BAB V
PROSEDUR DAN FUNGSI

Modul-modul dalam Visual Basic dibuat oleh suatu bagian/seksi deklarasi dimana anda mendeklarasikan tipe, konstan, dan variabel yang digunakan dalam modul serta kumpulan dari prosedur-prosedur.
Prosedur atau rutin dapat berupa Sub (Procedure) atau Function, tergantung bagaimana mereka mengembalikan nilai ke pemanggil.
Setiap prosedur mempunyai nama yang unik, jangkauan, dan daftar argumen yang diinginkan, dan jika merupakan sebuah fungsi, maka akan mengembalikan nilai.

SCOPE (JANGKAUAN)

Scope (jangkauan) dari suatu prosedur bisa merupakan Private, Public atau Friend. Sebuah prosedur Private dapat dipanggil hanya dari dalam modul dimana dia didefinisikan. Sebuah prosedur Public dapat dipanggil dari luar modul. Dikarenakan Public adalah atribut jangkauan default untuk prosedur, anda bisa mengabaikannya.

```
' Fungsi Public yang dapat diakses dari form mananpun dalam satu project.
Function Tambah(a As Integer, b As Integer) As Integer
   Tambah = a + b
End Function
```

Jika jangkauannya bukan *Public*, maka anda harus menentukannya :

```
' Semua prosedur Event (kejadian) bersifat Private.
Private Sub Form_Load()
   txtTotal.Text = ""
End Sub
```

Jangkauan dari suatu prosedur Friend adalah pertengahan di antara Private dan Public: seperti suatu prosedur yang dapat dipanggil dari manapun dalam Project yang bersangkutan, tapi tidak dari luar. Perbedaan ini sangat penting hanya jika anda membangun Project dengan tipe selain dari Standard EXE.
Jika anda bekerja dalam Project Standard EXE, atau dalam Private Class dari tipe Project apapun, maka atribut Friend dan Public adalah sama. Hal itu disebabkan prosedur tersebut tidak bisa dipanggil dari luar.

DAFTAR PARAMETER & PENGEMBALIAN NILAI

Baik rutin Sub atau Function dapat menerima argumen. Khusus fungsi, dia juga mengembalikan nilai. Anda bisa melewatkan suatu prosedur secara mudah dengan tpe-tipe data yang didukung oleh Visual Basic, meliputi : Integer, Boolean, Long, Byte, Single, dsb. Anda juga bisa mendeklarasikan parameter sebuah Object, Collection, Class yang ditentukan dalam program. Anda juga dapat melewatkan parameter array.

Argumen By Value Atau By Reference

Sebuah argumen dapat dilewatkan berdasarkan nilai (menggunakan kata kunci By Value) atau berdasarkan referensi (menggunakan kata kunci By Reference atau dengan mengabaikan pembatasnya). Argumen yang melewatkan By Reference dapat dimodifikasi oleh prosedur yang memanggilnya, dan nilai yang diubah dapat dibaca kembali oleh pemanggil. Sebaliknya, mengganti argumen dengan By Value tidak akan pernah dipanggil kembali ke pemanggil.

{Contoh penggunaan ada di contoh *source code* halaman terakhir}

Mendeklarasikan Tipe Data untuk Argumen Fungsi

Di samping dapat menentukan tipe data dari nilai yang dihasilkan sebuah rutin (function), anda juga dapat menentukan tipe data argumen fungsi anda. Caranya anda dapat menggunakan kata kunci As dan diikuti dengan nama tipe datanya. Selain dari contoh yang sudah tertera di awal, berikut ini contoh lain dari sebuah fungsi :

```
Function LUAS_LINGKARAN (jari As Integer) As Double
   LUAS_LINGKARAN = (3.14 * jari * jari)
End Function
```

Membuat dan Mendefinisikan Kata Kunci *Optional*

Dalam sebuah fungsi mungkin anda perlu beberapa argumen dan itu tidak hanya satu. Di antara sekian banyak argumen atau parameter yang ada tersebut, mungkin anda ingin mengatur salah satu argumen atau parameter tersebut untuk diset ke ***Optional*** (boleh diisi, boleh juga tidak diisi). Jika anda ingin melakukan hal ini maka anda dapat memasukkan kata kunci Visual Basic ***Optional*** di depan nama parameter.
Dalam hal ini tentu anda harus mendeteksi dan mengatasi kemungkinan jika pengguna mengabaikan argumen ***optional*** ini seperti dalam contoh berikut :

```
Function LUAS_LINGKARAN (Optional jari As Integer) As Double
   If IsMissing (jari) Then            ' Jika variabel jari tidak diisi
      jari = 0                         ' maka variabel jari akan diisi 0 (nol)
   End If
   LUAS_LINGKARAN = (3.14 * jari * jari)
End Function
```

Argumen ParamArray

Anda dapat menerapkan sebuah rutin yang menerima sejumlah argumen apapun dengan menggunakan kata kunci *ParamArray* :

```
Function Jumlah (ParamArray args() As Variant) As Double
   Dim i As Integer
   ' Semua ParamArrays adalah berbasis 0 (nol)
   For i = 0 To UBound(args)
      Jumlah = Jumlah + args(i)
   Next
End Function
```

Anda dapat memanggil fungsi **Jumlah** sebagai berikut :

```
Print Jumlah(10, 30, 20)     ` Menampilkan "60"
```

Aturan penggunaan *ParamArray* :
- Hanya boleh menggunakan satu kata kunci *ParamArray*, dan itupun harus diakhir daftar parameter.
- Array yang dideklarasikan oleh kata kunci *ParamArray* hanya bisa diberikan tipe data *Variant*.
- Argumen *ParamArray* tidak mengenal (tidak didahului) parameter *Optional*.

ERROR HANDLING (PENANGANAN KESALAHAN)

Penanganan kesalahan merupakan fitur yang penting yang disediakan bahasa Visual Basic dan sangat berhubungan dengan struktur dari program yang anda buat. Visual Basic menawarkan tiga *statement* (pernyataan) yang memberikan anda kendali atas apa yang terjadi ketika terdapat *error* (kesalahan) saat pengeksekusian kode.

1. Pernyataan ***On Error Resume Next***.
   Visual Basic diperintahkan untuk mengabaikan kesalahan apapun. Saat suatu kesalahan benar-benar terjadi, maka Visual Basic akan melanjutkan untuk mengeksekusi kode atau pernyataan berikutnya.
2. Pernyataan ***On Error Goto <Label>.***
   Visual Basic diperintahkan, bila terjadi kesalahan, maka akan *jump* (lompat) ke *Label* yang telah dibuat (ditentukan). *Label* ini harus terletak pada prosedur yang sama di tempat dimana pernyataan tersebut muncul (dituliskan). Anda bisa menggunakan nama *label* yang sama dalam prosedur yang berbeda, karena jangkauan sebuah *label* hanya dalam prosedur, bukan modul.
3. Pernyataan ***On Error Goto 0***.
   Visual Basic diperintahkan untuk membatalkan semua efek yang disebabkan oleh pernyataan On Error Resume Next ataupun On Error Goto <label>. Saat kesalahan terjadi, maka Visual Basic akan berkelakuan sebagaimana *trapping error* (penjebakan kesalahan) tidak difungsikan (*disabled*).

Pemilihan salah satu bentuk dari penanganan kesalahan ini tergantung dari gaya pemrograman anda dan kebutuhan dari rutin secara spesifik. Jadi tidak ada aturan tertentu yang disediakan yang valid untuk setiap kasus. Semua pernyataan On Error akan membersihkan *error code* (kode kesalahan) yang sedang atau pernah terjadi.

Latihan 1
Contoh menggunakan argumen *ByValue* dan *ByRef*

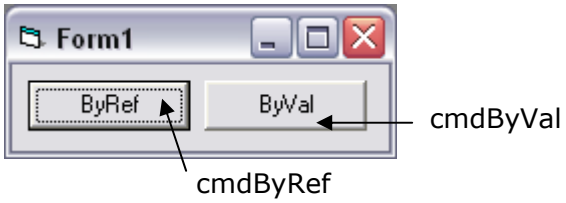
Komponen :

Komponen	Properti
CommandButton	Name = cmdByRef Caption = ByRef
CommandButton	Name = cmdByVal Caption = ByVal

Kode :

```
Option Explicit

Dim z As Integer

' Contoh prosedur dengan menggunakan ByRef
Private Sub KURANG_BYREF(x As Integer, y As Integer)
    x = x - y
    z = x
End Sub

' Contoh prosedur dengan menggunakan ByVal
Private Sub KURANG_BYVAL(ByVal x As Integer, y As Integer)
    x = x - y
    z = x
End Sub

Private Sub cmdByRef_Click()
    Dim a As Integer, b As Integer

    a = 5: b = 4
    Call KURANG_BYREF(a, b)
    MsgBox "Angka kiri = " & a & " dan nilai z = " & z
End Sub

Private Sub cmdByVal_Click()
    Dim a As Integer, b As Integer

    a = 5: b = 4
    Call KURANG_BYVAL(a, b)
    MsgBox "Angka kiri = " & a & " dan nilai z = " & z
End Sub
```

Tampilan.



Hasil penekanan tombol ByRef

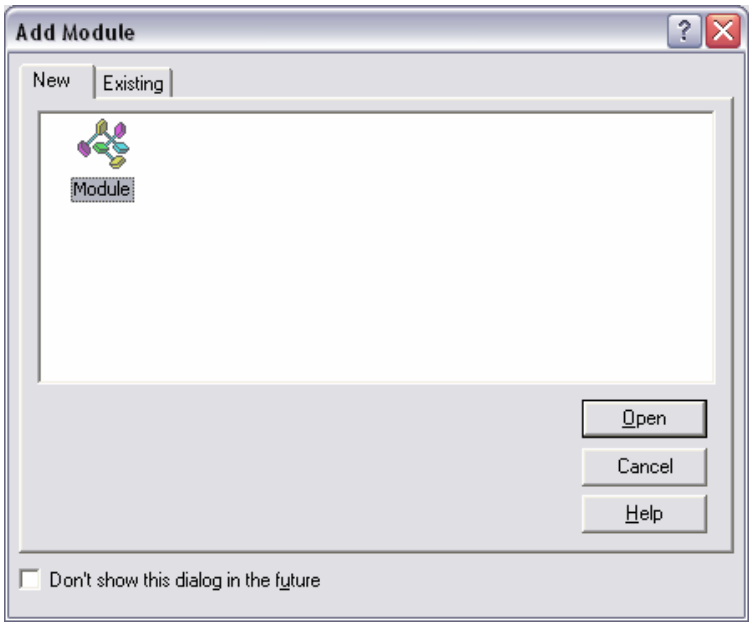
Hasil penekanan tombol ByVal

Latihan 2.
Pembuatan rutin dalam modul dan contoh argumen option.

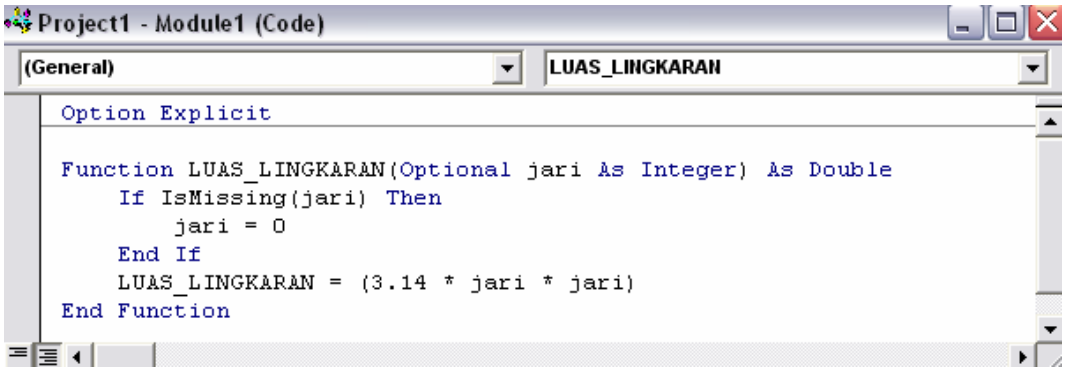
Komponen :
Hanya sebuah form dengan properti **AutoRedraw=True**

Kode :
Untuk membuat modul, langkah-langkahnya adalah sebagai berikut :

1. Tambahkan modul pilih menu **Project** > **Add Module** (Tekan **modul** kemudian **Open**)



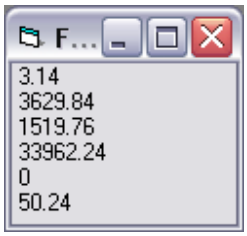
2. Kemudian dalam modul tersebut tambahkan kode berikut ini.



3. Dalam form utama tambahkan kode berikut.

```
Option Explicit

Private Sub Form_Load()
    Print LUAS_LINGKARAN(1)
    Print LUAS_LINGKARAN(34)
    Print LUAS_LINGKARAN(22)
    Print LUAS_LINGKARAN(104)
    Print LUAS_LINGKARAN()
    Print LUAS_LINGKARAN(4)
End Sub
```

Tampilan.



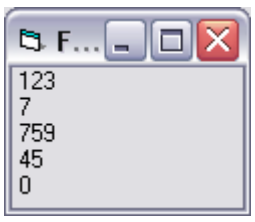
Latihan 3.
Penggunaan argumen *ParamArray*.

Komponen :
Sebuah form dengan properti **AutoRedraw=True**

Kode :

```
Option Explicit

Function Jumlah(ParamArray args() As Variant) As Double
    Dim i As Integer
    ' Semua ParamArrays adalah berbasis 0 (nol)
    For i = 0 To UBound(args)
        Jumlah = Jumlah + args(i)
    Next
End Function

Private Sub Form_Load()
    Print Jumlah(23, 43, 23, 34)
    Print Jumlah(3, 4)
    Print Jumlah(223, 413, 123)
    Print Jumlah(1, 2, 3, 4, 5, 6, 7, 8, 9)
    Print Jumlah()
End Sub
```

Tampilan.



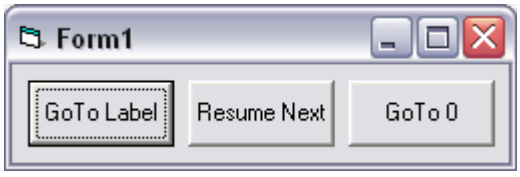
Latihan 4

Buat tiga buah tombol (*CommandButton*)

Kode :

```
Option Explicit

Private Sub Command1_Click()
    On Error GoTo salah      ' ← jenis error handling
    Dim a As Integer
    a = "a"
    MsgBox a
    Exit Sub
salah:
    MsgBox Err.Number & vbNewLine & Err.Description & vbNewLine & Err.Source
End Sub

Private Sub Command2_Click()
    On Error Resume Next     ' ← jenis error handling
    Dim a As Integer
    a = "a"
    MsgBox a
    Exit Sub
salah:
    MsgBox Err.Number & vbNewLine & Err.Description & vbNewLine & Err.Source
End Sub

Private Sub Command3_Click()
    On Error GoTo 0      ' ← jenis error handling
    Dim a As Integer
    a = "a"
    MsgBox a
    Exit Sub
salah:
    MsgBox Err.Number & vbNewLine & Err.Description & vbNewLine & Err.Source
End Sub
```

Tampilan



Hasil penekanan tombol **GoTo Label**

Hasil penekanan tombol **Resume Next**

Hasil penekanan tombol **GoTo 0**