Chapter 3 – CS1A Review – P2

Input / Output

Review

1. A compiler translates code written in a ______________________ language into ____________________ language.

2. T/F Syntax is the rules that dictate the meaning attached to instructions in a programming language.

3. A ________ is the name of a location in memory that has a data value that may be changed.

4. Values for these identifiers are obtained at ___________ time and the amount of memory to be reserved is determined at ________.

5. A ___________ is the name of a location in memory that has a data value that may not be changed.

6. Values for these identifiers are obtained at ______________ time and the amount of memory to be reserved is determined at _____.

7. The documentation next to the declarations for variables and named constants is called the _______________.

8. It tells the reader ________ and ________ their values are obtained.

9. What is an unsigned int?

10. What does the data type tell the compiler (2 things)?

11. What are the differences between how a constant is declared between a variable?

12. Explain the difference between the following declarations

```
char charVal;
char strVal[10];
```

2 of 56

Input & Ouput to the screen

cin >> *variable* → for input

places the value into a memory location

cout << *"what you want to output";*

→ for output

prompts the user for data (could also be a variable or expression)

They usually work in pairs.

3 of 56

Program – Basic Structure

- ***Pre-processor Directive(s)*** - information the program needs (a list of all necessary header files used in the program)
- ***Heading*** - int main ()
  - *functions by definition return a value*
  - (the above heading indicates that this function will an int)
- ***main function*** - {

```
named constant declarations
variable declarations
executable statements
return 0;
}
```

For Example

No ";" not a C++ statement

```
#include <iostream>
#include <iomanip>
using namespace std;

int main( )
{
    float floatVal;
    int   intVal;

    cout << "Enter a floating point number: ";
    cin  >> floatVal;

    cout << "Enter an integer: ";
    cin  >> intVal;
    cout << f                ecision(2);

    cout << "\n\nThe floating point value is " << setw(8) << floatVal << endl;
    cout << "The integer value is " << setw(6) << intVal << endl;
    return 0;
```

preprocessor directives for I/O

Tells the compiler to use predefined Standard C++ functions, variables, & classes

Don't put void
Write it just like this

Remember this ends a program statement

All C++ programs must start with this → It tells the compiler where to start

Declares a float variable

Declares an integer variable

These define a code block You must start with { and end with }

prompts for an input

Reads the input into floatVal

prompts for the 2nd input

Reads the input into intVal

Note the indent

These are output manipulators that **Formats floating point values - more on this later**

Tells the OS that the program terminated properly

Breaking it down

#include <iostream>

#include <iomanip>

- Pre-processor Directives
  - Tell the pre-processor that we want to use i/o functions so we #include them in our code
  - We need iostream to use cin / cout statements
  - We need iomanip to use output manipulators which dictate how our output will be formatted

using namespace std;

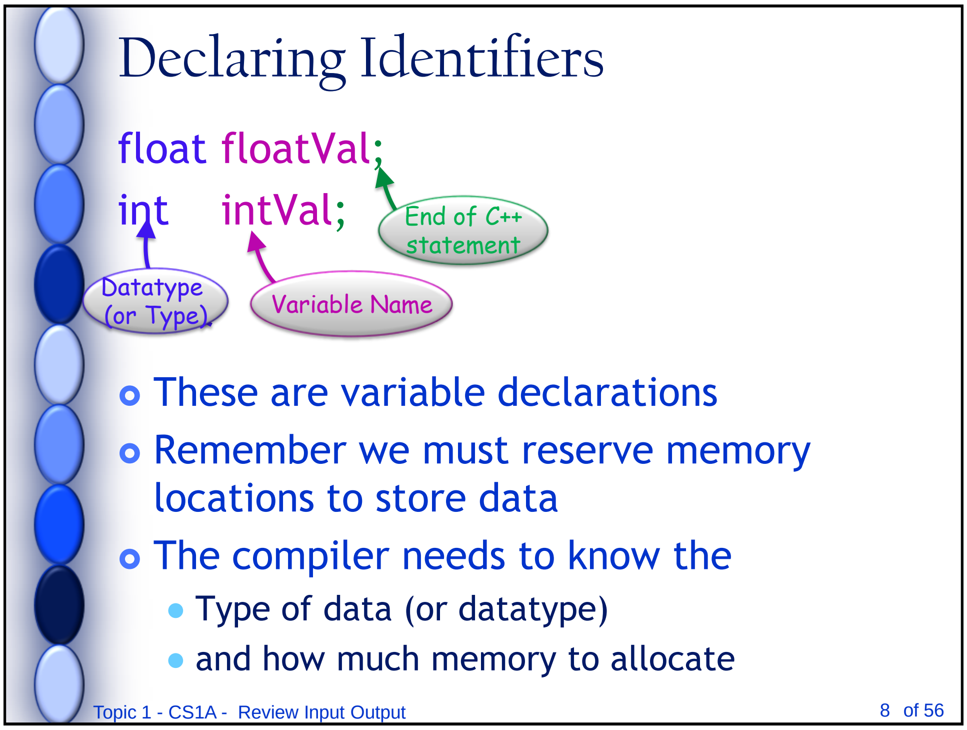
- Tells the compiler that we want to use all the standard C++ functions, variables and classes
- Functions are small code segments that we use to build our program

Main

```
int main ()
{
    body of function (i.e. program statements)
    return 0;
}
```

All executable statements are here

- Program execution begins with this function
- All C++ programs must have this function
- MUST BE an int
  - The book has an error → MAIN CANNOT BE VOID
- Must start with { and end with }
- Must have return 0; as last statement
  - This returns the value 0 to the system so it knows the program completed properly



Declaring Identifiers

```
float floatVal;
int   intVal;
```

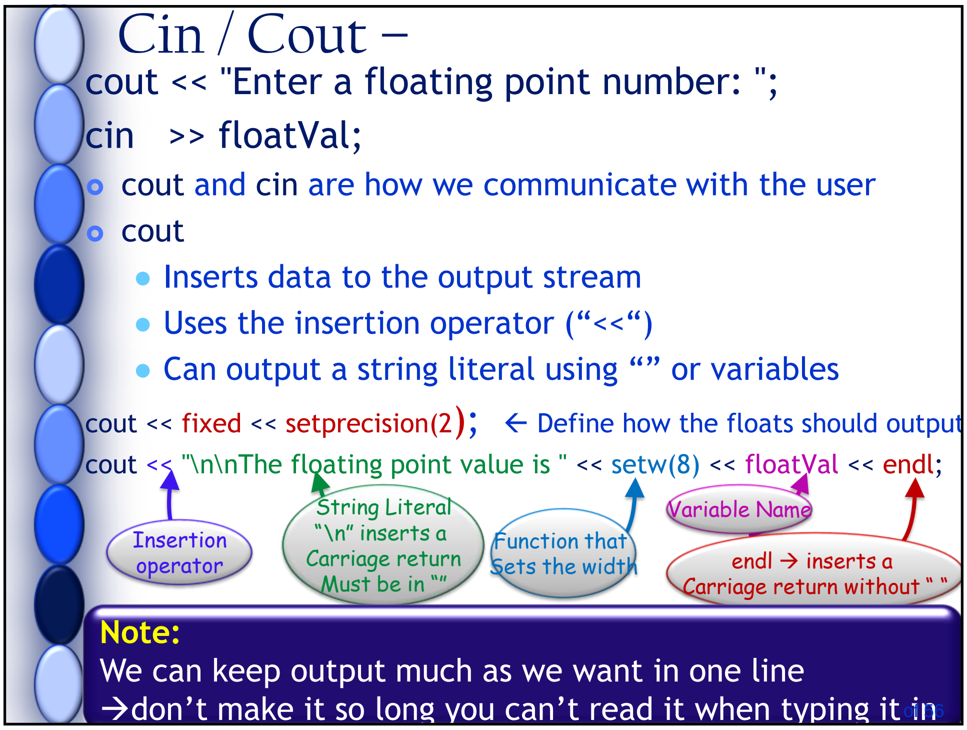
End of C++ statement

Datatype (or Type)

Variable Name

- These are variable declarations
- Remember we must reserve memory locations to store data
- The compiler needs to know the
  - Type of data (or datatype)
  - and how much memory to allocate

Topic 1 - CS1A - Review Input Output 8 of 56



Cin / Cout –

```
cout << "Enter a floating point number: ";
cin   >> floatVal;
```

- cout and cin are how we communicate with the user
- cout
  - Inserts data to the output stream
  - Uses the insertion operator (“<<“)
  - Can output a string literal using “” or variables

```
cout << fixed << setprecision(2);   ← Define how the floats should output
cout << "\n\nThe floating point value is " << setw(8) << floatVal << endl;
```

Insertion operator

String Literal "\n" inserts a Carriage return Must be in “”

Function that Sets the width

Variable Name

endl → inserts a Carriage return without “ ”

Note:
We can keep output much as we want in one line
→don’t make it so long you can’t read it when typing it in

CIN

```
cin >> floatVal;
```

Extraction operator

MUST BE A VARIABLE

End of C++ statement

- cin extracts information from the input buffer using the extraction operator “>>“
- In this example the contents of the input buffer are put into the variable floatVal

- YOU CAN ONLY HAVE VARIABLES on the RIGHT OF A CIN STATEMENT
  - → because the data must be stored somewhere

Note:
Generally speaking we want to pair cin with a cout so that the user knows the program is waiting for an input

Input in C++

cin	cin.get
Extraction operator (>>)	cin.ignore
cin.getline	cin.width

Basic Input in C++

We want to be able to extract data into a variable

→ This allows us to execute our programs with values that are not predefined

- cin - is a predefined variable in c++ that allows us extract input directly from the user

- There are many ways to extract input using the cin variable

- Anytime an input command is executed the program will wait for an input to be read in.

So far,

- We have discussed the extraction operator (>>). This ideal for reading in numbers. but not so effective for reading in strings, characters or whitespace

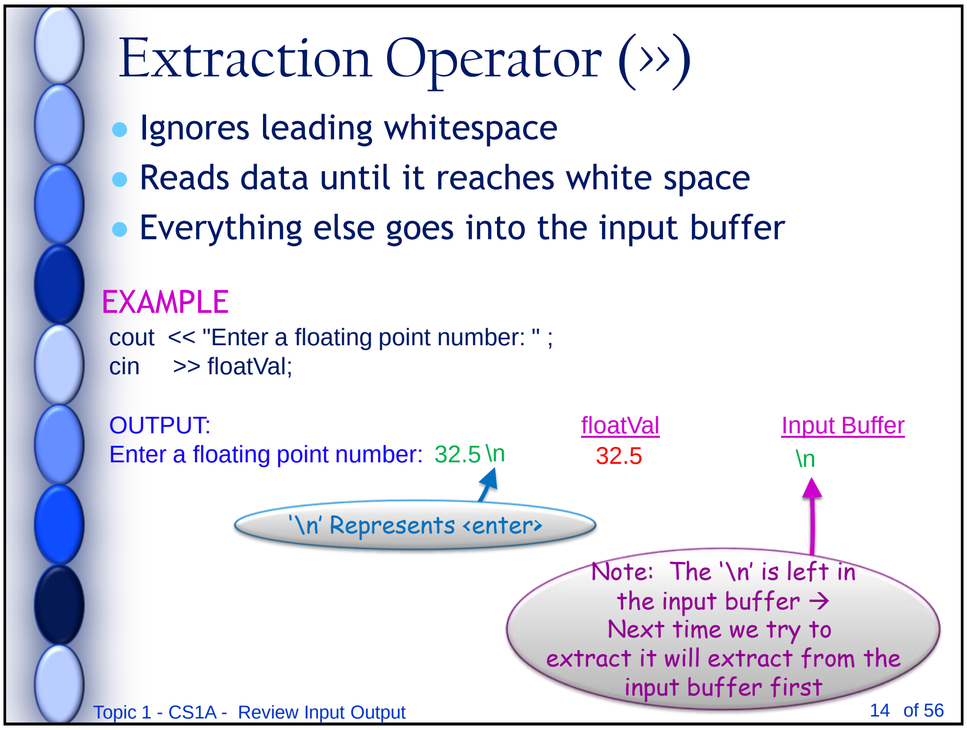
The Extraction operator (>>)

Syntax

cin >> *variable* >> *variable* ...;

- We use the extraction operator to read in numerical data
  - Only variables to the right of the extraction operator
    - Remember the purpose is to store data in a memory location
    - Variables are the only memory locations that can be modified at runtime
  - You *can* use more than one variable in one statement
  - Input should MATCH the data type
    - prompt appropriately
  - Input is extracted from the buffer

    → unless the buffer is empty then it reads in from the keyboard



Extraction Operator (>>)

- Ignores leading whitespace
- Reads data until it reaches white space
- Everything else goes into the input buffer

EXAMPLE

```
cout  << "Enter a floating point number: " ;
cin     >> floatVal;
```

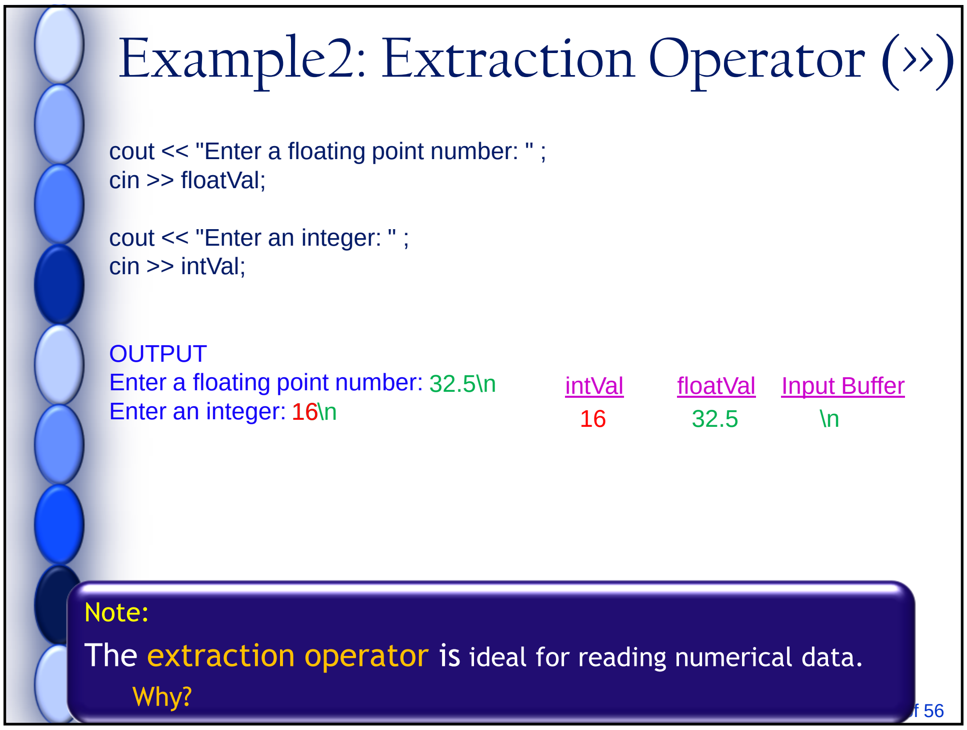
OUTPUT:

```
Enter a floating point number:  32.5 \n
```

floatVal	Input Buffer
32.5	\n

'\n' Represents <enter>

Note: The '\n' is left in the input buffer → Next time we try to extract it will extract from the input buffer first



Example2: Extraction Operator (>>)

```
cout << "Enter a floating point number: " ;
cin >> floatVal;

cout << "Enter an integer: " ;
cin >> intVal;
```

OUTPUT

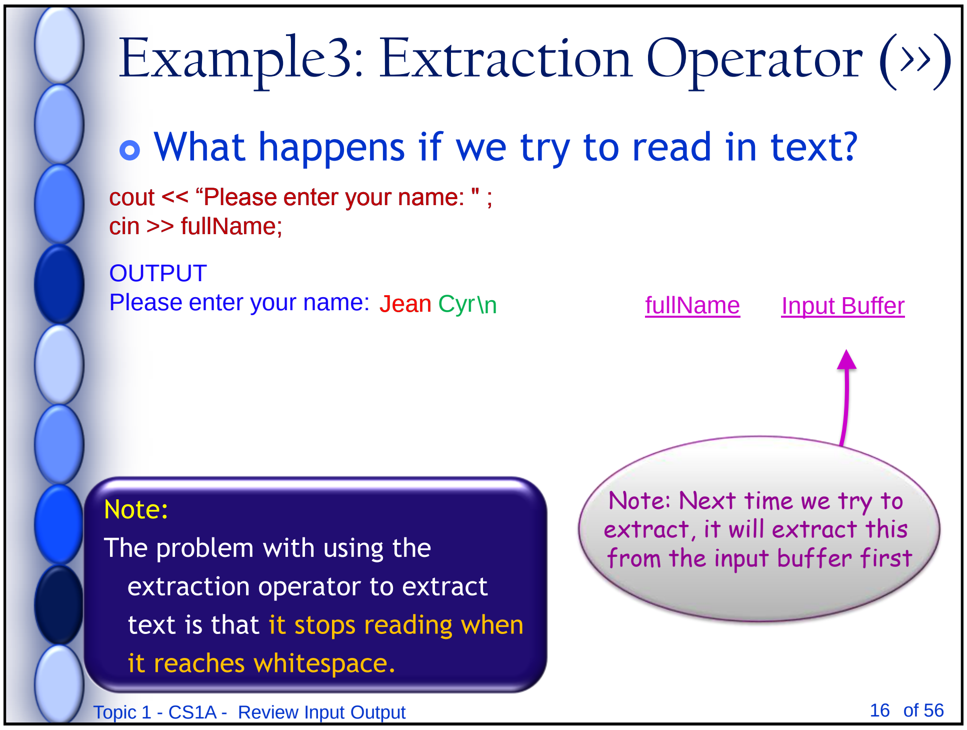
```
Enter a floating point number: 32.5\n
Enter an integer: 16\n
```

intVal	floatVal	Input Buffer
16	32.5	\n

Note:

The extraction operator is ideal for reading numerical data.

Why?



Example3: Extraction Operator (>>)

- What happens if we try to read in text?

```
cout << "Please enter your name: " ;
cin >> fullName;
```

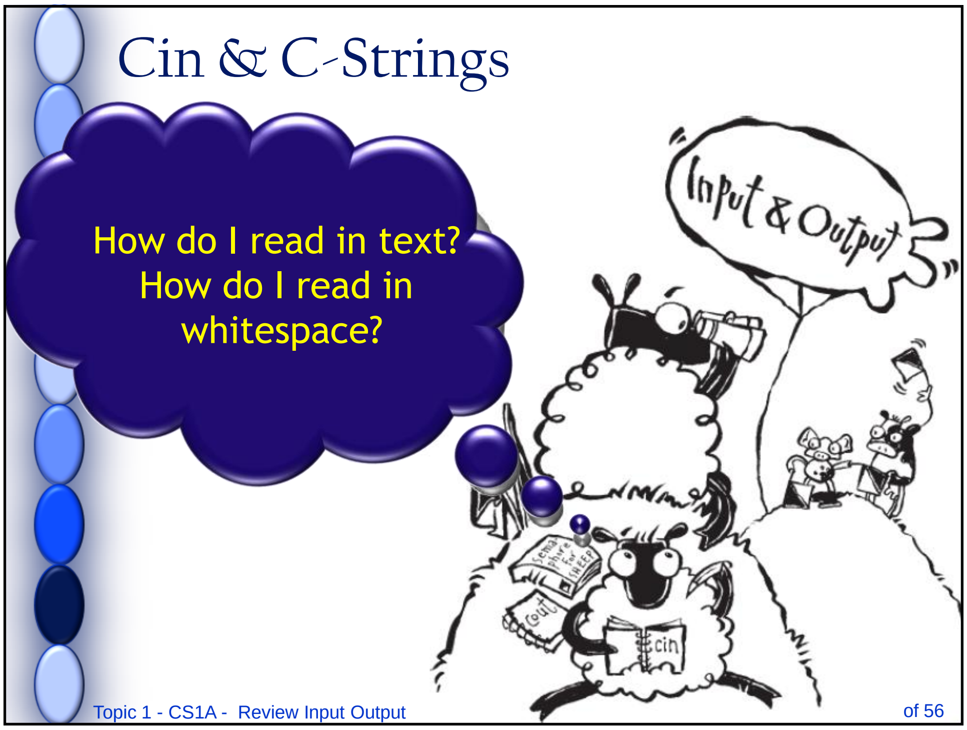
OUTPUT

```
Please enter your name: Jean Cyr\n
```

fullName   Input Buffer

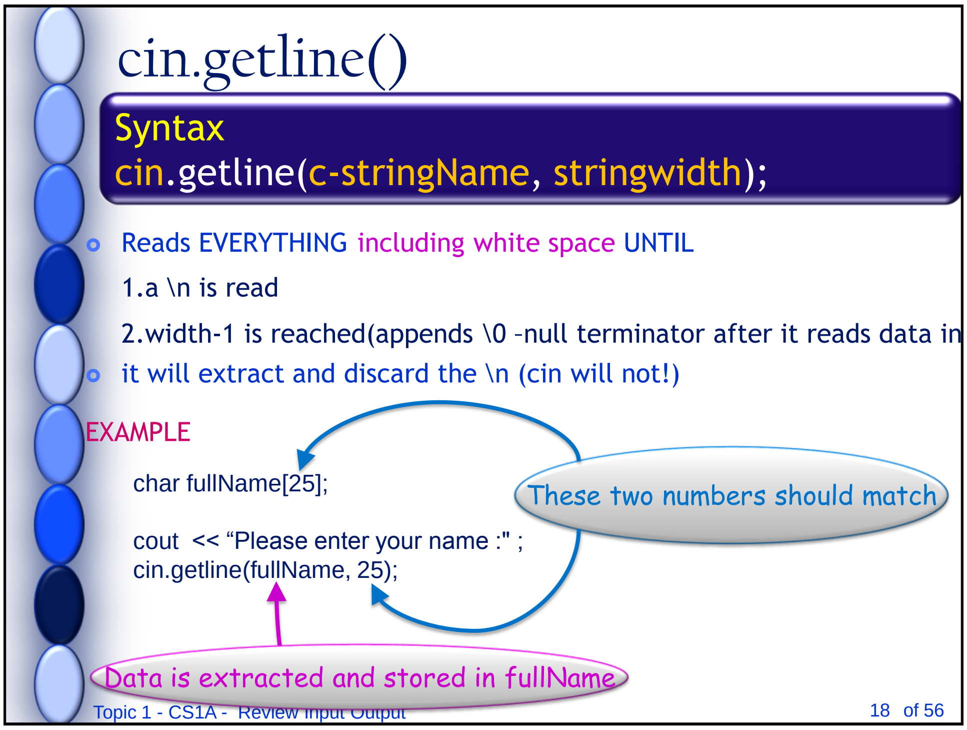
Note: Next time we try to extract, it will extract this from the input buffer first

Note:
The problem with using the extraction operator to extract text is that it stops reading when it reaches whitespace.



Cin & C-Strings

How do I read in text?
How do I read in whitespace?



cin.getline()

Syntax
cin.getline(c-stringName, stringwidth);

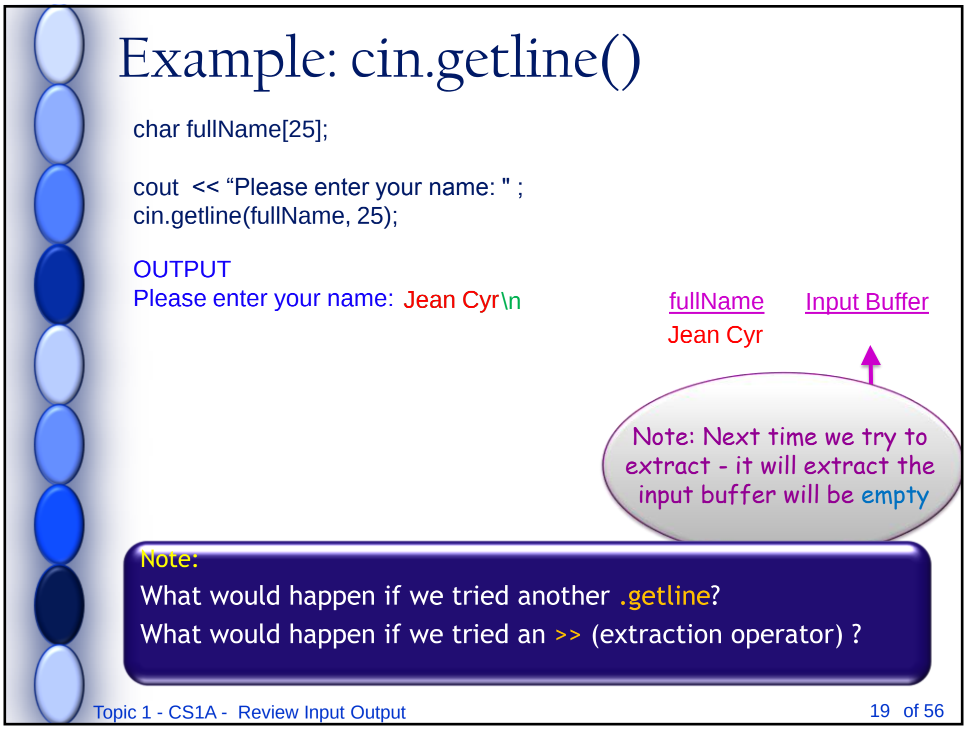
- Reads EVERYTHING including white space UNTIL
  1.a \n is read
  2.width-1 is reached(appends \0 -null terminator after it reads data in
- it will extract and discard the \n (cin will not!)

EXAMPLE

```
char fullName[25];

cout  << “Please enter your name :" ;
cin.getline(fullName, 25);
```

These two numbers should match

Data is extracted and stored in fullName



Example: cin.getline()

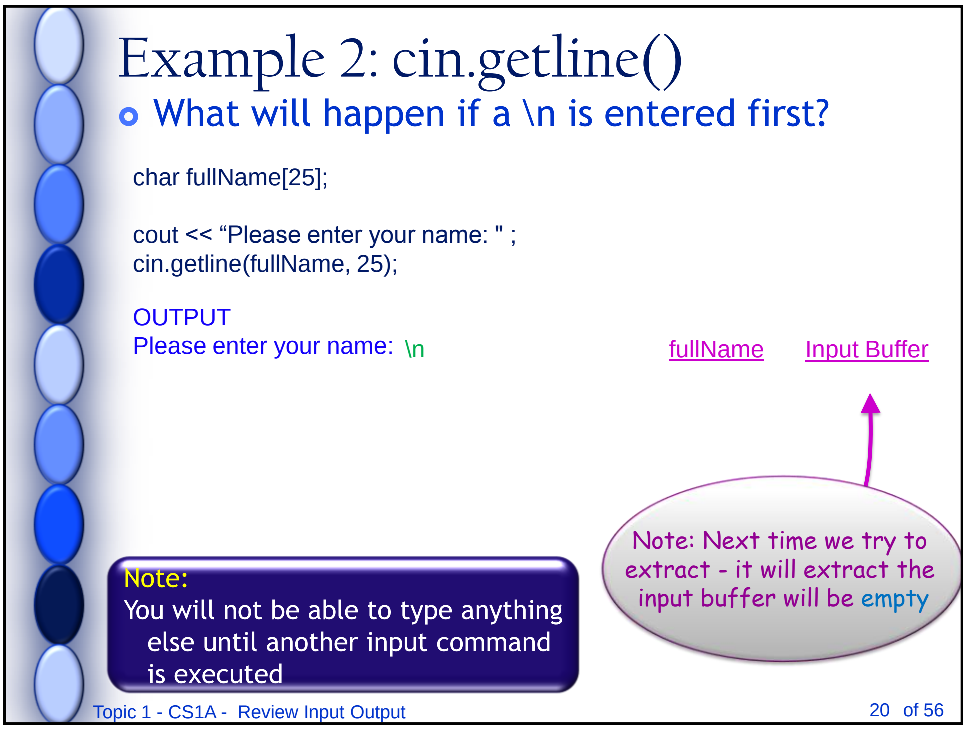
```
char fullName[25];

cout  << “Please enter your name: " ;
cin.getline(fullName, 25);
```

OUTPUT
Please enter your name: Jean Cyr\n

fullName	Input Buffer
Jean Cyr	

Note: Next time we try to extract - it will extract the input buffer will be empty

Note:
What would happen if we tried another .getline?
What would happen if we tried an >> (extraction operator) ?



Example 2: cin.getline()

- What will happen if a \n is entered first?

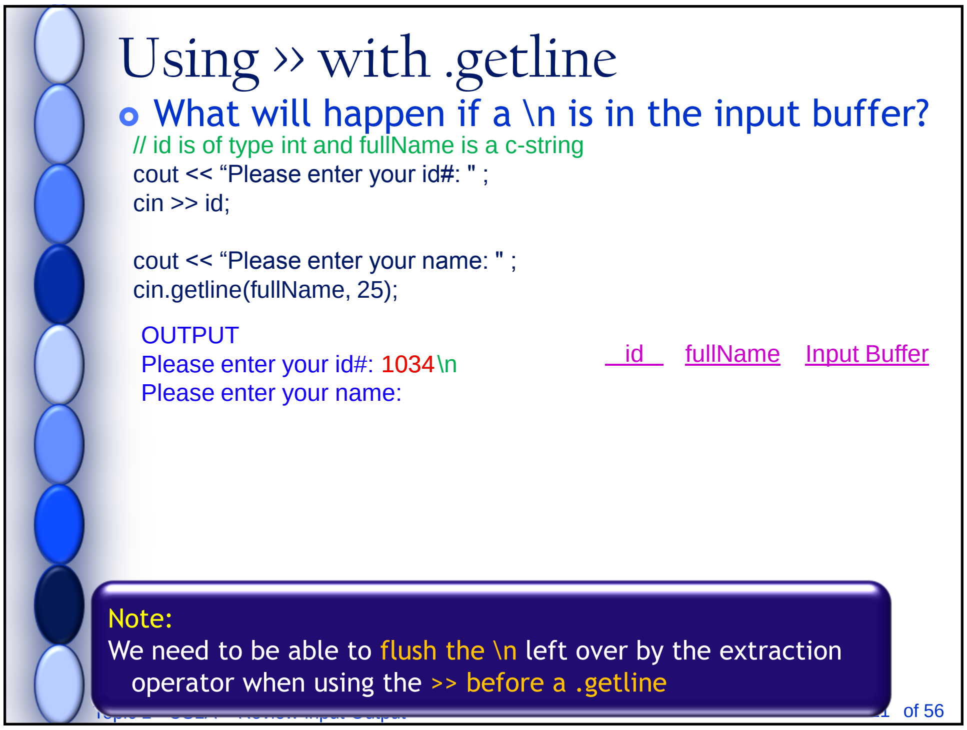
```
char fullName[25];

cout << “Please enter your name: " ;
cin.getline(fullName, 25);
```

OUTPUT
Please enter your name: \n

fullName   Input Buffer

Note: Next time we try to extract - it will extract the input buffer will be empty

Note:
You will not be able to type anything else until another input command is executed



Using >> with .getline

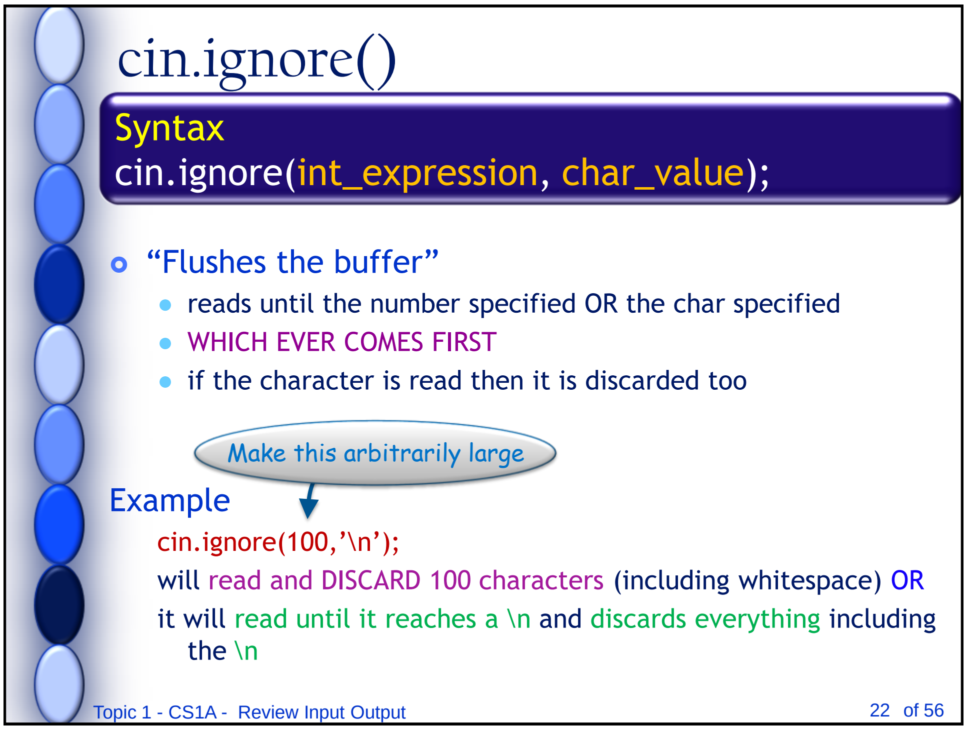
- What will happen if a \n is in the input buffer?

```
// id is of type int and fullName is a c-string
cout << “Please enter your id#: " ;
cin >> id;

cout << “Please enter your name: " ;
cin.getline(fullName, 25);
```

OUTPUT
Please enter your id#: 1034\n
Please enter your name:

id   fullName   Input Buffer

Note:
We need to be able to flush the \n left over by the extraction operator when using the >> before a .getline



cin.ignore()

Syntax
cin.ignore(int_expression, char_value);

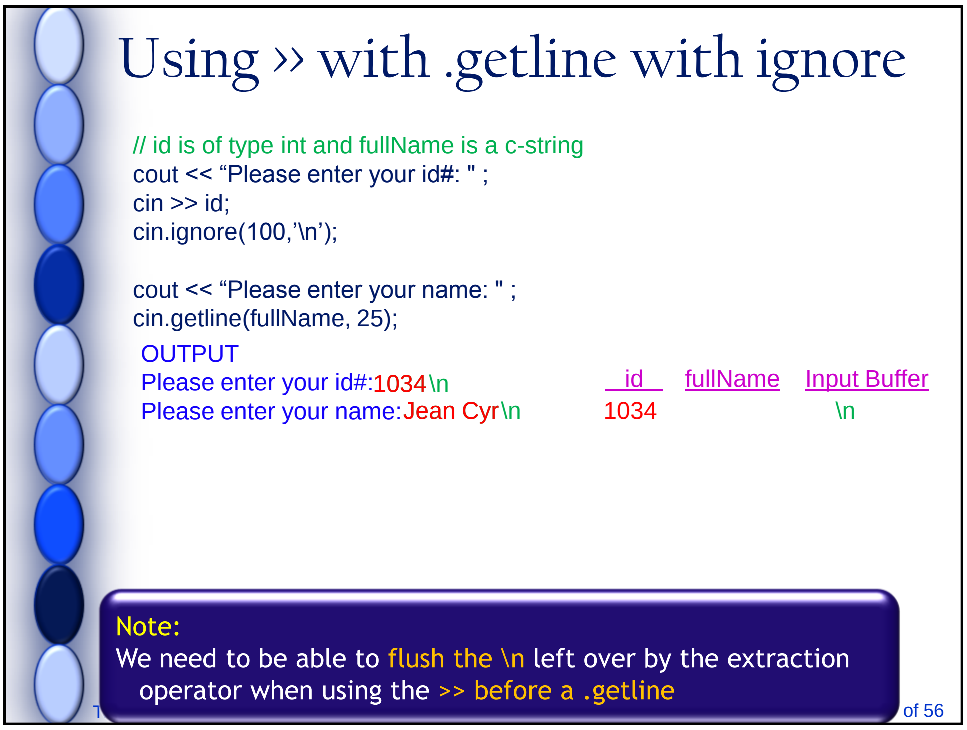
- “Flushes the buffer”
  - reads until the number specified OR the char specified
  - WHICH EVER COMES FIRST
  - if the character is read then it is discarded too

Make this arbitrarily large

Example

```
cin.ignore(100,’\n’);
```

will read and DISCARD 100 characters (including whitespace) OR
it will read until it reaches a \n and discards everything including the \n



Using ›› with .getline with ignore

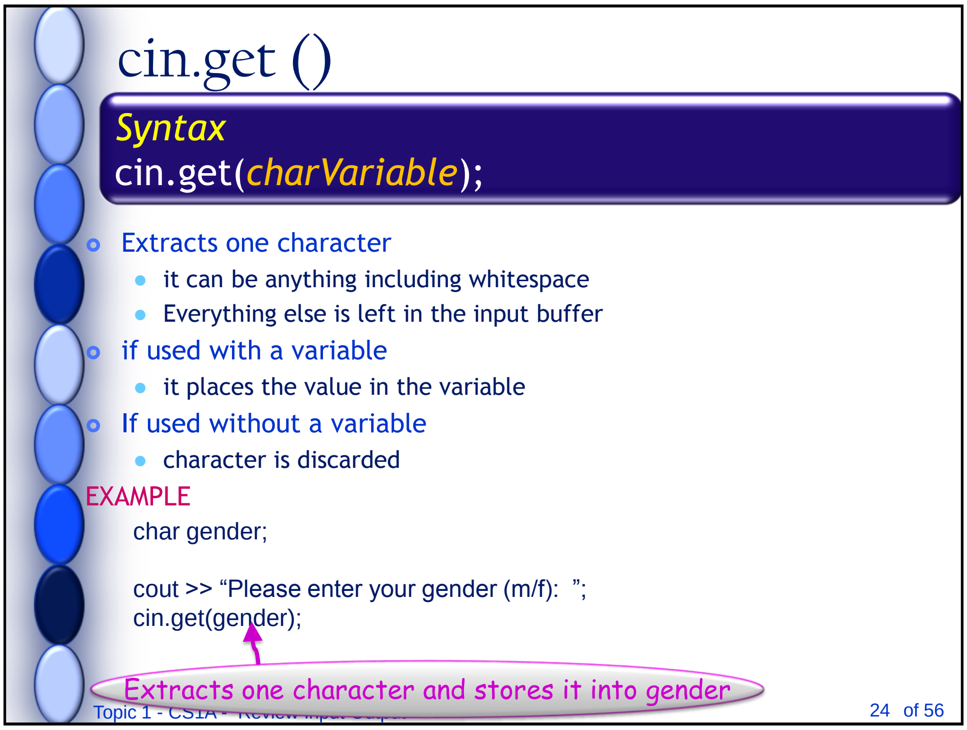
```
// id is of type int and fullName is a c-string
cout << “Please enter your id#: " ;
cin >> id;
cin.ignore(100,’\n’);

cout << “Please enter your name: " ;
cin.getline(fullName, 25);
```

OUTPUT

```
Please enter your id#:1034\n
Please enter your name:Jean Cyr\n
```

id	fullName	Input Buffer
1034		\n

Note:
We need to be able to flush the \n left over by the extraction operator when using the >> before a .getline



cin.get ()

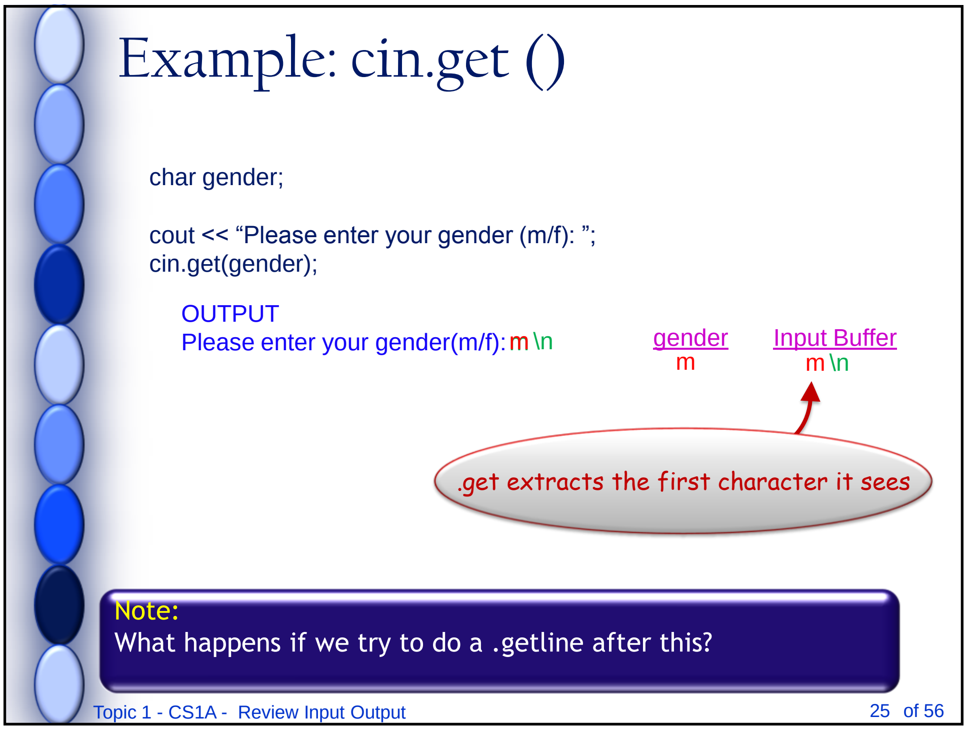
Syntax
cin.get(*charVariable*);

- Extracts one character
  - it can be anything including whitespace
  - Everything else is left in the input buffer
- if used with a variable
  - it places the value in the variable
- If used without a variable
  - character is discarded

EXAMPLE

```
char gender;

cout >> "Please enter your gender (m/f):  ";
cin.get(gender);
```

Extracts one character and stores it into gender



Example: cin.get ()

```
char gender;

cout << "Please enter your gender (m/f): ";
cin.get(gender);
```

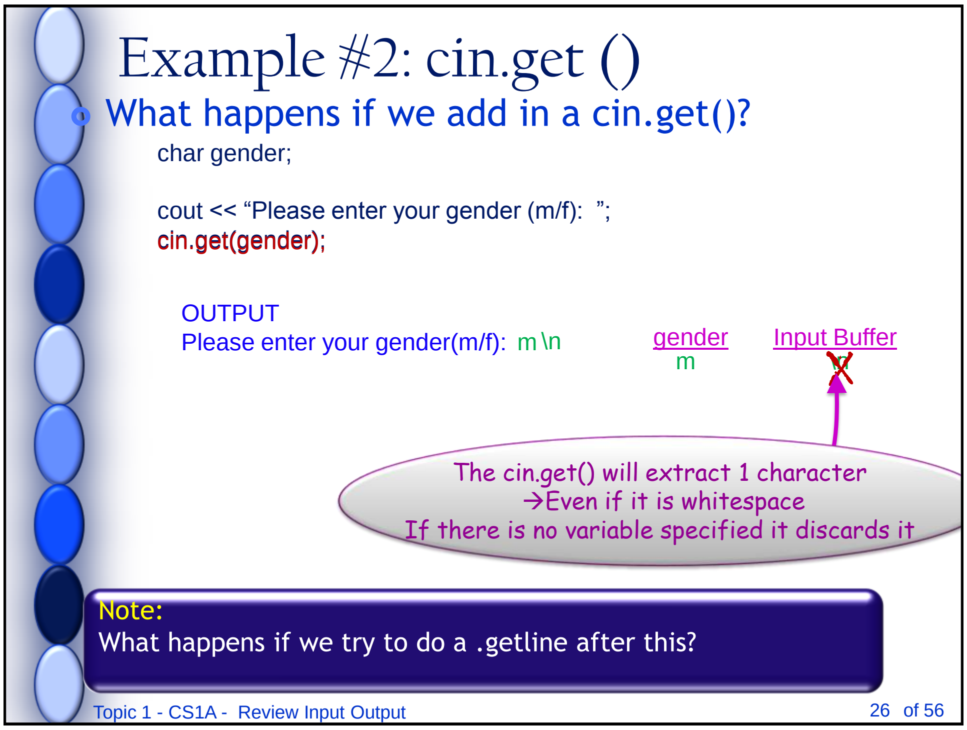
OUTPUT

```
Please enter your gender(m/f): m \n
```

gender	Input Buffer
m	m \n

.get extracts the first character it sees

Note:
What happens if we try to do a .getline after this?



Example #2: cin.get ()

- What happens if we add in a cin.get()?

```
char gender;

cout << "Please enter your gender (m/f):  ";
cin.get(gender);
```

OUTPUT
Please enter your gender(m/f): m \n

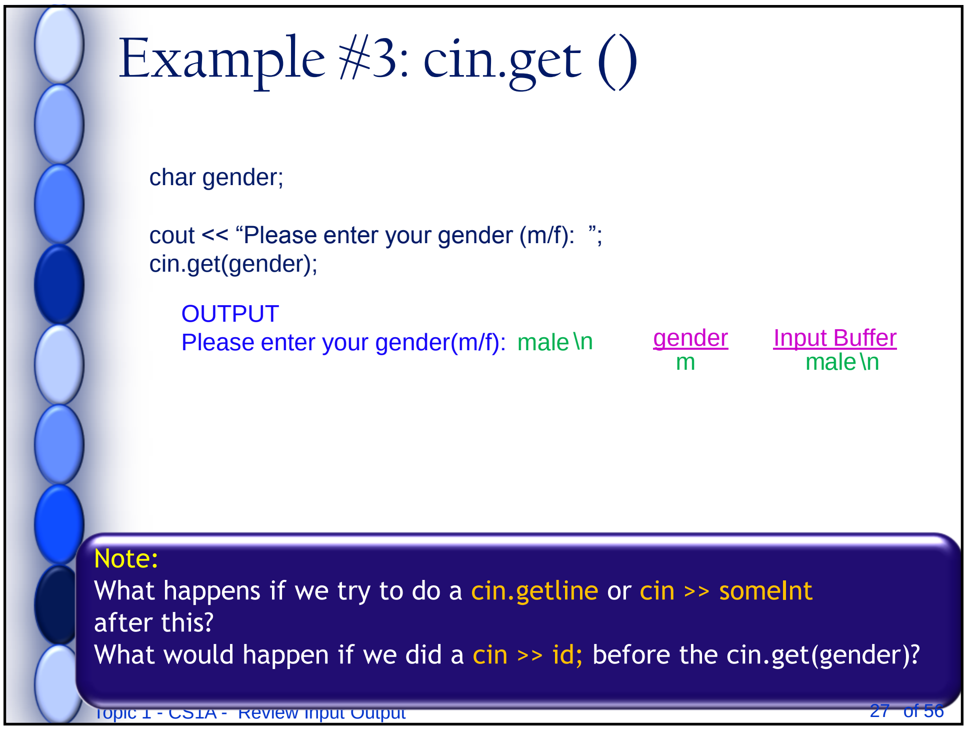
gender
m

Input Buffer
$\cancel{m}$

The cin.get() will extract 1 character
→Even if it is whitespace
If there is no variable specified it discards it

Note:
What happens if we try to do a .getline after this?

Topic 1 - CS1A - Review Input Output   26 of 56



Example #3: cin.get ()

```
char gender;

cout << "Please enter your gender (m/f):  ";
cin.get(gender);
```

OUTPUT
Please enter your gender(m/f): male \n

gender
m

Input Buffer
male \n

Note:
What happens if we try to do a cin.getline or cin >> someInt after this?
What would happen if we did a cin >> id; before the cin.get(gender)?

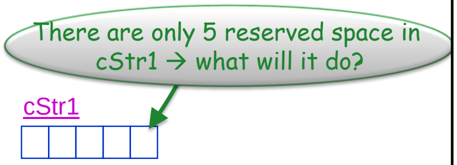
Topic 1 - CS1A - Review Input Output   27 of 56

Reading in strings using the (>>) Problem #2

- What if I don't need to account for spaces

```
char cStr1[5];
char cStr2[5];

cout << "Enter string#1:  ";
cin >> cStr1;
```

There are only 5 reserved space in cStr1 → what will it do?



```
OUTPUT
Enter string #1: abcdefghijkl\n
```

Input Buffer
abcdefghijkl\n

Controlling cin with cin.width() and setw()

Syntax
cin.width(stringWidth); or setw(n);

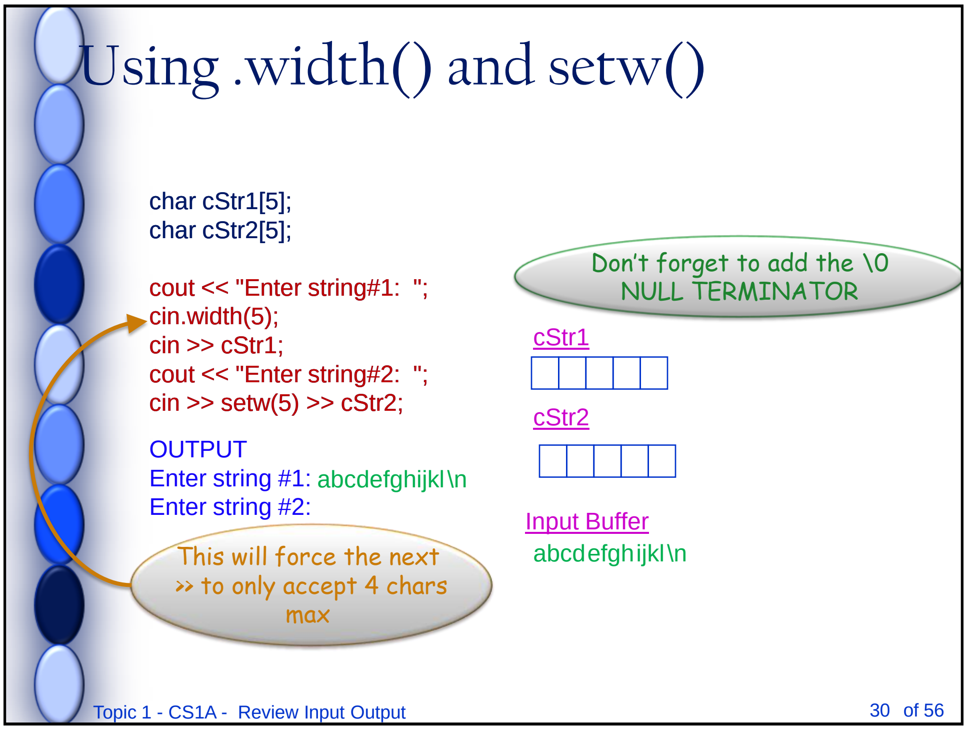
- Limit the input that is stored in memory to n spaces
- don't forget to count the null terminator (\0)

Example

```
char userString[5];
cout << "Enter a string: ";
cin.width(5);                  // output will be the same  as with
cin>> userString;              // cin>>setw(5)>>userString;
cout << "\n\n" << userString;
```

```
OUTPUT
Enter a string: abcdefghij

abcd
```

Note: you have to use these every time you want to limit what is read in from the buffer/keyboard

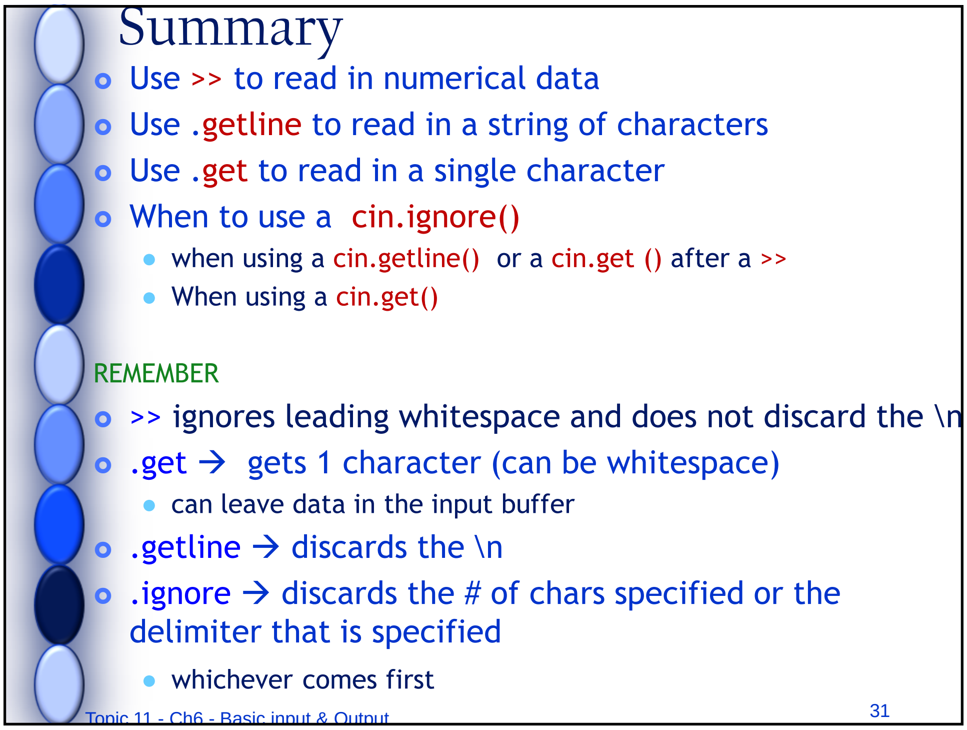


Using .width() and setw()

```
char cStr1[5];
char cStr2[5];

cout << "Enter string#1:  ";
cin.width(5);
cin >> cStr1;
cout << "Enter string#2:  ";
cin >> setw(5) >> cStr2;
```

OUTPUT
```
Enter string #1: abcdefghijkl\n
Enter string #2:
```

This will force the next >> to only accept 4 chars max
Don't forget to add the \0 NULL TERMINATOR
cStr1
cStr2
Input Buffer
abcdefghijkl\n



Summary

- Use >> to read in numerical data
- Use .getline to read in a string of characters
- Use .get to read in a single character
- When to use a cin.ignore()
  - when using a cin.getline() or a cin.get () after a >>
  - When using a cin.get()

REMEMBER

- >> ignores leading whitespace and does not discard the \n
- .get → gets 1 character (can be whitespace)
  - can leave data in the input buffer
- .getline → discards the \n
- .ignore → discards the # of chars specified or the delimiter that is specified
  - whichever comes first

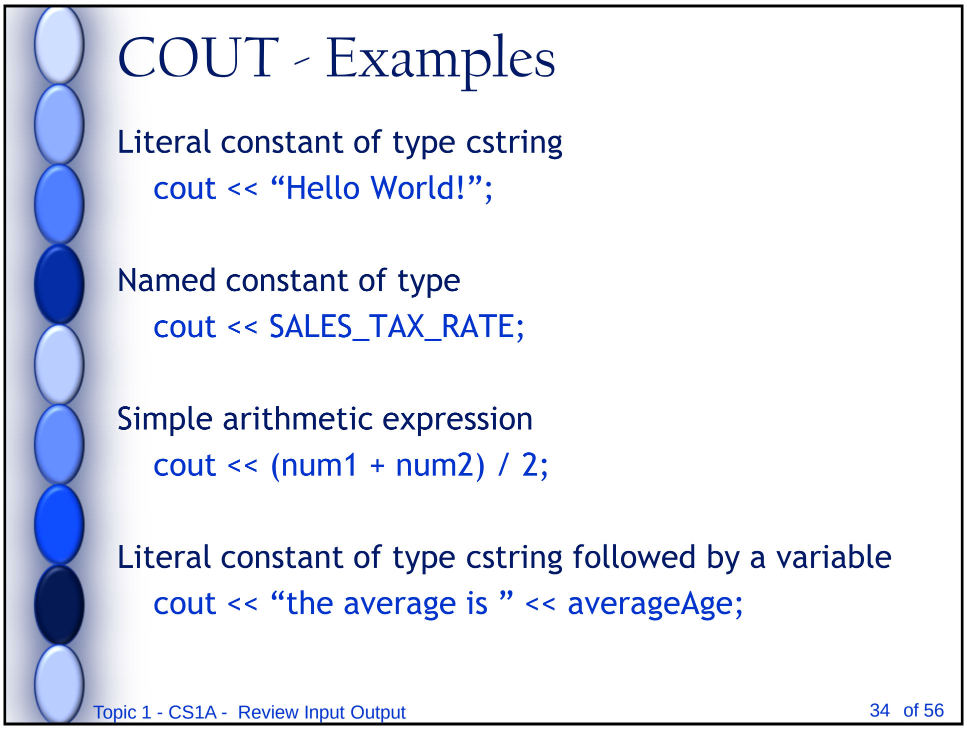
Output

How to output to the screen

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**

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



COUT - Examples

Literal constant of type cstring

```
cout << “Hello World!”;
```

Named constant of type

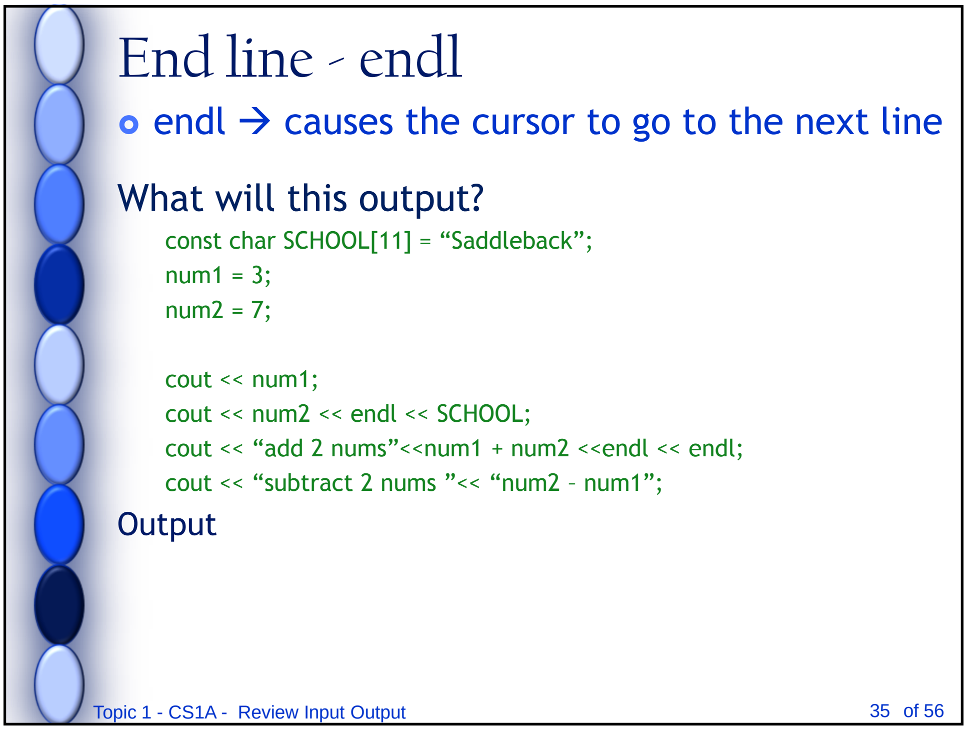
```
cout << SALES_TAX_RATE;
```

Simple arithmetic expression

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

Literal constant of type cstring followed by a variable

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



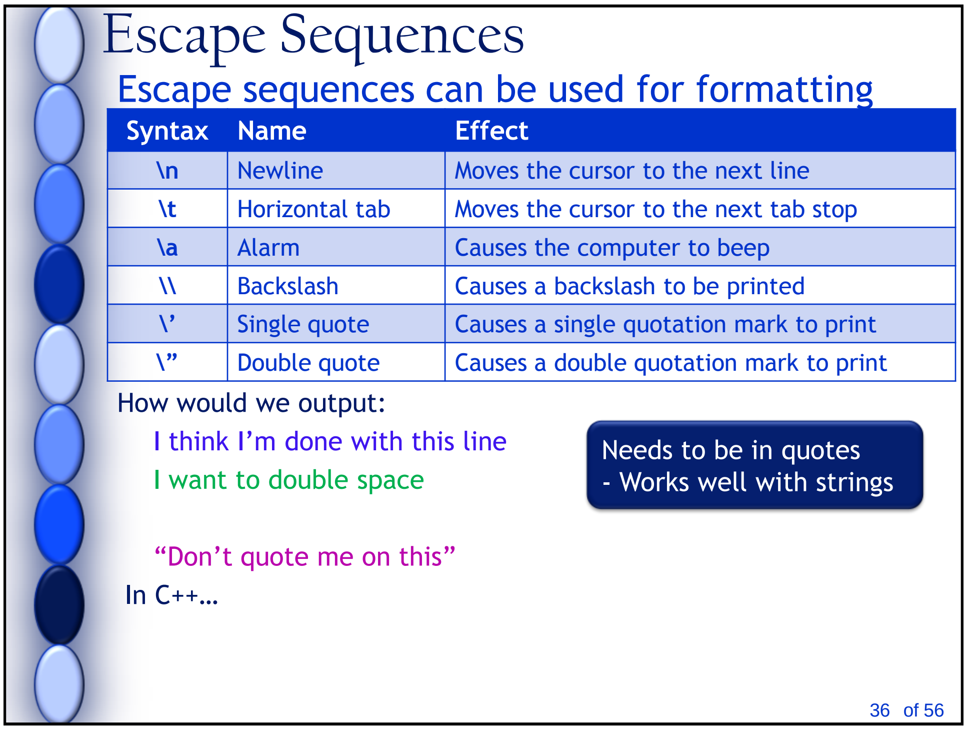
End line - endl

- 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



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

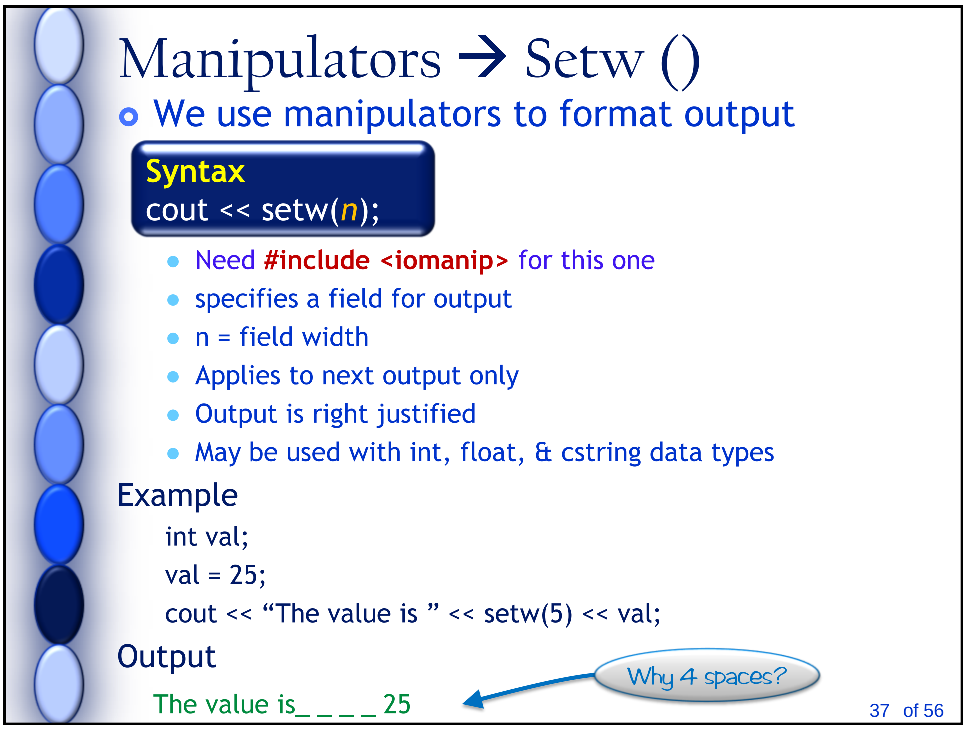
I want to double space

“Don’t quote me on this”

In C++…

Needs to be in quotes
- Works well with strings

36 of 56



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

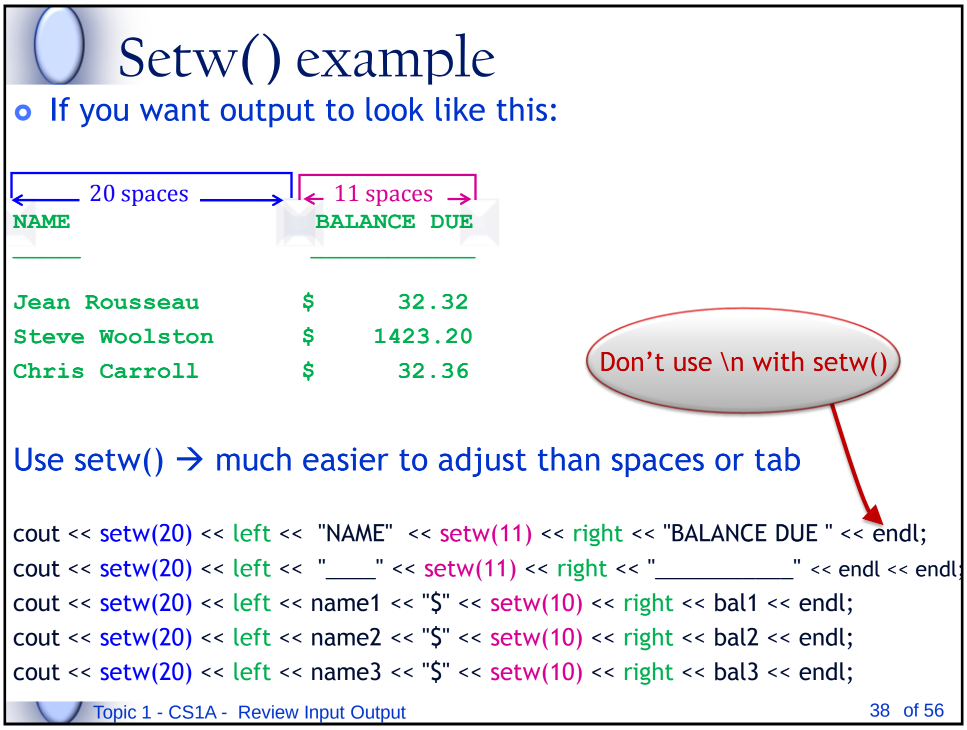
```
int val;
val = 25;
cout << “The value is ” << setw(5) << val;
```

Output

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

Why 4 spaces?

37 of 56



Setw() example

- If you want output to look like this:

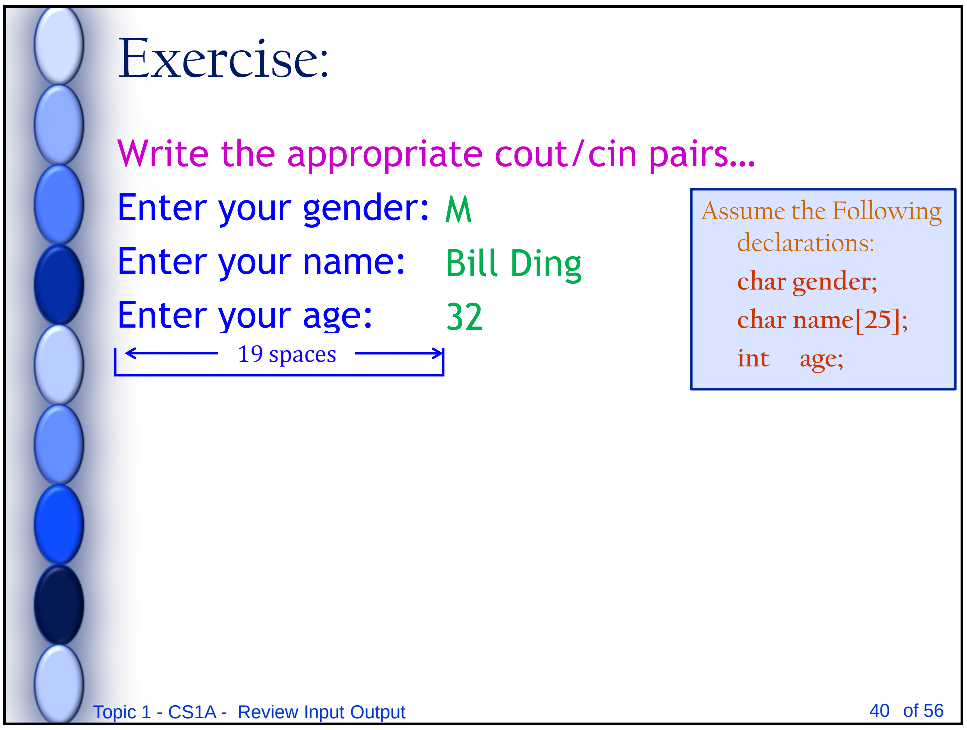
20 spaces | 11 spaces

```
NAME                 BALANCE DUE
____                 ___________

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

Don't use \n with setw()

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

```
cout << setw(20) << left <<  "NAME"  << setw(11) << right << "BALANCE DUE " << endl;
cout << setw(20) << left <<  "____" << setw(11) << right << "___________" << endl << endl;
cout << setw(20) << left << name1 << "$" << setw(10) << right << bal1 << endl;
cout << setw(20) << left << name2 << "$" << setw(10) << right << bal2 << endl;
cout << setw(20) << left << name3 << "$" << setw(10) << right << bal3 << endl;
```



Exercise:

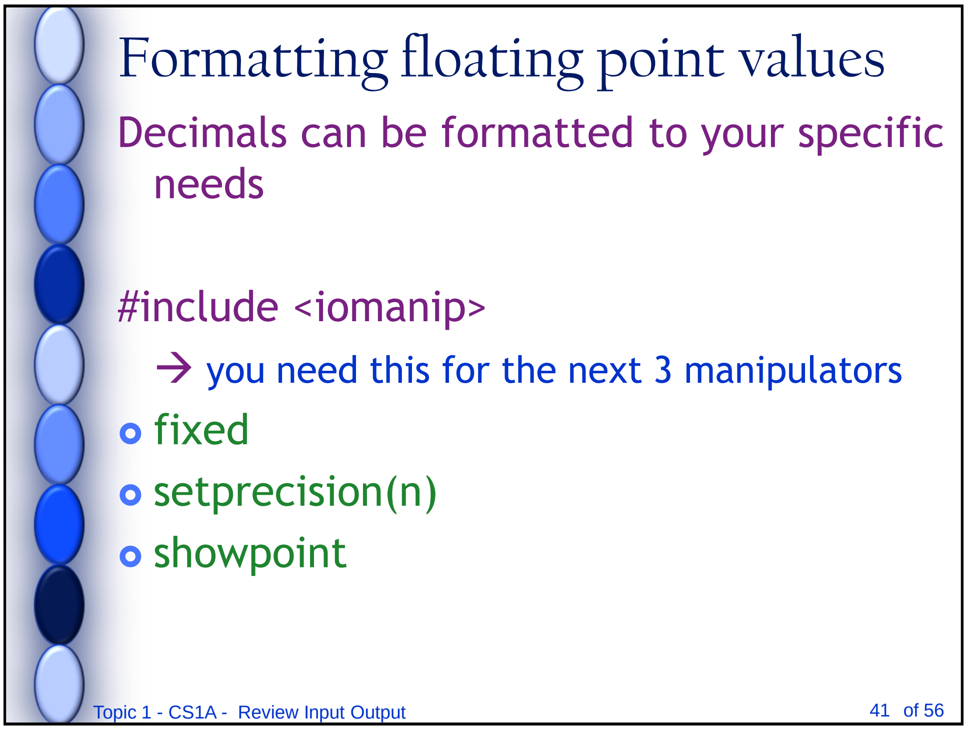
Write the appropriate cout/cin pairs…

```
Enter your gender: M
Enter your name:   Bill Ding
Enter your age:    32
```

19 spaces

Assume the Following declarations:

```
char gender;
char name[25];
int    age;
```



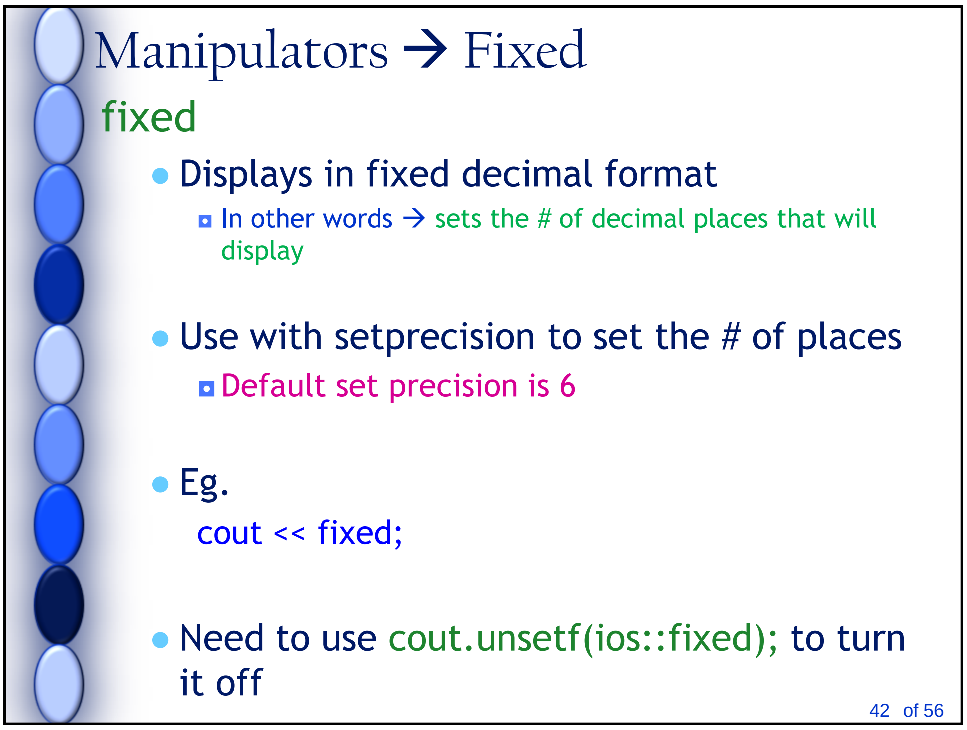
Formatting floating point values

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

```
{
    float val1;
    float val2;
    float 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;
}
```

default precision is set to 6

These will round

OUTPUT

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

Setprecision Example

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

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

default precision is set to 6

OUTPUT

```
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;
}
```

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

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;

 cout << showpoint;
 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;
}
```

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

OUTPUT

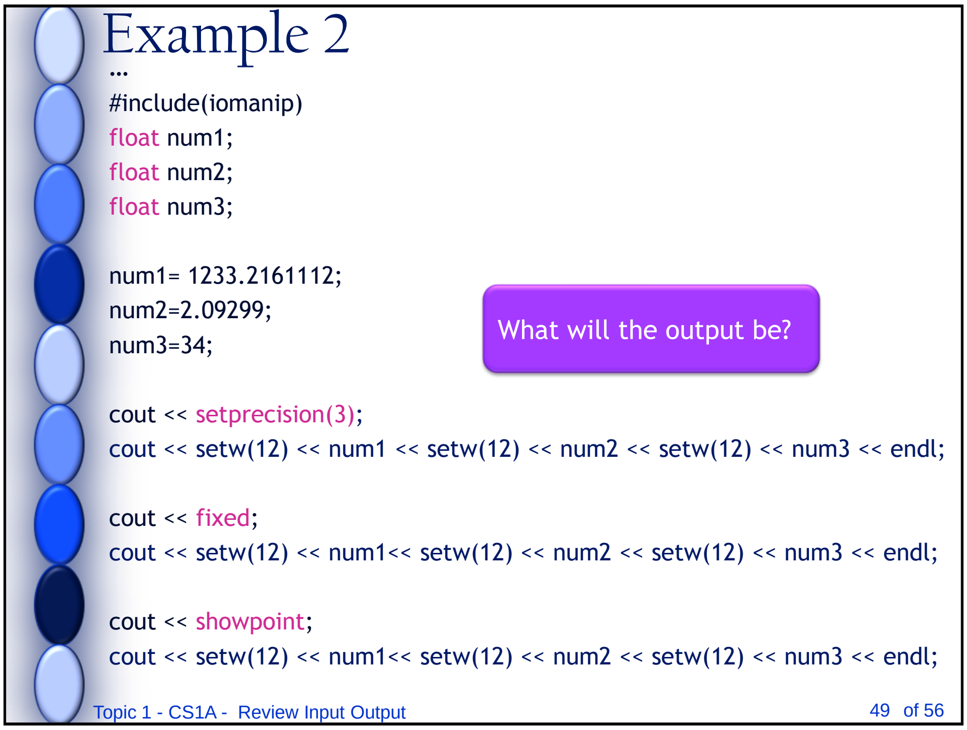
Set precision is w.r.t the # of digits

NOTE
Showpoint only effects floating point values that have a zero decimal value

Example 1

```
...
#include(iomanip)
float num1;
float num2;
float num3;

num1= 1233.2161112;
num2=2.09299;
num3=34;

cout << setw(12) << num1 << setw(12) << num2 << setw(12) << num3 << endl;

cout << showpoint;
cout << setw(12) << num1 << setw(12) << num2 << setw(12) << num3 << endl;
cout << setprecision(3);
cout << setw(12) << num1 << setw(12) << num2 << setw(12) << num3 << endl;
cout <<fixed;
cout << setw(12) << num1 << setw(12) << num2 << setw(12)<< num3 << endl;
```

What will the output be?

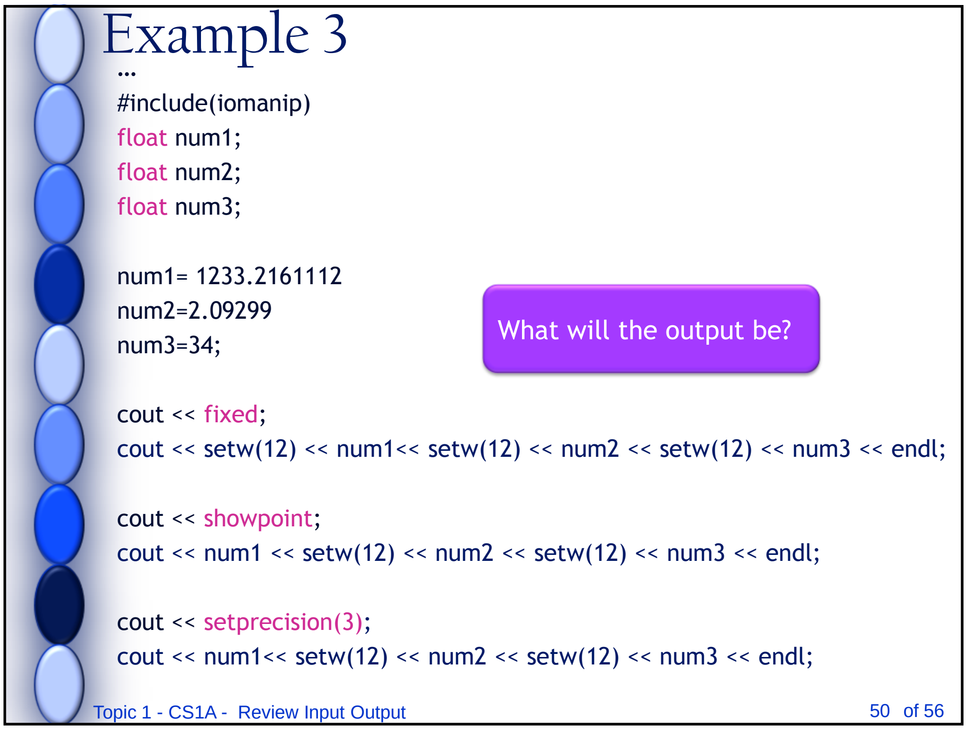


Example 2

```
…
#include(iomanip)
float num1;
float num2;
float num3;

num1= 1233.2161112;
num2=2.09299;
num3=34;

cout << setprecision(3);
cout << setw(12) << num1 << setw(12) << num2 << setw(12) << num3 << endl;

cout << fixed;
cout << setw(12) << num1<< setw(12) << num2 << setw(12) << num3 << endl;

cout << showpoint;
cout << setw(12) << num1<< setw(12) << num2 << setw(12) << num3 << endl;
```

What will the output be?

Topic 1 - CS1A - Review Input Output 49 of 56



Example 3

```
…
#include(iomanip)
float num1;
float num2;
float num3;

num1= 1233.2161112
num2=2.09299
num3=34;

cout << fixed;
cout << setw(12) << num1<< setw(12) << num2 << setw(12) << num3 << endl;

cout << showpoint;
cout << num1 << setw(12) << num2 << setw(12) << num3 << endl;

cout << setprecision(3);
cout << num1<< setw(12) << num2 << setw(12) << num3 << endl;
```

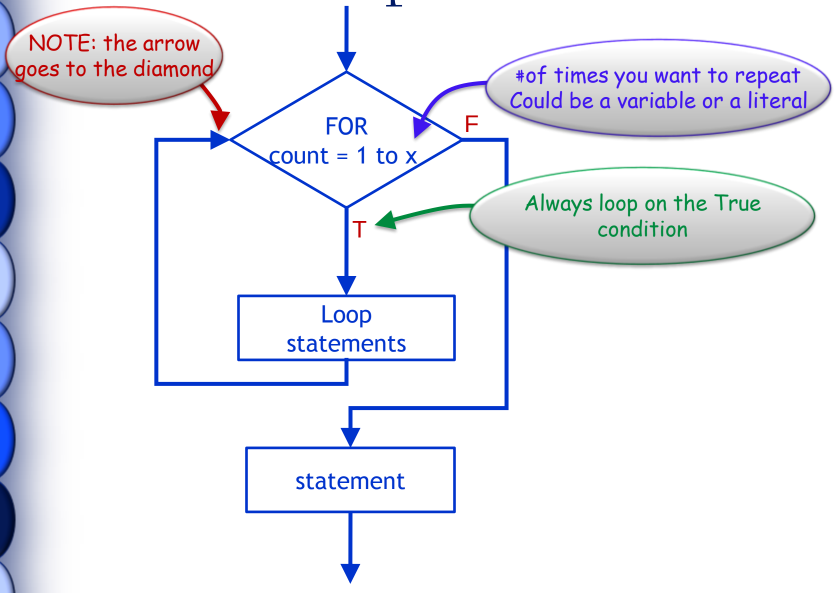
What will the output be?

Topic 1 - CS1A - Review Input Output 50 of 56

The For Loop

- For loop → repeats statements a set number of times
- Process:
  1. Initialization is executed
     → sets an initial value for the counter variable
  2. condition is checked
     If it is true the loop executes the loop instructions
     (that is the instructions that are to be repeated)
     else (if it is false) the loop exits
  3. The LCV is increased by the amount specified.
     Loop to step 2.

Flowchart: For loop


NOTE: the arrow goes to the diamond
#of times you want to repeat
Could be a variable or a literal
FOR
count = 1 to x
F
T
Always loop on the True condition
Loop statements
statement

For Loop Example

initialization check change

```
for(count = 1; count <= 3; count = count + 1)
{
   cout << “Enter Name: ”;
   cin.getline(username,25);

   cout << “Enter Age: “;
   cin >> age;
   cin.ignore(10000,’\n’);      //flush the buffer

   cout << “Press enter to continue”;
   cin.get();                  //get the \n off the buffer
}
```

getline is only for strings

Is this our best option?

53 of 56

Announcements

- Assignment #1 – input/output has been posted
- Please print code and output with line #s
- Output should be first
- Print in eclipse