Intermediate C++

Operators

| * | Indirection operator, can dereference a pointer to point to its location’s value |
| & | “Address of” operator, refers to a variable’s place in memory |
| . | Member selection operator, can access members of objects |

Arrays

```cpp
// standard array declaration
int arr[5];

// array initialization
int arr[] = {1, 2, 3, 4, 5};

// Access first element of array
int var = arr[0];

// Five rows, 2 columns
int arr[5][2];

// Dynamic allocation
int *arr = NULL;
const int VAR = 10;
arr = new int[VAR];
```

Important things to remember:
- Arrays are zero indexed
- Arrays are passed by reference, not by value
- Arrays are of fixed size and size can only be defined by constants
- C++ has no bounds checking, so it can read and write to things it shouldn’t
- This is bad, don’t let it do that

Classes

```cpp
class Object {
    private:  
        int attr;
    public:
        Object(int parameter) { 
            attr = parameter;
        }
        int getAttr() { 
            return attr;
        }
        void setattr(int value) { 
            attr = value;
        }
};
```

```cpp
int main() {
    Object object = Object(1);
    object.attr = 2;
    object.setAttribute(2);
    return 0;
}
```

1. Private section, all contents are only accessible by the object itself
2. An attribute, or a variable that belongs to the class object
3. Public section, accessible externally through the object
4. A constructor, the function that’s called when a new object is made
5, 6. A method, or a function owned by the class. These are getters and setters, allowing controlled access to an otherwise private variable
7. Creation of a new instance of the object (naming convention: classes are capitalized, objects are not
8. Illegal, as the attribute is private
9. A proper method call, setting ‘object’s attribute ‘attr’ to two using the setter method