Lecture 3

Input and Output

Review from last week

- Variable - place to store data in memory
  - identified by a name – should be meaningful
  - Has a type-
    - int
    - double
    - char
    - bool
  - Has a value – may be garbage
  - change value using assignment statements:

    Variable = Expression

Constants - Use a constant instead of a number

- Easier to change a single constant declaration than to find all uses of the constant in your program.
- Easier to read and understand the program
- Remember to use **const** in the declaration to prevent accidental modification of the value

Arithmetic Operations +,-,*,/

- Follow Rules of Precedence – Appendix 2
- Integer division – dividing by two integers will result in an integer even if there is a remainder, regardless of the variable's type.
- If one of the variables in an expression is double, the result will be double

❖Arithmetic with + = and * behaves as expected for both integer and floating point types.

❖Arithmetic for / behaves as expected for floating point numbers.

❖Arithmetic for / behaves in a somewhat surprising way for integer types (short, int, long)

❖Division for integers is **TRUNCATING** - it discards the fractional part.

❖Division /, and modulus % are complementary operations. Mod, or modulus, %, works ONLY for integer types.

```
   4        14 / 3 = 3   NOT 4.66
3 | 14
  -12
   2        14 % 3 = 2
```

Programming Style

- Grouping things that belong together
- Indenting
- Leaving a blank line
- Comments

  // comment follows until end of line

  /* multi-line comments must end with another */
- Header – use the template for your "starter" program for all lab assignments

Program Template

- Header
- Include and Using directives
- Variable and Constant Declarations
- Output Identification

Hints

- Read the Problem!
- Listen in class!
- What are the requirements?
- Ask Questions!

Streams and Basic I/O

- stream of characters - bytes
- input streams and output streams
- cin and cout are streams
- cin is connected to the keyboard
- cout is connected to the console window
- We will later look at streams that go to a file

Input and Output

- cout is the output stream, read See-Out.. It is attached to the monitor screen. << is the insertion operator
- cin is the input stream, read see-in, and is attached to the keyboard. >> is the extraction operator.
- cin and cout are defined in the **iostream** library.

```
cout <<“Press return after entering a number.\n”;
cout << “Enter the number of pods:\n”;
cin >> number_of_pods;
cout << “Enter the number of peas in a pod.\n”;
cin >> peas_per_pod;
```

- The first two lines sends a request to the user.
- The third gets an integer value (Number_of_pods) from the user.

COUT

Strings of text and values of variables may be output to the screen:

```
cout << num_of_bars << " candy bars\n";
```

OR

```
cout << num_of_bars;
cout << " candy bars\n";
```

OR

```
cout << "num_of_bars << " candy bars" << endl;
```

Well spaced output

Sample Dialogue

```
Press return after entering a number.
Enter the number of pods:
10
Enter the number of peas in a pod:
9
If you have 10 pea pods
and 9 peas in each pod, then
you have 90 peas in all the pods.
```

Poorly spaced output:

```
Press return after entering a number.
Enter the number of pods:
10
Enter the number of peas in a pod:
6
If you have 10pea pods
and 6peas in each pod, then
you have 60 peas in all the pods.
Press any key to continue
```

CIN

Getting input from the keyboard:

```
cout << “Enter the number of bars in a
  package\n”;

cin >> num_of_bars;
```

Escape Sequences

- New line \n (like endl)
- horizontal tab \t
- alert \a
- backslash \\
- double quote \”

The \ (backslash) preceding a character tells the compiler that the next character does not have the same meaning as the character by itself.

An escape sequence is two characters with no space between them.

\\ is a real backslash character, not the escape character, a backslash that does not have the property of changing the meaning of the next character.

Directives

```
#include <iostream>
using namespace std;
```

- Makes the library “iostream” available.
- iostream includes the definitions cin and cout.
- Made part of your program in the linking process.

Display 2.1 page 38

```
#include <iostream>
using namespace std;

int main( )
{
    int number_of_bars;                          //variable declaration
    double one_weight, total_weight;

    cout << "Enter the number of candy bars in a package\n";
    cout << "and the weight in ounces of one candy bar.\n";
    cout << "Then press return.\n";
    cin >> number_of_bars;
    cin >> one_weight;

    total_weight = one_weight * number_of_bars;

    cout << number_of_bars << " candy bars\n";
    cout << one_weight << " ounces each\n";
    cout << "Total weight is " << total_weight << " ounces.\n";
```

```
    cout << "Try another brand.\n";
    cout << "Enter the number of candy bars in a package\n";
    cout << "and the weight in ounces of one candy bar.\n";
    cout << "Then press return.\n";
    cin >> number_of_bars;
    cin >> one_weight;

    total_weight = one_weight * number_of_bars;

    cout << number_of_bars << " candy bars\n";
    cout << one_weight << " ounces each\n";
    cout << "Total weight is " << total_weight << " ounces.\n";

    cout << "Perhaps an apple would be healthier.\n";

    return 0;
}
```

Enter the number of candy bars in a package
and the weight in ounces of one candy bar
Then press return.
11 2.1
11 candy bars
2.1 ounces each
Total weight is 23.1 ounces.
Try another brand.
Enter the number of candy bars in a package
and the weight in ounces of one candy bar
Then press return.
12 1.8
12 candy bars
1.8 ounces each
Total weight is 21.6 ounces.
Perhaps an apple would be healthier.

Take a Break!!!

Formatting Decimals
(page 51 & 216)

The “magic” formula:

```
cout.setf(ios::fixed);          //fixed decimal point
cout.setf(ios::showpoint);      // pad with zeros
cout.precision(2);              //round to 2 decimal places
```

Member functions

- setf() (set flags) - unsetf() (unset flags)
  - for floating point numbers:
    ios::fixed, (always show the number as a fixed point number)
    ios::scientific, (always show the number in scientific notation)
    ios::showpoint, (always show the decimal point)
- width(n) (set the width of the following field to n characters)
- More flags for any type of value:
  ios::left, ios::right, (left or right justify the value in the specified field)
- precision(), (set the number of places to show for a floating point number)

Formatting Flags

Flag	Purpose
`ios::fixed`	Display floating point numbers in fixed format
`ios::scientific`	Display floating point numbers in scientific format
`ios::showpoint`	Always show the decimal point of a floating point number
`ios::left`	Display the data left justified in the field
`ios::right`	Display the data right justified in the field

Manipulators

- setiosflags() same as setf()
- resetiosflags() same as unsetf()
- setw() same as width()
- setprecision() same as precision()

Formatting Commands

Purpose	Function	Manipulator
Set flag	setf()	setiosflags()
Unset flag	unsetf()	resetiosflags()
Set field width	width()	setw()
Set number of digits	precision()	setprecision()

Reading

Section 2.4 pages 65-74

Sections 7.1 and 7.2 pages 335 – 364