Week 1
Operators, Data Types & I/O
Gaddis: Chapters 1, 2, 3

CS 5301
Fall 2017
Jill Seaman

Programming
• A program is a set of instructions that the computer follows to perform a task
• It must be translated from a programming language (C++) to machine code in order to run on the machine.

Structure of a C++ Program
• Hello world:

```cpp
//This program outputs a message to the screen
#include <iostream>
using namespace std;
int main() {
    cout << "Hello world!" << endl;
}
```

• In general:

```cpp
//This is a comment
#include <includefile> ...
using namespace std;
int main() {
    statements ...
}
```

Variables, Data Types
• **Variable**: portion of memory that stores a value
• **Identifier**: name of a program element
• Fundamental data types
  - short
  - float
  - bool
  - int
  - double
  - char
  - long
  - long double

• **Variable Declaration** statement
  - `datatype identifier;`
  - `float hours;`

• **Variable Initialization** statement
  - `datatype identifier = constant;`
  - `int count = 0;`
Integer types

- Integers are whole numbers such as 12, 7, and -99

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>short</td>
<td>-32,768 to 32,767</td>
</tr>
<tr>
<td>int</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>long</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
</tbody>
</table>

- char type stores characters such as 'A', '@', and '9'
  - The ascii code value (an integer) of the character is stored in memory.

Floating-point types (and bool)

- Floating point types store real numbers such as 12.45 and -3.8
  - They are stored using scientific notation.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>±3.4E-38 to ±3.4E38</td>
</tr>
<tr>
<td>double</td>
<td>±1.7E-308 to ±1.7E308</td>
</tr>
<tr>
<td>long double</td>
<td>±1.7E-308 to ±1.7E308</td>
</tr>
</tbody>
</table>

- bool type stores values that are true or false
  - false is 0, true is 1.

Constants

- Literals (specific value of a given type)
  - 1, 75, -2, 12.45, -3.8, 6.25e-5, true, false, 'A', '2'

- Named Constants: variable whose value cannot be changed
  - const datatype identifier = constant;
  - const double TAX_RATE = 0.0675;

Assignment statement, expressions

- To change the value of a variable:
  - variable = expression;
  - count = 10;

  - The lefthand side must be a variable
  - The righthand side is an expression of the right type

- What is an expression?
  - an expression has a type and evaluates to a value
    - literal
    - named constant
    - variable
    - arithmetic expression
    - etc.
Arithmetic Operations

- arithmetic operators:
  - + addition
  - - subtraction
  - * multiplication
  - / division
  - % modulo (remainder)

- Integer division:

  \[ 14 \div 3 = 4 \text{ r. } 2 \text{ (because } 4 \times 3 + 2 = 14) \]
  \[ 14/3 \Rightarrow 4 \text{ in C++} \]
  \[ 14\%3 \Rightarrow 2 \text{ in C++} \]
  \[ 14.0/3.0 \Rightarrow 4.6666667 \text{ in C++} \]

Operator precedence

- In an expression with multiple operators, which one happens first?
- Use this order for different operators:
  - * - (unary negation) associates right to left
  - *, /, %, +, - associate left to right
- Use this order for multiple occurrences of the same operator
  - - (unary negation) associates right to left

Basic Input/Output

- Output (cout and <<)
  - sends data to the screen (console)
  - \text{cout} \ll << \text{expression};
  - \text{cout} \ll << \text{expr1} \ll << \text{expr2};
  - \text{cout} \ll << "hello";
  - \text{cout} \ll << "Count is: " \ll << \text{count} \ll << \text{endl};

- Input (cin and >>)
  - receives data typed in from the keyboard (stops at space)
  - \text{cin} \gg << \text{variable};
  - \text{cin} \gg << \text{var1} \gg << \text{var2};
  - \text{cout} \ll << "Enter the height and width: ";
  - \text{cin} \gg << \text{height} \gg << \text{width};
  - \text{cout} \ll << "The height is " \ll << \text{height} \ll << \text{endl};

Formatting output

- Goal: control how output displays for numeric data
- these require \texttt{#include<iomanip>}

  - \texttt{setw(x)}: print next value in a field at least x spaces wide (right justified, padded with spaces).
  - \texttt{cout} \ll << \texttt{setw(6)} \ll << \texttt{1234} \ll << \texttt{setw(6)} \ll << \texttt{5} \ll << \texttt{endl};
  - \texttt{cout} \ll << \texttt{fixed} \ll << \texttt{setprecision(2)};

  - fixed: always use decimal notation (not scientific)
  - \texttt{setprecision(x)}: when used with \texttt{fixed}, print floating point values using x digits after the decimal
  - \texttt{cout} \ll << \texttt{fixed} \ll << \texttt{setprecision(2)};
  - \texttt{cout} \ll << \texttt{3.14159} \ll << \texttt{endl};
  - \texttt{float x = 20;}
  - \texttt{cout} \ll << \texttt{x} \ll << \texttt{endl};
The string class

- **string literals**: represent sequences of chars, inside of double quotes:
  
  ```
  cout << "Hello";
  ```
  
- To define string variables:
  
  ```
  string firstName, lastName;
  ```
  
- Operations include:
  - `=` for assignment
  - `.size()` function for length
  - `[n]` to access one character in the nth position.

```
string name;
name = "George";
cout << name.size() << " ";
cout << name[2] << endl;
```