Visual Basic Dasar

By ; Gapra. Email ga_pra_27@yahoo.c0.id

Bagian pertama ini akan menguraikan hal-hal yang sifatnya esensial dalam pemrograman Visual Basic. Meskipun dikatakan esensial, namun kita tidak boleh menyepelekan begitu saja. Pada kenyataannya, teknik-teknik dasar yang mungkin dianggap sepele seringkali mampu menyelesaikan berbagai persoalan pelik. Oleh karena itu, untuk memudahkan pembuatan program tingkat lanjut, akan lebih baik jika kita sudah memiliki bekal dasar.

1. Teknik Dasar Visual Basic

Jika kita membahas teknik dasar Visual Basic, sebenarnya tidak cukup hanya dengan satu bab, apalagi satu subbab. Namun, di sini akan dibatasi agar uraiannya tidak terlalu umum ataupun khusus. Artinya, bagian ini mencoba menguraikan hal-hal ringan yang dianggap penting dan sering diperlukan.

- **Awalan Penamaan Variabel**

Konsistensi sering dikaitkan dengan penulisan kode program yang standar. Ini tentu cukup beralasan karena penulisan kode yang konsisten menjadikan kode program lebih mudah dibaca dan dipahami. Sebagai contoh, dalam penamaan variabel, sangat dianjurkan agar Anda mengacu pada standar yang telah ditetapkan.

Contohnya seperti berikut:

```
' Jika hanya ada satu variabel String di satu
' ruang lingkup
Dim str As String
```

```
' Jika terdapat lebih dari satu
Dim strNama As String
Dim strAlamat As String
```

Daftar standar penamaan selengkapnya untuk variabel yang merepresentasikan suatu tipe data, kontrol-kontrol *built-in* dan ActiveX, serta objek-objek ADO bisa Anda lihat di lampiran.

- **Hindari Tipe Variant**

Tipe data Variant secara otomatis akan ditetapkan pada suatu variabel manakala Anda tidak menetapkan tipe datanya secara eksplisit.

```
' Variabel intX bertipe Variant
Dim intX, intY As Integer
' intA dan intB bertipe Integer
Dim intA As Integer, intB As Integer
```

Alasan utama untuk menghindari tipe data Variant adalah karena eksekusinya yang cenderung lebih lambat dan mengonsumsi memori lebih besar.

- **Ruang Lingkup Variabel**

Apabila variabel-variabel Anda hanya diperlukan oleh sebuah method (prosedur/fungsi), sebaiknya deklarasikan di level lokal. Teknik ini terkait erat dengan performa aplikasi, di mana variabel lokal akan lebih cepat dieksekusi dibanding variabel global.

- **Konkatenasi dan Multiline**

Pada saat Anda ingin menghubungkan string, gunakan operator konkatenasi (&). Adapun jika konkatenasi melibatkan baris baru, sebaiknya manfaatkan karakter garis bawah (*underscore*).

```
' Hello Indonesia adalah satu bagian
strData = "Hello " & _
"Indonesia"
' strA dan strB adalah dua bagian
```

```
Dim strA As String, _
strB As String
```

- **Definisi Tipe Data**

Anda pasti pernah menemui kode program Visual Basic yang mendefinisikan suatu tipe data menggunakan akhiran karakter tertentu, seperti $, %, dan sebagainya. Pendekatan yang juga disebut definisi tipe data secara implisit ini umumnya dilakukan untuk mempercepat eksekusi kode. Berikut ini simbol (karakter) yang dapat digunakan sebagai akhiran nama variabel untuk merepresentasikan tipe data.

Akhiran	Tipe Data	Contoh
%	Integer	IntUsia%
&	Long	Ing&
!	Single	Sng!
#	Double	Dbl#
@	Currency	Cur@
$	String	str$

Perlu diperhatikan, bagian ini tidak bermaksud menyarankan Anda untuk menggunakan pendekatan di atas, namun sekadar memperlihatkan bagaimana implementasi definisi secara implisit. Bagaimanapun juga, pendekatan seperti ini seringkali membingungkan, terutama bagi yang belum memahami makna karakter terkait.

- **Verifikasi Tipe Data**

Pada saat akan melakukan operasi terhadap suatu variabel, terkadang kita ingin memeriksa tipe datanya terlebih dahulu. Untuk memudahkan verifikasi ini, kita bisa memanfaatkan method-method yang tersedia.

Method	Keterangan (Nilai Kembalian)
IsArray()	**True jika variabel adalah suatu array**
IsDate()	**True jika ekspresi dapat dikonversi ke format tanggal**
IsEmpty()	**True jika variabel belum diinisialisasi**
IsError()	**True jika ekspresi merupakan nilai error**
IsMissing()	**True jika argumen opsional belum dilewatkan**
IsNull()	**True jika ekspresi null**
IsNumeric()	**True jika ekspresi dapat dievaluasi sebagai bilangan**
IsObject()	**True jika identifier merepresentasikan variabel objek**

- **Pernyataan Kondisional**

Dalam melakukan analisis dua kasus, Anda bisa meringkas penulisan kode program dengan memanfaatkan method IIf.

```
Dim bStatus As Boolean
Dim intResult As Integer
' Pendekatan If (standar)
If bStatus Then
intResult = 1
Else
intResult = 0
End If
```

```
Print intResult
' Pendekatan IIf
intResult = IIf(bStatus, 1, 0)
Print intResult
```

Namun perlu diperhatikan, eksekusi method IIf cenderung lebih lambat dibanding pernyataan kondisional If.

- **Keyword Step**

Disamping menyederhanakan penulisan kode program pengulangan, keyword Step juga mampu mempercepat eksekusi kode. Sebagai contoh, kedua program berikut akan menghasilkan keluaran sama.

```
Dim i As Integer
' Mencetak bilangan ganjil 1-10
For i = 0 To 9
i = i + 1
Print i
Next i
' Menggunakan pendekatan Step
For i = 1 To 10 Step 2
Print i
Next i
```

- **Keyword Optional**

Apabila Anda ingin mendeklarasikan suatu argumen yang sifatnya opsional (tidak wajib), gunakan keyword Optional.

```
' Deklarasi argumen lng sebagai opsi
Private Sub Test(str As String, _
Optional lng As Long = 1)
' ...
End Sub
```

Walaupun Anda diperkenankan untuk tidak menetapkan nilai default dari suatu argumen opsional, namun sebaiknya Anda menetapkannya. Langkah ini

bertujuan untuk memudahkan penulisan tubuh method dan meningkatkan kompatibilitas terhadap versi Visual Basic yang baru (VB.NET).

- **Array Argumen**

Kasus lain yang mungkin ditemui saat mendeklarasikan method adalah kemampuan untuk menerima sejumlah argumen. Kasus semacam ini bisa Anda selesaikan dengan memanfaatkan keyword ParamArray.

```
Private Function Sum(ParamArray args() _
As Variant) As Double
Dim i As Integer
' array argumen adalah array berbasis 0
For i = 0 To UBound(args)
Sum = Sum + args(i)
Next i
End Function
' Ekspresi berikut valid semua
Debug.Print Sum(2, 3)
Debug.Print Sum(2, 3, 4, 5)
```

Perlu diingat, ParamArray harus dideklarasikan sebagai array Variant, tidak boleh dengan tipe data lainnya. Selain itu, suatu method hanya boleh memiliki sebuah array argumen, dan harus berada di bagian paling akhir. Aturan terakhir, Anda tidak boleh meletakkan argumen opsional di depan array argumen.

2. Operasi String

Sudah bukan rahasia lagi, frekuensi penggunaan tipe data String sangat tinggi sekali. Ini tentu tidak lepas dari sifat dasarnya yang ìsederhanaî dan memiliki sejumlah operasi yang sangat bermanfaat. Oleh karena itu, tidak ada salahnya jika kita mengenal operasi-operasi string lebih lanjut.

- **Mencetak Spasi/Karakter**

Bagaimana mencetak 10 karakter kosong dengan cepat? Jika Anda sudah mengenal method Space, tentu tidak akan kesulitan melakukannya.

```
' Mencetak 10 spasi antara a dan b
Print "a" & Space$(10) & "b"
```

Sebagai alternatif, Anda juga bisa memanfaatkan method String untuk tujuan serupa. Selain itu, method ini lebih luas fungsionalitasnya.

```
' Mencetak 10 spasi antara a dan b
Print "a" & String$(10, " ") & "b"
' Sama, 32 adalah kode ASCII spasi
Print "a" & String$(10, 32) & "b"
' Mencetak karakter * sebanyak 10
Print String$(10, "*")
```

- **Mencari Substring**

Mencari bagian-bagian string merupakan operasi sederhana yang implementasinya terkadang tidak sesederhana bayangan kita. Berikut tip-tip ringan untuk menemukan substring di suatu string.

```
strText = "abcdefg"
Print Left$(strText, 3) ' Output: abc
Print Right$(strText, 2) ' Output: fg
' Mengambil 4 karakter, dimulai dari karakter ke 3
Print Mid$(strText, 3, 4) ' Output: cdef
```

Anda juga bisa memanfaatkan method Mid untuk memodifikasi karakter di dalam string.

```
' Replace strText mulai karakter ke 3
Mid$(strText, 3) = "123"
Print strText
' Output: ab123fg
```

- **Pemformatan String**

Pemformatan string seringkali diperlukan guna mendapatkan format nilai

yang sesuai. Bergantung pada kebutuhan, ada beberapa pendekatan yang bisa kita lakukan.

```
Dim str As String
str = "nadin savitri"
' Format huruf besar dan kecil
Print UCase$(str)
Print LCase$(str)
' Konversi ke huruf semestinya
Print StrConv(str, vbProperCase)
' Output: Nadin Savitri
```

Selain mengatur format huruf, Anda juga bisa menetapkan format angka.

```
Dim sng As Single: sng = 2.246
' Pembulatan dan pemformatan
Print Format$(sng, "0.00") ' Output: 2,25
Print FormatNumber$(sng, 2) ' Output: 2,25
' Pembulatan dan Format currency
Print FormatCurrency$(sng, 2) ' Output: Rp2,25
```

3. Konversi

Dalam pembuatan aplikasi perangkat lunak, seringkali kita perlu melakukan konversi, baik itu konversi tipe data maupun konversi nilai. Untuk beberapa tujuan, operasi ini dapat kita lakukan dengan mudah melalui method-method *built-in* yang disediakan oleh class Conversion.

Tabel berikut ini memperlihatkan keyword dan method untuk konversi berdasarkan kegunaannya.

Aksi Konversi	**Keyword/Method**
Nilai ANSI ke string	Chr, ChrW
String ke huruf kecil/besar	Format, LCase, UCase

Angka ke string	Format, Str
Satu tipe data ke tipe data lain	CBool, CByte, CDate, CDbl, CDec, CInt, CLng, CSng, CShort, CStr, CVar, Fix, Int, Hex, Oct
Tipe Date ke tanggal, bulan, atau tahun	Day, Month, Year
Tipe Time ke jam, menit, atau detik	Hour, Minute, Second
String ke nilai ASCII	Asc, AscW
String ke angka	Val

Sebagai contoh, untuk mengkonversi nilai string ke angka, gunakan method Val. Sebaliknya, manfaatkan method Str untuk mengembalikan representasi string dari angka.

```
Print Val("12A") ' Output: 12
Print Str$("12") ' Output: 12
```

Apabila Anda ingin mendapatkan nilai ANSI dari suatu kode karakter, gunakan method Chr atau ChrW. Adapun jika ingin mendapatkan nilai ASCII, manfaatkan method Asc atau AscW.

```
Print ChrW$(65) ' Output: A
Print AscW("A") ' Output: 65
```

Begitu pula halnya ketika Anda ingin mendapatkan nilai heksadesimal dari tipe data desimal atau sebaliknya, gunakan method Hex dan Val.

```
' Konversi Hexadecimal ke Decimal
Me.txtDec.Text = Val("&H" & Me.txtHex.Text)
' Konversi Decimal ke Hexadecimal
Me.txtHex.Text = Hex$(Me.txtDec.Text)
```

4. Validasi String dan Angka

Pada prinsipnya, string mampu menampung berbagai jenis karakter sehingga fleksibel dalam penggunaannya. Namun dibalik itu, diperlukan pekerjaan ekstra guna memperoleh data seperti yang diharapkan. Sebagai contoh, bagaimana memastikan bahwa user hanya mengisi huruf, angka, atau kombinasi keduanya? Sebagai tindakan lanjut, diperlukan langkah efektif, yaitu validasi data. Dalam implementasinya, validasi data bisa kita lakukan melalui berbagai pendekatan.

- **Validasi Input Huruf**

Untuk memastikan bahwa string masukan yang diberikan oleh user sudah valid, kita bisa memanfaatkan operator Like. Operator ini berfungsi mengevaluasi string yang diberikan dan memeriksa apakah sama dengan pola yang telah dispesifikasikan atau tidak.

```
Private Function IsHuruf(ByVal str As _
String) As Boolean
Dim strChar As String
Dim iLen As Integer, iCount As Integer
iLen = Len(str)
If iLen > 0 Then
' Loop sampai panjang string
For iCount = 1 To iLen
' Ambil karakter satu per satu
strChar = Mid$(str, iCount, 1)
' Jika tidak sesuai pola, berarti False
If Not strChar Like "[A-Za-z]" Then _
Exit Function
' Jika ingin menyertakan spasi,
' tambahkan di pola, misal [A-Z a-z]
Next iCount
IsHuruf = True
```

```
End If
End Function
```

- **Validasi Input Angka**

Dalam kasus ini, sebenarnya Anda bisa memanfaatkan method *built-in*, yaitu IsNumeric. Namun perlu diperhatikan, IsNumeric juga akan mengembalikan nilai true jika suatu ekspresi masih merepresentasikan angka, misalnya -1, +1, atau 1.1.

Apabila Anda hanya ingin menerima masukan angka (bilangan bulat), IsNumeric tentu kurang tepat digunakan. Sebagai solusinya, Anda bisa menggunakan pendekatan operator Like, seperti kasus sebelumnya. Adapun pola untuk angka adalah [0-9], contohnya seperti berikut:

```
Private Function IsAngka(ByVal str As _
String) As Boolean
Dim strChar As String
Dim iLen As Integer, iCount As Integer
iLen = Len(str)
If iLen > 0 Then
For iCount = 1 To iLen
strChar = Mid$(str, iCount, 1)
If Not strChar Like "[0-9]" Then _
Exit Function
Next iCount
IsAngka = True
End If
End Function
```

- **Kombinasi Huruf dan Angka**

Mengacu pada dua contoh validasi sebelumnya, tentu sangat mudah bagi Anda untuk mendapatkan nilai berupa kombinasi huruf dan angka. Lebih jelasnya, Anda tinggal menetapkan pola huruf dan angka, yaitu [0-9A-Za-z].

5. Operasi Array

Array merupakan tipe data agregat yang mampu menampung sejumlah variabel. Karakteristiknya yang fleksibel sangat membantu kita dalam mengelola koleksi item (elemen). Terkait hal ini, ada banyak operasi penting yang lazimnya dilakukan terhadap koleksi item di dalam array.

- **Split dan Join**

Disamping mengisi elemen array satu per satu (berbasis index), Anda bisa memecah bagian-bagian string kemudian memasukkannya ke dalam array. Untuk melakukan hal ini, manfaatkan fungsionalitas method Split. Adapun sebaliknya, untuk menggabung elemen array gunakan method Join.

```
Dim str As String
str = "Satu Dua Tiga"
Dim strArr() As String
' Memisah string, assign ke array
strArr = Split(str)
' Gabung elemen array, delimeter = spasi
Print Join$(strArr, " ")
```

- **Iterasi Elemen**

Iterasi merupakan salah satu operasi dasar yang sering dilakukan untuk mendapatkan elemen-elemen array. Walaupun Anda dapat menggunakan pendekatan `For...Next`, namun akan lebih praktis jika memanfaatkan pernyataan `For Each...Next`.

```
Dim i As Integer
Dim intMax As Integer
' Mendapatkan index maksimum
intMax = UBound(strArr)
For i = 0 To intMax
Print strArr(i)
```

```
Next i
Dim vnt As Variant
For Each vnt In strArr
Print vnt
Next vnt
```

- **Pencarian Elemen**

Pada saat Anda ingin mencari suatu elemen di dalam array, Anda bisa menggunakan method Filter. Method ini sebenarnya berfungsi menyaring elemen, namun juga sangat

memungkinkan untuk digunakan dalam mencari elemen.

```
Dim strRes() As String
' Penyaringan case-insensitive
strRes = Filter(strArr, "dua", True, _
vbTextCompare)
Dim vnt As Variant
' Ekstrak elemen yang sesuai penyaringan
For Each vnt In strRes
Print vnt
Next vnt
```

6. Date dan Time

Operasi tanggal dan waktu tidak hanya sebatas mendapatkan tanggal dan waktu saat ini (*current date/time*). Anda mungkin juga perlu melakukan operasi-operasi lanjut, seperti mengembalikan tanggal/waktu, kalkulasi tanggal/waktu, dan sebagainya. Untuk lebih jelasnya, berikut tip dalam menyelesaikan operasi-operasi seputar tanggal dan waktu.

- **Mendapatkan Tanggal dan Waktu**

Untuk mendapatkan informasi mengenai tanggal dan waktu saat ini, lazimnya kita menggunakan method Now. Sebagai alternatif, Anda bisa

memanfaatkan properti Date dan Time untuk mendapatkan tanggal dan waktu.

```
' Mendapatkan current date dan time
Print Now ' Tanggal dan waktu
Print Date ' Tanggal
Print Time ' Waktu
```

- **Pembentukan dan Ekstraksi**

Apabila Anda tidak ingin mendapatkan tanggal yang lengkap dari pendekatan sebelumnya, Anda bisa membentuk sendiri komponen-komponen tanggal/waktu.

```
' Mendapatkan bagian (DateTime) tertentu
Print Day(Now) & "-" & Month(Now) & "-" & _
Year(Now)
Print Hour(Now) & ":" & Minute(Now)
```

Seperti halnya ketika membentuk komponen-komponen tanggal/waktu, ada beberapa pendekatan untuk mengekstrak tanggal dan waktu.

```
' Mendapatkan bagian (DateTime) tertentu
Print Day(Now) & "-" & Month(Now) & "-" & _
Year(Now)
Print Hour(Now) & ":" & Minute(Now)

Print DatePart("d", Now) ' d= Day
Print DatePart("m", Now) ' m= Month
Print DatePart("yyyy", Now) ' yyyy= Year
```

- **Kalkulasi Tanggal dan Waktu**

Bergantung pada kebutuhan Anda, ada beberapa pendekatan untuk mengalkulasi tanggal dan waktu. Sebagai contoh, untuk menambah atau mengurangi tanggal/waktu, gunakan method DateAdd.

```
' 3 hari kemudian dan yang lalu
```

```
Print DateAdd("d", 3, Now)
Print DateAdd("d", -3, Now)

' Untuk bulan=m, tahun=yyyy
' 3 jam kemudian
Print DateAdd("h", 3, Now)
' Untuk menit= n, detik= s
```

Jika Anda ingin mendapatkan informasi mengenai selisih tanggal/waktu, gunakan method DateDiff.

```
' Jumlah hari sejak 20 September 2006
Print DateDiff("d", #9/20/2006#, Now)
```

- **Pemformatan Tanggal dan Waktu**

Jika diperlukan, informasi tanggal/waktu yang telah Anda dapatkan bisa diformat sesuai dengan kebutuhan. Hal ini diwujudkan dengan memanfaatkan method Format. Print Format(Now, "dddd, dd mmm yyyy")

```
' Output: (misal) Sabtu, 29 September 2006
Print Format(Now, "hh:mm:ss")
' Output: 21:30:48
```

7. Timer dan Alternatifnya

Banyak contoh kasus yang memerlukan keberadaan komponen Timer untuk mendapatkan nilai interval waktu. Disamping mampu menghasilkan nilai yang akurat, komponen ini juga mudah sekali digunakan. Sekadar contoh sederhana, jika Anda ingin melakukan suatu operasi dalam rentang waktu lima detik setelah kode program berjalan, tentu Timer menjadi pilihan guna mewujudkannya. Contoh kode programnya seperti berikut:

```
Private intBatas As Integer
Private Sub Command1_Click()
' Misal batas waktu = 5 detik
```

```
intBatas = 5
' Interval = 1000 milidetik (1 detik)
Me.tmr.Interval = 1000
Me.tmr.Enabled = True
End Sub
Private Sub tmr_Timer()
' Decrement batas waktu
intBatas = intBatas - 1
If intBatas <= 0 Then
Me.tmr.Enabled = False
' Unload form
Unload Me
Else
' Menampilkan hitungan mundur
Me.Caption = "TimeOut: " & intBatas
End If
End Sub
```

Sebagai alternatif, untuk tujuan yang sama, Anda bisa memanfaatkan pendekatan method DateAdd dan pengulangan. Tekniknya, spesifikasikan batas waktu, kemudian lakukan pengulangan hingga waktu saat ini (*current time*) melewati batas waktu.

```
Private Sub Command2_Click()
intBatas = 5
Dim intTOut As Integer
Dim dtm As Date
dtm = DateAdd("s", intBatas, Now)
' Loop sampai batas terlewati
Do Until Now >= dtm
DoEvents
intTOut = Second(dtm) - Second(Now)
' Hitungan mundur
Me.Caption = "TimeOut: " & intTOut
```

```
Loop
' Unload form
Unload Me
End Sub
```

Alternatif lain yang bisa Anda gunakan adalah melalui pendekatan method Timer. Method *read-only* ini akan mengembalikan nilai (detik) Single yang telah lewat sejak waktu tengah malam.

8. Mengirim Keystroke

Dalam beberapa kasus spesifik, Anda mungkin ingin mengirim keystroke ke aplikasi. Sebagai contoh, secara otomatis memindahkan fokus ke kontrol lain ketika batas pengisian TextBox sudah maksimal, mengubah perilaku key, dan sebagainya.

Untuk mendukung operasi-operasi di atas, manfaatkan method SendKeys. Method ini berfungsi mengirim keystroke ke window yang saat itu sedang mendapatkan fokus.

```
Private Sub Form_Load()
' Misal max panjang karakter =5
Me.txt1.MaxLength = 5
End Sub
Private Sub txt1_Change()
' Jika panjang karakter sudah terpenuhi
If Len(Me.txt1.Text) = Me.txt1.MaxLength Then
' Memindahkan fokus ke kontrol berikutnya
SendKeys ("{TAB}")
End If
End Sub
```

Kunci utama dalam pengiriman keystroke adalah argumen dari method SendKeys, yakni kode key (tombol). Secara keseluruhan, kode key mengacu

pada label yang ada di keyboard, kecuali untuk key spesifik. Di mana kode untuk key **Shift** adalah **+**, **Ctrl** adalah **^**, dan **Alt** adalah **%**. Jadi, ketika Anda ingin mengirim kombinasi key **Alt+F4** (menutup window), gunakan key **ì%{F4}î**.

```
' Mengaktifkan window notepad
' window sudah harus terbuka
Call AppActivate("Untitled - Notepad")
' Menutup window notepad
SendKeys ("%{F4}")
```

9. Argumen Command Line

Jika Anda menghendaki, aplikasi Visual Basic yang Anda buat dapat memiliki kemampuan untuk menerima argumen dari command line. Seperti halnya program-program berbasis teks, Anda tinggal mendefinisikan argumen di command line setelah nama file executable program. Kunci utama aplikasi semacam ini adalah bagaimana mendapatkan informasi argumen, dan ini kita lakukan dengan memanfaatkan method Command. Setelah argumen berhasil didapatkan, Anda bisa menggunakannya untuk mengatur perilaku aplikasi atau untuk keperluan lainnya.

```
Private Sub Form_Load()
Dim strCmd As String
' Mendapatkan argumen dari command line
strCmd = Command$()
If InStrB(strCmd, "-status MAX") > 0 Then
' Jika argumen didapatkan,
' Misalkan men-set status window
Me.WindowState = 2
Me.Caption = "Argumen: " & strCmd
Else
```

```
Me.WindowState = 0
End If
End Sub
```

Untuk menjalankan kode program di atas, misalkan nama file executable adalah Project1.exe, buka command prompt dan berikan perintah **Project1.exe ñstatus MAX**.

10. Error Handling

Bagaimana implementasi penanganan kesalahan? Dan bagaimana mendapatkan informasi yang bermanfaat terkait dengan kesalahan yang terjadi? Apabila Anda ingin mendapatkan informasi detail mengenai kesalahan yang berhasil ditangkap, gunakan method Err.

```
Dim dblRes As Double

On Error GoTo ErrHandler
' Ini akan menimbulkan kesalahan
' sehingga perlu di-trap
dblRes = 10 / 0

' Jika error, ini tidak akan dieksekusi
MsgBox ("Hasil: " & dblRes)

' Jika sukses, keluar dari blok
Exit Sub

ErrHandler:
' Misalkan kode kesalahan diketahui
If (Err.Number = 11) Then
MsgBox ("Pembagian dengan nol" & vbCrLf & _
"Kode Kesalahan: " & Err.Number & vbCrLf & _
"Deskripsi: " & Err.Description & vbCrLf & _
```

```
"Sumber: " & Err.Source)
Else
MsgBox ("Kode Kesalahan: " & Err.Number & _
"Deskripsi: " & Err.Description & vbCrLf & _
"Sumber: " & Err.Source)
End If
```

Dalam menangani kesalahan, tentunya tidak semua bagian kode program harus diberi pernyataan On Error. Artinya, ada saatnya kita bisa memanfaatkan fungsionalitas method. Sebagai contoh sederhana, ketika ingin melakukan verifikasi terhadap suatu array, akan lebih baik menggunakan method IsArray dibanding mendefinisikan On Error sebelum kode verifikasi.

Bagaimanapun juga, mekanisme penanganan kesalahan memerlukan tambahan alokasi memori. Artinya, penggunaan pernyataan On Error berbanding lurus terhadap waktu eksekusi. Dengan kata lain, semakin banyak blok On Error di suatu kode program, maka juga akan semakin lambat eksekusinya.

11. API Win32

Bagian ini secara khusus akan menguraikan tip mengenal *Application Programming Interface* (API) Windows, khususnya di platform 32-bit (Win32). Ini dimaksudkan untuk lebih mendekatkan Anda dengan fungsionalitas API Win32 dan juga terkait pembahasan-pembahasan selanjutnya. Pada prinsipnya, API Win32 terdiri atas koleksi Dynamic- Link Library (DLL) yang disebut DLL standar. Cara kerja DLL standar ini berbeda dengan *Componen Object Model* (COM). Idealnya, saat mengeksekusi method aplikasi berbasis COM, kita menggunakan COM untuk pemanggilan prosedur yang tersedia. Di sisi lain, pada pendekatan DLL standar, kita melakukan pemanggilan method secara langsung.

- **Kelebihan API Win32**

Sedikitnya ada dua kelebihan yang bisa didapatkan dari pendekatan API Windows. Pertama, kita dapat memperluas fungsionalitas aplikasi, dan kedua adalah meningkatkan performa aplikasi. Pada kenyataannya, DLL standar mampu menghasilkan kode yang lebih cepat dan efisien.

- **Fungsionalitas API Win32**

API Win32 mengekspos tiga DLL utama yang mencakup beberapa fungsionalitas spesifik, yaitu User32, Kernel32, dan GDI32. DLL User32 (user32.dll) mengelola menu, kontrolkontrol, dan kotak dialog Windows. Kernel32 (kernel32.dll) menangani tugas-tugas sistem operasi dan operasi level rendah, seperti manajemen memori, file, proses, waktu, dan kontrol Comm. Sementara itu, DLL terakhir (gdi32.dll), menangani pemrosesan grafik/gambar.

- **Mendeklarasikan DLL**

Ada dua tahap untuk mengimplementasikan DLL standar, yaitu deklarasi dan pemanggilan method. Sintaks pernyataan deklarasi diperlihatkan seperti berikut:

```
[Public|Private] Declare Function nama_fungsi _
Lib "nama_library" [Alias "nama_alias"] _
[(argumen)]
```

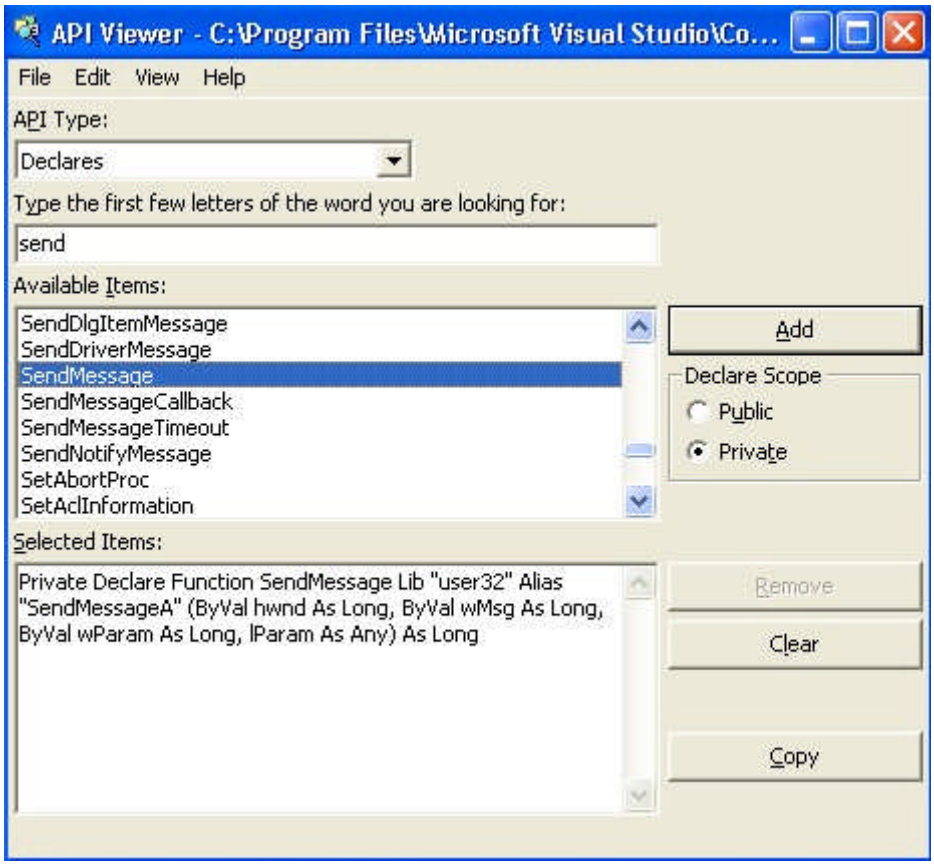
Apabila deklarasi dilakukan di level modul, gunakan access modifier Public, sedangkan untuk level form, sebaiknya gunakan Private.

- **Informasi dan Dokumentasi**

Walaupun API Win32 berpotensi besar untuk menghasilkan aplikasi yang kompleks, namun dokumentasinya sangat minim. Di Visual Basic, Anda bisa memanfaatkan tool **API Text Viewer** untuk melihat daftar method, tipe, serta konstanta yang tersedia. Shortcut tool ini tersedia di submenu

Microsoft Visual Basic 6.0 Tools, atau bisa juga diakses dari **Add-in Manager**.

Apabila Anda sangat tertarik mendalami API Win32 dan kegunaannya, Anda bisa mendapatkan dokumentasi API Win32 (Win32 Programmerís Reference) alamat ftp.borland.com/pub/delphi/techpubs/delphi2/win32.zip. Dokumentasi ini lebih detail karena menyertakan deskripsi serta beberapa informasi tambahan.



Gambar 1.1 Menggunakan API Viewer

12. Menangguhkan Eksekusi

Dalam kasus tertentu, kita perlu sekali untuk menangguhkan eksekusi kode, misalnya ketika melakukan sinkronisasi. Sayangnya, method yang memiliki fungsionalitas seperti ini tidak tersedia di Visual Basic. Pada

kenyataannya, Anda bisa memanfaatkan timer untuk menangguhkan eksekusi dan melanjutkan kembali dalam interval waktu tertentu. Namun, jika Anda menginginkan pendekatan yang lebih praktis, gunakan API Win32.

```
' Untuk menangguhkan eksekusi
Private Declare Sub Sleep Lib "kernel32" ( _
ByVal dwMilliseconds As Long)

Private Sub Pause(ByVal sngDetik As Single)
' Menangguhkan eksekusi dalam (waktu) detik
Call Sleep(Int(sngDetik * 1000#))
End Sub
```

Sebenarnya Anda bisa langsung menggunakan method Sleep, tetapi perlu diperhatikan, nilai argumennya adalah waktu dalam satuan milidetik. Oleh karena itu, untuk lebih memudahkan penggunaan method Sleep, kita membuat method Pause yang menerima argumen waktu dalam satuan detik. Contoh penggunaan method Pause diperlihatkan sebagai berikut:

```
Dim i As Integer
For i = 1 To 200
Me.Caption = i
If i = 100 Then
' Tangguhkan eksekusi selama 5 detik
Me.Caption = "Berhenti 5 detik"
Pause (5)
End If
Next i
```

13. Mengatur Project Visual Basic

Setiap kali kita membuat project, IDE akan menciptakan file .vbp yang berisi informasi-informasi penting. Adapun jika diperlukan, Anda juga bisa melakukan pengaturan spesifik terhadap project, terlepas dari

pengaturan default.

- **Mengatur Versi Aplikasi**

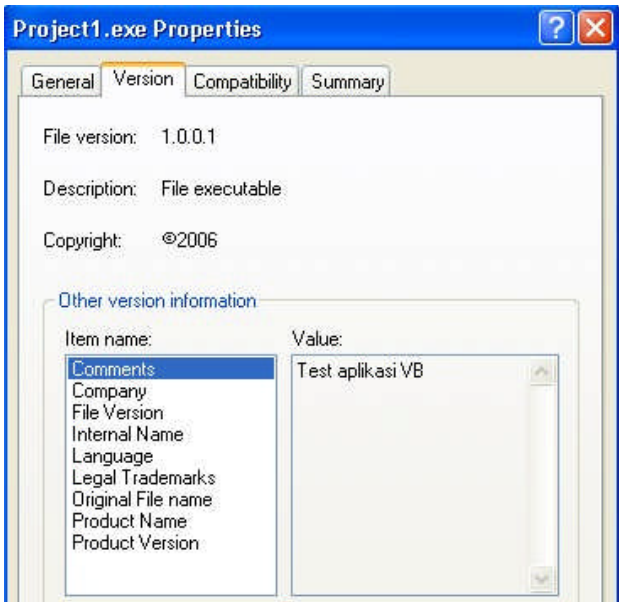
Apabila Anda ingin mengubah ikon default yang ada di form, gunakan properti Icon. Untuk menghasilkan tampilan yang proporsional, sebaiknya ukuran ikon kurang lebih 14 x 14 pixel (*picture element*).

- **Mengatur Versi Aplikasi**

Untuk menandai aplikasi, sebaiknya Anda memberikan versi yang berbeda di setiap versi rilis. Adapun cara yang praktis adalah dengan memanfaatkan fitur IDE. Terlebih dahulu buka kotak dialog **Project Properties** dari menu **Project**. Setelah itu, aktifkan tab **Make**, kemudian beri tanda centang di bagian **Auto Increment**.

- **Informasi Versi**

Sebelum Anda mendistribusikan aplikasi, akan lebih baik jika Anda memberikan informasi detail mengenai aplikasi. Langkah ini bisa Anda lakukan dari tab **Make** di kotak dialog **Project Properties**. Pada bagian **Version Information**, pilih **Type** informasi, kemudian isikan informasinya di bagian **Value**.



Gambar 1.2 Informasi aplikasi