do-while and for Loops

10-1-2001

Opening Discussion

- What did we talk about last class?
- At the end of last class I asked you to write a loop that would find the sum of the squares between 1 and n.

```
int i=1;
int sum=0;
while(i<=n) {
     sum+=i*i;
}
```

do-while Loop Syntax

- There is another loop construct nearly identical to the while loop that you use if you know that the body of the loop should execute at least once: the do-while loop.

```
do {
     statements;
} while(condition);
```

- It requires all the same "parts" as the the while loop but the condition is not checked until after it has executed once.

while vs. do-while

- A while loop can be made to do anything you need from a loop as can a do-while loop. However, both exist in the language because different ones come in handy at different times.
- If you know that a loop should execute at least once the do-while structure typically makes the logic a bit easier and cleaner.

for Loop Syntax

- In C++ the for loop simply allows you to combine initialization and iteration into the basic loop structure. It has the following syntax.

```
for(initialization; condition; iterator) {
    statements;
}
```

- It behaves much like a while loop. You can declare variables in the initialization.

Comma "Operator"

- Notice that in the for loop the semicolons play the role of separating the initializer from the condition and the condition from the iterator. That means if you want to have multiple expressions in them you have to separate then with something else.
- The comma works as an operator to separate expressions. The value of the complete expression is that of the last subexpression in the comma separated list.

Increment and Decrement

- While you have seen these is my examples before we should take a closer look at the increment (++) and decrement (--) operators.
- When placed before a variable the variable is either increased or decreased by one before the statement executes. If it is after then it happens after the statement executes. In most situations it doesn't matter.

Examples

- Now we will look at some sample code that illustrates how the three types of loops can be constructed and what the relative strengths of each are.
  - for - Great for numeric loops because of compact structure. Makes it harder to forget things too.
  - while - General purpose when initilizer or iterator are complex.
  - do-while - always executes once.

Minute Essay

- Write a for loop that computes x^n where n is an integer. Remember that '^' is not exponentiation in C++.
- Next class we will finish with loops before we start working on arrays.