Steps Required to Write a Program

1. Analyze the Problem

Make sure that you thoroughly understand the problem to be solved, and that you know what information will be needed by the program (program input), and what information will be produced by the program (program output). You may also wish to choose names for variables that will be needed in the program. Good variable names make it easier to think about the problem and also make step 2 easier.

2. Devise an Algorithm

An algorithm is a series of instructions written with enough detail so they can easily be translated into a computer language. It is rare that programmers directly write programs using a language such as Pascal or C++. Instead, they first devise an algorithm using some more informal means of expression, usually called *pseudocode*. If you are not familiar with pseudocode, you should ask your instructor for a copy of the handout, *An Informal Language for Writing Algorithms*. It is important to realize that constructing good algorithms is the key to successful programming and for many persons is the most difficult part of the programming process.

Once you have written an algorithm, be sure to test the design of the algorithm. Do not assume that your algorithm is correct! If you skip this step and go blindly forward in the programming process, you will end up translating an incorrect algorithm into an incorrect program! Check the logic of your algorithm by "feeding" it test data and following instructions as though you were a computer. If necessary, revise the algorithm.

3. Translate Algorithm into a Programming Language

It is usually a routine task to translate an algorithm into a programming language. However, you will probably be more successful if you work in a quiet place and use scratch paper rather than translating while working on the computer.

After translating the algorithm, carefully hand trace through the program, again pretending that you are the computer. Do not be in too great a hurry to move to step 4 until this step has been finished.

4. Enter the Program

Using a text editor, carefully type in the program. Be sure to use good styling conventions, such as proper indentation. Enter comments at this time, while the code is fresh in your mind.

5. Test and Debug the Program

Using several sets of input data, determine whether the program is correct. Make corrections if bugs are found. Serious bugs may require using a debugger or inserting temporary output statements. Bugs that resist early detection, should be handled by getting a printout and taking it to a quiet place. There, you should carefully hand-trace the program until the problem is found.

6. Polish the Program

Review the program, and if necessary, choose better variable names, write more comments, and improve program output. **Reread the original program specifications to make sure your program does exactly what it was supposed to do.**