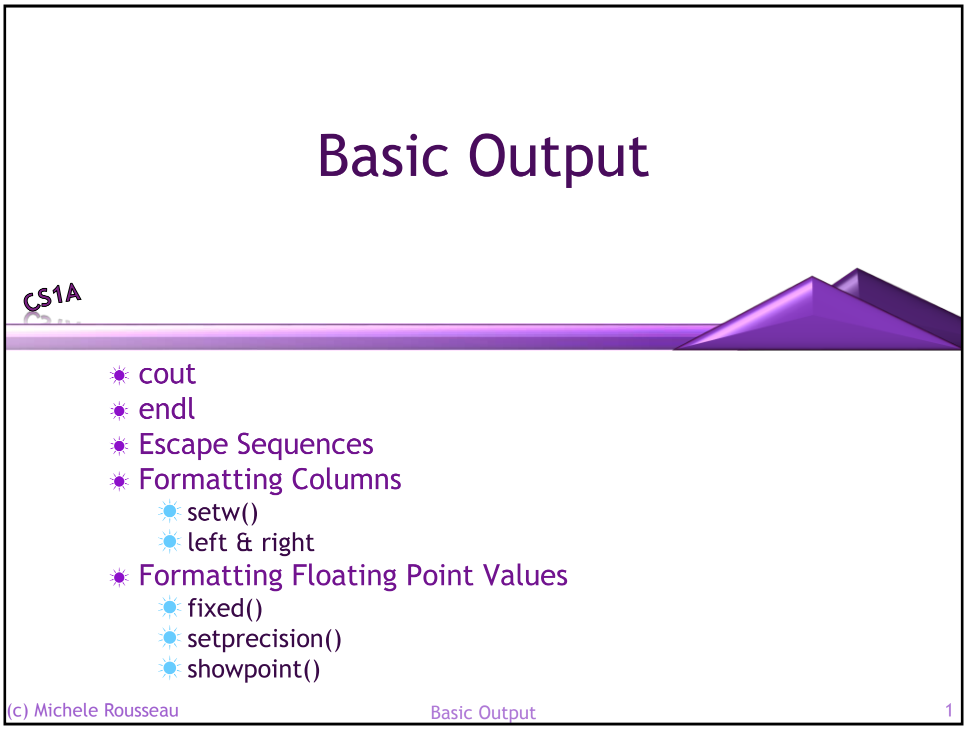
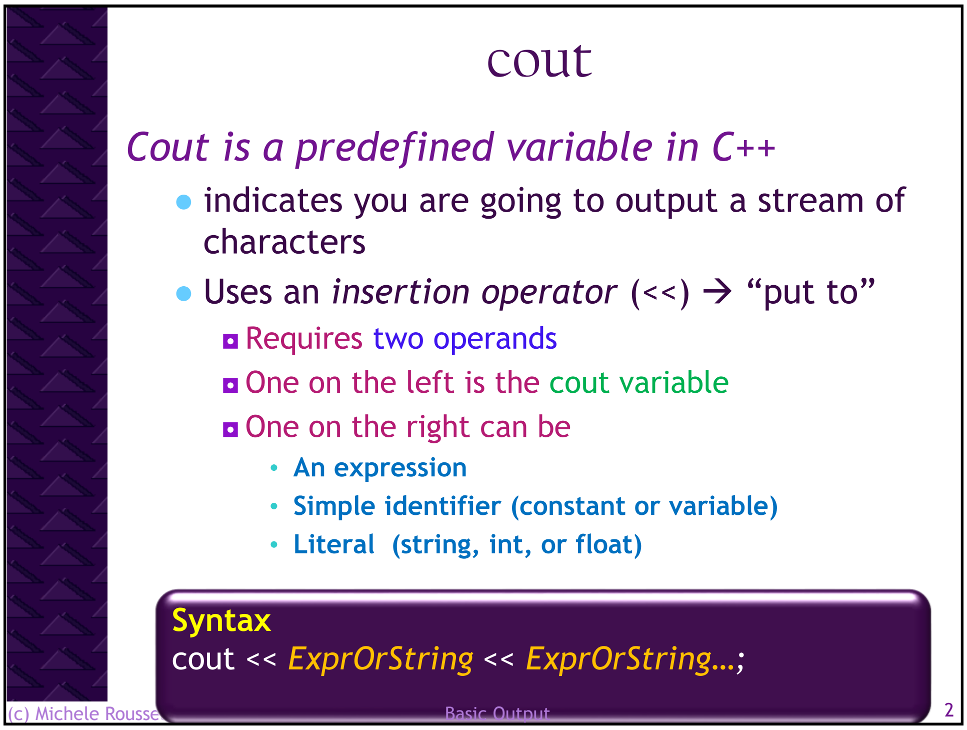


Basic Output

CS1A

- cout
- endl
- Escape Sequences
- Formatting Columns
  - setw()
  - left & right
- Formatting Floating Point Values
  - fixed()
  - setprecision()
  - showpoint()

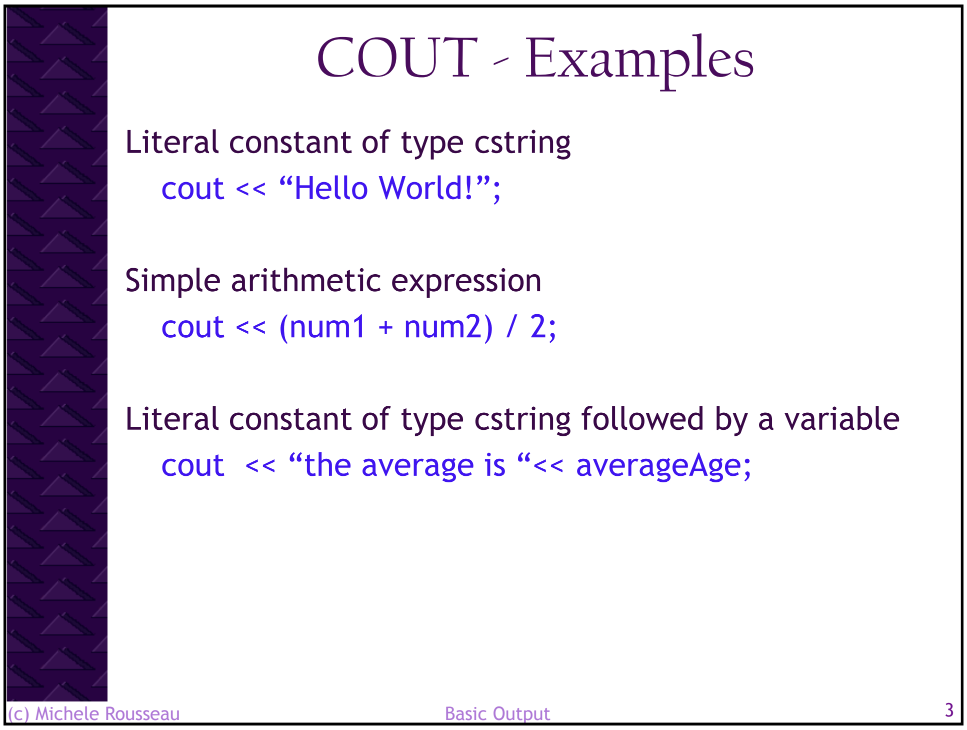


cout

Cout is a predefined variable in C++

- indicates you are going to output a stream of characters
- Uses an *insertion operator* (<<) → "put to"
  - Requires two operands
  - One on the left is the cout variable
  - One on the right can be
    - An expression
    - Simple identifier (constant or variable)
    - Literal (string, int, or float)

Syntax
cout << *ExprOrString* << *ExprOrString…*;



COUT - Examples

Literal constant of type cstring

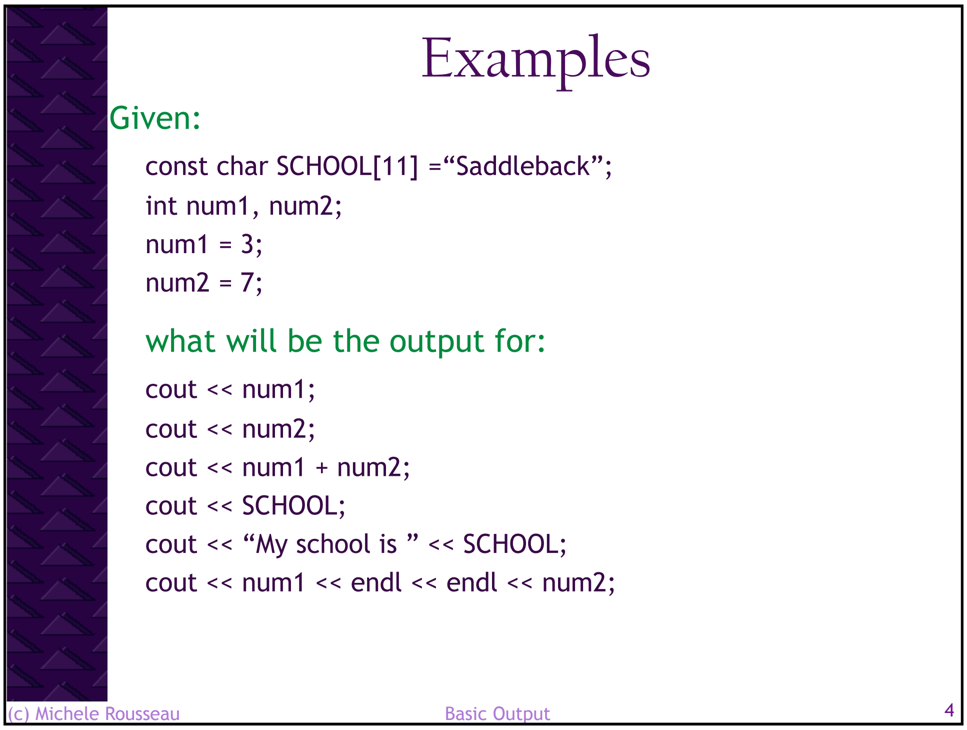
```
cout << “Hello World!”;
```

Simple arithmetic expression

```
cout << (num1 + num2) / 2;
```

Literal constant of type cstring followed by a variable

```
cout  << “the average is “<< averageAge;
```



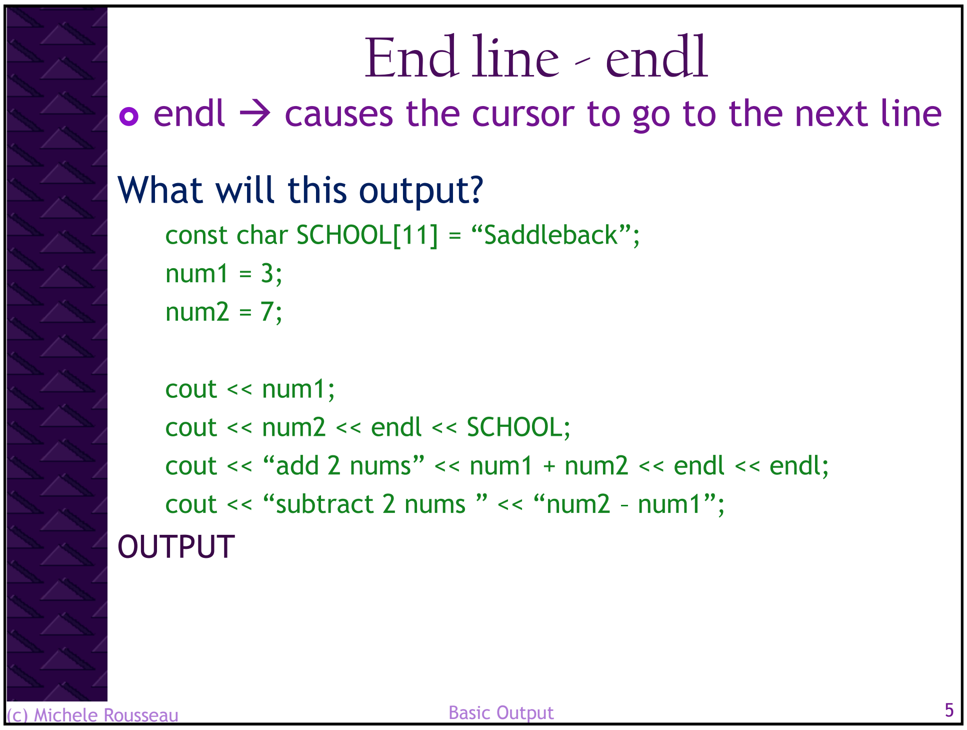
Examples

Given:

```
const char SCHOOL[11] =“Saddleback”;
int num1, num2;
num1 = 3;
num2 = 7;
```

what will be the output for:

```
cout << num1;
cout << num2;
cout << num1 + num2;
cout << SCHOOL;
cout << “My school is ” << SCHOOL;
cout << num1 << endl << endl << num2;
```



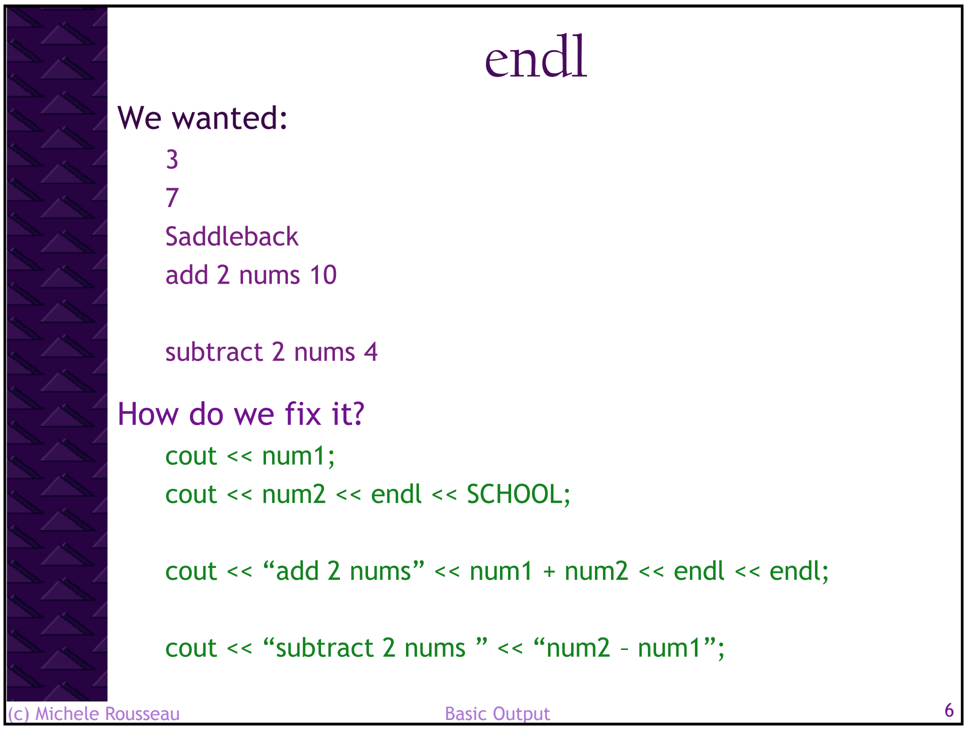
End line - endl

- endl → causes the cursor to go to the next line

What will this output?

```
const char SCHOOL[11] = “Saddleback”;
num1 = 3;
num2 = 7;

cout << num1;
cout << num2 << endl << SCHOOL;
cout << “add 2 nums” << num1 + num2 << endl << endl;
cout << “subtract 2 nums ” << “num2 - num1”;
```

OUTPUT



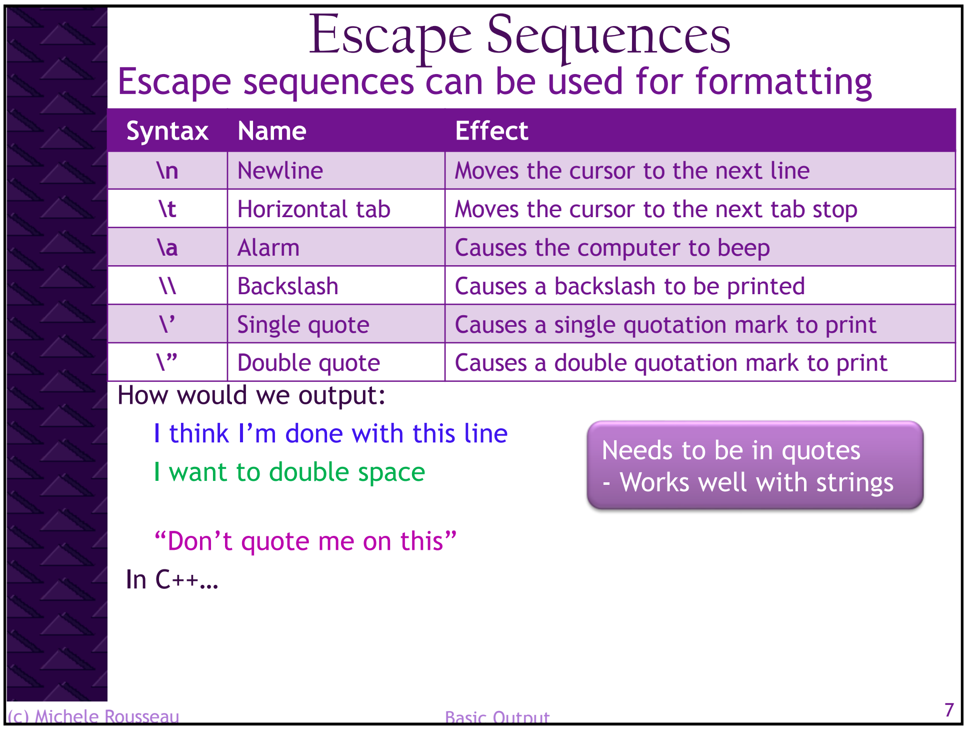
endl

We wanted:

```
3
7
Saddleback
add 2 nums 10

subtract 2 nums 4
```

How do we fix it?

```
cout << num1;
cout << num2 << endl << SCHOOL;

cout << “add 2 nums” << num1 + num2 << endl << endl;

cout << “subtract 2 nums ” << “num2 - num1”;
```



Escape Sequences

Escape sequences can be used for formatting

Syntax	Name	Effect
\n	Newline	Moves the cursor to the next line
\t	Horizontal tab	Moves the cursor to the next tab stop
\a	Alarm	Causes the computer to beep
\\	Backslash	Causes a backslash to be printed
\’	Single quote	Causes a single quotation mark to print
\”	Double quote	Causes a double quotation mark to print

How would we output:

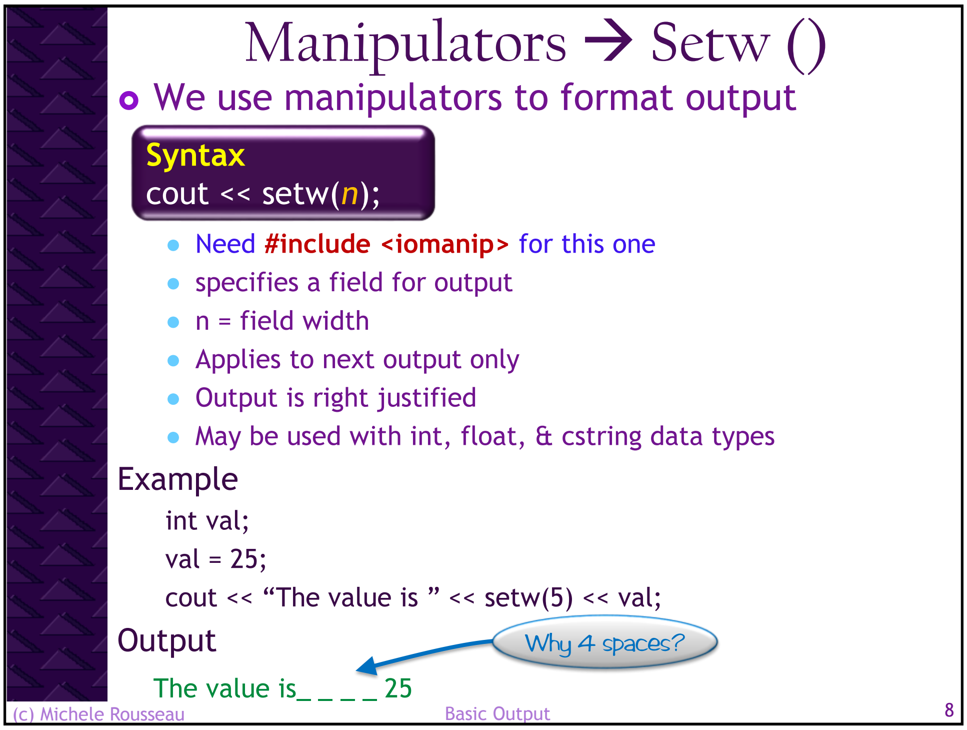
I think I’m done with this line

I want to double space

“Don’t quote me on this”

In C++…

Needs to be in quotes
- Works well with strings



Manipulators → Setw ()

- We use manipulators to format output

Syntax
cout << setw(*n*);

- Need **#include <iomanip>** for this one
- specifies a field for output
- n = field width
- Applies to next output only
- Output is right justified
- May be used with int, float, & cstring data types

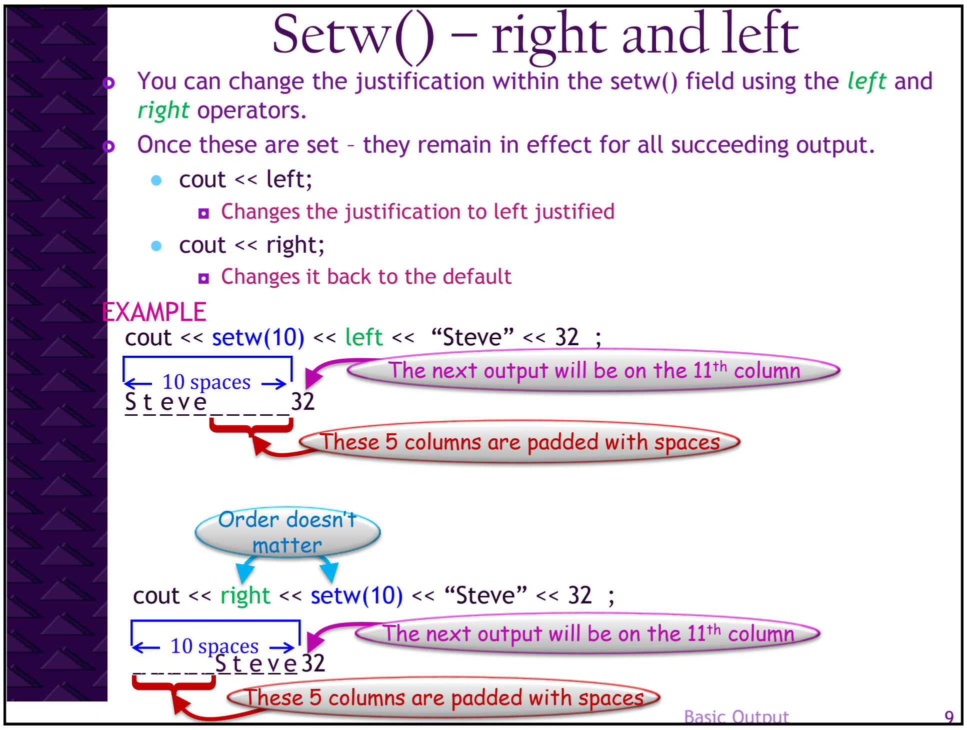
Example

```
int val;
val = 25;
cout << “The value is ” << setw(5) << val;
```

Output

```
The value is_ _ _ _ 25
```

Why 4 spaces?



Setw() – right and left

- You can change the justification within the setw() field using the *left* and *right* operators.
- Once these are set - they remain in effect for all succeeding output.
  - cout << left;
    - Changes the justification to left justified
  - cout << right;
    - Changes it back to the default

EXAMPLE

```
cout << setw(10) << left <<  "Steve" << 32  ;
```

← 10 spaces →

```
Steve_____32
```

The next output will be on the 11th column

These 5 columns are padded with spaces

Order doesn't matter

```
cout << right << setw(10) << "Steve" << 32  ;
```

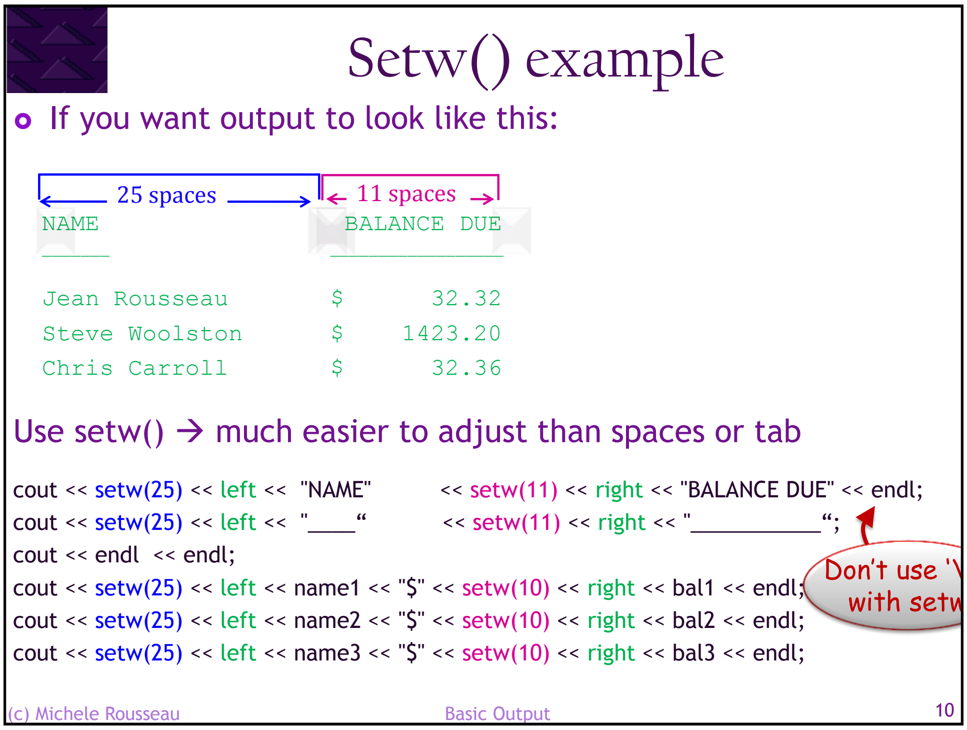
← 10 spaces →

```
_____Steve32
```

The next output will be on the 11th column

These 5 columns are padded with spaces

Basic Output 9



Setw() example

- If you want output to look like this:

← 25 spaces → ← 11 spaces →

```
NAME                      BALANCE  DUE
____                      ____________

Jean Rousseau            $       32.32
Steve Woolston           $     1423.20
Chris Carroll            $       32.36
```

Use setw() → much easier to adjust than spaces or tab

```
cout << setw(25) << left <<  "NAME"        << setw(11) << right << "BALANCE DUE" << endl;
cout << setw(25) << left <<  "____"        << setw(11) << right << "___________";
cout << endl  << endl;
cout << setw(25) << left << name1 << "$" << setw(10) << right << bal1 << endl;
cout << setw(25) << left << name2 << "$" << setw(10) << right << bal2 << endl;
cout << setw(25) << left << name3 << "$" << setw(10) << right << bal3 << endl;
```

Don't use '\
with setw

(c) Michele Rousseau Basic Output 10

How can we format cout/cin pairs?

Example:

```
Enter your name: Bill Ding
Balance Due:     32.5
```

Formatting floating point values

Decimals can be formatted to your specific needs

#include <iomanip>

→ you need this for the next 3 manipulators

- fixed
- setprecision(n)
- showpoint

Manipulators → Fixed

fixed

- Displays in fixed decimal format
  - In other words → sets the # of decimal places that will display
- Use with setprecision to set the # of places
  - Default set precision is 6
- Eg.

```
cout << fixed;
```

- Need to use cout.unsetf(ios::fixed); to turn it off

Fixed Example

```
{
    double val1;
    double val2;
    double val3;

    val1 = 423.353607;
    val2 = 3.1455929;
    val3 = 5;

    cout << setw(12) << val1 << endl;
    cout << setw(12) << val2 << endl;
    cout << setw(12) << val3 << endl << endl << endl;

    cout << fixed;
    cout << setw(12) << val1 << endl;
    cout << setw(12) << val2 << endl
    cout << setw(12) << val3 << endl;
}
```

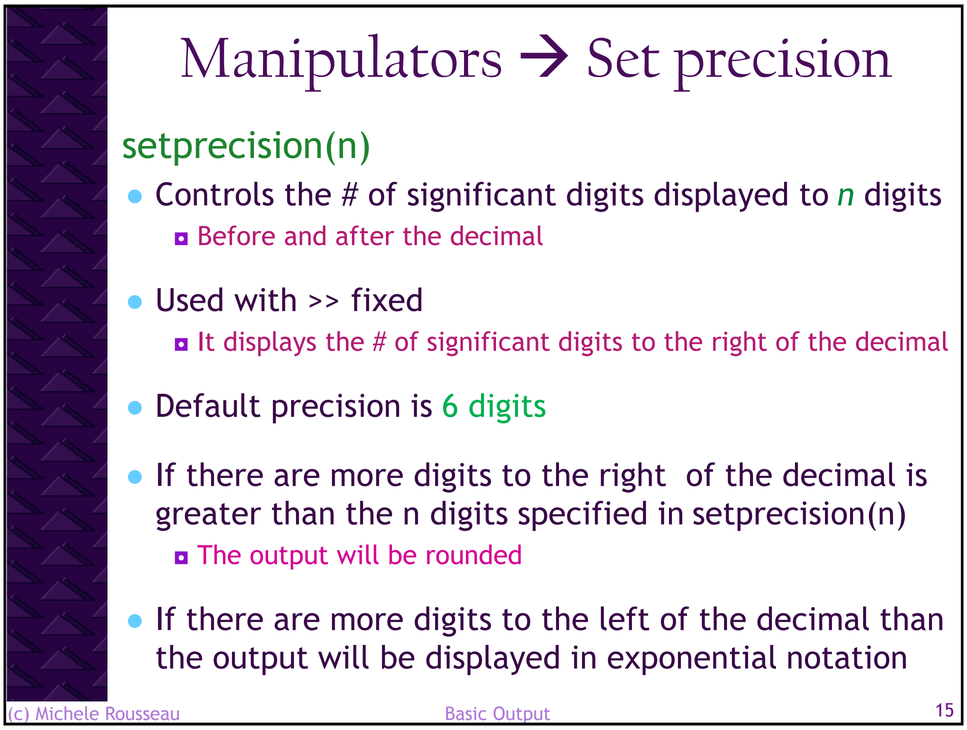
OUTPUT

```
     423.354
     3.14559
           5

  423.353607
    3.145593
    5.000000
```

default precision is set to 6

It rounds

With fixed it forces 0s to the current precision → Note there are 6 0s

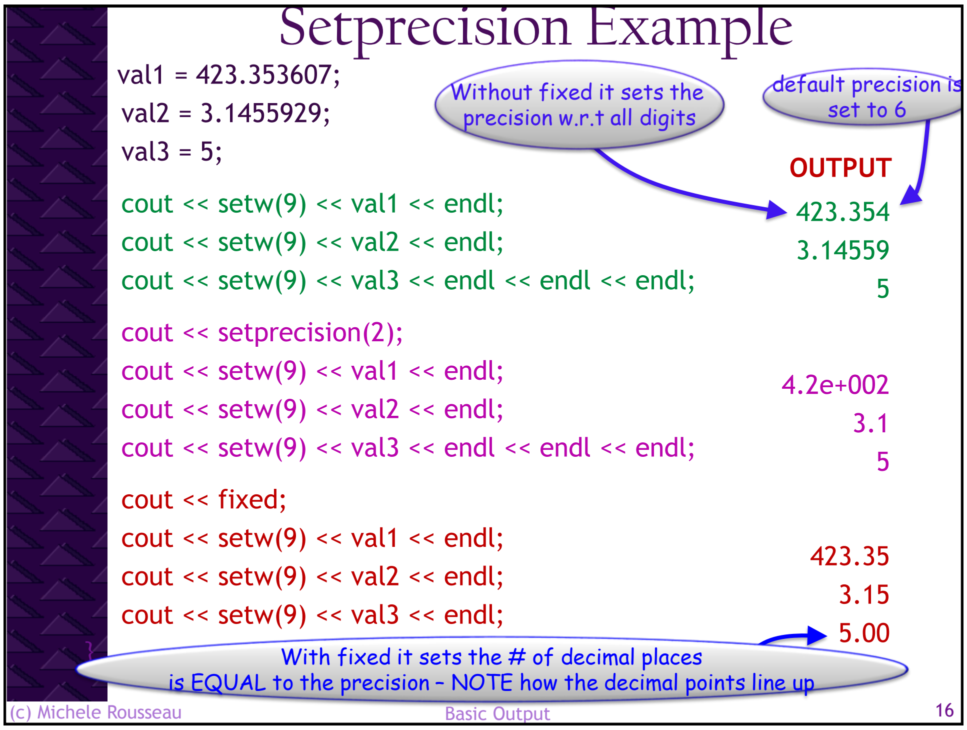


Manipulators → Set precision

setprecision(n)

- Controls the # of significant digits displayed to *n* digits
  - Before and after the decimal
- Used with >> fixed
  - It displays the # of significant digits to the right of the decimal
- Default precision is 6 digits
- If there are more digits to the right of the decimal is greater than the n digits specified in setprecision(n)
  - The output will be rounded
- If there are more digits to the left of the decimal than the output will be displayed in exponential notation

(c) Michele Rousseau Basic Output 15



Setprecision Example

```
val1 = 423.353607;
val2 = 3.1455929;
val3 = 5;

cout << setw(9) << val1 << endl;
cout << setw(9) << val2 << endl;
cout << setw(9) << val3 << endl << endl << endl;

cout << setprecision(2);
cout << setw(9) << val1 << endl;
cout << setw(9) << val2 << endl;
cout << setw(9) << val3 << endl << endl << endl;

cout << fixed;
cout << setw(9) << val1 << endl;
cout << setw(9) << val2 << endl;
cout << setw(9) << val3 << endl;
```

OUTPUT

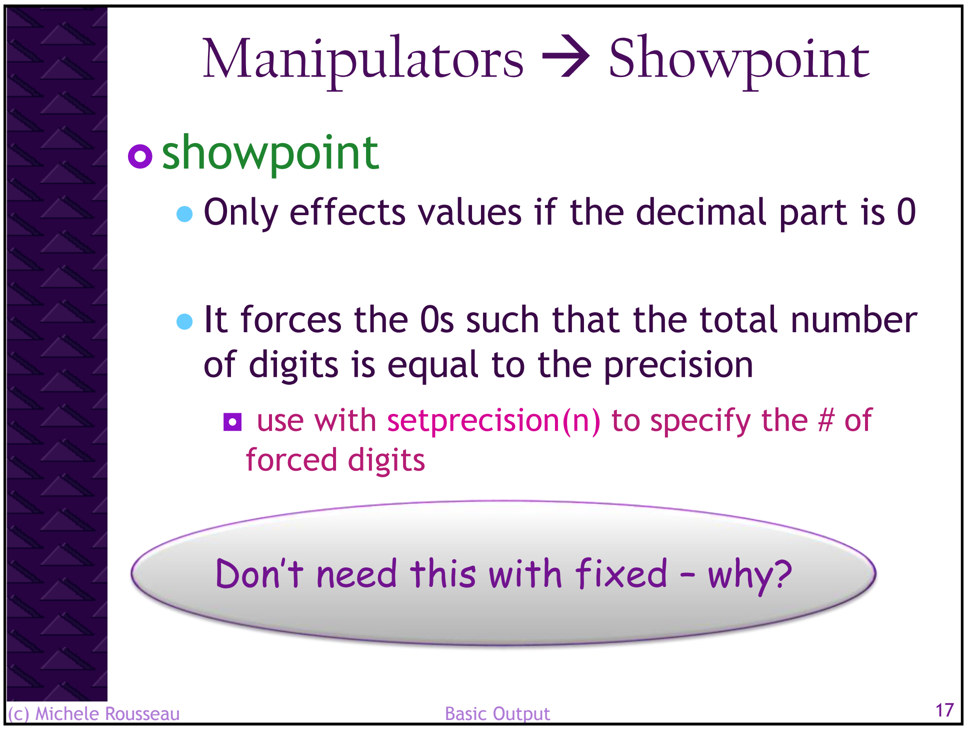
```
  423.354
  3.14559
        5


  4.2e+002
      3.1
        5


   423.35
     3.15
     5.00
```

Without fixed it sets the precision w.r.t all digits

default precision is set to 6

With fixed it sets the # of decimal places is EQUAL to the precision – NOTE how the decimal points line up

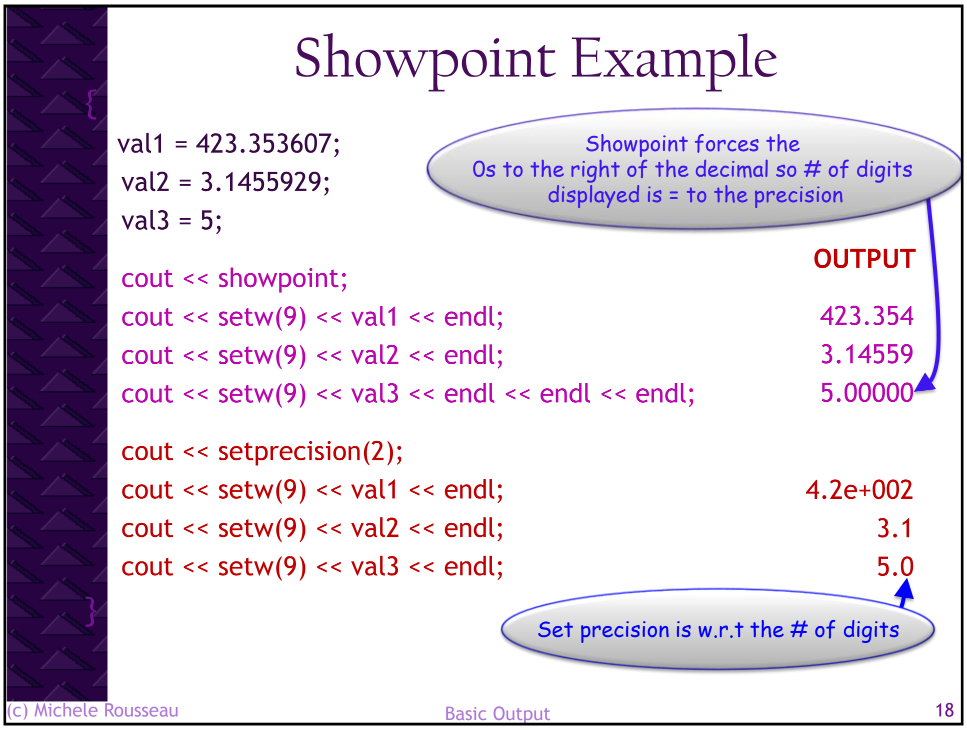
(c) Michele Rousseau Basic Output 16



Manipulators → Showpoint

- showpoint
  - Only effects values if the decimal part is 0
  - It forces the 0s such that the total number of digits is equal to the precision
    - use with setprecision(n) to specify the # of forced digits

Don't need this with fixed – why?

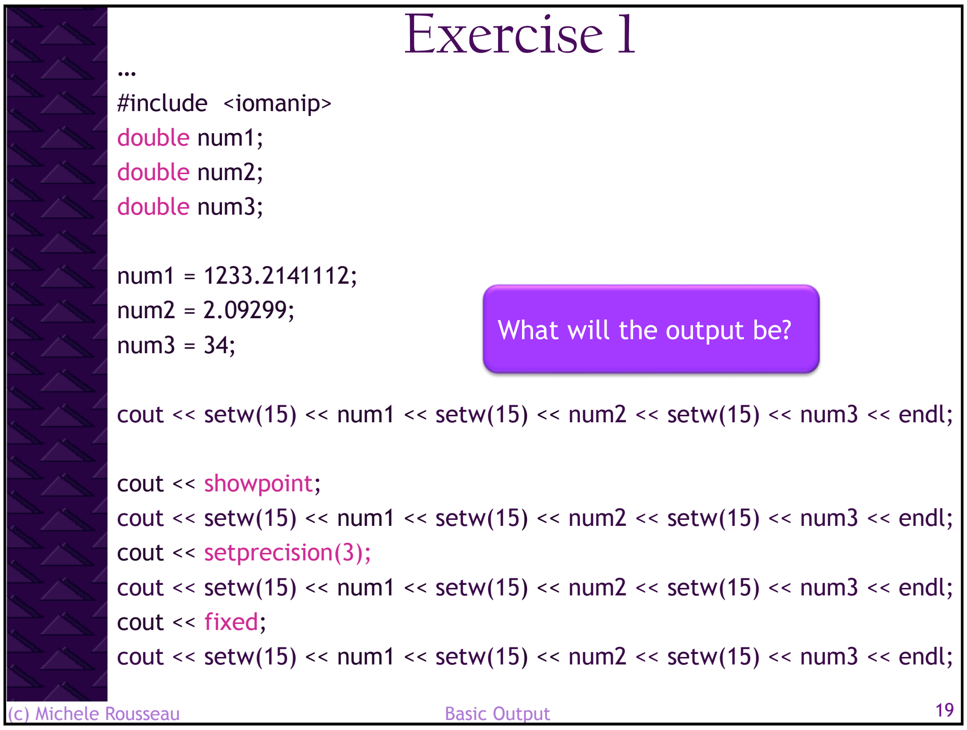


Showpoint Example

```
{
val1 = 423.353607;
val2 = 3.1455929;
val3 = 5;
                                                          OUTPUT
cout << showpoint;
cout << setw(9) << val1 << endl;                          423.354
cout << setw(9) << val2 << endl;                          3.14559
cout << setw(9) << val3 << endl << endl << endl;          5.00000

cout << setprecision(2);
cout << setw(9) << val1 << endl;                         4.2e+002
cout << setw(9) << val2 << endl;                              3.1
cout << setw(9) << val3 << endl;                              5.0
}
```

Showpoint forces the 0s to the right of the decimal so # of digits displayed is = to the precision

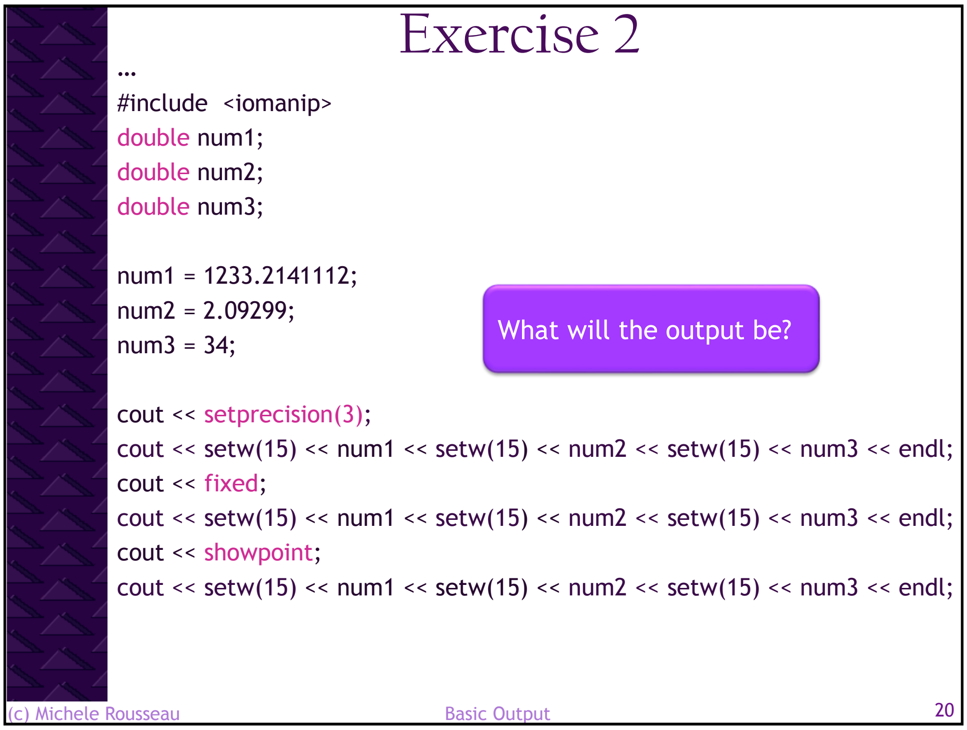
Set precision is w.r.t the # of digits



Exercise 1

```
…
#include  <iomanip>
double num1;
double num2;
double num3;

num1 = 1233.2141112;
num2 = 2.09299;
num3 = 34;

cout << setw(15) << num1 << setw(15) << num2 << setw(15) << num3 << endl;

cout << showpoint;
cout << setw(15) << num1 << setw(15) << num2 << setw(15) << num3 << endl;
cout << setprecision(3);
cout << setw(15) << num1 << setw(15) << num2 << setw(15) << num3 << endl;
cout << fixed;
cout << setw(15) << num1 << setw(15) << num2 << setw(15) << num3 << endl;
```

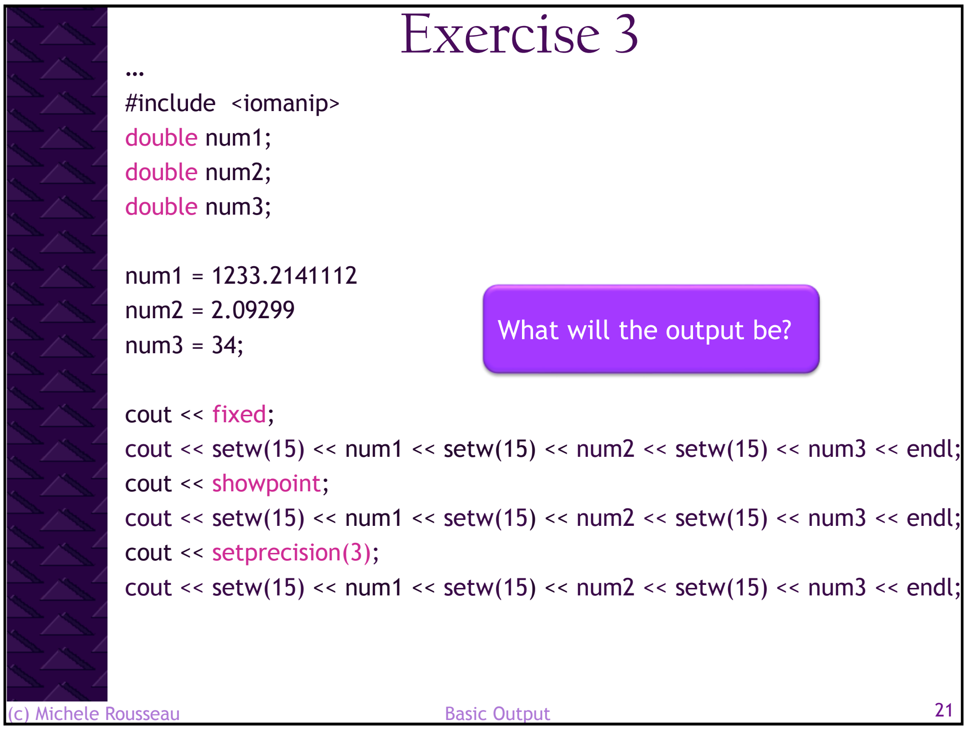
What will the output be?



Exercise 2

```
…
#include  <iomanip>
double num1;
double num2;
double num3;

num1 = 1233.2141112;
num2 = 2.09299;
num3 = 34;

cout << setprecision(3);
cout << setw(15) << num1 << setw(15) << num2 << setw(15) << num3 << endl;
cout << fixed;
cout << setw(15) << num1 << setw(15) << num2 << setw(15) << num3 << endl;
cout << showpoint;
cout << setw(15) << num1 << setw(15) << num2 << setw(15) << num3 << endl;
```

What will the output be?



Exercise 3

```
…
#include  <iomanip>
double num1;
double num2;
double num3;

num1 = 1233.2141112
num2 = 2.09299
num3 = 34;

cout << fixed;
cout << setw(15) << num1 << setw(15) << num2 << setw(15) << num3 << endl;
cout << showpoint;
cout << setw(15) << num1 << setw(15) << num2 << setw(15) << num3 << endl;
cout << setprecision(3);
cout << setw(15) << num1 << setw(15) << num2 << setw(15) << num3 << endl;
```

What will the output be?