Chapter 1

C++ Basics

Hello World

```cpp
// File: helloWorld.C
#include <iostream>    // Library for input and output
using namespace std;  // Location of iostream library

int main() {
    // Execution starts here
    cout << "Hello World!" << endl;
    return 0;          // Execution ends here
}
```

- To compile:     `g++ helloWorld.C`
- To run:         `./a.out`
Keywords

- Special class of identifiers
- C++ reserved words
- Cannot be used as names for variables or anything else

**Examples:**

<table>
<thead>
<tr>
<th>bool</th>
<th>break</th>
<th>class</th>
<th>default</th>
<th>else</th>
<th>false</th>
<th>if</th>
<th>new</th>
<th>return</th>
<th>this</th>
</tr>
</thead>
</table>

Variables

- Memory location to store data for a program
- Start with a letter or underscore and rest must be letters, digits or underscores
- Case sensitive
- Use meaningful names
- Use mixed case
  - e.g. `timeOfArrival`
Variables

• Which of these are legal names?

<table>
<thead>
<tr>
<th>_abc</th>
<th>3X</th>
<th>data-1</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>myRATE</em></td>
<td>file.name</td>
<td>amtOf$</td>
</tr>
<tr>
<td>%change</td>
<td>ABC123z7</td>
<td>12</td>
</tr>
</tbody>
</table>

Variables

• Must be declared before use
  – One per line or combined
  – At beginning of block or just before first use
• Must be initialized

Simple types:

<table>
<thead>
<tr>
<th>short int</th>
<th>int</th>
<th>long int</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>char</td>
<td>bool</td>
</tr>
<tr>
<td>double</td>
<td>long double</td>
<td></td>
</tr>
</tbody>
</table>
Assignment Statements

• Used to change value of a variable
• Equal sign ( = ) is assignment operator
• Variable on left side, expression on right
  - salary = rate * hours;
• Type mismatch
  - int x = 2.99; // some compilers will complain
  - double y = 2; // this is ok
  - bool flag = 0; // false - 0, true - 1

Literals and Constants

• Literals cannot change values during execution
  - 2, 5.75, 'Z', “hello world”
• Use named constants instead (when possible)
  - const int BRANCH_COUNT = 100;
• Common naming convention for consts
Arithmetic Operators & Expressions

• Operators: +  -  *  /  %

• Standard precedence rules apply
  – Use parentheses to reduce errors and increase readability
  – \( x + y \times z \) should be \( x + (y \times z) \)

• Assignment shorthands available

<table>
<thead>
<tr>
<th>Shorthand</th>
<th>Equivalent to</th>
</tr>
</thead>
<tbody>
<tr>
<td>count += 2;</td>
<td>count = count + 2;</td>
</tr>
<tr>
<td>total -= discount;</td>
<td>total = total – discount;</td>
</tr>
<tr>
<td>amt *= count1 + count2;</td>
<td>amt = amt * (count1 + count2);</td>
</tr>
</tbody>
</table>

Arithmetic Precision

• Highest order operand determines precision
  – If all values are int, result is int
  – If one value is double/float, result is double/float
  – Division with negative numbers varies

10/4 equals ?  10/4.0 equals ?

int num = 1, denom = 2;
num / denom equals ?

1 / 2 / 3.0 / 4 equals ?
Type Casting

• Used to change a value of one type to a value of another type
• Does not change the original variable

```c
int num = 1, denom = 2;
double result;
result = static_cast<double>(num) / denom;
```

Increment & Decrement

• Shorthand to add or subtract 1 from a variable: `m++  n--`
• Pre-increment: `++m`
  – Increments variable then uses new value
• Post-increment: `m++`
  – Uses current value of variable then increments it

```c
int n = 3;
int ans = 2 * (n++);
ans equals ?
```
Console Input/Output

- cout for output, cin for input
  - cout << “Hello World”;  
  - cin >> name;
- Must include <iostream> library
- Must use std namespace

// Any program which contains input or output should start with // these lines

#include <iostream>
using namespace std;

Cout

- Can output variables, constants, literals & expressions
- Use \n or endl for a newline
- Use \t for a tab
- Separate variables from strings using <<

cout << “\nThe cost of the “ << movieName
    << “ DVD is $” << (basePrice + tax) << endl;
Formatting Numbers

- To display numbers with fixed digits after the decimal place use these flags

```cpp
cout.setf(ios::fixed);
cout.setf(ios::showpoint);

cout. precision(2);
cout << “The price is $” << price << endl;

cout.precision(3);
cout << “The extended price is $” << price << endl
```

Cin

- Always include a prompt before every cin
- Can read in multiple values separated by a space
- Datatype matters

```cpp
int hours, mins;
char colon;
cout << “Enter the time in 24hr format (hh:mm) : ”;
cin >> hours >> colon >> mins;

int age, height;
cout << “Enter your age followed by your height in inches: ”;
cin >> age >> height;
```
General Program Style

• Code must be easy to read and modify
  – Must have adequate whitespace
• Use lots of comments
  – // for one line comments
  – /* ... */ for multi line comments
• Variable names must be meaningful

CS1129 Style Guide

• Indent by 2 spaces
• Reserve /* */ for debugging
• Leave a space after //
• Declare all variables at top of block
• Use Java style braces
• Variable naming
  – ALL_CAPS for constants
  – lowerToUpperCase for variables