## Parts of a Program

There are literally thousands of high level programming languages.  All have a similar function in that they facilitate the i*nput, process, and output* of data by a computer.  Input refers to a method of entering data into the computer.  This can be done though the keyboard or read from a storage device such as a disk drive.  Process refers to what is done to the data.  For example a program may be created to calculate an employee's pay cheque when the hours worked and pay rate are inputted.  Output, of course, refers to the display and or storage of the result of the process.  For most languages the basic components of the program are also similar.  Only the syntax, or how the language is formally written, is different.

#### Statement

A statement instructs a computer to do something.  It is one step in the sequence of steps that make a complete program.  Each of the following is an example of a single statement:

* Print "Hello"
* Let A = B + C
* Input "Enter the Tax"; T
* END

#### Mathematical Operators

The first computers were used as calculators.  They were a little different than the handheld calculators of today as they filled entire rooms and consumed huge amounts of electricity.  All languages support the basic mathematical operations:

* +  addition
* -   subtraction
* ×  multiplication
* /   division
* ^  exponentiation
* \   integer division
* Mod  modulus

#### Variables

A variable is a symbol that represents a storage place for some kind of data (a number, a name, etc.) that can change as a program is executed.  Similarly you use variables in mathematics to represent and manipulate numbers.  In a program you do the same thing but there are some important differences.  Giving a variable a value is called variable assignment.  For example:

**Let X = 6**

**Let Name$ = "Jon Doe"**

In the first example the value 5 is assigned to the variable X.  In the second, the character string Jon Doe is assigned to the variable Name$.  You need to use the term "assigned to" instead of equals.  The next example will show why.  Consider the following:

**Let X = X + 1**

At first glance this would seem to be inconsistent.  How can X be equal to X + 1? However if you read it (from right to left) it will make more sense:

Assign to X a new value which is the old value of X plus 1.

In fact this statement is very common in programming and is known as incrementing a variable.  The word incrementing simply means increasing the value by a constant amount—in this case by 1.  You could also decrement, for example

**Let X = X - 1**

#### Variable Types

To a computer everything is a number.  A screen image is just a large group of numbers that assign colors to various parts of the screen.  Your homework assignment is a set of numbers that represent the text you typed and stored on disk.  Every programming language has several types of variables that can represent the various types of numbers and characters.  Some of these include:

* **Integer**. Whole signed numbers
* **Currency**. Decimal numbers eg. dollars and cents
* **Date**. Usually an 8 character variable representing day, month, year
* **String**. Letters, digits, and special characters
* **Double**. Large numbers with 14 digits of accuracy
* **Boolean**. Variables that have only 2 values (true or false)

A variable can be any of the above and is **declared** as a certain variable type in the program (Note Dim is short for Dimension).

**Dim flag as Boolean**or  
**Dim X as Integer**

#### Boolean Operators

George Boole, a nineteenth century English Mathematician, invented a form of logic where there are only 2 outcomes, true or false.  Since the next lesson deals entirely with Boolean logic only a small introduction is given.  Boolean (also called logical) operators are used in high level languages to evaluate the outcome (either true or false) of a given statement.  The Boolean operators are AND, OR, and NOT.  Consider the following examples:

**I am 16 years old AND I am a female**

**I am 16 years old OR I am a female**

In the first statement the whole statement is true only if both statements are true.  In the second, only one of the statements needs to be true for the whole statement to be true.  Again, more on this in the next lesson.

#### Relational Operators

Also called comparison operators, relational operators are used to compare statements.  The relation operators are familiar as you have probably used some or all in your study of mathematics.  The relational operators are:

* =   equals
* >   greater than
* <   less than
* ≥   equal to or greater than
* ≤   less than or equal to
* ≠   not equal to

#### Conditional Statements

Conditional statements make use of the relational and Boolean operators to give the computer the ability to make decisions.  The most common usage in a high level language is the IF...THEN statement.

**If I have  > $10 Then I can go to the movie**

In this example If I have > $10 is called the condition.  If the condition is true then the consequence (going to the movie) will happen.  Boolean operators add to the power of this structure

**If I have > $10 AND I can get a ride Then I can go to the movie**

In most languages, the IF...THEN statement also includes a statement, ELSE,  that specifies an alternate consequence:

**If I have >$10 Then   
  I can go to the Movie  
Else  
  I can watch TV**

#### Looping Statements

Looping means repeating something. A looping statement is used to cause a computer to repeat the same block of code over and over.  This is commonly called looping.  There are several different type of loops or.

##### Definite Loop

The definite loop is one that repeats a given piece of code a definite or known number of times.  The most common form is the FOR....NEXT structure.

**For I = 1 To 5  
  Print "Hello"  
Next I**

In this example, the word Hello will be displayed on a computer monitors 5 times.

##### Indefinite Loop

An indefinite loop is one that causes a piece of code to be repeated indefinitely or endlessly.  Programmers try to avoid this kind of loop but you will probably code one as you learn a language.  The only way to end an indefinite loop is to "break" the execution of the program or turn off the computer!

**Do  
  Print "Hello"  
Loop**

In this example, the word Hello is repeated down the display screen indefinitely.

##### Conditional Loop

In the conditional loop, a piece of code is repeated until a certain condition is met.  The relational and Boolean operators can be used in the conditional loop.

**Do  
  Print "Get out of bed"  
Loop until Time = 11:00AM**

or

**Do While C < 10  
  C = C - 1  
  Print C  
Loop**

There are many more code structures in high level languages, for example code that deals with character strings, higher mathematical functions, graphics, audio, and animation to name a few.  However the examples given above are common to all high level languages and form the basis for code that does everything from simple word processing to controlling the space shuttle as it orbits the earth.

#### Sub-routines and Procedures

A sub-routine or procedure is a block of code that has some specific function.  The code may be used more than once in a program and can even be exported and used in other programs.  A sub-routine or procedure is "called" from another part of a program when needed.  An example of a general sub-procedure follows:

**Public Sub TaxCalculation**

**Tax = SalePrice \* .15  
TotalPrice = SalePrice + Tax**

**End Sub**