

## روال‌ها و توابع

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

- مفهوم روال، تابع و تفاوت‌های آن‌ها را بیان کند؛
- برنامه‌ها را در صورت لزوم به کمک توابع و روال‌ها به قسمت‌های کوچک تقسیم و برنامه‌نویسی کند.

### ۱-۱ - استفاده از روال‌ها در ویژوال بیسیک

ویژوال بیسیک، یک زبان برنامه‌نویسی روالی است. بعد از نامگذاری یک بلاک کد، می‌توان آن را فراخوانی و اجرا کرد. به عبارت دیگر، می‌توان چند خط کد نوشت و آن را در یک بلاک قرار داده و نامی به آن اختصاص داد. سپس بلاک کد را هنگام نیاز فراخوانی کرد. این بلاک کد تقریباً شبیه برنامه‌ای در داخل برنامه دیگر است. این برنامه‌های کوچک را که داخل برنامه‌های بزرگ هستند در صورتی که مقدار برگردانند، "Function" و در غیر این صورت "Sub" (در حقیقت sub مخفف subroutine است.) می‌نامند. روال‌های رویدادی مثل Click() و Load از نوع Sub هستند و LoadPicture() و Len() توابعی هستند که قبلاً با آن‌ها کار کرده‌اید.

برنامه‌نویسی با این مفاهیم، سال‌هاست که رواج دارد این روش کدنویسی را ساده‌تر، سریع‌تر و کارآمدتر می‌کند. همچنین استفاده از این مفاهیم، امکان نوشتن کدهایی که قابلیت استفاده مجدد را دارند فراهم می‌کند.

روالها امکان تغییر ساده کد را فراهم می‌کنند. اگر نیاز به استفاده مکرر از کدی را دارید، آن را در یک روال قرار دهید. در این صورت اگر نیاز به تغییر کد داشته باشید، به سادگی می‌توانید به آن

رجوع کرده و تغییرات را اعمال کنید. اگر کد را در یک روال قرار ندهید، مجبور خواهید بود که به هر نمونه‌ای از کد در برنامه رجوع کرده و تغییرات مورد نیاز را اعمال کنید؛ البته انجام تغییرات مؤثر و کارآمد با این روش، مشکل خواهد بود.

## ۱-۲- ایجاد و فراخوانی یک Sub ساده

یک sub روالی است که خطوطی از کد داخل بلاک را اجرا می‌کند ولی مقداری را بر نمی‌گرداند. شکل کلی یک sub ساده به صورت زیر است:

```
[private|public] sub SubName()
```

خطوطی از کد ...

```
End Sub
```

- [private|public] کلید واژه‌های اختیاری هستند که حوزه عمل sub را تعریف می‌کنند.
  - Sub کلید واژه‌ای است که نوع روال را تعیین می‌کند.
  - SubName نامی است که برای روال تعیین می‌شود.
  - End Sub کلید واژه‌هایی هستند که پایان بلاک کد را مشخص می‌کنند.
- کد زیر، مثالی از یک sub ساده است:

```
Public Sub DataFinding (
```

```
MgBox"Data Not Found", vbInformation
```

```
End Sub
```

هنگامی که این sub را از سایر نواحی کد، فراخوانی می‌کنید، sub کادر پیغامی را با رشته Data Not Found نمایش می‌دهد.

کد زیر نشان می‌دهد که یک sub با دستور Call فراخوانی شده است. استفاده از دستور Call اختیاری است. اگرچه می‌توان یک sub را بدون کلیدواژه Call فراخوانی کرد (با نوشتن نام آن) ولی استفاده از این کلید واژه، خوانایی کد را افزایش می‌دهد:

```
Private Sub itmOpen_Click()
```

```
Call DataFinding
```


```
End Sub
```

---

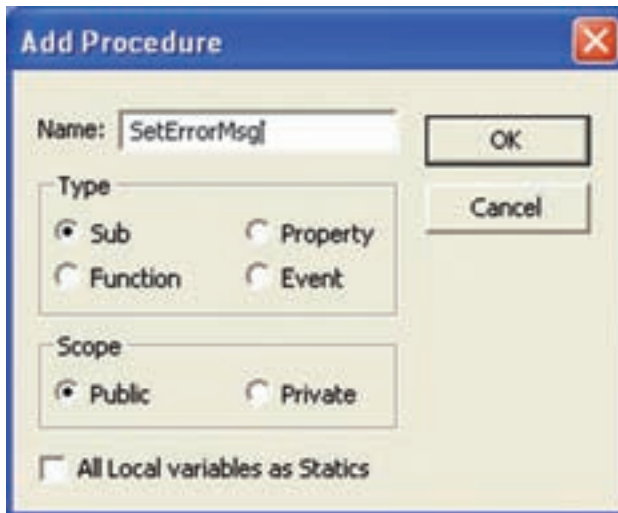
۱- حوزه عمل در ادامه همین فصل توضیح داده شده است.

### ۱-۳- ایجاد یک Sub ساده

- می‌توان یک sub را به دو روش به پروژه اضافه کرد :
- با نوشتن مستقیم کد در بخش General Declarations یک فرم یا مدول.
- با استفاده از گزینه Add Procedure منوی Tools.

 **نکته:** برای فعال کردن گزینه Add Procedure باید در پنجره code فرم یا مدول موردنظر باشید.

- ۱- مراحل اضافه کردن sub به پروژه با روش دوم، به صورت زیر است :
- ۱- از منوی Tools گزینه Add Procedure را انتخاب کنید تا کادر محاوره‌ای مربوطه باز شود.
- ۲- نامی را برای sub وارد کنید (شکل ۱-۱).
- ۳- Sub را از گزینه‌های Type انتخاب کنید.
- ۴- از گزینه‌های Scope نوع حوزه عمل sub را انتخاب کنید.



شکل ۱-۱- کادر محاوره‌ای Add Procedure امکان ایجاد Subs و توابع برای انواع پروژه‌های VB را فراهم می‌کند.

- ۵- روی Ok کلیک کرده و بلاک کد را به فرم یا مدول اضافه کنید (شکل ۱-۲).

بعد از این که بلاک کد را با کادر محاوره‌ای Add Procedure ایجاد کردید، کد روال را بین اعلان sub و کلید واژه End sub اضافه کنید. بعد از End Sub کدی را وارد نکنید، انجام این کار سبب بروز خطا در زمان کامپایل خواهد شد.

```

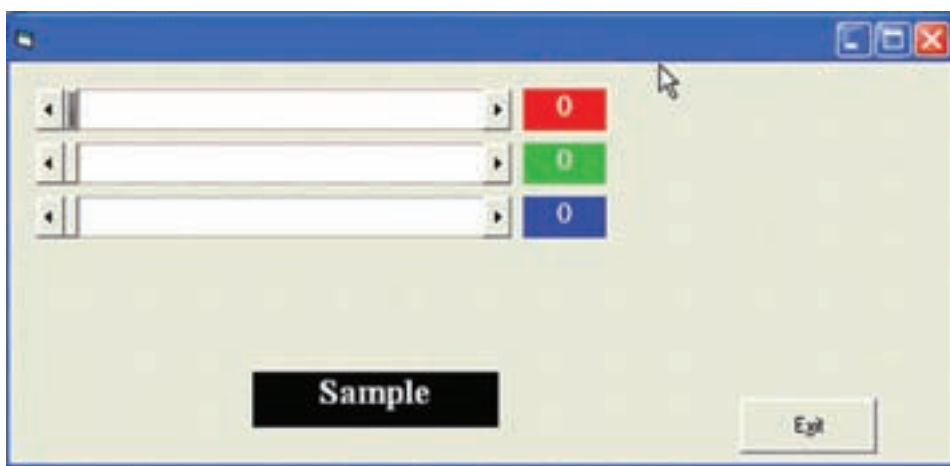
Project1 - Form1 (Code)
(General) SetErrorMsg
Public Sub SetErrorMsg()
End Sub

```

شکل ۱-۲ sub جدیدی را در بخش General فرم یا مدول به دست خواهید آورد.

مثال ۱-۱

فرمی به صورت شکل ۱-۳ ایجاد نمایید که با تغییر نوارهای لغزان مربوط به سه رنگ قرمز، آبی، سبز بتوان رنگ برجسب را تغییر داد.



شکل ۱-۳

```
Private Sub setcolor()  
    r = HScroll1.Value  
    g = HScroll2.Value  
    b = HScroll3.Value  
    Lblr.Caption = r  
    Lbly.Caption = g  
    Lblb.Caption = b  
    Labell.BackColor = RGB(r,g,b)  
End Sub  
  
Private Sub Command1_Click()  
    End  
End Sub  
  
Private Sub Form_Load()  
    Labell.BackColor = RGB(0,0,0)  
End Sub  
  
Private Sub HScroll1_Change()  
    Call setcolor  
End Sub  
  
Private Sub HScroll2_Change()  
    Call setcolor  
End Sub  
  
Private Sub HScroll3_Change()  
    Call setcolor  
End Sub
```

## ۴-۱- ایجاد یک تابع ساده

تابع روالی است که خطوطی از کد را اجرا می‌کند و مقداری را برمی‌گرداند. شکل کلی اعلان یک تابع ساده به صورت زیر است :

```
[private|Public] Function FunctionName() As DataType
```

خطوطی از کد ...

```
Function Name = ReturnValue
```

```
End Function
```

- Private|Public کلید واژه‌های اختیاری هستند که حوزه عمل تابع را تعریف می‌کنند.
- Function کلید واژه‌ای است که مشخص می‌کند، روال از نوع تابع است.
- FunctionName نام تابع است.
- As کلید واژه‌ای برای تعیین نوع داده‌ای است که تابع برمی‌گرداند.
- DataType نوع داده‌ای است که تابع برمی‌گرداند.
- ReturnValue مقداری است که به وسیلهٔ تابع برگردانده می‌شود.
- End Function کلید واژه‌هایی هستند که پایان بلاک کد را مشخص می‌کنند.

کد زیر، تابعی را نشان می‌دهد که مجموع دو عدد تعریف شده در داخل خود تابع را

برمی‌گرداند :

```
Public Function GetNumber() As Integer
```

```
Dim a As Integer
```

```
Dim b As Integer
```

```
Dim c As Integer
```

```
a = 7
```

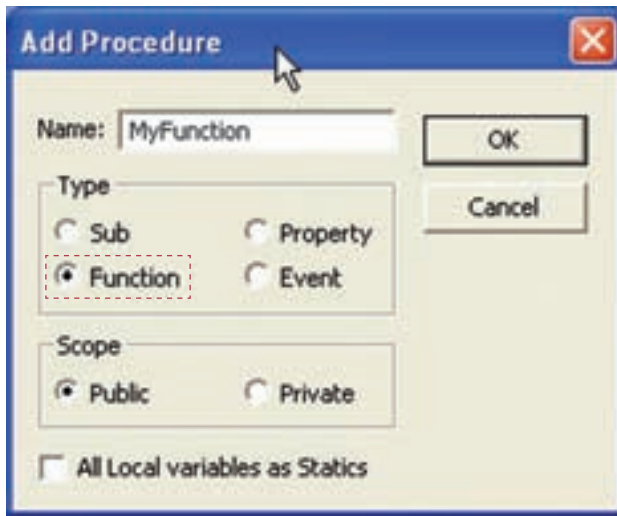
```
b = 12
```

```
c = a+b
```

```
GetNumber = c
```

```
End Function
```

تابع را نیز می‌توان مانند Sub با همان دو روش به فرم یا مدول اضافه کرد (شکل ۴-۱).



شکل ۱-۴- تابع را در کادر محاوره‌ای Add Procedure اضافه کنید.

## ۱-۵- ارسال آرگومان‌ها به روال‌ها

آرگومان (پارامتر)، متغیری است که به عنوان جانگهدار برای مقادیری که به تابع یا Sub ارسال می‌شوند، عمل می‌کند. می‌توان قدرت و همه منظوره بودن روال‌ها را با استفاده از آرگومان‌ها افزایش داد. می‌توان آرگومان‌ها را با قرار دادن آن‌ها در داخل پراتزهای دستور اعلان روال‌ها، ایجاد کرد. کد زیر، اعلان تابع GetGreaterNum که دو آرگومان می‌گیرد را نشان می‌دهد:

```
Public Function GetGreaterNum (NumOne As Integer, NumTwo As Integer) As Integer
```

استفاده از آرگومان‌ها کارایی کد را افزایش می‌دهد. به عنوان مثال، فرض کنید که چندین بار در یک برنامه نیاز دارید که بزرگ‌ترین مقدار بین دو عدد را به دست آورید. مناسب‌ترین روش این است که این کد را در یک تابع بنویسید و هر جایی که می‌خواهید آن را فراخوانی کنید.

کد زیر، تابع GetGreaterNum را نشان می‌دهد که بزرگ‌ترین مقدار بین دو عدد دریافتی را محاسبه می‌کند و برمی‌گرداند.

```
Public Function GetGreaterNum(NumOne As Integer, _
    NumTwo As Integer) As Integer
    If NumOne > NumTwo Then
        GetGreaterNum = NumOne
```

```

Else
    GetGreaterNum=NumTwo
End If
End Function

```

کد زیر چگونگی فراخوانی تابع فوق را از داخل یک روال رویداد Click نشان می دهد :

```

Private Sub cmdGreaterNum_Click()
    Dim i As Integer
    Dim j As Integer
    Dim RetVal As Integer
    i= CInt(txtNumOne.Text)
    j= CInt(txtNumTwo.Text)
    RetVal=GetGreaterNum(i,j)
    cmdGreaterNum.Caption=CStr(RetVal)
End Sub

```

هنگام استفاده از آرگومان‌ها یکسان بودن نوع و ترتیب آن‌ها خیلی مهم است. اگر روالی دارید که سه آرگومان از نوع Integer دارد، باید سه عدد صحیح ارسال کنید. در صورتی که دو عدد صحیح و یک رشته ارسال کنید، کامپایلر یک خطا تولید خواهد کرد. به عنوان مثال، اگر تابعی به نام EndDay() دارید که به صورت زیر اعلان می شود :

```

Public Function EndDay(iNum As Integer, dAccount As Double) As Double

```

و تابع را با استفاده از کد زیر فراخوانی می کنید،

```

dMyResult = EndDay (6,"056R")

```

این فراخوانی، خطایی را تولید می کند. "056R" از نوع رشته ای است ولی تابع برای آرگومان دوم انتظار داده ای از نوع Double را دارد.

همچنین تعداد آرگومان‌ها نیز باید یکسان باشند. به عنوان مثال، فرض کنید تابعی دارید که



به صورت زیر تعریف شده است :

```
Public Function Bar(iNum As Integer, dNum As double, strName As string)
As Integer
```

و با استفاده از کد زیر، آن را فراخوانی می کنید :

```
iMyResult = Bar(6,7)
```

این نیز سبب بروز خطا شود. تابع انتظار سه آرگومان را دارد ولی فقط دو آرگومان ارسال شده است.

می توان آرگومانی را به این منظور از کلید واژه Optional، در اعلان تابع قبل از آرگومان مورد نظر استفاده کرد. آرگومان های اختیاری باید از نوع Variant باشند.

### ۱-۵-۱- کاربرد آرگومان های نام دار: هنگام فراخوانی روال ها می توان از آرگومان های نام دار

برای ارسال ساده تر مقادیر به آن ها استفاده کرد. به عنوان مثال، اگر تابعی به نام GetGreaterNum دارای دو آرگومان از نوع Integer باشد و به صورت زیر تعریف شود :

```
GetGreaterNum (NumOne As Integer, NumTwo as Integer) As Integer
```

هنگام فراخوانی این تابع بعد از اسامی آرگومان ها از نویسه های = : استفاده کرده و مقداری را برای آن ها به صورت زیر تعیین کنید.

```
X=GetGreaterNum(NumOne: =3, NumTwo: =4)
```

### ۱-۶- خروج از روال ها

بعضی مواقع قبل از پایان روال، نیاز به خروج از آن دارید. می توانید این کار را با کلید واژه های Exit Function و Exit sub به ترتیب برای خروج از روال های از نوع Function و sub انجام دهید. کد زیر مربوط به تابع TestExit() است که دو آرگومان X و Y را دریافت می کند و حاصل عبارت زیر را نمایش می دهد :

$$F(x,y) = \frac{x^2y + 2y + 4x}{x}$$

اگر مقدار آرگومان X صفر بود، عبارت جواب ندارد و باید از تابع خارج شود. (خطای تقسیم

بر صفر)

```
Public Function TestExit(x As Integer, y As Integer) As Integer
```

```

If x = 0 then
    MsgBox("Division By Zero")
Exit Function

Else
    TestExit = (x^2*y+2*y+4*x)/x
End If
End Function

```

یک مثال برای Exitsub ذکر شود.

## ۱-۷- آشنایی با حوزه عمل

حوزه عمل (میدان دید)، قابلیت است که دو متغیر مختلف با اسامی یکسان می‌توانند مقادیر مختلفی را نگهداری کنند و دارای دوره حیات متفاوتی هستند. کد زیر، دو تابع `GetNumber()` و `Bar()` را نشان می‌دهد:

حوزه عمل (میدان دید)، محدوده اعتبار متغیرها را تعیین می‌کند. به مثال زیر توجه کنید:

```
01 Public Function GetNumber() as Integer
```

```
02 Dim x as Integer
```

```
03 Dim y as Integer
```

```
04
```

```
05 x = 2
```

```
06 y = 7
```

```
07 GetNumber = x+y
```

```
08 End Function
```

```
09
```

```
10 Public Function Bar() as Integer
```

```
11 Dim x as Integer
```

```
12 Dim y as Integer
```

```
13
```

14  $x = 12$

15  $y = 34$

16  $Bar = x*y$

17 End Function

توجه کنید که هر تابع، متغیرهای  $x$  و  $y$  را اعلان می‌کند. همچنین این متغیرها در هر تابع، مقادیر مختلفی را می‌گیرند. انجام این کار، بدین دلیل است که هر مجموعه‌ای از متغیرها فقط در همان جایی که ایجاد شده‌اند به کار برده می‌شوند. در تابع `GetNumber()`، متغیرهای  $x$  و  $y$  در خطوط ۲ و ۳ ایجاد می‌شوند. هنگامی که تابع خاتمه می‌یابد، متغیرها از حافظه حذف می‌شوند (این را خروج از حوزه عمل می‌نامند). این مطلب دربارهٔ متغیرهای  $x$  و  $y$  در تابع `Bar()` نیز صدق می‌کند. برای این که میدان دید متغیرها محدود به روال نباشد، آن‌ها را در بخش `General Declarations` فرم یا مدول و با استفاده از کلید واژه‌های `Public` یا `Private` اعلان کنید.



```
Public Dim x,y As Integer
```

```
Public Function GetNumber() As Integer
```

```
GetNumber = x+y
```

```
End Function
```

```
Public Function Bar() As Integer
```

```
Bar = x*y
```

```
End Function
```

میدان دید متغیرهای  $x$  و  $y$  در این مثال فرم یا مدولی است که این کد در آن نوشته شده است.

## ۸-۱- مستندسازی روال‌ها

مستندسازی روال‌ها به سایر برنامه‌نویسان امکان می‌دهد که به‌طور کامل از برنامه شما استفاده کرده و در صورت لزوم آن را تغییر دهد. زیرا قبل از هر عبارتی، توضیحی برای کاربرد آن نوشته شده است. تمام روال‌ها دارای یک سرآیند (`header`) خواهند بود. سرآیند بخشی است که در ابتدای بلاک کد آورده شده و توضیحاتی را دربارهٔ روال ارائه می‌دهد.

## ۹-۱- تعیین نقطه ورودی با SubMain()

به طور پیش فرض، هنگامی که پروژه‌ای را در VB شروع می‌کنید، پروژه در آغاز اولین فرم ایجاد شده را فراخوانی می‌کند. اگر دارای پروژه‌ای با چندین فرم هستید، برای دسترسی به فرم‌های دیگر می‌توانید آن‌ها را از داخل اولین فرم، بارگذاری (فراخوانی) کنید :

```
Private Sub Form_Load()  
    frmAnotherForm.Show  
End Sub
```

این روش زمانی مفید است که تعداد فرم‌ها محدود باشد.

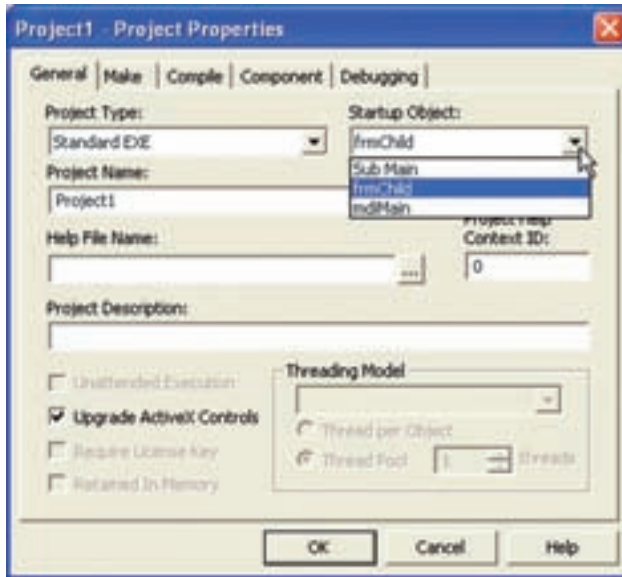
در پروژه‌هایی که دارای هیچ فرمی نیستند (مثل برنامه‌های اینترنتی که در روی سرور کار می‌کنند) و چیزی برای بارگذاری وجود ندارد چه کاری باید برای نقطه شروع (نقطه ورودی) برنامه انجام دهید؟

ویژوال بیسیک، یک نقطه شروع غیر مبتنی بر فرم را برای برنامه ارایه می‌کند (روال Sub Main()). Sub Main() روال خاصی است که به وسیله ویژوال بیسیک به عنوان روال شروع هر پروژه‌ای رزرو شده است. Sub Main() باید در یک مدول اعلان شود و برای هر پروژه فقط می‌توان یک Sub Main() در نظر گرفت. مراحل زیر، چگونگی تعیین این روال به عنوان نقطه شروع را نشان می‌دهند :

۱- از منوی Project گزینه Properties مربوط به پروژه جاری را انتخاب کنید تا کادر محاوره‌ای مربوطه باز شود.

۲- در سر برگ General از لیست بازشوی Startup Object گزینه Sub Main را انتخاب کنید.

۳- روی Ok کلیک کنید.



شکل ۱-۵-۱ می‌توان (Sub Main) یا هر فرم دیگری را به‌عنوان شیء شروع در پروژه انتخاب کرد.

بعد از تعیین (Sub Main) به عنوان شیء شروع پروژه، باید (Sub Main) را در مدول ایجاد کنید. می‌توان از کادر محاوره‌ای Add Procedure برای ایجاد روال‌ها استفاده کرد یا اعلان را در بخش General مدول انتخابی وارد کرد. به خاطر داشته باشید که یک پروژه فقط می‌تواند یک (Sub Main) داشته باشد. بعد از ایجاد (Sub Main)، نیاز به نوشتن کد Startup دارید. کد زیر یک (Sub Main) را نشان می‌دهد که دو فرم را با استفاده از متد Show نمایش می‌دهد و سپس بعد از این که همه فرم‌ها ظاهر شدند، کادر پیامی را نشان می‌دهد.

Sub Main()

frmMain.Show

frmOther.Show

MsgBox "Everything shown"

End Sub

## ۱-۱۰-۱- توابع تشخیص نوع داده

این توابع نوع داده آرگومان ورودی را مشخص می‌کنند. برنامه‌ها معمولاً با داده‌های مختلفی سروکار دارند، ولی گاهی برنامه‌نویس از قبل نمی‌تواند حدس بزند که با چه نوع داده‌ای سروکار خواهد

داشت. مثلاً، قبل از آن که محاسبه‌ای انجام دهد باید مطمئن شود که داده‌ها از نوع عددی هستند. در جدول ۱-۱ توابع Is....() را مشاهده می‌کنید؛ آرگومان این توابع همگی از نوع Variant است.

جدول ۱-۱- توابع تشخیص نوع داده

تابع	مفهوم
IsDate()	آیا آرگومان تاریخ (یا قابل تبدیل به تاریخ) است؟
IsEmpty()	آیا آرگومان مقدار گرفته؟
IsNull()	آیا آرگومان مقدار Null دارد؟
IsNumeric()	آیا آرگومان یک عدد است (یا می‌تواند به عدد تبدیل شود)؟

در ادامه برای هر یک از توابع مثالی ذکر شده است.  
در برنامه زیر، چگونگی استفاده از تابع IsEmpty() نشان داده شده است.

```

'Code that tests the Is() functions
Dim var1 As Variant, var2 As Variant
Dim var3 As Variant, var4 As Variant
Dim intMsg As Integer    'MsgBox return
'Fill variables with sample values to test
var1 = 0    'Zero value
var2 = Null 'Null value
var3 = ""   'Null string
'Call each Is() function
If IsEmpty(var1) Then
    intMsg = MsgBox("var1 is empty", vbOKOnly)
End If
    
```

```


If IsEmpty(var2)Then
    intMsg = MsgBox("var2 is empty", vbOKOnly)
End If
If IsEmpty(var3)Then
    intMsg = MsgBox("var3 is empty", vbOKOnly)
End If
If IsEmpty(var4)Then
    intMsg = MsgBox("var4 is empty", vbOKOnly)
End If

```

برنامه فوق پس از اجرا شدن، خروجی زیر را نمایش خواهد داد :

var4 is empty

چون تمام متغیرهای دیگر مقدار گرفته‌اند (حتی Null هم یک مقدار محسوب می‌شود). برای تست کردن Null می‌توانید از IsNull() استفاده کنید.

 **نکته:** توجه داشته باشید که شرط زیر :

```
If(varA = Null) Then . . . .
```

حتی اگر متغیر varA واقعاً Null باشد، True نخواهد شد. در این موارد تنها راه حل استفاده از تابع IsNull() است.

به قطعه کد زیر توجه کنید :

```

If IsNull(txtHoursWorked) Then
    intMsg = MsgBox("You didn't enter hours worked!", vbOKOnly)
Else
    "Thank them for the good hours
    intMsg = MsgBox("Thanks for entering hours worked!", vbOKOnly)
End If

```

در این جا برنامه قبل از ادامه کار، خالی نبودن یکی از فیلدهای برنامه (txtHoursWorked)

را بررسی می‌کند.

تابع () IsNumeric با آرگومان‌های عددی (یا هر چیزی که قابل تبدیل به یک عدد باشد) مقدار True را برخواهد گرداند. مقادیر عددی عبارت‌اند از:

- Empty (به صفر تبدیل می‌شود)

- اعداد صحیح (Integer)

- اعداد صحیح بلند (Long)

- اعداد اعشاری (Single)

- اعداد اعشاری با دقت مضاعف (Double)

- واحد پول (Currency)

- تاریخ

- رشته (اگر شبیه یک عدد باشد)

قطعه کد زیر، سن کاربر را (در یک متغیر Variant) گرفته و در صورت پاسخ اشتباه کاربر، به وی اخطار می‌دهد:

```
Dim varAge As Variant
Dim intMsg As Integer
varAge = InputBox("How old are you?", "Get Your Age")
If IsNumeric(varAge) Then
    intMsg = MsgBox("Thanks!", vbOKOnly)
Else
    intMsg = MsgBox("What are you trying to hide?", _
        vbOKOnly+vbQuestion)
End If
```

درستی پاسخ کاربر در خط ۴ بررسی می‌شود.

اگر می‌خواهید نوع یک متغیر را بدانید، باید از تابع () VarType استفاده کنید. جدول ۱-۲ مقادیر برگشتی این تابع را نشان می‌دهد.



جدول ۱-۲ - مقادیر برگشتی تابع VarType()

نوع داده	ثابت نام دار	مقدار برگشتی
Empty	vbEmpty	0
Null	vbNull	1
Integer	vbInteger	2
Long	vbLong	3
Single	vbSingle	4
Double	vbDouble	5
Currency	vbCurrency	6
Date	vbDate	7
String	vbString	8
Object	vbObject	9
یک مقدار خطا	vbError	10
Boolean	VbBoolean	11
Variant	vbVariant	12
یک شیء دسترسی داده	vbDataObject	13
Decimal	vbDecimal	14
Byte	vbByte	17
یک آرایه	vbArray	8194

در برنامه زیر، با دستور Select Case نوع داده ارسال شده به تابع مشخص شده است.

```
Private Sub PrntType(varA)
```

```
Dim intMsg As Integer
```

```
Select Case VarType(varA)
```

Case 0

```
intMsg = MsgBox("The argument is Empty")
```

Case 1

```
intMsg = MsgBox("The argument is null")
```

Case 2

```
intMsg = MsgBox("The argument is Integer")
```

Case 3

```
intMsg = MsgBox("The argument is Long")
```

Case 4

```
intMsg = MsgBox("The argument is Single")
```

Case 5

```
intMsg = MsgBox("The argument is Double")
```

Case 6

```
intMsg = MsgBox("The argument is Currency")
```

Case 7

```
intMsg = MsgBox("The argument is Date")
```

Case 8

```
intMsg = MsgBox("The argument is String")
```

Case 9

```
intMsg = MsgBox("The argument is Object")
```

Case 10

```
intMsg = MsgBox("The argument is Error")
```

Case 11

```
intMsg = MsgBox("The argument is Boolean")
```

Case 12

```
intMsg = MsgBox("The argument is a Variant Array")
```

Case 13

```
intMsg = MsgBox("The argument is a Data Access Object")
```

Case 14

```
intMsg = MsgBox("The argument is Decimal")
```

Case 17

```
intMsg = MsgBox("The argument is Byte")
```

Case 8194

```
intMsg = MsgBox("The argument is Array")
```

End Select

End Sub

## ۱-۱-۱- توابع تبدیل نوع


در جدول زیر، توابع تبدیل نوع را مشاهده می‌کنید؛ به حرف C (سرنام کلمه Convert) در اول نام این توابع دقت کنید. هر تابع آرگومان خود را از نوعی به نوع دیگر تبدیل می‌کند. توجه دارید که این توابع در صورتی می‌توانند به درستی عمل کنند که امکان تبدیل نوع وجود داشته باشد. مثلاً، عدد ۱۲۳۴۵۶۷۸۹ اساساً امکان تبدیل به نوع Byte را ندارد چون بزرگ‌ترین عددی که یک متغیر Byte می‌تواند در خود ذخیره کند ۲۵۵ است. برخلاف Int() و Fix()، تابع Cint آرگومان خود را به نزدیک‌ترین عدد صحیح گرد می‌کند. به مثال‌های زیر توجه کنید :

```
intA1 = Cint(8.5)           'Stores an 8 in intA1
```

```
intA2 = Cint (8.5001)      'Stores an 9 in intA2
```

```
intA3 = Cint (9.5)         Stores an 10 in intA3
```

چون توابع تبدیل نوع می‌توانند روی عبارات هم عمل کنند، می‌توانید حاصل محاسبات را قبل از ذخیره در متغیرها به نوع مناسب تبدیل کنید.

 **نکته:** اگر آرگومان تابع Cint دارای مقدار اعشار 0.5 باشد آن را به نزدیک‌ترین عدد زوج گرد می‌کند.

### جدول ۱-۳- توابع تبدیل نوع

تابع	مفهوم
CBool()	آرگومان خود را به نوع Boolean تبدیل می‌کند.
CByte()	آرگومان خود را به نوع Byte تبدیل می‌کند.
CCur()	آرگومان خود را به نوع Currency تبدیل می‌کند.
CDate()	آرگومان خود را به نوع Date تبدیل می‌کند.
CDbl()	آرگومان خود را به نوع Double تبدیل می‌کند.
CDec()	آرگومان خود را به نوع Decimal تبدیل می‌کند.
CInt()	آرگومان خود را به نوع Integer تبدیل می‌کند.
CLng()	آرگومان خود را به نوع Long تبدیل می‌کند.
CSng()	آرگومان خود را به نوع Single تبدیل می‌کند.
CStr()	آرگومان خود را به نوع String تبدیل می‌کند.
CVar()	آرگومان خود را به نوع Variant تبدیل می‌کند.

### ۱-۱۲- تابع Array

تابع Array یک آرایه از نوع Variant را در زمان اجرا، ایجاد کرده و برمی‌گرداند. مرز پایین آرایه بازگشت داده شده، بستگی به Option Base با مقدار ۰ یا ۱ دارد. برنامه زیر چگونگی استفاده از تابع Array را نشان می‌دهد.

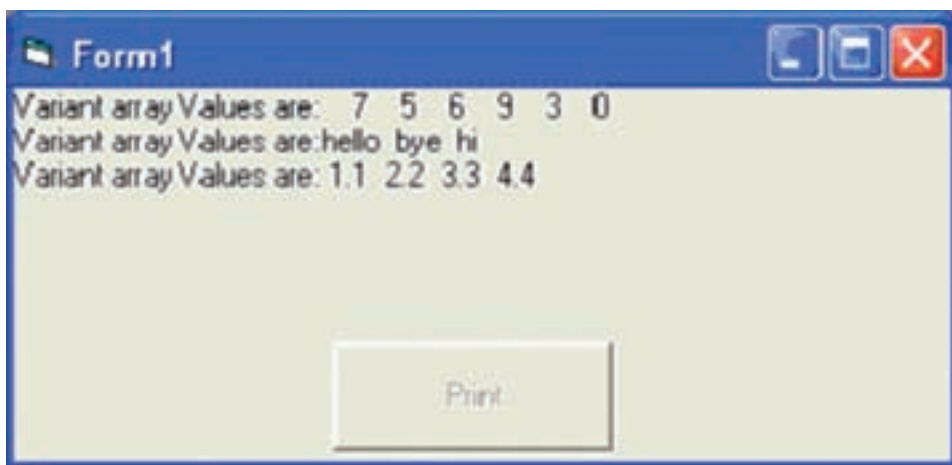
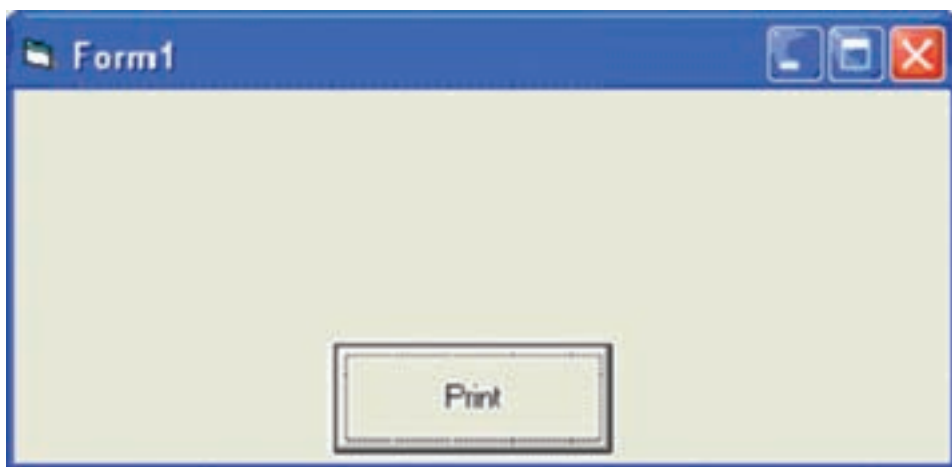
1. 'Demonstrating function Array
2. Option Explicit 'General declaration
3. Option Base 1 'General declaration

```

4. Private sub cmdPrint_Click()
5. Dim v As Variant, x As Integer
6. V = Array (7,5,6,9,3,0)
7. Print "Variant array Values are: ";
8. For x = LBound(v) To UBound(v)
9. Print Format$(v(x),"@@@");
10. Next x
11. Print
12. V = Array("hello","bye","hi")
13. Print"Variant array Values are.";
14. For x = LBound(v) To UBound(v)
15. Print v(x) Space$(2);
16. Next x
17. Print
18. V = Array (1.1,2.2,3.3,4.4)
19. Print"Variant array Values are: ";
20. For x = LBound(v) To UBound(v)
21. Print v(x) & Space$(2);
22. Next x
23. cmdPrint.Enabled = False
24. End Sub

```

در برنامه فوق آرایه V با سه نوع داده متفاوت (Double، String، Integer) ایجاد و چاپ شده است و برای جلوگیری از امکان چاپ مجدد در انتها کنترل CmdPrint را غیرفعال نموده است.



شکل ۶-۱- استفاده از تابع Array()

## خودآزمایی

- ۱- برنامه‌ای بنویسید که دو عدد را دریافت کند و با استفاده از یک تابع، عدد کوچک را به توان عدد بزرگ برساند.
- ۲- برنامه‌ای بنویسید که عددی را دریافت کند و فاکتوریل آن را به کمک یک تابع محاسبه کند و نمایش دهد.
- ۳- برنامه‌ای بنویسید که عددی چندرقمی را دریافت کند و با استفاده از یک روال، مجموع ارقام آن عدد را محاسبه کند و نمایش دهد.
- ۴- برنامه‌ای بنویسید که مضارب معادله درجه ۲ را دریافت کند و با استفاده از یک روال، ریشه‌های آن را به دست آورده و نمایش دهد.
- ۵- برنامه‌ای بنویسید که عددی را دریافت کند و با استفاده از یک روال، مقسوم‌علیه‌های آن را به دست آورد و نمایش دهد.
- ۶- برنامه‌ای بنویسید که رشته‌ای را دریافت کند و تعداد فضاهای خالی آن را به کمک یک روال، شمارش کند.