Modularity in Code

10-14-2002

Opening Discussion

- What did you talk about last class?
- Do you have any questions about the assignment that is due on Wednesday?
- Let's revisit the algorithm for making a peanut butter and jelly sandwich. What did it look like? How would you break it into functions?

Pass by Reference (Quick Review)

- On Wednesday you talked about pass-by-reference. This is when you pass pointers as arguments into a function.
- The power of pass by reference is that the function has the ability to change the values that the pointer points to and as a result, change the value of something visible to the rest of the program.
- This can be used to "return" multiple values.

Problem Solving Again

- Now that you know how to write functions and have some idea of how to use them we can revisit the ideas of breaking problems into pieces as well as the top-down and bottom-up approaches to design.
- Functions are what give you the ability to do this. Often, you can't see how the whole program works, but you can build it one function at a time.

Bite Size Pieces

- Functions are the little bite sized pieces that you can break your program up into and easily tackle one piece at a time.
- Sometimes a function will do nothing more than call some other functions. Writing these types of functions gives you that ability to organize your thoughts on what the rest of the program should do.

Not So Bite Sized Pieces

- Quite often you will find that there are groups of functions that go together in some logical way to solve a problem and are somehow logically separate from the rest of your code.
- These larger chunks are often called modules and when done well they can be reusable as libraries.
- It is good programming practice to reduce the dependencies between modules.

The Software Lifecycle

- The process of making software is often broken into 5 pieces (not always linear).
  - Analysis - Figuring out what the problem you are trying to solve really is.
  - Design - Decide how to solve that problem in code.
  - Implementation - Write the code.
  - Debugging - Find and correct the problems in your code.
  - Maintenance - Changes and fixes for users.

Role of Modularity

- Modularity plays a big role in the last four phases of the software lifecycle.
- Building good modules into the design makes it easier to keep the code organized.
- Modules can be tested/debugged independently much of the time.
- Changes made to one module should have a limited effect on other parts.

Minute Essay

- Assignment #4 has you breaking a larger problem up into pieces. In some manner those pieces are related. What functions might be a module in the problem that you are working on?
- Next class we start talking about arrays. Especially if you are working on the second option for assignment #4, looking ahead a bit now could help shorten your code.