Formatting Output

- By default, consecutive integer, floating-point, and string values are output with no spaces between them.
- Manipulators are used to control the horizontal spacing of the output. (`endl` is one we've already seen)
- Manipulators we'll use now

`endl`	first three defined in **`iostream`**
`fixed`	
`showpoint`	
`setw`	last two defined in **`iomanip`**
`setprecision`	

`setw`

- `setw` (set width) lets us control how many character positions the next data item should occupy when it is output. (used for numbers and strings, not **`char`s**).
- The argument to `setw` is an integer fieldwidth specification, the data item is right-justified within the fieldwidth.
- If you don't specify enough characters, the minimum number of characters is used anyway.

`setw` Examples

Statement (ans = 33, num = 7132) Output (means blank)

```
cout << setw(4) << ans                    33
```


`setw` Examples

Statement (ans = 33, num = 7132) Output (means blank))

```
cout << setw(4) << ans << setw(5)        33 7132
     << num
```


`setw` Examples

Statement (ans = 33, num = 7132) Output (means blank)

```
cout << setw(4) << ans << setw(5)        33 7132  Hi
     << num << setw(4) << "Hi";
```


`setw` Examples

Statement (ans = 33, num = 7132) Output (means blank)

```
cout << setw(4) << ans << setw(5)        33 7132  Hi
     << num << setw(4) << "Hi";
cout << setw(2) << ans                   33
```

setw Examples

Statement (ans = 33, num = 7132) Output (means blank)

```
cout << setw(4) << ans << setw(5)        33 7132  Hi
     << num << setw(4) << "Hi";
cout << setw(2) << ans << setw(4)      337132
     << num
```


setw Examples

Statement (ans = 33, num = 7132) Output (means blank)

```
cout << setw(4) << ans << setw(5)        33 7132  Hi
     << num << setw(4) << "Hi";
cout << setw(2) << ans << setw(4)      337132Hi
     << num << setw(2) << "Hi";
```


setw Examples

Statement (ans = 33, num = 7132) Output (means blank)

```
cout << setw(4) << ans << setw(5)        33 7132  Hi
     << num << setw(4) << "Hi";
cout << setw(2) << ans << setw(4)      337132Hi
     << num << setw(2) << "Hi";
cout << setw(6) << ans                       33
```


setw Examples

Statement (ans = 33, num = 7132) Output (means blank)

```
cout << setw(4) << ans << setw(5)        33 7132  Hi
     << num << setw(4) << "Hi";
cout << setw(2) << ans << setw(4)      337132Hi
     << num << setw(2) << "Hi";
cout << setw(6) << ans << setw(3)          33 Hi
     << "Hi"
```


setw Examples

Statement (ans = 33, num = 7132) Output (means blank)

```
cout << setw(4) << ans << setw(5)        33 7132  Hi
     << num << setw(4) << "Hi";
cout << setw(2) << ans << setw(4)      337132Hi
     << num << setw(2) << "Hi";
cout << setw(6) << ans << setw(3)          33 Hi 7132
     << "Hi" << setw(5) << num;
```


setw Examples

Statement (ans = 33, num = 7132) Output (means blank)

```
cout << setw(4) << ans << setw(5)        33 7132  Hi
     << num << setw(4) << "Hi";
cout << setw(2) << ans << setw(4)      337132Hi
     << num << setw(2) << "Hi";
cout << setw(6) << ans << setw(3)          33 Hi 7132
     << num << setw(5) << "Hi";
cout << setw(7) << "Hi" << setw(4)          Hi7132
     << num;
cout << setw(1) << ans 33
```

setw Examples

Statement (ans = 33, num = 7132) Output (means blank)

```
cout << setw(4) << ans << setw(5)        33 7132  Hi
     << num << setw(4) << "Hi";
cout << setw(2) << ans << setw(4)      337132Hi
     << num << setw(2) << "Hi";
cout << setw(6) << ans << setw(3)          33 Hi 7132
     << num << setw(5) << "Hi";
cout << setw(7) << "Hi" << setw(4)          Hi7132
     << num;
cout << setw(1) << ans << setw(5)      33 7132
     << num;
```


setw Examples

Statement (ans = 33, num = 7132) Output (means blank)

```
cout << setw(4) << ans << setw(5)        33 7132  Hi
     << num << setw(4) << "Hi";
cout << setw(2) << ans << setw(4)      337132Hi
     << num << setw(2) << "Hi";
cout << setw(6) << ans << setw(3)          33 Hi 7132
     << num << setw(5) << "Hi";
cout << setw(7) << "Hi" << setw(4)          Hi7132
     << num;
cout << setw(1) << ans << setw(5)      33 7132
     << num;
cout << "Hi"                           Hi
```


setw Examples

Statement (ans = 33, num = 7132) Output (means blank)

```
cout << setw(4) << ans << setw(5)        33 7132  Hi
     << num << setw(4) << "Hi";
cout << setw(2) << ans << setw(4)      337132Hi
     << num << setw(2) << "Hi";
cout << setw(6) << ans << setw(3)          33 Hi 7132
     << num << setw(5) << "Hi";
cout << setw(7) << "Hi" << setw(4)          Hi7132
     << num;
cout << setw(1) << ans << setw(5)      33 7132
     << num;
cout << "Hi" << setw(5) << ans         Hi   33
```


setw Examples

Statement (ans = 33, num = 7132) Output (means blank)

```
cout << setw(4) << ans << setw(5)        33 7132  Hi
     << num << setw(4) << "Hi";
cout << setw(2) << ans << setw(4)      337132Hi
     << num << setw(2) << "Hi";
cout << setw(6) << ans << setw(3)          33 Hi 7132
     << num << setw(5) << "Hi";
cout << setw(7) << "Hi" << setw(4)          Hi7132
     << num;
cout << setw(1) << ans << setw(5)      33 7132
     << num;
cout << "Hi" << setw(5) << ans         Hi   337132
     << num;
```


Manipulating Floating-Point Numbers

- `setw` also works with floating-point numbers (remember that a decimal point is a character)
- If you don't want numbers to appear in scientific notation, use `fixed`.
- If you want whole numbers printed with a .0 appended, use `showpoint`.
- If you want to control the number of decimal places that are displayed, use `setprecision`(n), where n is the number of decimal places desired. Unlike `setw`, `setprecision` stays in effect until you explicitly change it.

Floating-Point Output Examples

Value of x	Statement	Output (means blank)

```
     << fixed;
cout    setw
        setprecision(2) << x;
```

Floating-Point Output Examples

Value of x	Statement	Output (means blank)

Value		
	`cout << fixed;`	
310.0	`cout << setw(10) << setprecision(2) << x;`	310.00
310.00	`cout << setw(10) << setprecision(5) << x;`	310.00000
310.0	`cout << setw(7) << setprecision(5) << x;`	310.00000

Floating-Point Output Examples

of x	Statement	Output (means blank)

Value		
	`cout << fixed;`	
310.0	`cout << setw(10) << setprecision(2) << x;`	310.00
310.00	`cout << setw(10) << setprecision(5) << x;`	310.00000
310.0	`cout << setw(7) << setprecision(5) << x;`	310.00000
4.827	`cout << setw(6) << setprecision(2) << x;`	4.83
4.827	`cout << setw(6) << setprecision(1) << x;`	4.8

Bad Style Example

Bad Style Example Continued

```
const float PRICE = 150000.0; // Selling price less land
int main(){ float grossFootage; // Total square footage
    float livingFootage;        // Living area
float costPerFoot;  // Cost/foot of living area
    cout << fixed << showpoint;  // Set up floating pt.
 //   output format

grossFootage = LENGTH * WIDTH * STORIES;livingFootage =
grossFootage - NON_LIVING_SPACE; costPerFoot = PRICE /
livingFootage; cout << "Cost per square foot is "
<< setw(6) << setprecision(2) << costPerFoot << endl;
return 0;
}
```

Good Style Example

```
                                                                **
// HouseCost
// This program computes the cost per square foot of

// the house, the number of stories, the size of the

//*************************************************************
#include  iostream>
 include <       >    // For     () and setprecision
using           std;

      float WIDTH = 30.0;
const       LENGTH = 40.0;              // Length of the house
const float STORIES = 2.5;              // Number of full stories
const float NON_LIVING_SPACE = 825.0;   // Garage, closets, etc.
const float PRICE = 150000.0;           // Selling price less land
```


Example Continued

```
    mai()
{
   f bat          ;        // Tot l square fo tage
   f bat           ;       // Living area
        costPerFoot;

   cout <       < sho point   // Set up floating pt.


   gross Bo tage
                = gross Bo tage  NON_LIVING_SPACE;
   costPerFoot = PRICE /           ;

       << "Cost per sq are fo t is"
        <    (6)<<setprecisi n      osPerFoot <    ;
   return 0;
}
```


Additional

- Now, we consider four functions that operate on strings: **size**, **substr**
- The **length** and **size** functions both return an unsigned integer value equal to the number of characters in the string
  The
  substring
  substring is found, it returns the position where the match begins, if not found, it returns a special number.
- The **substr** function returns a particular substring of a string.

and **size**

If
length looks like this:
.length()

- **length** function requires no arguments, but you must still use parentheses
  Also, returning, so the call

Example Using and **size**

Example:

```
string  firstName
        fullName;


firstName
cout    firstName.length()<<     ;  // Prints 9
         = firstName
     << fullName            end;
```


More About **string**

- **string** is a C++ class, which has data types and functions associated with it
- Other classes may have a **length** function. To get the **length** function associated with **string**, we use the dot operator with a
  .
- **string** has a data type associated with it
  string length
  Another class could
  .
- **string::size_type** specifies the **size_type** associated with **string**.

More About `string`

- Example

```
string firstName;
string::size_type len;
firstName = "Alexandra";
len = firstName.length();
```


The `find` Function

- Example Usage

```
string str1, str2;
str1.find("the")
str1.find(str2)
str1.find(str2 + "abc")
```

- If the string is not found, a special value (**`string::npos`**) is returned. The return value type is **`string::size_type`**.

Examples Using `find`

```
         01234567890123456789012345678901234567890
str1 = "Programming and Problem Solving";
str2 = "gram";
```

Function Call	Value Returned by Function
`str1.find("and")`	12

Examples Using `find`

```
         01234567890123456789012345678901234567890
str1 = "Programming and Problem Solving";
str2 = "gram";
```

Function Call	Value Returned by Function
`str1.find("and")`	12
`str1.find("Programming")`	0

Examples Using `find`

```
         01234567890123456789012345678901234567890
str1 = "Programming and Problem Solving";
str2 = "gram";
```

Function Call	Value Returned by Function
`str1.find("and")`	12
`str1.find("Programming")`	0
`str2.find("and")`	`string::npos`

Examples Using `find`

```
         01234567890123456789012345678901234567890
str1 = "Programming and Problem Solving";
str2 = "gram";
```

Function Call	Value Returned by Function
`str1.find("and")`	12
`str1.find("Programming")`	0
`str2.find("and")`	`string::npos`
`str1.find("Pro")`	0

Examples Using **find**

```
        01234567890123456789012345678901234567890
str1 = "Programming and Problem Solving";
str2 = "gram";
```

Function Call	Value Returned by Function
str1.find("and")	12
str1.find("Programming")	0
str2.find("and")	string::npos
str1.find("Pro")	0
str1.find("ro" + str2)	1

Examples Using **find**

```
        01234567890123456789012345678901234567890
str1 = "Programming and Problem Solving";
```

Function Call	Value Returned by Function
str1.find("and")	12
str1.find("Programming")	0
str2.find("and")	string::npos
str1.find("Pro")	0
str1.find("ro" + str2)	1

Examples Using **find**

```
        01234567890123456789012345678901234567890
str1 = "Programming and Problem Solving";
str2 = "gram";
```

Function Call	Value Returned by Function
str1.find("and")	12
str1.find("Programming")	0
str2.find("and")	string::npos
str1.find("Pro")	0
str1.find("ro" + str2)	1
str1.find("Pr" + str2)	string::npos
str1.find(' ')	11

The **substr** Function

- Sample function call:
  myString.substr(5, 20)
- Substr returns a string, the arguments to it are of type **string::size_type**.

Examples Using **substr**

```
          012345678901234567890123456789
myString = "Programming and Problem Solving";
```

Function Call	String Returned by Function
myString.substr(0, 7)	"Program"

Examples Using **substr**

```
          012345678901234567890123456789
myString = "Programming and Problem Solving";
```

Function Call	String Returned by Function
myString.substr(0, 7)	"Program"
myString.substr(7, 8)	"ming and"

Examples Using **substr**

```
                    01234567890123456789012345678901234567890
myString = "Programming and Problem Solving";
```

Function Call	String Returned by Function
myString.substr(0, 7)	"Program"
myString.substr(7, 8)	"ming and"
myString.substr(10, 0)	""

Examples Using **substr**

```
                    01234567890123456789012345678901234567890
myString = "Programming and Problem Solving";
```

Function Call	String Returned by Function
myString.substr(0, 7)	"Program"
myString.substr(7, 8)	"ming and"
myString.substr(10, 0)	""
myString.substr(24, 40)	"Solving"

Examples Using **substr**

```
                    01234567890123456789012345678901234567890
myString = "Programming and Problem Solving";
```

Function Call	String Returned by Function
myString.substr(0, 7)	"Program"
myString.substr(7, 8)	"ming and"
myString.substr(10, 0	""
myString.substr(24, 40)	"Solving"
myString.substr(40, 24)	Error

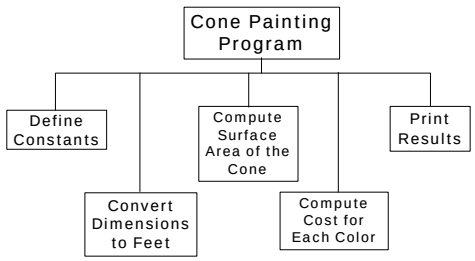
StringOps Program

```
//*************************************************************
// This program demonstrates several string operations
//*************************************************************
#include <iostream>
#include <string>     // For string type
using namespace std;
int main()
{
    string fullName;
    string name;
    string::size_type startPos;
    fullName = "Jonathan Alexander Peterson";
    startPos = fullName.find("Peterson");
    name = "Mr. " + fullName.substr(startPos, 8);
    cout << name << endl;
    return 0;
}
```


Problem-Solving Case Study

- Problem: You are asked to calculate the total cost of painting traffic cones in three different colors. The cone company uses the area painted to estimate the total cost.
- Output: The surface area of the cone in square feet, and the costs of painting the cone in the three different colors, all displayed in floating point form to three decimal places.
- Discussion: Cones are measured in inches. A typical cone is 30 inches high and 8 inches in diameter. Red, blue, and green paint cost 10, 15, and 18 cents per square foot, respectively. The non-base surface area of a cone is $\pi r(r^2 + h^2)^{1/2}$, where r is the radius of the cone and h is its height.

High-Level Algorithm



ConePaint Program

```
//****************************************************************
// This program computes the cost of painting traffic cones in
// each of three different colors, given the height and diameter
// of a cone in inches, and the cost per square foot of each of
// the paints
//****************************************************************
#include <iostream>
#include <iomanip>     // For setw() and setprecision()
#include <cmath>       // For sqrt()

using namespace std;

const float HT_IN_INCHES = 30.0;    // Height of a typical cone
const float DIAM_IN_INCHES = 8.0;   // Diameter of base of cone
const float INCHES_PER_FT = 12.0;   // Inches in 1 foot
```


More ConePaint Program

```
const float RED_PRICE = 0.10;       // Price per square foot
                                    //   of red paint
const float BLUE_PRICE = 0.15;      // Price per square foot
                                    //   of blue paint
const float GREEN_PRICE = 0.18;     // Price per square foot
                                    //   of green paint
const float PI = 3.14159265;        // Ratio of circumference
                                    //   to diameter

int main()
{
    float heightInFt;               // Height of the cone in feet
    float diamInFt;                 // Diameter of the cone in feet
    float radius;                   // Radius of the cone in feet
    float surfaceArea;              // Surface area in square feet
```


Yet More ConePaint

```
    float redCost;         // Cost to paint a cone red
    float blueCost;        // Cost to paint a cone blue
    float greenCost;       // Cost to paint a cone green

    cout << fixed << showpoint;      // Set up floating-pt.
                                     //   output format

    // Convert dimensions to feet
    heightInFt = HT_IN_INCHES / INCHES_PER_FT;
    diamInFt = DIAM_IN_INCHES / INCHES_PER_FT;
    radius = diamInFt / 2.0;

    // Compute surface area of the cone
    surfaceArea = PI * radius *
                  sqrt(radius*radius + heightInFt*heightInFt);
```


Cone Paint Program Concluded

```
    // Compute cost for each color
    redCost = surfaceArea * RED_PRICE;
    blueCost = surfaceArea * BLUE_PRICE;
    greenCost = surfaceArea * GREEN_PRICE;

    // Print results
    cout << setprecision(3);
    cout << "The surface area is " << surfaceArea << " sq. ft."
         << endl;
    cout << "The painting cost for" << endl << "   red is";
    cout << setw(8) << redCost << " dollars" << endl;
    cout << "   blue is" << setw(7) << blueCost << " dollars"
         << endl;
    cout << "   green is" << setw(6) << greenCost << " dollars"
         << endl;
    return 0;
}
```