Mini-Lecture No. 1

Introduction to Computer Programming and Problem Solving Course

As the name of this course indicates, you will be studying the fundamentals of programming and problem solving in a computer language. Although you will be working in the Visual Basic 2005 environment, what you learn here will be the basis for programming in other computer languages as well.

A program is a list of instructions to the computer, designed in order to achieve a certain result. The program may be written in any of several different languages, but its basic logic will be the same.

As you probably know, a computer will do only what you tell it to do, not what you *mean* to tell it to do. When writing programs, you will need to follow particular rules, such as properly spelling key words. This falls under the category of s*yntax*. By definition, syntax refers to the programming rules of the language used to create a computer program. Since in this course you will do your programming in Visual Basic, you will be learning the syntax of that language. A program that is syntactically correct, however, is not necessarily logically correct.

A familiar example of the need to use logic is in baking a cake. Would you throw some flour and water into a baking pan, cook these ingredients, and then add the egg yolks? Not for a cake that you would actually eat! Likewise, unless you follow the proper logic when writing a program, you cannot produce a program that will be of use to you or others.

We will be exploring the proper logical order to be followed in writing computer programs prior to working with VB itself. In your future career as a programmer, you will find that the logic for writing the program will be consistent, regardless of the language.

Common Features of Computer Programs

Most programs carry out three functions: they take data in (input); they manipulate (process) that data; and, they produce the desired information (output). If you think about it, when you use many simple machines, you follow the same procedure. Close your eyes and imagine using a food processor or a blender. First, you will put the item or items to be processed into the machine (input), press the proper buttons to manipulate the contents (process) and, finally, empty (output) the finished product. When you write a program, you must follow these same steps; that is, you input the data provided to you, have the computer process that data and, then, output (display and/or print) the results of the processing.

Analyzing the Problem

Only when you know where you are going on a trip can you plan how to get there. Before writing the actual code, programmers need to know: *what is this program supposed to do* *(output)?* Once that information is ascertained, the programmer can begin formulating strategies for writing a program that will get him or her there.

Suppose you are asked to write a program that adds two numbers and, then, produces the result.

- We learned above that before trying to write any program, we must ask ourselves: *what is this program supposed to produce (output)?* Since the answer in addition is called a *sum*, we know that the program should output the sum of two numbers.
- Therefore, we will have to enter the two numbers into the computer (input).
- Once the numbers have been entered, the computer will be able to do the computation for us (process), following our specific instructions to add them together.

Programming Tools

Programmers have many tools at their disposal that aid them in planning their programs in advance of writing any instructions. Some of these tools include pseudocode, HIPO charts, and flowcharts.

Pseudocode uses English-like phrases together with some terms from the specific language you are using to outline the task; for example, for the number program you might jot down:

```
Get 2 numbers (input)
Add number 1 to number 2 (process)
Display the answer (output)
```

A **HIPO (Hierarchy plus Input, Process, Output) chart** provides a visual table displaying the modules of the program in a hierarchy quite similar to an organization chart, followed by a diagram that lists all input, all processes, and all output.

A **flowchart** is a graphical representation of a programmer's logic. Flowcharts are preferred by the majority of programmers. They provide students with a means of constructing a clear visual picture of what will be involved in writing the program before writing any of the actual instructions (coding).

Taking the Programming Journey

With all of this in mind, we will perform the following steps throughout this course:

1. analyze the problem;
2. draw a flowchart for the solution;
3. write the program.