CS150 Intro to CS I

Fall 2015

Chapter 3
Formatting Output

- Reading: Chapter 3 (3.7 pp. 108-117)
- Good Problems to Work: pp. 117-118[3.17, 3.19]

Lab Review

- Constants
- char variables
- if statements
- Checking that the user has selected a valid menu choice

const Declarations

- Constant declaration

```
const double PI = 3.14;

const int MAX_SCORE = 100;
```

- Constant declarations are fixed and cannot be changed
- By convention, constants are always UPPERCASE
- Separate words using underscore _

Formatting Output

- How can we force output to look a particular way?
  1. Precision of numbers
  2. Spacing around the output

```
Here are some floating point numbers:
       72.0
       72.00
       72.000
Here is a table of data:
         4   cat     15
       100     6    2.1
```

Precision

```
const double PI = 3.141592653589793;
cout << PI << endl; // default output 3.14159
```

- Floating-point numbers can be rounded to a number of significant digits (precision)

```
cout << setprecision (3) << PI; // output 3.14
```

Precision

- Precision can also be used to set the number of digits after the decimal point
- What is the output?

```
const double PI = 3.141592653589793;

cout << fixed << setprecision (2) << PI;
```

Precision of numbers

```
#include <iostream>
#include <iomanip> //New Library!

using namespace std;

int main()
{
  const double PI = 3.141592653589793;

  cout << PI << endl; // default output
  cout << fixed << setprecision (4) << PI << endl;
  cout << fixed << setprecision (3) << PI << endl;
  cout << fixed << setprecision (2) << PI << endl;
  cout << fixed << setprecision (1) << PI << endl;

  return EXIT_SUCCESS;
}
```

Precision

- Precision and fixed are sticky (i.e they remain in effect until changed)
- What is the output?

```
const double PI = 3.141592653589793;
cout << fixed << setprecision (4) << PI << endl;
cout << setprecision (2) << PI << endl;
cout << PI << endl;
```

Output with Spacing

```
#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

int main()
{
  string name = "cs150";
  int integer = 42;

  cout << setw (6) << name << setw (6) << integer << endl;
  cout << setw (4) << integer << endl;

  return EXIT_SUCCESS;
}
```

setw

- setw is not sticky
  - you must specify setw every time you want a specific field width specified
- What is the output?

```
int integer = 42;
cout << setw (6) << integer << integer << endl;
```

Problem

- Write a program segment that allows the user the ability to input two integer values. Display both integer values as shown below, always displaying the smaller number first.

```
Please enter two numbers: 100 9
The numbers are:
                    9
                  100
```

setw justify

- By default, setw justifies on the right
- To justify on the left, precede setw by **`left:`**

```
int integer = 42;
cout << left << setw (6) << integer << endl;
```

What is the output?

```
int integer = 42;
cout << left << setw (6) << integer << integer <<
endl;
```

How would we output the following to line it all up correctly?