CS 121 Final Lab

General Problem Solving with C++ Paul Cao

As an exercise to practice what we have learned in this semester, we will use the last two lab sections to solve a more complex problem with relatively more complicated code. The deadlines this final lab is given later in the lab manual. This lab will counts for two labs towards your final grade.

Let's first have a review of the materials we have learned.

1. C++ basics

In chapters 1 and 2, we learned how to declare, initialize, and use variables. Of all the C++ types, double, int, char, and bool are the mostly commonly used variables in a regular code. string type variables are also widely used but we have to include the string directive. The values of variables can be changed using the assignment operator, =. The value of the expression on the right side of the = operator will be assigned to the variable on the left. We also learned simple I/O using cin and cout in these two chapters.

2. Flow Control

In chapter 3, we learned how to control the execution of statements (i.e. each line terminated by a semi-colon). The default flow is sequential: starting from the first statement in main, statements will be executed sequentially until the last statement in main. With the help of branching statements (if, if…else…, if…else if…else, and switch) and loops (while, do…while, and for), we can control our code in a more flexible manner to model a real problem. It is guaranteed that branching and loop structures are enough for model *any* problem in the real world!

3. Procedure Abstraction (aka Functions)

To make our code more flexible and more portable, C++ allow programmers to use functions. There are many pre-defined standard functions available for us to use directly. All we need to do is to include correct directives before we use them. As programmers, we can also write our own functions and use them in our code. The three important components of a function is function declaration, function call, and function definition. Each of those components has its purpose and strict syntax rules should be followed properly. Through chapters 4 and 5, we introduced call-by-value functions, call-by-reference functions, overloaded functions, and functions with default values. The basic principle of function design is the idea of blackbox. Any function that we write should allow the user to use it without knowing how we implemented it. Thus proper documentation is very important.

4. I/O Streams

I/O are doors for the program to communicate with the outside world. cin/cout are default I/O streams for keyboard and screen interactions. In chapter 6, file I/O is introduced. We need to create appropriate file I/O objects from ifstream or ofstream classes. The sequence of method calls (i.e. open(), fail() and close()) are necessary to ensure a complete I/O operation. In the later part of chapter 6, we will discuss more on format control and character I/O.

5. Microsoft Visual Studio IDE

We have used VC 2008 as our default programming environment. The most important feature is the debugging environment in VC. With the help of breaking point and stepping, we can find the logic and run-time bugs in our code. Concise comments may also help avoiding unnecessary errors in our code.

The purpose of this project is to allow you to utilize what we have learned in this semester, and use them to solve a problem of your interest. Here are a few guidelines for the project.

- You are encouraged to form a group of 2 persons. But if you prefer to work on your own, it is ok. If you prefer that I assign a problem to you like in our regular lab, please inform me before 4pm on 11/19/2010. If you need help finding a partner for the project, please let me know and I will help.
- Your group should pick a problem of your interest and the problem should be completed within a 2-week period. It is important that you don't pick a problem that is too big. I would recommend that a code of at most 400 statements for a group project and 200 lines for a single-person project.
- Each group member should contribute to the project. In the final presentation of the project, it is required that each member demonstrates the part of the project that is implemented by him or her.
- The timeline of this project is the following

  Project proposal due: 11/22/2010 before 4pm (via email. Each group only needs to submit one proposal.)
  Project presentation: 12/3/2010 in class
  Final code due: 12/8/2010 before 4pm (using the dropbox on angel)

The project proposal should include the following information. I will send you my comments on the project.

- The group information (group members)
- The problem you want to tackle for the project. It is important that you are precise in describing the problem. For example, what is the input, what is the output, what are the conditions you want to verify in the code, etc.
- How the project will be divided between group members
- The general strategy to solve the problem. A flowchart might be very helpful. You can handwrite the flowchar and scan it. You don't have to use professional software to draw the flowchart.

Each group will have about 5-10 minutes to present their project in the last lecture of our class. Your group should demonstrate the code and illustrate how you solved the problem. Each group member should explain the part of the code that he/she wrote.

A dropbox will be available on angel for the final code. No final report is required for this project.

Requirements

1. There must be at least two programmer-defined functions. You should carefully design the functions and include detailed explanations as comments about what the function does, what

it takes, what I/O it does, and what it returns (including call by reference variables). You should include an example of a sample function call in the comments.
2. Your code should use file I/O at least once.
3. Your code must consider normal conditions and handle user or calculation errors properly in your code.
4. Your code must be written with clear tabulations and is well commented.

Grading

Soundness of coding:	10%
I/O operations:	20%
Error handling:	20%
Function usage/definition:	30%
Documentation/Comments:	20%

Be Creative! Have Fun!