

Procedural Decomposition:**Water Bill**

For this project, you will design and implement a simple program to create water bills for a utility system. The primary focus of this assignment is the use of functions; you must design your program by analyzing the specification below to identify the major tasks that must be carried out, and then create a program that uses cooperating functions to carry out those tasks.

You will be given data about the water usage by a customer over a number of months. From this, you will compute the appropriate charge for each month, according to the rules given below, and the total amount of water used and the total charge over all the relevant months.

For each month in which some water was used, the customer is charged a base amount of \$22.50. If the customer used more than 1000 gallons, then s/he is charged \$0.10 for each gallon over 1000 that was used. In addition, if the customer used more than 1500 gallons then s/he is subject to a conservation surcharge of \$10.00 for that month.

The comments and requirements about representing monetary amounts that were made in the specification for the previous projects still apply.

Sample input and corresponding output:

Here is a sample input file for the program, named "WaterUsage.txt". The first two lines identify the customer, the third gives the account number, the fourth is blank, the fifth specifies the number of months for which usage data will be given, the sixth is blank, and each of the remaining lines specifies the name of a month and the number of gallons of water used during that month.

```
Customer:   Joe Bob Hokie
            1401 Tall Oaks Drive, Blacksburg VA  24060
Account:    W13421

Months billed: 5

January    1325
February   975
March      1040
April      1215
May        905
```

In the address lines, there is a single tab immediately before the customer name, and a newline immediately after it. There is a single tab before the street address, a comma and space after the address, a tab after the city name, and two spaces separate the city name from the zip code. There is a newline immediately after the zip code. There is a tab character immediately before the account number. Be sure to design your input code to follow that specification.

There is a single space immediately before the number of months, which will always be a non-negative integer. There is no given limit on the number of months for which data will be given. In each line of usage data, there is a single tab and possibly one or more spaces immediately after the name of the month and before the usage. The usage values will always be non-negative integers.

The input file is guaranteed to conform to this description. It is possible, although unlikely, that the usage data will include more than the specified number of months. In that case, you should be sure your implementation processes the specified number of months and ignores the rest. As always, you should check for input failure as well.

The output must be written to a file named "WaterBill.txt". As usual, it begins with two lines identifying the programmer (you) and the specific project, followed by a blank line.

Next, there are three lines reporting the name and address of the customer. Note that the customer's address is not formatted as it was in the input file. That means you must parse the address into parts and write it out as shown in the

sample output file. All the whitespace shown in the output file is spaces, not tabs. That won't matter at all to the Curator, but it does make it easier for you to align your output nicely.

```
Programmer: <your name here>
CS 1044:    Water Bill

Bill to: Joe Bob Hokie
          1401 Tall Oaks Drive
          Blacksburg, VA 24060
Acct #: W13421

Month      Gallons      Charge
-----
January    1325        55.00
February   975         22.50
March      1040        26.50
April      1215        44.00
May        905         22.50
-----
Totals:    5460        170.50
```

The account number is echoed after the customer address, followed by a blank line. After that, there is a table reporting the water usage and corresponding charge for each month that was processed. Be sure to use the label text shown above. Finally, there is a line reporting the total gallons used and the total charge.

If you have read the *Student Guide to the Curator*, you already know that all the fixed text must be precisely as shown in the sample output. The output should be aligned for easy readability.

Additional samples of input and correct output will be available on the course website.

Suggested implementation plan:

You should begin by creating an outline that identifies the major steps in the algorithm for creating the water bill, and then adding detail for each of these steps until the entire process becomes clear.

You should then implement your design feature-by-feature. For instance, begin by writing the code necessary to read the customer data and write it out in the specified format. Then write the code to read the remaining input data and write out an incomplete water bill that does not include any computed charges. (It might be helpful to just write out dummy values at this point.) Be sure to follow the specific requirements for writing functions that are given in the next section of the specification.

Once you have the input/output working, you should implement the logic to compute the monthly charges and test that. Once that is working correctly, implement the logic to compute the totals for the last line.

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 may also evaluate 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 function aside from `main()` must have a header comment block as described on the course website.
- 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 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 required to write user-defined functions for this program. In particular, you are required to implement at least four functions besides `main()`. You must implement at least one function that reads input from the input file, and at least two that write data to the output file. One of the output functions must take an integer value and an output stream as parameters and write the integer as a properly formatted dollar amount. (My implementation includes two that read input and four that write output.) You must implement at least one function that is not `void`.

Because the requirements for functions are so important to the role of the assignment in the course, when the TAs evaluate your submission (and they will), they will assign a final score of zero to anyone whose program consists only of `main()`.

Do not implement your program first without functions and then try to break it up into separate functions. All that will accomplish is waste your time and make your task harder. It may also tempt you to submit a solution that is NOT decomposed into separate functions. If you do so, and you get 100, then that will be graded by the TAs, and you will receive a final score of zero for the project.

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* can be found at: <http://www.cs.vt.edu/curator/>

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