C++ Part 2

Recap

- Compile c++ code using g++ command
- Include libraries using #include <blah>
- Use iostream library and std::cout to print to screen and std::cin for input
- Logical flow controls with if and else statements

Non-primitive types and class methods

- Non-primitive types are typically defined by classes
  - Like primitive types, it is usually necessary to initialize non-primitive objects
- Class objects usually have associated properties and methods
- Class methods generally modify or return information about class properties

```
myclass myobj = <something>;
myobj.mymethod1();
myobj.mymethod2(myargument);
```

Casting data types

- It is sometimes possible to change (cast) from one data type to another
- Primitive types can be changed, but sometimes information is lost
- Non-primitive types may or may not be changed, depending on definitions
- Implicit and explicit casting are possible

Implicit:

```
float myfloat = 7.3;
int myint = myfloat;
```

Explicit:

```
int myvar1 = 7;
int myvar2 = 5;
float myvar3 = float(myvar1) / float(myvar2);
```

Auto data type

- Introduced in C++11
  - Need to use -std=c++11 flag with GCC (not necessary for some other compilers)
- Automatically assigns correct data type based on return type of a function
- Useful when return type may change or to save typing for long type names

```
int var1 = 7;
auto var2 = var1;
```

CMath library and random numbers

- C++ provides most common math functions with the cmath library
  - Find more details at https://en.cppreference.com/w/cpp/header/cmath
- Many functions such as std::sqrt(...) and std::sin(...)
- Mathematical constants such as M_PI
- Random numbers can be generated using cstdlib library
- std::rand() generates random integer between 0 and RAND_MAX
  - Not actually random, but created from an algorithm that starts with a seed number
  - Use std::srand(...) to set seed
  - More details: https://en.cppreference.com/w/cpp/numeric/random/rand

Strings

- A std::string is a class that holds a variable length sequence of characters
  - https://en.cppreference.com/w/cpp/string/basic_string
- Include the string library
- Generally initialized using "text in quotation marks"
- Many methods available such as append(), find() and replace()
- Strings can be added together with the += operator
- More complex strings can be built with std::stringstream (include sstream)
  - Beyond the scope of this class, but worth looking into individually
  - https://en.cppreference.com/w/cpp/io/basic_stringstream

Arrays

- A fixed-length set of values of a single type
- Declare arrays as (with optional initialization):

```
float myarray[3] = {5, 2, 9};
```

- Access value at index i to read or set:

```
myarray[i] = 6;
```

- Indexing begins at 0: a length-3 array uses indices 0, 1, and 2
- Trying to access beyond the end of an array can cause problems
    - Reading will result in undefined results, setting can lead to segmentation violations

Vectors

- Variable-length set of values of a single type (include vector library)
  - Allocated memory is dynamically allocated as values are added

```
std::vector<double> myvec;
```

- Add elements to vector:

```
myvec.push_back(5.8);
```

- Access element i with myvec.at(i) or myvec[i]
- Get the size of a vector with myvec.size()
- Empty the contents of a vector with myvec.clear()

While loops

- Iteratively repeat steps as long as a condition is met
- Skip an iteration with continue
- Exit out of the loop early with break

```
int num = 1;
while(num < 20)
{
    if(num%3 == 0) continue;
    std::cout << num << std::endl;
    num += 2;
}
```

For loops

- Iteratively repeat steps for a defined number of times
- Range-based and for-each are commonly used
    - Other methods such as using iterators are available, but somewhat archaic

Range-based:

```
std::vector<int> vec = {3,7,2,9};
for(int iNum = 0; iNum < vec.size(); iNum++)
{
    std::cout << vec.at(iNum) << std::endl;
}
```

For-each:

```
std::vector<int> vec = {3,7,2,9};
for(auto num : vec)
{
    std::cout << num << std::endl;
}
```

User arguments

- argc is the number of arguments and argv is an array of the values

```
// hello.cxx file
#include <iostream>
int main(int argc, char** argv)
{
    std::cout << argc << std::endl;

    for (int i = 0; i < argc; i++) {
        std::cout << argv[i] << std::endl;
    }

    return 0;
}
```

Resources

- https://www.w3schools.com/ - Great online learning resource
- https://www.youtube.com/@codebreakthrough - Excellent tutorial videos
- https://en.cppreference.com/w/ - Thorough documentation
- https://stackoverflow.com/ - Ask questions to experts