Declaring Variables

Променливите се използват за съхраняване на стойности. В процедура или модул за деклариране на променлива се ползва Dim оператор. Синтаксиса му е:

Dim *variablename* [As *type*]

В този оператор се задават имена за променливите. Те трябва да отговарят на следните условия:

- Трябва да започват с буква.
- Не могат да съдържат точка и подобни символи.
- Не трябва да съдържат повече от 255 символа.
- Трябва да бъдат уникални в обхвата им.

Опцията As *type* се използва за посочване на типа на данните, които се обявяват. Типа на данните определя типа на информацията, която променливата може да съдържа. Примери за типова данни са String, Integer и Currency. Ако се пропусне тази клауза, Visual Basic задава за променливата тип Variant, който е тип по подразбиране (докато не се промени чрез оператора Def *type*). Променливите могат да включват и обекти от Access, такива като Form и Control обекти, или обекти от други приложения.

See Also For more information on manipulating objects, see Chapter 5, “Working with Objects and Collections.”

Главните и малките букви могат да се използват в произволни комбинации, но използването им не влияе върху променливите. Когато изпишете една променлива с друга комбинация от малки и главни букви, то Visual Basic я трансформира във вида, който сте задали в Dim оператора.

Tip When you name variables, you may find it useful to include prefixes to indicate each variable's data type. For example, you might name a variable dblTemp to indicate that its data type is Double. This practice makes your code easier to read and helps you avoid type mismatch errors.

Implicit Declaration

Въпреки, че не се препоръчва, може да използвате променливи без явно да ги декларирате. Например може да напишете следния код:

```
Function SafeSqr(ByVal dblNum As Double) As Double

dblTemp = Abs(dblNum)

SafeSqr = Sqr(dblTemp)

End Function
```

Тук няма деклариране на dblTemp като променлива. В такъв случай Visual Basic сам създава променлива с такова име, т.е. можете да пишете код без да декларирате променливи. Въпреки, че това е много удобно, то възможно е да се допусне грешка в изписването на имената. Например, представете си, че сте сгрешили в следващата функция:

```
Function SafeSqr(ByVal dblNum As Double) As Double

dblTemp = Abs(dblNum)
```

```
SafeSqr = Sqr(dblTmp)                    ' dblTemp is misspelled.

End Function
```

На пръв поглед тази функция е като предходната. Но тъй като dblTemp е сгрешена, тази функция винаги връща нула. Когато Visual Basic срещне нова променлива, той не може да знае дали това е наистина нова променлива или е грешка. Той приема, че искате да създадете променлива с име dblTmp и я създава за Вас.

Explicit Declaration

За заобикаляне на този проблем, задайте оператора Option Explicit в раздела за декларациите във Вашите модули. Този оператор ще Ви задължи да ползвате в модула само предварително обявени променливи. Ако Visual Basic срещне променлива, която не е предварително обявена, ще изведе съобщение за грешка.

Ако имаше оператор Option Explicit в модула, съдържащ горната функция SafeSqr, Visual Basic щеше да разпознае dblTemp и dblTmp като недекларирани променливи и щеше да изведе съобщения за грешка. Ще трябва явно да декларирате dblTemp, както е в следващия код:

```
Function SafeSqr(ByVal dblNum As Double) As Double

Dim dblTemp As Double

dblTemp = Abs(dblNum)
```

```
SafeSqr = Sqr(dblTmp)


End Function
```

Сега Access ще изведе съобщение за грешка при неправилно изписване на dblTmp и проблемът ще бъде изчистен. Затова Ви препоръчваме да включите Option Explicit оператор в своя код.

Можете да зададете задължително деклариране на променливите за всеки нов модул, като от командата Options (Tools менюто), изберете Module табулатора, и след това маркирате кутията Require Variable Declaration. Това ще доведе до автоматично включване на Option Explicit във всеки нов модул.

Note The Option Explicit statement operates on a per-module basis; it must be placed in the Declarations section of every form, report, and standard module for which you want Visual Basic to enforce explicit variable declarations. If you select the Require Variable Declaration check box, Visual Basic inserts the Option Explicit statement in all subsequent form, report, and standard modules, but doesn't add it to existing code. You must manually add the Option Explicit statement to any existing modules within an application.

See Also For more information on the Option Explicit statement, search the Help index for "Option Explicit statement."

Scope and Lifetime of Variables

Когато декларирате една променлива в процедура, само кода от тази процедура може да я ползва. Нейния обхват е локален за тази процедура. Много вероятно е да пожелаете да ползвате променлива, която да е достъпна от всички процедури в модула, или дори от всички процедури от всички модули. Във Visual Basic имате възможност да уточните обхвата на променливата когато я обявявате.

Scope of Variables

В зависимост от начина на обявяване, една променлива, може да бъде от ниво модул или от ниво процедура.

Scope

Private

Public

Procedure-level

Променливите са локални за процедурата в която са обявени.

Не могат да се декларират променливи тип public в процедура.

Module-level

Променливите са локални за модула в който са обявени.

Променливите са достъпни за всички модули.

Variables Used Within a Procedure

Променливите от процедурно ниво се разпознават само от процедурата в която са обявени. Известни са като локални променливи. Обявяват се с ключовите думи Dim и Static, както следва:

Dim intTemp As Integer

Static intPermanent As Integer

Локална променлива, декларирана с Dim съществува само по време на изпълнение на процедурата, докато тези с Static - за цялото време на изпълнение на приложението.

Note Implicitly declared variables always have a local scope.

See Also For more information on procedures, see Chapter 2, “Introducing Visual Basic.”

Variables Used Within a Module

Променливи от ниво модул са достъпни за всички процедури от модула, но не са достъпни от други модули. За обявяването им се ползват думите Dim или Private в раздела за деклариране в началото на модула. Например, възможни са следните оператори за обявяване:

```
Dim intTemp As Integer
```

```
Private intTemp As Integer
```

На ниво модул няма разлика между Dim и Private. Но ползването на Private се препоръчва, тъй като контрастира с Public, което прави и кода по-лесен за четене.

See Also For more information on modules, see Chapter 2, “Introducing Visual Basic.”

Variables Used by All Modules

За да се ползва една променлива от други модули, тя трябва да се обяви като Public, в

секцията за деклариране в началото на модула. Така обявена променлива е достъпна за всички процедури от всички модули на приложението. Пример за обявяване е такава променлива е:

```
Public intX As Integer
```

Note You can't declare public variables within a procedure.

От модул на форма, може да се обърнете към public променлива обявена в друга форма посредством записване от вида: Forms!Orders.intX.

See Also For more information on form and report modules, see Chapter 2, "Introducing Visual Basic."

Променливи от тип public могат да се декларират във всеки един модул, но много по удобно е всички такива променливи да са обявени само в един модул. Това прави кода много по-лесен за четене.

Scope and Variable Names

Една променлива не може да променя своя обхват по време на изпълнение на кода. Ако има public променливи с еднакви имена, обявени в различни модули, то правилното позоваване към тях включва и името на модула в който са обявени, например Module1.intX и Module2.intX.

Public vs. Local Variables

Може да има променливи с еднакви имена и различен обхват. Например, може да има

.local променлива с име intX в процедура, и public променлива с име intX. Обръщането към intX от процедурата ще даде обръщане към локалната променлива, а всяко обръщане към intX извън процедурата ще е обръщане към public променливата.

```
' This code is in the form module for Form1. Within the Test Sub

' procedure, the name intX always refers to the local variable, not

' to the variable declared in the Public statement at the top.

Public intX As Integer

Sub Test()

Dim intX As Integer

intX = 2                                        ' intX has a value of 2 (even though

MsgBox Forms!Form1.intX         ' MsgBox shows the value of intX in

End Sub                                                 ' Form1, which is 1).

Private Sub Form_Load()
```

```
intX = 1

End Sub

Private Sub Command1_Click()

Test                                                    ' intX has a value of 2.

End Sub
```

Using Variables and Properties with the Same Name

Правилата за подобие се отнасят за характеристиките и контролите на фирми и отчети, дефинирани потребителски типове, константи и процедури както за променливи от модулно ниво в модулите на форма или отчет. Не може да има характеристика или контрол във форма със същото име както променлива от модулно ниво, константа, дефиниран потребителски тип или процедура, понеже имат еднакъв обхват.

Във модул на форма или отчет, локалните променливи със същите имена като контролите във форма са подобни на контролите. Трябва или да посочите изрично контрол от формата, или трябва да се използва ключовата дума Me за получаване или задаване на всяка една тяхна характеристика. Следващият код показва единия възможен подход:

```
Private Sub Form_Click()

Dim Text1, Caption
```

```
' Assume there is also a control on the form called Text1.

Text1 = "Variable"                ' Variable shadows control.

Me.Text1 = "Control"              ' Must qualify with Me to get control.

Text1.Top = 0                     ' This causes an error!

Me.Text1.Top = 0                  ' Must qualify with Me to get control.

Caption = "Orders"                ' Variable shadows property.

Me.Caption = "Orders"     ' Must qualify with Me to get form property.

End Sub
```

Using Variables and Procedures with the Same Name

Възможни са проблеми с имената на променливите от локално модулно ниво и глобално модулно ниво с имена от процедурите. Променливите от модул не могат да имат същите имена като процедури или типове дефинирани в модула. Възможно е обаче, да има еднакви имена както глобални процедури, типове или променливи дефинирани в други модули. В този случай, когато променлива се извиква от друг модул, трябва да се посочи името на модула.

Тъй като тези правила на подобие могат да доведат до значителни проблеми, желателно е имената, които задавате за променливите да се различават от тези, ползвани за всичко останало в кода.

Lifetime of Variables

Променливите имат и продължителност (lifetime). Стойностите на променливи от локално и глобално модулно ниво се ползват докато формата или отчета са отворени. Локалните променливи от процедура са достъпни само докато процедурата се изпълнява, т.е. когато процедурата завърши паметта използвана от нейните променливи се освобождава и стойностите им се губят. След повторно изпълнение на процедурата нейните локални променливи преинициализират. Може да се укажи обаче на Visual Basic да направи локалните променливи като променливи тип static.

Static Variables

За да се направи една локална променлива от процедура от тип static, се ползва ключовата дума Static.

```
Static intX As Integer
```

Например, следващата функция пресмята нова стойност на базата на предишна стойност, съхранена в променливата от тип static dblAccumulate.

```
Function RunningTotal(ByVal dblNum As Double) As Double

Static dblAccumulate As Double
```

```
dblAccumulate = dblAccumulate + dblNum

RunningTotal = dblAccumulate

End Function
```

Ако беше декларирана променливата dblAccumulate с Dim вместо с ключовата дума Static, то предишната стойност на променливата нямаше да бъде съхранена за повторното изпълнение на функцията.

Същият резултат би могъл да се получи и като се декларира променливата в секцията за деклариране в модула, правейки я променлива от модулно ниво. Но когато се промени обхвата на променливата по този начин, и тя добие нова стойност чрез използването й от други процедури, то е възможно да се получат неверни резултати.

Declaring All Local Variables as Static

За да се направят всички променливи от тип Static, се поставя ключовата дума Static в началото на процедурата, както е в следващият пример:

```
Static Function RunningTotal(ByVal dblNum As Double) As Double
```

Това прави всички локални променливи от процедурата от тип static, независимо дали са обявени с ключовите думи Static или Dim. Този подход може да се прилага за всички Function и Sub процедури, включително и тези, които са декларирани като Private.

Fundamental Variable Data Types

Когато се обявява променлива може да се посочи и типа на данните за променливата. По подразбиране, когато не се посочи типа на променливата се приема, че тя е от тип Variant.

The Variant Data Type

Променлива от тип Variant може да съдържа данни от произволен тип. Не е нужно да се грижите за конвертиране между различните типове данни, тъй като това се прави автоматично от Visual Basic както е показано в следващият пример:

```
Dim varX                    ' Variant by default.

varX = "17"                 ' varX contains the two-character string "17".

varX = varX - 15            ' varX now contains the numeric value 2.

varX = "U" & varX           ' varX now contains the string "U2".
```

Може да се използва IsNumber функцията, за да се определи дали променливата от тип Variant съдържа коректна числена стойност, преди да я използвате в израз. Например:

```
If IsNumeric(varX) And IsNumeric(varY) Then

varZ = varX * varY
```

```
Else

varZ = Null

End If
```

Когато ще извършвате конкатенация на стрингове присвоени на променливи тип Variant, използвайте оператор “&”, а не оператор “+”.

Ако две променливи тип Variant съдържат числа, тогава “+” оператора задава действие събиране. Ако и двете обаче съдържат стрингове, оператор “+” задава конкатенация. Но когато едната е стринг, а другата число, ситуацията е по-сложна. Visual Basic първо се опитва да превърне стринга в число. Ако това приключи успешно, числата се събират, но ако не е възможно такова превръщане, се генерира съобщение за грешка - “Type mismatch”. Затова, за да бъдете сигурни че ще се изпълни точно действието конкатенация, използвайте оператор “&”, както е в следващият пример.

```
Sub Test()

Dim varX As Variant, varY As Variant

varX = "6"

varY = "7"

Debug.Print varX + varY, varX & varY
```

```
varX = 6

Debug.Print varX + varY, varX & varY

End Sub
```

в Debug прозореца ще се изведат следните резултати:

```
67              67

13              67
```

Important When typing your code, it's important to leave a space between any variable name and the & operator. If you don't leave a space, Visual Basic assumes you intended to use the & as the type-declaration character for the variable name.

Освен това променливите от тип Variant могат да съдържат стойности от тип дата/време, например:

```
Function Century() As Integer

Dim varToday As Variant

varToday = Now
```

```
If varToday >= #1/1/2001# Then

Century = 21

Else

Century = 20

End If

End Function
```

По същият начин, както се използва IsNumeric функцията, може да се ползва и функцията IsDate, за да се определи дали променливата от тип Variant съдържа коректна дата/време стойност, например:

```
Function Century (ByVal varDate As Variant) As Variant

If IsDate(varDate) Then

Century = ((Year(varDate) - 1)  100) + 1

Else
```

```
Century = Null

End If

End Function
```

The Empty Value

Когато се обяви една променлива от тип Variant, то тя добива стойност Empty. Това е специална стойност, различна от нула или празен стринг (“”), както и от Null стойност. Може да се ползва функцията IsEmpty, за да се определи дали променливата има стойност Empty.

```
If IsEmpty(varX) Then varX = 0
```

Когато използвате променлива със стойност Empty в израз, то тази стойност еволюира до нула (при израз с числа) и празен стринг (при конкатенация на стрингове).

The Null Value

Променливи от тип Variant могат да съдържат специалната стойност Null. Тя се използва за да се укажат липсващи данни. Полета и контроли, които не са били инициализирани имат стойност Null. Може да се ползва функцията IsNull, за да се определи дали променливата има стойност Null.

```
If IsNull(varX) And IsNull(varY) Then
```

varZ = Null

Else

varZ = 0

End If

Стойността Null има някои специфични характеристики:

- Изрази съдържащи променлива със стойност Null винаги връщат стойност Null.
- Повечето функции връщат стойност Null, ако им се зададе стойност Null като аргумент.

Стойност Null се задава с ключовата дума Null. Например: varZ = Null

Само променливи Variant могат да имат стойност Null. При присвояване на Null на променлива от друг тип, то ще се изведе съобщение за грешка.

Tip The fact that Null propagates makes it useful as an error value. If you write Function procedures that return Null when an error occurs, and then combine these functions in expressions, you can use the IsNull function to test the final result of the expression to see if an error has occurred. Because Null propagates, the final result is Null if an error has occurred in any of the functions; you don't have to test the result of each function separately.

Other Fundamental Data Types

Данните от тип Variant са универсални. Но често може да се създаде много по компактен и бърз код, чрез използване на другите типове данни. Например, ако променливата винаги ще приема цели стойности, в интервала от 1-500, може да спестите по няколко бита за всяка стойност, като укажете Integer тип за променливата, вместо Variant.

Типовете данни във Visual Basic са:

Data type	Description	Range
Byte	1-byte binary data	0 to 255
Integer		

2-byte integer

– 32,768 to 32,767

Long

4-byte integer

– 2,147,483,648 to 2,147,483,647

Single

4-byte floating-point number

– 3.402823E38 to – 1.401298E – 45 (negative values)

1.401298E – 45 to 3.402823E38 (positive values)

Double

8-byte floating-point number

– 1.79769313486231E308 to – 4.94065645841247E – 324
(negative values)

4.94065645841247E – 324 to 1.79769313486231E308
(positive values)

Currency

8-byte number with fixed decimal point

– 922,337,203,685,477.5808 to 922,337,203,685,477.5807

String

String of characters

Zero to approximately two billion characters

Variant

Date/time, floating-point number, integer, string,
or object. 16 bytes, plus
1 byte for each character if a string value.

Date values: January 1, 100 to December 31, 9999
Numeric values: same range as Double
String values: same range as String
Can also contain Error or Null values

Boolean

2 bytes

True or False

Date

8-byte date/time value

January 1, 100 to December 31, 9999

Object

4 bytes

Any Object reference

See Also You can also declare arrays of any of these fundamental data types. For more information, see “Arrays” later in this chapter.

Когато се декларира променлива посредством операторите Dim, Public, Private или Static, се използва As *type* клаузата, за да се уточни типа на данните за променливата. Следващият пример декларира съответно Integer, Currency, Double и String променливи:

```
Dim intX As Integer

Public curBillsPaid As Currency

Private dblAmt As Double

Static strName As String
```

See Also For more information on the Dim, Public, Private, or Static statements, search the Help index for the name of the statement.

Операторът за обявяване може да комбинира обявяване на няколко променливи:

```
Dim intX As Integer, dblAmt As Double

Dim strName As String, curBillsPaid As Currency
```

Dim varTest, intY As Integer, varAmount

Important In a multiple declaration statement, you must use the As *type* clause for each variable whose data type you want to specify. If you don't specify a data type, Visual Basic declares the variable as Variant. In the last line of code in the preceding example, Visual Basic declares only the variable intY as Integer. The variables Test and Amount are each declared as Variant.

Most of the Visual Basic data types match the data types for fields that contain data. The few field data types that aren't directly matched by a Visual Basic data type can be handled by another Visual Basic data type.

Field data type	Compatible Visual Basic data type
AutoNumber (Long Integer)	Long
AutoNumber (Replication ID)	No compatible Visual Basic type

Currency

Currency

Date/Time

Date

Hyperlink

No compatible Visual Basic type

Memo

String

Number (Byte)

Byte

Number	(Integer)
Integer	
Number	(Long Integer)
Long	
Number	(Single)
Single	
Number	(Double)
Double	
Number	(Replication ID)
No	compatible Visual Basic type

OLE Object

Array with the Byte data type

Text

String

Yes/No

Boolean

Note SQL data types are also used in Microsoft Access queries. For information on these data types, search the Help index for “ANSI SQL data types.”

Ако една променлива има вероятност да приема и стойност Null, то трябва да се декларира като Variant.

Numeric Data Types

Ако сте наясно, че променливата винаги ще съхранява цяло число (например 12), декларирайте я като тип Integer. Операциите тогава се изпълняват значително по-бързо и се ангажира значително по-малко памет. Този тип е особено удобен за брояч в For... Next оператор.

Ако променливата ще съдържа дробна част, декларирайте я като Single, Double или Currency променлива. Променливи от тип Single и Double имат много по-малък диапазон, но същевременно при тях грешките са много по-малки при закръгляне, отколкото при Currency.

Note Floating-point values can be expressed as *mmm*E*eee*, in which *mmm* is the mantissa and *eee* is the exponent (a power of ten). The highest positive value of a Single data type is 3.402823E+38, or 3.4 times 10 to the 38th power; the highest positive value of a Double data type is .79769313486231E+308, or about 1.8 times 10 to the 308th power.

Ако променливата ще съдържа двоични данни следва да се декларира като Byte тип на данните. Всички аритметични оператори работят и с двоичен тип данни.

Всички променливи от числов тип могат да получават стойности една от друга, включително и Variant тип. Visual Basic закръгля (не отрязва) дробната част на променливите с плаваща запетая преди присвояването им към целочислени променливи.

The String Data Type

Когато едно променлива ще съдържа символи (не числа), то тя следва да се обяви като променлива от тип String.

Dim strAny As String

Такава променлива може да се обработва със стрингови функции.

strAny = "Database"

```
strAny = Left(strAny, 4)
```

Стринг с фиксирана дължина в стандартен модул следва да се обяви като Public или Private. В модул на форма или отчет, такъв стринг се обявява чрез Private.

По подразбиране променливите от тип стринг са с променлива дължина, което означава, че може да приемат стрингове с ръзлична дължина. Може да се укаже обаче, че променливата ще приема стрингове само с фиксирана дължина, което се постига със следния синкаксис:

String * *size*

Следващият код декларира променлива с фиксирана дължина 50 символа:

```
Dim strEmpName As String * 50
```

Ако се присвои на променливата strEmpName стринг с по-малко от 50 символа, то се извършва допълване с интервали до фиксираният размер на променливата, а ако се присвои стринг с повече от 50 символа, то излишните се отрязват. Например кода:

```
Dim strJust4 As String * 4

Dim strAny As String

strAny = "Database"
```

Debug.Print strAny

strJust4 = strAny

Debug.Print strJust4

извежда следният резултат в прозорец Debug:

Database

Data

Тъй като стринговите променливи с фиксирана дължина се попълва с интервали, при тях често се ползват функциите Trim, LTrim и RTrim.

See Also For more information on the Trim, LTrim, or RTrim functions, search the Help index for the name of the function.

Visual Basic сравнява стринговете по няколко начина, в зависимост от оператор Option Compare включен в секцията на декларациите в модулите. Може да се ползват възможностите: Option Compare Database, Option Compare Binary или Option Compare Text.

Access автоматично вмъква оператор Option Compare Database във всеки нов модул, с което се приема, че стринговете ще се сравняват на базата на приетия начин за сортиране в базата от данни.

See Also For more information on the Option Compare statement, search the Help index for “Option Compare statement.”

The Boolean Data Type

Ако в една променлива винаги ще се съхраняват стойности от тип yes/no или on/off, то тя може да се обяви като тип Boolean. Стойността по подразбиране за този тип е False. В следващият пример, blnCreditExceeded е Boolean променлива, която приема стойности True или False.

```
Dim blnCreditExceeded As Boolean

' Add all charges.

Do Until rstCharges.EOF

curAmt = curAmt + rstCharges("Amount").Value

rstCharges.MoveNext

Loop

' Ask if the credit limit is exceeded.
```

```
If curAmt > curLimit Then blnCreditExceeded = True
```

The Date Data Type

Стойности от тип дата/време могат да се съхраняват в променливи от тип Data или Variant.

See Also For more information, see “The Variant Data Type” earlier in this chapter.

Когато други числови променливи се конвертират в Date променливи, стойностите от ляво на десетичния знак са информация за датата, а от дробната част са за времето. Полунощ е 0, а обедно време е 0.5. Отрицателните числа представят дати преди December 30, 1899.

The Object Data Type

Променливите от тип object се използват (чрез оператор Set) за обръщане към всеки реален обект от приложението, например:

```
Const conFilePath As String = "C:Program FilesMicrosoft OfficeOfficeSamples"

Dim objDb As Object

Set objDb = OpenDatabase(conFilePath & "Northwind.mdb")
```

Tip You may want to think of the objects listed in the Classes box in the Object Browser as

additional data types that are available to you. You can declare objects from other applications —and other applications can declare objects from your application—in the same way that you declare ordinary object data types in Visual Basic.

See Also For more information on objects and the Object Browser, see Chapter 5, “Working with Objects and Collections.”

Argument Data Types

Аргументите за процедурите, които пишете, ползват Variant тип по подразбиране. Но може да се декларират и други типове за аргументите. Например следващата функция приема аргументи от тип String и Integer:

```
Function Reverse (strAny As String, ByVal intChars As Integer) As String

' Reverses the first intChars characters in strAny.

Dim strTemp As String, intCount As Integer

If intChars > Len(strAny) Then intChars = Len(strAny)

For intCount = intChars To 1 Step - 1

strTemp = strTemp + Mid(strAny, intCount, 1)

Next
```

```
Reverse = strTemp + Right(strAny, Len(strAny) - intChars)

End Function
```

Function Return Data Types

Стойностите връщани от функциите имат тип на данните. Когато се дефинира функцията може да се посочи и типа на данните, които тя ще връща.

Както при променливите, Visual Basic работи много по ефикасно с функциите, ако явно е указан типа на стойностите, които те връщат. Ако не се посочи типа на връщаната стойност, функциите ползват Variant.

Creating Your Own Data Types

Допустимо е комбиниране на променливи от различни типове за създаване на потребителски тип данни (познати като *structures* в езика C). Потребителският тип данни има приложение, когато се желае единична променлива да съхранява свързани порции от информация.

Declaring a User-Defined Type

Дефиниран от потребителя тип се задава с оператор Type, който трябва да бъде разположен в секцията за деклариране в стандартния модул. Този тип може да се определи чрез ключовите думи Private или Public, както е показано в следващите примери:

Private Type YourType

Public Type YourType

Например, може да се създаде потребителски тип, който да съдържа информация за компютърната система, чрез разполагането на следващият код в раздела за деклариране:

Public Type SystemInfo

varCPU As Variant

lngMemory As Long

intVideoColors As Integer

curCost As Currency

dtePurchase As Variant

End Type

Declaring Variables of a User-Defined Type

Може да се декларират локални, private модулно ниво и public модулно ниво променливи на потребителския тип. Например:

```
Dim sysMine As SystemInfo, sysYours As SystemInfo
```

Следващата таблица илюстрира къде, и с какъв обхват можете да декларирате потребителски тип и неговите променливи.

Procedure/Module	
You can create a user-defined type as	
Variables of a user-defined type can be	
Procedures	
Not applicable	
Not applicable	

Standard modules

Private or public

Private or public

Form or report modules

Private only

Private only

Assigning and Retrieving Values

Присвояване и връщане на стойности на/от елементите на потребителски тир променлива е като задаването и получаването на характеристиките:

sysMine.varCPU = "486"

If sysMine.dtePurchase > #1/1/92# Then

Може да се присвоява стойност на променлива към друга, ако и двете са от същия

потребителски тип. Това присвоява всички елементи от едната на съответстващите елементи от другата.

sysYours = sysMine

User-Defined Types That Contain Arrays

Потребителски дефиниран тип може да съдържа и фиксиран (fixed-size) масив, както е на следващият пример:

Type SystemInfo

varCPU As Variant

lngMemory As Long

strDiskDrives(25) As String ' Fixed-size array.

intVideoColors As Integer

curCost As Currency

dtePurchase As Variant

```
End Type
```

Този тип може да съдържа и динамичен (dynamic) масив:

```
Type SystemInfo

varCPU As Variant

lngMemory As Long

strDiskDrives() As String                    ' Dynamic array.

intVideoColors As Integer

curCost As Currency

dtePurchase As Variant

End Type
```

Достъпът до стойностите в масив от потребителски тип е идентичен с достъпа до характеристиката на обект, например:

```
Dim sysMine As SystemInfo
```

sysMine.strDiskDrives(0) = "1.44 MB"

Може да се декларира масив от потребителски тип:

Dim sysAll(100) As SystemInfo

И тук важат същите правила за достъп до компонентите на всеки елемент от масива:

sysAll(5).varCPU = "386SX"

sysAll(intX).strDiskDrives(2) = "100M SCSI"

See Also For more information on arrays, see “Arrays” later in this chapter.

Declaring Procedure Arguments

Може да се декларира и процедурен аргумент от потребителски тип.

Sub FillSystem(sysAny As SystemInfo)

sysAny.varCPU = "486"

```
sysAny.lngMemory = "24"

sysAny.curCost = "$3000.00"

sysAny.dtePurchase = Now

End Sub
```

Note If you want to pass a user-defined type in a form or report module, the procedure must be private.

Може да се връща потребителски тип от функция, както и да се задава потребителски тип променлива в процедурата като един от аргументите, както е посочено в предходния пример.

User-Defined Types That Contain Objects

Потребителският тип може да съдържа също и обекти, например:

```
Private Type AccountPack

frmInput As Form

dbsPayRollAccount As Database
```

End Type

Tip Because the Variant data type can store many different types of data, a Variant array can be used in many situations where you may expect to use a user-defined type. A Variant array is actually more flexible than a user-defined type, because you can change the type of data you store in each element at any time, and you can make the array dynamic so that you can change its size as necessary. However, a Variant array always uses more memory than an equivalent user-defined type.

Nesting Data Structures

Влагането на структури от данни е възможно, когато това Ви е необходимо. Потребителски тип може да съдържа други потребителски тип, както е посочено в следващият пример. За да е кода лесен за четене и поправяне на съществуващи в него грешки, старайте се да разполагате целият код, дефиниращ потребителски типове да бъде в един модул, например:

Type DriveInfo

Type As String

Size As Long

End Type

Type SystemInfo

varCPU As Variant

lngMemory As Long

strDiskDrives(26) As DriveInfo

curCost As Currency

dtePurchase As Variant

End Type

Следващият код показва как може да се обърнете към вложен потребителски тип:

Dim sysAll(100) As SystemInfo

sysAll(1).strDiskDrives(0).Type = "Floppy"

Constants

Вашият код може да съдържа непроменящи се стойности, които се използват многократно.

В такъв случай трябва да използвате константи. Не можете да променяте стойностите на константите както е при променливите. Константите са два вида:

- *Вътрешни или системни (Intrinsic* or *system-defined)* константи са компоненти на приложението, които се ползват наготово от потребителя.

- *Символни или дефинирани от потребителя (Symbolic* or *user-defined)* константи, декларирани с оператор Const.

See Also For more information on intrinsic constants, see “Intrinsic Constants” later in this chapter. For more information on user-defined constants, see the following section, “Creating Your Own Constants.”

Creating Your Own Constants

Чрез оператор Const се декларират символните константи. Синтаксиса на Const оператора е:

[Public | Private] Const *constantname* [As *type*] = *expression*

Аргумента *constantname* е валидно символно име (правилата са същите прилагани за имена на променливите), а *expression* е съставен от цифрови и стрингови константи и оператори.

Const оператора може да задава математически или дата/време стойности, например:

Const conPi = 3.14159265358979

Public Const conMaxPlanets = 9

```
Const conReleaseDate = #1/1/95#
```

Този оператор може да се ползва и за задаване на стрингови константи:

```
Public Const conVersion = "07.10.A"
```

```
Const conCodeName = "Enigma"
```

В един ред може да се зададат няколко константи, ако се разделят със запетая:

```
Public Const conPi = 3.14, conMaxPlanets = 9, conWorldPop = 6E+09
```

Изразите от дясно на знака за присвояване са често числа или стрингове, но могат да бъдат и изрази, резултиращи в число или стринг, но този израз не може да включва функции. В израза може да се ползват също и предварително дефинирани константи:

```
Const conPi2 = conPi * 2
```

След като се декларира константа, тя може да се ползва в кода, за да стане той по лесен за четене, както е на следващият пример:

```
Static SolarSystem(1 To conMaxPlanets)
```

```
If lngNumPeople > conWorldPop Then Exit Function
```

Defining the Scope of User-Defined Constants

Const оператора има обхват както при променливите, като се прилагат следните правила:

- За създаване на константа достъпна само от процедура, то тя се декларира в процедурата.
- За създаване на константа, достъпна от всички процедури в модула, но не и от процедури извън модула, константата се декларира в раздела за обявяванията в модула.
- За създаване на константа достъпна за цялото приложение, то тя се декларира в раздела за обявяванията на стандартен модул и преди оператор Const се поставя ключовата дума Public. Константи от този тип не могат да се декларират в модули на форма или отчет.

Avoiding Circular References

Трябва да сте предупредени за едно възможно усложнение когато използвате public константи. Тъй като константа може да бъде дефинирана чрез друга константа, трябва да бъдете много внимателни за да не се получи циклично позоваване между две или повече константи. Циклично позоваване се получава, когато има две или повече константи, всяка от които е дефинирана на базата на другата, както е в следващият пример:

```
' In Module 1.

Public Const conA = conB * 2

' In Module 2.

Public Const conB = conA / 2
```

Intrinsic Constants

Освен с константите, декларирани с Const, Access автоматично предоставя и вътрешни константи. Те са достъпни във всички модули.

Тъй като не може да ги отстраните, то Вашите константи не могат да имат същите имена като вътрешните константи.

Important Because the values represented by the intrinsic constants may change in future versions of Microsoft Access, you should use the constants instead of their actual values.

Вътрешните константи се ползват като декларираните от потребителя. Следващият пример как се ползва вътрешната константа vbCurrency за определяне дали varTest е от тип Variant - VarType (Currency).

```
If VarType(varTest) = vbCurrency Then

Debug.Print "varTest contains Currency data."

Else

Debug.Print "varTest does not contain Currency data."

End If
```

Списък на вътрешните константи на Access може да се видят в Object Browser. За да ги видите, чукнете върху съответната библиотека на обекта (object library) в Project/Library кутията на Object Browser, и след точа чукнете Constants в Classes кутията.

Important In order for constants from other applications to appear in the Object Browser, you must set a reference to the application's object library. Open a module, and then in the References dialog box (Tools menu), select the check box for the object library you want to set a reference for.

See Also For more information on using the Object Browser, see Chapter 5, "Working with Objects and Collections."

Имената на вътрешните константи започват с двусимволен префикс, определящ библиотеката в която тя е дефинирана. Имената на константите на Access започват с "ac", например acDataErrContinue. Константите от VBA библиотеката започват с "vb"; например vbTileHorizontal. Константите от DAO библиотеката започват с "db"; например dbRelationUnique. Всички тези вътрешни константи са достъпни от Access без да е необходимо да се декларират.

Note In Microsoft Access versions 2.0 and earlier, constant names appeared in all capital letters with underscores—for example, A_REFRESH. Microsoft Access 97 supports intrinsic constants from these early versions. To see them, click the Access object library in the Project/Library box of the Object Browser, and then click OldConstants in the Classes box.

See Also For a list of intrinsic constants and more information about them, search the Help index for "intrinsic constants."

Arrays

Масивите са серия от променливи със същото име, при които се използва индекс. Това дава възможност да се правят кратки кодове в много ситуации. Масивите имат долна и горна граница, и елементите на масива са в тези граници.

Всички елементи от един масив са от един и същи тип.

See Also For more information on user-defined types, see “Creating Your Own Data Types” earlier in this chapter. For information on object types, see Chapter 5, “Working with Objects and Collections.”

Declaring Fixed-Size Arrays

Декларирането на масив с фиксиран размер (ordinary, fixed-size) е възможно по три начина, в зависимост от обхвата който е необходим:

- За създаването на *public масив*, използвайте Public оператор в секцията за деклариране на модула където декларирате масива.
- За създаването на *module-level масив*, използвайте Private или Dim оператор в секцията за деклариране на модула където декларирате масива.
- За създаването на *local масив*, използвайте Dim или Static оператор в процедурата за обявяването на масив.

Съществуват допълнителни правила, които се прилагат при динамичните масиви.

See Also For more information on dynamic arrays, see “Dynamic Arrays” later in this chapter.

Setting Upper and Lower Bounds

Когато обявявате масив, след името на масива се задава горната граница на елементите му в скоби. Тази стойност е от тип Long (в интервала от -231 до 231). Например, следващият масив може да се декларира в секцията за обявяванията в модул:

```
Dim intCount(14) As Integer          ' Declares an array with 15
 ' elements (0 through 14).

Dim dblSum(20) As Double                              ' Declares an array with 21
 ' elements (0 through 20).
```

За public масив, се поставя Public на мястото на Dim (или Private):

```
Public intCount(14) As Integer

Public dblSum(20) As Double
```

За създаване на local масив, се ползва Dim (или Static) оператор:

```
Dim intCount(14) As Integer

Static dblSum(20) As Double
```

Първото обявяване създава масив с 15 елемента, с индексни номера от 0 до 14. Второто създава масив с 21 елемента, с индексни номера от 0 до 20. Подразбиращата се долна граница е 0, но тя може да се промени, например на 1, чрез разполагането на следващият Option Base оператор в секцията за обявяване на модула:

```
Option Base 1
```

See Also For more information on the Option Base statement, search the Help index for “Option Base statement.”

Друга възможност за задаване на долната граница е чрез явното й указване с ключовата дума To. Например:

```
Dim intCount(1 To 15) As Integer
```

```
Dim dblSum(100 To 120) As Double
```

В тези обявявания, индексите за intCount са от 1 до 15 (15 елемента), а индексите за dblSum са от 100 до 120 (21 елемента).

Note With some versions of Basic, you can use an array without first declaring it. With Visual Basic, this isn’t possible; you must declare an array before using it.

Multidimensional Arrays

Във Visual Basic може да се декларират масиви с до 60 размерности. Например, следващият оператор обявява двумерен масив с 10 на 10 елемента в процедура:

```
Static dblMatrix(9, 9) As Double
```

Едната или и двете размерности могат да бъдат с явно посочване на долната граница:

```
Static dblMatrix(1 To 10, 1 To 10) As Double
```

Обявяването на тримерен масив е посочено в следващият пример:

```
Dim intMultiD(3, 1 To 10, 1 To 15) As Integer
```

Това обявяване създава тримерен масив 4 на 10 на 15. Общия брой на елементите в този масив е 4 x 10 x 15 = 600 елемента.

Note Because the total storage needed by the array increases dramatically when you start adding dimensions to it, be sure to use multidimensional arrays with care. Be especially careful with Variant arrays, because Variant arrays are larger than arrays containing other data types.

Using Loops to Manipulate Arrays

Цикличната обработка представлява удобен начин за управление на масиви. Например, следващата обработка задава стойност 5 на всеки елемент на масива:

```
Static intCount(1 To 15) As Integer

Dim intX As Integer

For intX = 1 To 15

intCount(intX) = 5
```

```
Next intX
```

За многомерни масиви удобни са вложените For. Например, следващите оператори задават стойности на елементите от масива съответстващи на местоположението им в него:

```
Dim intX As Integer, intY As Integer

Static dblMatrix(1 To 10, 1 To 10) As Double

For intX = 1 To 10

For intY = 1 To 10

dblMatrix(intX, intY) = intX * 10 + intY

Next intY

Next intX
```

Dynamic Arrays

Понякога не е възможно предварително да знаете колко голям да създадете един масив. Ще се наложи да се променя размера му в процеса на работа на кода. Тогава масива се обявява като динамичен.

To create a dynamic array

1 1 Деклариране на масив с Public, Private или Dim оператор на модулно ниво (ако е необходим масив на модулно ниво) или с операторите Static или Dim на процедурно ниво (ако е необходимо масива да е локален). Задаването на динамичен масив става със задаването на празен списък за размерност, например:

```
Dim intDynArray() As Integer
```

2 2 Определянето на реалния брой елементи в масива става с оператор ReDim. Например:

```
ReDim intDynArray(intX + 1)
```

Операторът ReDim може да се ползва само еднократно в процедура.

See Also For more information on array syntax, see “Arrays” earlier in this chapter.

Например, създавате динамичен масив intMatrix чрез деклариране на модулно ниво:

```
Dim intMatrix() As Integer
```

Във функцията след това се определя броя на елементите в масива:

```
Function CalcValuesNow () As Integer

ReDim intMatrix(19, 29)

End Function
```

ReDim операторът задава размерност 20 x 30 (общо 600 елемента). Възможно е да се ползват и променливи в този оператор:

```
ReDim intMatrix(intX, intY)
```

Preserving the Contents of Dynamic Arrays

Важно е да се знае, че всяко ново изпълнение на ReDim оператор, всички текущи стойности на масива се губят. Visual Basic инициализира елементите на масива с Empty стойност (за Variant масиви), с нула (за числови масиви), със стринг с нулева дължина ("") (за стрингови масиви), или с Nothing (за масиви с обекти).

Това е полезно, когато желаете да подготвите масива за нови данни или да го намалите за да заема малко памет. Но ако желаете да промените размера на масива без да губите данните от него, то трябва да се ползва ReDim с ключовата дума Preserve. Например, може да увеличите масива с един елемент без да губите стойностите от него, както е показано тук:

```
ReDim Preserve intMatrix(UBound(intMatrix) + 1)
```

Само горната граница на последната размерност може да се увеличава чрез Preserve. При опит за промяна на границите на другите размерности се получава грешка, т.е.

можете да ползвате следващият код:

```
ReDim Preserve intMatrix(10, UBound(intMatrix, 2) + 1)
```

Но не можете да ползвате следващият код:

```
ReDim Preserve intMatrix(UBound(intMatrix, 1) + 1, 10)
```

See Also For more information on the Preserve keyword, search the Help index for “ReDim statement.”