Lecture 1

Chapter 1.2
Topics

• Variables
• Expressions
• Assignment Statements
• Assignment Compatibility
• Type Casting
• Escape Sequences
• Increment and Decrement
But first some C++ Terminology

• Functions – procedures, methods, functions, or subprograms

• Example
  – int main()
Variables

• The name given to a variable is called an **identifier**

• Identifier start with either a letter or and underscore

• These are all valid identifiers
  – x xi x1 n_1 ABC123 sum count gpa
  – But you have no idea what x xi x1 n_1 ABC123 are
    • So make sure to make the identifier meaningful
  – Identifiers are case sensitive
Variables

• Convention dictates
  – To always start with a lower case
  – To start all word boundaries with an uppercase letter
  – So the sumOfThatThingYouAreTalkingAbout
  – Is more clear sumOfThatThingYouAreTalkingAbout
  – Identifiers with all upper case letters are usually reserved for constants
    • The value of constants cannot be changed once it is set
      – const int BIRTHYEAR = 1987;
Variables

• Some identifiers are **keywords** meaning that their meaning is predefined and cannot be changed
  – if
  – else
  – float
  – int
Variables

• Declaring a variable
  – This tells the compiler what kind of data you are storing in the variable
  – Examples
    • int numberOfCookies;
    • float PI;
## Variables

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Memory Used</th>
<th>Size Range</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>short int</td>
<td>2 bytes</td>
<td>-32,768 to 32,767</td>
<td>NA</td>
</tr>
<tr>
<td>int</td>
<td>4 bytes</td>
<td>-2,147,483,648 to 2,147,483,647</td>
<td>NA</td>
</tr>
<tr>
<td>float</td>
<td>4 bytes</td>
<td>Approx $10^{-38}$ to $10^{38}$</td>
<td>7 digits</td>
</tr>
<tr>
<td>double</td>
<td>8 bytes</td>
<td>Approx $10^{-308}$ to $10^{308}$</td>
<td>15 digits</td>
</tr>
<tr>
<td>long double</td>
<td>10 bytes</td>
<td>Approx $10^{-4932}$ to $10^{4932}$</td>
<td>19 digits</td>
</tr>
<tr>
<td>char</td>
<td>1 byte</td>
<td>All ASCII characters</td>
<td>NA</td>
</tr>
<tr>
<td>bool</td>
<td>1 byte</td>
<td>true,false</td>
<td>NA</td>
</tr>
</tbody>
</table>
Variables

• There are also unsigned versions integers such as
  – unsigned short
  – unsigned int
  – unsigned long
  – These can only be non negative
  – These have a larger positive range
Assignment Statements

• int numOfCakesLeft still means nothing unless you assign it a value
  – numOfCakesLefts = 3;
  – numOfCakesLefts = 2 + 2;
  – numOfCakesLefts = numOfCakesLefts – 1;
Assignment Statements

• Also
  – numOfCakesLefts = ( m = 2);
  – numOfCakesLefts = m = 2;
  – But this is **bad convention**
  – This could easily be a typo for
    numOfCakesLefts = m + 2;
Assignment Statements

• More ways to assign
  – int puppies = 2, kitties = 3;
  – int puppies(2), kittens(2);
## Assignment Statements

<table>
<thead>
<tr>
<th>Example</th>
<th>Equivalent to</th>
</tr>
</thead>
<tbody>
<tr>
<td>count += 2;</td>
<td>count = count + 2;</td>
</tr>
<tr>
<td>count -= 2;</td>
<td>count = count - 2;</td>
</tr>
<tr>
<td>numBunnies *= 2;</td>
<td>numBunnies = numBunnies * 2;</td>
</tr>
<tr>
<td>velocity /= 2;</td>
<td>velocity = velocity / 2;</td>
</tr>
</tbody>
</table>

But makes sure it has a value to begin with
Assignment Compatibility

• Mixing types can be dangerous
  – int cake = 45.9999;
    • Since cake is an int it will most likely be set to 45
    • But don’t trust it
  – double life = 100;
    • Assigning an int to a double or float is okay
    • The compiler will automatically convert it
  – Integer and Booleans can be mixed but it is considered poor style
Type Casting

• If int m has to be used in an operation with float t then it should be type cast
  – int answerToTest = t / static_cast<float>(m);
Escape sequences

• Combination of two characters that do not mean what they appear to mean in the string
  – \n means new line
  – \t means tab
  – \’ means add a single quote
Increment and Decrement

- n++ has the effect of using the value of the variable \( n \) \textbf{then} incrementing \( n \) by 1
- ++n has the effect of incrementing \( n \) by 1 \textbf{then} using the value