Fundamental Statements in QBasic

QBasic Statement Components

A ***program*** is a sequence of instructions that tells the computer how to solve a problem. All QBasic ***statements*** are composed of programming commands, called ***keywords*** (which have special meanings to QBasic) and other language elements.

Constants

Constants are values that do not change during the execution of a program. There are two kinds: numeric constants and character string constants.

Numeric Constants

A number included in a statement is called a ***numeric constant***. Numeric constants can be **real numbers** which include decimal points, or **integers**, which are numbers without decimal points.

When using numbers, remember these rules:

- No commas can be included in numbers when entering them into the computer. The computer interprets the digits before and after a comma as two separate numbers.
- If a number has no sign, the computer assumes it is positive.
- If a number is negative, the negative sign must precede the digits.
- Fractions must be written in decimal form.
- QBasic uses exponential notation (or scientific notation) to represent very large or very small numbers. (**Note**: we do not input numbers into our program code using exponential/scientific notation).

The general format used in exponential/scientific notation is `±x.xxxD±n`

- The symbol ± represents the sign of the number, positive or negative. The plus sign is optional with positive numbers, but the minus sign is mandatory for negative numbers.
- `x.xxxx` is the *mantissa* and represents the digits of the number.
- The letter `D` indicates this is a double-precision number.
- `±n` is the positive or negative exponential value indicating that the decimal point should be moved to the right (+) `n` number of places or to the left (−) `n` number of places.

Character String Constants

A ***character string constant*** (or ***string*** for short) is a collection of symbols called alphanumeric data. A string can include any combination of letters, numbers, and special characters including dashes, commas, blanks, and so forth. All character strings must be enclosed in double quotation marks. Note that we can include single quotation marks within a string that is enclosed within double quotation marks.

The length of a string is determined by counting all its characters with the maximum length being 32,767 characters.

Variables

Before proceeding any further, it is important that we understand how data is stored in the main memory of the computer. The main memory is divided into many separate storage locations, each with a specific address. *A storage location containing a value that can change during execution is referred to as a* ***variable***. A variable can contain only one value at a time; when a new value is assigned to a variable, the old value is lost. These storage locations can be referred to by their address, just as post office boxes can be referred to by the numbers assigned to them. In machine language programming, a storage location is always referred to by its actual address but, it is difficult for programmers to keep track of these addresses. Fortunately, in QBasic (and other high-level languages) the programmer is allowed to use ***variable names*** for storage locations. Variable names are much easier to remember than addresses.

Variable names can have any number of characters; however, QBasic recognizes only the first 40 characters. This means that if the first 40 characters of two names are identical, QBasic sees the name as being identical, even if the 41st character is different. The first character of the name must be a letter. The remaining characters may be letters, numbers, and periods. QBasic does not differentiate between uppercase and lowercase letters in variable names; therefore, it sees these three names as being identical:

```
FirstDown

FIRSTDOWN

firstdown
```

Many programmers use a combination of uppercase and lower case letters to improve readability and we'll follow that practice this semester.

Good programming habits include the use of descriptive variable names, that is, names that describe the value they identify.

Numeric Variables

Numeric variables are used to store numbers that are either supplied to the computer by the programmer or internally calculated during program execution. A numeric variable name must being with a letter, followed by letters, digits, and/or periods. It cannot have embedded blanks.

The following are invalid:

`Maximum/Average`	(Contains an invalid character)
`1stChoice`	(Name must start with a letter)
`Square Yards`	(Blanks are not allowed within the name)

There are actually several types of numeric variables that we'll discuss.

String Variables

A ***string variable*** is used to store a character string, such as a name, an address, or a social security number. As with numeric variables, string variables can store only one value at a time. A string variable name begins with a letter followed by letters or digits.

Keywords

Certain words have specific meanings to QBasic. These are referred to as ***keywords*** (or ***reserved words***) and cannot be used as variable names. A list of the keywords in QBasic can be found by clicking the `Help` tab in the QBasic editor then clicking on the `Index` option.

Rules for Creating Variable Names

- Must begin with a letter.
- Can contain letters, numeric digits, and the period.
- Cannot contain blanks.
- Can have any number of characters; however, only the first 40 characters are significant.
- Cannot be a keyword.

Simple QBasic Statements

Clearing the Output Screen

The statement

`CLS`

removes an existing text from the Output screen and positions the cursor in the upper-left corner of the screen.

Documenting a Program

The **`REM`**, or remark statement, provides information for the human half of the programming team. It is ignored by the computer. Statements that are ignored by the computer are referred to as ***non-executable statements***. This information is referred to as ***documentation*** and its function is to explain the purpose of the program, what variable(s) are used in the program, and anything else of importance. Because documentation does not affect program execution, it can be placed anywhere in the program.

It is a common practice to indicate documentation by placing a single quotation mark (**`'`**) to the left of the comment rather than the **`REM`** keyword. This allows remarks to be placed following a program statement yet on the same line.

Well documented programs make it easier not only for the programmer but also for anyone else trying to understand a program. Documentation is particularly useful if a program needs to be changed in the future. It is important to develop the habit of documenting programs as they are written and revising the documentation as the programs are debugged or otherwise changed.

Assigning Values to Variables

The assignment statement stores a value in main memory in the location allotted to the stated variable. The assignment statement can be used to assign values directly to variables or to assign the result of a calculation to a variable. In either case, the expression to the right of the assignment operator (**=** the equal sign) is assigned to the variable on the left side.

It is possible for the same variable name to appear on both sides of an assignment operator.

```
DIM count AS INTEGER
count = 10
count = count + 1
```

The first statement allocates system memory for an integer and associates the name **`count`** with this memory address. The second statement stores the value **`10`** in **`count`** and the second statement increments **`count`** by **`1`**. Therefore, after the third statement is executed, **`count`** will be equal to **`11`**.

It is possible to assign a number to a string variable if the number is placed in quotation marks:

```
DIM zipCode AS STRING
zipCode = "35401"
```

Because this values is stored as a character string, it is stored exactly as it appears inside the quotation marks. In addition, it cannot be used in arithmetic operations.

Arithmetic Operators

The more commonly used arithmetic operators are:

Operator	Operation	Qbasic Expression
+	Addition	A + B
–	Subtraction	A – B
/	Division	A / B
*	Multiplication	A * B
^	Exponentiation	A ^ B
mod	Modulus – returns the remainder of A divided by B	A mod B

When more than one operation is performed in a single arithmetic expression, the computer follows a hierarchy of operations to determine the order in which the expressions are to be evaluated.

Priority	Operation	Symbol
First	Exponentiation	^
Second	Multiplication, division	*, /
Third	Addition, subtraction	+, –
Fourth	Any operation on the same level are performed left to right	

In the statement

```
x = 10 + 4 * 2
```

x is assigned the value of 18 because the multiplication is performed before the addition.

When parentheses are used in an express, the operations inside the parentheses are performed first. If parentheses are nested (placed one inside of another), the operations are performed from the inside out.

Displaying Results

The **PRINT** statement is used to display the results of computer processing. The output of a **PRINT** statement can be formatted in many different ways as we'll see later.

Displaying the Value of a Variable

We can tell the computer to output the value assigned to a storage location simply by using the keyword **PRINT** with the variable name after it. Printing has no effect on the contents of the storage location being printed.

Displaying Literals

A literal is a group of characters containing any combination of alphabetic, numeric, and/or special characters. It is essentially the same as a constant, but the term is applies to constants used in **PRINT** statements. There are two types: character string and numeric.

A ***character string literal*** is a group of letters, numbers, and/or special characters enclosed in quotation marks. Everything inside the quotation marks is displayed exactly as is. For example when the following statement is executed:

```
PRINT "Sample @%Output 12"
```

The following is output to the screen:

```
Sample @%Output 12
```

Note that the quotation marks are not displayed.

Numeric literals also can be placed in **PRINT** statements. They do not have to be enclosed in quotation marks.

```
PRINT 103
```

will output:

```
103
```

Displaying the Value of an Expression

Qbasic can print not only the literals and the values of variables, but also the values of arithmetic expressions such as:

```
DIM x AS INTEGER
DIM y AS INTEGER
X = 15
Y = 5
PRINT (x + y) / 2, x / y
END
```

The expression in the fifth line is evaluated and the result is displayed:

```
10          3
```

The **END** Statement

The **END** statement instructs the computer to stop program execution.

The Immediate Window

Programming statements are ordinarily executed in programming mode; execution does not begin until we instruct QBasic to start. However, we can use the **Immediate** window to execute statements as soon as the Enter key is pressed. This is useful when we want to test a statement quick or find a result.

As soon as we press the Enter key, the answer will appear in the Output screen, Press any key to return to the **Immediate** window. To move the cursor from the View window to the **Immediate** window, we press the **F6** key. This makes the next window the active window.