access key

A key pressed while holding down the ALT key that allows the user to open a menu**,** carry out a command, select an object, or move to an object. For example, ALT+F opens the **File** menu.

ActiveX control

An object that you place on a form to enable or enhance a user's interaction with an application. ActiveX Controls have events and can be incorporated into other controls. These controls have an .ocx file name extension.

ActiveX object

An object that is exposed to other applications or programming tools through Automation interfaces.

add-in

A customized tool that adds capabilities to the Visual Basic development environment.

ANSI Character Set

American National Standards Institute (ANSI) 8-bit character set used to represent up to 256 characters (0 – 255) using your keyboard. The first 128 characters (0 – 127) correspond to the letters and symbols on a standard U.S. keyboard. The second 128 characters (128 – 255) represent special characters, such as letters in international alphabets, accents, currency symbols, and fractions.

application

A collection of code and visual elements that work together as a single program. Developers can build and run applications within the development environment, while users usually run applications as executable files outside the development environment.

argument

A constant, variable, or expression passed to a procedure.

array

A set of sequentially indexed elements having the same intrinsic data type. Each element of an array has a unique identifying index number. Changes made to one element of an array don't affect the other elements.

ASCII Character Set

American Standard Code for Information Interchange (ASCII) 7-bit character set used to represent letters and symbols found on a standard U.S. keyboard. The ASCII character set is the same as the first 128 characters (0 – 127) in the ANSI character set.

automatic formatting

A feature that automatically formats code as you enter it by capitalizing the first letter for keywords, standardizing spacing, adding punctuation, and setting the foreground and background colors.

Automation object

An object that is exposed to other applications or programming tools through Automation interfaces.

base class

Original class from which other classes can be derived by inheritance.

bitmap

An image represented by pixels and stored as a collection of bits in which each bit corresponds to one pixel. On color systems, more than one bit corresponds to each pixel. A bitmap usually has a .bmp file name extension.

bitwise comparison

A bit-by-bit comparison between identically positioned bits in two numeric expressions.

Boolean data type

A data type with only two possible values, **True** (-1) or **False** (0). **Boolean** variables are stored as 16-bit (2-byte) numbers.

Boolean expression

An expression that evaluates to either **True** or **False**.

bound control

A data-aware control that can provide access to a specific field or fields in a database through a **Data** control. A data-aware control is typically bound to a **Data** control through its **DataSource** and **DataField** properties. When a **Data** control moves from one record to the next, all bound controls connected to the **Data** control change to display data from fields in the current record. When users change data in a bound control and then move to a different record, the changes are automatically saved in the database.

break mode

Temporary suspension of program execution in the development environment. In break mode, you can examine, debug, reset, step through, or continue program execution. You enter break mode when you:

- Encounter a breakpoint during program execution.
- Press CTRL+BREAK during program execution.
- Encounter a **Stop** statement or untrapped run-time error during program execution.
- Add a **Break When True** watch expression. Execution stops when the value of the watch changes and evaluates to **True**.
- Add a **Break When Changed** watch expression. Execution stops when the value of the watch changes.

breakpoint

A selected program line at which execution automatically stops. Breakpoints are not saved with your code.

by reference

A way of passing the address of an argument to a procedure instead of passing the value. This allows the procedure to access the actual variable. As a result, the variable's actual value can be changed by the procedure to which it is passed. Unless otherwise specified, arguments are passed by reference.

by value

A way of passing the value of an argument to a procedure instead of passing the address. This allows the procedure to access a copy of the variable. As a result, the variable's actual value can't be changed by the procedure to which it is passed.

Byte data type

A data type used to hold positive integer numbers ranging from 0 – 255. Byte variables are stored as single, unsigned 8-bit (1-byte) numbers.

character code

A number that represents a particular character in a set, such as the ANSI character set.

class

The formal definition of an object. The class acts as the template from which an instance of an object is created at run time. The class defines the properties of the object and the methods used to control the object's behavior.

class module

A module that contains the definition of a class, including its property and method definitions.

code module

A module containing public code that can be shared among all modules in a project. A code module is referred to as a standard module in later versions of Visual Basic.

code pane

A pane contained in a code window that is used for entering and editing code. A code window can contain one or more code panes.

collection

An object that contains a set of related objects. An object's position in the collection can change whenever a change occurs in the collection; therefore, the position of any specific object in the collection can vary.

command line

The path, file name, and argument information provided by the user to run a program.

comment

Text added to code that explains how the code works. In Visual Basic, a comment line can start with either an apostrophe (**'**) or with the **Rem** keyword followed by a space.

comparison operator

A character or symbol indicating a relationship between two or more values or expressions. These operators include less than (**<**), less than or equal to (**<=**), greater than (**>**), greater than or equal to (**>=**), not equal (**<>**), and equal (**=**). Additional comparison operators include **Is** and **Like**. Note that **Is** and **Like** can't be used as comparison operators in a **Select Case** statement.

compile time

The period during which source code is translated to executable code.

compiler directive

A command used to alter the action of the compiler.

conditional compiler constant

A Visual Basic identifier that is defined using the **#Const** compiler directive or defined in the host application and used by other compiler directives to determine when or if certain blocks of Visual Basic code are compiled.

constant

A named item that retains a constant value throughout the execution of a program. A constant can be a string or numeric literal, another constant, or any combination that includes arithmetic or logical operators except **Is** and exponentiation. Each host application can define its own set of constants. Additional constants can be defined by the user with the **Const** statement. You can use constants anywhere in your code in place of actual values.

container

An object that can contain other objects.

control

An object you can place on a form that has its own set of recognized properties, methods, and events. You use controls to receive user input, display output, and trigger event procedures. You can manipulate most controls using methods. Some controls are interactive (responsive to user actions), while others are static (accessible only through code).

control array

A group of controls that share a common name, type, and event procedures. Each control in an array has a unique index number that can be used to determine which control recognizes an event.

Currency data type

A data type with a range of -922,337,203,685,477.5808 to 922,337,203,685,477.5807. Use this data type for calculations involving money and for fixed-point calculations where accuracy is particularly important. The at sign (**@**) type-declaration character represents **Currency** in Visual Basic.

data type

The characteristic of a variable that determines what kind of data it can hold. Data types include **Byte**, **Boolean**, **Integer**, **Long**, **Currency**, **Decimal**, **Single**, **Double**, **Date**, **String**, **Object**, **Variant** (default), and user-defined types, as well as specific types of objects.

Date data type

A data type used to store dates and times as a real number. Date variables are stored as 64-bit (8-byte) numbers. The value to the left of the decimal represents a date, and the value to the right of the decimal represents a time.

date expression

Any expression that can be interpreted as a date, including date literals, numbers that look like dates, strings that look like dates, and dates returned from functions. A date expression is limited to numbers or strings, in any combination, that can represent a date from January 1, 100 – December 31, 9999.

Dates are stored as part of a real number. Values to the left of the decimal represent the date; values to the right of the decimal represent the time. Negative numbers represent dates prior to December 30, 1899.

date literal

Any sequence of characters with a valid format that is surrounded by number signs (**#**). Valid formats include the date format specified by the locale settings for your code or the universal date format.

For example, `#12/31/92#` is the date literal that represents December 31, 1992, where English-U.S. is the locale setting for your application. Use date literals to maximize portability across national languages.

date separators

Characters used to separate the day, month, and year when date values are formatted. The characters are determined by system settings or by the **Format** function.

Decimal data type

A data type that contains decimal numbers scaled by a power of 10. For zero-scaled numbers, that is, numbers with no decimal places, the range is +/-79,228,162,514,264,337,593,543,950,335. For numbers with 28 decimal places the range is +/-7.9228162514264337593543950335. The smallest non-zero number that can be represented as a **Decimal** is 0.0000000000000000000000000001.

Note that at this time the **Decimal** data type can only be used within a **Variant**. You cannot declare a variable to be of type **Decimal**. You can, however, create a **Variant** whose subtype is **Decimal** using the **CDec** function.

declaration

Nonexecutable code that names a constant, variable, or procedure, and specifies its characteristics, such as data type. For DLL procedures, declarations specify names, libraries, and arguments.

designer

Provides a visual design window in the Visual Basic development environment. You can use this window to design new classes visually. Visual Basic has built-in designers for forms. The Professional and Enterprise editions of Visual Basic include designers for ActiveX controls and ActiveX documents.

design time

The time during which you build an application in the development environment by adding controls, setting control or form properties, and so on. In contrast, during run time, you interact with the application like a user.

development environment

The part of the application where you write code, create controls, set control and form properties, and so on. This contrasts with running the application.

docked window

A window that is attached to the frame of the main window.

document

Any self-contained work created with an application and given a unique file name.

DBCS

A character set that uses 1 or 2 bytes to represent a character, allowing more than 256 characters to be represented.

Double data type

A data type that holds double-precision floating-point numbers as 64-bit numbers in the range -1.79769313486232E308 to -4.94065645841247E-324 for negative values; 4.94065645841247E-324 to 1.79769313486232E308 for positive values. The number sign (**#**) type-declaration character represents the **Double** in Visual Basic.

dynamic data exchange (DDE)

An established protocol for exchanging data through active links between applications that run under Microsoft Windows.

dynamic-link library (DLL)

A library of routines loaded and linked into applications at run time. DLLs are created with other programming languages such as C, MASM, or FORTRAN.

Empty

Indicates that no beginning value has been assigned to a **Variant** variable. An **Empty** variable is represented as 0 in a numeric context or a zero-length string ("") in a string context.

error number

A whole number in the range 0 – 65,535, that corresponds to the **Number** property setting of the **Err** object. When combined with the **Name** property setting of the **Err** object, this number represents a particular error message.

event source object

An object that is the source of events that occur in response to an action. An event source object is returned by a property. For example, the **CommandBarEvents** property returns the **CommandBarEvents** object.

executable file

A Windows-based application that can run outside the development environment. An executable file has an .exe file name extension.

expression

A combination of keywords, operators, variables, and constants that yields a string, number, or object. An expression can be used to perform a calculation, manipulate characters, or test data.

file number

Number used in the **Open** statement to open a file. Use file numbers in the range 1 – 255, inclusive, for files not accessible to other applications. Use file numbers in the range 256 – 511 for files accessible from other applications.

focus

The ability to receive mouse clicks or keyboard input at any one time. In the Microsoft Windows environment, only one window, form, or control can have this ability at a time. The object that "has the focus" is normally indicated by a highlighted caption or title bar. The focus can be set by the user or by the application.

form

A window or dialog box. Forms are containers for controls. A multiple-document interface (MDI) form can also act as a container for child forms and some controls.

form module

A file in a Visual Basic project with an .frm file name extension that can contain graphical descriptions of a form; its controls and their property settings; form-level declarations of constants, variables, and external procedures; and event and general procedures.

Function procedure

A procedure that performs a specific task within a program and returns a value. A **Function** procedure begins with a **Function** statement and ends with an **End Function** statement.

general procedure

A procedure that must be explicitly called by another procedure. In contrast, an event procedure is invoked automatically in response to a user or system action.

graphics method

A method that operates on an object such as a **Form**, **PictureBox**, or **Printer**, and performs run-time drawing operations such as animation or simulation. The graphics methods are **Circle**, **Cls**, **Line**, **PaintPicture**, **Point**, **Print**, and **PSet**.

host application

Any application that supports the use of Visual Basic for Applications, for example, Microsoft Excel, Microsoft Project, and so on.

identifier

An element of an expression that refers to a constant or variable.

icon

A graphical representation of an object or concept; commonly used to represent minimized applications in Microsoft Windows. An icon is a bitmap with a maximum size of 32 x 32 pixels. Icons have an .ico file name extension.

in process

Running in the same address space as an application.

insertable object

An application object that is a type of custom control, such as a Microsoft Excel worksheet.

Integer data type

A data type that holds integer variables stored as 2-byte whole numbers in the range -32,768 to 32,767. The **Integer** data type is also used to represent enumerated values. The percent sign (**%**) type-declaration character represents an **Integer** in Visual Basic.

intrinsic constants

A constant provided by an application. Visual Basic constants are listed in the object library and can be viewed with the **Object Browser**. Because you can't disable intrinsic constants, you can't create a user-defined constant with the same name.

keyword

A word or symbol recognized as part of the Visual Basic programming language; for example, a statement, function name, or operator.

line-continuation character

The combination of a space followed by an underscore (_) used in the development environment to extend a single logical line of code to two or more physical lines. However, you can't use a line-continuation character to continue a line of code within a string expression.

line label

Used to identify a single line of code. A line label can be any combination of characters that starts with a letter and ends with a colon (**:**). Line labels are not case sensitive and must begin in the first column.

line number

Used to identify a single line of code. A line number can be any combination of digits that is unique within the module where it is used. Line numbers must begin in the first column.

linked window

A window that is joined to another window other than the main window.

linked window frame

A window frame containing multiple windows that have been linked together.

locale

The set of information that corresponds to a given language and country. The code locale setting affects the language of terms such as keywords and defines locale-specific settings such as the decimal and list separators, date formats, and character sorting order.

The system locale setting affects the way locale-aware functionality behaves, for example, when you display numbers or convert strings to dates. You set the system locale using the **Control Panel** utilities provided by the operating system.

Although the code locale and system locale are generally set to the same setting, they may differ in some situations. For example, in Visual Basic, Standard Edition and Visual Basic, Professional Edition, the code is not translated from English-U.S. The system locale can be set to the user's language and country, but the code locale is always set to English-U.S. and can't be changed. In this case, the English-U.S. separators, format placeholders, and sorting order are used.

logic error

A programming error that can cause code to produce incorrect results or stop execution. For example, a logic error can be caused by incorrect variable names, incorrect variable types, endless loops, flaws in comparisons, or array problems.

Long data type

A 4-byte integer ranging in value from -2,147,483,648 to 2,147,483,647. The ampersand (**&**) type-declaration character represents a **Long** in Visual Basic.

margin indicator

An icon displayed in the **Margin Indicator** bar in the **Code** window. Margin indicators provide visual cues during code editing.

MDI child

A form contained within an MDI form in a multiple-document interface (MDI) application. To create a child form, set the **MDIChild** property of the MDI form to **True**.

MDI form

A window that makes up the background of a multiple-document interface (MDI) application. The MDI form is the container for any MDI child forms in the application.

member

An element of a collection, object, or user-defined type.

metafile

A file that stores an image as graphical objects such as lines, circles, and polygons rather than as pixels. There are two types of metafiles, standard and enhanced. Standard metafiles usually have a .wmf file name extension. Enhanced metafiles usually have a .emf file name extension. Metafiles preserve an image more accurately than pixels when the image is resized.

method

A procedure that acts on an object.

module

A set of declarations followed by procedures.

module level

Describes code in the Declarations section of a module. Any code outside a procedure is referred to as module-level code. Declarations must be listed first, followed by procedures.

module variable

A variable declared outside **Function**, **Sub**, or **Property** procedure code. Module variables must be declared outside any procedures in the module. They exist while the module is loaded and are visible in all procedures in the module.

named argument

An argument that has a name that is predefined in the object library. Instead of providing a value for each argument in a specified order expected by the syntax, you can use named arguments to assign values in any order. For example, suppose a method accepts three arguments:

DoSomeThing *namedarg1, namedarg2, namedarg3*

By assigning values to named arguments, you can use the following statement:

```
DoSomeThing namedarg3 := 4, namedarg2 := 5, namedarg1 := 20
```

Note that the named arguments don't have to appear in the normal positional order in the syntax.

Null

A value indicating that a variable contains no valid data. **Null** is the result of an explicit assignment of **Null** to a variable or any operation between expressions that contain **Null**.

numeric data type

Any intrinsic numeric data type (**Byte**, **Boolean**, **Integer**, **Long**, **Currency**, **Single**, **Double**, or **Date**).

numeric expression

Any expression that can be evaluated as a number. Elements of an expression can include any combination of keywords, variables, constants, and operators that result in a number.

numeric type

Any intrinsic numeric data type (**Byte**, **Boolean**, **Integer**, **Long**, **Currency**, **Single**, **Double**, or **Date**) or any **Variant** numeric subtype (**Empty**, **Integer**, **Long**, **Single**, **Double**, **Currency**, **Decimal**, **Date**, **Error**, **Boolean**, or **Byte**).

object

A combination of code and data that can be treated as a unit, for example, a control, form, or application component. Each object is defined by a class.

Object box

A list box at the upper-left corner of the **Code** window that lists the form and controls in the form to which the code is attached, or a list box located at the top of the **Properties** window that lists the form and its controls.

Object Browser

A dialog box in which you can examine the contents of an object library to get information about the objects provided.

Object data type

A data type that represents any **Object** reference. **Object** variables are stored as 32-bit (4-byte) addresses that refer to objects.

object expression

An expression that specifies a particular object and can include any of the object's containers. For example, an application can have an **Application** object that contains a **Document** object that contains a **Text** object.

object library

A file with the .olb extension that provides information to Automation controllers (like Visual Basic) about available objects. You can use the **Object Browser** to examine the contents of an object library to get information about the objects provided.

object module

A module that contains code specific to an object, for example, class module, form module, and document module. Object modules contain the code behind their associated objects. The rules for object modules differ from those for standard modules.

object type

A type of object exposed by an application through Automation, for example, **Application**, **File**, **Range**, and **Sheet**. Use the **Object Browser** or refer to the application's documentation for a complete listing of available objects.

object variable

A variable that contains a reference to an object.

parameter

Variable name by which an argument passed to a procedure is known within the procedure. This variable receives the argument passed into the procedure. Its scope ends when the procedure ends.

path

A string expression specifying a directory or folder location. The location can include a drive specification.

pi

A mathematical constant equal to approximately 3.1415926535897932.

point

A point is 1/72 inch. Font sizes are usually measured in points.

print zone

Print zones begin every 14 columns. The width of each column is an average of the width of all characters in the point size for the selected font.

Private

Variables that are visible only to the module in which they are declared.

procedure

A named sequence of statements executed as a unit. For example, **Function**, **Property**, and **Sub** are types of procedures. A procedure name is always defined at module level. All executable code must be contained in a procedure. Procedures can't be nested within other procedures.

Procedure box

A list box at the upper-right corner of the **Code** window and the **Debug** window that displays the procedures recognized for the object displayed in the **Object** box.

procedure call

A statement in code that tells Visual Basic to execute a procedure.

procedure level

Describes statements located within a **Function**, **Property**, or **Sub** procedure. Declarations are usually listed first, followed by assignments and other executable code.

Note that module-level code resides outside a procedure block.

project

A set of modules.

Project window

A window that displays a list of the form, class, and standard modules; the resource file; and references in your project. Files with .ocx and .vbx file name extensions aren't displayed in the **Project** window.

Properties window

A window used to display or change properties of a selected form or control at design time. Some custom controls have customized **Properties** windows.

property

A named attribute of an object. Properties define object characteristics such as size, color, and screen location, or the state of an object, such as enabled or disabled.

Property procedure

A procedure that creates and manipulates properties for a class module. A **Property** procedure begins with a **Property Let**, **Property Get**, or **Property Set** statement and ends with an **End Property** statement.

Public

Variables declared using the **Public** statement are visible to all procedures in all modules in all applications unless **Option Private Module** is in effect. In that case, the variables are public only within the project in which they reside.

referenced project

The project you directly create a link to from the current project you are working on. A project referenced by one of the current project's directly referenced projects is called an indirectly referenced project. Its **Public** variables are not accessible to the current project except through qualification with its project name. Any combination of direct and indirect references between projects is valid as long as they do not result in a complete cycle.

referencing project

The current project. How you create a link to a project depends on the host application. For example, to directly reference a project in Microsoft Excel, select the project from the **References** dialog box of the **Tools** menu. **Public** variables in a directly referenced project are visible to the directly referencing project, but **Public** variables in a directly referencing project are not visible to a directly referenced project.

registry

In Microsoft Windows version 3.1, OLE registration information and file associations are stored in the registration database, and program settings are stored in Windows system initialization (.ini) files. In Microsoft Windows 95, the Windows registry serves as a central configuration database for user, application, and computer-specific information, including the information previously contained in both the Windows version 3.1 registration database and .ini files.

resource file

A file in a Visual Basic project with an .res file name extension that can contain bitmaps, text strings, or other data. By storing this data in a separate file, you can change the information without editing your code. Only one resource file can be associated with a project.

run time

The time during which code is running. During run time, you can't edit the code.

run-time error

An error that occurs when code is running. A run-time error results when a statement attempts an invalid operation.

scope

Defines the visibility of a variable, procedure, or object. For example, a variable declared as **Public** is visible to all procedures in all modules in a directly referencing project unless **Option Private Module** is in effect. When **Option Private Module** is in effect, the module itself is private and therefore not visible to referencing projects. Variables declared in a procedure are visible only within the procedure and lose their value between calls unless they are declared **Static**.

seed

An initial value used to generate pseudorandom numbers. For example, the **Randomize** statement creates a seed number used by the **Rnd** function to create unique pseudorandom number sequences.

Single data type

A data type that stores single-precision floating-point variables as 32-bit (2-byte) floating-point numbers, ranging in value from -3.402823E38 to -1.401298E-45 for negative values, and 1.401298E-45 to 3.402823E38 for positive values. The exclamation point (**!**) type-declaration character represents a **Single** in Visual Basic.

sort order

A sequencing principle used to order data, for example, alphabetic, numeric, ascending, descending, and so on.

stack

A fixed amount of memory used by Visual Basic to preserve local variables and arguments during procedure calls.

standard module

A module containing only procedure, type, and data declarations and definitions. Module-level declarations and definitions in a standard module are **Public** by default. A standard module is referred to as a code module in earlier versions of Visual Basic.

statement

A syntactically complete unit that expresses one kind of action, declaration, or definition. A statement generally occupies a single line, although you can use a colon (**:**) to include more than one statement on a line. You can also use a line-continuation character (**_**) to continue a single logical line onto a second physical line.

string comparison

A comparison of two sequences of characters. Use **Option Compare** to specify binary or text comparison. In English-U.S., binary comparisons are case sensitive; text comparisons are not.

string constant

Any constant (defined using the **Const** keyword) consisting of a sequence of contiguous characters interpreted as the characters themselves rather than as a numeric value.

String data type

A data type consisting of a sequence of contiguous characters that represent the characters themselves rather than their numeric values. A **String** can include letters, numbers, spaces, and punctuation. The **String** data type can store fixed-length strings ranging in length from 0 to approximately 63K characters and dynamic strings ranging in length from 0 to approximately 2 billion characters. The dollar sign (**$**) type-declaration character represents a **String** in Visual Basic.

string expression

Any expression that evaluates to a sequence of contiguous characters. Elements of a string expression can include a function that returns a string, a string literal, a string constant, a string variable, a string **Variant**, or a function that returns a string **Variant** (**VarType** 8).

string literal

Any expression consisting of a sequence of contiguous characters surrounded by quotation marks that is literally interpreted as the characters within the quotation marks.

Sub procedure

A procedure that performs a specific task within a program, but returns no explicit value. A **Sub** procedure begins with a **Sub** statement and ends with an **End Sub** statement.

syntax checking

A feature that checks your code for correct syntax. If the syntax checking feature is enabled, a message is displayed when you enter code that contains a syntax error and the suspect code is highlighted.

syntax error

An error that occurs when you enter a line of code that Visual Basic doesn't recognize.

Note that syntax rules for individual keywords are defined in the Syntax section of the associated Help topic. To get Help on a keyword from within the development environment, select the keyword and press F1.

tab order

The order in which the focus moves from one field to the next as you press TAB or SHIFT+TAB.

time expression

Any expression that can be interpreted as a time. This includes any combination of time literals, numbers that look like times, strings that look like times, and times returned from functions.

Times are stored as part of a real number. Values to the right of the decimal represent the time. For example, midday (12:00 P.M.) is represented by 0.5.

twip

A unit of screen measurement equal to 1/20 point. A twip is a screen-independent unit used to ensure that placement and proportion of screen elements in your screen application are the same on all display systems. There are approximately 1440 twips to a logical inch or 567 twips to a logical centimeter (the length of a screen item measuring one inch or one centimeter when printed).

type-declaration character

A character appended to a variable name indicating the variable's data type. By default, variables are of type **Variant** unless a corresponding **Def***type* statement is present in the module.

type library

A file or component within another file that contains standard descriptions of exposed objects, properties, and methods that are available for Automation. Object library files (.olb) contain type libraries.

Unicode

International Standards Organization (ISO) character standard. Unicode uses a 16-bit (2-byte) coding scheme that allows for 65,536 distinct character spaces. Unicode includes representations for punctuation marks, mathematical symbols, and dingbats, with substantial room for future expansion.

universal date format

The universal date format is `#yyyy-mm-dd hh:mm:ss#`. However, both the date component (`#yyyy-mm-dd#`) and the time component (`#hh:mm:ss#`) can be represented separately.

user-defined type

Any data type defined using the **Type** statement. User-defined data types can contain one or more elements of any data type. Arrays of user-defined and other data types are created using the **Dim** statement. Arrays of any type can be included within user-defined types.

variable

A named storage location that can contain data that can be modified during program execution. Each variable has a name that uniquely identifies it within its scope. A data type can be specified or not.

Variable names must begin with an alphabetic character, must be unique within the same scope, can't be longer than 255 characters, and can't contain an embedded period or type-declaration character.

Variant data type

A special data type that can contain numeric, string, or date data as well as the special values **Empty** and **Null**. The **Variant** data type has a numeric storage size of 16 bytes and can contain data up to the range of a **Decimal**, or a character storage size of 22 bytes (plus string length), and can store any character text. The **VarType** function defines how the data in a **Variant** is treated. All variables become **Variant** data types if not explicitly declared as some other data type.

variant expression

Any expression that can evaluate to numeric, string, or date data, as well as the special values **Empty** and **Null**.

watch expression

A user-defined expression that enables you to observe the behavior of a variable or expression. Watch expressions appear in the **Watch** window of the **Visual Basic Editor** and are automatically updated when you enter break mode. The **Watch** window displays the value of an expression within a given context. Watch expressions are not saved with your code.

z-order

The visual layering of controls on a form along the form's z-axis (depth). The z-order determines which controls are in front of other controls.