The Programming Process

**Concept**: The programming process consists of several steps, which include design,

creation, testing, and debugging activities.

Quite often when inexperienced students are given programming assignments, they have trouble getting started because they don’t know what to do first. If you find yourself in this dilemma, the following steps may help.

1. Clearly define and understand what the program is to do.
2. Visualize the program running on the computer.
3. Use design tools, like *pseudocode*, to create a model of the program.
4. Enter the code and compile it.
5. Correct any errors found during compilation. Repeat steps 4 and 5 as many times as necessary.
6. Run the program with test data for input.
7. Correct any runtime errors found. Repeat steps 4 through 7 as many times as necessary.
8. Validate the results of the program.

These steps emphasize the importance of planning. Just as there are good ways and bad ways to paint a house, there are good ways and bad ways to create a program. A good program always begins with planning. With a program that will calculate the gross pay of a user serving as our example, let’s look at each of the steps in more detail.

1. **Clearly define and understand what the program is to do.**

This step commonly requires you to identify the purpose of the program, the data that is to be input, the processing that is to take place, and the desired output. Let’s examine each of these requirements for the calculate gross program.

*Purpose:* To calculate the user’s gross pay.

*Input:* Number of hours worked, hourly pay rate.

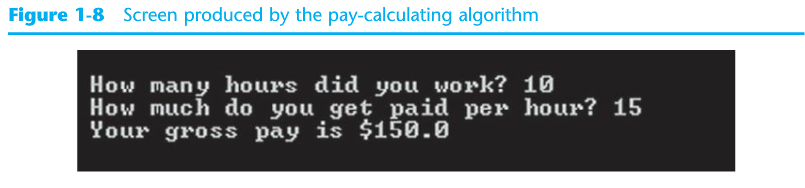
*Process:* Multiply number of hours worked by hourly pay rate. The result is the

user’s gross pay.

*Output:* Display a message indicating the user’s gross pay.

1. **Visualize the program running on the computer.**

Before you start a program, you should first create it in your mind. Try to imagine what the computer screen will look like while the program is running. For instance, Figure 1-8 shows the screen we might want produced by a program that implements the gross pay algorithm.



In this step, you must put yourself in the shoes of the user. What messages should the program display? What questions should it ask? By addressing these concerns, you can determine most of the program’s output.

1. **Use design tools, like *pseudocode*, to create a model of the program.**

*Pseudocode,* is a cross between human language and a programming language and is especially helpful when designing an algorithm. For example, here is pseudocode that describes the gross pay alogorithm:

*Get payroll data.*

*Calculate gross pay.*

*Display gross pay.*

Although this pseudocode gives a broad view of the program, it does not reveal all the program’s details. Here is a more detailed version of the pseudocode:

*Display “How many hours did you work?”*

*Input hours.*

*Display “How much do you get paid per hour?”*

*Input rate.*

*Store the value of hours times rate in the pay variable.*

*Display the value in the pay variable.*

1. **Enter the code and compile it.**

Once the model of the program has been created, the programmer is ready to write source code on the computer. The programmer saves the source code to a file and begins the process of compiling it. During this step the compiler will find any syntax errors that may exist in the program.

1. **Correct any errors found during compilation. Repeat Steps 4 and 5 as many times as necessary.**

If the compiler reports any errors, they must be corrected. Steps 4 and 5 must be repeated until the program is free of compile-time errors.

1. **Run the program with test data for input**

Once an executable file is generated, the program is ready to be tested for runtime errors. A runtime error is an error that occurs while the program is running. These are usually logical errors, such as mathematical mistakes, like dividing by zero.

Testing for runtime errors requires that the program be executed with sample data or sample input. The sample data should be such that the correct output can be predicted. If the program does not produce the correct output, a logical error is present in the program.

1. **Correct any runtime errors found. Repeat Steps 4 through 7 as many times as necessary.**

When runtime errors are found, they must be corrected. You must identify the step where the error occurred and determine the cause. If an error is a result of incorrect logic (such as improperly stated math formula), you must correct the statement or statements involved in the logic. The program must then be saved, recompiled, and retested. This means Steps 4 through 7 must be repeated until the program reliably produces satisfactory results.

1. **Validate the results of the program.**

When you believe you have corrected all the runtime errors, enter test data and determine if the program solves the original problem.