

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 به روش 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.

۱-۴- آرایه چیست؟

همان‌طور که 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 خانه آن، یک مقدار نگه‌داری کرد. به هر خانه

آرایه یک عنصر گفته می‌شود. عنصر دارای شماره 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 ۲-۴). یک آرایه ممکن است دارای سه عنصر بوده و دیگری ۳۰ عنصر داشته باشد. حتی ممکن است یک آرایه هیچ عنصری نداشته باشد (امکان اضافه کردن عنصر به آن وجود دارد).

I can hold a Double



dVar as Double

شکل ۱-۴- متغیر، محلی برای نوع داده خاصی است.



شکل ۲-۴- آرایه مجموعه‌ای از متغیرهاست.

۲-۴-۲- اعلان آرایه

می‌توان یک آرایه را به دو روش اعلان کرد:

- مانند یک متغیر

- با استفاده از کلید واژه To

۱-۲-۴- 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:

- Dim, Public, Private، کلیدواژه‌های اعلان آرایه و تعیین حوزه عمل آن هستند. درباره

حوزه 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.

● **Data Type** نوع داده 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 آرایه **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"

```

۲-۲-۴ اعلان آرایه با کلید واژه **To** : همچنین می‌توان آرایه را با استفاده از کلید واژه **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
```

۳-۴ 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 اندازه آرایه وجود دارد. هنگامی که تعداد عناصر آرایه را تغییر دهید، بعد آن آرایه تغییر می‌کند. برای انجام این کار، از کلید واژه **ReDim** به شکل زیر استفاده کنید :

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

- **ReDim** کلید واژه ویروال بیسیک است که مشخص می‌کند آرایه تغییر بُعد می‌یابد.
- **Preserve** یک کلید واژه 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 خانه آرایه است.
- `As` کلید واژه‌ای است که اعلان نوع را مشخص می‌کند. هنگام تغییر بُعد یک آرایه، کلید واژه `As` اختیاری است.

● `DataType` یک نوع داده معتبر در ویژوال بیسیک است. هنگام تغییر بُعد یک آرایه، `DataType` اختیاری است و با کلید واژه `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)
```

توجه

اگر آرایه‌ای ایجاد می‌کنید که بعداً می‌خواهید آن را تغییر اندازه دهید، اندازه آن را هنگام اعلان، تعیین نکنید. بنابراین، در عمل نمی‌توان آرایه `strMyArray` را به صورت 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 واژه Preserve استفاده نکرده است و دلیل آن هم این است که در آرایه مقدراری وجود ندارد که نگه‌داری شود. در دومین دستور ReDim، کلید واژه 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، آرایه 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

همان‌طور که 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 این که 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 ۱-۴

فرمی ایجاد کنید که دارای یک کادر متن به نام Text1 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) Text1.Text : A مقداردهی عنصر i ام آرایه A

Text1.Text = "" پاک کردن محتوای Text1

If i < 9 Then ۱۰ عنصر آرایه مقداردهی نشده‌اند؟

i i 1

else

Text1.Visible = False ، غیرقابل رؤیت شدن Text1

cmdAdd.Visible = False ، غیرقابل رؤیت شدن cmdAdd

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 ۲-۴

کد زیر، مثال دیگری از کاربرد حلقهٔ Next ... For برای پیمایش آرایه است. این کد مربوط به رویداد Click دکمه‌ای به نام cmdTraverse است که آرایه‌ای به طول ۲۰ عنصر را ایجاد می‌کند. حلقهٔ Next ... For دوبار استفاده شده است (ابتدا برای تعیین مقادیر عناصر آرایه و سپس پیدا کردن مقدار هر عنصر و ایجاد رشته‌ای که مقدار هر عنصر را گزارش می‌کند). به شکل ۳-۴ 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،
```

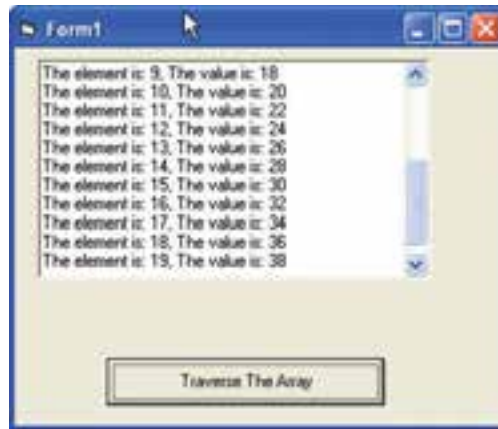
```
        iMyArray(i) = i*2
```



```

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 ۳-۴- این برنامه هر عنصر آرایه را با دو برابر شماره آن عنصر مقداردهی می‌کند.

کد فوق ساده است. همان طور که 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 مکان نما به سر خط بعدی استفاده شده است.

۴-۴ کنترل‌های ListBox و ComboBox

در بین همه 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 مشخصه List قرار داده شده‌اند. عملیاتی که می‌توان با این کنترل‌ها انجام داد، اضافه و حذف کردن رشته‌ها از مشخصه List است.

ComboBox در واقع ترکیب کنترل‌های ListBox و TextBox است.

تفاوت این دو کنترل عبارت است از:

● در کادر لیست، لیست رشته‌ها معلوم است ولی در کادر ترکیبی، لیست به صورت منوی باز شو است.

● در کادر لیست، 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 آرایه‌ای است 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: شماره خط جاری لیست را برمی‌گرداند.

● Sorted: در صورت True بودن، لیست را مرتب می‌کند.

۴-۴-۱ اضافه کردن رشته‌ها به ListBox یا ComboBox

اضافه کردن رشته‌ها به دو روش انجام می‌شود.

۱- در زمان طراحی فرم: در مشخصه List مقادیر رشته‌ای را وارد می‌کنیم. دقت کنید که

باید پس از هر رشته کلیده‌های Ctrl Enter را فشار دهید تا مکان‌نما به سر خط رفته و آماده دریافت رشته بعدی شود.

۲- در زمان کدنویسی: از متدی به نام AddItem استفاده می‌شود که فرم کلی آن به صورت

زیر است:

Object.AddItem StringToAdd

● Object، مشخصه Name کنترل کادر لیست یا کادر ترکیبی است.

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

مثال ۳-۴

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

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

۲-۴-۴ انتخاب عنصرها از لیست: برای آشنایی با چگونگی تعیین مقدار رشته 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. همان‌طور که می‌دانید برای دسترسی به عنصر دوم آرایه‌ای به نام MyArray، دستوری به صورت زیر می‌نویسیم (عنصر اول در خانه صفر قرار دارد):

```
MyValue MyArray (1)
```

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

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

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

۱- در هر کنترل ListBox و ComboBox، شماره عنصر انتخاب شده در لیست، در مشخصه‌ای به نام ListIndex قرار می‌گیرد. بنابراین، برای تعیین مقدار رشته انتخاب شده در List1، کد صفحه بعد را به کار خواهید برد:

List1 . list (شماره عنصر انتخاب شده)

⇒ List1 . List(list1 . listIndex)

List1 . ListIndex شماره عنصر انتخاب شده

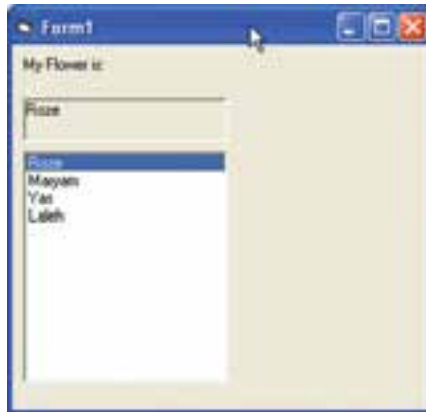
در کد زیر هنگامی که کاربر روی رشته‌ای در ListBox کلیک می‌کند، کد اجرا شده و مقدار

انتخاب شده در لیست را به مشخصه Caption برجسیبی به نام Label1 انتساب می‌دهد. شکل ۴-۴ رویدادهایی را که هنگام انتخاب یک رشته رخ می‌دهند نشان می‌دهد.

```
Private Sub List_Click()
```

```
Label1 . Caption = List1 . List(List1 . ListIndex)
```

```
End Sub
```



شکل ۴-۴ می‌توان برای رویدادهای Click، MouseUp یا MouseDown یک ListBox کد نوشت تا مقدار انتخاب شده را به دست آورد.

۲- روش سریع‌تر برای به دست آوردن مقدار یک رشته انتخاب شده در کنترل‌های ListBox یا

ComboBox، استفاده از مشخصه Text است. به عنوان 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
```

۳-۴-۴ حذف عنصرها از لیست : با استفاده از متد RemoveItem می‌توان رشته‌ای را از لیست ListBox و ComboBox حذف کرد :

Object. RemoveItem Index

- Object، مشخصه Name کنترل ListBox یا ComboBox است.
 - RemoveItem کلیدواژه ویزوال بیسیک برای متدی است که عناصر را از لیست حذف می‌کند.
 - Index محل رشته‌ای است که می‌خواهید از لیست حذف کنید. برای حذف عنصر انتخاب شده از لیست، از مشخصه ListIndex استفاده کنید.
- شکل ۴-۵ بهینه سازی شده برنامه شکل ۴-۴ 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 ۴-۵ دکمه فرمان روشی برای حذف عنصری از لیست است.

۴-۴-۴ پاک کردن لیست : در صورتی که می‌خواهید تمام رشته‌های موجود در ListBox یا ComboBox را حذف کنید، از متد Clear استفاده کنید :

Object. Clear

بنابراین، برای پاک کردن لیست شکل ۴-۷ خواهیم نوشت :

List1.Clear

۴-۴-۵ 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 داده 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.

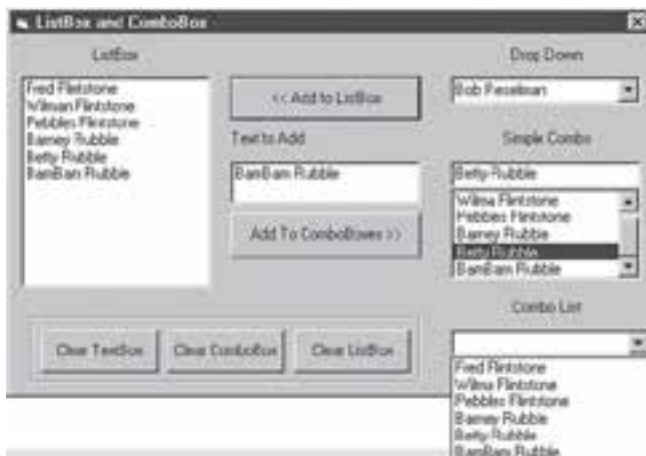
مشخصه **Style** کنترل **ComboBox** امکان تغییر مشخصه‌های عملیاتی و ظاهر کنترل را فراهم می‌کند. جدول ۴-۱ این شیوه‌ها را شرح می‌دهد و شکل ۴-۶ شیوه‌های **ComboBox** اعمال شده به فرم را نشان می‌دهد.

نتیجه

هنگامی که یک **ComboBox** را با شیوه **Simple Combo-1** به فرم اضافه می‌کنید، **ComboBox** تغییر شکل یافته و به صورت **ComboBox** مشاهده نمی‌شود. مشخصه **Height** را افزایش دهید تا **ListBox** آن نمایش یابد.

جدول ۴-۱- مقادیر مربوط به مشخصه **Style** کنترل **ComboBox**

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



شکل ۴-۶- توجه کنید که با شیوهٔ ComboBox بازشو، می‌توان داده‌های جدیدی را در زمان اجرا به کنترل اضافه کرد.

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

مثال ۴-۴

فرمی به صورت زیر ایجاد کنید :



شکل ۴-۷

یک کادر لیست و یک کادر ترکیبی به نام‌های Lstflower و Cboprice به همراه دو کادر متن و یک دکمه فرمان وجود دارد. با کلیک روی دکمه Go متن موجود در هر کادر متن، در کادر لیست مربوطه قرار می‌گیرد. کد مربوط به این مثال به صورت 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 ۴-۴ را طوری تغییر دهید که:
الف) اگر نامی تکراری وارد شد، به لیست اضافه نکند.
ب) دکمه‌ای برای پاک کردن 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 Labeled نمایش داده شود (راهنمایی):
گل و قیمت آن در لیست‌ها دارای شماره یکسان هستند.)

مثال ۴-۵

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

می پردازد و نتیجه را به صورت عبارتی به لیست پایین پنجره اضافه می کند.

```
Dim C As Integer
```

```
Private Sub cmdComp Click()
```

```
    C = StrComp(Text1.Text, Text2.Text)
```

```
    Select case C
```

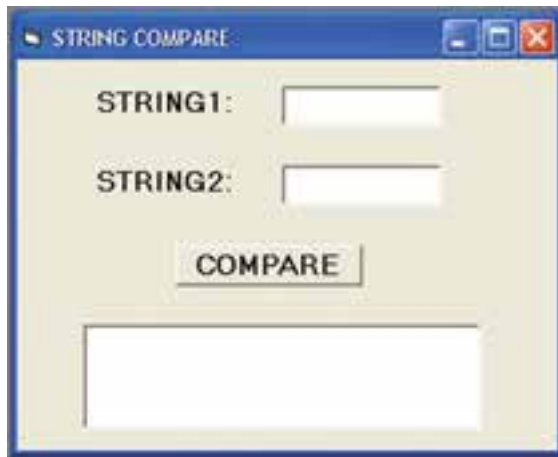
```
        Case 0 : List1.AddItem Text1.Text < Text2.Text
```

```
        Case 1 : List1.AddItem Text1.Text > Text2.Text
```

```
        Case 1 : List1.AddItem Text1.Text < Text2.Text
```

```
    End Select
```

```
End Sub
```



شکل ۸-۴

از تابع `StrComp()` برای مقایسه دو رشته استفاده می شود 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 می گیرد و نوع آن عددی

صحیح است.

● **String 1, String 2** : دو رشته ای هستند که با هم مقایسه خواهند شد. اگر دو رشته با

یکدیگر برابر باشند، تابع مقدار صفر را برمی گرداند و اگر رشته اول کوچک تر از رشته دوم باشد، تابع

مقدار ۱ و در صورت بزرگ تر بودن رشته اول از رشته دوم، مقدار ۱ برگردانده می شود (مقایسه رشته ها بر اساس کداسکی انجام می شود).

● **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 می شود.

مثال ۴-۶

فرمی به صورت زیر طراحی کنید که با انتخاب نام فصل از یک ComboBox نام ماه های آن فصل را در یک ListBox نمایش دهد.



شکل ۴-۹

نام کادر ترکیبی را Combo1 و نام کادر لیست را List1 قرار دهید.
قبل از اضافه کردن ماه های فصل انتخاب شده به List1 باید محتوای List1 پاک شود (دستور List1.Clear)

Private Sub Combol_Click()

Select case Combol.Text

Case بهار

List1.Clear

List1.AddItem فروردین

List1.AddItem اردیبهشت

List1.AddItem خرداد

Case تابستان

Listl. Clear

Listl. AddItem تیر

Listl. AddItem مرداد

Listl. AddItem شهر یور

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



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 دکمه فرمان روی فرم قرار داده و ظاهر آن را مطابق شکل صفحه بعد تنظیم کنید:

کد برنامه به صورت زیر خواهد بود:

Private Sub Form_Load()

lstCities.Clear

lstCities.AddItem شیراز

lstCities.AddItem اصفهان

lstCities.AddItem تبریز



شکل ۱۰-۴

IstCities.AddItem ارومیه

IstCities.AddItem اهواز

IstCities.AddItem یزد

IstCities.AddItem بوشهر

IstCities.AddItem زنجان

IstCities.AddItem رشت

IstCities.AddItem ساری

IstCities.AddItem گرگان

IstCities.AddItem مشهد

IstCities.AddItem زاهدان

IstCities.AddItem کرمان

IstCities.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 = 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.

۵-۴- آرایه کنترلی

در ویژوال بیسیک می‌توان آرایه‌هایی از انواع داده 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.

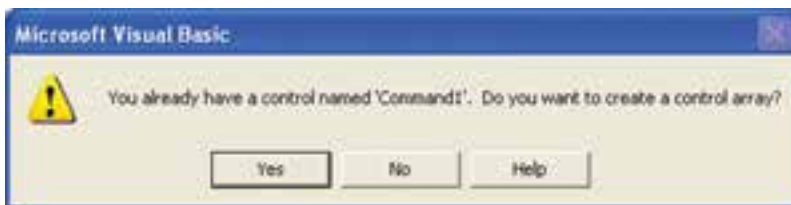
تمام کنترل‌های ذاتی را می‌توان به صورت آرایه‌های کنترلی به کار برد. همهٔ این کنترل‌ها دارای مشخصهٔ Index هستند که برای شناسایی کنترل خاصی در آرایه مورد استفاده قرار می‌گیرند.

۴-۵-۱- 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 ۸-۴

ایجاد آرایهٔ کنترلی از CommandButtons

- ۱- پروژهٔ جدیدی را شروع کنید و نام فرم بیش فرض را به frmMain تغییر دهید.
- ۲- 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.
- ۳- دکمهٔ فرمان را انتخاب کنید. از منوی Edit گزینهٔ Copy را انتخاب کنید تا دکمهٔ فرمان به حافظهٔ Clipboard کپی شود.
- ۴- 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 ۴-۱۱).



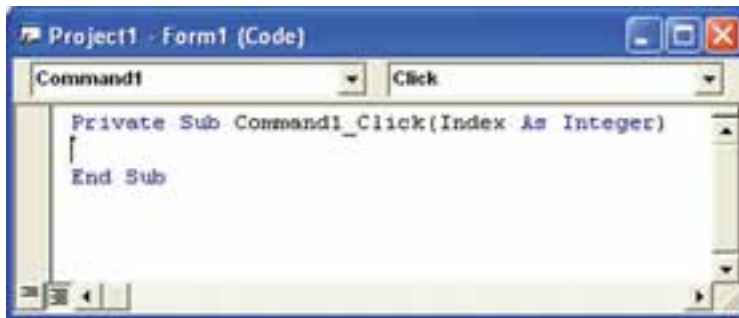
شکل ۴-۱۱- کادر محاوره‌ای تأیید ایجاد آرایهٔ کنترلی

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 هر کدام دارای اندیس خاصی هستند (شکل ۴-۱۲).



شکل ۴-۱۲- 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 ۴-۱۳). این آرگومان یک عدد صحیح است که اندیس کنترل را مشخص می‌کند. به دلیل این که 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 ۴-۱۳- هر زمانی که رویدادی برای کنترلی از یک آرایهٔ کنترلی رخ دهد، ویزوال بیسیک آرگومان Index را مقداردهی می‌کند.

کد زیر نشان می‌دهد که روی کدام دکمهٔ فرمان از آرایهٔ کنترلی کلیک شده است. شکل ۴-۱۴ اجرای این کد را نشان می‌دهد.

```
Private Sub cmdMyButton_Click(Index As Integer)
```

```
Me.Caption = "You clicked button # " & Index & " ."
```

```
End Sub
```



شکل ۱۴-۴- بعد از کلیک کاربر روی دکمه سمت راست، عنوان فرم مطابق آن تغییر می‌یابد.

همانطور که 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 آرایه کنترلی است روال رویدادهای آرایه‌های کنترلی دارای آرگومان ورودی Index است که مشخص‌کننده شماره عنصری از آرایه کنترلی است که رویداد برای آن رخ داده است.

۴-۵-۲- اضافه کردن عناصرها به آرایه کنترلی در زمان اجرا : ایجاد آرایه 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 آرایه کنترلی در زمان اجرا، حل کرد.

مثال ۹-۴

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

۱- پروژة جدیدی را شروع کنید. نام فرم را به frmDArray تغییر دهید.

۲- یک دکمه فرمان به گوشه سمت چپ بالای فرم اضافه کنید و نام آن را cmdCtrlArray قرار دهید.

۳- در پنجره Properties، مقدار مشخصه Index دکمه فرمان را با صفر (۰) مقداردهی کنید.

۴- مشخصه Caption دکمه فرمان را به Button#0 تغییر دهید.

۵- 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
```

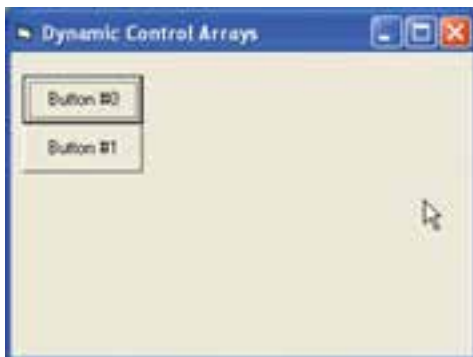
۶- کد را ذخیره کرده و اجرا کنید.

بعد از اجرای کد، برنامه یک دکمه فرمان جدیدی را ایجاد می‌کند و آن را زیر اولین دکمه فرمان

قرار می‌دهد (شکل ۴-۱۵).

نتیجه

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



شکل ۴-۱۵- استفاده از دستور Load دکمه فرمان دیگری را روی فرم قرار می‌دهد. بقیه کد محل دقیق کنترل را تعیین می‌کند.

کنترل‌های جدیدی که ایجاد می‌کنید دقیقاً تکرار اولین عنصر آرایه کنترل‌ها هستند. مقادیر همه 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 مشخصه Top و Left ممکن است. همان‌طور که در مثال قبل مشاهده کردید، مزیت استفاده از آرایه‌های کنترل، توانایی داشتن مجری رویداد مشترک است.



آرایه‌ای ۵ عنصری از تصاویر داریم که می‌خواهیم با هر بار کلیک روی دکمه Next Picture تصویر بعدی نمایش داده شود. ۱- پروژه‌ای به صورت زیر ایجاد کنید (شکل ۴-۱۶). (این پروژه دارای ۵ عنصر تصویر است که روی هم قرار گرفته‌اند)



شکل ۴-۱۶

۲- کد زیر را برای فرم وارد کنید :

```
Dim cnt As Byte,i As Byte
Private Sub Cmdpic_Click()
    img1(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
```

۳- پروژه را اجرا کنید.

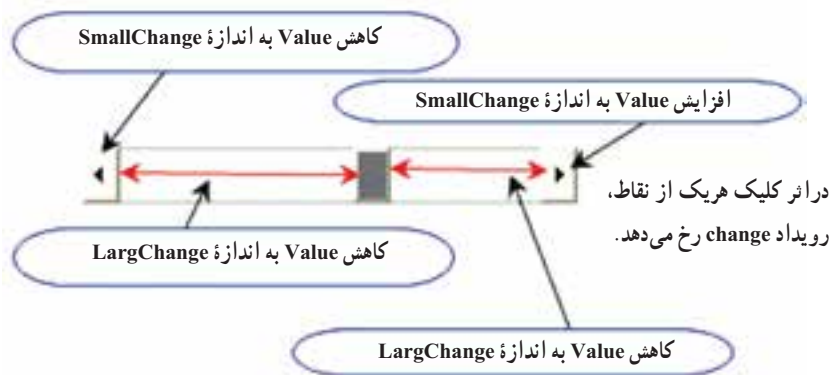
۴-۶ کاربرد کنترل‌های Scroll Bar

کنترل‌های نوار لغزان استاندارد (VScrollBar و HScrollBar) امکان تغییر داده‌ها یا جابه‌جایی در محدوده‌ای از مقادیر را با کلیک کردن روی دکمه‌های Up و Down یا جابه‌جایی دکمه لغزان، فراهم می‌کند. کنترل‌های نوار لغزان دارای چند مشخصه خاص هستند که در جدول ۴-۲ با آن‌ها آشنا خواهید شد.

جدول ۴-۲ مشخصه‌های خاص کنترل‌های HScrollBar و VScrollBar

شرح	مشخصه
کمترین مقدار کنترل را هنگامی که دکمه 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.	Min
بیشترین مقدار ممکن کنترل را هنگامی که دکمه لغزان در پایین یا سمت راست نوار لغزان قرار دارد، تعیین می‌کند. مقدار پیش فرض، ۳۲۷۶۷ است.	Max
موقعیت دکمه لغزان را نسبت به مشخصه‌های Min و Max تعیین می‌کند.	Value
مقدار تغییر مشخصه Value را هنگامی که کاربران بین دکمه لغزان و فلش کلیک می‌کنند تعیین می‌کند.	LargChange
مقدار تغییر مشخصه Value را هنگامی که کاربران روی فلش‌ها کلیک می‌کنند تعیین می‌کند.	SmallChange

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 با دکمه لغزان جابه‌جا شود رویداد Change رخ می‌دهد. هنگام حرکت دادن دکمه لغزان چندین رویداد Scroll رخ می‌دهد و به محض متوقف شدن دکمه یک رویداد Change رخ می‌دهد.



شکل ۱۷-۴- میزان تغییرات value در اثر کلیک نقاط مختلف در ScrollBar

مثال ۱۱-۴

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

- ۱- پروژه جدیدی به نام SmpScroll.vbp ایجاد کنید. نام فرم را frmMain قرار دهید.
- ۲- کنترل‌های VscrollBar و HscrollBar را به فرم اضافه کنید. نام کنترل VscrollBar را vscrNS و کنترل HscrollBar را به hscrWE تغییر دهید.
- ۳- دو کنترل به کادر متن اضافه کنید؛ نام یکی را txtNS و دیگری را txtWE قرار دهید. مشخصه Text هر دو کنترل را یک رشته خالی قرار دهید (شکل ۱۸-۴).
- ۴- مشخصه‌های کنترل‌های VscrollBar و HscrollBar را مطابق جدول ۳-۴ مقاردهی کنید.

کنید.

جدول ۳-۴- تنظیم مشخصه‌های کنترل‌های ScrollBar

مقدار	مشخصه
0	Min
20	Max
1	SmallChange
2	LargeChange

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

نمایش مقدار مشخصه Value نوار لغزان افقی در کادر متن txtWE

```
Private Sub hscrWE_Change()
```

```
txtWE.Text = CStr(hscrWE.Value)
```

```
End Sub
```

نمایش مقدار مشخصه Value نوار لغزان عمودی در کادر متن txtNS

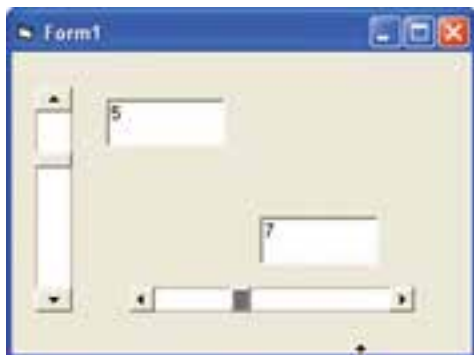
```
Private Sub vscrNS_Change()
```

```
txtNS.Text = CStr(vscrNS.Value)
```

```
End Sub
```

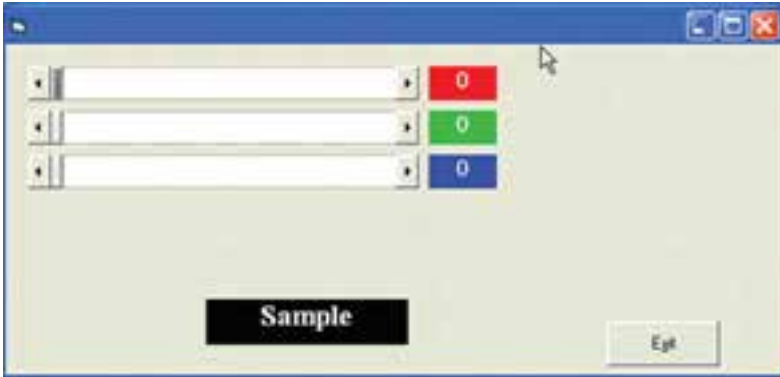
۶- پروژه را ذخیره کرده و اجرا کنید.

هنگامی که پروژه را اجرا کردید، با کلیک روی فلش‌های کنترل‌های HScrollBar و VScrollBar، مقدار کادر متن مربوطه یک واحد تغییر می‌کند (براساس مقدار مشخصه SmallChange). ولی اگر بین فلش و دکمه لغزان کلیک کنید، مقدار کادر متن مربوطه، دو واحد تغییر می‌کند (براساس مقدار مشخصه LargeChange) (شکل ۴-۱۸).



شکل ۴-۱۸- به دلیل این که مشخصه Max با ۲۰ مقداردهی شده است، هنگام جابه‌جایی نوار لغزان، مقادیر نشان داده شده در کادرهای متن تا ۲۰ می‌رسند.

فرمی به صورت شکل ۴-۱۹ ایجاد کنید. با تغییر نوارهای لغزان مربوط به سه رنگ قرمز، آبی و سبز می توان رنگ برجسب را تغییر داد.



شکل ۴-۱۹

مقدار Value هر نوار لغزان روی برجسبی که کنار آن قرار دارد نشان داده می شود. و رنگ زمینه برجسب Label1 براساس مقدار Value هر سه نوار لغزان تعیین می شود.
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 HScroll1 Change()
```

```
r HScroll1.Value
```

```
g HScroll2.Value
b HScroll3.Value
Lblr.Caption r
Lblg.Caption g
Llbl.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
    Llbl.Caption b
    Label1.BackColor RGB(r, g, b)
```

End Sub

```
Private Sub HScroll12 Change()
```

```
    r HScroll1.Value
    g HScroll2.Value
    b HScroll3.Value
    Lblr.Caption r
    Lblg.Caption g
    Llbl.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
```


Label.BackColor RGB(r, g, b)

End Sub

تمرین ۱: در مثال ۱۲-۴ کنترل‌های نوار لغزان و برچسب‌های کنار آن را به صورت آرایه‌های کنترلی طراحی کنید.

تمرین ۲: مثال ۱۲-۴ را به گونه‌ای تغییر دهید که رنگ زمینه از طریق یک کادر لیست انتخاب شود و شدت تمام رنگ‌ها با یک scrollbar تعیین شود.

۷-۴- 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 تکراری بودن نام، شماره تلفن‌ها را در کتابچه راهنمای تلفن مرتب می‌کند که با این روش، جست‌وجوی شماره 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 عناصر A را به صورت افزایشی (صعودی) مرتب کنیم یعنی

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

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 سوومی باید استفاده کنیم.

در تکه کد زیر برای جابجا کردن 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

مقایسه دو عنصر کنار هم و در صورت لزوم جابجا کردن محتوای آن‌ها
عمل مقایسه دو به دو در اولین مرحله به تعداد $n-1$ 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 $n-1$ ، در مرحله دوم تعداد مقایسه‌ها $n-2$ ، ... می‌باشد. لذا تعداد مقایسه‌های هر مرحله
به شماره آن مرحله بستگی دارد و اگر شماره مرحله را در متغیر `intpass` قرار دهیم تعداد مقایسه‌ها در
هر مرحله `intpass - n` 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` به صورت:

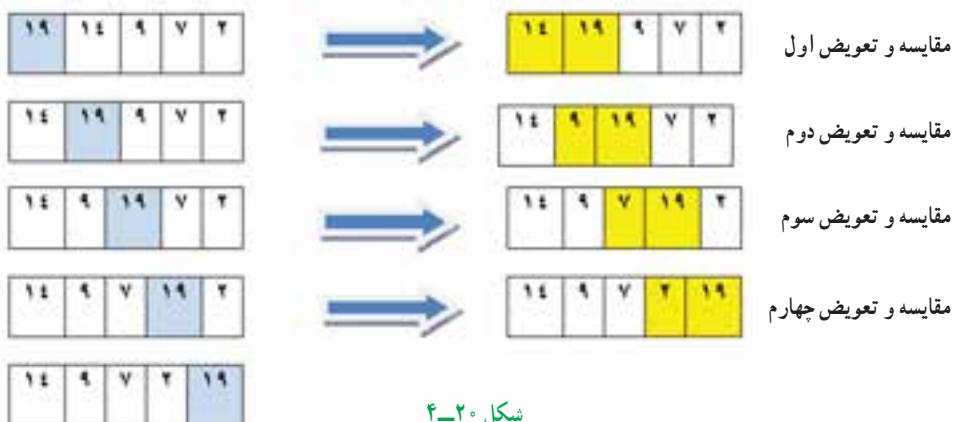
A(1)	A(2)	A(3)	A(4)	A(5)
۱۸	۱۴	۹	۷	۲

را در نظر بگیرید.

در شکل ۲-۴ نتیجه اجرای قطعه کد قبل برای `intpass = 1` روی آرایه `A` مشاهده

می‌کنید.

مرحله اول



شکل ۲-۴

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

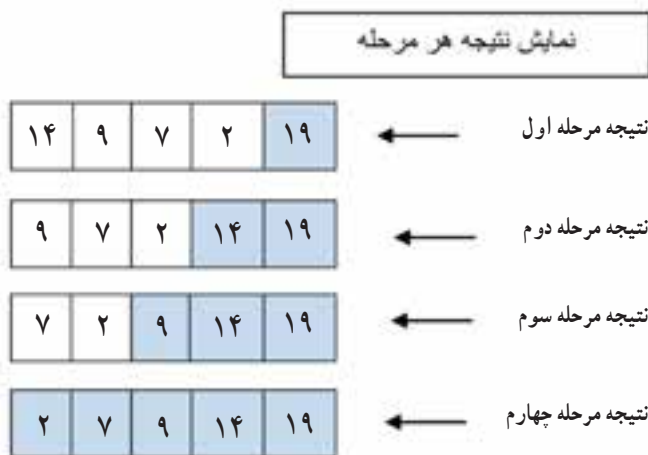
برای مرتب شدن تمام عناصر آرایه، این مراحل حداکثر n بار باید تکرار شود لذا تکه کد قبل باید درون یک حلقه For قرار گیرد که متغیر این حلقه شماره مرحله را نگهداری می کند.

1- For intpass LBound (intarray A) To UBound (intarray A)

تکه کد مربوط به اجرای هر مرحله

Next

خروجی آرایه A پس از هر مرحله در شکل ۴-۲۱ آمده است.



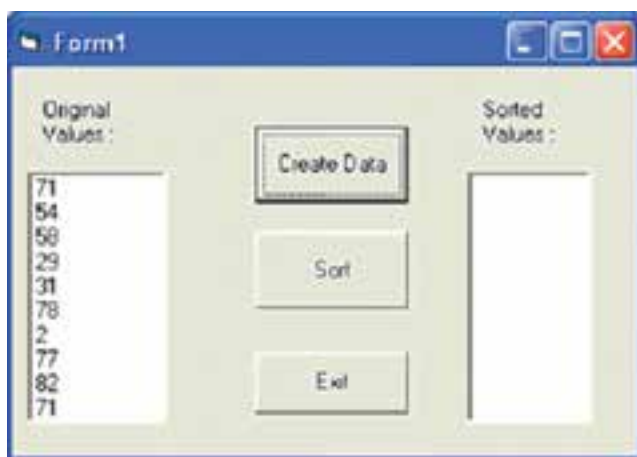
شکل ۴-۲۱

همانطور که در جدول مشاهده می کنید در هر مرحله یکی از عناصر آرایه مرتب می شود. در مرحله اول $A(5)$ ، در مرحله دوم $A(4)$ ، در مرحله سوم $A(3)$ و در مرحله چهارم $A(2)$ و $A(1)$ با هم مرتب می شوند. از آنجا که دو عنصر $A(2)$ و $A(1)$ با هم مرتب می شوند تعداد مراحل از تعداد عناصر آرایه یکی کمتر است.

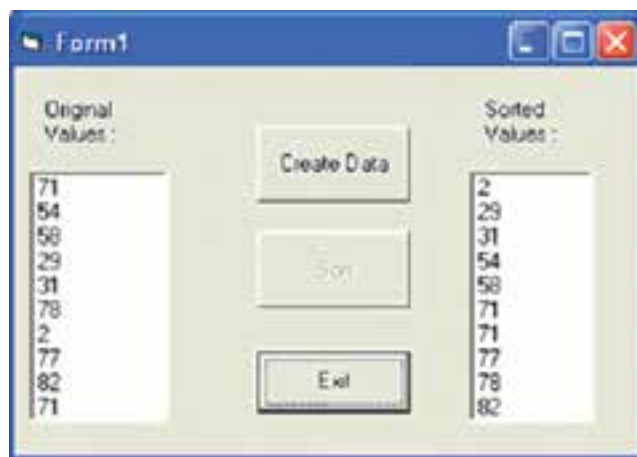
مثال ۴-۱۳

پروژه ای ایجاد کنید که با کلیک روی دکمه Create Data ده (10) عدد صحیح تصادفی در یک کادر لیست قرار بگیرند، سپس با کلیک روی دکمه Sort این اعداد مرتب شده و در کادر لیست دوم کپی شوند.

برنامه زیر، مقادیر آرایه 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 ۴-۲۲



شکل ۴-۲۳

Option Explicit

اندیس اولین عنصر آرایه یک می‌باشد 1 Base Option

Dim intarrayA(10) As Integer

Private Sub cmdGenerate Click ()

Dim inti As Integer

IstOriginal.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)

IstOriginal.AddItem intarrayA(inti)

قراردادن عناصر آرایه در کادر لیست اولیه

Next

IstSorted.Clear خالی کردن کادر لیست IstSorted

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 حلقه شمارنده مراحل

For intpass LBound(intarrayA) To UBound (intarrayA) - 1

حلقه مقایسه دو به دو 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 + 1) = inttemp

```

        End If
    Next
Next
        LSt sorted عناصر آرایه مرتب شده به کادر لیست
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

```

۸-۴- جست و جو

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.

۸-۴-۱- 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.

برنامه‌ای بنویسید که ۱۰ عدد حداکثر ۴ رقمی تولید کند و در صورت یافتن کلید جست‌وجو شماره عنصر و در صورت نیافتن پیغام مناسب نمایش دهد.



شکل ۲۴-۴- 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
```

مقایسه عناصر آرایه با کلید جست و جو

```
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 اضافه کردن به کادر لیست lstData ***

For inti LBound(intarrayA) To UBound(intarrayA)

intarrayA(inti) Int(Rnd*10000)

lstData.AddItem intarrayA (inti)

Next

End sub

۲-۸-۴- جست و جوی دودویی: این روش جست و جو برای آرایه‌های بزرگ مناسب

است. به دلیل این که در این نوع آرایه‌ها با روش جست و جوی خطی، تعداد دفعات جست و جو زیاد خواهد بود که سبب اتلاف زمان می‌شود.

روش جست و جوی دودویی، روی آرایه‌های مرتب قابل اجراست. بنابراین، در صورتی که آرایه مورد نظر نامرتب باشد، ابتدا باید آن را مرتب کنید.

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

• عنصر میانی با کلید جست و جو مساوی است. در این حالت جست و جو موفق بوده و خاتمه می‌یابد و اندیس عنصر میانی به عنوان مکان عنصر مورد جست و جو برگردانده می‌شود.

• عنصر میانی از کلید جست و جو بزرگ‌تر است. بنابراین از تمام عناصر قبل از آن نیز بزرگ‌تر خواهد بود و می‌توان جست و جو را در نیمه بالایی (سمت راست عنصر میانی) ادامه داد.

• عنصر میانی از کلید جست و جو کوچک‌تر است. بنابراین از تمام عناصر بعد از آن نیز کوچک‌تر خواهد بود و می‌توان جست و جو را در نیمه پایینی (سمت چپ عنصر میانی) ادامه داد.

به مثال زیر توجه کنید:

در لیست ۲۴، ۷، ۸، ۱۹ نیمه پایینی $\{2, 4, 7, 8, 19\}$ و نیمه بالایی $\{7, 8, 19\}$

۱- جست و جوی عدد ۷: عنصر میانی مساوی کلید جست و جو است (۷ > ۷).

۲- جست و جوی عدد ۴: عنصر میانی از کلید جست و جو بزرگ‌تر است پس در نیمه بالایی به دنبال عدد می‌گردیم (۷ > ۴).

۳- جست و جوی عدد ۱۹: عنصر میانی از کلید جست و جو کوچک‌تر است لذا در نیمه پایینی به دنبال عدد می‌گردیم (۷ < ۱۹).

در این روش، جست و جو 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 باز کرده 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 ۱۵-۴

فرض کنید آرایه‌ای از اعداد مرتب به صورت 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 به دست 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 به صورت 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 به دست 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 در نتیجه سه حالت اتفاق می‌افتد:

- ۱- کلید مساوی با مقدار عنصر وسط لیست است.
 - ۲- کلید بزرگ‌تر از مقدار عنصر وسط لیست است.
 - ۳- کلید کوچک‌تر از مقدار عنصر وسط لیست است.
- در حالت اول کار جست‌وجو تمام شده است.

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

حالت سوم مشابه حالت دوم است با این تفاوت که کلید نمی‌تواند در نیمه پایینی لیست باشد و باید نیمه بالای لیست را جست‌وجو کنیم.

در این روش با یک عمل مقایسه نصف عناصر لیست را کنار زده‌ایم. در واقع در یک لیست ۱۰۰,۰۰۰ عنصری با یک عمل مقایسه نصف عناصر لیست را کنار می‌گذاریم.

با مقایسه 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 ۱۶-۴

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



شکل ۲۵-۴

Option Explicit

```
Private Sub cmdCreate Click()  
    Dim inti As integer  
    Dim intnum As Integer  
    Lstnumber.Clear  
    txtkey " "  
    '*** تولید ۱۰ عدد تصادفی دورقمی ۱۰ ≤ intNum < ۱۰۰ ***  
    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
    Else
        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

```

۹-۴- آرایه‌های چندبعدی

آرایه‌هایی که تا اینجای فصل ایجاد کردیم، همان‌طور که در شکل ۲-۴ 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 به وجود می‌آیند). هر خانه دارای موقعیت تعریف‌شده‌ای به صورت شمارهٔ سطر و شمارهٔ ستون است.

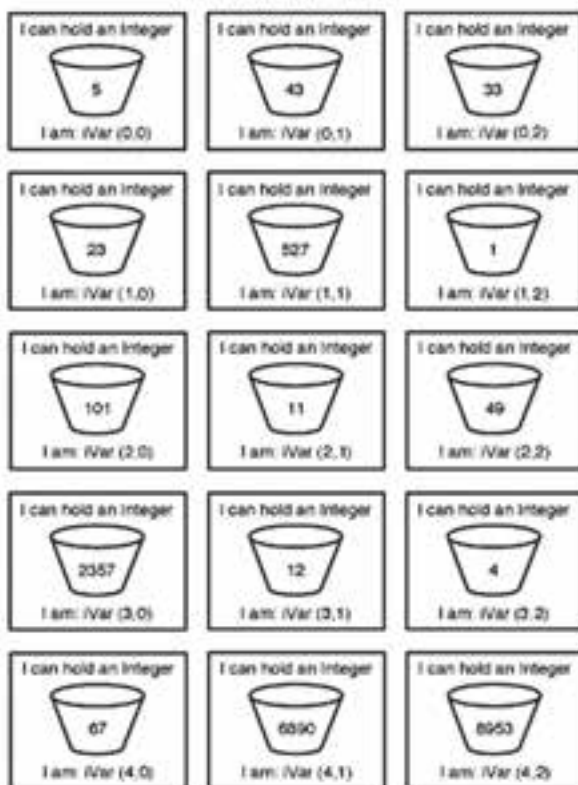
توجه داشته باشید که هر عنصر بر اساس مختصات سطر و ستون تعریف می‌شود. به عنوان مثال، عنصر $iVar(0,0)$ دارای مقدار ۵ و عنصر $iVar(2,2)$ دارای مقدار ۴۹ است (شکل ۲۶-۴).
برای ایجاد آرایهٔ دوبعدی، از شکل کلی زیر استفاده کنید:

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

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

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

Dim iVar(2,4) as Integer



شکل ۲۶-۴ – آرایهٔ دوبعدی را به صورت مجموعه‌ای از ظرف‌ها که در ستون‌ها و سطرها سازماندهی شده‌اند، تصور کنید.

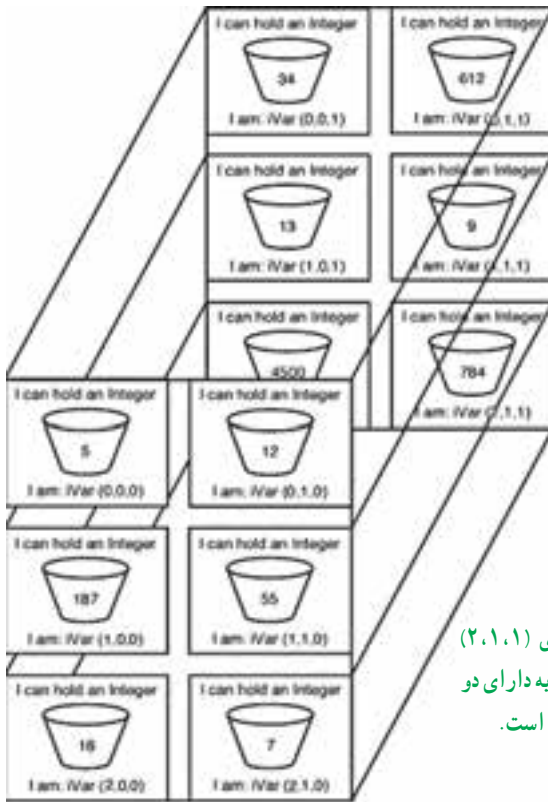
بنابراین، برای اعلان آرایه شکل ۲۶-۴ می‌توان نوشت :

$\text{Dim iVar (4,2) As Integer}$

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

$\text{Dim iVar (2,1,1) As Integer}$

شکل ۲۷-۴ این آرایه سه‌بعدی را نشان می‌دهد.



شکل ۲۷-۴ آرایه سه‌بعدی (۲،۱،۱) دارای ۱۲ عنصر است. این آرایه دارای دو ستون و سه سطر به عمق ۴ است.

(تعداد صفحات عمق × تعداد سطر × تعداد ستون) تعداد عناصر آرایه سه‌بعدی

در آرایه iVar مقدار تعیین‌شده برای هر عنصر که در شکل ۲۷-۴ نشان داده شده است،

به صورت زیر خواهد بود :

$\text{iVar}(0,0,0) \quad 5$

$\text{iVar}(1,0,0) \quad 187$

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 یک بعدی، 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 سه بعدی 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 نمی‌تواند تغییر کنند.

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

۲- برنامه‌ای بنویسید که عناصر یک ماتریس 4×4 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.

۳- برنامه‌ای بنویسید که عناصر یک ماتریس 4×4 را دریافت کند. سپس عددی را گرفته، آن را در ماتریس جست و جو کند و تعداد تکرار آن را نمایش دهد.

۴- برنامه‌ای بنویسید که عناصر یک ماتریس 3×3 را دریافت کند. سپس این ماتریس را نشان داده و اعلام کند بالا مثلثی است یا پایین مثلثی.

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