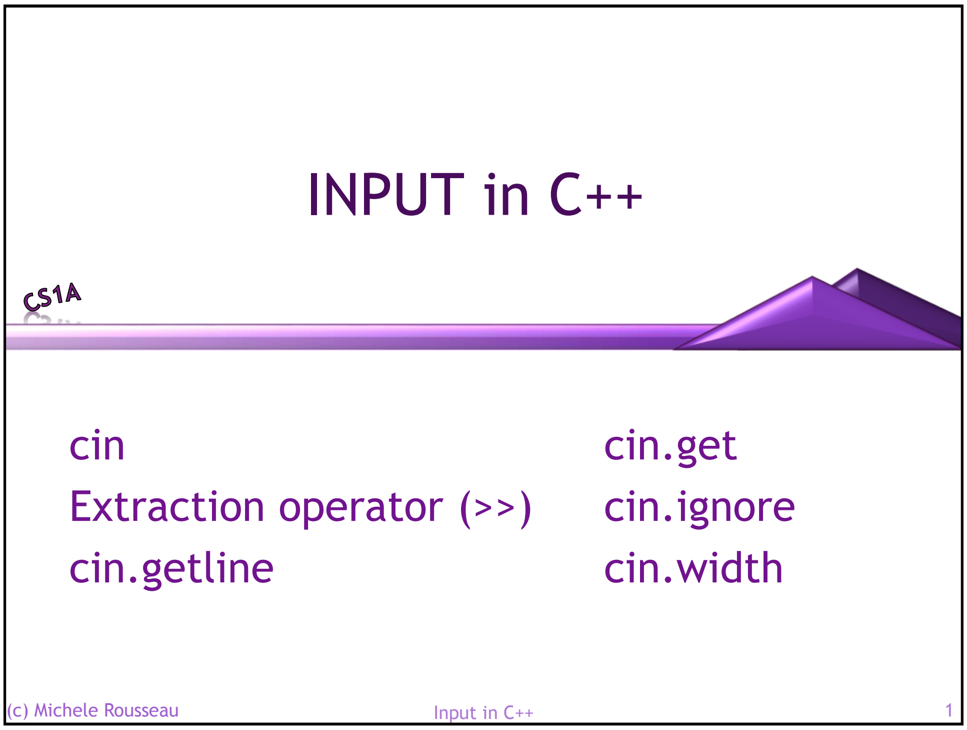
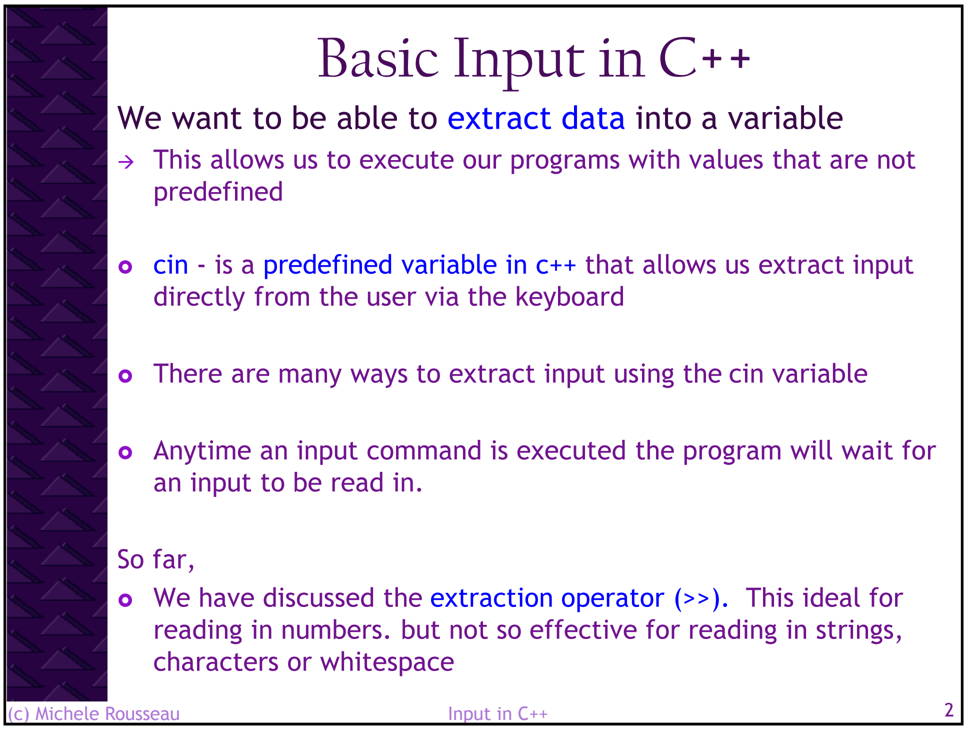


INPUT in C++

CS1A

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

(c) Michele Rousseau Input in C++ 1



Basic Input in C++

We want to be able to extract data into a variable

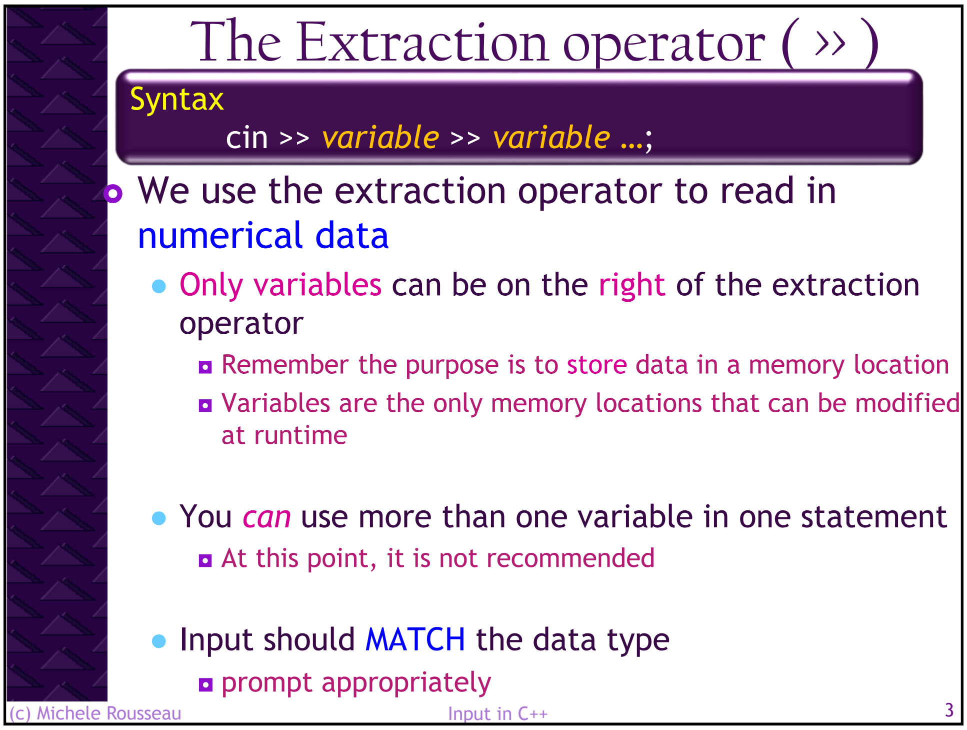
→ This allows us to execute our programs with values that are not predefined

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

(c) Michele Rousseau Input in C++ 2

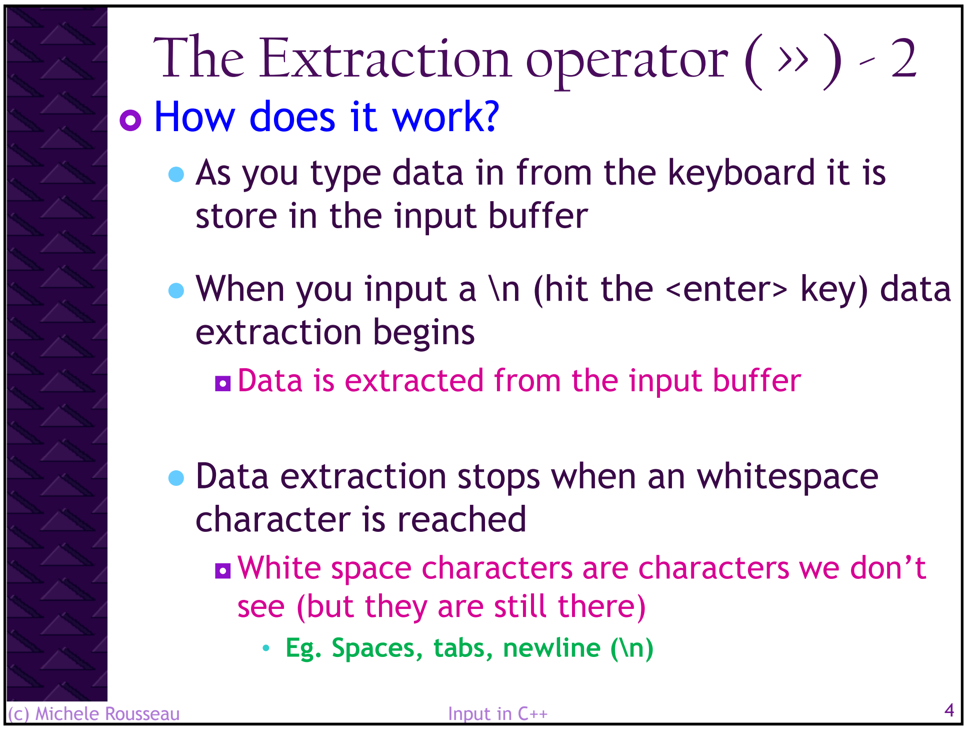


The Extraction operator (>>)

Syntax

```
cin >> variable >> variable …;
```

- We use the extraction operator to read in numerical data
  - Only variables can be on 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
    - At this point, it is not recommended
  - Input should MATCH the data type
    - prompt appropriately



The Extraction operator (>>) - 2

- How does it work?
  - As you type data in from the keyboard it is store in the input buffer
  - When you input a \n (hit the <enter> key) data extraction begins
    - Data is extracted from the input buffer
  - Data extraction stops when an whitespace character is reached
    - White space characters are characters we don't see (but they are still there)
      - Eg. Spaces, tabs, newline (\n)

Examples

```
cin >> length;
cin >> width;
cin >> length >> width;
```

What will happen here?

→ Although we probably want some cout statements with these – why?

Example #1: Extraction Operator (>>)

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

EXAMPLE

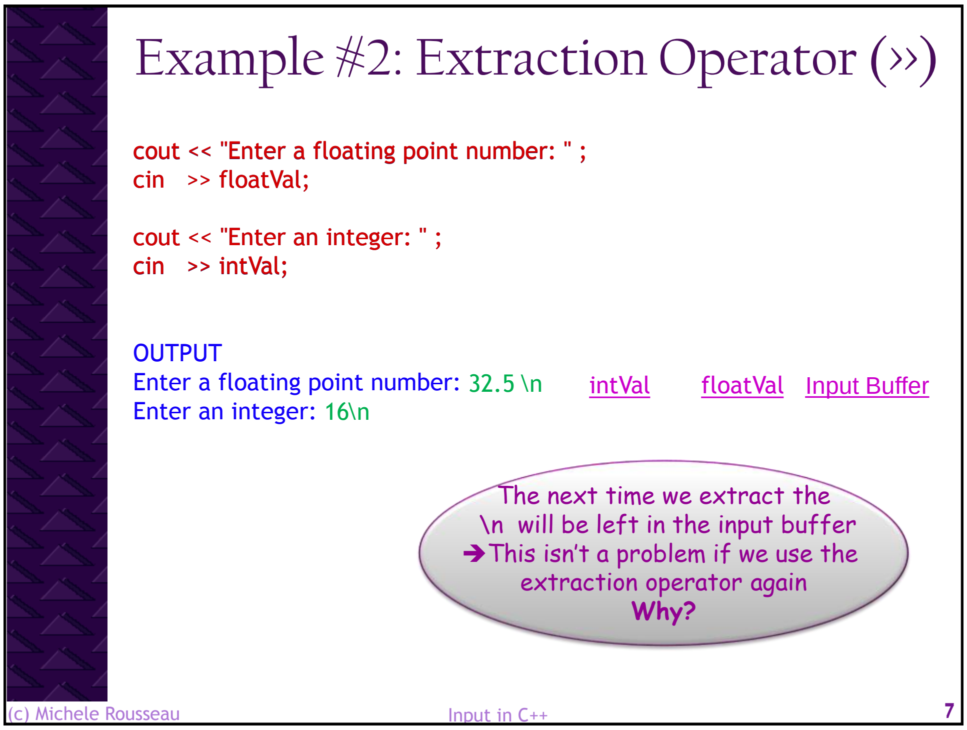
```
cout << "Enter a floating point number: " ;
cin   >> floatVal;
```

OUTPUT:
Enter a floating point number: 32.5\n

floatVal

Input Buffer
$\underbrace{32.5}$\n

- Once you hit enter (\n) the program starts extracting
- First, we check the input buffer.
- It is empty so it waits for input from the keyboard..
- >> ignores leading whitespace
  - →starts extracting when it reaches a non-whitespace character
  - →stops reading when it reaches a whitespace character
- The \n (newline char) stays in the input buffer
- **Note**: Next time we try to extract it will extract from the input buffer first



Example #2: Extraction Operator (>>)

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

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

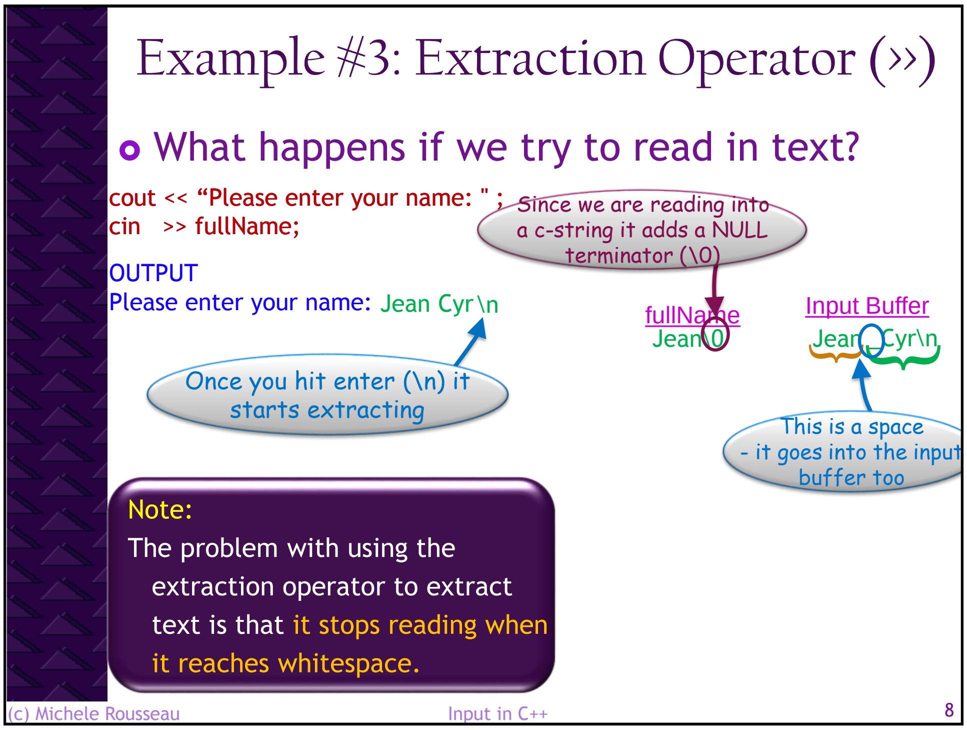
OUTPUT

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

intVal floatVal Input Buffer

The next time we extract the \n will be left in the input buffer
➔This isn't a problem if we use the extraction operator again
Why?

(c) Michele Rousseau Input in C++ 7



Example #3: Extraction Operator (>>)

- What happens if we try to read in text?

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

Since we are reading into a c-string it adds a NULL terminator (\0)

OUTPUT

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

fullName
Jean\0

Input Buffer
Jean Cyr\n

Once you hit enter (\n) it starts extracting

This is a space - it goes into the input buffer too

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

(c) Michele Rousseau Input in C++ 8

Cin & C-Strings

How do I read in text & manage all the whitespace?

Image: Courtesy of © Christine Jopling

cin.getline()

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

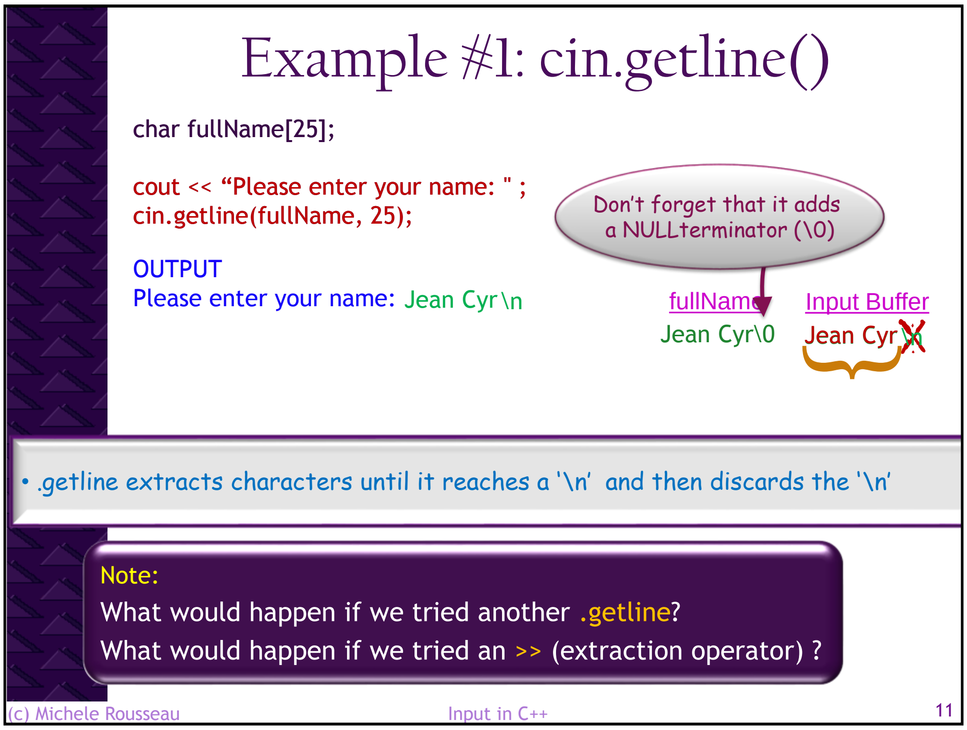
- Reads EVERYTHING including white space UNTIL
  1. a \n is read
  2. width-1 is reached(appends \0 -null terminator after it reads data
- it will extract and discard the \n (cin will not!)

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 #1: cin.getline()

```
char fullName[25];

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

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

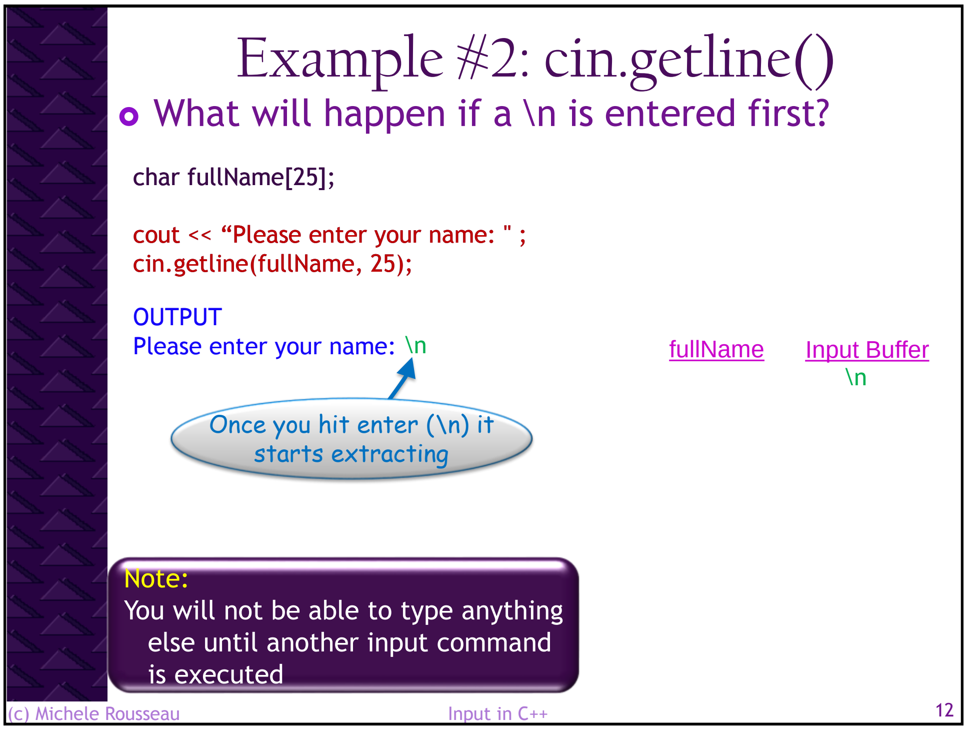
Don't forget that it adds a NULLterminator (\0)

fullName	Input Buffer
Jean Cyr\0	$\underbrace{\text{Jean Cyr}}$ ~~\n~~

- .getline extracts characters until it reaches a '\n' and then discards the '\n'

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

(c) Michele Rousseau Input in C++ 11



Example #2: cin.getline()

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

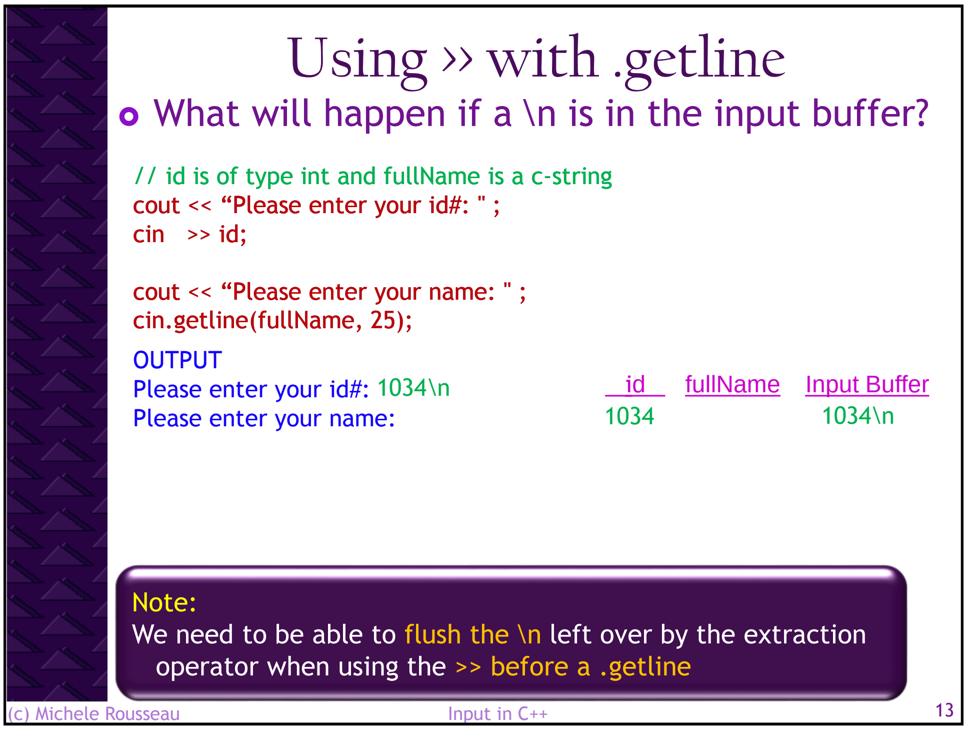
```
char fullName[25];

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

OUTPUT
Please enter your name: \n
```

Once you hit enter (\n) it starts extracting

fullName	Input Buffer
	\n

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

(c) Michele Rousseau Input in C++ 12



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);
```

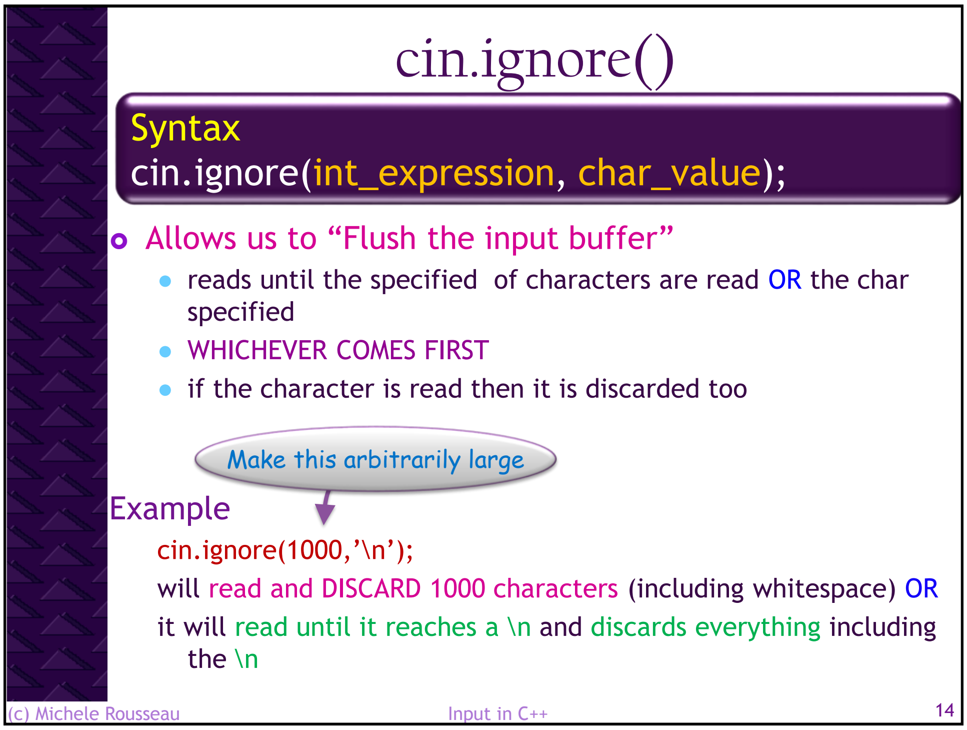
OUTPUT

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

id	fullName	Input Buffer
1034		1034\n

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

(c) Michele Rousseau Input in C++ 13



cin.ignore()

Syntax
cin.ignore(int_expression, char_value);

- Allows us to "Flush the input buffer"
  - reads until the specified of characters are read OR the char specified
  - WHICHEVER COMES FIRST
  - if the character is read then it is discarded too

Make this arbitrarily large

Example

```
cin.ignore(1000,'\n');
```

will read and DISCARD 1000 characters (including whitespace) OR
it will read until it reaches a \n and discards everything including the \n

(c) Michele Rousseau Input in C++ 14

Using .ignore to flush the input buffer

- 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;
cin.ignore(10000, '\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 ()

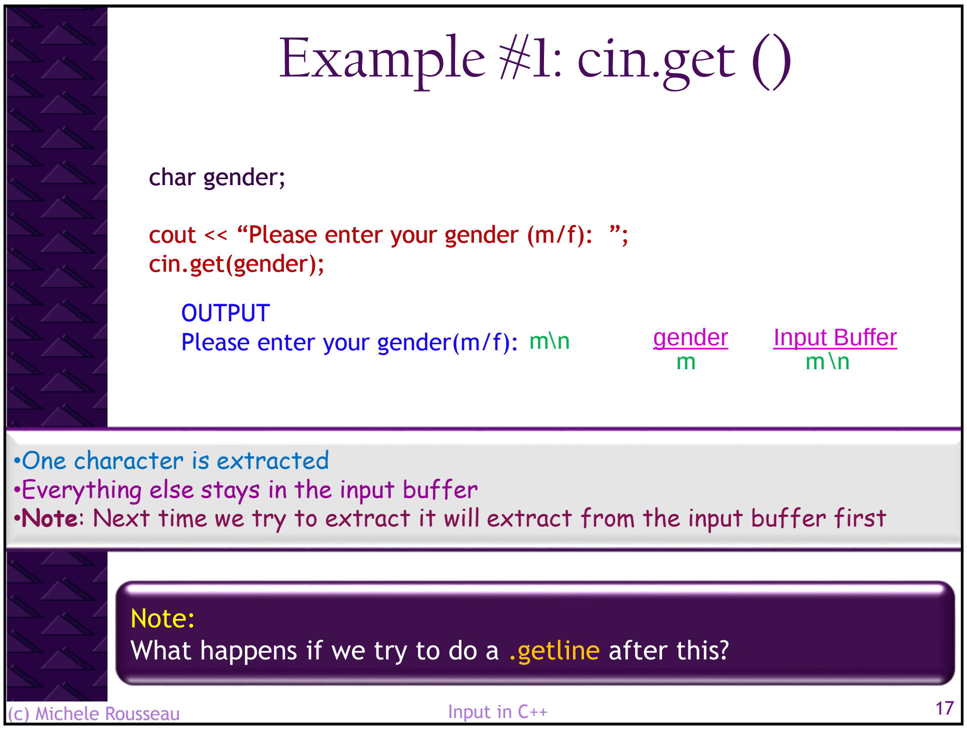
Syntax
cin.get(*charVariable*);

- 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 #1: cin.get ()

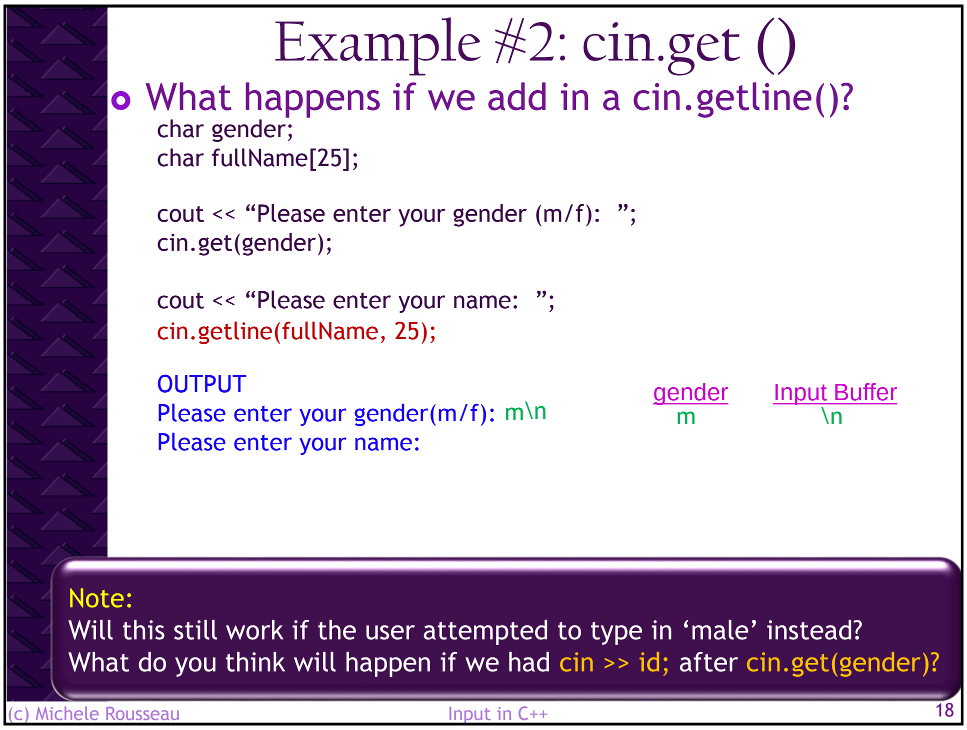
```
char gender;

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
```

- One character is extracted
- Everything else stays in the input buffer
- **Note**: Next time we try to extract it will extract from the input buffer first

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

(c) Michele Rousseau Input in C++ 17



Example #2: cin.get ()

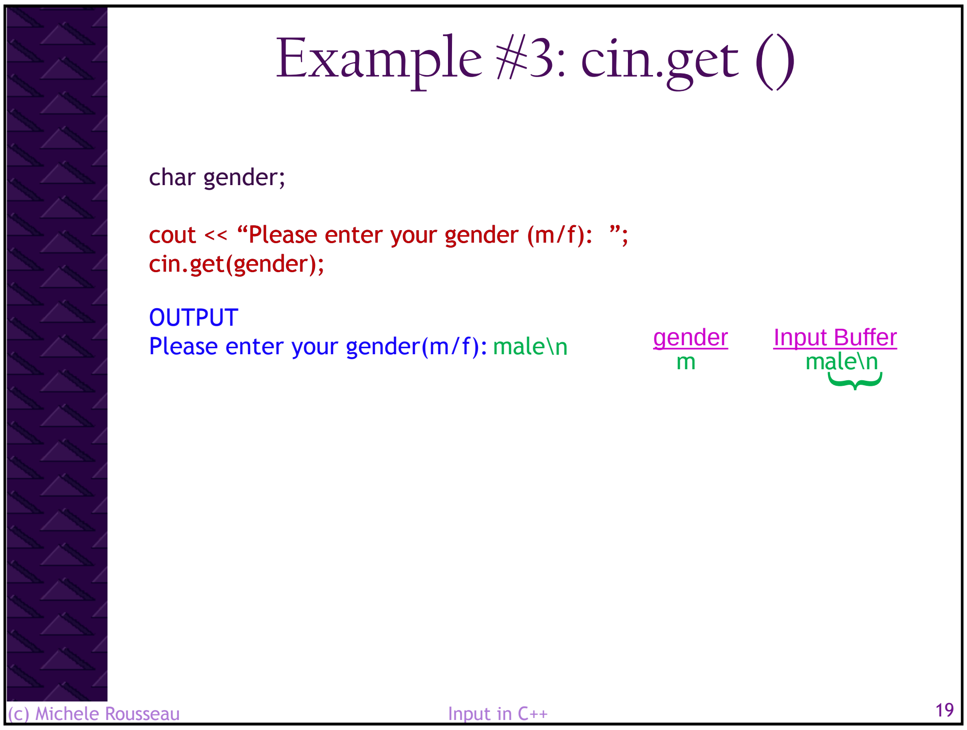
- What happens if we add in a cin.getline()?

```
char gender;
char fullName[25];

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

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

```
OUTPUT
Please enter your gender(m/f): m\n        gender     Input Buffer
Please enter your name:                     m            \n
```

Note:
Will this still work if the user attempted to type in ‘male’ instead?
What do you think will happen if we had cin >> id; after cin.get(gender)?

(c) Michele Rousseau Input in C++ 18



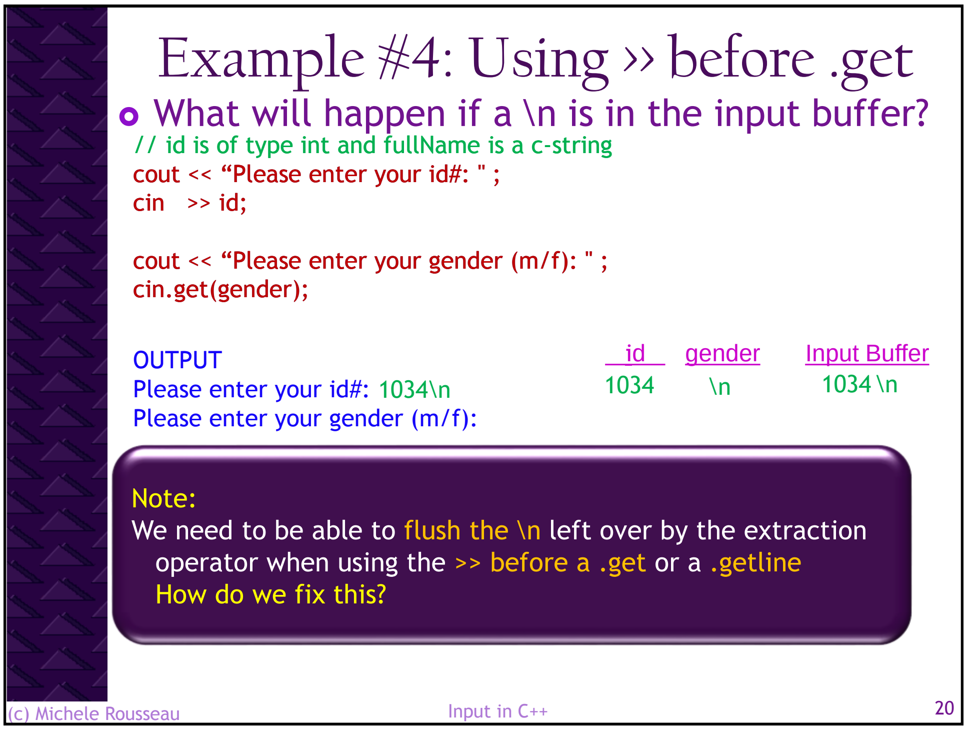
Example #3: cin.get ()

```
char gender;

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

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

gender	Input Buffer
m	male\n

(c) Michele Rousseau Input in C++ 19



Example #4: Using >> before .get

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

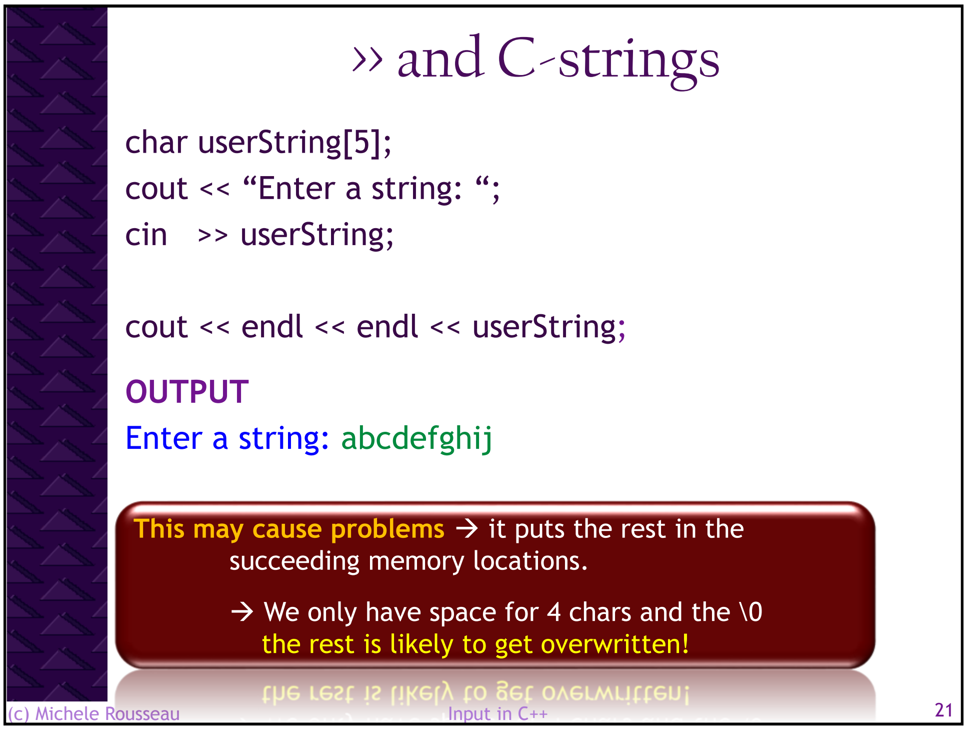
```
// id is of type int and fullName is a c-string
cout << “Please enter your id#: " ;
cin   >> id;

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

```
OUTPUT
Please enter your id#: 1034\n
Please enter your gender (m/f):
```

id	gender	Input Buffer
1034	\n	1034 \n

Note:
We need to be able to flush the \n left over by the extraction operator when using the >> before a .get or a .getline
How do we fix this?

(c) Michele Rousseau Input in C++ 20



>> and C-strings

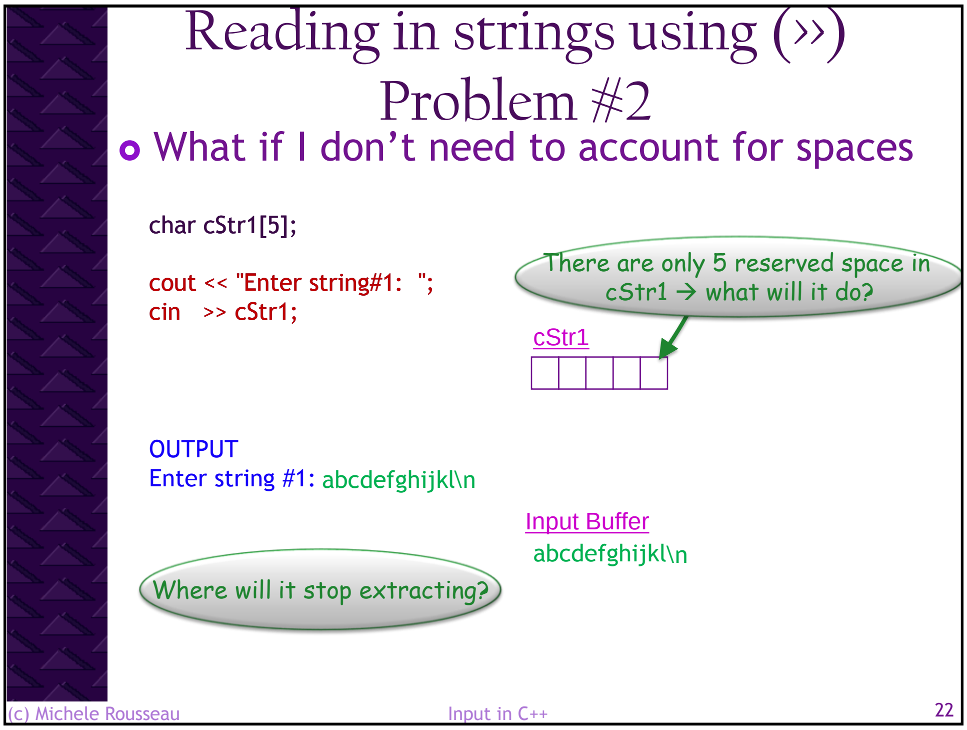
```
char userString[5];
cout << “Enter a string: “;
cin   >> userString;

cout << endl << endl << userString;
```

OUTPUT

```
Enter a string: abcdefghij
```

This may cause problems → it puts the rest in the succeeding memory locations.

→ We only have space for 4 chars and the \0 the rest is likely to get overwritten!



Reading in strings using (>>) Problem #2

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

```
char cStr1[5];

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

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

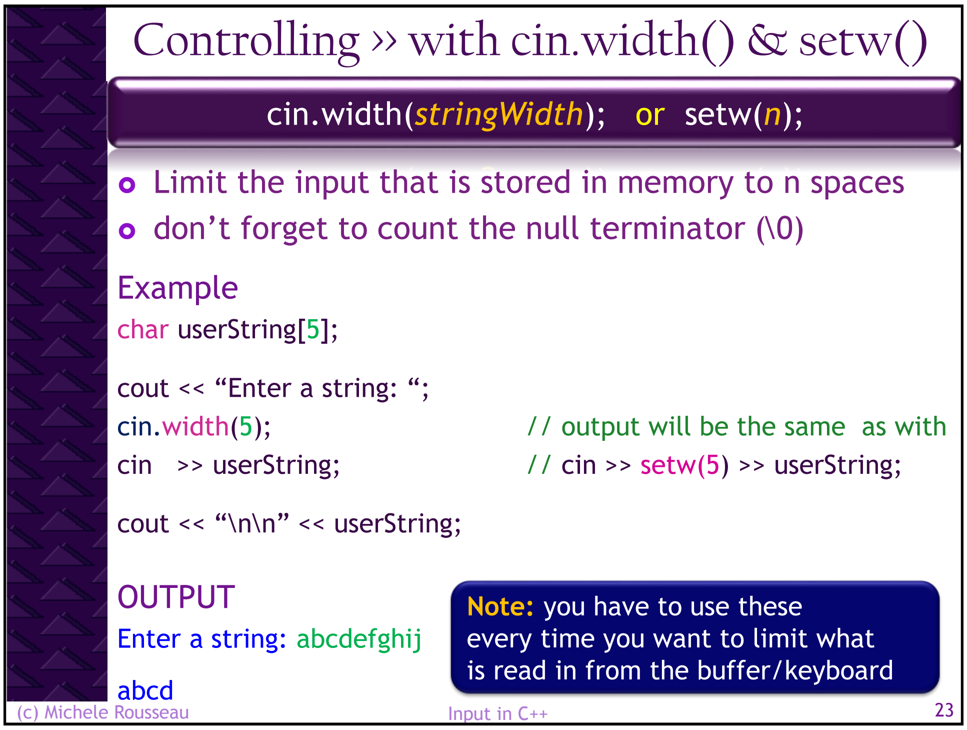
OUTPUT

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

Input Buffer

abcdefghijkl\n

Where will it stop extracting?



Controlling >> with cin.width() & setw()

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

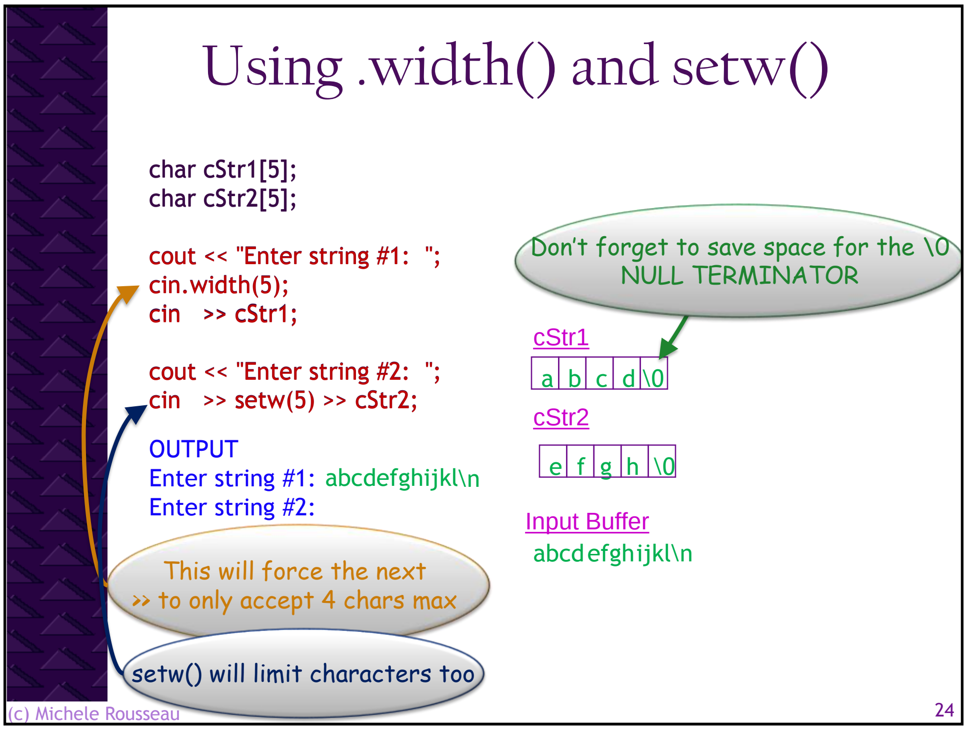
```
char userString[5];

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

(c) Michele Rousseau Input in C++ 23



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

setw() will limit characters too

Don't forget to save space for the \0 NULL TERMINATOR

cStr1

a	b	c	d	\0

cStr2

e	f	g	h	\0

Input Buffer

abcdefghijkl\n

(c) Michele Rousseau 24

Example

how can we do this?
(have the program wait for any input)

Press <enter> to continue

>> can't accept the Enter key as a character for input

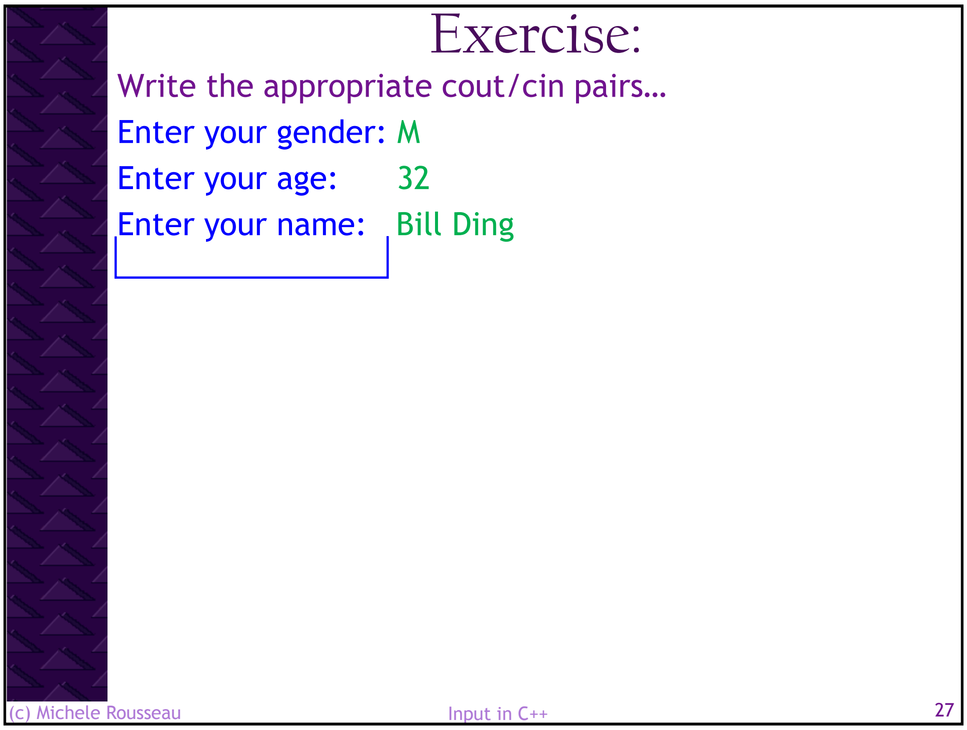
```
cout << "Press <enter> to continue";
cin.ignore(1000, '\n');
```

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



Exercise:

Write the appropriate cout/cin pairs…

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