CSC161 2011S *Imperative Problem Solving*

Class 20: Control Flow: Conditionals (2)

Held: Friday, 25 February 2011

Summary: We consider additional forms of conditionals in C.

Related Pages:

- EBoard.
- Reading: K&R 3.4.

Notes:

- Celebration of MathCounts! success yesterday.
- Sorry for not posting obvious links to the reading last night.
- Next reading: K&R 3.4-3.5.
- Assignment 5 is now ready.
- EC: CS Table today!

Overview:

- Case statements.
- Conditional expressions.
- Lab.

The Switch Statement

Form:

```
switch (integer-expression):
  {
    case val0:
    case val1:
      statements;
    case val2:
      statements;
      break;
    default:
      statements;
      break;
  } // switch
```

Meaning:

- Evaluate the expression.
- Step through the cases until you hit the first matching value.
- Execute any subsequent statements until you hit `break`, `return`, or the end of the statement.

Useful because

- Less code than the nested if statements.
- Can be compiled to more efficient code than the nested if statements. (It isn't always compiled to more efficient code, but it can be.)
- The "fall through" option is powerful (although dangerous).

Conditional Expressions

Context:

- Conditional statements (particularly the `if`) can be used to select between different sequences of instructions.
- What if we want to select between different expressions, as in Scheme's `if`?
- "Let x be the larger of y and z."
- Enter the conditional expression.

Form: *`test ? consequent : alternate`*

Meaning:

- Evaluate the test.
- If the test evaluates to a non-zero number, evaluate the consequent and use its value.
- If the test evaluates to zero, evaluate the alternate and use its value.

Examples

- `x = y *gt; z ? y : z;`

Lab

- Do the lab.
- Be prepared to reflect.
