فصل چهارم

آرایه‌ها

هدف‌های رفتاری: پس از آموزش این فصل، هنرجو می‌تواند:

- **مفهوم آرایه‌های یک بعدی و چند بعدی را شرح داده و از آن‌ها در برنامه‌های خود استفاده کند؛**
- **روش مرتب‌سازی حبابی را شرح داده و عناصر آرایه را به‌روش حبابی مرتب کند؛**
- **روش‌های جست‌وجوی خطی و دودویی را توضیح داده و از آن برای جست‌وجوی عنصری در بین عناصر آرایه استفاده کند؛**
- **از کنترل‌های ListBox ، ComboBox و Scrolls در برنامه‌های خود استفاده کند؛**
- **مفهوم آرایه‌های کنترلی را شرح دهد؛**
- **آرایه‌های دینامیکی را اعلان کند و به‌کار ببرد.**

۱ــ۴ــ آرایه چیست؟

همان‌طورکه می‌دانید متغیر ظرفی است که یک مقدار را در خود نگه می‌دارد و با ورود مقدار جدید، مقدار قبلی از بین می‌رود. بعضی مواقع در برنامه‌ها نیاز به نگه‌داری چندین مقدار هم‌نوع داریم. آیا باید چندین متغیر تعریف کنیم؟ واضح است که پاسخ منفی است. راه‌حل این مشکل، استفاده از آرایه است. آرایه تعدادی خانه هم‌جوار و هم‌نام در حافظه است که برای نگهداری مقدارهای هم‌نوع استفاده می‌شود.

آرایه متغیری است با چندین خانه که می‌توان در هر خانهٔ آن، یک مقدارنگه‌داری کرد. به هر خانهٔ

آرایه یک **عنصر** گفته می‌شود. عنصر دارای شمارهٔ خاصی است که محل (موقعیت) عنصر در داخل آرایه را برمی‌گرداند. اولین عنصر آرایه معمولاً در موقعیت صفر قرار دارد. آرایه‌ها می‌توانند اندازه‌های مختلفی داشته باشند (شکل ۲ــ۴). یک آرایه ممکن است دارای سه عنصر بوده و دیگری ۳۰ عنصر داشته باشد. حتی ممکن است یک آرایه هیچ عنصری نداشته باشد (امکان اضافه‌کردن عنصر به آن وجود دارد).

شکل ۱ــ۴ــ متغیر، محلی برای نوع دادهٔ خاصی است.

شکل ۲ــ۴ــ آرایه مجموعه ای از متغیرهاست.

۲ــ۴ــ اعلان آرایه

می‌توان یک آرایه را به دو روش اعلان کرد :

- مانند یک متغیر
- با استفاده از کلید واژهٔ To

۱ــ۲ــ۴ــ اعلان آرایه مانند یک متغیر : برای اعلان یک آرایه، از شکل کلی زیر استفاده کنید :

```
Dim |Public|Private ArrayName (Subscript) As DataType
```

در این اعلان :

- Public، Dim و Private کلیدواژه‌های اعلان آرایه و تعیین حوزهٔ عمل آن هستند. دربارهٔ حوزهٔ عمل در فصل توابع توضیح خواهیم داد. اگر از Dim استفاده کنید، آرایه برای روالی که در

آن تعریف شده است قابل شناسایی خواهد بود. کلمات کلیدی Public و Private را در فصل توابع شرح خواهیم داد.

- ArrayName نام آرایه است که از قوانین نامگذاری متغیرها پیروی می‌کند.

نکته

هنگام اعلان یک آرایه، اولین عنصر آرایه معمولاً در مکان صفر قرار دارد. می‌توان با نوشتن Option Base 1 در بخش General کدنویسی، اولین عنصر را در مکان یک قرار داد.

- Subscript مکان آخرین عنصر آرایه را تعیین می‌کند. اولین عنصر آرایه معمولاً در مکان صفر است. بنابراین، اگر آرایه‌ای با اندیس ۶ اعلان کنید، آرایه دارای ۷ مکان و عنصر خواهد بود.
- As کلید واژه‌ای است که اعلان نوع را مشخص می‌کند.
- DataType نوع دادهٔ معتبر در ویژوال بیسیک است مثل Integer یا Double و غیره.

بنابراین، برای اعلان آرایه‌ای از اعداد صحیح با ۵ عنصر، خواهیم نوشت :

```
Dim iMyArray(4) As Integer
```

برای تعیین یک مقدار برای هر عنصر آرایهٔ iMyArray، خواهیم داشت :

```
iMyArray(0) = 9
iMyArray(1) = 342
iMyArray(2) = 2746
iMyArray(3) = 0
iMyArray(4) = 8901
```

برای تغییر مقدار چهارمین عنصر این آرایه از ۰ به ۴۵، به این صورت عمل می‌کنیم :

```
iMyArray(3) = 45
```

به‌عنوان مثال، برای اعلان آرایه‌ای از ۹ رشته (نام گل‌های مختلف)، از کد زیر استفاده خواهیم کرد :

```
Public strMyArray(8) As String

strMyArray(0) = "Maryam"
```

```
strMyArray(1) = "Roze"
strMyArray(2) = "Niloofar"
strMyArray(3) = "Yakh"
strMyArray(4) = "Yas"
strMyArray(5) = "Banafsheh"
strMyArray(6) = "Zanbagh"
strMyArray(7) = "Magnolia"
strMyArray(8) = "Aftabgardan"
```

۲ــ۲ــ۴ــ اعلان آرایه با کلید واژهٔ To : همچنین می‌توان آرایه را با استفاده از کلید واژهٔ To در داخل اندیس‌های آرایه اعلان کرد. به‌عنوان مثال، اگر می‌خواهید آرایه‌ای از ۵ متغیر عدد صحیح ایجاد کنید که اولین عنصر در مکان ۱ و آخرین عنصر در مکان ۵ باشد، از دستور زیر استفاده کنید :

```
Dim iMyArray (1 to 5) As Integer
```

این روش، راه ساده‌ای برای شروع آرایه از مکانی غیر از صفر (حتی مقدار منفی) است.

```
Dim  A (−5,2) As string
```

۳ــ۴ــ تغییر تعداد عناصر آرایه

اگرچه معمولاً تعداد عناصر آرایه هنگام اعلان آن تعیین می‌شود ولی مواردی وجود دارد که هنگام اعلان آرایه تعداد عناصر آن را نمی‌دانیم و یا اینکه می‌خواهیم از این آرایه با اندازه‌های متفاوت در برنامه استفاده کنیم که در این موارد امکان تغییر اندازهٔ آرایه وجود دارد. هنگامی که تعداد عناصر آرایه را تغییر دهید، بُعد آن آرایه تغییر می‌کند. برای انجام این کار، از کلید واژهٔ ReDim به شکل زیر استفاده کنید :

```
ReDim [Preserve] ArrayName (Subscript) As DataType
```

- ReDim کلید واژهٔ ویژوال بیسیک است که مشخص می‌کند آرایه تغییر بُعد می‌یابد.
- Preserve یک کلید واژهٔ اختیاری است که امکان نگه‌داری مقادیر تمام عناصر از قبل تعریف شده در آرایه را فراهم می‌کند. اگر از این کلید واژه استفاده نکنید، مقدار تمام عناصر برای نوع داده‌های

عددی به صفر و برای رشته‌هایی با طول متغیر به رشته‌ای به طول صفر تغییر می‌یابند. رشته‌های با طول ثابت، با Null پر شده و متغیرهای Variant با EMPTY مقداردهی خواهند شد که بسته به نوع عبارت، به صفر یا رشته‌ای به طول صفر تبدیل می‌شوند.

- ArrayName نام آرایه است.
- Subscript اندیس آخرین خانهٔ آرایه است.
- As کلید واژه‌ای است که اعلان نوع را مشخص می‌کند. هنگام تغییر بُعد یک آرایه، کلید واژهٔ As اختیاری است.
- DataType یک نوع دادهٔ معتبر در ویژوال بیسیک است. هنگام تغییر بُعد یک آرایه، DataType اختیاری است و با کلید واژهٔ ReDim نمی‌تواند تغییر یابد.

برای مثال اگر بخواهیم پس از دریافت تعداد دانش‌آموزان یک کلاس نام و نام خانوادگی آن‌ها را در آرایه‌ای وارد کنیم، تعداد عناصر آرایه پس از دریافت تعداد دانش‌آموزان کلاس تعیین می‌شود لذا باید هنگام اعلان آرایه تعداد عناصر را تعیین نکنیم.

```
Dim fnam () As String
Count = InputBox ("please Enter count of students")
ReDim fnam (Count)
```

نکته

اگر آرایه‌ای ایجاد می‌کنید که بعداً می‌خواهید آن را تغییر اندازه دهید، اندازهٔ آن را هنگام اعلان، تعیین نکنید. بنابراین، در عمل نمی‌توان آرایهٔ strMyArray را به‌صورت زیر تغییر اندازه داد:

```
Public strMyArray(8)As string
ReDim Preserve StrMyArray (9)
StrMyArray (9) = "Glaiol"
```

برای ایجاد آرایه‌ای که بعداً تغییر اندازه خواهد داد، باید ابتدا آرایه را بدون هیچ عنصری ایجاد کنید. کد زیر، روش مناسب را برای ایجاد آرایه‌ای که بعداً تغییر اندازه خواهد یافت نشان می‌دهد :

```
Dim strMyArray() As String
ReDim strMyArray (8)
```

```
strMyArray(0) = "Maryam"
strMyArray(1) = "Roze"
strMyArray(2) = "Niloofar"
strMyArray(3) = "Yakh"
strMyArray(4) = "Yas"
strMyArray(5) = "Banafsheh"
strMyArray(6) = "Zanbagh"
strMyArray(7) = "Magnolia"
strMyArray(8) = "Aftabgardan"
ReDim Preserve strMyArray(9)
strMyArray(9) = "Glaiol"
```

توجه داشته باشید که در کد فوق، اولین دستور ReDim از کلید واژهٔ Preserve استفاده نکرده است و دلیل آن هم این است که در آرایه مقداری وجود ندارد که نگه‌داری شود. در دومین دستور ReDim، کلید واژهٔ Preserve خیلی مهم است زیرا عناصر آرایه دارای مقدار هستند که نمی‌خواهیم از دست بروند. اگر از این کلید واژه در این خط استفاده نکنید، آرایهٔ strMyArray دارای مقادیر زیر خواهد بود :

```
strMyArray (0) = ""
strMyArray (1) = ""
strMyArray (2) = ""
strMyArray (3) = ""
strMyArray (4) = ""
strMyArray (5) = ""
strMyArray (6) = ""
strMyArray (7) = ""
strMyArray (8) = ""
strMyArray (9) = "Glaiol"
```

کاربرد حلقه‌ها برای پیمایش آرایه

همان‌طورکه بیان شد، آرایه متغیری با چندین خانه است که در هر خانه می‌توان یک مقدار را نگه داشت. برای مقداردهی، چاپ مقادیر و انجام عملیات روی آرایه، باید دستورات را به تعداد خانه‌های آن تکرار کرد و برای تکرار باید از ساختارهای تکرار استفاده کرد. به دلیل این‌که تعداد دفعات تکرار در آرایه‌ها مشخص است، معمولاً از حلقهٔ For . . . Next استفاده می‌شود که مقدار شمارندهٔ آن از کران پایین تا کران بالای آرایه خواهد بود.

از دستورات LBound و UBound برای تعیین کران پایین و بالای آرایه استفاده می‌شود.

```
LBound (ArrayName)
UBound (ArrayName)
```

مثال ۱ــ۴

فرمی ایجاد کنید که دارای یک کادر متن به نام Textl و یک دکمهٔ فرمان به نام cmdAdd باشد. می‌خواهیم ۱۰ اسم را در کادر متن تایپ کنیم و با کلیک روی دکمهٔ Add آن‌ها را به آرایه اضافه کنیم و بعد از گرفتن دهمین اسم ، کادر متن و دکمهٔ فرمان، غیرقابل رؤیت باشند و تمام اسامی را به همراه طول آن‌ها نمایش دهد.

```
Dim A(9) As String
Dim i As Integer

Private Sub cmdAdd_Click()
        A(i) = Textl. Text  : A مقداردهی عنصر iام آرایه
        Text1. Text = ""  : Text1 پاک‌کردن محتوای
        If i < 9 Then  : آیا تمام ۱۰ عنصر آرایه مقداردهی نشده‌اند؟
            i=i+l
        else
            Textl. Visible = False  ، Textl غیرقابل رؤیت شدن
            cmdAdd. Visible = False  ،cmdAdd غیرقابل رؤیت شدن
            For i = 0 To 9  ، چاپ عناصر آرایه به همراه طول آن‌ها
```

```
            Print A(i), Len(A(i))
        Next
    End If
End Sub
```

شمارهٔ عناصر آرایهٔ A در متغیر سراسری i قرار دارد. لذا مقدار اولیهٔ آن را در رویداد Form–Load تعیین می‌کنیم.

```
Private Sub Form_Load()
    i = 0
End Sub
```

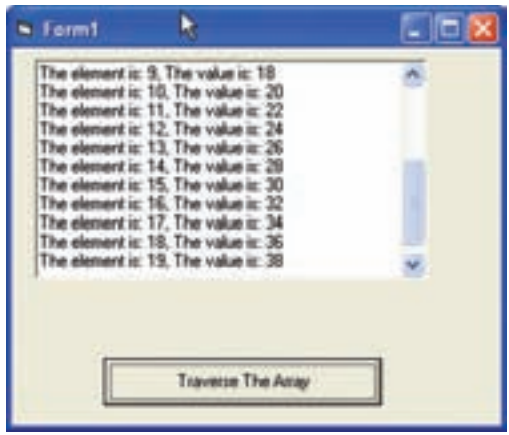
تمرین: برنامه را با کلید f8 خط به خط اجرا کرده و به کمک پنجره watch محتوای متغیر i و آرایهٔ A را مشاهده کنید.

مثال ۲ــ۴

کد زیر، مثال دیگری از کاربرد حلقهٔ For . . . Next برای پیمایش آرایه است. این کد مربوط به رویداد Click دکمه‌ای به نام cmdTraverse است که آرایه‌ای به‌طول ۲۰ عنصر را ایجاد می‌کند. حلقهٔ For . . . Next دوبار استفاده شده است (ابتدا برای تعیین مقادیر عناصر آرایه و سپس پیداکردن مقدار هر عنصر و ایجاد رشته‌ای که مقدار هر عنصر را گزارش می‌کند). به شکل ۳ــ۴ توجه کنید.

```
Private Sub cmdTraverse_Click()
    Dim i As Integer
    Dim iMyArray(19) As Integer
    Dim BeginMsg As String*15
    Dim MidMsg As String*15
    Dim LoopMsg As String*30
    Dim FullMsg As String*500
    For i = 0 To 19  → مقداردهی عناصر آرایه،
        iMyArray(i) = i*2
```

```
        Next i
        BeginMsg = "The element is: "
        MidMsg = ",The value is: "
        For i = 0 To 19      → چاپ مقدار عناصر آرایه ،
            LoopMsg = LoopMsg & BeginMsg & CStr(i)
            LoopMsg = LoopMsg & MidMsg & iMyArray(i)
            FullMsg = FullMsg & LoopMsg & vbCrLf
            LoopMsg =""
        Next i
        txtTraverse. Text = FullMsg
End Sub
```



شکل ۳ــ۴ــ این برنامه هر عنصر آرایه را با دو برابر شمارهٔ آن عنصر مقداردهی می‌کند.

کد فوق ساده است. همان طورکه می‌دانید حلقهٔ For . . . Next یک متغیر شمارنده دارد که در این برنامه مقدار اولیهٔ آن، با کران پایین آرایه و مقدار نهایی آن، با کران بالای آرایه مقداردهی شده است. هنگامی که حلقه را اجرا می‌کنید متغیر شمارنده همیشه شمارهٔ در یک عنصر آرایه خواهد بود.

متغیر LooPMsg مجموع رشته‌هایی است که در یک خط قرار می‌گیرند و متغیر FullMsg مجموع رشته‌های تمام خطوط است که در txtTraverse نمایش داده می‌شود. از عملگر & برای اتصال رشته‌ها و از ثابت VbCrlf برای انتقال مکان‌نما به سر خط بعدی استفاده شده است.

۴ــ۴ــ کنترل‌های ListBox و ComboBox

در بین همهٔ کنترل‌های استاندارد، ListBox (کادرلیست) و ComboBox (کادر ترکیبی) مناسب‌ترین کنترل برای گروه‌بندی و لیست‌بندی اطلاعاتی است که کاربران می‌توانند انتخاب کنند. روش کار با این دو کنترل، یکسان است.

لیست در ویژوال بیسیک، آرایه‌ای از رشته‌هاست که در مشخصهٔ List قرار داده شده‌اند. عملیاتی که می‌توان با این کنترل‌ها انجام داد، اضافه و حذف کردن رشته‌ها از مشخصهٔ List است.

ComboBox در واقع ترکیب کنترل‌های ListBox و TextBox است.

تفاوت این دو کنترل عبارت است از :

- در کادرلیست، لیست رشته‌ها معلوم است ولی در کادر ترکیبی، لیست به‌صورت منوی بازشو است.
- در کادرلیست، امکان چندستونی نوشتن وجود دارد ولی در کادر ترکیبی چنین نیست.

مشخصه‌های مهم این دو کنترل عبارت است از :

- List؛ محتوای این دو کنترل از طریق این مشخصه وارد می‌شود. List نام آرایه‌ای‌است که اطلاعات در آن قرار دارد و از نوع رشته‌ای می‌باشد. شماره اولین عنصر آن صفر است ((List (0)
- ListCount؛ تعداد خطوط لیست را برمی‌گرداند.
- ListIndex؛ شمارهٔ خط جاری لیست را برمی‌گرداند.
- Sorted؛ در صورت True بودن، لیست را مرتب می‌کند.

۱ــ۴ــ۴ــ اضافه کردن رشته‌ها به ListBox یا ComboBox

اضافه‌کردن رشته‌ها به دو روش انجام می‌شود.

۱ــ در زمان طراحی فرم : در مشخصهٔ List مقادیر رشته‌ای را وارد می‌کنیم. دقت کنید که باید پس از هر رشته کلیدهای Ctrl+Enter را فشار دهید تا مکان‌نما به سر خط رفته و آمادهٔ دریافت رشتهٔ بعدی شود.

۲ــ در زمان کدنویسی : از متدی به نام AddItem استفاده می‌شود که فرم کلی آن به‌صورت زیر است :

```
Object. AddItem StringToAdd
```

- Object ، مشخصهٔ Name کنترل کادر لیست یا کادر ترکیبی است.

- AddItem، کلید واژهٔ VB برای متد است.
- StringToAdd ، رشته‌ای است که می‌خواهید به لیست کنترل اضافه کنید.

مثال ۳ــ۴

فرض کنید کنترلی به نام List1 داریم و می‌خواهیم نام ۴ گل را در آن وارد کنیم. برای انجام این کار به‌صورت زیر عمل می‌کنیم :

```
Private Sub Form_Load()
    List1 . AddItem "Roze"
    List1 . AddItem "Maryam"
    List1 . AddItem "Yas"
    List1 . AddItem "Laleh"
End Sub
```

۲ــ۴ــ۴ــ انتخاب عنصرها از لیست : برای آشنایی با چگونگی تعیین مقدار رشتهٔ انتخابی در List کنترل ListBox و ComboBox، نیاز دارید بدانید که List آرایه‌ای از رشته‌هاست و عناصر آن به ترتیب با نام List(0) ، List(1)، List(2) ، ... می‌باشد.

همان‌طورکه می‌دانید برای دسترسی به عنصر دوم آرایه‌ای به نام MyArray، دستوری به‌صورت زیر می‌نویسیم (عنصر اول در خانهٔ صفر قرار دارد) :

```
MyValue = MyArray (l)
```

کنترل‌های ListBox و ComboBox نیز از قالب مشابهی استفاده می‌کنند. بنابراین برای به‌دست‌آوردن دومین رشته در ListBox به نام List1، دستوری به‌صورت زیر خواهیم داشت :

```
secondString = List1 . List (1)
```

برای دسترسی به عنصر انتخاب شده از لیست دو روش وجود دارد :

۱ــ در هر کنترل ListBox و ComboBox، شمارهٔ عنصر انتخاب شده در لیست، در مشخصه‌ای به نام ListIndex قرار می‌گیرد. بنابراین، برای تعیین مقدار رشتهٔ انتخاب‌شده در List1، کد صفحه بعد را به‌کار خواهید برد :

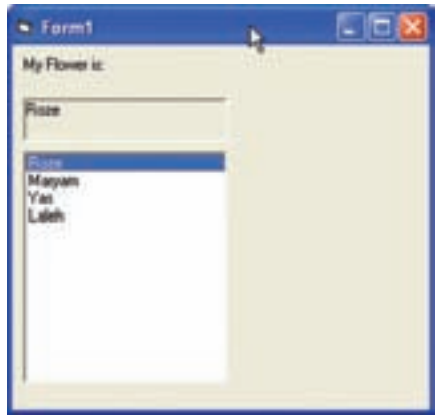
List1 . list (شماره عنصر انتخاب شده)

⇒List1 . List(list1 .listIndex)

شماره عنصر انتخاب شده = List1 . ListIndex

در کد زیر هنگامی که کاربر روی رشته‌ای در ListBox کلیک می‌کند، کد اجراشده و مقدار انتخاب‌شده در لیست را به مشخصهٔ Caption برچسبی به نام LabeL1 انتساب می‌دهد. شکل ۴ــ۴ رویدادهایی را که هنگام انتخاب یک رشته رخ می‌دهند نشان می‌دهد.

```
Private Sub List_Click()
    LabeL1 . Caption = List1 . List (List1 . ListIndex)
End Sub
```



شکل ۴ــ۴ــ می‌توان برای رویدادهای Click، MouseUp یا MouseDown یک ListBox کد نوشت تا مقدار انتخاب‌شده را به‌دست آورد.

۲ــ روش سریع‌تر برای به‌دست‌آوردن مقدار یک رشتهٔ انتخاب‌شده در کنترل‌های ListBox یا ComboBox، استفاده از مشخصهٔ Text است. به‌عنوان مثال، می‌توان از کد زیر برای یک ListBox استفاده کرد :

```
Dim strMyStr As String
StrMyStr = List1 . Text
```

برای یک ComboBox، از کد زیر استفاده کنید :

```
Dim StrMyStr As String
StrMyStr = Combo1 . Text
```

۳ــ۴ــ۴ــ حذف عنصرها از لیست : با استفاده از متد RemoveItem می‌توان رشته‌ای را از لیست ListBox و ComboBox حذف کرد :

Object. RemoveItem Index

- Object، مشخصهٔ Name کنترل ListBox یا ComboBox است.
- RemoveItem کلیدواژهٔ ویژوال بیسیک برای متدی است که عناصر را از لیست حذف می‌کند.
- Index محل رشته‌ای است که می‌خواهید از لیست حذف کنید. برای حذف عنصر انتخاب‌شده از لیست، از مشخصهٔ ListIndex استفاده کنید.

شکل ۵ــ۴ بهینه سازی شدهٔ برنامهٔ شکل ۴ــ۴ را نشان می‌دهد. یک دکمه برای حذف رشته‌ای که کاربر از ListBox انتخاب می‌کند، اضافه شده است. کد زیر، چگونگی استفاده از متد RemoveItem را نشان می‌دهد.

```
Private Sub cmdRemove_Click()
    List1. RemoveItem (List1. ListIndex)
End Sub
```

شکل ۵ــ۴ــ دکمهٔ فرمان روشی برای حذف عنصری از لیست است.

۴ــ۴ــ۴ــ پاک‌کردن لیست : درصورتی که می‌خواهید تمام رشته‌های موجود در ListBox یا ComboBox را حذف کنید، از متد Clear استفاده کنید :

```
Object.Clear
```

بنابراین، برای پاک‌کردن لیست شکل ۷ــ۴ خواهیم نوشت :

```
List1.Clear
```

۵ــ۴ــ۴ــ آشنایی با شیوه‌های ComboBox : کنترل‌های ListBox و ComboBox دارای وجوه مشترک زیادی هستند ولی هر کدام دارای محدودیت کاربرد می‌باشند. یک ListBox فضای بیشتری را نسبت به ComboBox اشغال می‌کند و نمی‌توان در آن یک دادهٔ خارج از لیست را انتخاب یا وارد کرد. ComboBox قابلیت انعطاف بیشتری از خود نشان می‌دهد و از فضای فرم به صورت کارآمد استفاده می‌کند.

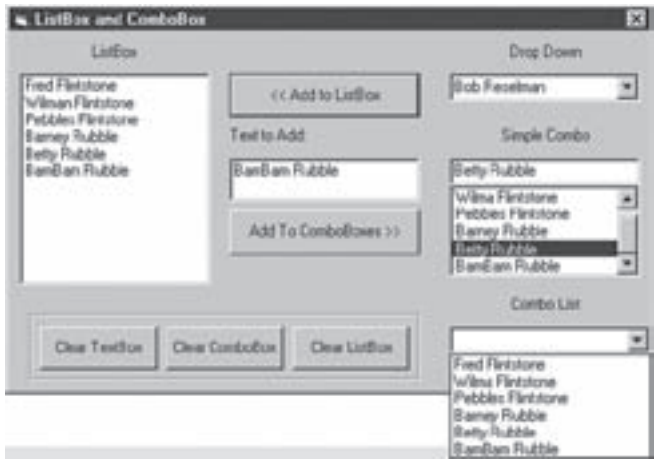
مشخصهٔ Style کنترل ComboBox امکان تغییر مشخصه‌های عملیاتی و ظاهر کنترل را فراهم می‌کند. جدول ۱ــ۴ این شیوه ها را شرح می‌دهد و شکل ۶ــ۴ شیوه‌های ComboBox اعمال شده به فرم را نشان می‌دهد.

نکته

هنگامی که یک ComboBox را با شیوهٔ 1–Simple Combo به فرم اضافه می‌کنید، ComboBox تغییر شکل یافته و به‌صورت ComboBox مشاهده نمی‌شود. مشخصهٔ Height را افزایش دهید تا ListBox آن نمایش یابد.

جدول ۱ــ۴ــ مقادیر مربوط به مشخصهٔ Style کنترل ComboBox

تنظیم	شرح
0 - Drop – Down Combo	یک لیست بازشو. کاربران می‌توانند داده‌های جدیدی را به ComboBox وارد کنند.
1–Simple Combo	ترکیبی از TextBox و ListBox که بازشو نیست. کاربران می‌توانند داده‌ها را از ListBox انتخاب کرده یا دادهٔ جدیدی را در TextBox وارد کنند. اندازهٔ این نوع ComboBox شامل بخش‌های ویرایش و لیست است.
2– Drop – Down List	یک لیست بازشوی فقط خواندنی که کاربران فقط می‌توانند داده‌ها را انتخاب کنند. کاربران نمی‌توانند داده‌ها را وارد کنترل کنند.



شکل ۶ــ۴ــ توجه کنید که با شیوهٔ ComboBox بازشو، می‌توان داده‌های جدیدی را در زمان اجرا به کنترل اضافه کرد.

متداول‌ترین شیوه‌های ComboBox شیوه‌های ۰ و ۲ هستند. همان‌طورکه قبلاً نیز بیان شد، هنگامی که می‌خواهید به کاربران امکان اضافه‌کردن داده‌های جدیدی را که در لیست نیستند بدهید از شیوهٔ ComboBox بازشو (۰) استفاده کنید. درصورتی که می‌خواهید کاربران فقط امکان انتخاب داشته باشند، از شیوهٔ لیست بازشو (۲) استفاده کنید.

مثال ۴ــ۴

فرمی به‌صورت زیر ایجاد کنید :

شکل ۷ــ۴

یک کادر لیست و یک کادر ترکیبی به نام‌های Lstflower و Cboprice به همراه دو کادرمتن و یک دکمهٔ فرمان وجود دارد. با کلیک روی دکمهٔ Go متن موجود در هر کادر متن، در کادر لیست مربوطه قرار می‌گیرد. کد مربوط به این مثال به‌صورت زیر خواهد بود :

```
Private Sub cmdGo_Click()
    If txtprice. Text <>"" Then
        Lstprice.AddItem txtprice. Text
        txtprice.Text = ""
    End If
    If txtflower. Text <>"" Then
        cboflower. AddItem txtflower. Text
        txtflower. Text = ""
    End If
End Sub
```

تمرین : مثال ۴ـ۴ را طوری تغییر دهید که:

الف) اگر نامی تکراری وارد شد، به لیست اضافه نکند.

ب) دکمه‌ای برای پاک‌کردن هر دو لیست اضافه کند.

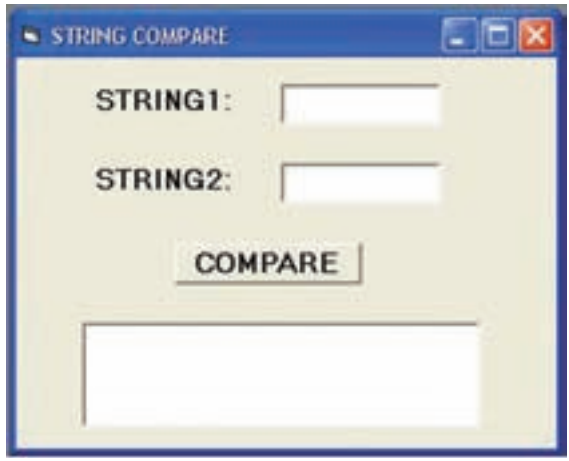
ج) دکمه‌ای اضافه کند که با کلیک آن قیمت گلی تعیین شود که نام آن در txtflowers آمده است. قیمت گل در کنترل Labled نمایش داده شود (راهنمایی: گل و قیمت آن در لیست‌ها دارای شمارهٔ یکسان هستند.)

مثال ۵ــ۴

برنامهٔ زیر از تابع StrComp برای مقایسهٔ دو رشته استفاده می‌کند. رابط کاربر برای این برنامه شامل چندین برچسب (Label)، دو کادر متن (TextBox) و یک دکمهٔ فرمان (Command Button) به همراه یک لیست است. برای مقایسهٔ دو رشته، ابتدا رشته‌ها را در درون دو کادر متن تایپ کرده و سپس بر روی دکمهٔ Compare کلیک نمایید. برنامه با استفاده از تابع StrComp به مقایسهٔ دو رشته

می‌پردازد و نتیجه را به‌صورت عبارتی به لیست پایین پنجره اضافه می‌کند.

```
Dim C As Integer
Private Sub cmdComp_Click()
    C = StrComp(Text1.Text,Text2.Text)
    Select case C
        Case 0 : List1. AddItem   Text1. Text + "=" + Text2. Text
        Case 1 : List1. AddItem   Text1. Text + ">" + Text2. Text
        Case −1 : List1. AddItem   Text1. Text + "<" + Text2. Text
    End Select
End Sub
```



شکل ۸ــ۴

از تابع StrComp() برای مقایسهٔ دو رشته استفاده می‌شود و شکل کلی آن به صورت زیر است :

Variable Name = StrComp(String1, String2 [,Compare])

● **Variable Name** : متغیری است که خروجی این تابع در آن قرار می‌گیرد و نوع آن عددی صحیح است.

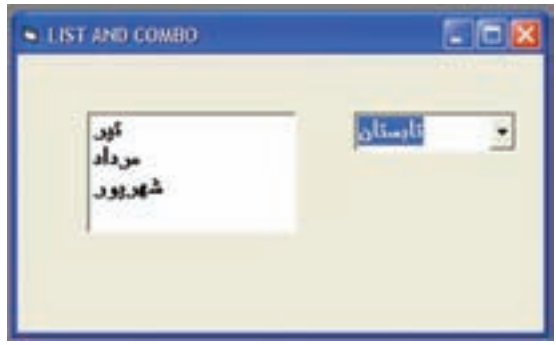
● **String 1, String 2** : دو رشته‌ای هستند که با هم مقایسه خواهند شد. اگر دو رشته با یکدیگر برابر باشند، تابع مقدار صفر را برمی‌گرداند و اگر رشتهٔ اول کوچک‌تر از رشتهٔ دوم باشد، تابع

مقدار ۱- و در صورت بزرگ‌تر بودن رشتهٔ اول از رشتهٔ دوم، مقدار ۱ برگردانده می‌شود (مقایسهٔ رشته‌ها براساس کداسکی انجام می‌شود).

● **Compare** : پارامتری است اختیاری که اگر مقدارش ۱ باشد، بدین معنی است که بین حروف کوچک و بزرگ، تفاوتی قائل نشود ولی اگر صفر باشد، بین این دو نوع حروف تفاوت قائل می‌شود.

مثال ۶ــ۴

فرمی به‌صورت زیر طراحی کنید که با انتخاب نام فصل از یک ComboBox نام ماه‌های آن فصل را در یک ListBox نمایش دهد.



شکل ۹ــ۴

نام کادر ترکیبی را Combo1 و نام کادر لیست را List1 قرار دهید.

قبل از اضافه کردن ماه‌های فصل انتخاب شده به List1 باید محتوای List1 پاک شود (دستور List1.Clear)

```
Private Sub Combol_Click()
    Select case Combol. Text
        Case "بهار"
            Listl. Clear
            Listl. AddItem "فروردین"
            Listl. AddItem "اردیبهشت"
            Listl. AddItem "خرداد"
        Case "تابستان"
```

```
            Listl. Clear
            Listl. AddItem "تیر"
            Listl. AddItem "مرداد"
            Listl. AddItem "شهریور"
        Case "پاییز"
            Listl. Clear
            Listl. AddItem "مهر"
            Listl. AddItem "آبان"
            Listl. AddItem "آذر"
        Case "زمستان"
            Listl. Clear
            Listl. AddItem "دی"
            Listl. AddItem "بهمن"
            Listl. AddItem "اسفند"
    End Select
End Sub
```

مثال ۷ــ۴

در این مثال می‌خواهیم شهر مقصد، محل صندلی و غذای دلخواه مسافران هواپیما را انتخاب کنیم. یک کادر لیست، دو کادر ترکیبی (کومبو)، سه برچسب و دو دکمهٔ فرمان روی فرم قرار داده و ظاهر آن را مطابق شکل صفحهٔ بعد تنظیم کنید :

کد برنامه به‌صورت زیر خواهد بود :

```
Private Sub Form_Load()
    lstCities.Clear
    lstCities.AddItem  "شیراز"
    lstCities. AddItem  "اصفهان"
    lstCities.AddItem  "تبریز"
```

شکل ۱۰ــ۴

```
lstCities.AddItem "ارومیه"
lstCities.AddItem "اهواز"
lstCities.AddItem "یزد"
lstCities.AddItem "بوشهر"
lstCities.AddItem "زنجان"
lstCities.AddItem "رشت"
lstCities.AddItem "ساری"
lstCities.AddItem "گرگان"
lstCities.AddItem "مشهد"
lstCities.AddItem "زاهدان"
lstCities.AddItem "کرمان"
lstCities.ListIndex = 0          قرار دادن اولین عنصر لیست در حالت انتخاب

cboSeat.AddItem "ردیف اول"
cboSeat.AddItem "وسط"
cboSeat.AddItem "کنار پنجره"
cboSeat.ListIndex = 0
```

```
cboMeal.AddItem "جوجه"
cboMeal.AddItem "برگ"
cboMeal.AddItem "سبزیجات"
cboMeal.AddItem "میوه"

cboMeal. Text = "فرقی نمی کند"
End Sub


Private Sub cmdAssign_Click()
    Dim Message As String
    Message = "مقصد"+ lstCities. Text+vbCr
    Message = Message + "محل صندلی"+ cboSeat. Text+vbCr
    Message = Message + "غذای دلخواه"+ cboMeal. Text+vbCr
    MsgBox Message,vbOKOnly +vbInformation,"سفارش شما"
End Sub


Private Sub cmdExit_Click()
    End
End Sub
```

تمرین : برنامه‌ای بنویسید که الف) نام و نام خانوادگی و نمرات برنامه‌سازی دانش‌آموزان را از طریق دو کادر متن دریافت کرده و در دو کادر لیست قرار دهد. ب) امکان حذف نام دانش‌آموز را به برنامه اضافه کنید و در صورت حذف دانش‌آموز نمره دانش‌آموز نیز حذف شود.

۵ــ۴ــ آرایهٔ کنترلی

در ویژوال بیسیک می‌توان آرایه‌هایی از انواع دادهٔ مختلف ایجاد کرد. همچنین می‌توان آرایه‌ای از کنترل‌ها نیز ایجاد کرد. آرایه‌های کنترلی یک ویژگی ذاتی در ویژوال بیسیک هستند که توانایی و کارایی زبان برنامه‌نویسی را افزایش می‌دهند. می‌توان از یک روال رویداد استفاده کرد تا عمل مشترکی

را روی همهٔ عناصر آرایهٔ کنترلی انجام دهد. همچنین می‌توان از آرایه‌های کنترلی برای اضافه کردن و حذف پویای کنترل‌ها و فرم‌ها در زمان اجرا استفاده کرد.

تمام کنترل‌های ذاتی را می‌توان به صورت آرایه‌های کنترلی به کار برد. همهٔ این کنترل‌ها دارای مشخصهٔ Index هستند که برای شناسایی کنترل خاصی در آرایه مورد استفاده قرار می‌گیرند.

۱ــ۵ــ۴ــ ایجاد آرایهٔ کنترلی در زمان طراحی : اغلب آرایه‌های کنترلی که ایجاد خواهید کرد، در زمان طراحی ساخته خواهند شد. همزمان با این‌که کنترل‌ها را روی فرم اضافه می‌کنید، نیاز خواهید داشت تا بعضی از آن‌ها را در آرایه‌های کنترلی گروه‌بندی کنید. مثال زیر چگونگی انجام این کار را بیان می‌کند.

مثال ۸ــ۴

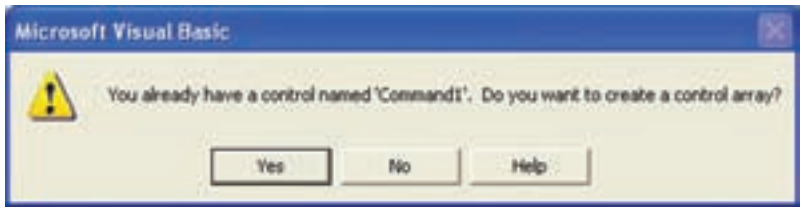
ایجاد آرایهٔ کنترلی از CommandButtons

۱ــ پروژهٔ جدیدی را شروع کنید و نام فرم پیش فرض را به frmMain تغییر دهید.

۲ــ یک دکمهٔ فرمان به مرکز فرم frmMain اضافه کنید و نام آن را cmdMyButton قرار دهید. مشخصهٔ Caption این دکمهٔ فرمان را به Action تغییر دهید.

۳ــ دکمهٔ فرمان را انتخاب کنید. از منوی Edit گزینهٔ Copy را انتخاب کنید تا دکمهٔ فرمان به حافظهٔ Clipboard کپی شود.

۴ــ از منوی Edit گزینهٔ Paste را انتخاب کنید. یک کادر محاوره‌ای ظاهر شده و از شما سؤال می‌کند که آیا می‌خواهید یک آرایهٔ کنترلی ایجاد کنید؟ روی Yes کلیک کنید تا آرایهٔ کنترلی ایجاد شود (شکل ۱۱ــ۴).

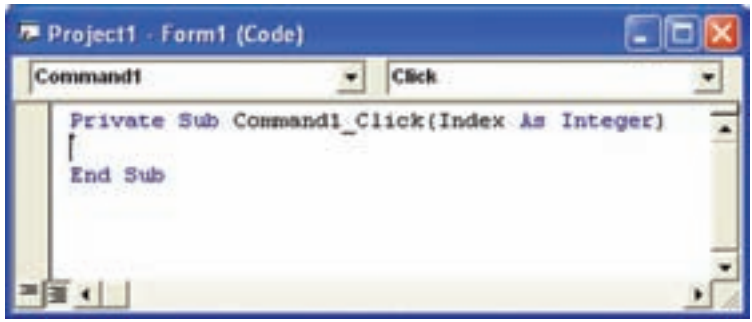


شکل ۱۱ــ۴ــ کادر محاوره‌ای تأیید ایجاد آرایهٔ کنترلی

بعد از ایجاد آرایهٔ کنترلی، اگر به پنجرهٔ Properties رجوع کنید و لیست بازشوی Object را نمایش دهید، مشاهده خواهید کرد که دو دکمهٔ فرمان با نام یکسان cmdMyButton وجود دارد که هرکدام دارای اندیس خاصی هستند (شکل ۱۲ــ۴).

شکل ۱۲ــ۴ــ هنگامی که کنترل بخشی از آرایهٔ کنترلی باشد، برای دسترسی به آن باید همیشه به اندیس آن اشاره کرد.

روی دکمهٔ فرمان دوبار کلیک کنید تا روال رویداد Click را مشاهده کنید. این روال رویداد دارای آرگومان Index خواهد بود (شکل ۱۳ــ۴). این آرگومان یک عدد صحیح است که اندیس کنترل را مشخص می‌کند. به دلیل این‌که همهٔ کنترل‌های آرایهٔ کنترلی نام یکسانی دارند، تفاوت بین آن‌ها با مقدار Index در روال رویداد مشخص می‌شود. صفر (۰) اولین کنترل، ۱ دومین کنترل و ۲ سومین کنترل و الی آخر را مشخص می‌کند.



شکل ۱۳ــ۴ــ هر زمانی که رویدادی برای کنترلی از یک آرایهٔ کنترلی رخ دهد، ویژوال بیسیک آرگومان Index را مقداردهی می‌کند.

کد زیر نشان می‌دهد که روی کدام دکمهٔ فرمان از آرایهٔ کنترلی کلیک شده است. شکل ۱۴ــ۴ اجرای این کد را نشان می‌دهد.

```
Private Sub cmdMyButton_Click(Index As Integer)
    Me.Caption = "You clicked button # " & Index & "."
End Sub
```

شکل ۱۴ــ۴ــ بعد از کلیک کاربر روی دکمهٔ سمت راست، عنوان فرم مطابق آن تغییر می‌یابد.

همانطورکه در مثال بالا دیدید روال رویدادها برای تمام عناصر آرایه کنترلی مشترک است (یک روال رویداد کلیک برای دو دکمه در مثال بالا) و برای تشخیص اینکه رویداد رخ داده مربوط به کدام عنصر آرایهٔ کنترلی است روال رویدادهای آرایه‌های کنترلی دارای آرگومان ورودی Index است که مشخص‌کنندهٔ شمارهٔ عنصری از آرایهٔ کنترلی است که رویداد برای آن رخ داده است.

۲ــ۵ــ۴ــ اضافه‌کردن عنصرها به آرایهٔ کنترلی در زمان اجرا : ایجاد آرایهٔ کنترلی در زمان طراحی هنگامی مفید خواهد بود که تعداد کنترل‌های مورد نیاز برای آرایه را بدانید. ولی اگر تعداد کنترل‌هایی را که در زمان اجرای برنامه نیاز خواهید داشت نمی‌دانید، چه کاری باید انجام دهید؟ این مشکل را می‌توان با استفاده از دستور Load برای افزودن کنترل‌ها به آرایهٔ کنترلی در زمان اجرا، حل کرد.

مثال ۹ــ۴

فرض کنید یک دکمهٔ فرمان به عنوان اولین عنصر آرایهٔ کنترلی داریم و می‌خواهیم در هنگام اجرا عنصر بعدی آن را ایجاد کنیم. مراحل کار به صورت زیر خواهد بود :

۱ــ پروژهٔ جدیدی را شروع کنید. نام فرم را به frmDArray تغییر دهید.

۲ــ یک دکمهٔ فرمان به گوشهٔ سمت چپ بالای فرم اضافه کنید و نام آن را cmdCtrlArray قرار دهید.

۳ــ در پنجرهٔ Properties، مقدار مشخصهٔ Index دکمهٔ فرمان را با صفر (۰) مقداردهی کنید.

۴ــ مشخصهٔ Caption دکمهٔ فرمان را به Button#0 تغییر دهید.

۵ــ کد زیر را به رویداد Form_Load اضافه کنید.

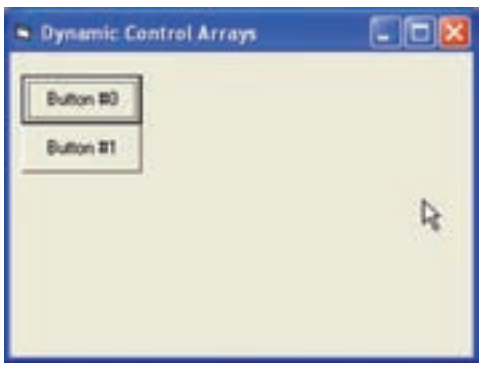
```
Private Sub Form_Load()
    Load cmdCtrlArray(1)
    cmdCtrlArray(1).Left = cmdCtrlArray(0). Left
    cmdCtrlArray(1).Top = cmdCtrlArray(0). Top + cmdCtrlArray (0).Height
    cmdCtrlArray(1).Caption = " Button # 1"
    cmdCtrlArray(1).Visible = True
End Sub
```

۶ــ کد را ذخیره کرده و اجرا کنید.

بعد از اجرای کد، برنامه یک دکمهٔ فرمان جدیدی را ایجاد می‌کند و آن را زیر اولین دکمهٔ فرمان قرار می‌دهد (شکل ۱۵ــ۴).

نکته

تمام عناصر جدیدی که از آرایهٔ کنترلی در زمان اجرا ایجاد می‌کنید دارای مشخصهٔ Visible با مقدار False هستند. هنگامی که کنترل‌های جدیدی را در زمان اجرا ایجاد می‌کنید، فراموش نکنید خطی در کد قرار دهید که مشخصهٔ Visible را با True مقداردهی کند. در غیر این صورت نمی‌توانید کنترل را مشاهده کنید.



شکل ۱۵ــ۴ــ استفاده از دستور Load دکمهٔ فرمان دیگری را روی فرم قرار می‌دهد. بقیهٔ کد محل دقیق کنترل را تعیین می‌کند.

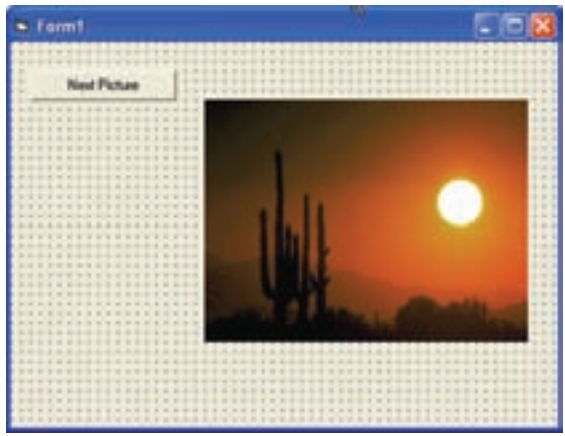
کنترل‌های جدیدی که ایجاد می‌کنید دقیقاً تکرار اولین عنصر آرایهٔ کنترلی هستند. مقادیر همهٔ مشخصه‌ها به جز Index و Visible یکسان هستند. بنابراین، هنگامی که کنترل جدیدی را ایجاد می‌کنید، این کنترل روی اولین کنترل آرایه قرار می‌گیرد و باید آن را به محل دیگری انتقال دهید که انجام این کار در زمان اجرا با تغییر مشخصهٔ Left و Top ممکن است.

همان‌طور که در مثال قبل مشاهده کردید، مزیت استفاده از آرایه‌های کنترلی، توانایی داشتن مجری رویداد مشترک است.

مثال ۱۰ــ۴

آرایه‌ای ۵ عنصری از تصاویر داریم که می‌خواهیم با هر بار کلیک روی دکمهٔ Next Picture تصویر بعدی نمایش داده شود.

۱ــ پروژه‌ای به صورت زیر ایجاد کنید (شکل ۱۶ــ۴). (این پروژه دارای ۵ عنصر تصویر است که روی هم قرار گرفته‌اند)



شکل ۱۶ــ۴

۲ــ کد زیر را برای فرم وارد کنید :

```
Dim cnt As Byte,i As Byte
Private Sub Cmdpic_Click()
    imgl(cnt). Visible = False
```

```
If cnt < 4 Then cnt = cnt + 1 else  Cnt = 0
img1(cnt). Visible = True
End Sub
```

در Form_load ویژگی visible تمام تصاویر به جز تصویر اول (img1(0)) را False می‌کنیم.

```
Private Sub Form_Load()
For i=1 to 4
Img1(i). Visible = False
NEXT i
cnt = 0
End Sub
```

۳ــ پروژه را اجرا کنید.

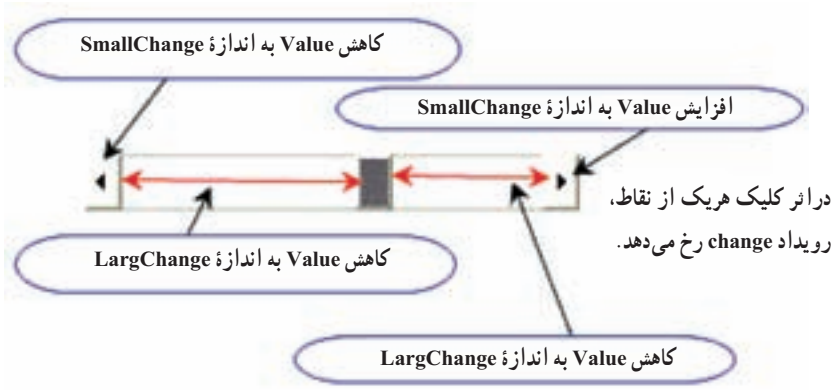
۶ــ۴ــ کاربرد کنترل‌های Scroll Bar

کنترل‌های نوار لغزان استاندارد (HscrollBar و VscrollBar) امکان تغییر داده‌ها یا جابه‌جایی در محدوده‌ای از مقادیر را با کلیک‌کردن روی دکمه های Up و Down یا با جابه‌جایی دکمهٔ لغزان، فراهم می‌کند. کنترل‌های نوار لغزان دارای چند مشخصهٔ خاص هستند که در جدول ۲ــ۴ با آن‌ها آشنا خواهید شد.

جدول ۲ــ۴ــ مشخصه‌های خاص کنترل‌های HscrollBar و VscrollBar

مشخصه	شرح
Min	کمترین مقدار کنترل را هنگامی که دکمهٔ لغزان در بالا یا سمت چپ نوار لغزان قرار دارد، تعیین می‌کند. مقدار پیش فرض، صفر است ولی اعداد منفی را نیز می‌توان به کار برد.
Max	بیشترین مقدار ممکن کنترل را هنگامی که دکمهٔ لغزان در پایین یا سمت راست نوار لغزان قرار دارد، تعیین می‌کند. مقدار پیش فرض، ۳۲۷۶۷ است.
Value	موقعیت دکمهٔ لغزان را نسبت به مشخصه‌های Min و Max تعیین می‌کند.
LargChange	مقدار تغییر مشخصهٔ Value را هنگامی که کاربران بین دکمهٔ لغزان و فلش کلیک می‌کنند تعیین می‌کند.
SmallChange	مقدار تغییر مشخصهٔ Value را هنگامی که کاربران روی فلش‌ها کلیک می‌کنند تعیین می‌کند.

کنترل‌های نوار لغزان دارای دو رویداد Change و Scroll هستند. هرگاه مقدار Value تغییر کند بر روی فلش‌ها و یا فضای بین فلش‌ها کلیک کنیم و یا دکمه لغزان جابه‌جا شود رویداد Change رخ می‌دهد. هنگام حرکت‌دادن دکمهٔ لغزان چندین رویداد Scroll رخ می‌دهد و به محض متوقف‌شدن دکمه یک رویداد Change رخ می‌دهد.



شکل ۱۷ــ۴ــ میزان تغییرات value دراثر کلیک نقاط مختلف در ScrollBar

مثال ۱۱ــ۴

پروژه‌ای ایجاد کنید که با تغییر نوارهای لغزان افقی و عمودی، مقدار آن‌ها را در کادر متن‌های جداگانه نمایش دهد.

۱ــ پروژهٔ جدیدی به نام SmpScroll. vbp ایجاد کنید. نام فرم را frmMain قرار دهید.

۲ــ کنترل‌های VscrollBar و HscrollBar را به فرم اضافه کنید. نام کنترل VscrollBar را به vscrNS و کنترل HscrollBar را به hscrWE تغییر دهید.

۳ــ دو کنترل به کادر متن اضافه کنید؛ نام یکی را txtNS و دیگری را txtWE قرار دهید. مشخصهٔ Text هر دو کنترل را یک رشتهٔ خالی قرار دهید (شکل ۱۸ــ۴).

۴ــ مشخصه‌های کنترل‌های VscrollBar و HscrollBar را مطابق جدول ۳ــ۴ مقداردهی کنید.

جدول ۳ــ۴ــ تنظیم مشخصه‌های کنترل‌های ScrollBar

مشخصه	مقدار
Min	0
Max	20
SmallChange	1
LargeChange	2

۵ــ کد زیر را در بخش General فرم frmMain اضافه کنید :

نمایش مقدار مشخصهٔ Value نوار لغزان افقی در کادر متن txtWE

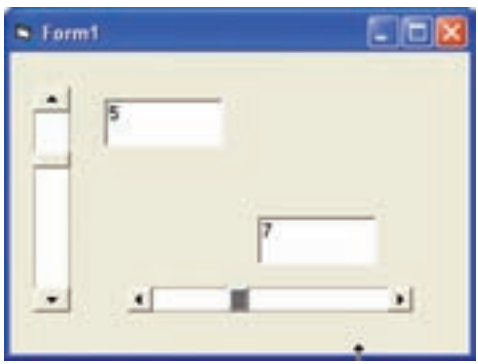
```
Private Sub hscrWE_Change()
    txtWE.Text = CStr(hscrWE.Value)
End Sub
```

نمایش مقدار مشخصهٔ Value نوار لغزان عمودی در کادر متن txtNS

```
Private Sub vscrNS_Change()
    txtNS.Text = CStr(vscrNS.Value)
End Sub
```

۶ــ پروژه را ذخیره کرده و اجرا کنید.

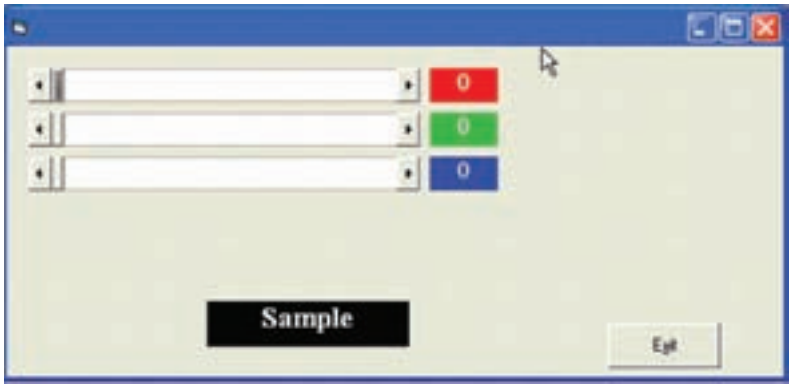
هنگامی که پروژه را اجرا کردید، با کلیک روی فلش‌های کنترل‌های HscrollBar و VscrollBar، مقدار کادر متن مربوطه یک واحد تغییر می‌کند (براساس مقدار مشخصهٔ SmallChange). ولی اگر بین فلش و دکمهٔ لغزان کلیک کنید، مقدار کادر متن مربوطه، دو واحد تغییر می‌کند (براساس مقدار مشخصهٔ LargeChange) (شکل ۱۸ــ۴).



شکل ۱۸ــ۴ــ به دلیل این‌که مشخصهٔ Max با ۲۰ مقداردهی شده است، هنگام جابه‌جایی نوار لغزان، مقادیر نشان داده شده در کادرهای متن تا ۲۰ می‌رسند.

مثال ۱۲ــ۴

فرمی به صورت شکل ۱۹ــ۴ ایجاد کنید. با تغییر نوارهای لغزان مربوط به سه رنگ قرمز، آبی و سبز می‌توان رنگ برچسب را تغییر داد.



شکل ۱۹ــ۴

مقدار Value هر نوار لغزان روی برچسبی که کنار آن قرار دارد نشان داده می‌شود. و رنگ زمینهٔ برچسب Label1 براساس مقدار Value هر سه نوار لغزان تعیین می‌شود.

کد مربوطه را به فرم اضافه کنید :

```
Dim r,g,b As Byte
Private Sub Command1_Click()
    End
End Sub

Private Sub Form_Load()
    LabeL1 . BackColor = RGB(0, 0, 0)
End Sub

Private Sub HScroll l_Change()
    r = HScroll1 . Value
```

```
        g = HScroll2.Value
        b = HScroll3.Value
        Lblr.Caption = r
        Lblg.Caption = g
        Lblb.Caption = b
        LabeL1.BackColor = RGB(r,g,b)
End Sub
Private Sub HScroll1_Scroll()
        r = HScroll1.Value
        g = HScroll2.Value
        b = HScroll3.Value
        Lblr.Caption = r
        Lblg.Caption = g
        Lblb.Caption = b
        LabeL1. BackColor = RGB(r, g, b)
End Sub

Private Sub HScro112_Change()
        r = HScroll1.Value
        g = HScroll2.Value
        b = HScroll3.Value
        Lblr.Caption = r
        Lblg.Caption = g
        Lblb.Caption = b
        LabeL1. BackColor = RGB(r, g, b)
End Sub
```

```
Private Sub HScroll2_Scroll()
        r = HScroll1 . Value
        g = HScroll2 .  Value
        b = HScroll3 .  Value
        Lblr . Caption = r
        Lblg . Caption = g
        Lblb . Caption = b
        LabeL1 .  BackColor = RGB(r, g, b)
End Sub

Private Sub HScroll3_Change()
        r = HScroll1 . Value
        g = HScroll2 .  Value
        b = HScroll3 .  Value
        Lblr . Caption = r
        Lblg . Caption = g
        Lblb . Caption = b
        LabeL1 .  BackColor = RGB(r, g, b)
End Sub

Private Sub HScroll3_Scroll()
        r = HScroll1 . Value
        g = HScroll2 . Value
        b = HScroll3 . Value
        Lblr . Caption = r
        Lblg . Caption = g
        Lblb . Caption = b
```

```
    Labell. BackColor = RGB(r, g, b)
End Sub
```

تمرین ۱: در مثال ۱۲ـ۴ کنترل‌های نوار لغزان و برچسب‌های کنار آن را به‌صورت آرایه‌های کنترلی طراحی کنید.

تمرین ۲: مثال ۱۲ـ۴ را به گونه‌ای تغییر دهید که رنگ زمینه از طریق یک کادر لیست انتخاب شود و شدت تمام رنگ‌ها با یک Scrollbar تعیین شود.

۷ـ۴ـ مرتب‌سازی عنصرهای آرایه‌ها

مرتب‌کردن داده‌ها، یکی از عملیات مهم در برنامه‌های کاربردی است. الگوریتم‌های متعددی برای مرتب‌کردن n عنصر وجود دارد که از جهات مختلف قابل دسته‌بندی و مقایسه هستند. براساس خصوصیات این الگوریتم‌ها، هر کدام ممکن است در مسائل خاصی، عملکرد بهتری داشته باشند.

در عناصر مرتب‌شونده، بخشی از هر عنصر که مقایسه و مرتب‌سازی براساس آن بخش انجام می‌شود، «کلید» نام دارد. در یک الگوریتم مرتب‌سازی همواره دو عمل «مقایسه» و «تعویض» انجام می‌گیرد.

شرکت مخابرات برحسب نام خانوادگی افراد، و در صورت تکراری‌بودن نام، شماره تلفن‌ها را در کتابچهٔ راهنمای تلفن مرتب می‌کند که با این روش، جست‌وجوی شمارهٔ تلفن فرد مورد نظر آسان‌تر خواهد شد.

یکی از روش‌های مرتب‌سازی عناصر آرایه روش حبابی (Bubble Sort) است. در این روش عناصر آرایه دو به دو مقایسه می‌شوند و عنصر کوچکتر قبل از عنصر بزرگتر قرار می‌گیرد. فرض کنید که A لیستی از n عدد باشد. می‌خواهیم عنصار A را به‌صورت افزایشی (صعودی) مرتب کنیم یعنی

$$A(0) < A(1) < A(2) < \ldots < A(n)$$

برای شروع، عناصر A را دو به دو مقایسه می‌کنیم. A(0) با A(1) اگر A(0) بزرگتر از A(1) بود، محتوای آن‌ها را جابجا می‌کنیم. سپس A(1) با A(2) مقایسه می‌شود و به همین ترتیب A(2) با A(3)، A(3) با A(4)، ... تا A(n−1) با A(n). تا اینجا یک مرحله از مقایسه‌ها انجام شده و باید این مراحل را تکرار کنیم تا تمام عناصر مرتب شود. حداکثر تعداد این مراحل برای مرتب‌شدن کل آرایه (n−1) می‌باشد.

برای جابه‌جایی محتوای دو خانه از متغیر سومی باید استفاده کنیم.

در تکه کد زیر برای جابجاکردن محتوای دو خانه از متغیر inttemp استفاده شده است.

```
If intarray A(intComp) > intarray A (intComp+1) then
    inttemp = intarray A(intComp)
```

```
intarray A (intComp) = int array A(intComp+1)
intarray A(intComp+1) = inttemp
End If
```

مقایسهٔ دو عنصر کنارهم و در صورت لزوم جابجاکردن محتوای آن‌ها

عمل مقایسهٔ دو به دو در اولین مرحله به تعداد n−۱ بار انجام می‌شود و در هر مرحله یکی از عناصر مرتب می‌شود لذا از تعداد مقایسه‌های دو به دو یکی کم می‌شود، به گونه‌ای که در مرحله اول تعداد مقایسه‌ها n−۱، در مرحله دوم تعداد مقایسه‌ها n−۲، ... می‌باشد. لذا تعداد مقایسه‌های هر مرحله به شمارهٔ آن مرحله بستگی دارد و اگر شماره مرحله را در متغیر intpass قرار دهیم تعداد مقایسه‌ها در هر مرحله n−intpass می‌باشد که n تعداد عناصر آرایه (کران بالای آرایه) است لذا تکه کد زیر عمل مقایسه در هر مرحله را نشان می‌دهد.

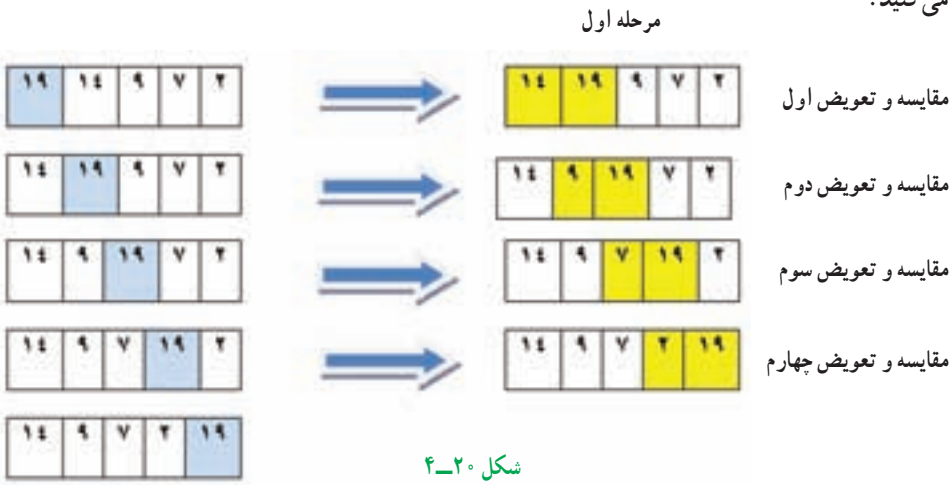
```
For intComp = LBound (intarray A) To UBound (intarray A) – intPass
```

برای مثال آرایه A به‌صورت :

A(1)	A(2)	A(3)	A(4)	A(5)
۱۸	۱۴	۹	۷	۲

را در نظر بگیرید.

در شکل ۲۰ــ۴ نتیجه اجرای قطعه کد قبل برای intpass=1 را روی آرایه A مشاهده می‌کنید.



شکل ۲۰ــ۴

نتیجه آرایه پس از مرحله اول و مرتب‌شدن عنصر آخر آرایه (عدد ۱۹)

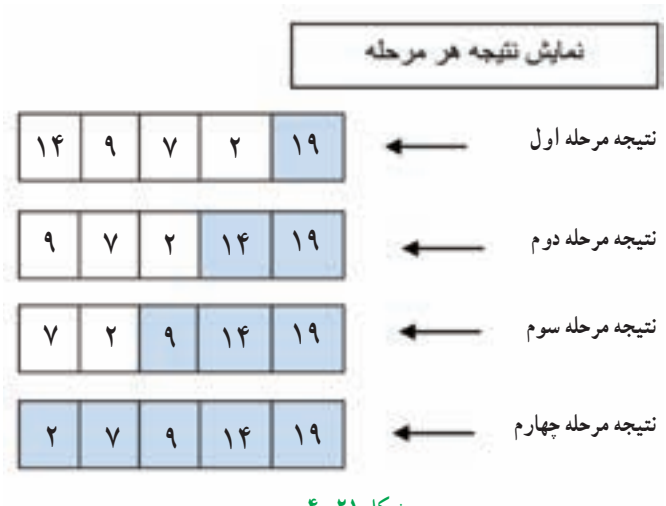
برای مرتب‌شدن تمام عناصر آرایه، این مراحل حداکثر n−۱ بار باید تکرار شود لذا تکه کد قبل باید درون یک حلقه For قرار گیرد که متغیر این حلقه شماره مرحله را نگهداری می‌کند.

۱ـ For intpass = LBound (intarray A) To UBound (intaray A)

تکه کد مربوط به اجرای هر مرحله

Next

خروجی آرایه A پس از هر مرحله در شکل ۲۱ـ۴ آمده است.



شکل ۲۱ـ۴

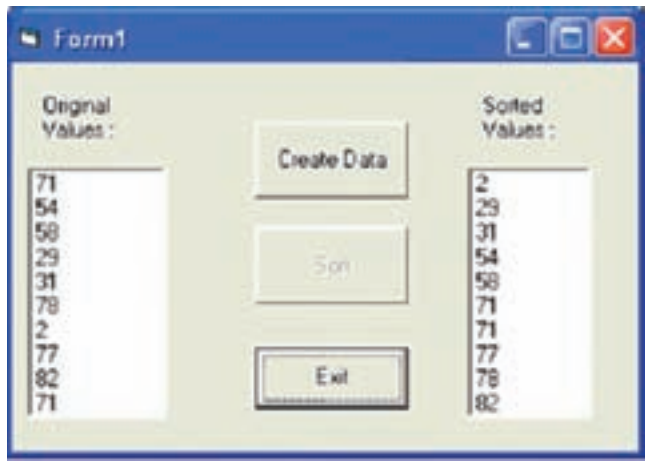
همانطورکه در جدول مشاهده می‌کنید در هر مرحله یکی از عناصر آرایه مرتب می‌شود. در مرحله اول A(۵) (۱۸)، در مرحله دوم A(۴) (۱۴)، در مرحله سوم A(۳) (۹) و در مرحله چهارم A(۱) و A(۲) (۲،۷) با هم مرتب می‌شوند. از آن‌جا که دو عنصر A(۱) و A(۲) با هم مرتب می‌شوند تعداد مراحل از تعداد عناصر آرایه یکی کمتر است.

مثال ۱۳ـ۴

پروژه‌ای ایجاد کنید که با کلیک روی دکمهٔ Create Data ده (۱۰) عدد صحیح تصادفی در یک کادر لیست قرار بگیرند، سپس با کلیک روی دکمهٔ Sort این اعداد مرتب شده و در کادر لیست دوم کپی شوند.

برنامهٔ زیر، مقادیر آرایهٔ mArray را به صورت صعودی مرتب می‌کند. تکنیکی که ما در این برنامه استفاده کرده‌ایم، Bubble Sort (مرتب‌سازی حبابی) است. روال Bubble Sort عمل مرتب‌سازی را انجام می‌دهد.

شکل ۲۲ــ۴



شکل ۲۳ــ۴

```
Option Explicit
Option Base 1
```

اندیس اولین عنصر آرایه یک می‌باشد

```
Dim intarrayA(10) As Integer
Private Sub cmdGenerate_Click ()
    Dim inti As Integer
    lstOriginal.Clear        خالی کردن کادر لیست اولیه
    For inti  = LBound(intarrayA) To UBound(intarrayA)
    مقداردهی عناصر آرایه با اعداد تصادفی کوچکتر از ۱۰۰
        intarrayA(inti) = Int (Rnd*100)
        lstOriginal.AddItem intarrayA(inti)
    قراردادن عناصر آرایه در کادر لیست اولیه
    Next
    lstSorted.Clear                    lstSorted خالی کردن کادر لیست
    cmdSort. Enabled = True
End Sub

Private Sub cmdSort_Click()
    Dim intpass As Integer, intcomp As Integer
    Dim inttemp As Integer
    Dim inti As Integer
    IstSorted.Clear        حلقهٔ شمارنده مراحل
    For intpass = LBound(intarrayA) To UBound (intarrayA) – 1
    حلقهٔ مقایسهٔ دو به دوی اعداد
        For intcomp = LBound (intarrayA) To UBound(intarrayA) – intpass
    جابه‌جا کردن عناصر آرایه در صورت لزوم
            If  intarrayA(intcomp) > intarrayA(intcomp + 1) Then
                inttemp = intarrayA(intcomp)
                intarrayA(intcomp) = intarrayA(intcomp + 1)
                intarrayA(intcomp + l) = inttemp
```

```
                End If
            Next
        Next
                        اضافه‌کردن عناصر آرایهٔ مرتب‌شده به کادر لیست LSt sorted
        For inti = LBound(intarrayA) To UBound (intarrayA)
            lstSorted.AddItem intarrayA(inti)
        Next
End Sub

Private Sub cmdExit_Click()
        End
End Sub
Private Sub Form_Load()
        Randomize
        cmdSort.Enabled = False
End Sub
```

۸ــ۴ــ جست‌وجو

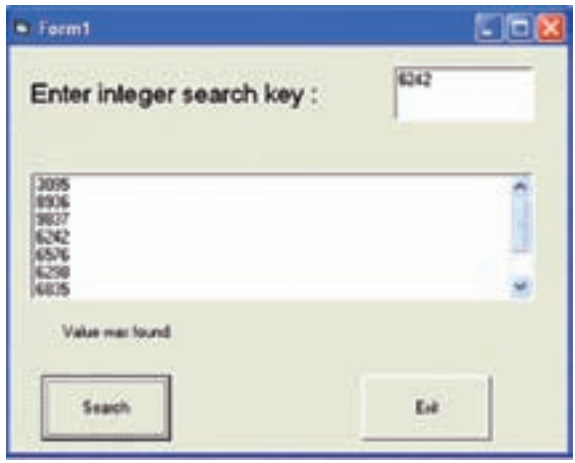
در اغلب موارد می‌خواهید با مقادیر زیادی از داده‌ها که در آرایه‌ها ذخیره شده‌اند، کار کنید و ممکن است تعیین مقداری که در آرایه وجود دارد، ضروری باشد. عملیاتی که به منظور یافتن عضوی در آرایه صورت می‌گیرد به نام جست‌وجو (Searching) معروف است.

۱ــ۸ــ۴ــ جست‌وجوی خطی : جست‌وجوی خطی در مورد آرایه‌های کوچک یا آرایه‌های مرتب نشده خوب عمل می‌کند ولی در مقابل آرایه‌های بزرگ و مرتب کارایی بهینه ندارد.

در جست‌وجوی خطی، هر عضو آرایه با مقداری که کلید جست‌وجو (search key) نامیده می‌شود مقایسه می‌شود. در این روش جست‌وجو، به طور میانگین، برنامه کلید جست‌وجو را با نصف اعضای آرایه مورد مقایسه قرار می‌دهد. برای تعیین این که مقدار مورد نظر در آرایه وجود ندارد، برنامه باید کلید جست‌وجو را با تمام اعضای آرایه مقایسه کند.

مثال ۱۴ــ۴

برنامه‌ای بنویسید که ۱۰ عدد حداکثر۴رقمی تولید کند و در صورت یافتن کلید جست‌وجو شمارهٔ عنصر و در صورت نیافتن پیغام مناسب نمایش دهد.



شکل ۲۴ــ۴ــ جست‌وجوی خطی

```
Option Explicit
Option Base 1
Dim intarrayA(10) As Integer

Private Sub cmdExit_Click()
    End
End Sub

Private Sub cmdSearch_Click()
    Dim intkey As Integer
    Dim inti As Integer
    Dim intindex As Integer
```

```
    Dim blnfount As Boolean
    lblresult.Caption = " "
    intkey = Val(txtkey)
    blnfound = False
```

مقایسهٔ عناصر آرایه با کلید جست‌وجو

```
    For inti = LBound(intarrayA)To UBound(intarrayA)
        If intarrayA(inti) = intkey Then
            blnfound = True
            intindex = inti
            Exit For
        End If
    Next
    If blnfound = True Then
        lblResult = "Item=" & intindex
    Else
        lblResult = "Not Found"
    End If
End Sub

Private Sub Form_Load()
    Dim inti As Integer
    Call Randomize
    Call lstData.Clear
    lblResult = " "
    txtkey = " "
```

*** تولید اعداد تصادفی و اضافه‌کردن به کادر لیست LstData ***

```
    For inti = LBound(intarrayA) To UBound(intarrayA)
        intarrayA(inti) = Int(Rnd*10000)
        lstData. AddItem intarrayA (inti)
    Next
End sub
```

۲ــ۸ــ۴ــ جست‌وجوی دودویی : این روش جست‌وجو برای آرایه‌های بزرگ مناسب است. به دلیل این‌که در این نوع آرایه‌ها با روش جست‌وجوی خطی، تعداد دفعات جست‌وجو زیاد خواهد بود که سبب اتلاف زمان می‌شود.

روش جست‌وجوی دودویی، روی آرایه‌های مرتب قابل اجراست. بنابراین، در صورتی که آرایه موردنظر نامرتب باشد، ابتدا باید آن را مرتب کنید.

فرض کنید آرایه به صورت صعودی مرتب شده است. برای جست‌وجوی دودویی، ابتدا عنصر میانی (وسط) آرایه را به‌دست آورده و با کلید جست‌وجو مقایسه می‌کنیم. در این لحظه سه حالت ممکن است رخ دهد :

- عنصر میانی با کلید جست‌وجو مساوی است. در این حالت جست‌وجو موفق بوده و خاتمه می‌یابد و اندیس عنصر میانی به عنوان مکان عنصر مورد جست‌وجو برگردانده می‌شود.
- عنصر میانی از کلید جست‌وجو بزرگ‌تر است. بنابراین از تمام عناصر قبل از آن نیز بزرگ‌تر خواهد بود و می‌توان جست‌وجو را در نیمهٔ بالایی (سمت راست عنصر میانی) ادامه داد.
- عنصر میانی از کلید جست‌وجو کوچک‌تر است. بنابراین از تمام عناصر بعد از آن نیز کوچک‌تر خواهد بود و می‌توان جست‌وجو را در نیمهٔ پایینی (سمت چپ عنصر میانی) ادامه داد.

به مثال زیر توجه کنید :

در لیست ۲۴،۷،۸،۱۹ نیمه پایینی ۱۹ ۸ ↓ ۷ ۴ ۲ نیمه بالایی

۱ــ جست‌وجوی عدد ۷ : عنصر میانی مساوی کلید جست‌وجو است (۷=۷).

۲ــ جست‌وجوی عدد ۴ : عنصر میانی از کلید جست‌وجو بزرگتر است پس در نیمه بالایی به دنبال عدد می گردیم (۴ < ۷).

۳ــ جست‌وجوی عدد ۱۹ : عنصر میانی از کلید جست‌وجو کوچکتر است لذا در نیمه پایینی به دنبال عدد می‌گردیم. (۱۹ > ۷).

در این روش، جست‌وجو تا زمانی ادامه می‌یابد که کران پایین بزرگ‌تر از کران بالا باشد و اگر

چنین حالتی رخ دهد، می‌گوییم جست‌وجو ناموفق بوده است.

به عنوان مثال، فرض کنید می‌خواهیم محل نامی را در دفترچهٔ تلفن (یا واژه‌ای را در فرهنگ لغت) پیدا کنیم. واضح است که برای انجام چنین کاری از جست‌وجوی خطی استفاده نمی‌کنیم و به جای آن وسط دفترچه را باز کرده و تعیین می‌کنیم که جست‌وجو در کدام نیمهٔ دفترچه است. سپس نیمهٔ موردنظری را از وسط بازکرده و تعیین می‌کنیم که اسم موردنظر در کدام یک چهارم می‌باشد، این کار را روی یک چهارم مورد نظر نیز انجام داده و تعیین می‌کنیم که اسم در کدام یک هشتم می‌باشد و الی آخر. این کار را تا زمانی ادامه می‌دهیم که عنصر موردنظر را پیدا کنیم. این الگوریتم سریع‌تر از الگوریتم جست‌وجوی خطی است، زیرا تعداد عناصر در حال کم شدن است.

مثال ۱۵ــ۴

فرض کنید آرایه‌ای از اعداد مرتب به‌صورت زیر وجود دارد و می‌خواهیم از روش جست‌وجوی دودویی برای پیدا کردن محل یک عنصر استفاده کنیم.

List: 2,5,7,14,26

Search Key: 14

Beg = 0, End = 4, Mid = Int [(Beg+End)/2] = [(0+4)/2] =2

List(2) = 7

Beg کران پایین، End کران بالا و Mid وسط آرایه است. بعد از به‌دست آوردن عنصر میانی آرایه (یعنی عدد ۷) آن را با کلید جست‌وجو (عدد ۱۴) مقایسه می‌کنیم. چون ۱۴ از ۷ بزرگ‌تر است، پس از اعداد قبل از آن نیز (اعداد ۲ و ۵) بزرگ‌تر است. بنابراین جست‌وجو را باید در نیمهٔ دوم آرایه انجام دهیم. برای انجام این کار، کران پایین آرایه به‌صورت زیر تغییر می‌کند :

Beg = Mid + 1 = 2 + 1 = 3, End = 4, Mid = [(3+4)/2] = 3

List(3) = 14

با تغییر کران پایین، دوباره عنصر میانی لیست جدید را به‌دست می‌آوریم و با کلید جست‌وجو مقایسه می‌کنیم. در این حالت کلید جست‌وجو با عنصر میانی برابر است. بنابراین می‌توان گفت که جست‌وجو موفق بوده و مکان کلید در آرایه ۳ است.

تمرین : در مثال فوق، جست‌وجوی عدد ۳ را به صورت گام به گام انجام دهید.

این روش جست‌وجو فقط مخصوص لیست‌هایی است که از قبل مرتب شده باشند. در این روش به‌جای آن‌که عمل جست‌وجو و مقایسه از اولین عنصر لیست آغاز شود ابتدا عنصر وسط لیست با کلید مقایسه می‌گردد و درنتیجه سه حالت اتفاق می‌افتد :

۱ــ کلید مساوی با مقدار عنصر وسط لیست است.

۲ــ کلید بزرگ‌تر از مقدار عنصر وسط لیست است.

۳ــ کلید کوچک‌تر از مقدارعنصر وسط لیست است.

در حالت اول کار جست‌وجو تمام شده است.

ولی اگر لیست صعودی باشد در حالت دوم محال است که کلید در نیمهٔ بالایی لیست باشد. پس نیمهٔ پایین لیست را جست‌وجو می‌کنیم.

حالت سوّم مشابه حالت دوّم است با این تفاوت که کلید نمی‌تواند در نیمه پایینی لیست باشد و باید نیمه بالای لیست را جستجو کنیم.

در این روش با یک عمل مقایسه نصف عناصر لیست را کنار زده‌ایم. درواقع در یک لیست ۱۰۰،۰۰۰ عنصری با یک عمل مقایسه ۵۰،۰۰۰ عنصر لیست را کنار می‌گذاریم.

با مقایسهٔ بعدی مطمئن می‌شویم کلید در ۲۵،۰۰۰ عنصر دیگر لیست وجود ندارد. یعنی هر بار لیست نصف می‌شود مثلاً اگر لیستی ۸۰۰،۰۰۰ عنصر داشته باشد، حداکثر ۲۰ مقایسه کافی است تا کلید پیدا شود.

مثال ۱۶ــ۴

برنامه‌ای بنویسید که ۱۰ عدد تصادفی دو رقمی تولید کند سپس هر عددی را وارد کردیم موقعیت آن عدد را در لیست جست‌وجو کند و جواب بدهد. (جست‌وجوی دودویی)

شکل ۲۵ــ۴

```
Option Explicit

Private Sub cmdCreate _ Click()
        Dim inti As integer
        Dim intnum As Integer
        Lstnumber.Clear
         txtkey = " "
        '*** ۱۰≤intNum<۱۰۰ تولید ۱۰ عدد تصادفی دورقمی ***
             For inti =1 To 10
           Intnum = Int(Rnd * 90) + 10
           Lstnumber.AddItem intnum
       Next
End Sub
Private Sub cmdExit_Click()
        End
End Sub
Private Sub cmdsearch_Click()
        Dim intkey As Integer
        Dim intLow As Integer
        Dim intHigh As Integer
        Dim intMid As Integer
        Dim intPosition As Integer
        Dim blnfound As Integer
        lblPosition.Caption = " "
        intkey = Val (txtkey)
        blnfound = False
        intLow = 0:  intHigh = 9
```

```
        Do While intLow <= intHigh And Not blnfound
            intMid = (intLow + intHigh) \ 2
            If intkey < Val (Istnumber.List (intMid) Then
                intHigh = intMid − 1
            ElseIf intkey > Val (Istnumber.List (intMid)) Then
                intLow = intMid + 1
            EIse
            blnfound = True
            intPosition = intMid
            End If
        Loop
        If blnfound = True Then
                lblPosition = "Item = " & intPosition
        Else
                lblPosition = "Not Found"
        End If
End Sub

Private Sub From_Load()
    Randomize Timer
End Sub
```

۹ــ۴ــ آرایه‌های چندبعدی

آرایه‌هایی که تا اینجای فصل ایجاد کردیم، همان‌طورکه در شکل ۲ــ۴ مشاهده کردید، آرایه‌های یک‌بعدی (مجموعه‌ای یک‌سطری از متغیرها) بودند. در ویژوال بیسیک می‌توان آرایه‌هایی ایجاد کرد که حداکثر ۶۰ بعد داشته باشند. معمولاً آرایه‌های دوبعدی برای اغلب پروژه‌های برنامه‌نویسی کافی خواهد بود و به‌ندرت نیاز به ایجاد آرایه‌های سه‌بعدی و بیشتر خواهید داشت.

آرایهٔ دوبعدی را شبیه صفحهٔ بازی دوز در نظر بگیرید. (مجموعه‌ای از ستون‌ها و سطرها که از تداخل آن‌ها خانه‌هایی به‌وجود می‌آیند). هر خانه دارای موقعیت تعریف‌شده‌ای به‌صورت شمارهٔ سطر و شمارهٔ ستون است.

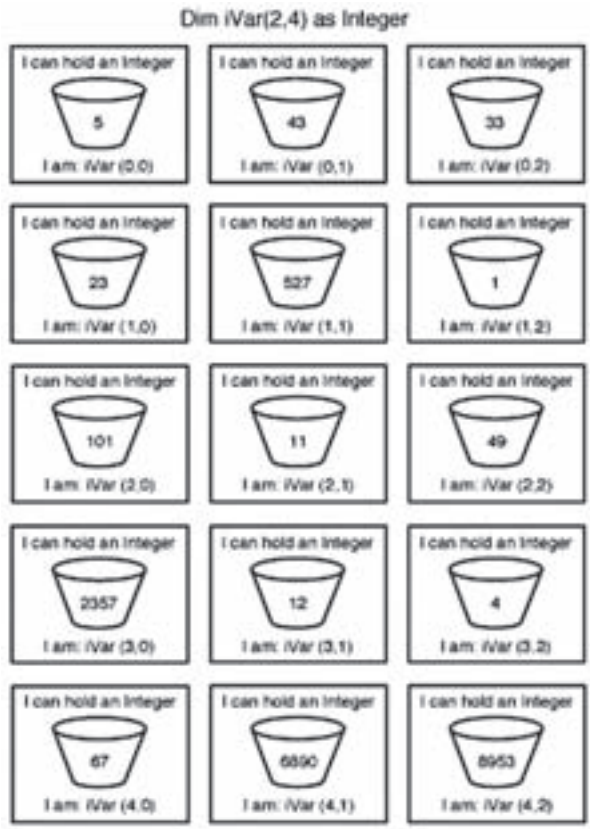
توجه داشته باشید که هر عنصر براساس مختصات سطر و ستون تعریف می‌شود. به‌عنوان مثال، عنصر iVar(0,0) دارای مقدار ۵ و عنصر iVar (2,2) دارای مقدار ۴۹ است (شکل ۲۶ــ۴).

برای ایجاد آرایهٔ دوبعدی، از شکل کلی زیر استفاده کنید :

```
Dim | Public | Private ArrayName (SubscriptOfRows, SubscriptOfCols) As Data Type
```

با اغلب کلید واژه‌های این دستور آشنا هستید. دو مورد جدید، عبارتند از :

- SubscriptOfCols ــ اندیس آخرین ستون آرایه است.
- SubscriptOfRows ــ اندیس آخرین سطر آرایه است.



شکل ۲۶ــ۴ــ آرایهٔ دوبعدی را به‌صورت مجموعه‌ای از ظرف‌ها که در ستون‌ها و سطرها سازماندهی شده‌اند، تصور کنید.

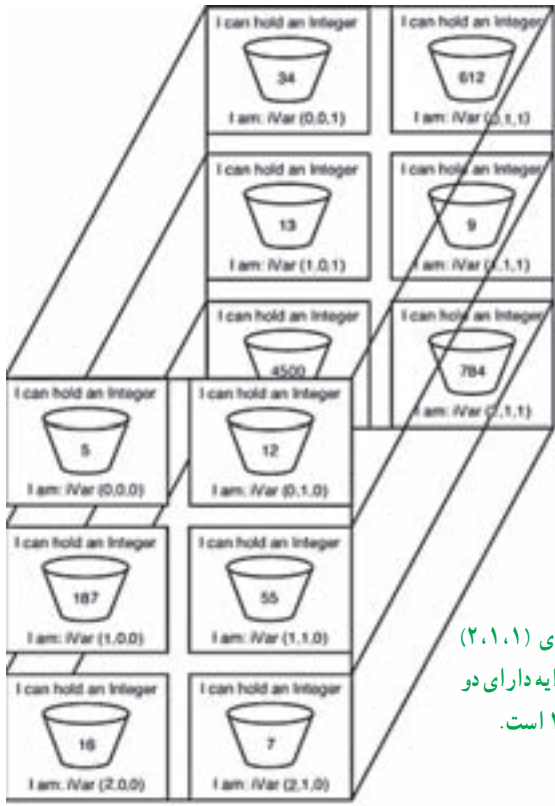
بنابراین، برای اعلان آرایهٔ شکل ۲۶ــ۴ می‌توان نوشت :

```
Dim iVar (4,2) As Integer
```

اگر یک آرایهٔ دوبعدی را به‌صورت مستطیل تصور کنید، می‌توانید آرایهٔ سه‌بعدی را به شکل یک مکعب مستطیل در نظر بگیرید. برای اعلان آرایهٔ سه‌بعدی از اعداد صحیح، دستور زیر را وارد کنید :

```
Dim iVar (2,1,1) As Integer
```

شکل ۲۷ــ۴ این آرایهٔ سه‌بعدی را نشان می‌دهد.



شکل ۲۷ــ۴ــ آرایهٔ سه‌بعدی (۲،۱،۱) دارای ۱۲ عنصر است. این آرایه دارای دو ستون و سه سطر به عمق ۲ است.

(تعداد صفحات عمق × تعداد سطر × تعداد ستون) = تعداد عناصر آرایهٔ سه‌بعدی

در آرایهٔ iVar مقدار تعیین‌شده برای هر عنصر که در شکل ۲۷ــ۴ نشان داده شده است، به‌صورت زیر خواهد بود :

```
iVar(0,0,0) = 5
iVar(1,0,0) = 187
```

```
iVar (2,0,0) = 16
iVar (0,1,0) = 12
iVar (1,1,0) = 55
iVar (2,1,0) = 7
iVar (0,0,1) = 34
iVar (1,0,1) = 13
iVar (2,0,1) = 4500
iVar (0,1,1) = 612
iVar (1,1,1) = 9
iVar (2,1,1) = 784
```

مشابه آرایهٔ یک‌بعدی، می‌توان از کلید واژهٔ To برای اعلان محدودهٔ اندیس‌های هر بعد در آرایه‌های چندبعدی نیز استفاده کرد. پس دستور زیر :

```
Dim dMyArray (1 To 5, 3 To 8, 3 To 5) As Double
```

یک آرایهٔ سه‌بعدی از مقادیر اعشاری خواهد بود که دارای ۵ سطر، ۶ ستون و ۳ صفحهٔ عمق خواهد بود.

همچنین می‌توان از کلید واژهٔ ReDim برای تغییر اندازهٔ آرایهٔ چندبعدی استفاده کرد. اگر از کلید واژهٔ Preserve استفاده کنید، فقط آخرین بعد آرایهٔ چندبعدی می‌تواند تغییر اندازه یابد و تعداد ابعاد نمی‌توانند تغییر کنند.

خودآزمایی

۱ــ ۱۰ دانش‌آموز و ۳ نمره از هر یک را از ورودی دریافت کنید، سپس نام هر دانش‌آموز را به همراه معدل سه نمرهٔ او نمایش دهید. در پایان نام دانش‌آموزانی که بالاترین و پایین‌ترین معدل را دارند اعلام کنید.

۲ــ برنامه‌ای بنویسید که عناصر یک ماتریس ۴×۴ را دریافت کند. سپس این ماتریس را روی فرم نشان دهد و با پیغامی اعلان کند که آیا عناصر قطر اصلی ماتریس با هم برابر هستند یا نه.

۳ــ برنامه‌ای بنویسید که عناصر یک ماتریس ۴×۴ را دریافت کند. سپس عددی را گرفته، آن را در ماتریس جست‌وجو کند و تعداد تکرار آن را نمایش دهد.

۴ــ برنامه‌ای بنویسید که عناصر یک ماتریس ۳×۳ را دریافت کند. سپس این ماتریس را نشان داده و اعلام کند بالا مثلثی است یا پایین مثلثی.

۵ــ برنامه‌ای بنویسید که نام و نمره ریاضی ۸ نفر را گرفته و اعلام کند رتبه دوم کیست.