CS 102
Introduction to Programming Using C++

Chapter 2

Fundamental Data Types

Variables

- A variable is a name for a place in storage
- There are various types of data
- We will start with numeric data
- The two main data types for numeric data are
  - int for integer data
  - double for data with decimal points

Naming Variables

- C++ has rules for naming variables
  - Variable names must start with a letter
  - Variable names can be made up of letters, numbers, and the underscore character (_)
  - Variable names should look like they are just one word
    - If you want a name with more than one word, use underscores between the words
    - For example: number_of_attendees
    - For not-an-example: number of attendees
      - C++ will think this is three variables!

More Ideas about Naming Variables

- Variable names are case-sensitive
  - No surprise here; everything in C++ is
- You cannot use reserved words as variable names
  - For example, you could not use return as a variable name
- You should use a name that describes what the variable contains

What Can You do with Variables?

- The subtitle of this slide is
  - Why Do I Care About Variables Anyway?
- Variables in programming are similar to variables in math
- You calculate something and store the result in a variable
- Examples

```
number_of_people = number_of_adults +
                   number_of_children;
```

Math Notation vs. C++

- In math the equation

  $x = x + 1$
- has no solution
- Think of this for whole numbers
  - No number can equal the next number
- In C++, this is perfectly legal
- It means
  - Calculate the value on the right side
  - Store that value into the variable on the left
- We use this statement (or a variation of it) often
  - It's how we count things

Creating Your Own Variables

- Before using a variable, you must create it
  - We also say you define or declare a variable
- Creating a variable is simple
  - You need to decide on its type and its name
  - You choose a name that represents what it is
  - Two examples
    - int number_of_employees;
    - double price_per_gallon;

Starting Values

- In addition to giving a value to a variable in a program, you can give it a starting value
- You can do this when you declare it

```
int  i = 10;
```

– We call this initializing the variable

Constants

- A constant is a variable whose value doesn't change
- We use constants so we can remove numbers from inside of programs
- When we name constants, we use capital letters
- For example

```
const double PRICE_OF_COFFEE = 2.35;
```

Readability

- Readability means understandability
- A program with high readability is easily understood by other programmers
- Anything you can do to increase the readability of a program is good
- Inserting blank lines in a program is one way to increase readability
- Another way to increase readability is to use comments

Putting Comments in a Program

- There are two types of comments
  - Single line comments start with //
    - They may be used anywhere in a line
    - The rest of the line is ignored by the compiler
  - Comments that take several lines may be inside /* and */
  - For example

```
/* This code calculates the polynomial that comes closest
   to going through several points. It uses the Lagrange
   interpolating polynomial. The input for this function is
   … and the output is … */
```

Comments

- You should use enough comments to explain what's going on in your programs
- You should explain every function except main()
- You should also explain tricky or complex code with comments
- If you think you have enough comments, ask someone else to look at your program
  - Getting a second opinion is good!

Another Example

- Let's check volume2.cpp (p. 50)

- Notes:

1. Notice the comments
2. Notice the constant
3. Where should the variable declarations go?
4. Line 29. cout << fixed << setprecision(2);

   fixed << setprecision (2) means to print two digits after the decimal point
5. Line 2. #include <iomanip>

   The compiler can't find fixed, setprecision without this
6. Line 12. cin >> pack_price;

   cin is used to read input from the keyboard

Floating Point Variables

- C++ has several different types of variables
  - A variable stores something
  - You might want to store a count, your name, or a price
- A type that can hold decimal data is floating point
  - Floating point data comes in float and double
  - float holds only 6-7 digits of accuracy
  - double holds more digits (14-15)
  - double is more common than float
- C++ has several other types of data

Floating Point Data

- Floating point data is often imprecise
  - For example, 0.2 as a decimal number is 0.001100110011… as a binary number
    - Binary numbers are numbers with 0s and 1s
    - All computer data is stored as binary numbers
- The stored value will be very close to the actual value, but may not be exactly equal
  - For this reason, we don't usually ask if two floating point numbers are equal

Arithmetic

- Here are more examples of using the assignment statement
  - That's the statement: variable = calculation
- total_cost = subtotal + tax;
- area_of_rectangle = length * width;
- number_of_boxes = number_of_items
  / box_size; (Hmmm…)
- average = (value_1*frequency_1+ value_2*frequency_2)
  / (frequency_1 + frequency_2);

The Operators

- The standard math operators are
- + for add
- - for subtract
- * for multiply
- / for divide

- Special operators
- ++ increment (add 1)

  x++; is the same as x = x+ 1;
- -- decrement (subtract 1)
- x- -; is the same as x = x – 1;

More Operators and Integer Math

- If you have these statements

```
int number_of_boxes = 19 / 10;
cout << number_of_boxes;
```

- what prints is 1
- Huh?
- Integer arithmetic gives integer answers
  - Fractions are rounded down, not rounded off
- There is another integer operator
  - It's %
- % indicates the remainder after division
  - What is 11 % 4?

More Integer Arithmetic: Rounding off Numbers

- Suppose you have these statements

```
double price = 2.85;
int dollars = price;
```

- According to integer math, dollars will have the value 2
  - Again, it rounds down, not off
- A roundoff trick is to add 0.5

```
int dollars = price + 0.5;
```

- will work correctly

More Math: Powers and Roots

- To calculate x^{10}, use pow (x, 10)
- To calculate $\sqrt{}x$, use sqrt (x)
- To solve $ax^2 + bx + c = 0$ using the quadratic formula, use

```
x_1 = (–b + sqrt (b*b – 4*a*c)) / (2 * a);
x_2 = (–b – sqrt (b*b – 4*a*c)) / (2 * a);
```

  - You could also use pow (b, 2) for b^2

    b * b   is better

  - You could also use pow (?, 0.5) for sqrt (?)

    sqrt   is better

A Common Mistake

- Consider this program fragment

```
int main ()
{
    int score_1, score_2;
    cout << "Type in your first test score: ";
    cin >> score_1;
    cout << endl << "Type in your second test score: ";
    cin >> score_2;
    double average = ((double) score_1 + score_2)  /  2;
    cout << "The average is " << fixed << setprecision (2) << average;
}
```

- There is a logic error; what is it?

Calculations in C++

- When C++ calculates, it follows the normal math order of operations
    - So 2 + 5 * 4 = 22
- It calculates piece by piece, just like in algebra
- For each calculation within an expression, it promotes all variables to the most precise type
    - double is more precise than float
    - float is more precise than integer
- For example

```
  10.0 + 4 / 5
= 10.0 +   0
= 10.0
```

Casting

- You are in full control of what your program does
- You can force data to be any type in a calculation
- The way to do this is called casting
- Example

```
unit_price = (double) 10 / 3;
```

- We just forced C++ to treat 10 as if it was of type double
- The calculation worked as we had hoped

Writing Programs

- Understanding the problem
  - This often generates test data
- The first step in the programming process is to understand the problem
- One good way to understand the problem is to try cases
- Suppose we want to write a payroll program

The Task

- We want to calculate the weekly pay of an employee
- We will be given the number of hours the employee works and the hourly rate of pay
- We have to calculate the pay according to these rules
  - All employees work at least 40 hours
  - All employees must be paid time-and-a-half for the overtime (hours over 40)
    - This means we pay 1.5 times the normal rate, but only for the overtime

Some Cases-Part 1

- For this example, we will assume employees earn $15 per hour
- Case 1
  - The employee works 45 hours
  - Then the regular rate of pay is used for the first 40 hours
    - This is $15*40 = $600
  - The employee also earns $15*5*1.5 for overtime
    - This is $112.50
  - The total pay is $600 + $112.50, which is $712.50

Some Cases-Part 2

- Case 2
  - The employee works 50 hours
  - Then the regular rate of pay is used for the first 40 hours
    - This is $15*40 = $600
  - The employee also earns $15*10*1.5 for overtime
    - This is $225
  - The total pay is $600 + $225, which is $825

The Value of Studying Cases

- Since we worked through some cases, we now know how to write the program
  - The total pay is
    - $600 (for regular hours) and $15*n*1.5
      - where n is the number of hours over 40
  - Or, if h = total number of hours, the pay is
    - $600 + $15*(h-40)*1.5
- We also have some data to test if our program works

Strings

- A string is another data type
- To use strings, you need

```
#include <string>
```

- To store a value into a string, use quotation marks

```
string name = “Harvey”;
```

- Concatenation
  - This means appending one string onto the end of another
  - You do this using +

```
full_name = first_name + “ “ + last_name;
```

More String Ideas

- You can read a string with cin

  ```
  cin >> street_address;
  ```

  - This is just like reading anything
- However, this reads only one word
  - You must use multiple cins to read multiple words

Functions

- A function is a small program that does a task
  - There are two types
    - One type sends back an answer
    - The other type performs a task
- So far, we know main()
- Since its type is int, it needs to return an integer answer

String Functions

- C++ has several built-in string functions
    - int name_len = name.length ();
    - string new_string = old_string.substr (3, 2);
        - This creates a new string starting at character 3 in old_string and having length 2
        - Character 3 is actually the fourth character in the string
            - We start counting at 0!
- Some string functions are summarized on p. 60

Homework

- These problems are from p. 63+
- Do R2.1, 2, 3, 5, 6, 7, 8, 9, 13, 15, 16
- These problems are due

- Again, in the upper right corner, after your name, add
- The chapter the problems are from
- The page number the problems are from
- A list of the problems themselves

Our Second Programming Assignment

- Write a program that solves P2.4, 7, or 8
- All of these programs involve calculations
- Be sure to follow the directions closely

- Again, start your program with these four+ lines

```
// Your name
// The date you wrote the program
// Which lab this is (Lab 2 in this case)
// A brief description of the program
```

Another Program

- For this chapter, you have to write two programs
- The second program should be one of P2.17, 18, and 20
- All of these programs involve manipulating strings
- This program is due

Academic Honesty

- This is the academic honesty policy for our class:
  - All programs must be written by you alone
- Exceptions
  - Since this is a class, you may base your programs on programs in the textbook
  - You can get help from me
- Any program not written completely by you will get a score of 0

Questions?

- Are there any questions?