Announcements

- Lab will be in LL12
- Lectures in LL15
- Last time we completed up to section 2.4 in the book

Purpose of Datatypes

- Different ones allow compiler to know how to represent value
- Different datatypes can use different operations
- The integer 2 is different from 2.0 and the character 2 (all stored differently)

Declarations

- Declarations are at the beginning of a program
- They list the variables used
- Format:

```
datatype identifier;
```

Constants

- Associate names with memory locations whose values never change
- Format:
  - `const datatype identifier = value;`
- Contrast with variables whose values are always changing

```
//program:  silly.cpp
//author:  Shereen Khoja
void main()
{
      const int pi = 3.14;
      float num;
      int i,j;

      num = e2;
      i = 4,000;
      ch = “b”; j = i;
      pi = 5;
}
```

Executable Statements

- Assignment statements
  - Store a value or computed result in a variable
  - `kms = miles * KM_PER_MILE;`

- Input/output operations

Input/Output Operations

- Output operations allow you to write information to a computer monitor screen
- Input operations allow you to read information in from keyboard
- Other possible sources of I/O: files, printers, etc
  - We'll talk about those later
- Stream: sequence of characters
- Must have: **`#include<iostream>`**

Input

- Input operator (extraction operator): **>>**
- Gets input from some device/file
- Standard input (from keyboard): **cin**
- Skips spaces before data item
- Continues as long as data read in is of that data type
- Format:

```
cin >> miles;
cin >> letter1 >> letter2 >> lastname;
cin >> num1 >> num2;
```

Output

- Output operator (insertion operator): **<<**
- Displays output values
- Standard output (monitor screen): **cout**
- Return character: **endl**
- Examples:

```
cout << miles;
cout << “The distance in kilometers is ”;
cout << kms << endl;
cout << “Hello “ << letter1 << “. “ << letter2;
cout << “. “ << lastname << endl;
```

The **`return`** Statement

- Transfers control from your program to the operating system

- Form:

```
    return 0;
```

- Returning 0 from the function main indicates to the operating system that your program executed without error

Caveats

- Make sure data types match input
  - Example: if reading in prices, use float

- Do not put carriage returns in the middle of output strings

```
cout << "The number of kilometers
     is" << kms;
```

Programs

- Write a program that reads in the user's first and last names and prints out a greeting message

- Write a program that reads in last week's and this week's gas prices and prints out the difference

What’s the output?

```
cout << “Enter two numbers: “;
cin >> a >> b;
a = a + 5.0;
b = 3.0 * b;
cout << “a = “ << a << endl;
cout << “b = “ << b << endl;
```

- Assume 5.0 and 7.0 are entered for a & b

What’s the output?

```
cout << “My name is: ”;
cout << “Doe, Jane.” << endl;
cout << “I live in “;
cout << “Ann Arbor, MI “;
cout << “and my zip code is “
  << 48109 << “. “ << endl;
```

- How would we add a blank line between sentences?

What is the Output?

- Assume x = 2, y = 3

- `cout << x;`
- `cout << x + x;`
- `cout << “x=“;`
- `cout << x + y << “ = “ << y + x;`
- `z = x + y;`
- `cin >> x >> y;`
- `// cout << “x + y = “ << x + y;`
- `cout << “\n”;`

General Form of a C++ Program

```
// Programmer: John Doe
// Instructor: Shereen Khoja
// Date: Aug 30, 2003

// Purpose: converts distances from miles to
//          kilometers

compiler directives
using namespace std;

int main()
{
  declaration statements
  executable statements
}
```

Arithmetic Expressions

- Arithmetic expressions manipulate numeric data
- We’ve seen simple ones
- We’ll learn all the rules for using expressions

Arithmetic Operators

+	addition
-	subtraction
*	multiplication
/	division
%	remainder (modulus)

Division

- The division operator can be used with both integers and floats

- If the operands are both **`floats`**, the result is a **`float`**
  - Example: `7.0/2.0 is 3.5`

- If the operands are both **`ints`**, the result is an **`int`**
  - Example: `7/2 is 3`

- If mixed, the **`int`** operand is converted to a **`float`** and the result is a **`float`**
  - Example: `5/2.5 is 2.0`

Division Continued

- Divisor (second operand) cannot be 0
- Division with negative integers may or may not be allowed

Modulus

- % returns the integer remainder of integer division
- Both operands must be integers
- If second operand is negative, results will vary from system to system
- The value of m%n must be less than divisor n

- Examples

3%5 =
4%5 =
5%5 =
6%5 =
7%5 =
15%-7 system dependent

5%3 =
5%4 =
15%5 =
15%6 =
8%0 undefined

Assignment Statements and Expressions

- When assignment statement is executed, expression is evaluated and result is assigned to variable on left.

- Example: if **`a`** is a **`float`**
  - **`a = 10;`**
  - **`a = 10/3;`**

- What happens when types are mixed?

Mixed-type assignments

- a = 10/3;
- n = 10.5 + 3.7;
- **a** is a **float** and **n** is an **int**

Unary and Binary Operators

- Unary: One operand
  - Unary + and -
  - Example: x = -y;   y = +x;

- Binary: Two operands
  - Example: x = y+x;

Expressions with Multiple Operators

- Example:

```
x = 5 + 3 * 2 - 1;
```

- What’s the value of x?
- There are rules for the order of evaluation so every computer will calculate the same expression the same way every time

Order of Evaluation

- Anything in parentheses is evaluated first.
  - Innermost first.
  - Any with the same level are evaluated left to right.

- Operator precedence
  - Unary + and -
  - Operators *,/,%
  - Binary +, -

- Binary operators evaluated left to right and unary right to left.

Example

- Put in parentheses to indicate order of evaluation
- x * y * z + a / b - c * d

Program

- Design and write a program to calculate how much money your little sister has in nickels and pennies.