Loop Basics and while Loops

9-28-2001

Opening Discussion

- What did we discuss last class?
- One of the concepts introduced last class was how to use recursion to create looping logic in a program. Starting today we will look at the more customary method of having loops in programs.
- There are three methods of implementing loops in C++. Can you name one?

Other Notes

- On assignment #2 you probably NEED to do a design. A good solution to it will involve multiple functions and will exercise quite a few of the things we talked about with function calls, variables, and arguments. You need to THINK about how you will do things before you try to write it.
- Reading from a file in assignment #2. (You're only going to fake it.)

while Loops

- The simplest loop construct in C++ is the while loop. It has the following syntax.

```
while(conditional) {
    statements;
}
```

- When a while loop is reached during execution, the computer executes the code inside until the condition is false at the end of the block.
- Let's look at two examples.

Conditionals in Loops

- Conditions in while loops are boolean expressions just like those we saw for if statements. In the if statements the conditional determined whether a section of code would be executed or not. In a while loop it determines how many times it will be executed.

Initializing

- Before you get to a while loop you typically need to do something it initialize the state of the system. This generally means that you need to have declared and put values into the variables that you are using in the conditional expression.
- In a loop that counts from 0 to n-1 this would involve having n set up and declaring a counter variable that starts at 0. This can be viewed at setting the loop preconditions.

Iterator

- A loop also needs to do something that "advances" it to the next case it is going to process. Without this the state of the conditional will never change and you have an infinite loop.
- In a counting type loop the iterator will increment or decrement the counter variable by some value. Not every loop is a counter loop and other iterators could be reading from an input source or moving through a linked data structure.

Processing

- The code in the block of the while loop that isn't part of the iterator does whatever type of processing is intended for that loop.
- Let's look closely at what type of processing was done in the two examples that we have been using.

Loop Invariants

- When trying to write correct loops it can be very helpful to isolate sets of statements that are true through the entire execution of the loop. The most helpful sets of statements are the ones that are most closely related to what the loop should be doing.
- Good invariants can actually allow you to do proofs of correctness where you show that a loops will terminate and will perform the calculation you are interested in.

Minute Essay

- Write a loop that calculates the sum of the squares of all the numbers between 1 and n, where n is a local integer variable.
- Next week we will continue with our discussion of loops. Remember that assignment #2 is due on Monday. I need printed versions of your design and your code plus an electronic copy of the code via e-mail. Please don't use lpr for printing.