Outline

- In this lesson, we will:
  - Learn how to request data from the console
  - Introduce streams and review whitespace
  - Look at entering characters, integers, floating-point numbers and Boolean types
  - Introduce the string class
  - See how to read in strings

Background

- Previously, we've seen how we can print data to the screen:

```
std::cout << "Hi " << 3 << ',' << 3.14 << std::endl;
```

- How do we get data from the keyboard?
  - Previously, a console was often a single unit consisting of both a keyboard and screen
  - *Console input* is from the keyboard
  - *Console output* is to the screen

Jason Scott

Background

- The `<<` operator appears to be *directing* data to the console output:

```
std::cout << "Hi " << 3 << ',' << 3.14 << std::endl;
```

- To request data from the keyboard, we use `std::cin`
- Any data received from the console must be temporarily stored
  - We will use local variables
- The `>>` operator appears to be *directing* data from the console to the variable:

```
typename x{};   // Use default value, just in case
std::cin >> x;
```

Background

- For example:

```
int main() {
    int m{};
    std::cout << "Enter an integer: ";
    std::cin >> m;

    if ( m < 0 ) {
        std::cout << m << " is less than 0." << std::endl;
    } else if ( m == 0 ) {
        std::cout << "You entered 0." << std::endl;
    } else {
        std::cout << m << " is greater than 0." << std::endl;
    }
}
```


Background

- Try it yourself:

```
#include <iostream>

// Function declarations
int main();

// Function definitions

// Request the user enter a number and the describe it relative to zero
int main() {
    int m{};
    std::cout << "Enter an integer: ";
    std::cin >> m;

    if ( m < 0 ) {
        std::cout << m << " is less than 0." << std::endl;
    } else if ( m == 0 ) {
        std::cout << "You entered 0." << std::endl;
    } else {
        std::cout << m << " is greater than 0." << std::endl;
    }
}
```


Background

- May appear peculiar
  - Characters are only processed once the user presses **Enter**
  - The characters are internally stored in the order they arrive
    - As you keep typing, any more characters are added to the end of the *stream*

	H	i	!	→	I	'	l	l		t	a	k	e		3	↲			· · ·

Characters

- Depending on the type of the variable, `std::cin` will attempt to satisfy the request

```
char ch{};
std::cin >> ch;
```

	H	i	!	→	I	'	l	l		t	a	k	e		3	↲			· · ·

- It will assign to `ch` the first non-whitespace character
  - In this case, `'H'`

i	!	→	I	'	l	l		t	a	k	e		3	↲				· · ·

- Whatever value was previous assigned to `ch` is lost

Integers

- If you request an integer, it will start looking for decimal digits and interpret it as a base-10 number

```
int m{};
std::cin >> m;
```

i	!	→	I	'	l	l		t	a	k	e		3	↲					· · ·

- It stops if it finds a non-decimal-digit
  - If no other digits were found, it assigns `m` the value `0`

i	!	→	I	'	l	l		t	a	k	e		3	↲					· · ·

Integers

- Suppose, instead, after this statement executes, the user actually typed something that starts with digits

```
int m{};
std::cin >> m;
```

3	5	2	4	0	-	1	5	↲											· · ·

- It finds `35240`, assigns it to `m` and stops reading from the stream

-	1	5	↲																· · ·

Other types

- Floating-point numbers are similar to integers, only we can include a decimal point and possibly scientific notation
- The Boolean type `bool` is different:
  - It requires a numeric value: `0` or `1` followed by **Enter** or whitespace
  - Anything else is considered an error, although the variable is set to `true`

Multiple inputs

- The following requests two integers:

```
std::cout << "Enter two integers: ";
int m{}, n{};
std::cin >> m >> n;
```

- Possible inputs include:

```
42 91↲

42      91↲

42↲
91↲
```

Strings

- We have seen how to print strings:

```
std::cout << "Enter a word: ";
```

- How do we read strings from the console? What type must `str` be?

```
std::cin >> str;
```

- In C++, there is the `string` library, which contains a `std::string` *class*
  - In addition to primitive types, variables can be defined to be of one of these classes
  - The `std::cin` object is aware of this class

Strings

- As an example:

```
#include <iostream>
#include <string>

int main();

int main() {
    std::string str{};

    std::cout << "Enter a word: ";
    std::cin >> str;

    std::cout << "You entered the word \"" << str << "\"."
              << std::endl;

    return 0;
}
```


Strings

- Try it yourself:

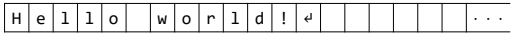
```
#include <iostream>
#include <string>

// Function declarations
int main();

// Function definitions

// Ask the user to enter a word and then print that word to the screen
int main() {
    std::string str{};

    std::cout << "Enter a word: ";
    std::cin >> str;

    std::cout << "You entered the word \"" << str << "\"."
              << std::endl;

    return 0;
}
```


Strings

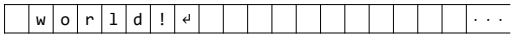
- When asking for a string, `std::cin` will:
  - Ignore any leading whitespace characters
  - Read in all non-whitespace characters, and
  - Stop before the next whitespace character
- Suppose you request a string:

```
std::cin >> str;
```

and the user enters this line:



- The string `str` is assigned the string "Hello" and the stream is now:



Summary

- Following this lesson, you now:
  - Understand that one way to request data is through `std::cin`
  - Are aware of input and output streams
  - Know how to request the primitive datatypes from the user
    - Whitespace is ignored
    - Only characters that can be interpreted are
  - Are aware of the `std::string` class and the standard string library

References

[1] No references?

Colophon

These slides were prepared using the Georgia typeface. Mathematical equations use Times New Roman, and source code is presented using `Consolas`.

The photographs of lilacs in bloom appearing on the title slide and accenting the top of each other slide were taken at the Royal Botanical Gardens on May 27, 2018 by Douglas Wilhelm Harder. Please see

https://www.rbg.ca/

for more information.



Disclaimer