Exponents & Output

page 85-87 & Section 3.8
Advanced Output  Section 3.8

• How can we force output to look a particular way?
  o Precision of numbers
  o Spacing around 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
Spacing

• How can we output data in a table?

```cpp
string name = "cs150";

int integer = 42;
cout << setw(6) << name << endl;
cout << setw(6) << integer;
```
#include <iostream>
#include <iomanip>  //New Library!
#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 0;
}
Setw

- Setw is not *sticky*
  - you must specify it every time

```cpp
const double PI = 3.141592653589793;
int integer = 42;
cout << setw(6) << integer << integer << endl;
cout << PI << endl;
//output?
```
Practice

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

Please enter two numbers: 100 9
The numbers are:

9
100
const double PI = 3.141592653589793;
cout << PI << endl; // default output

• What does this output?

• Precision

cout << setprecision(2) << PI;

Output:
Spreading around output

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

using namespace std;

int main()
{
    const double PI = 3.141592653589793;
    string name = "cs150";
    int integer = 42;

    cout << setw(6) << name << setw(6) << integer << endl;
    cout << setw(6) << fixed << setprecision(3) << PI;
    cout << setw(4) << integer << endl;

    return 0;
}
```
Precision

- Precision can also be used to set the number of digits after the decimal point

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

- Output:
#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 0;
}

What if we had an int instead of a double?
Precision

- Precision and fixed are **sticky**
  - remains in effect until changed

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

// Output?
double

- a **double** has a range of:
  - ±1.7E-308 to ±1.7E308
  - however, only tracks 16 significant digits

```cpp
double bignumber = 1234567891.123456789;
cout << fixed << setprecision(20);
cout << bignumber << endl;
bignumber = 9234567891.123456789;
cout << bignumber << endl;
```

- Output:
Practice

• Using the variables below, create the output shown:

```cpp
const double PI = 3.141592653589793;
string name = "cs150";
string animal = "cat";
string cover = "hat";
int integer = 42;
```

```
cat • 3.1416
hat • cs150
42 • 42 • 42 • 42
3.14159265 • 3.1
```

A • represents a blank space
Exponents (page 85-87)

- The exponent operator was missing from the list! \( x^2 \) \( y^n \)
- C++ does not provide an exponent operator as part of the language
- Use \texttt{pow()}\ in\ the\ \texttt{cmath}\ library

\begin{verbatim}
#include <cmath>

double area;

area = pow(4, 2); // area = 4^2
\end{verbatim}
• pow() is not an operator
  ◦ it is a function
  ◦ like main()
  ◦ double pow(double x, double y)
  ◦ it takes as arguments two doubles
     * x and y
  ◦ it produces a double
Practice using exponents!

// Calculate the area of a square

double lengthOfSide = 4.9;

// Calculate the volume of a cube