

Overtime Pay Program — Making Decisions, Simple I/O and Calculations

The next step in learning to program in C++ is to be able to handle making decisions. In this project, you will write a modification of the third project that will read simple payroll data from an input file, apply some simple rules to determine the correct pay amount for the employee, and print a summary to an output file.

This program will handle calculating gross pay, including overtime, for an hourly employee. The input file will be almost exactly the same as in project 3. However, this time you will not calculate any tax amounts. Instead, you will just calculate the employee's gross pay, using the following rules:

- For each of the first 40.0 hours worked, the employee is paid at his/her base hourly rate, which is given in the input file.
- For each of the next 8.0 hours worked, the employee is paid 1.5 times his/her base hourly rate. This is known as "time-and-a-half".
- For each additional hour worked, beyond the first 48.0 hours, the employee is paid 2.0 times his/her base hourly rate. This is known as "double-time".

For example, suppose that an employee's base hourly rate is \$10.00 and that she worked 50 hours in this pay period. Then her gross pay would be:

```
Gross pay = 40.0 * 10.0 // first 40 hours at base rate
           + 8.0 * 1.5 * 10.0 // next 8 hours at time-and-a-half
           + 2.0 * 2.0 * 10.0 // next 2 hours at double-time
```

Of course, an employee might work fewer hours, and receive only base pay and time-and-a-half, or even receive no overtime pay at all. That's why your program must make decisions.

The input file:

The input file for this program is named "PayData.txt". A sample input file is given below:

```
Current pay period data for employee:
432A.0037

Hours worked > 49.5
Hourly rate > 8.40
FIT rate > 15.0
SIT rate > 5.75
Health Ins > 7.50
```

For the program input, there is only one change from Project 3. The delimiting symbol used to separate the labels from the numerical data has been changed from a vertical bar to a '>'. All of the other specifications for the input that were given in Project 3 are still in effect.

What must be calculated:

Your program must calculate how many hours are to be paid at each of the three possible rates (base, time-and-a-half, double-time), how much pay is earned in each of those categories, and the total pay earned.

In order to produce the most accurate results possible in C++, all the decimal values must be stored using variables of type double, not type float.

The output file:

The output file is named "OTPay.txt". An output file, which corresponds to the given input file, is shown below:

```
Programmer: Bill McQuain
CS 1044 Project 4 Fall 2000

-----
Pay data for: 432A.0037

      Regular      Step 1      Step 2
Hours    40.00      8.00      1.50
Rate     8.40      12.60     16.80
Pay     336.00     100.80     25.20

Gross Pay: 462.00
-----
```

The first two lines specify the programmer and the assignment, and the third is blank. The fourth line just separates the header information from the actual pay stub information.

The pay summary lists the employee's ID "number", followed by a table showing the hours worked, pay rate, and pay earned for each rate category, formatted with labels as shown in the sample output file.

Finally, the summary lists the employee's total gross pay amount, labeled as shown.

If you have read the description of how the Curator scores your program in the *Student Guide*, you know that is important that you use the same spelling and capitalization for all the labels shown above. The horizontal spacing does not effect scoring unless you combine things that should be separate or separate things that should be combined. You must, however, use blank lines exactly as shown in the sample output.

Documentation and other requirements:

You must meet the following requirements (in addition to designing and implementing a program that merely produces correct output):

- Write a header comment with your identification information, the required pledge statement (below), and a brief description of what the program does.
- Write a comment explaining the purpose of every variable and named constant you use.
- Write comments describing what most of the statements in your program do.
- Use descriptive identifiers for variables and for constants.
- Use named constants instead of "magic numbers" whenever it is appropriate. Note: there are LOTS of candidates in this category.

Submitting your program:

You will submit this assignment to the Curator System (read the *Student Guide*), and it will be graded automatically. Instructions for submitting, and a description of how the grading is done, are contained in the *Student Guide*.

You will be allowed up to five submissions for this assignment. Use them wisely. Test your program thoroughly before submitting it. Make sure that your program produces correct results for every sample input file posted on the course website. If you do not get a perfect score, analyze the problem carefully and test your fix with the input file returned as part of the Curator e-mail message, before submitting again. The highest score you achieve will be counted.

The *Student Guide* can be found at: <http://ei.cs.vt.edu/~eags/Curator.html>

The submission client can be found at: <http://spasm.cs.vt.edu:8080/curator/>

Evaluation:

Your submitted program will be assigned a score based upon the runtime testing performed by the Curator System. We may very well evaluate your submission of this program for documentation style, and to see whether you followed the requirements given in this specification (such as the use of `double` variables). Therefore, you should compare your comments to those given in the programs for projects 1 and 2. The programs serve as useful a guide to acceptable documentation style at this point in the course.

If your program is evaluated for documentation and requirements, your instructor will specify how that score will be counted.

Pledge:

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the header comment for your program:

```
// On my honor:  
//  
// - I have not discussed the C++ language code in my program with  
// anyone other than my instructor or the teaching assistants  
// assigned to this course.  
//  
// - I have not used C++ language code obtained from another student,  
// or any other unauthorized source, either modified or unmodified.  
//  
// - If any C++ language code or documentation used in my program  
// was obtained from another source, such as a text book or course  
// notes, that has been clearly noted with a proper citation in  
// the comments of my program.  
//  
// - I have not designed this program in such a way as to defeat or  
// interfere with the normal operation of the Curator System.  
//  
// <Student Name>
```

Failure to include this pledge in a submission is a violation of the Honor Code.