Week 5: switch statements
and programming with conditions

Gaddis: 4.10-15
CS 1428
Fall 2015
Jill Seaman

4.11 Validating User Input

- Input validation: inspecting input data to determine whether it is acceptable
- Invalid input is an error that should be treated as an exceptional case.
  - The program can ask the user to re-enter the data
  - The program can exit with an error message

```cpp
cout << "Enter a positive number: ";
cin >> x;
if (x > 0) {
    // Do something with x here
} else {
    cout << "You entered a negative number or 0." << endl;
    cout << "The program is ending." << endl;
}
```

4.12 Comparing Characters and Strings

- Characters are compared using their ASCII values
  
  - 'A' < 'B'
    - This is true.
    - ASCII value of 'A' (65) is less than the ASCII value of 'B' (66)

  - '1' < '2'
    - This is true.
    - ASCII value of '1' (49) is less than the ASCII value of '2' (50)

- Lowercase letters have higher ASCII codes than uppercase letters, so 'a' > 'Z'

Comparing string objects

- Like characters, strings are compared using their ASCII values

  ```cpp
  string name1 = "Mary";
  string name2 = "Mark";
  name1 > name2  // true
  name1 <= name2 // false
  name1 != name2 // true
  name1 < "Mary Jane" // true
  ```

The characters in each string must match exactly in order to be equal
Otherwise, use first non-equal character as basis of the comparison ('y'>'k')
If a string is a prefix of the other, then it is less than the other
4.14 The switch statement

- Like a nested if/else, used to select one of multiple alternative code sections.
- tests one integer/char expression against multiple constant integer/char values:

```
switch (expression) {
    case const1: statements
    ...
    case const2: statements
    default: statements
}
```

**switch statement behavior**

- expression is evaluated to an int/char value
- execution starts at the case labeled with that int/char value
- execution starts at default if the int/char value matches none of the case labels

**switch statement syntax**

- expression must have int/char type
- const1, const2 must be constants! a literal or named constant
- statements is one or more statements (braces not needed and not recommended!)
- default: is optional

```
switch (expression) {
    case const1: statements
    ...
    case const2: statements
    default: statements
}
```

**switch statement example**

- Example:

```
int quarter;
...
switch (quarter) {
    case 1: cout << “First”;
         break;
    case 2: cout << “Second”;
         break;
    case 3: cout << “Third”;
         break;
    case 4: cout << “Fourth”;
         break;
    default: cout << “Invalid choice”;
}
```
The **break** Statement

- The break statement causes an immediate exit from the switch statement.
- Without a break statement, execution continues on to the next set of statements (the next case).
- Sometimes this is useful: the textbook has some nice examples.

```
char ch;
...
switch (ch) {
    case 'a':
    case 'A':
        cout << "Option A";
        break;
    case 'b':
    case 'B':
        cout << "Option B";
        break;
    case 'c':
    case 'C':
        cout << "Option C";
        break;
    default:
        cout << "Invalid choice";
}
```

Multiple labels

- if ch is ‘a’, it falls through to output “Option A” (then it breaks)

```
4.10 Menus

- **Menu-driven program**: program controlled by user selecting from a list of actions
- **Menu**: list of choices on the screen
- Display list of numbered/lettered choices
- Prompt user to make a selection
- Test the selection in nested if/else or switch
  - Match found: execute corresponding code
  - Else: error message (invalid selection).

```
// Display the menu and get a choice.
int choice;
double charges;
int months = 12;
for (int choice = 1; choice <= 3; choice++) {
    cout << "Health Club Membership Menu\n\n";
    cout << "1. Standard Adult Membership\n";
    cout << "2. Child Membership\n";
    cout << "3. Senior Citizen Membership\n";
    cout << "Enter your choice: ";
    cin >> choice;
    switch (choice) {
        case 1:
            charges = months * 40.0;
            cout << "The total charges are $" << charges << endl;
            break;
        case 2:
            charges = months * 20.0;
            cout << "The total charges are $" << charges << endl;
            break;
        case 3:
            charges = months * 30.0;
            cout << "The total charges are $" << charges << endl;
            break;
        default:
            cout << "ERROR: The valid choices are 1 through 3." << endl;
    }
```
4.15 More about blocks and scope

- The scope of a variable is the part of the program where the variable may be used.
- The scope of a variable is the innermost block in which it is defined, from the point of definition to the end of that block.
- Note: the body of the main function is just one big block.

Variables with the same name

- In an inner block, a variable is allowed to have the same name as a variable in the outer block.
- When in the inner block, the outer variable is not available (it is hidden).
- Not good style: difficult to trace code and find bugs

Scope of variables in blocks

```c++
int main()
{
    double income; //scope of income is red + blue
    cout << "What is your annual income? ";
    cin >> income;

    if (income >= 35000) {
        int years; //scope of years is blue;
        cout << "How many years at current job? ";
        cin >> years;
        if (years > 5)
            cout << "You qualify.\n";
        else
            cout << "You do not qualify.\n";
    }
    else
        cout << "You do not qualify.\n";
    cout << "Thanks for applying.\n";
    return 0;
}
```

Cannot access years down here

Variables with the same name

```c++
int main()
{
    int number;
    cout << "Enter a number greater than 0: ";
    cin >> number;
    if (number > 0) {
        int number; // another variable named number
        cout << "Now enter another number ";
        cin >> number;
        cout << "The second number you entered was ";
        cout << number << endl;
    }
    cout << "Your first number was " << number << endl;
}
```

Enter a number greater than 0: 88
Now enter another number 2
The second number you entered was 2
Your first number was 88