Recursion

3-18-2002

Opening Discussion

- I hope everyone had a good spring break.
- What did we talk about in the last class before the break?
- I want to post code for assignment #3 on Wednesday. Assignment #4 has been posted on the web page.
- Can you tell me what a recursive function is? Can you give me any examples where recursive functions are helpful?

Recursive Functions

- A function that can call itself is called a recursive function.
- They exist in mathematics as functions that are defined partially in terms of themselves.
- We actually use them for many tasks in computer programming.
  - When we can divide problems.
  - When we need to test multiple "paths" from a point.

Our Simple Example

- We will discuss the example of printing out 10 numbers that are input by the user.
- This seems like a remarkably simple thing to do, but it actually can do a remarkably good job of illustrating many different features of recursive functions.

Proof by Induction

- One of the things we like to be able to do in computer science (when possible) is prove that our program actually work.
- One of the best tools we have for this is proof by induction. It works in much the same way that recursive functions do in general.
  - Show something is true for a base case.
  - Show that if it is true for a set of cases it is also true for the next one.

Why Do Induction Here?

- We cover induction along with recursion in large part because the logic behind them is very similar.
- When we build recursive solutions to problems we typically envision it as something where we can solve a very simple case and where we know how to build a "larger" solution using one or more smaller ones.

Recursive Base Cases

- Any successful recursive function has to have a base case. This is the situation where it doesn't recurse further. Without this you will get infinite recursion which gives you segmentation faults that gdb can't give you a where on (because you kill the stack).
- A base case is typically small, but doesn't have to be. It just has to be directly solvable.

Our Example

- How are we going to write a recursive function for our example? What is the base case of our example?
- What do the other cases do?
- Can we do anything with this that would be more difficult to do with a loop? What about printing the numbers in the reverse order of how they were entered?

How it Works

- When you call functions in C++, they are put on the "stack". This creates new copies of local variables and arguments passed by value.
- You have to be very careful with what things are passed by value and what are passed by reference. If you pass a reference or a pointer you don't get a new copy. This reduces overhead, but can produce results you didn't want.

Driver Routines

- Sometimes there is extra work that needs to be done in the first call to the function only, or there are "hidden" parameters that the end user doesn't need to know about. Sometimes the user doesn't even know what to pass the "hidden" params.
- For this we often have one function that the user calls that then calls our recursive function.

Overuse of Recursion

- Just because something can be solved with recursion doesn't mean it should be.
- The classic example of this is Fibonacci numbers. They are most easily defined as a recursive function, but it is very inefficient to implement them that way.
- A loop solution is O(n), a recursive one is O(2^n).

Minute Essay

- Give an example of a function or a problem where you think using a recursive function would be helpful. Write some pseudocode for it.
- The design for assignment #4 is due next Monday. From here out the due dates are on Mondays. You don't want to start on Friday!