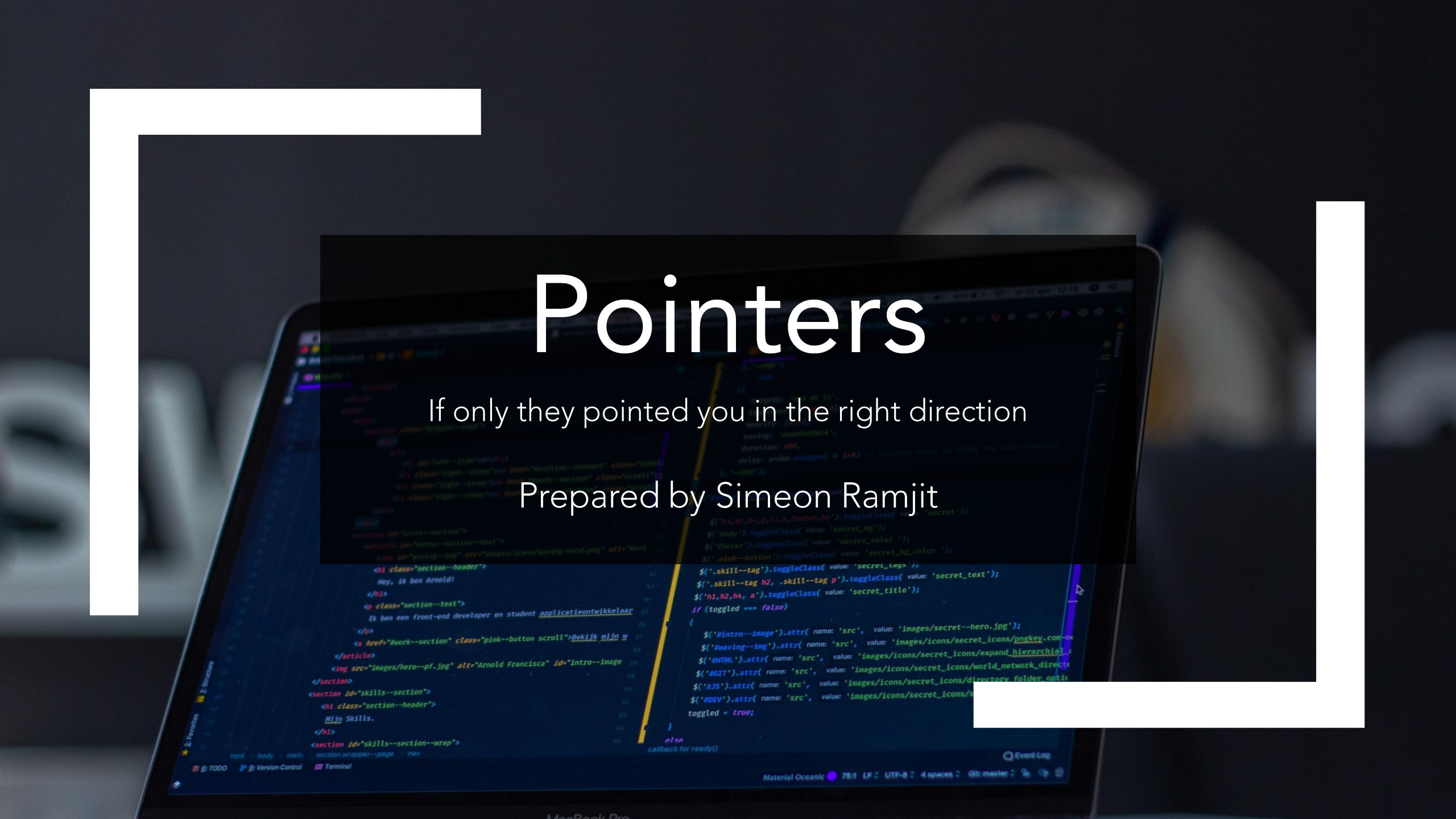

Pointers

If only they pointed you in the right direction

Prepared by Simeon Ramjit

What is a pointer?

- We'll get there soon, we first need to evaluate how we've been handling variables all this time
- It was done by declaring a datatype, variable name and then an assignment to some value
- The operating system decides where in memory (RAM) that variable will live
- That place also has a memory **address.** Normally, to use the variable we use the variable name, however it may be useful to use the address of the variable instead.

Variable Address

- To get the address of a variable, we need to use the address-of operator, &

```
#include<iostream>
int main() {
  int myvariable = 90; //declare and init variable
  std::cout << &myvariable << std::endl; //print address of variable
  std::cout << myvariable << std::endl; //print value of variable
  return 0;
}
```

- Run this program again, note the different address on every run but the same value
- The OS will usually place it in a different place in memory every time.

Variable Address

- The ability to access the address of a variable means we can use the address of variable in calculations or pass them into functions for a performance boost
- When you use the variable name, you pass a copy of the variable which is slow. Using the address will help especially when your variables are large as in the case of arrays and vectors (pass by reference vs pass by value)

So, what is a pointer?

- It's a **variable** that **stores the address** of another variable.
- They have the name pointer since they 'point to' the variable who's address they store
- Let's assume we have 'myvariable' and then 'mypointer' has the address of 'myvariable'

So, what is a pointer?

- The **value** that a pointer has is the **address** of the variable it points to

Variable Name	Value	Address
myvariable	90	0x80808080
mypointer	0x80808080 = 2,155,905,152	0x11110000

Pointers in C++

- Pointers themselves have to match the datatype of the variable they're pointing to
- Pointers have a special declaration in C++
- To get the value of the variable that's being pointed to from the pointer, the pointer has to be dereferenced 🤯
- E.g. if we wanted the value of 'myvariable' from 'mypointer' we'd have to dereference 'mypointer'
- The dereference operator in C++ is *

Pointers in C++

```
#include<iostream>
int main() {
 char mychar = 'A';
 int myint = 90;
 //declare a character ptr and init with address of char variable
 char* mycharptr = &mychar;
 //declare an int ptr and init with address of int variable
 int* myintptr = &myint;
 std::cout << "The character value is: " <<*mycharptr
           <<". The address is: "<< static_cast<void*>(mycharptr)
           <<std::endl;

 std::cout << "The integer value is: " << *myintptr
           << ". The address is: " << myintptr
           << std::endl;
 return 0;
}
```

Pointers in C++

- What is `static_cast<void*>(mycharptr)`?
- When taking the address of **`mycharptr`** , you get **`char*.`** The operator `<<` interprets that as a C string, and attempts to display a character sequence instead of the address
- That was needed only to display the address correctly but the underlying address is there to be used as normal
- Unlikely you'll need this knowledge soon so don't get bogged down with it

Pointers in C++

- Sometimes we need to initialize a pointer but we don't have use for it yet. In that case there exists `nullptr`

```
int* myintptr = nullptr;
```

- Do not use a pointer that has been initialized with `nullptr` unless a valid value have been written to it. That will result in undefined behaviour

That’s a wrap!

- Challenge for you
  - Create a function that adds two numbers but the arguments are passed via reference
  - Create a function that takes in array of 5 integers, increments each value by 1 and returns the modified array to main. Print the array before and after modification
    https://www.tutorialspoint.com/cplusplus/cpp_passing_arrays_to_functions.htm

Questions?

simeon.ramjit@sta.uwi.edu

github.com/simeon9696/programmingworkshop