

Pay Stub Program — Simple I/O and Calculations

Once you've learned how to perform numerical computations in C++, the next step is to learn to handle input of data and formatted output of computed results. In this project, you will write a short program that will read simple payroll data from an input file, do some calculations, and then print nicely formatted results to an output file.

This program will produce a simple payroll stub for an hourly employee. You will be given (see the input file description below) the number of hours the employee has worked in the current pay period and the hourly rate the employee is paid. You will also be given the tax rates for federal income tax (FIT) and state income tax (SIT) for this employee, and the amount the employee is charged, in each pay period, for his/her health insurance plan.

From this information you can calculate the employee's gross pay, which is just the amount the employee has earned before any tax and insurance withholding is subtracted. You can also calculate the amount of federal income tax and state income tax to withhold.

The employee must also pay two additional taxes, the Social Security Old Age tax (FICA) and the Medicare tax (FMED). The rate for the FICA tax is 6.20% and the rate for the FMED tax is 1.45%.

Given that information, you can also calculate the amount to withhold for FICA and FMED. Then, you can calculate the employee's net pay, that is, the amount of her/his gross pay that the employee actually receives after all the tax and insurance withholding.

Note: all tax rates are applied to the employee's gross pay amount.

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 | 37.5
Hourly rate  | 8.40
FIT rate     | 15.0
SIT rate     | 5.75
Health Ins   | 7.50
```

The input file begins with one header line, which is meaningful to a human reader but must be ignored by the program.

The second line of the input file contains the employee's ID "number" which is actually just a character string of arbitrary length. The third line of the input file is blank.

Each of the remaining lines begins with a text label (again, useful only to a human reader), followed by a numerical value. As shown in the sample input file, each of the text labels will be separated from the numerical value by the vertical bar character ('| ').

The data lines will always be given in the order shown in the sample input file, and you may assume that the values are logically correct (no negative values, for instance). **No input line will contain more than 255 characters.**

What must be calculated:

From the given data, you must calculate the four tax withholding amounts described above, and then calculate the employee's final net pay amount.

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

```
Programmer: Bill McQuain
CS 1044 Project 3 Fall 2000

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

Gross pay:    315.00

Tax deductions:  FIT      FMED      FICA      SIT
                  47.25    4.57     19.53     18.11

Other deductions: Health Ins
                   7.50

Net pay:      218.04

-----
```

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 stub gives the employee's ID "number", then the gross pay amount, then the tax deductions and the insurance deduction, and finally the net pay amount, formatted with labels as shown in the sample output file.

When a decimal number is written, the number of digits displayed after the decimal point is called the precision of the displayed value. All the dollar amounts must be written with a precision of 2 (i.e., dollars and cents) 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. In any case, the given code will exactly match the labels and spacing shown above.

Numerical note: most calculations involving decimal quantities cannot be performed exactly on a digital computer. Given the size of the numbers used in this program, and that you will use variables of type `double` in your program, we should all get the same dollar and cent amounts.

However, when you print these amounts with precision 2, the actual value stored in the computer's memory will be rounded off for printing. But, the net pay calculation will use the actual value stored in memory, not the rounded value. So, it is possible that the net pay amount may appear to be "off" by a cent or two. That shouldn't cause any problems in grading, but it illustrates the difficulty of producing completely accurate results even using variables of type `double`.

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.

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.