

Payroll Program — Putting It All Together

This project will (more or less) combine the last two projects, with some modifications and some extensions. You should be able to recycle some of your code, and quite a bit of your designs, from those projects. This program is substantially more complex than the earlier ones, so it is important that you design your solution carefully, and then implement your design one piece at a time.

As in project 4, this program will handle calculating gross pay, including overtime, for an hourly employee. In addition, this program will calculate withholding amounts, similarly to project 3, and compute a final net pay amount. However, the input file will be substantially different from the previous two projects. The employee's gross pay amount will be calculated using exactly the same rules as in project 4, except that the input file will specify one of three pay level codes:

Pay Code	Hourly Wage
M	17.50
C	9.50
L	8.00

The FIT withholding will be calculated using a “bracketed” approach (similar to the gross pay calculation). The tax rate to be applied depends upon the employee’s gross pay amount, as shown in the table below:

Bracket	Rate
0.00 to 350.00	12.5%
350.01 to 500.00	25.0%
500.01 and up	32.5%

For instance, if an employee’s gross pay amount was 700.00, then her FIT withholding would be:

```
FIT == 350.00 * 0.125 // first 350.00 taxed at 12.5%
+    150.00 * 0.25  // next 150.00 taxed at 25%
+    200.00 * 0.325 // last 200.00 taxed at 32.5%
==    146.25
```

The FICA withholding is still just 6.2% of the gross pay amount, the FMED withholding is still just 1.45% of the gross pay amount, and the SIT withholding is just 5% of the gross pay amount (regardless of how much the gross pay is).

The insurance premium depends upon which of three plans the employee has selected. This will be indicated by an insurance plan code, corresponding to the following premiums:

Ins. Code	Premium
A	12.75
B	8.00
C	0.00

(The premium for the third plan is covered entirely by the employer.)

The input file:

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

Project 5 sample input file			
Name	Hours	Pay	Ins
=====			
Dwarf, Sneezy	42.50	L	A
Dwarf, Grumpy	32.75	L	C
Dwarf, Happy	40.00	M	A
White, Snow	32.00	C	C
Dwarf, Doc	46.00	C	A
Dwarf, Lazy	8.50	L	C

The input file begins with four header lines, which your program should ignore. Each of the remaining lines contains four pieces of payroll data for a single employee:

- employee name a string of no more than 25 characters, containing no tabs
- hours worked a nonnegative decimal value
- pay rate code one of the characters 'M', 'L' or 'C'
- insurance plan code one of the characters 'A', 'B', or 'C'

There will be a single tab separating these items, and all are guaranteed to be logically correct. A newline character will immediately follow the insurance plan code.

There is no specified limit on the number of employee data lines, so your program must be designed to detect when it's out of input and stop automatically. To do this, you should use the read-to-input-failure loop discussed in the notes (4.22–4.24).

What must be calculated:

For each employee, your program must calculate the gross pay, withholding for FIT, FMED, FICA, and SIT, and the insurance premium, and finally the net pay amount. These must be computed according to the rules given on the first page of this specification.

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

Programmer: Bill McQuain							
CS 1044 Project 5 Fall 2000							
Name	Gross	FIT	FICA	FMED	SIT	Ins	Net

Dwarf, Sneezy	350.00	43.75	21.70	5.08	17.50	12.75	249.23
Dwarf, Grumpy	262.00	32.75	16.24	3.80	13.10	0.00	196.11
Dwarf, Happy	700.00	146.25	43.40	10.15	35.00	12.75	452.45
White, Snow	304.00	38.00	18.85	4.41	15.20	0.00	227.54
Dwarf, Doc	465.50	72.63	28.86	6.75	23.28	12.75	321.24
Dwarf, Lazy	68.00	8.50	4.22	0.99	3.40	0.00	50.90

The first two lines specify the programmer and the assignment, and the third is blank. The fourth and fifth lines provide column labels and separate the header information from the computed payroll information.

As before, all monetary amounts must be printed with precision two. The ordering of the columns must conform to the example given above, and you must print a delimiting line for the bottom of the table, as shown. The exact spacing is unimportant, but you should align your output neatly.

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 separated, 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.
- Use variables (or constants) of type `string` to character string data (not `char` variables or `char` arrays).
- Format your source code sensibly. For example, you should indent the bodies of if-else statements and loops (say three or four spaces) to improve readability.

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 will definitely have a TA 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, appropriate use of named constants, etc.). Therefore, you should compare your comments to those given in the programs for projects 1 and 2. The programs serve as a useful a guide to acceptable documentation style at this point in the course.

Your instructor will specify how the score from the TAs 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.