Philip Pesic

Week 8

October 9 2022

Week 8 Lecture 8 Notes

Address Operator

The & symbol is an address operator. When placed in front of a variable, it returns the variable's address, rather than the value. This is possible due to C++, which allows the programmer to access the RAM.

Ex:

```
Int x = 1;
Cout << &x;
```

Output: 7ad882ed

Pointer Variable

A pointer variable is a variable whose contents are a RAM address. It is declared with an asterisk (int * ptr;).

Ex:

```
Int * ptr = NULL;
Int x = 1;
ptr = &x;
Cout << ptr;
```

Output: 77aed113

Philip Pesic

Week 8

October 9 2022

Week 8 Lecture 8 Notes

Dereferencing

Dereferencing is the manipulation of a value in the RAM address that a pointer points to. By using an asterisk before assigning value to a pointer, it will instead assign that value to the RAM address being pointed to.

Ex:

```
Int x = 1;
Int * ptr = x;
*ptr = 2; //Assigns 2 to the RAM address ptr points to, in this case, it's x
Cout << x;
```

Output: 2

Pointers in Classes

In classes, pointers function similarly to normal objects. A pointer in a class is declared with an asterisk, and called using an arrow instead of a dot.

Ex:

```
myClass * Ptr;
Ptr -> getX();
```

Philip Pesic

Week 8

October 9 2022

Week 8 Lecture 8 Notes

Dynamic Variables and Deallocating RAM

Dynamic variables are a form of assigning RAM addresses to a pointer. Dynamic variables do not have variable names, but instead simply assign the RAM address of the first byte they take up to another variable. After being used, C++ does not automatically delete dynamic variables. Instead, we use the delete command to delete the contents of an address that is being pointed to.

Ex:

```
Int * ptr = new int; //Creates a single-use integer variable that assigns its RAM address to ptr
Cout << ptr;
Delete ptr; //Deletes the dynamic int
```

Output: 103a91f

The Problem with Pointers

1 - When deleting a pointer variable, the delete command will not give it a NULL value, breaking the pointer.

Solution - Redeclare pointer

2 - When a pointer variable goes out of scope or is deleted, the OS still holds the contents of the variable it was pointing to.

Solution - Use less delete commands