BAB VII PUSTAKA VISUAL BASIC

BEKERJA DENGAN BILANGAN

Visual Basic menawarkan operator dan fungsi matematika yang kaya dan bermacam-macam. Hampir semua dari operator-operator ini adalah *polymorphic* dalam artian bahwa mereka bekerja secara bersamaan dengan argumen dari berbagai tipe, meliputi Integer, Long, Single, Double, Date, dan Currency. Tergantung dari operator atau fungsi tertentu, compiler Visual Basic dapat memutuskan untuk mengkonversi operand-operand ke sebuah tipe data yang lebih sesuai.

OPERATOR-OPERATOR MATEMATIKA

Secara umum, Visual Basic mendukung keempat operator. Ketika menggabungkan 2 nilai dari tipe yang berbeda, Visual Basic secara otomatis akan melakukan "pemaksaan" data dan mengkonversi tipe yang paling sederhana ke yang lebih umum/luas (contoh, Integer ke Long, atau Single ke Double). Menariknya, operator pembagian (/) selalu mengkonversi kedua operand ke Double, dimana dapat menyebabkan efek tambahan yang tidak diinginkan. Jika anda membagi sebuah bilangan Integer dengan Long oleh bilangan Integer atau Long yang lain dan anda tidak tertarik pada bagian desimal dari hasil pembagian, sebaiknya anda menggunakan operator pembagian Integer (\), dimana bekerja lebih cepat :

```
Dim a As Long, b As Long, result As Long
result = a / b      ' Pembagian pada tingkat Float (real)
result = a \ b      ' Ini akan bekerja lebih cepat sekitar 4 kali.
```

Visual Basic juga mendukung operator perpangkatan (^), dimana menaikkan sebuah bilangan ke sebuah pangkat. Dalam kasus ini, hasil selalu bertipe Double, bahkan jika anda menaikkan sebuah bilangan integer ke sebuah pangkat integer. Secara umum, operator ^ adalah relatif lambat, dan untuk pangkat integer yang kecil anda sebaiknya menggunakan rangkaian operasi perkalian :

```
Dim x As Double, result As Double
x = 1.2345
result = x ^ 3
result = x * x * x   ' Ini bekerja lebih cepat sekitar 20 kali.
```

Operator **MOD** mengambil sisa dari suatu pembagian di antara nilai-nilai integer. Dia sering digunakan untuk menguji apakah sebuah bilangan adalah perkalian yang tepat dari bilangan lain. Operator ini sangat efisien tetapi mempunyai sebuah batasan: Dia mengkonversi operand-nya ke Long dan oleh karena itu tidak dapat digunakan dengan nilai-nilai besar yang terpisah. Dia juga memotong bagian desimalnya. Berikut contoh fungsi yang bekerja dengan nilai Double apapun :

```
Function FPMOD(ByVal Number As Double, ByVal divisor As Double) As Double
    ' Catatan: ini berbeda dari MOD saat bilangan adalah negatif.
    FPMOD = Number - Int(Number / divisor) * divisor
End Function
```

Beberapa fungsi lain yang sering berguna ketika anda bekerja dengan bilangan :

- **Abs** mengembalikan nilai absolute dari argumennya.
- **Sgn** mengembalikan -1, 0, atau +1 jika argumen adalah negatif, nol, atau positif, secara berurutan.
- **Sqr** mengembalikan akar kuadrat dari sebuah bilangan.
- **Exp** membangkitkan e (basis dari logaritma natural) ke pangkat yang dilewatkan dalam argumen.
- **Log** mengembalikan logaritma natural dari argumennya. Anda dapat mengevaluasi sebuah logaritma desimal dengan menggunakan fungsi berikut :

```
Function Log10(Number As Double) As Double
    Log10 = Log(Number) / 2.30258509299405
End Function
```

OPERATOR PERBANDINGAN

Visual Basic mendukung 6 operator perbandingan, yang dapat diterapkan pada operand numerik dan string :

```
= < <= > >= <>
```

Operator-operator ini digunakan dalam blok IF, tetapi anda seharusnya juga memperhatikan bahwa mereka secara konsep tidak berbeda dari operator matematika yang lainnya, dalam hal mereka menerima 2 operand dan mengirimkan sebuah hasil. Hasil bisa berupa False (0) atau True (-1).

```
' Baris-baris berikut ini adalah sama.
If x > y Then x = x - 1
x = x + (x > y)
```

OPERATOR BOOLEAN DAN BIT-WISE

Visual Basic mendukung beberapa operator Boolean, dimana secara khusus berguna untuk mengkombinasikan perkalian sub-sub ekspresi Boolean. Operator yang paling sering digunakan adalah AND, OR, XOR, dan NOT. Berikut ini contoh dalam penggunaan operator Boolean untuk menentukan tanda dari dua variabel :

```
If (x > 0) And (y > 0) Then
    ' X dan Y adalah positif.
ElseIf (x = 0) Or (y = 0) Then
    ' Salah satu dari X atau Y adalah nol.
ElseIf (x > 0) Xor (y > 0) Then
    ' Salah satu dari X atau Y (bukan kedua-duanya) adalah positif.
ElseIf Not (x > 0) Then
    ' X adalah bukan positif.
End If
```

ROUND (PEMBULATAN) DAN TRUNCATE (PEMOTONGAN)

Fungsi **Int** memotong sebuah bilangan ke nilai integer yang sama atau lebih rendah dari argumennya. Hal ini berbeda dari sekedar mengatakan "memotong bagian desimal dari sebuah bilangan". Perbedaan akan nampak ketika argumen adalah negatif.

```
Print Int(1.2)      ' Menampilkan "1"
Print Int(-1.2)    ' Menampilkan "-2"
```

Fungsi yang sebenarnya memotong bagian desimal dari sebuah bilangan adalah **Fix** :

```
Print Fix(1.2)      ' Menampilkan "1"
Print Fix(-1.2)    ' Menampilkan "-1"
```

Visual Basic 6 memperkenalkan sebuah fungsi matematika baru, **Round**, yang membuat anda membulatkan sebuah bilangan desimal ke bilangan dari digit yang anda inginkan (atau ke integer terdekat, jika argumen kedua diabaikan) :

```
Print Round(1.45)   ' Menampilkan "1"
Print Round(1.55)   ' Menampilkan "2"
Print Round(1.23456, 4) ' Menampilkan "1.2346"
```

Round mempunyai suatu kebiasaan atau kelakuan yang tak terdokumentasi: Ketika bagian pecahan tepat 0.5, dia dibulatkan ke atas jika bagian integer adalah sebuah bilangan ganjil dan dibulatkan ke bawah jika genap :

```
Print Round(1.5), Round(2.5) ' Keduanya menampilkan "2".
```

MENINGKONVERSI DI ANTARA BASIS NUMERIK YANG BERBEDA

VBA mendukung konstan numerik dalam sistem desimal, heksadesimal, dan oktal :

```
value = &H1234      ' Menghasilkan nilai 4660 dari konstan heksadesimal
value = &O11064     ' Nilai yang sama dihasilkan dari konstan oktal
```

Anda dapat mengkonversi suatu string heksa atau oktal ke desimal menggunakan fungsi **Val** :

```
' Jika terdapat kontrol Text1 yang menyimpan nilai heksadesimal
value = Val("&H" & Text1.Text)
```

Anda dapat melakukan konversi kebalikannya, yaitu dari desimal ke heksa atau oktal dengan menggunakan fungsi **Hex** dan **Oct** :

```
Text1.Text = Hex$(value)
```

Dalam *source code* terdapat contoh dalam pengkonversian sistem biner yang tidak disediakan oleh Visual Basic.

PEMILIHAN FORMAT UNTUK BILANGAN

Semua versi dari bahasa VBA mencakup fungsi **Format**, yang merupakan perangkat yang handal yang dapat anda gunakan untuk keperluan pemformatan.

```
result = Format(Expression, [Format], _
    [FirstDayOfWeek As VbDayOfWeek = vbSunday], _
    [FirstWeekOfYear As VbFirstWeekOfYear = vbFirstJan1])
```

Untungnya, dua argumen pertama mencukupi untuk semua tugas anda kecuali dalam pemformatan tanggal. Saat memformat bilangan, fungsi **Format** mendukung pemformatan penamaan dan model (jenis). Pemformatan penamaan meliputi string-string berikut: Bilangan umum (tidak ada pemformatan khusus, gunakan notasi *scientific* jika diperlukan), *Currency* (simbol currency, pemisah ribuan dan dua digit desimal), *Fixed* (dua digit desimal), Standar (pemisah ribuan dan dua digit desimal), Persen (sebuah persentase, dengan ditambahkan simbol %), *Scientific* (notasi scientific), Yes/No, True/False, On/Off (False atau Off jika 0, dan True atau On sebaliknya). **Format** adalah sebuah fungsi lokal dan secara otomatis menggunakan simbol *currency*, pemisah ribuan, dan pemisah desimal yang sesuai.

Jika pemformatan penamaan tidak bekerja, anda dapat membuat format yang dapat anda tentukan sendiri dengan menggunakan string format yang terdiri atas karakter-karakter spesial.

```
' Pemisah desimal dan ribuan.
Print Format(1234.567, "#,##0.00") ' "1,234.57"
' Nilai persentase
Print Format(0.234, "#.##%") ' "23.4%"
' Notasi Scientific
Print Format(12345.67, "#.###E+") ' "1.235E+4"
Print Format(12345.67, "#.###E-") ' "1.235E4"
```

Kemampuan hebat dari fungsi **Format** adalah kemampuannya untuk menerapkan format string yang berbeda jika bilangan adalah positif, negatif, 0, atau Null. Anda dapat menggunakan *semicolon* sebagai pembatas dari seksi dalam format string yang anda buat. (Anda dapat menentukan satu, dua, tiga, atau empat seksi yang berbeda).

```
' Dua digit desimal untuk bilangan positif, menyertakan bilangan negatif dalam
sepasang tanda kurung buka tutup, kosong untuk nol, dan "N/A" untuk nilai Null.
Print Format(number, "##,###.00;(##,###.00); ;N/A")
```

Visual Basic 6 telah mengenalkan 3 fungsi pemformatan baru untuk bilangan, yaitu **FormatNumber**, **FormatPercent**, dan **FormatCurrency**.

```
hasil = FormatNumber(expr, [DecDigits], [InclLeadingDigit], _
    [UseParens], [GroupDigits] )
hasil = FormatPercent(expr, [DecDigits], [InclLeadingDigit], _
    [UseParens], [GroupDigits] )
hasil = FormatCurrency(expr, [DecDigits], [InclLeadingDigit], _
    [UseParens], [,GroupDigits] )
```

Dalam semua kasus, **DecDigits** adalah jumlah digit desimal yang anda inginkan (defaultnya 2); **InclLeadingDigit** menunjukkan bahwa bilangan yang berada pada jarak [-1,1] ditampilkan dengan diawali 0; **UseParens** menentukan apakah sebuah bilangan negatif berada pada tanda kurung; **GroupDigits** menunjukkan apakah pemisah ribuan seharusnya digunakan. Tiga argumen opsional lain dapat berupa salah satu dari 3 nilai berikut : 0-vbFalse, -1-vbTrue, or -2-vbUseDefault. Jika anda mengabaikan nilainya, *vbUseDefault* dianggap sebagai default.

BILANGAN RANDOM (ACAK)

Visual Basic menawarkan hanya satu pernyataan dan satu fungsi untuk membangkitkan nilai acak. Anda dapat menginisialisasi bibit dari pembangkit bilangan acak internal dengan menggunakan pernyataan **Randomize**. Anda dapat melewati sebuah bilangan yang akan digunakan sebagai bibit; sebaliknya, Visual Basic secara otomatis menggunakan nilai yang dikembalikan oleh fungsi Timer.

```
Randomize 10
```

Fungsi **Rnd** mengembalikan sebuah nilai acak setiap saat anda memanggilnya. Nilai yang dikembalikan selalu lebih kecil dari 1 dan lebih besar dari atau sama dengan 0, jadi anda perlu untuk mengatur skala dari hasil untuk mendapatkan sebuah bilangan dalam jarak yang anda inginkan :

```
' Contoh pengacakan dadu (menampilkan bilangan 1-6 secara acak)
Randomize
For i = 1 To 10
    Print Int(Rnd * 6) + 1
Next
```

BEKERJA DENGAN STRING

Visual Basic untuk aplikasi meliputi banyak fungsi string yang handal

FUNGSI DAN OPERATOR STRING DASAR

Operator string dasar **&** melakukan konkatenasi (penggabungan) string. Hasilnya adalah sebuah string yang terdiri dari semua karakter pada string pertama diikuti string kedua.

```
Print "ABCDE" & "1234"    ' Menampilkan "ABCDE1234"
```

Meski anda bisa menggunakan operator **+** untuk melakukan penggabungan string, sebaiknya dihindari. Hal itu akan berdampak pada kemudahan pembacaan kode dan dapat menyebabkan kelakuan yang tidak diharapkan ketika kedua operand adalah bukan string.

Berikutnya adalah beberapa fungsi string yang sering dipakai, meliputi **Left\$, Right\$,** dan **Mid\$,** yang mengekstrak sebuah substring dari awal, akhir, atau tengah dari string sumber.

```
Teks = "123456789"
Print Left$(Teks, 3)    ' Menampilkan "123"
Print Right$(Teks, 2)   ' Menampilkan "89"
Print Mid$(Teks, 3, 4)  ' Menampilkan "3456"
```

Mid\$ dapat juga bekerja sebagai suatu perintah yang mengijinkan anda untuk memodifikasi satu atau lebih karakter dalam sebuah string.

```
Teks = "123456789"
Mid$(Teks, 3, 4) = "abcd" ' Isi baru Teks = "12abcd789"
```

Fungsi **Len** mengembalikan panjang dari string terpilih. Seringkali digunakan untuk mengetes apakah sebuah string berisi suatu karakter :

```
Print Len("12345")      ' Menampilkan "5"
If Len(Teks) = 0 Then ... ' Lebih cepat daripada membandingkan dgn string kosong.
```

Untuk menghilangkan karakter kosong (misal: spasi) di akhir atau awal, anda dapat menggunakan fungsi **LTrim\$**, **RTrim\$**, dan **Trim\$** :

```
Teks = " abcde "
Print LTrim$(Teks)      ' Menampilkan "abcde "
Print RTrim$(Teks)     ' Menampilkan " abcde"
Print Trim$(Teks)      ' Menampilkan "abcde"
```

Fungsi **Asc** mengembalikan kode ASCII dari huruf pertama dalam string.

```
If Asc(Text) = 32 Then      ' Menguji apakah karakter pertama adalah spasi.
If Left$(Text, 1) = " " Then ' Efeknya sama, tapi 2 - 3 kali lebih lambat
```

Ketika anda menggunakan fungsi **Asc**, anda harus memastikan bahwa string tidak dalam keadaan kosong sebab akan menyebabkan kesalahan.

Berikutnya fungsi **Chr\$** adalah kebalikan dari **Asc**, yaitu mengubah kode numerik ke dalam karakter yang bersesuaian.

```
Print Chr$(65)           ' Menampilkan "A"
```

Fungsi **Space\$** dan **String\$** adalah sama. Fungsi pertama mengembalikan sebuah string berisi spasi dengan panjang yang anda inginkan, dan yang kedua mengembalikan string berisi karakter yang dispesifikasikan dalam parameter kedua diulang sebanyak jumlah yang telah anda tuliskan di parameter pertama.

```
Print Space$(5)          ' Menampilkan "     " (lima spasi)
Print String$(5, " ")    ' Efek yang sama
Print String$(5, 32)     ' Efek yang sama, dengan menggunakan kode karakter
Print String$(50, ".")   ' Satu baris string berisi 50 titik.
```

Berikutnya adalah fungsi **StrComp** mengijinkan anda membandingkan string secara *case-insensitive* dan mengembalikan -1, 0, atau 1 jika argumen pertama lebih kecil, sama dengan, atau lebih besar dari argumen kedua. Argumen ketiga menspesifikasikan apakah perbandingan dilakukan dengan jalan *case-insensitive*.

```
Select Case StrComp(satu, dua, vbTextCompare)
Case 0
    ' satu = dua (contoh: "VISUAL BASIC" vs. "Visual Basic")
Case -1
    ' satu < dua (contoh: "C++" vs. "Visual Basic")
Case 1
    ' satu > dua (contoh: "Visual Basic" vs. "Delphi")
End Select
```

FUNGSI KONVERSI

Fungsi yang paling banyak dipakai untuk mengkonversi string adalah **UCase\$** dan **LCase\$**, dimana mengubah argumennya ke *uppercase* (huruf besar) and *lowercase* (huruf kecil), secara berurutan :

```
Teks = "Teknik Informatika, UNIKOM"
Print UCase$(Teks)      ' "TEKNIK INFORMATIKA, UNIKOM"
Print LCase$(Teks)     ' "teknik informatika, unikom"
```

Fungsi **StrConv** meliputi kedua fungsi yang telah disebutkan di atas serta terdapat fungsi tambahan lain. Anda dapat menggunakannya untuk mengkonversi ke *uppercase*, *lowercase*, and *propercase* (yaitu setiap karakter pertama dari sebuah kata diubah ke huruf besar, dan yang lainnya ke huruf kecil).

```
Print StrConv(Teks, vbUpperCase) ' " TEKNIK INFORMATIKA, UNIKOM "
Print StrConv(Teks, vbLowerCase) ' " teknik informatika, unikom "
Print StrConv(Teks, vbProperCase) ' " Teknik Informatika, Unikom "
```

Fungsi **Val** mengkonversi suatu string ke dalam representasi desimalnya. Fungsi **Val** juga dapat mengenali titik desimal dan akan berhenti melakukan pengkonversian apabila menemukan karakter yang tidak valid (termasuk simbol currency atau tanda koma yang digunakan untuk mengelompokkan digit ribuan dalam dolar).

Fungsi **Str\$** mengkonversi sebuah bilangan ke dalam representasi string-nya. Perbedaan utama antara **Str\$** dan **CStr** adalah fungsi pertama akan menambahkan sebuah spasi apabila argumennya positif, sedangkan fungsi kedua tidak.

MENEMUKAN DAN MENGGANTI SUBSTRING

Fungsi **InStr** digunakan untuk mencari sebuah substring di dalam string lain, baik dalam mode *case-sensitive* atau *case-insensitive*. Anda tidak dapat mengabaikan indeks awal jika anda ingin melewati argumen yang menspesifikasikan jenis pencarian yang ingin anda lakukan.

```
Print InStr("abcde ABCDE", "ABC") ' Menampilkan "7" (case sensitive)
Print InStr(8, "abcde ABCDE", "ABC") ' Menampilkan "0" (indeks awal > 1)
Print InStr(1, "abcde ABCDE", "ABC", vbTextCompare) ' Menampilkan "1" (case insensitive)
```

Visual Basic 6 mengijinkan anda untuk melakukan pencarian balik, yaitu dengan menggunakan fungsi baru **InStrRev**. Sintaknya sama dengan fungsi asli **InStr**, tetapi urutan dari argumennya yang berbeda.

```
ketemu = InStrRev(StringSumber, StringCari, [Mulai], [MetodePembanding])
```

Berikut ini adalah sedikit contoh. Perhatikan bahwa jika anda mengabaikan argumen Mulai, awal pencarian adalah pada akhir string :

```
Print InStrRev("abcde ABCDE", "abc") ' Menampilkan "1" (case sensitive)
Print InStrRev("abcde ABCDE", "abc", ,vbTextCompare) ' Menampilkan "7" (case insensitive)
Print InStrRev("abcde ABCDE", "ABC", 4, vbTextCompare)
' Menampilkan "1" (case insensitive, awal<>0)
```

Visual Basic juga mencakup operator string yang bagus, operator **Like**, dimana lebih hemat waktu ketika anda mem-*parsing* sebuah string dan melakukan pencarian yang kompleks. Sintaknya adalah sebagai berikut :

```
hasil = string Like corak
```

Dimana string adalah string yang akan di-*parsing* dan corak adalah string yang tersusun atas karakter spesial yang menentukan kondisi pencarian. Karakter yang paling sering dipakai adalah ? (atau karakter tunggal apapun), * (nol atau lebih karakter), dan # (digit tunggal apapun). Contohnya adalah sebagai berikut :

```
' Operator Like dipengaruhi oleh pilihan pengaturan Compare (pembanding).
Option Compare Text          ' Memaksakan perbandingan secara case-insensitive
' Memeriksa bahwa string terdiri atas "AB" diikuti 3 digit karakter.
If value Like "AB###" Then ... ' contoh: "AB123" atau "ab987"
' Memeriksa bahwa string diawali dengan "ABC" dan diakhiri dengan "XYZ".
If value Like "ABC*XYZ" Then ... ' contoh: "ABCDEFGHI-VWXYZ"
' Memeriksa bahwa string diawali dengan "1", diakhiri dengan "X", dan terdiri atas
' 5 karakter.
If value Like "1???X" Then ... ' contoh: "1234X" atau "1uvwX"
```

Anda dapat juga menspesifikasikan karakter mana yang ingin anda masukkan (atau keculikan) dalam pencarian dengan menyisipkan sebuah daftar dalam tanda kurung kotak "[]".

```
' Salah satu huruf "A","B","C" diikuti oleh tiga digit karakter.
If value Like "[A-C]###" Then ... ' contoh: "A123" atau "c456"
' Tiga karakter, karakter pertama harus termasuk huruf vokal.
If value Like "[AEIOU][A-Z][A-Z]" Then... ' contoh: "IVB" atau "OOP"
' Minimal 3 karakter, karakter pertama tidak boleh berupa bilangan numerik.
' Catatan: awalan simbol "!" mengecualikan karakter-karakter dalam daftar.
If value Like "[!0-9]??* " Then ... ' contoh: "K12BC" atau "ABHIL"
```

Visual Basic 6 memperkenalkan fungsi baru **Replace**, dimana dapat dengan cepat menemukan dan mencari substring. Sintaknya adalah sebagai berikut :

```
Teks = Replace(Sumber, StringCari, StringPengganti, [Mulai], [Cacah], [MetodePembanding])
```

Bentuk sederhana pencarian substring dalam mode *case-sensitive* dan penggantian adalah sebagai berikut :

```
Print Replace("abc ABC abc", "ab", "123") ' Hasil "123c ABC 123c"
```

Dengan bermain pada argumen lain, anda dapat memulai pencarian dari posisi berbeda, membatasi jumlah penggantian, dan melakukan pencarian *case-insensitive*. Perhatikan bahwa sebuah nilai untuk awal (mulai) yang lebih besar dari 1 sebenarnya akan memangkas argumen sumbernya sebelum memulai pencarian.

```
Print Replace("abc ABC abc", "ab", "123", 5, 1) ' Hasil "ABC 123c"
Print Replace("abc ABC abc", "ab", "123", 5, 1, vbTextCompare) ' Hasil "123C abc"
```

Anda dapat juga menggunakan fungsi **Replace** untuk menghitung suatu substring dalam string lain.

```
Function InstrCount(Source As String, Search As String) As Long
' Anda mendapatkan jumlah substring dengan mengurangi panjang string
' aslinya dari string yang anda dapatkan dengan menggantikan substring
' dengan string lain yang lebih panjang satu karakter.
InstrCount = Len(Replace(Sumber, StringCari, StringCari & "*")) - Len(Sumber)
End Function
```

Fungsi baru **StrReverse** secara cepat membalikkan urutan karakter-karakter dalam string. Fungsi ini agak kurang berguna sebenarnya, namun dia menambah fungsi pemrosesan string lainnya.

```
' Mengganti hanya substring yang pertama kali ditemukan dari arah belakang.
Function ReplaceLast(Source As String, Search As String, _
    ReplaceStr As String) As String
    ReplaceLast = StrReverse(Replace(StrReverse(Source), _
    StrReverse(Search), StrReverse(ReplaceStr), , 1))
End Function
```

Anda dapat juga menggunakan fungsi **Split** untuk menemukan semua item yang dibatasi oleh pembatas dalam sebuah string. Sintaknya adalah sebagai berikut :

```
arr() = Split(Sumber, [Pembatas], [JumlahBatasan], [MetodePembanding])
```

Dimana Pembatas adalah karakter yang digunakan untuk membatasi item-item secara individual. Anda dapat melewati sebuah nilai positif untuk argumen JumlahBatasan jika tidak ingin jumlah item lebih dari nilai yang diberikan, dan anda dapat melewati nilai *vbTextCompare* pada argumen terakhir untuk melakukan pencarian dalam mode case-insensitive. Karena karakter pembatas defaultnya adalah spasi, anda dapat dengan mudah mengekstrak semua kata dalam sebuah kalimat dengan menggunakan kode ini :

```
Dim kata() As String
kata() = Split("Teknik Informatika Kampus UNIKOM")
' kata() sekarang adalah sebuah array berbasis nol dengan 4 elemen.
```

Fungsi **Join** adalah pelengkap fungsi **Split** dalam hal dia menerima sebuah *array* string dan satu karakter pembatas dan membangun kembali string aslinya.

```
' Melanjutkan contoh sebelumnya ...
' Argumen pembatas di sini adalah opsional, karena defaultnya adalah " ".
Print Join(kata, " ") ' Menampilkan " Teknik Informatika Kampus UNIKOM "
```

Perhatikan bahwa argumen pembatas dalam fungsi **Split** dan **Join** dapat berisi lebih dari satu karakter.

Fungsi tambahan lainnya dalam bahasa VBA adalah fungsi **Filter**, dimana dengan cepat membaca pencarian *array* untuk sebuah substring dan mengembalikan *array* lainnya yang berisi hanya item-item yang termasuk (atau selain) substring yang dicari. Contoh penggunaan sintak untuk fungsi **Filter** adalah sebagai berikut :

```
arr() = Filter(Sumber(), Cari, [Include], [MetodePembanding])
```


MEMBANGUN DAN MENGEKSTRAK NILAI TANGGAL DAN WAKTU

Terdapat banyak jalan untuk memperoleh nilai tanggal. Misal, anda dapat menggunakan konstan tanggal, contoh :

```
TanggalMulai = #4/10/2005 9:20:57 AM#
```

Fungsi **DateSerial** membangun sebuah nilai tanggal dari komponen tahun/bulan/tanggal nya; begitu juga, fungsi **TimeSerial** membangun sebuah nilai waktu dari komponen jam/menit/detik nya.

```
Print DateSerial(2006, 4, 14)      ' Menampilkan "4/14/2006"
Print TimeSerial(12, 20, 30)      ' Menampilkan "12:20:30 P.M."
' Perhatikan bahwa mereka tak menimbulkan error dengan argumen yang tidak valid
Print DateSerial(2006, 2, 31)      ' Menampilkan "3/3/2006"
```

Fungsi **DateSerial** juga berguna untuk menentukan secara tidak langsung apakah tahun tertentu merupakan tahun kabisat.

```
Function ApaTahunKabisat(tahun As Integer) As Boolean
    ' Apakah 29 Februari dan 1 Maret berbeda tanggal ?
    ApaTahunKabisat = DateSerial(tahun, 2, 29) <> DateSerial(tahun, 3, 1)
End Function
```

Fungsi **DateValue** dan **TimeValue** mengembalikan bagian tanggal dan waktu dari argumennya, dimana bisa berupa ekspresi string atau *date* (tanggal).

```
' Tanggal untuk seminggu kemudian dari sekarang
Print DateValue(Now + 7)          ' Menampilkan "9/4/2006"
```

Dalam bahasa VBA terdapat fungsi yang mengijinkan anda mengekstrak informasi tanggal dan waktu dari sebuah ekspresi atau variabel tanggal.

```
' Memperoleh informasi dari tanggal sekarang.
y = Year(Now): m = Month(Now): d = Day(Now)
Print (d & "-" & m & "-" & y)      ' Menampilkan "2-4-2006"
' Fungsi berikut juga mendukung suatu format tanggal yang valid.
Print Year("4/2/2006 10:10:26 AM") ' Menampilkan "2006"
```

Fungsi **Weekday** mengembalikan sebuah bilangan di antara 1 sampai 7, dimana menunjukkan hari dari argumen *Date* (tanggal).

```
Print Weekday("4/2/06")          ' Menampilkan "1" (= vbMonday)
```

Fungsi **Weekday** mengembalikan 1 ketika tanggal merupakan hari pertama dalam seminggu. Namun hal itu bisa juga berbeda untuk sistem Windows yang lain. Jika anda ingin memastikan bahwa 1 adalah Sunday (Minggu), 2 adalah Monday (Senin), dan seterusnya, anda dapat memaksakan fungsi untuk mengembalikan nilai yang konsisten terhadap semua sistem windows, adalah sebagai berikut :

```
Print Weekday(Now, vbSunday)
```

Meskipun menggunakan argumen opsional kedua memaksakan fungsi untuk mengembalikan nilai yang benar, dia tidak mengubah lokasi sistem. Akhirnya anda dapat mengekstrak informasi tanggal dan waktu dari sebuah nilai *Date* (tanggal) atau ekspresi dengan menggunakan fungsi **DatePart**, sintaknya adalah :

```
Hasil = DatePart(Interval, Tanggal, [HariPertamaPerMinggu], [MingguPertamaPerTahun])
```

Contoh aplikasi :

```
' The quarter (perempat) dimana kita berada
Print DatePart("q", Now)      ' Menampilkan "2"
' Jumlah minggu dimana kita berada (sejak 1 Januari)
Print DatePart("ww", Now)    ' Menampilkan "14"
```

Argumen pertama dapat berupa salah satu dari konstan String yang terdapat dalam Tabel 1. Untuk informasi mengenai argumen opsional kedua, lihat deskripsi dari fungsi **DateAdd** pada seksi berikutnya.

Tabel 1. Nilai-nilai untuk Argumen Interval dalam Fungsi DatePart, DateAdd, dan DateDiff.

Pengaturan	Deskripsi
"yyyy"	Year (Tahun)
"q"	Quarter (Perempat)
"m"	Month (Bulan)
"y"	Hari dalam setahun (sama sebagaimana d)
"d"	Day (tanggal)
"w"	Weekday (hari dalam minggu)
"ww"	Week (minggu dalam tahun)
"h"	Hour (jam)
"n"	Minute (menit)
"s"	Second (detik)

ARITMATIK (PENGHITUNGAN) TANGGAL

Dalam kebanyakan kasus, anda tidak perlu suatu fungsi khusus untuk melakukan penghitungan tanggal. Semua yang anda perlu ketahui adalah bagian integer dalam variabel *Date* (tanggal) yang menyimpan informasi tanggal, dan bagian kecil lain yang menyimpan informasi waktu.

```
' 2 hari dan 12 jam dari sekarang
Print Now + 2 + #12:00:00 PM# ' Menampilkan "4/5/2006 12:49:48 A.M."
```

Untuk matematika tanggal yang lebih pengalaman (maju), anda dapat menggunakan fungsi **DateAdd**, dimana sintaknya adalah sebagai berikut :

```
NewDate = DateAdd(interval, number, date)
```

Interval adalah string yang mengindikasikan sebuah unit tanggal atau waktu (lihat Tabel 1), *number* adalah jumlah unit yang anda tambahkan, dan *date* adalah tanggal mulai. Anda dapat menggunakan fungsi ini untuk menambah dan mengurangi nilai dari tanggal dan waktu.

```
' Tanggal untuk 3 bulan kemudian dari sekarang.
Print DateAdd("m", 3, Now)           ' Menampilkan "7/2/2006 12:58:22 P.M."
' Satu tahun yang lalu (Otomatis menghitung tahun kabisat)
Print DateAdd("yyyy", -1, Now)       ' Menampilkan "4/2/2005 12:58:22 P.M."
' Jumlah bulan sejak 30 Jan 2006
Print DateDiff("m", #1/30/2006#, Now) ' Menampilkan "3"
' Jumlah hari sejak 30 Jan 2006, anda dapat menggunakan "d" atau "y".
Print DateDiff("y", #1/30/2006#, Now) ' Menampilkan "62"
' Jumlah minggu sejak 1 Jan 2006
Print DateDiff("w", #1/30/2006#, Now) ' Menampilkan "8"
' Jumlah minggu sejak abad 21 (tahun 2000)
' Catatan: gunakan "ww" untuk mengembalikan jumlah Sunday (Minggu)
' dalam interval tanggal
Print DateDiff("ww", #1/1/2000#, Now) ' Menampilkan "327"
```

Ketika anda mempunyai dua tanggal dan anda ingin mengevaluasi perbedaan di antara keduanya, misal rentang waktu antara tanggal satu dengan tanggal lainnya, maka anda sebaiknya menggunakan fungsi **DateDiff**. Sintaknya adalah :

```
Result = DateDiff(interval, startdate, enddate [, FirstDayOfWeek[, FirstWeekOfYear]])
```

Dimana *interval* adalah seperti yang tertera pada Tabel 1, *FirstDayOfWeek* adalah sebuah argumen opsional yang dapat anda pakai untuk menspesifikasikan *weekday* (hari dalam minggu) mana yang anda gunakan sebagai hari pertama dalam satu minggu (anda dapat menggunakan konstan vbSunday, vbMonday, dst), dan *FirstWeekOfYear* adalah argumen opsional lainnya yang memungkinkan anda untuk menspesifikasi minggu mana yang seharusnya dianggap sebagai minggu pertama dalam setahun (lihat Tabel 2).

Tabel 2. Nilai untuk Argumen FirstWeekOfYear Dalam Fungsi DateDiff.

Konstan	Nilai	Deskripsi
vbUseSystem	0	Menggunakan aturan NLS API.
vbFirstJan1	1	Minggu pertama adalah salah satu yang meliputi 1 Januari (Ini adalah nilai default untuk aturan ini).
vbFirstFourDays	2	Minggu pertama adalah minggu pertama yang mempunyai minimal 4 hari dalam tahun baru.
vbFirstFullWeek	3	Minggu pertama adalah minggu pertama yang sama sekali terkandung dalam tahun baru.

BEKERJA DENGAN FILE

Visual Basic mencakup perintah-perintah yang cukup handal yang berhubungan dengan file teks dan binary. Obyek *FileSystemObject* membuatnya menjadi lebih mudah jika berhubungan dengan file dan direktori.

PENANGANAN FILE

Secara umum, anda tidak dapat berbuat banyak pada sebuah file tanpa anda membukanya. Visual Basic memungkinkan anda menghapus sebuah file (dengan menggunakan perintah **Kill**), memindahkan atau me-*rename* (dengan menggunakan perintah **Name ... As**), dan menyalinnya di mana saja (dengan menggunakan perintah **FileCopy**).

```
' Semua operasi file harus dilindungi dari error kesalahan.
' Tak satupun dari fungsi berikut yang bekerja pada file yang terbuka.
On Error Resume Next
' Me-rename sebuah file-anda harus menentukan path (jalur) dari file target,
' jika tidak file akan dipindahkan ke direktori terkini.
Name "c:\frunlog.txt" As "c:\frunlog.$$$"
' Memindahkan file ke direktori lain, bisa juga ke drive lain.
Name "c:\frunlog.$$$" As "d:\oke\baru.Dat"
' Menyalin file-anda dapat mengganti namanya juga ketika melakukan penyalinan
' dan anda dapat mengabaikan bagian nama file dari file target.
FileCopy "d:\oke\baru.Dat", "c:\frunlog.txt"
' Menghapus satu atau lebih file--Kill juga mendukung wildcards.
Kill "d:\oke\baru.*"
```

Anda dapat membaca dan memodifikasi atribut dari suatu file dengan menggunakan fungsi **GetAttr** dan perintah **SetAttr**, secara berurutan. Fungsi **GetAttr** mengembalikan nilai kode bit, jadi anda perlu menguji bit-bitnya secara individu menggunakan konstan intrinsik yang disediakan oleh VBA.

```
' Rutin ini juga bekerja pada file yang terbuka
' dan akan menimbulkan error jika file tidak ada.
Function GetAttrDescr>NamaFile As String) As String
  Dim hasil As String, attr As Long
  attr = GetAttr>NamaFile)
  ' GetAttr juga bekerja pada direktori.
  If attr And vbDirectory Then hasil = hasil & " Directory"
  If attr And vbReadOnly Then hasil = hasil & " ReadOnly"
  If attr And vbHidden Then hasil = hasil & " Hidden"
  If attr And vbSystem Then hasil = hasil & " System"
  If attr And vbArchive Then hasil = hasil & " Archive"
  ' Tidak mengikutsertakan tambahan spasi pada awal string.
  GetAttrDescr = Mid$( hasil, 2)
End Function
```

Begitu juga, anda dapat mengganti atribut dari file atau direktori dengan melewati perintah **SetAttr** dengan kombinasi nilai, berikut ini contohnya :

```
' Menandai file sebagai Archive dan Read-only.
NamaFile = "C:\frunlog.txt"
SetAttr NamaFile, vbArchive + vbReadOnly
' Mengubah file dari hidden (tersembunyi) ke visible (terlihat), dan sebaliknya
SetAttr NamaFile, GetAttr(NamaFile) Xor vbHidden
```

PEMROSESAN FILE TEKS

File teks adalah tipe paling sederhana untuk diproses. Anda tinggal membuka mereka dengan menggunakan pernyataan **Open** dengan klausa **For Input**, **For Output**, atau **For Appending**, dan kemudian membaca data darinya atau menuliskan data padanya. Untuk membuka file baik teks atau binary anda perlu nomer file, lihat contoh berikut :

```
' Terjadi error jika file #1 sudah terbuka
Open "readme.txt" For Input As #1
```

Dalam sebuah aplikasi tunggal, anda biasanya dapat memberikan sebuah nomer file yang unik kepada rutin yang berbeda yang berhubungan dengan file, langkah ini akan menambah kerumitan aplikasi, maka sebaiknya anda menggunakan fungsi **FreeFile** untuk meminta Visual Basic menyediakan nomer file pertama yang tersedia.

```
Dim fnum As Integer
fnum = FreeFile()
Open "readme.txt" For Input As #fnum
```

KET: Lihat modul *source code* untuk contoh pembacaan dan penulisan ke file teks.

PEMROSESAN FILE BINER

Untuk membuka file biner, gunakan pernyataan **Open** dengan pilihan **For Random** atau **For Binary**. Dalam mode biner, anda menulis ke file menggunakan pernyataan **Put** dan membaca data kembali dengan pernyataan **Get**. Visual Basic menentukan berapa byte yang dituliskan atau dibaca dengan melihat struktur dari variabel yang anda lewatkan pada argumen terakhir.

```
Dim numEls As Long, teks As String
numEls = 12345: teks = "Ada 22 karakter string"
' File biner secara otomatis dibuat bila perlu.
Open "data.bin" For Binary As #1
Put #1, , numEls      ' Put (menaruh) 4 byte.
Put #1, , teks       ' Put (menaruh) 16 byte (format ANSI).
```

Ketika membaca kembali data, anda harus mengulang urutan yang sama dari pernyataan tetapi itu terserah anda untuk memperbaiki dimensi panjang variabel string. Anda tidak perlu menutup dan membuka kembali sebuah file biner karena anda dapat menggunakan pernyataan **Seek** untuk memposisikan kembali penunjuk file ke byte tertentu.

```

Seek #1, 1           ' Kembali ke awal (byte pertama adalah byte 1)
Get #1, , numEls     ' Semua nilai Long adalah 4 byte.
teks = Space$(16)   ' Siapkan untuk membaca 16 byte.
Get #1, , teks       ' Kerjakan.

```

HIRARKI FILESYSTEMOBJECT

Visual Basic 6 mempunyai sebuah pustaka baru meliputi perintah file, dimana memungkinkan programmer untuk secara mudah membaca drive dan direktori, melakukan operasi file dasar (termasuk copy, delete, move, dsb), dan mengekstrak informasi yang tak tersedia melalui fungsi Visual Basic biasa.

OBYEK AKAR FILESYSTEMOBJECT

Pada akar dari hirarki obyek *FileSystemObject*, menyingkap banyak metode dan hanya satu properti, **Drive**, dimana mengembalikan koleksi dari semua drive dalam sistem. Obyek *FileSystemObject* (disingkat FSO) adalah obyek yang dapat dibuat hanya dalam hirarki, sehingga dia satu-satunya obyek yang dapat dideklarasikan dengan menggunakan kata kunci *New*.

```

Dim fso As New Scripting.FileSystemObject, dr As Scripting.Drive
On Error Resume Next      ' Diperlukan untuk drive yang sedang tidak siap
For Each dr In fso.Drives
    Print dr.DriveLetter & " [" & dr.TotalSize & "]"
Next

```

Tabel 3 berikut menggambarkan banyak metode yang diekspos oleh obyek FSO. Beberapa dari mereka juga tersedia (sering kali dengan nama dan sintak yang berbeda) sebagai metode dari obyek Folder dan File kedua. Kebanyakan dari metode ini menambah fungsionalitas ke perintah-perintah yang sudah terdapat dalam Visual Basic. Misal, anda dapat menghapus folder yang tidak kosong (hati-hati!) dan menyalin dan mengganti banyak file dan direktori dengan satu perintah tunggal. Anda juga dapat dengan mudah mengekstrak bagian dari nama file tanpa harus menulis rutin khusus.

Tabel 3. Semua Metode Dalam Obyek FileSystemObject

Sintak	Deskripsi
BuildPath (Path, Name)	Mengembalikan nama lengkap file, didapatkan dengan menggabungkan <i>path</i> (jalur) relatif atau absolut dan nama.
CopyFile Source, Destination, [Overwrite]	Menyalin satu atau lebih file: <i>Source</i> bisa berisi <i>wildcards</i> , dan <i>Destination</i> dianggap sebagai direktori jika diakhiri dengan <i>backslash</i> . Dia akan menimpa file yang lama kecuali anda set <i>Overwrite</i> ke <i>False</i> .
CopyFolder Source, Destination, [Overwrite]	Sama seperti <i>CopyFile</i> , tetapi menyalin folder beserta seluruh isinya (subfolder dan file). Jika <i>Destination</i> tidak terhubung ke direktori yang ada, dia akan dibuat (kecuali jika <i>Source</i> berisi <i>wildcards</i>).

CreateFolder(Path) As Folder	Membuat obyek folder baru dan mengembalikannya; membangkitkan <i>error</i> jika folder sudah ada.
CreateTextFile(FileName, [Overwrite], [Unicode]) As TextStream	Membuat obyek <i>TextFile</i> baru dan mengembalikannya; atur <i>Overwrite</i> = False untuk mencegah penimpaan pada file yang sudah ada; atur <i>Unicode</i> = True untuk membuat obyek <i>TextFile Unicode</i> .
DeleteFile FileSpec, [Force]	Menghapus satu atau lebih file. <i>FileSpec</i> dapat menggunakan <i>wildcards</i> ; atur <i>Force</i> = True untuk memaksa penghapusan file read-only.
DeleteFolder(FolderSpec, [Force])	Menghapus satu atau lebih folder, bersama dengan isinya; atur <i>Force</i> = True untuk memaksa penghapusan file read-only.
DriveExists(DriveName)	Mengembalikan True jika drive <i>logic</i> yang diberikan ada.
FileExists(FileName)	Mengembalikan True jika file yang diberikan ada. (Jalur bisa relatif terhadap direktori terkini).
FolderExists(FolderName)	Mengembalikan True jika folder yang diberikan ada. (Jalur bisa relatif terhadap direktori terkini).
GetAbsolutePathName(Path)	Mengkonversi jalur relatif pada direktori terkini ke jalur absolut.
GetBaseName(Filename)	Mengekstrak nama file dasar (tanpa jalur dan ekstensinya); dia tidak memeriksa apakah file dan/atau jalurnya memang ada.
GetDrive(DriveName) As Drive	Mengembalikan obyek <i>Drive</i> yang berhubungan dengan abjad atau jalur UNC yang dilewatkan sebagai argument. (Dia memeriksa apakah drive memang ada).
GetDriveName(Path)	Mengekstrak <i>drive</i> dari <i>path</i> (jalur).
GetExtensionName(FileName)	Mengekstrak string ekstensi dari nama file.
GetFile(FileName)	Mengembalikan obyek File yang berhubungan dengan nama yang dilewatkan sebagai argumen. (Dapat berupa absolut atau relatif terhadap direktori terkini).
GetFileName(	Mengekstrak nama file (tanpa jalurnya namun dengan ekstensinya); dia tidak memeriksa apakah file dan/atau jalurnya memang ada).
GetFolder(FolderName) As Folder	Mengembalikan obyek Folder yang berhubungan dengan jalur yang dilewatkan sebagai argumen. (Dapat berupa absolut atau relatif terhadap direktori terkini).
GetParentFolderName(Path)	Mengembalikan nama dari direktori induk dari

	direktori yang dilewatkan sebagai argumen (atau sebuah string kosong jika direktori induk tidak ada).
GetSpecialFolder(SpecialFolder) As Folder	Mengembalikan obyek Folder yang berhubungan dengan direktori spesial Windows. <i>SpecialFolder</i> dapat berupa 0-WindowsFolder, 1-SystemFolder, 2-TemporaryFolder.
GetTempName()	Mengembalikan nama file yang tidak ada (tidak sedang dipakai) yang dapat digunakan sebagai file temporer.
MoveFile(Source, Destination)	Sama seperti <i>CopyFile</i> , tetapi dia menghapus file sumber. Dia dapat pula memindahkan antar drive yang berbeda, jika fungsi ini didukung oleh sistem operasi.
MoveFolder(Source, Destination)	Sama seperti <i>MoveFile</i> , tetapi bekerja pada direktori.
OpenTextFile(FileName, [IOMode], [Create], [Format])As TextStream	Membuka file teks dan mengembalikan obyek <i>TextStream</i> yang bersangkutan. <i>IOMode</i> dapat merupakan satu atau kombinasi (gunakan operator OR) dari konstan berikut : 1-ForReading, 2-ForWriting, 8-ForAppending; atur Create ke True jika anda ingin membuat file baru; Format dapat berupa 0-TristateFalse (ANSI), -1-TristateTrue (Unicode) atau -2-TristateUseDefault (ditentukan oleh sistem).

OBJEK DRIVE

Objek ini hanya mengekspos properti (bukan metode), semuanya telah dilampirkan dalam Tabel 4. Semua properti adalah *read-only*, kecuali properti **VolumeName**. Kode singkat berikut menentukan drive lokal yang siap dan mempunyai ruang bebas minimal 100 MB.

```
Dim fso As New Scripting.FileSystemObject, dr As Scripting.Drive
For Each dr In fso.Drives
  If dr.IsReady Then
    If dr.DriveType = Fixed Or dr.DriveType = Removable Then
      ' 2 ^ 20 sama dengan satu megabyte.
      If dr.FreeSpace > 100 * 2 ^ 20 Then
        Print dr.Path & " [" & dr.VolumeName & "] = " _
          & dr.FreeSpace
      End If
    End If
  End If
End If
Next
```

Tabel 4. Semua Properti Obyek Drive

Sintak	Deskripsi
AvailableSpace	Ruang yang tersedia di <i>drive</i> , dalam byte; dia biasanya serupa dengan nilai yang dikembalikan properti <i>FreeSpace</i> , kecuali sistem operasi mendukung kuota disk.
DriveLetter	<i>Letter</i> (huruf abjad) yang berhubungan dengan drive atau sebuah string kosong untuk drive jaringan yang tidak berhubungan dengan suatu huruf abjad.
DriveType	Sebuah konstan yang mengindikasikan tipe <i>drive</i> : 0-Unknown, 1-Removable, 2-Fixed, 3-Remote, 4-CDRom, 5-RamDisk.
FileSystem	Sebuah string yang menggambarkan sistem file yang sedang digunakan: FAT, NTFS, CDFS.
FreeSpace	Ruang bebas di <i>drive</i> . (lihat AvailableSpace.)
IsReady	True jika <i>drive</i> dalam keadaan siap, False jika sebaliknya.
Path	Jalur yang berhubungan dengan <i>drive</i> , tanpa <i>backslash</i> (misal, C:).
RootFolder	Obyek Folder yang berhubungan dengan akar direktori.
SerialNumber	Sebuah bilangan Long yang berhubungan dengan nomor serial disk.
ShareName	Nama drive yang di- <i>share</i> (berbagi) dalam jaringan atau sebuah string kosong jika bukan merupakan drive jaringan.
TotalSize	Kapasitas total dari <i>drive</i> , byte.
VolumeName	Label disk (dapat dibaca dan ditulis).

OBJEK FOLDER

Obyek Folder merepresentasikan sebuah subdirektori secara individual. Anda dapat mendapatkan referensi ke obyek tersebut dengan cara yang berbeda-beda: dengan menggunakan metode **GetFolder** atau **GetSpecialFolder** dari obyek *FileSystemObject*, melalui properti **RootFolder** dari sebuah obyek Drive, melalui properti **ParentFolder** dari sebuah obyek File atau obyek File yang lainnya, atau dengan mengiterasi melalui koleksi **SubFolders** dari obyek Folder yang lainnya. Obyek Folder mengekspose sejumlah properti yang menarik (lihat Tabel 5), namun hanya properti **Attribute** dan **Name** yang dapat ditulis. Properti yang paling menarik kemungkinan adalah koleksi **SubFolders** dan **Files**, dimana memungkinkan anda mengiterasi melalui subdirektori dan file dengan menggunakan sintak yang luwes dan ringkas sebagai berikut :

```
' Mencetak nama-nama direktori level satu pada semua drive dan nama pendeknya 8.3.
Dim fso As New Scripting.FileSystemObject
Dim dr As Scripting.Drive, fld As Scripting.Folder
On Error Resume Next
For Each dr In fso.Drives
  If dr.IsReady Then
    Print dr.RootFolder.Path      ' Folder akar.
    For Each fld In dr.RootFolder.SubFolders
      Print fld.Path & " [" & fld.ShortName & "]"
    Next
  End If
Next
```

Tabel 5. Semua Properti dari Obyek Folder dan File.

Sintak	Deskripsi	Diterapkan Ke
Attributes	Atribut dari file atau folder, dengan kombinasi dari konstan berikut: 0- Normal, 1-ReadOnly, 2-Hidden, 4-System, 8-Volume, 16-Directory, 32-Archive, 64-Alias, 2048-Compressed. Atribut Volume, Directory, Alias, dan Compressed tidak dapat dimodifikasi.	Folder dan File
DateCreated	Tanggal pembuatan (nilai <i>Date</i> yang <i>read-only</i>).	Folder dan File
DateLastAccessed	Tanggal terakhir diakses (nilai <i>Date</i> yang <i>read-only</i>).	Folder dan File
DateLastModified	Tanggal terakhir dimodifikasi (nilai <i>Date</i> yang <i>read-only</i>).	Folder dan File
Drive	Obyek <i>Drive</i> dimana file atau folder terletak.	Folder dan File
Files	Koleksi dari semua isi obyek File.	Folder only
IsRootFolder	True jika merupakan folder akar pada drivanya.	Folder only
Name	Nama dari folder atau file. Memberikan nilai baru untuk <i>me-rename</i> obyek	Folder dan File
ParentFolder	Obyek Folder induk.	Folder dan File
Path	Jalur dari folder atau file. (Ini adalah properti default).	Folder dan File
ShortName	Nama obyek dalam format 8.3 MS-DOS.	Folder dan File
ShortPath	Jalur dari obyek dalam format 8.3 MS-DOS.	Folder dan File
Size	Ukuran dalam byte dari obyek file; jumlah ukuran dari semua file dan sub folder pada obyek folder.	Folder dan File
SubFolders	Koleksi dari semua sub folder pada folder, termasuk yang beratribut <i>system</i> dan <i>hidden</i> .	Folder only
Type	Sebuah string deskripsi dari obyek. Misalnya: <code>fso.GetFolder("C:\Recycled")</code> . Mengembalikan <i>Type</i> "Recycle Bin"; untuk obyek file, nilai ini tergantung pada ekstensinya (contoh, "Text Document" untuk sebuah ekstensi TXT).	Folder dan File

Tabel 6. Semua Metode dari Obyek Folder dan File.

Sintak	Deskripsi	Diterapkan Ke
Copy Destination, [OverWriteFiles]	Menyalin obyek file atau folder terpilih ke jalur lain; ini sama dengan metode <i>CopyFolder</i> dan <i>CopyFile</i> pada FSO, dimana mampu menyalin banyak obyek dalam satu operasi.	Folder dan File
CreateTextFile(FileName, [Overwrite], [Unicode]) As TextStream	Membuat file teks pada folder terpilih dan mengembalikan obyek <i>TextStream</i> yang bersangkutan. Lihat metode FSO yang bersangkutan untuk penjelasan dari argumen secara individual.	Folder only
Delete [Force]	Menghapus obyek File dan Folder (dengan semua subfolder dan file di dalamnya). Sama seperti metode <i>DeleteFile</i> dan <i>DeleteFolder</i> pada FSO.	Folder dan File
Move DestinationPath	Memindahkan obyek File dan Folder ke jalur lain; sama dengan metode <i>MoveFile</i> dan <i>MoveFolder</i> pada FSO.	Folder dan File
OpenAsTextStream([IOMode], [Format]) As TextStream	Membuka obyek File sebagai file teks dan mengembalikan obyek <i>TextStream</i> yang bersangkutan.	File only

OBYEK FILE

Obyek File merepresentasikan file tunggal dalam disk. Anda dapat mendapatkan referensi ke obyek tersebut dengan dua jalan: dengan menggunakan metode **GetFile** dari obyek FSO atau dengan mengiterasi melalui koleksi file dari obyek folder induknya.

```
' Daftar semua file DLL di direktori C:\WINDOWS\SYSTEM.
Dim fso As New Scripting.FileSystemObject, fil As Scripting.File
For Each fil In fso.GetSpecialFolder(SystemFolder).Files
    If UCase$(fso.GetExtensionName(fil.Path)) = "DLL" Then
        Print fil.Name
    End If
Next
```

OBYEK TEXTSTREAM

Obyek *TextStream* merepresentasikan sebuah file yang dibuka dalam mode teks. Anda dapat mendapatkan referensi ke obyek tersebut dengan cara berikut: dengan menggunakan metode **CreateTextFile** atau **OpenTextFile** dari obyek FSO, dengan menggunakan metode **CreateTextFile** dari obyek Folder, atau dengan menggunakan metode **OpenAsTextStream** dari obyek File. Obyek *TextStream* mengekspose sejumlah metode dan properti yang read-only, yang dilampirkan dalam Tabel 7.

Tabel 7. Semua Properti dan Metode dari Obyek TextStream.

Properti atau Metode	Sintak	Deskripsi
Properti	AtEndOfLine	True jika penunjuk file ada di akhir baris terpilih.
Properti	AtEndOfFile	True jika penunjuk file ada di akhir file (sama dengan fungsi EOF pada VBA).
Metode	Close	Menutup file (sama dengan pernyataan Close pada VBA).
Properti	Column	Nomer kolom terkini.
Properti	Line	Nomer baris terkini.
Metode	Read(Characters)	Membaca jumlah karakter yang dispesifikasikan dan mengembalikan sebuah string (sama dengan fungsi Input\$ pada VBA).
Metode	ReadAll()	Membaca isi file ke dalam sebuah string (sama dengan fungsi Input\$ pada VBA jika digunakan dengan fungsi LOF).
Metode	ReadLine()	Membaca baris berikut dari teks dan mengembalikan sebuah string (sama dengan pernyataan Line Input pada VBA).
Metode	Skip Characters	Meloncati sejumlah karakter yang telah dispesifikasikan.
Metode	SkipLine	Meloncati sebuah baris dari teks.
Metode	Write Text	Menulis sebuah string karakter, tanpa diikuti karakter <i>Newline</i> (baris baru). Sama dengan perintah Print# dengan diakhiri sebuah <i>semicolon</i> .
Metode	WriteBlankLines Lines	Menulis sejumlah baris kosong (sama dengan satu atau lebih perintah Print# tanpa argumen apapun).
Metode	WriteLine [Text]	Menulis sebuah string karakter, dengan diakhiri karakter <i>Newline</i> (sama dengan perintah Print# tanpa diakhiri <i>semicolon</i>).