Objects and Problem Solving

9-5-2001

Opening Discussion

- Last class period we spent most of the time going over the syllabus and discussing what material will be covered in the class.
- Are there any questions that have occurred to you since last class?
- Slides are on the web.

Quote on Debugging

- "As soon as we started programming, we found to our surprise that it wasn't as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs."
  - Maurice Wilkes discovers debugging, 1949

Blowing up a Balloon

- Through this class I am going to use the example of blowing up a balloon.
- The intention is to show you the value of each step of the process that we will be discussing.

Objects

- This course will focus on the methods and processes of Object-Oriented Programming. In object oriented programs the key component is the object.
- An object in a program is very much like an object in real life, only simplified so that it only includes the details actually needed for that program.
- Objects have attributes (state information) and behaviors (what they can do).

Software Lifecycle

- The act of creating an application on a computer is more than just programming. When a program is large and has many functions, the programming step can be one of the smallest parts.
- Focussing on the other parts of the "software lifecycle" makes the process of writing the code easier and more direct and can help to insure that the code you produce works properly.

Analysis

- The first step in the software lifecycle is analysis. This is basically the act of figuring out what the problem is that you are trying to solve.
- This might seem pedantic, but it can often be one of the most challenging parts of the problem.
- For us this includes determining what objects are part of the problem.
- This part of the process is typically done without thinking about the computer and the results of it are often called specifications.

Design

- The second step in the software lifecycle is design. This is where you decide how the specification that was developed will actually be made to come alive on a computer.
- In this stage you have to decide what the attributes and behaviors are of the various objects and how they will work together.
- If you find your analysis was improper it can be changed.

Implementation

- This is the step that most people think about as programming, when you actually write code in the chosen language that brings the design to life.
- Over the course of the semester we will focus on how to do this in C++. While this step is vital it is in many ways the easiest and most straightforward to do, especially if you have a good design.

Testing

- Once you have a code that you believe works you have to determine if it really does what the specifications asked for. This is the phase called testing.
- During testing you try to code under all possible conditions and inputs to make sure that it is stable (doesn't crash) and performs the task that you want it to.
- Good testing requires taking the time to think of good tests.

Error Handling

- Testing can and should be done one piece at a time with code. Part of the testing in small pieces of code is what would be called error handling.
- When a small piece of code is not able to function the way it should for one reason or another, it should tell the rest of the program that in some well specified way.
- Each part of an algorithm should make sure that is "getting" something it can work with.

Maintenance

- For professional software the end of the line is not when it goes out the door. Instead that just ushers in the last step in the process, maintenance.
- This is the process of fixing the bugs that made it through in house testing and making modifications as users request new functions.
  - Note that that last part basically requires going all the way back to analysis.

Compilers

- Because we do not speak the same "language" as the machine there is a significant difference between what we write as a program and what the computer understands.
- To bridge this gap we use a "compiler". On our computer systems the compiler is the GNU compiler and for C++ is is called g++.

Value of Object-Orientation

- Object-orientation has many benefits.
  - The book mentions code reuse where you only have to write the code once and it can work in many ways for you.
  - Also important are encapsulation, separation of interface and implementation, and general abstraction. I think that in the real world these often wind up being more important than code reuse.

Minute Essay

- What are the steps in the software lifecycle and why is each one important? Is it clear why you don't want to just "hack" out code?
- Set up your CS account before Friday's class.
- You need to read Chapter 2 by Friday (at least the first 3 sections). There will be a quiz at the beginning of class so make sure you are on time.