QBasic Input and Output

Data in Variables - INPUT

The **INPUT** statement allows the user to enter data at the keyboard while the program is executing.

FORMAT STATEMENT:

```
INPUT ["prompt";] variable1 [, variable2]...
```

INPUT

The statement must begin with the keyword **INPUT**. (Any uppercase words in a format statement must be placed in the statement exactly as they appear in the format description.)

["prompt";]

prompt is an optional literal string. (Anything placed in brackets **[ ]** is optional.) If a prompt is used, it is placed in quotes and may be followed by a semicolon or a comma. If the prompt is followed by a semicolon (**;**) a question mark is automatically appended to the end of the literal string. If the prompt is followed by a comma (**,**) nothing will be appended to the end of the literal string.

variable1

The name of at least one variable must come next. (**variable1** is used here to represent a variable name.)

[, variable2]...

Additional variables, preceded by a comma, are optional. The dots (**...**) indicate that as many variable names, as are needed, may be listed in this statement.

Formatting Your Results - PRINT

Semicolon

The semicolon is often used to separate two or more variables in a single **PRINT** statement. It instructs QBasic to print the next item starting at the next available print position.

```
PRINT "John"; "Drake"
JohnDrake

Print "John"; " Drake"
John Drake
```

When a numeric value is output with no (+/-) sign, a preceding space is printed. If the numeric value has a sign no preceding space is printed (because the sign is printed in that position). In either case, a space is always

added to the left of numeric values, for greater readability. Therefore, when numeric values are separated by a semicolon, the printed digits are not contiguous in the way they are in character strings.

```
PRINT 100; -200; 300
 100 -200  300
```

When a semicolon appears as the last character in a **PRINT** statement the next **PRINT** statement will print on the same line (in the next available print position). In other words, it prevents the output of the next **PRINT** statement from starting on a new line.

```
PRINT 3567;
PRINT "Neil";
PRINT " Patrick"
3567 Neil Patrick
```

Print Zones

Each line of the console window is divided into sections called print zones. Each print zone is 14 characters wide. There are five print zones per line. The beginning column, for each print zone, are as follows:

Zone 1	Zone 2	Zone 3	Zone 4	Zone 5
Column 1	Column 15	Column 29	Column 43	Column 57

Commas, like semicolons, can be used within a **PRINT** statement to control the format of your output. A comma indicates that the next item to be printed will start at the beginning of the next available print zone.

```
PRINT "SEE", "YOU", "LATER"
SEE            YOU            LATER
```

Tab Function

The **TAB** function allows output to be printed in any available column in an output line, thus giving greater flexibility in formatting. As with the comma and semicolon, one or more **TAB** functions can be used in a **PRINT** statement.

```
PRINT TAB(10); "Hi there!"; TAB(25); "Bye!"
         Hi there!       Bye!
```

SPC Function

The **SPC** (an abbreviation for SPACE) function is similar to the **TAB** in that it is used to control output. However, instead of causing output to begin in a specified column, it determines how many spaces the output will be moved to the right of the preceding printed data before printing begins.

```
PRINT "WORD"; SPC(10); "LETTER"
WORD          LETTER
```

When the statement **PRINT "WORD"; SPC(10); "LETTER"** is executed, WORD appears in print positions 1 through 4. Because of the **SPC(10)** function the computer leaves ten black spaces between the end of WORD and the beginning of LETTER. Therefore, LETTER is printed starting in column 15.

LOCATE Statement

The **LOCATE** statement is used to position the cursor at a specified row and column on the screen. The row value must be from 1 through 25 and the column value from 1 through 80.

FORMAT STATEMENT:
LOCATE row, column

```
LOCATE 10, 65
PRINT "QBasic"
```

The first statement **(LOCATE 10, 65)** positions the cursor at row 10 and column 65. The second statement **(PRINT "QBasic")** outputs the literal string **QBasic** to the console starting in row 10 column 65.

The PRINT USING Statement

Like any **PRINT** statement the **PRINT USING** statement displays the values of expressions, such as variable and/or constants. However, the **PRINT USING** statement tells QBasic to display these values using a specified format. The following special ***control characters*** are used to indicate the format:

#	Hash sign: one symbol is used for each digit to be printed; blanks are added to the left of the number to fill the field.
$	Dollar sign; printed exactly as is.
$$	Two Dollar signs: cause the dollar sign to be printed immediately before first digit.
******	Leading asterisks; prints fill-in asterisks in place of blanks.
.	Decimal point; printed exactly as is.
,	Places a comma in front of each group of three digits to the left of the decimal point.
-	Placed at the end of a formatting field, to print a trailing minus sign for negative numbers only.
**\ **	Reserves n + 2 spaces for a character string where n Is the number of spaces between the back slashes.
&	Causes the entire string to be printed, regardless of its length.

```
num1 = 4.562
PRINT USING "###.##"; num1

num2 = 78.907
PRINT USING "###.##"; num2

num3 = 0.03
PRINT USING "###.##"; num3

num4 = 1493
PRINT USING "###.##"; num4
```

```
   4.56
  78.91
   0.03
1493.00
```

In this output the 2 at the end of 4.652 was truncated.
In this output the 7 at the end of 78.907 was rounded up.
In this output the first two positions were not used.
In this output two trailing zeros were added.

```
PRINT USING "$####.##; num1
$   4.56

PRINT USING "$$####.##"; num1
$4.56
```

A second method of using the **PRINT USING** statement is to assign the format control characters to a string variable. This variable name can then be referred to in the **PRINT USING** statement.

```
DIM format1 AS STRING
format1 = "####.###"

num1 = 4.562
PRINT USING format1; num1

num2 = 78.907
PRINT USING format1; num2

num3 = 0.03
PRINT USING format1; num3

num4 = 1493
PRINT USING format1; num4
```

This will produce the same output as the preceding method.

Another useful control character is the backslash (\), which is used to format character strings. The string will be left-justified in the output field.

```
DIM item1 AS STRING
DIM cost1 AS SINGLE
item1 = "Bicycle"
cost1 = 149.50
PRINT USING "\    \ $$###.##"; item1; cost1
Bicycl    $149.50
```

Here you'll notice that the last character in the literal string "Bicycle" (the e) was truncated. This is because we entered four spaces between the backslashes, making the maximum size of the field 6 (the number of spaces between the backslashes + 2). If our string has more than 6 characters (as Bicycle does) the output is truncated (cut off) after the sixth character. If our string is shorter than six characters, the string is left-justified in the field (that is, the first character of the string is output at the left margin) and any blank spaces are placed at the end of the string.

If our statement were rewritten with 10 spaces between the backslashes (allowing for a 12 characters string; 10 + 2 = 12) then there would have five blank spaces after Bicycle, because it is five spaces shorter than the maximum spaces allowed (12 – 7 = 5).

```
DIM item1 AS STRING
DIM cost1 AS SINGLE
item1 = "Bicycle"
cost1 – 149.50

PRINT USING "\          \" $$###.##"; item1, cost1


Bicycle     $149.50
```

QBasic's IDE - Edit Menu

The **Edit** menu allows portions of a program to be moved from one place to another. It also allows you to delete or copy program segments. You can access the **Edit** menu by using the mouse to position the cursor on the menu and clicking. You can then click on the option you want. Alternatively, the menu can be opened by pressing **Alt + E** on the keyboard. The options then appear. Note: the menu lists shortcut keys for each command, they are not the same as standard shortcut keys.

Selecting a Program Segment

Before a program segment can be cut, copied, or deleted, it must be selected. This process tells QBasic exactly what portion of the program you want to manipulate. One selection method is to position the cursor at the beginning of the segment, hold down the keyboard **Shift** key while pressing the **right arrow key** until the entire section is highlighted. If you need to select a large segment, you can use the **down arrow key** to select entire lines at a time.

Alternatively, the mouse can be used to select a segment. Position the mouse pointer at the start of the segment and hold down the left mouse button while moving the mouse to the end of the segment. Release the mouse button and the segment will be highlighted.

Cut

The **Cut** option deletes the selected portion of a program and places it on the Clipboard. After you have selected the segment to be cut, choose **Cut** from the **Edit** menu. The selected segment is deleted from the program and stored on the Clipboard.

Copy

The **Copy** option works in the same manner as **Cut**, except the selected portion is not deleted. It remains in the program but is also on the Clipboard. **Copy** is useful when you want the same program segment to appear in two different places.

Paste

To insert the contents of the Clipboard at a new location, position the cursor where you want it and choose **Paste** from the **Edit** menu. Program segments can quickly and conveniently be moved to new locations using this method.

Clear

The **Clear** option is similar to **Cut** except that the selected portion is not placed on the Clipboard; it is deleted from the computer's memory and cannot be retrieved. Therefore, be very careful when using **Clear**. Note: after text is selected, the **DEL** key on the keyboard can also be used to erase it.