

Basic I/O and calculations:**Paystub Program**

It's finally time to write a complete program. This project will use many of the C++ features that were illustrated in the source code you were given for the programming assignment as well as use some input and output basics that were discussed in class. The resulting program will read simple data that is necessary to compute a paystub for an hourly employee. Your program will perform some simple calculations with this data and print a paystub to the output file.

You will be given (see the input description below) the number of hours the employee has worked in the current pay period and the hourly rate that this employee is paid. You will also be given the tax rate 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 deductions are 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, which is the amount of the gross pay that the employee actually receives after all of the taxes and insurance subtractions.

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

Sample input data:

Here is a sample input file, named `PayData.txt`, for the program

Employee Name:	Joe Hokie
Employee ID:	52314.fdsjl
Hours worked	37.5
Hourly rate	8.40
FIT Rate	15.0
SIT Rate	5.75
Health Ins	7.50

The first line contains the name of the employee. There will be a single tab between the text "Employee Name:" and the beginning of the employee's name. The second line contains the employee id. There will be a single tab between the text "Employee ID:" and the beginning of the id value. The id is just string data and will not contain any whitespace. The third line of the input file is blank.

Each of the remaining lines begins with a text description (useful only to the reader) followed by a vertical bar. After the vertical bar, there may be whitespace preceding the numerical value.

The data lines will always be given in the order shown in the sample input file. The input values are guaranteed to be syntactically and logically correct. No input line will contain more than 255 characters.

Calculations:

In order to produce the paystub, you must calculate the gross pay and the four tax withholding amounts described above, and then use this data to calculate the employee's net pay.

There is one important data representation issue in this project. As mentioned in class, binary representation of decimal real numbers introduces error when there is a fractional component. For this reason, monetary values (such as the Hourly Rate and the Health Insurance deduction) must be stored as integers, not doubles or floats. All other numerical inputs, such as the tax rates and the hours worked, should be stored as doubles. Any internal variable in your program that stores a monetary value must be an integer as well. Failure to do this may result in different results when the Curator grades your program. In addition, the GTA will deduct points on those programs that do not use this implementation strategy.

Sample output:

Here is a sample output file, named `PayStub.txt`, which corresponds to the input data given above:

```
Programmer: Rich Wheaton
CS 1044 Project 2 Summer II 2003

-----
Employee:   Joe Hokie
ID Number:  52314.fdsjl

Gross Pay:      315.00

Tax deductions:      FIT          FMED          FICA          SIT
                    47.25         4.56          19.53         18.11

Other deductions: Health Ins
                   7.50

Net Pay:         218.05
-----
```

The first two lines just identify the programmer and project. Next there is a blank line followed by a separator line. The next 2 lines are the employee name and the employee id respectively. The next 3 lines are a blank line, the gross pay, and another blank line. The next 2 lines contain a descriptive header line followed by a blank line. The next 2 lines contain the computed taxes followed by a blank line. The next 3 lines contain a descriptive header line followed by the health insurance amount, followed by yet another blank line. Finally, the last 2 lines contain the net pay and a separator line.

All monetary amounts are to be reported to two decimal places and all formatting should follow the sample output file above..

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 are free to experiment with the horizontal layout but you should try to align the columns neatly.

Suggested implementation plan:

You should design your solution piece by piece. Begin by identifying the major tasks that have to be done, and then add detail for each of those tasks. Your implementation should be developed in the same manner. Here's a suggested order of implementation.

First, implement the input code to read the employee name and id, and write the output code to print them back out in the specified format. This will require declaring the appropriate stream variables and setting up the file streams, and declaring

and using variables to store the input values. Test this code and make sure it produces correct results. Once you know this part works you should never have to worry about it again.

Second, implement the input code to read the other data values. Add code to just echo the values you read in, to be sure you're reading them correctly.

Then add the code to calculate the taxes and gross pay and output them. Test this code. Now you should be ready to test all of the code.

By implementing the program feature by feature you let yourself concentrate on solving one small part of the problem at a time. You also should only have to test one part of it at a time as well.

Evaluation:

Everything that you have been told about testing in class applies here. Do not waste submissions to the Curator in testing your program! There is no point in submitting your program until you have verified that it produces correct results on the sample data files that are provided. If you waste all of your submissions because you have not tested your program adequately then you will receive a low score on this assignment. You will not be given extra submissions.

Your submitted program will be assigned a score, out of 100, based upon the runtime testing performed by the Curator System. We will also be evaluating your submission of this program for documentation style and a few good coding practices. This will result in a deduction (ideally zero) that will be applied to your score from the Curator to yield your final score for this project.

Read the *Programming Standards* page on the CS 1044 website for general guidelines. You should comment your code in the same manner as the code given for the first two programming assignments. In particular:

- You should have a header comment identifying yourself, and describing what the program does.
- Every constant and variable you declare should have a comment explaining its logical significance in the program.
- Every major block of code should have a comment describing its purpose.
- Adopt a consistent indentation style and stick to it.

Your implementation must also meet the following requirements:

- Choose descriptive identifiers when you declare a variable or constant. Avoid choosing identifiers that are entirely lower-case.
- Use named constants instead of literal constants when the constant has logical significance.
- Use C++ streams for input and output, not C-style constructs.
- Use C++ string variables to hold character data, not C-style character pointers or arrays.
- Note: you are explicitly forbidden to write any user-defined functions for this program. This may make the program somewhat repetitive, and physically longer than the alternative. To some extent, that's the reason for this restriction.

Understand that the list of requirements here is not a complete repetition of the *Programming Standards* page on the course website. It is possible that requirements listed there will be applied, even if they are not listed here.

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* and submission link can be found at:

<http://www.cs.vt.edu/curator/>

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.