Conditionals

9-26-2003

Opening Discussion

- What did we talk about last class?
- Do you have any questions about the assignment?
- Pass by value limitations.

What can we do now?

- We have seen how to declare variables, write expressions involving numeric expressions, print things to screen, read input from the user, and break things into functions.
- This allows us to do almost any sort of straightforward calculation that we would do on a calculator. An example might be figuring out how much a meal should cost or how much someone makes.

Branching Code and Conditionals

- The problem with what we know now is that the same statements get executed every time we run the program. It always goes from the top to the bottom, jumping to the same functions at the same time.
- Today we will fix that by allowing our code to branch with conditional statements. With a conditional we can say that certain parts of the code only should execute under certain conditions.

Example

- What if we wanted to calculate the wages of a person where we took into account overtime if the person worked more than 40 hours in a week? What if we they got double time for hours over 60?
- That is difficult to do if the same code gets executed all the time. For that it helps to have conditional execution.

if-else Statements

- The most basic conditional is the if statement. It can have an else clause but doesn't have to.

```
if(condition) {
     statements1;
} else {
     statements2;
}
```

or

```
if(condition) {
     statements;
}
```

- If the *condition* is true, the first set of statements is executed. Otherwise the else block is executed if one is there.

Boolean Expressions

- The *condition* should be a form of Boolean expression. This is an expression that is true or false.
- The way that things work in C, Booleans are actually integers where zero is false and everything else is true. This can cause problems unless you are careful.
- Boolean expressions typically have comparisons joined together by Boolean operators.

Comparisons

- The comparisons you use in Java are like what you are used to, though perhaps typed in a bit differently.
  - == tests equality (this can cause you problems)
  - != tests inequality
  - <, <= for less than or less than or equal to
  - >, >= for greater than or greater than or equal to

Boolean Operators

- We often want to combine simple comparisons or other Boolean expressions together to make more complex Boolean expressions. We do this with Boolean logic operators.
  - || - or (this is a short-circuit operator)
  - && - and (this is a short-circuit operator)
  - ! - not

Order of Operations

- As with numeric operators there is a fixed order in which operators are evaluated in an expression for Boolean operations as well. Again here, when in doubt use parentheses.
- In general the comparisons happen before the Boolean operators. && should happen before ||.

Basic Recursion

- A recursive function is a function that calls itself. The thing is it can't always call itself so we have to have conditionals to do it.
- When we do this we have an argument to the function that can change and when it has a certain value we don't call the function again.
- This is a simple way of doing looping like functionality.

Revisit the Wages Example

- So now let's write some code for the example of calculating wages. Has anyone thought of how you could do this example without conditionals?
- Let's also write something to do simple recursion. We can print numbers for counting or calculate a factorial.

Minute Essay

- Write a section of code that checks a variable age and sets a variable rate depending on it. If age is less than 30 set rate to 0.1*age. If age is between 20 and 50 add another 0.05*age to what rate was.