

## 07 - Loops and Formatting

Dr. Robert Lowe

Division of Mathematics and Computer Science  
Maryville College

# Outline

1 Repeating Code

2 Formatting

# Outline

1 Repeating Code

2 Formatting

# Loop Overview

- Loops allow segments of code to be repeated.

# Loop Overview

- Loops allow segments of code to be repeated.
- Loop operation is similar to branches; they are based on true/false conditions.

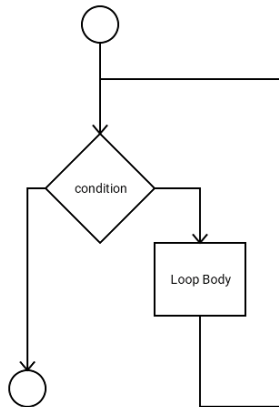
# Loop Overview

- Loops allow segments of code to be repeated.
- Loop operation is similar to branches; they are based on true/false conditions.
- C++ provides 3 types of loops: `while`, `do..while`, and `for`.

# The While Loop

## While Loop Syntax

```
while ( condition )  
    statement/block
```

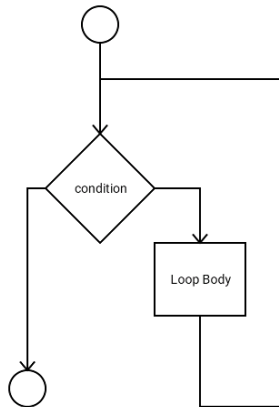


# The While Loop

## While Loop Syntax

```
while ( condition )  
    statement/block
```

- If the *condition* is true, the loop body is executed.



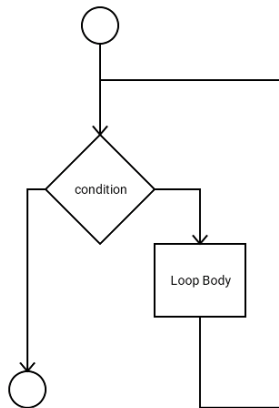


# The While Loop

## While Loop Syntax

```
while ( condition )  
    statement/block
```

- If the *condition* is true, the loop body is executed.
- After the loop body executes, the process begins again.

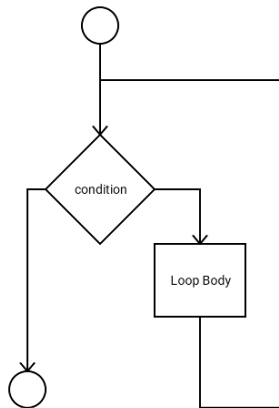


# The While Loop

## While Loop Syntax

```
while ( condition )  
    statement/block
```

- If the *condition* is true, the loop body is executed.
- After the loop body executes, the process begins again.
- How many times will the loop body execute?

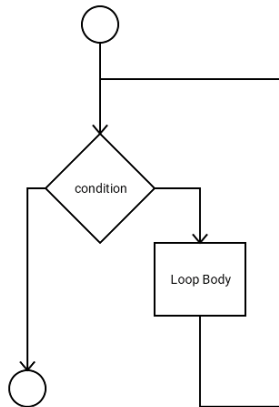


# The While Loop

## While Loop Syntax

```
while ( condition )  
    statement/block
```

- If the *condition* is true, the loop body is executed.
- After the loop body executes, the process begins again.
- How many times will the loop body execute?
  - Zero or more times!



# Example: count.cpp

```
//start at zero
num = 0;

//count to 10
while(num <= 10) {
    //display the number
    cout << num << endl;

    //go to the next number
    num = num + 1;
}
```

## Example: count.cpp

- count.cpp is in your examples/07-Loops directory.

```
//start at zero
num = 0;

//count to 10
while(num <= 10) {
    //display the number
    cout << num << endl;

    //go to the next number
    num = num + 1;
}
```

## Example: count.cpp

- count.cpp is in your examples/07-Loops directory.
- Examine this file (use less)

```
//start at zero
num = 0;

//count to 10
while(num <= 10) {
    //display the number
    cout << num << endl;

    //go to the next number
    num = num + 1;
}
```

## Example: count.cpp

- `count.cpp` is in your `examples/07-Loops` directory.
- Examine this file (use `less`)
- Compile and run the `count` program.

```
//start at zero
num = 0;

//count to 10
while(num <= 10) {
    //display the number
    cout << num << endl;

    //go to the next number
    num = num + 1;
}
```

## Example: count.cpp

- count.cpp is in your examples/07-Loops directory.
- Examine this file (use less)
- Compile and run the count program.
- The loop part of the file is shown below:

```
//start at zero
num = 0;

//count to 10
while(num <= 10) {
    //display the number
    cout << num << endl;

    //go to the next number
    num = num + 1;
}
```



# Increment and Decrement Operators

- Adding (or subtracting) 1 to a variable is quite common in loops.

# Increment and Decrement Operators

- Adding (or subtracting) 1 to a variable is quite common in loops.
- To facilitate this, C++ has an increment operator (`++`) and a decrement operator (`--`).

# Increment and Decrement Operators

- Adding (or subtracting) 1 to a variable is quite common in loops.
- To facilitate this, C++ has an increment operator (++) and a decrement operator(--).
- Both operators have a prefix version:

++x

--x

# Increment and Decrement Operators

- Adding (or subtracting) 1 to a variable is quite common in loops.
- To facilitate this, C++ has an increment operator (++) and a decrement operator(--).

- Both operators have a prefix version:

++x

--x

- Both operators have a postfix version:

x++

x--

# Prefix Increment and Decrement

## Prefix Increment and Decrement

The prefix operators increment the variable and returns the **new** value.

### Sample Evaluation

Statement	Screen Output	Value of X

# Prefix Increment and Decrement

## Prefix Increment and Decrement

The prefix operators increment the variable and returns the **new** value.

### Sample Evaluation

Statement	Screen Output	Value of X
x=0;		Undefined

# Prefix Increment and Decrement

## Prefix Increment and Decrement

The prefix operators increment the variable and returns the **new** value.

### Sample Evaluation

Statement	Screen Output	Value of X
<code>x=0;</code>		Undefined
<code>0;</code>		0

# Prefix Increment and Decrement

## Prefix Increment and Decrement

The prefix operators increment the variable and returns the **new** value.

### Sample Evaluation

Statement	Screen Output	Value of X
<code>x=0;</code>		Undefined
<code>0;</code>		0
<code>cout &lt;&lt; ++x;</code>		0



# Prefix Increment and Decrement

## Prefix Increment and Decrement

The prefix operators increment the variable and returns the **new** value.

### Sample Evaluation

Statement	Screen Output	Value of X
<code>x=0;</code>		Undefined
<code>0;</code>		0
<code>cout &lt;&lt; ++x;</code>		0
<code>cout &lt;&lt; 1;</code>		1

# Prefix Increment and Decrement

## Prefix Increment and Decrement

The prefix operators increment the variable and returns the **new** value.

### Sample Evaluation

Statement	Screen Output	Value of X
<code>x=0;</code>		Undefined
<code>0;</code>		0
<code>cout &lt;&lt; ++x;</code>		0
<code>cout &lt;&lt; 1;</code>		1
<code>cout;</code>	1	1

# Postfix Increment and Decrement

## Postfix Increment and Decrement

The postfix operators increment the variable and returns the **old** value.

## Sample Evaluation

Statement	Screen Output	Value of X

# Postfix Increment and Decrement

## Postfix Increment and Decrement

The postfix operators increment the variable and returns the **old** value.

### Sample Evaluation

Statement	Screen Output	Value of X
x=0;		Undefined

# Postfix Increment and Decrement

## Postfix Increment and Decrement

The postfix operators increment the variable and returns the **old** value.

### Sample Evaluation

Statement	Screen Output	Value of X
<code>x=0;</code>		Undefined
<code>0;</code>		0

# Postfix Increment and Decrement

## Postfix Increment and Decrement

The postfix operators increment the variable and returns the **old** value.

### Sample Evaluation

Statement	Screen Output	Value of X
<code>x=0;</code>		Undefined
<code>0;</code>		0
<code>cout &lt;&lt; x++;</code>		0

# Postfix Increment and Decrement

## Postfix Increment and Decrement

The postfix operators increment the variable and returns the **old** value.

### Sample Evaluation

Statement	Screen Output	Value of X
<code>x=0;</code>		Undefined
<code>0;</code>		0
<code>cout &lt;&lt; x++;</code>		0
<code>cout &lt;&lt; 0;</code>		1

# Postfix Increment and Decrement

## Postfix Increment and Decrement

The postfix operators increment the variable and returns the **old** value.

### Sample Evaluation

Statement	Screen Output	Value of X
<code>x=0;</code>		Undefined
<code>0;</code>		0
<code>cout &lt;&lt; x++;</code>		0
<code>cout &lt;&lt; 0;</code>		1
<code>cout;</code>	0	1



# Operator Precedence (thus far)

Operator	Description	Associativity
a++, a--	Postfix increment and decrement	Left-to-Right
not, !	Logical Not	Right-to-Left
++a, --a	Prefix increment and decrement	
a*b, a/b, a%b	Multiply, Divide, Modulus	Left-to-Right
a+b, a-b	Addition and Subtraction	Left-to-Right
«, »	Insertion and Extraction	Left-to-Right
<, <= >, >=	Relational Operators	Left-to-Right
==, !=	Equality Operators	Left-to-Right
and, &&	Logical And	Left-to-Right
or,	Logical Or	Left-to-Right
=, +=, -= *=, /= %=	Assignment and Assignment	Right-to-Left

# Lab Activity: Using Increment Operator With `count.cpp`

- 1 Copy `count.cpp` to your `labs/week4` directory.

# Lab Activity: Using Increment Operator With `count.cpp`

- 1 Copy `count.cpp` to your `labs/week4` directory.
- 2 Open your copy in your favorite text editor, and locate the following line:

```
num = num + 1;
```

# Lab Activity: Using Increment Operator With `count.cpp`

- 1 Copy `count.cpp` to your `labs/week4` directory.
- 2 Open your copy in your favorite text editor, and locate the following line:

```
num = num + 1;
```

- 3 Change this line to:  

```
num++;
```

# Lab Activity: Using Increment Operator With `count.cpp`

- 1 Copy `count.cpp` to your `labs/week4` directory.
- 2 Open your copy in your favorite text editor, and locate the following line:  

```
num = num + 1;
```
- 3 Change this line to:  

```
num++;
```
- 4 Test your program to make sure it still works.

# Lab Activity: Using Increment Operator With `count.cpp`

- 1 Copy `count.cpp` to your `labs/week4` directory.
- 2 Open your copy in your favorite text editor, and locate the following line:  

```
num = num + 1;
```
- 3 Change this line to:  

```
num++;
```
- 4 Test your program to make sure it still works.
- 5 **Discuss:** Does it matter if we use the prefix or postfix operator in this case?

## Lab Activity: `count2.cpp`

We are going to create a new program that counts from `start` to `end` by some `increment`.

## Lab Activity: `count2.cpp`

We are going to create a new program that counts from `start` to `end` by some `increment`.

For example, count from 0 to 100 by fives



# Lab Activity: `count2.cpp`

We are going to create a new program that counts from `start` to `end` by `some increment`.

For example, count from 0 to 100 by fives

- 1 Copy `count.cpp` to `count2.cpp`

# Lab Activity: `count2.cpp`

We are going to create a new program that counts from `start` to `end` by `some increment`.

For example, count from 0 to 100 by fives

- 1 Copy `count.cpp` to `count2.cpp`
- 2 Open `count2.cpp` in the text editor of your choice.

# Lab Activity: `count2.cpp`

We are going to create a new program that counts from `start` to `end` by `some` `increment`.

For example, count from 0 to 100 by fives

- 1 Copy `count.cpp` to `count2.cpp`
- 2 Open `count2.cpp` in the text editor of your choice.
- 3 Add variables for `start`, `end`, and `increment`.

```
int start; //The first number
int end;   //The last number
int increment; //The amount to add each time
```

# Lab Activity: `count2.cpp`

We are going to create a new program that counts from `start` to `end` by some `increment`.

For example, count from 0 to 100 by fives

- 1 Copy `count.cpp` to `count2.cpp`
- 2 Open `count2.cpp` in the text editor of your choice.
- 3 Add variables for `start`, `end`, and `increment`.

```
int start; //The first number
int end;   //The last number
int increment; //The amount to add each time
```

- 4 Modify the program so that the first thing it does is prompt the user and read in these three variables.

# Lab Activity: `count2.cpp` Continued

- 4 Modify the program so that instead of starting at zero, it starts at `start`.

```
//start at start  
num = start;
```

# Lab Activity: `count2.cpp` Continued

- 4 Modify the program so that instead of starting at zero, it starts at `start`.

```
//start at start  
num = start;
```

- 4 Modify the program so that instead of adding 1 to `num` each time through the loop, it adds `increment`.

```
//go to the next number  
num += increment;
```

# Lab Activity: `count2.cpp` Continued

- 4 Modify the program so that instead of starting at zero, it starts at `start`.

```
//start at start  
num = start;
```

- 4 Modify the program so that instead of adding 1 to `num` each time through the loop, it adds `increment`.

```
//go to the next number  
num += increment;
```

- 4 Compile and test your program.

# Loop Problem: Fahrenheit Scale

- In 1724, Daniel Fahrenheit proposed a precise way to measure temperature.



**Daniel Fahrenheit**

Image Source: [https://en.wikipedia.org/wiki/](https://en.wikipedia.org/wiki/Daniel_Gabriel_Fahrenheit)

[Daniel\\_Gabriel\\_Fahrenheit](https://en.wikipedia.org/wiki/Daniel_Gabriel_Fahrenheit)

Maryville  
COLLEGE



# Loop Problem: Fahrenheit Scale

- In 1724, Daniel Fahrenheit proposed a precise way to measure temperature.
- His scale was reproducible and brilliant!



**Daniel Fahrenheit**

Image Source: [https:](https://en.wikipedia.org/wiki/Daniel_Gabriel_Fahrenheit)

[//en.wikipedia.org/wiki/](https://en.wikipedia.org/wiki/Daniel_Gabriel_Fahrenheit)

[Daniel\\_Gabriel\\_Fahrenheit](https://en.wikipedia.org/wiki/Daniel_Gabriel_Fahrenheit)

# Loop Problem: Fahrenheit Scale

- In 1724, Daniel Fahrenheit proposed a precise way to measure temperature.
- His scale was reproducible and brilliant!
- 0 was fixed at the temperature achieved by mixing equal quantities of water, ice, and ammonia chloride.



**Daniel Fahrenheit**

Image Source: [https://en.wikipedia.org/wiki/](https://en.wikipedia.org/wiki/Daniel_Gabriel_Fahrenheit)

[https://en.wikipedia.org/wiki/](https://en.wikipedia.org/wiki/Daniel_Gabriel_Fahrenheit)

Daniel\_Gabriel\_Fahrenheit

# Loop Problem: Fahrenheit Scale

- In 1724, Daniel Fahrenheit proposed a precise way to measure temperature.
- His scale was reproducible and brilliant!
- 0 was fixed at the temperature achieved by mixing equal quantities of water, ice, and ammonia chloride.
- 100 is the temperature of healthy blood.



**Daniel Fahrenheit**

Image Source: [https://en.wikipedia.org/wiki/](https://en.wikipedia.org/wiki/Daniel_Gabriel_Fahrenheit)

[https://en.wikipedia.org/wiki/](https://en.wikipedia.org/wiki/Daniel_Gabriel_Fahrenheit)

Daniel\_Gabriel\_Fahrenheit

# Loop Problem: Fahrenheit Scale

- In 1724, Daniel Fahrenheit proposed a precise way to measure temperature.
- His scale was reproducible and brilliant!
- 0 was fixed at the temperature achieved by mixing equal quantities of water, ice, and ammonia chloride.
- 100 is the temperature of healthy blood.
- The scale is divided into equal marks from 0 to 100.



**Daniel Fahrenheit**

Image Source: [https://en.wikipedia.org/wiki/](https://en.wikipedia.org/wiki/Daniel_Gabriel_Fahrenheit)

[https://en.wikipedia.org/wiki/](https://en.wikipedia.org/wiki/Daniel_Gabriel_Fahrenheit)

Daniel\_Gabriel\_Fahrenheit

# Loop Problem: Celsius Scale

- In 1742, Anders Celsius proposed a new scale.



**Anders Celsius**

Image Source:

[https://en.wikipedia.org/  
wiki/Anders\\_Celsius](https://en.wikipedia.org/wiki/Anders_Celsius)

# Loop Problem: Celsius Scale

- In 1742, Anders Celsius proposed a new scale.
- Celsius's scale was, I think you'll agree, completely unrelatable.



**Anders Celsius**

Image Source:

[https://en.wikipedia.org/  
wiki/Anders\\_Celsius](https://en.wikipedia.org/wiki/Anders_Celsius)

# Loop Problem: Celsius Scale

- In 1742, Anders Celsius proposed a new scale.
- Celsius's scale was, I think you'll agree, completely unrelatable.
- For 0, Celsius used the freezing point of water at sea level.



**Anders Celsius**

Image Source:

[https://en.wikipedia.org/  
wiki/Anders\\_Celsius](https://en.wikipedia.org/wiki/Anders_Celsius)

# Loop Problem: Celsius Scale

- In 1742, Anders Celsius proposed a new scale.
- Celsius's scale was, I think you'll agree, completely unrelatable.
- For 0, Celsius used the freezing point of water at sea level.
- For 100, Celsius used the boiling point of water at sea level.



**Anders Celsius**

Image Source:

[https://en.wikipedia.org/  
wiki/Anders\\_Celsius](https://en.wikipedia.org/wiki/Anders_Celsius)



# Loop Problem: Celsius Scale

- In 1742, Anders Celsius proposed a new scale.
- Celsius's scale was, I think you'll agree, completely unrelatable.
- For 0, Celsius used the freezing point of water at sea level.
- For 100, Celsius used the boiling point of water at sea level.
- Only the United States holds fast to the bitter salt slush to blood scale.



**Anders Celsius**

Image Source:

[https://en.wikipedia.org/  
wiki/Anders\\_Celsius](https://en.wikipedia.org/wiki/Anders_Celsius)

# Loop Problem: Celsius Scale

- In 1742, Anders Celsius proposed a new scale.
- Celsius's scale was, I think you'll agree, completely unrelatable.
- For 0, Celsius used the freezing point of water at sea level.
- For 100, Celsius used the boiling point of water at sea level.
- Only the United States holds fast to the bitter salt slush to blood scale.
- God bless the USA, and long live the imperial system of units!



**Anders Celsius**

Image Source:

[https://en.wikipedia.org/  
wiki/Anders\\_Celsius](https://en.wikipedia.org/wiki/Anders_Celsius)

# Loop Problem: Fahrenheit to Celsius Table

- In reluctant deference to the people who use the inferior scale (96% of the world's population), we will produce a table converting from Fahrenheit to Celsius.

# Loop Problem: Fahrenheit to Celsius Table

- In reluctant deference to the people who use the inferior scale (96% of the world's population), we will produce a table converting from Fahrenheit to Celsius.
- The table will allow the user to specify the starting temperature, ending temperature, and increment.

# Loop Problem: Fahrenheit to Celsius Table

- In reluctant deference to the people who use the inferior scale (96% of the world's population), we will produce a table converting from Fahrenheit to Celsius.
- The table will allow the user to specify the starting temperature, ending temperature, and increment.
- The table should be nicely formatted.

# Loop Problem: Fahrenheit to Celsius Table

- In reluctant deference to the people who use the inferior scale (96% of the world's population), we will produce a table converting from Fahrenheit to Celsius.
- The table will allow the user to specify the starting temperature, ending temperature, and increment.
- The table should be nicely formatted.
- The formula for doing the conversion is:

$$c = \frac{5}{9}(f - 32)$$

# Loop Problem: Fahrenheit to Celsius Table

- In reluctant deference to the people who use the inferior scale (96% of the world's population), we will produce a table converting from Fahrenheit to Celsius.
- The table will allow the user to specify the starting temperature, ending temperature, and increment.
- The table should be nicely formatted.
- The formula for doing the conversion is:

$$c = \frac{5}{9}(f - 32)$$

- Let's talk about the design of this program for a minute.

# Loop Problem: Fahrenheit to Celsius Table (continued)

1 Copy `count2.cpp` to `fahrenheit.cpp`



# Loop Problem: Fahrenheit to Celsius Table (continued)

- 1 Copy `count2.cpp` to `fahrenheit.cpp`
- 2 Change all mentions of the variable `num` to `f`

# Loop Problem: Fahrenheit to Celsius Table (continued)

- 1 Copy `count2.cpp` to `fahrenheit.cpp`
- 2 Change all mentions of the variable `num` to `f`
- 3 Change all variable types to `double`

# Loop Problem: Fahrenheit to Celsius Table (continued)

- 1 Copy `count2.cpp` to `fahrenheit.cpp`
- 2 Change all mentions of the variable `num` to `f`
- 3 Change all variable types to `double`
- 4 Add a variable `c`

# Loop Problem: Fahrenheit to Celsius Table (continued)

- 1 Copy `count2.cpp` to `fahrenheit.cpp`
- 2 Change all mentions of the variable `num` to `f`
- 3 Change all variable types to `double`
- 4 Add a variable `c`
- 5 Alter the program so that the first thing it does is display a welcome message.

Fahrenheit to Celsius Temperature Table

# Loop Problem: Fahrenheit to Celsius Table (continued)

- 1 Copy `count2.cpp` to `fahrenheit.cpp`
- 2 Change all mentions of the variable `num` to `f`
- 3 Change all variable types to `double`
- 4 Add a variable `c`
- 5 Alter the program so that the first thing it does is display a welcome message.

Fahrenheit to Celsius Temperature Table

- 6 Just before the `while` loop, have the program print a header on a line by itself:  
Fahrenheit    Celsius

# Loop Problem: Fahrenheit to Celsius Table (continued)

- 1 Copy `count2.cpp` to `fahrenheit.cpp`
- 2 Change all mentions of the variable `num` to `f`
- 3 Change all variable types to `double`
- 4 Add a variable `c`
- 5 Alter the program so that the first thing it does is display a welcome message.

Fahrenheit to Celsius Temperature Table

- 6 Just before the `while` loop, have the program print a header on a line by itself:

Fahrenheit    Celsius

- 7 At the beginning of the loop body, add code to compute the Celsius value `c` from the fahrenheit value `f`.

# Loop Problem: Fahrenheit to Celsius Table (continued)

- 1 Copy `count2.cpp` to `fahrenheit.cpp`
- 2 Change all mentions of the variable `num` to `f`
- 3 Change all variable types to `double`
- 4 Add a variable `c`
- 5 Alter the program so that the first thing it does is display a welcome message.

Fahrenheit to Celsius Temperature Table

- 6 Just before the `while` loop, have the program print a header on a line by itself:  
Fahrenheit   Celsius
- 7 At the beginning of the loop body, add code to compute the Celsius value `c` from the fahrenheit value `f`.
- 8 Alter the line that prints out `f` so that it prints the fahrenheit and Celsius temperatures.

# Loop Problem: Fahrenheit to Celsius Table (continued)

- ❶ Copy `count2.cpp` to `fahrenheit.cpp`
- ❷ Change all mentions of the variable `num` to `f`
- ❸ Change all variable types to `double`
- ❹ Add a variable `c`
- ❺ Alter the program so that the first thing it does is display a welcome message.

Fahrenheit to Celsius Temperature Table

- ❻ Just before the `while` loop, have the program print a header on a line by itself:  
Fahrenheit   Celsius
- ❼ At the beginning of the loop body, add code to compute the Celsius value `c` from the fahrenheit value `f`.
- ❽ Alter the line that prints out `f` so that it prints the fahrenheit and Celsius temperatures.
- ❾ Compile and test your program.



# Outline

1 Repeating Code

2 Formatting

# cout and Fields

- `cout` does its work by translating its input into a string of characters.

# cout and Fields

- `cout` does its work by translating its input into a string of characters.
- Every `<<` operations that results in output is called a **field**.

# cout and Fields

- `cout` does its work by translating its input into a string of characters.
- Every « operations that results in output is called a **field**.
- There exist a series of flags which affect how `cout` performs formats its output.

# cout and Fields

- `cout` does its work by translating its input into a string of characters.
- Every « operations that results in output is called a **field**.
- There exist a series of flags which affect how `cout` performs formats its output.
- Managing these flags individually is tedious and painful!

# iomanip

- `iomanip` is a collection of **input manipulators** that make working with `cout` easier.

# iomanip

- `iomanip` is a collection of **input manipulators** that make working with `cout` easier.
- To use `iomanip` you have to include it:  

```
#include <iomanip>
```

# iomanip

- `iomanip` is a collection of **input manipulators** that make working with `cout` easier.
- To use `iomanip` you have to include it:  
`#include <iomanip>`
- Go ahead and add this to `fahrenheit.cpp`



# Setting the Field Width

- Most io manipulators are used inline with insertion operators.

# Setting the Field Width

- Most io manipulators are used inline with insertion operators.
- One very handy manipulator is `setw`, which sets the width of the next field.

# Setting the Field Width

- Most io manipulators are used inline with insertion operators.
- One very handy manipulator is `setw`, which sets the width of the next field.
- For example, modify the `fahrenheit` program's line which displays the temperatures:

```
//display the values  
cout << setw(10) << f << "  "  
      << setw(7) << c << endl;
```

# Setting the Field Width

- Most io manipulators are used inline with insertion operators.
- One very handy manipulator is `setw`, which sets the width of the next field.
- For example, modify the `fahrenheit` program's line which displays the temperatures:  

```
//display the values  
cout << setw(10) << f << "  "  
      << setw(7) << c << endl;
```
- Test and run your programs. See if you can guess why I selected the numbers 10 and 7.

# Number Formatting

- The precision of floating point number displays is controlled using the `set_precision` function.

# Number Formatting

- The precision of floating point number displays is controlled using the `set_precision` function.
- In default or normal mode, precision indicates the maximum total number of digits displayed.

# Number Formatting

- The precision of floating point number displays is controlled using the `set_precision` function.
- In default or normal mode, precision indicates the maximum total number of digits displayed.
- In `fixed` mode, precision indicates the number of digits to the right of the decimal point.

# Number Formatting

- The precision of floating point number displays is controlled using the `set_precision` function.
- In default or normal mode, precision indicates the maximum total number of digits displayed.
- In `fixed` mode, precision indicates the number of digits to the right of the decimal point.
- Once set, the numeric mode remains in effect until it is changed.



# Number Formatting

- The precision of floating point number displays is controlled using the `set_precision` function.
- In default or normal mode, precision indicates the maximum total number of digits displayed.
- In `fixed` mode, precision indicates the number of digits to the right of the decimal point.
- Once set, the numeric mode remains in effect until it is changed.
- Just before your loop in `fahrenheit.cpp` add the following:

```
//set up number format  
cout << fixed << setprecision(2);
```

# Number Formatting

- The precision of floating point number displays is controlled using the `set_precision` function.
- In default or normal mode, precision indicates the maximum total number of digits displayed.
- In `fixed` mode, precision indicates the number of digits to the right of the decimal point.
- Once set, the numeric mode remains in effect until it is changed.
- Just before your loop in `fahrenheit.cpp` add the following:

```
//set up number format  
cout << fixed << setprecision(2);
```

- Compile and test your program. Isn't that nice?

# Week 4 Lab Requirements

For full credit, your week 4 lab directory should contain:

- 1 A fully corrected `proportions.cpp`
- 2 `stock.cpp` with working menu messages.
- 3 `count.cpp` using the increment operator.
- 4 `count2.cpp`
- 5 `fahrenheit.cpp`