

7.3 Sử dụng các điều khiển ràng buộc dữ liệu

Chúng ta sử dụng các thành phần điều khiển như Textbox, Label, Button để đưa cơ sở dữ liệu lên form. Để trình bày được ta tạo mối ràng buộc dữ liệu (data binding), tức là dữ liệu hiển thị lên trong các điều khiển sẽ phụ thuộc vào nguồn dữ liệu có trong DataSet hay DataAdapter.

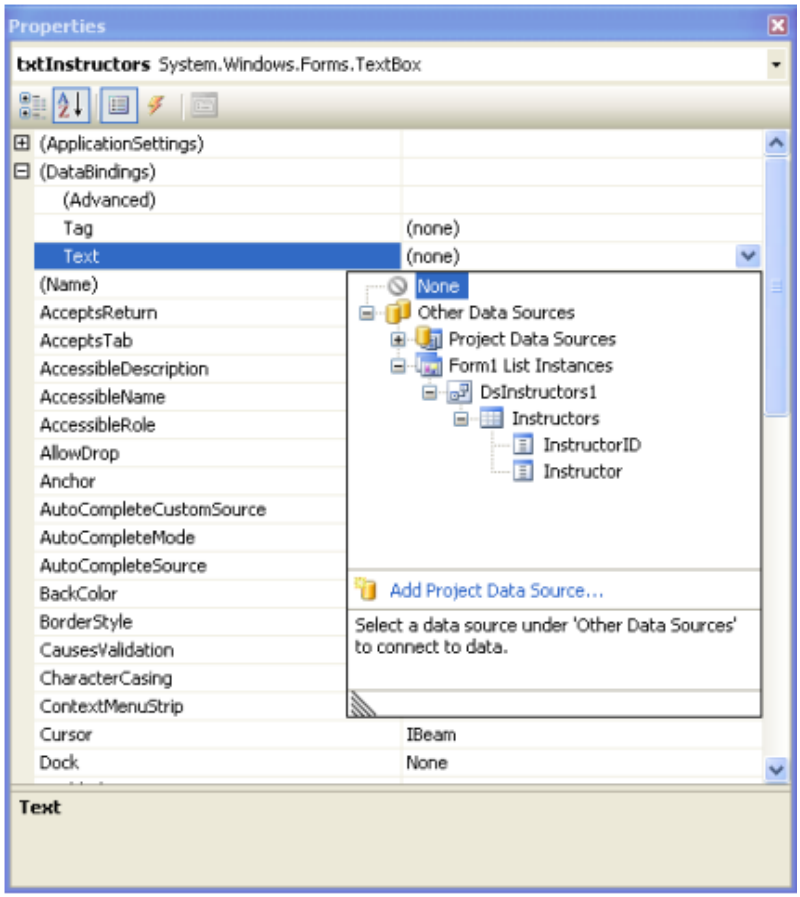
Chúng ta ràng buộc dữ liệu với các điều khiển sau: TextBox, Label, ListBox, ComboBox, RadioButon, DataGrid và PictureBox. Trong đó đặc biệt là DataGrid, nó cho phép ta hiển thị toàn bộ nội dung của DataSet.

Trong ví dụ ta ràng buộc dữ liệu vào TextBox để hiển thị thông tin trong bảng Instructors của cở dữ liệu Students.mdb. Chúng ta thiết kế giao diện form như hình 8.5 trên. Trong đó thuộc tính của các điều khiển như sau:

- Button First: Name – btnFirst, enable – False
- Button Last: Name – btnLast, enable – False
- Button Next: Name – btnNext, enable – False
- Button Previous: Name – btnPrevious, enable – False
- Button Load Data: Name – btnLoadData
- TextBox1: Name - txtInstructors

Các điều khiển còn lại có thuộc tính như hình.

Bây giờ ta sẽ tiến hành ràng buộc dữ liệu là các trường (cột dữ liệu – field) vào textbox txtInstructors. Để làm điều này, ta chọn ô textbox và mở Properties của nó ra. Tích chuột vào dấu (+) bên cạnh nhánh thuộc tính DataBindings và chọn ô text, tích chuột vào nút mũi tên đi xuống và ta có thể nhìn thấy nguồn dữ liệu DsInstructors1 hiển thị trong danh sách như hình sau:



Hình 7.6 Giao diện chọn nguồn dữ liệu và thuộc tính cho từng thành phần

Chọn cột Instructor để chỉ định trường này sẽ hiển thị trong ô textbox txtInstructor. Ta đã thực hiện ràng buộc xong, sau đó viết mã để xuất dữ liệu khi chương trình thực thi.

Chúng ta tạo thủ tục btnLoadData_Click bằng cách từ cửa sổ thiết kế form và nhấn đúp chuột vào nút Load Data rồi nhập đoạn mã chương trình :

```
DsInstructors1.Clear()
OleDbDataAdapter1.Fill(DsInstructors1)
btnFirst.Enabled = True
btnLast.Enabled = True
btnNext.Enabled = True
btnPrevious.Enabled = True
```

Đây là mã để xóa sạch dữ liệu trong DataSet DsInstructors1 ban đầu. Sau đó bộ điều phối DataAdapter1 cập nhật dữ liệu vào đối tượng DataSet DsInstructors1 mà chúng ta đã tạo ra ở bước 3 bằng phương thức Fill().

Chạy thử chương trình: chọn F5 để kiểm thử chương trình. Khi chương trình chạy, chọn nút Load Data để chương trình hiển thị bản ghi đầu tiên của trường Instructor trong bảng dữ liệu. Sau đó chúng ta mở rộng một số chức năng khác của ứng dụng cơ sở dữ liệu như duyệt các bản ghi, đếm và hiển thị số bản ghi hiện hành.

Các thao tác cơ sở dữ liệu gồm:

- Tạo kết nối đến cơ sở dữ liệu cần truy xuất.
- Tạo đối tượng điều phối DataAdapter.
- Tạo đối tượng trình diễn DataSet.
- Tạo ràng buộc dữ liệu vào các điều khiển hiển thị dữ liêu.

Ta có thể thêm các yêu cầu: cập nhật, thống kê, tìm kiếm, …

7.4 Tạo các điều khiển duyệt xem dữ liệu

Ở đây chúng ta chỉ dừng lại ở việc ràng buộc dữ liệu và hiển thị được bản ghi đầu tiên vào ô textbox. Trong phần tiếp theo chúng ta sẽ tạo ra các nút cho phép duyệt qua các bản ghi khác nhau, xem từ bản ghi đầu tiên đến cuối cùng.

ADO.NET cho phép quản lý và duyệt qua các bản ghi (record) bằng đối tượng CurrentManager. Với đối tượng này ta có thể biết được vị trí hiện hành của bản ghi, đi đến bản ghi sau cùng, trở về bản ghi đầu tiên, đến bản ghi kế, đến bản ghi trước. Mỗi DataSet đều có sẵn đối tượng CurrentManager và mỗi đối tượng form đều có thuộc tính BindingContext theo dõi tất cả đối tượng CurrentManager trên form.

Trong phần trên chúng ta đã tạo ra bốn nút chọn lần lượt First, Last, Next, Previous. Giờ chúng ta sẽ viết mã cho chúng. BindingContext, CurrentManager để duyệt qua các bản ghi.

Tạo thủ tục btnFirst_Click với nội dung như sau:

```
Me.BindingContext(DsInstructors1,"Instructors").Position = 0
btnFirst.Enabled = False
btnNext.Enabled = True
btnLast.Enabled = True
```

Mã này hiển thị bản ghi đầu tiên của DsInstructors1 sử dụng đối tượng BindingContext. Nó gán giá trị 0 cho thuộc tính Position (vị trí) để con trỏ hiện hành của dữ liệu chuyển đến bản ghi đầu tiên.

Tạo thủ tục btnLast_Click và nhập đoạn mã sau:

```
'Đếm tổng số bản ghi
Dim tongsobanghi As Integer = Me.BindingContext
(DsInstructors1, "Instructors").Count
'Chuyển con trỏ đến bản ghi cuối cùng
Me.BindingContext(DsInstructors1, "Instructors").Position =
tongsobanghi - 1
btnLast.Enabled = False
btnFirst.Enabled = True
btnPrevious.Enabled = True
```

```
btnNext.Enabled = False
```

Tạo thủ tục btnNext_Click và nhập vào đoạn mã sau:

```
'Đếm số bản ghi hiện hành
Dim tongsobanghi As Integer = Me.BindingContext
(DsInstructors1, "Instructors").Count
'Nếu chưa phải là bản ghi cuối thì next lên 1
If Me.BindingContext(DsInstructors1, "Instructors").Position <
tongsobanghi - 1 Then
Me.BindingContext(DsInstructors1, _
"Instructors").Position += 1
btnFirst.Enabled = True
btnPrevious.Enabled = True
btnLast.Enabled = True
Else
btnNext.Enabled = False
btnLast.Enabled = False
btnFirst.Enabled = True
btnPrevious.Enabled = True
End If
```

Thủ tục btnPrevious_Click:

```
'Nếu chưa phải là bản ghi đầu thì lùi lại 1
If Me.BindingContext(DsInstructors1,"Instructors").Position >
0 Then
Me.BindingContext(DsInstructors1,"Instructors").Position -= 1
btnFirst.Enabled = True
btnLast.Enabled = True
btnNext.Enabled = True
Else
btnFirst.Enabled = False
btnPrevious.Enabled = False
End If
```

Vậy là chúng ta đã tạo xong các nút cho phép duyệt qua các bản ghi. Chọn F5 để chạy chương trình. Ấn nút Load Data để hiển thị dữ liệu vào textbox. Ấn các phím để duyệt qua các bản ghi trong cơ sở dữ liệu. Chọn nút Close ở góc phải trên của form để đóng chương trình lại. Chúng ta sẽ tạo điều khiển label cho hiển thị vị trí bản ghi hiện hành để người dùng tiện quan sát.

7.5 Hiển thị vị trí của bản ghi hiện hành

Ngoài cơ chế duyệt xem các bản ghi, ta cũng cần cho người dùng biết đó là bản ghi thứ mấy. Chúng ta sẽ thêm một nhãn Label để hiển thị thứ tự của bản ghi. Từ thiết kế form và thêm vào một nhãn label1 có thuộc tính Name là lblIndexOfRecord, thuộc tính Text của nhãn là "Record 0 of 0". Ta tạo một thủ tục có tên count() ở ngay dưới phát biểu khai báo form1 như sau:

```
Private Sub Count()
```

```
Dim tongsobanghi, banghihienhanh As Integer
tongsobanghi = Me.BindingContext _
(DsInstructors1, "Instructors").Count
banghihienhanh = Me.BindingContext _
(DsInstructors1, "Instructors").Position + 1
lblIndexOfRecord.Text = "Record " & _
banghihienhanh.ToString & "Of " & tongsobanghi.ToString
End Sub
```

Thủ tục này sẽ gán thuộc tính count của đối tượng BindingContext vào biến tongsobanghi và thuộc tính Position của nó cho biến banghihienhanh nhưng cộng thêm 1 vì thứ tự bản ghi trong bảng dữ liệu được tính từ 0. Sau đó hai giá trị của hai biến trên được gán cho thuộc tính Text của điều khiển Label lblIndexOfRecord.

Để thủ tục này phát huy tác dụng thì bạn sẽ thêm lời gọi thủ tục này trong các thủ tục khác như btnFirst_Click, btnLast_Click, btnPrevious_Click, btnNext_Click như sau: Count()

Chương trình của chúng ta đến đây là hoàn thiện. Chọn F5 để chạy chương trình.

7.6 Sử dụng DataGrid để hiển thị dữ liệu trong bảng

Trong phần này chúng ta dùng DataGrid để hiển thị dữ liệu của bảng trong cơ sở dữ liệu Students.mdb. Chúng ta điền đầy đủ nội dung khung lưới bằng dữ liệu của bảng ở dạng chuỗi sau đó thực hiện một số thao tác định dạng, sắp xếp và ghi lại những thay đổi trong DataGrid trở lại cơ sở dữ liệu. Giống như TextBox, Ta có thể ràng buộc dữ liệu trong DataSet vào DataGrid. Ràng buộc này thông qua hai thuộc tính là DataSource và DataMember. Trong MyDataGridBinding chúng ta đưa toàn bộ nội dung của bảng Instructors có trong DsInstructors1 hiển thị trong khung lưới DataGrid.

Ví dụ MyDataGridBinding:

Tạo mới một Solution và thêm vào một dự án cùng tên là MyDataGridBinding.

Kết nối cơ sở dữ liệu:

Nếu trong ví dụ trước chúng ta đã kết nối với cơ sở dữ liệu rồi thì trong cửa sổ Server Explorer sẽ có một kết nối đến cơ sở dữ liệu nhưng có thêm một gạch đỏ ở kết nối đó. Nếu muốn sử dụng lại kết nối này ta chỉ việc chọn nút Refresh. Trong Ví dụ này, giả sử ta copy file cơ sở dữ liệu Students.mdb vào cùng thư mục với dự án để tiện thao tác. Ta chọn nút để thực hiện kết nối đến cơ sở dữ liệu như đã biết. Chọn cơ sở dữ liệu mà chúng ta vừa copy vào thư mục chứa dự án. Chọn OK để hoàn thành kết nối.

Tạo đối tượng điều phối DataAdapter:

Tạo thêm đối tượng OleDbDataAdapter vào trong form bằng cách kéo nó từ ToolBox ở tab data vào trong form. Khi đó một cửa sổ Data Adapter Configuration xuất hiện.

Chọn Next hai lần để hiện cửa sổ Generate SQL Statements. Tại đây Ta có thể tự gõ câu lệnh SQL hay sử dụng Query Builder. Ví dụ ta chọn nhập trực tiếp câu lệnh SQL.

```
SELECT Extension, PhoneNumber, Instructor, InstructorID
FROM Instructors
```

Câu lệnh này lấy dữ liệu ở cả bốn trường trong bảng Instructors. Chọn Next để xem kết quả của Winzard. Lúc này, trình Winzard tự tạo ra các câu lệnh là Update (cập nhật), Select,

Insert (chèn), Delete (xóa). Chọn Finish để kết thúc quá trình xây dựng tạo đối tượng điều phối DataAdapter có tên OleDbDataAdapter1.

Tạo đối tượng trình diễn DataSet:

Từ Form: Chọn Data | Generate DataSet từ menu làm hiện hộp thoại Generate DataSet như đã biết. Tại ô New bạn nhập vào tên DsInstructors và đánh dấu vào ô checkBox Add this DataSet To The Designer để Visual studio tạo ra đối tượng DataSet và đưa nó vào khay hệ thống. Chọn OK để Visual studio tạo đối tượng DataSet cho bảng Instructors trong cơ sở dữ liệu Students.mdb. Lúc này cửa sổ form có thêm các đối tượng vừa chọn.

Chúng ta đã hoàn thành ba bước đầu của thao tác với cơ sở dữ liệu Tiếp sử dụng DataGrid để trình bày dữ liệu.

Tạo đối tượng DataGrid:

Kéo form cho kích thước rộng hơn để chứa đủ khung lưới DataGrid với 4 cột và 10 dòng. Đưa điều khiển DataGrid trên ToolBox vào trong form. Kéo chiều dài của nó cho phù hợp với chiều kích thước của form.

Tạo thêm một nút nhấn nữa vào form. Đặt thuộc tính Name là btnLoad và text là "Load Data". Mở Properties của DataGrid và đặt thuộc tính Anchor của nó là cả Left, Right, Top, Bottom.

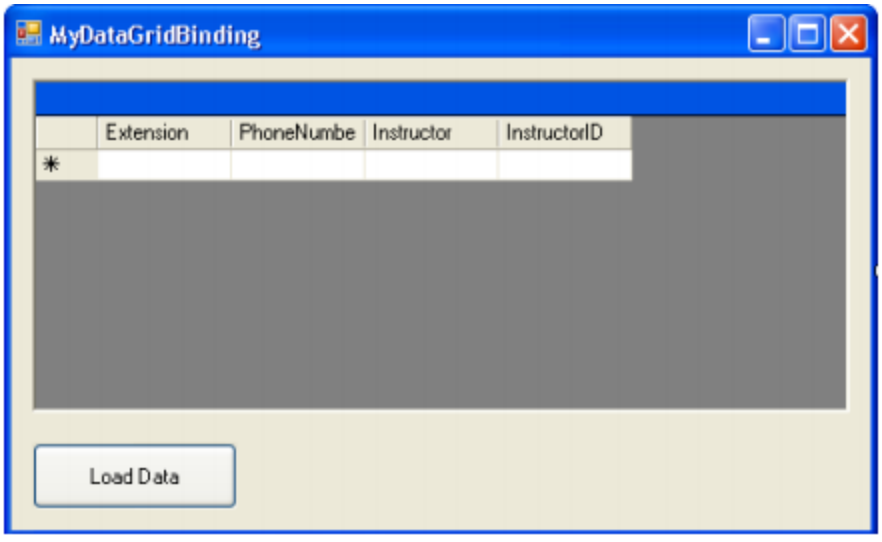
Tiếp theo ta sẽ dùng thuộc tính DataSource và DataMember để ràng buộc dữ liệu trong DsInstructors1 vào khung lưới DataGrid. Chúng ta hiển thị các tùy chọn của thuộc tính DataSource trong cửa sổ Properties. Một chương trình có thể có rất nhiều DataSet nhưng tại một thời điểm khung lưới chỉ có thể thể hiện một DataSet mà thôi, ta chọn DsInstructors1.

Tiếp chọn thuộc tính DataMember là Instructors.

Ngay sau khi ta chọn xong hai thuộc tính DataSource và DataMember thì khung lưới sẽ hiển thị các cột dữ liệu dù chưa có dòng dữ liệu nào hiển thị. Dữ liệu sẽ được đưa vào khung lưới khi chương trình thực thi.

Chọn nút Load Data và đặt thuộc tính Anchor của nó là Bottom, Left. Lúc này giao diện form thiết kế sẽ như hình H. 7.7.

Sau đó, chúng ta cần viết mã để đổ dữ liệu vào khung lưới bằng phương thức Fill như đã viết trong mục 7.5.



Hình 7.7 Cửa sổ form khi thiết kế xong

Tạo thủ tục btnLoad_Click bằng cách double click vào nút LoadData và nhập mã sau:

```
DsInstructors1.Clear()
OleDbDataAdapter1.Fill(DsInstructors1)
```

Nhấn nút Save All để lưu lại các thay đổi và chạy thử chương trình. Chọn F5 để chạy chương trình. Chọn nút Load Data để nạp dữ liệu vào trong khung lưới DataGrid: Ta có thể kéo để thay đổi kích thước form sao cho các thông tin về cơ sở dữ liệu xuất hiện đầy đủ. Chọn nút Close để đóng chương trình.

7.7 Định dạng các ô lưới trong DataGrid

Chúng ta định dạng các thành phần trong DataGrid thông qua thuộc tính của nó lúc thiết kế hay khi thực thi chương trình.. Với ví dụ trên ta làm như sau: từ cửa sổ thiết kế form và mở thuộc tính Properties của khung lưới DataGrid.

- Đặt thuộc tính PreferredColumnWidth là 110 (rộng 110 đơn vị đo Pixel).

- Đặt thuộc tính ColumnHeadersVisible là False. Với thiết lập này thì phần tiêu đề của các cột sẽ không hiển thị.

- Chọn thuộc tính BackColor, chọn màu vàng nhạt hiển thị cho nội dung chuỗi chứa trong ô lưới tạo các dòng xen kẽ nhau.

- Đặt thuộc tính GridLineColor, chọn màu xanh.

Còn rất nhiều thuộc tính khác ta ta có thể lựa chọn.

7.8 Cập nhật cơ sở dữ liệu trở lại bảng

DataSet chỉ tiến hành sao chép bảng của cơ sở dữ liệu, chứ không làm thay đổi nội dung cơ sở dữ liệu cho đến khi có yêu cầu cập nhật bằng phương thức UpDate. Cùng với thuộc tính ReadOnly của DataSet sẽ cho phép có thay đổi hay không với cơ sở dữ liệu.

Từ cửa sổ thiết kế form mở thuộc tính properties của DataGrid và thiết lập giá trị TRUE đối với thuộc tính ReadOnly cho phép có những thay đổi dữ liệu trong khung lưới.

Tạo một nút nữa trên form. Thuộc tính: Name – btnUpdate, Text – "Update". Nút Update sẽ hiển thị khi có những thay đổi trong DataGrid và khi cập nhật trở lại cơ sở dữ liệu khi người dùng tích chuột vào nó.

Tạo thủ tục btnUpdate_Click và nhập mã như sau:

```
Try
OleDbDataAdapter1.Update(DsInstructors1)
Catch ex As Exception
MsgBox(ex.ToString)
End Try
```

Thủ tục này sử dụng phương thức Update của OleDbDataAdapter1 để yêu cầu các thay đổi trong tập DataSet DsInstructors1 trở lại bảng cở dữ liệu.

Khi chạy chương trình, ta thay đổi nội dung một cột nào đó hay có thể thêm một bản ghi nữa và chọn nút Update để cập nhật vào cơ sở dữ liệu. Sau đó lại chọn nút Load Data để xem cơ sở dữ liệu có thay đổi gì không.