Project 6: C++

Abstract

In this project, we explored features related to user input, file input and output and exception handling in C++.

Word Counter

The first task was to implement a word counter that can read a text file and output the 20 most common words. As in previous projects, we had a number of occasions to learn about useful containers and methods in the C++ standard library that made the task all but trivial in comparison to the same task in pure C. For example, this implementation relied on the std::unordered_map container for counting the words. This container is easily initialized and requires no definition of additional structs or convenience methods – the types of data it holds are indicated right away:

```
// hashmap container for the words
unordered_map<string, int> umap;
```

The std::unordered_map is a hash map, therefore it requires constant (or linear in the absolute worst case scenario) access time – which makes it a perfect tool to use for a word counter that can hold large numbers of words.

File I/O

File input and output in C++ is typically done with the fstream library. File reading and writing is done with separate objects – ifstream and ofstream, respectively. By performing these operations with different kinds of objects, the language prevents all kinds of file writing errors and data corruption.

```
char inp[100];

// get information from user and write it to a file
ofstream ofile;
ofile.open("info.txt");

cout << "What is your name? ";
cin.getline(inp, 100);
ofile << inp << endl;
```

Terminal input and output, as well as file input and output, are performed with objects in C++, using the "<<" and ">>" operators.

We wanted to find out if there are quick and easy ways to read data from a URL, but it turn out that C++ does not provide that functionality and it is only available through external libraries that are tailored for particular applications.

Exception Handling

C++ inherits C's error signal codes (it was designed with backwards compatibility in mind, after all). But it also has exception handling capability. Exception handling can be very useful in C++, because it is often a better alternative than using if statements and cluttering the code with control flow blocks. However, try, catch, and throw statements should only be used in the context of resource acquisition at run time i. e.external circumstances, not for checking bugs in the code itself.

```cpp
// can throw an invalid argument exception
void set_class_year(string num){
    int cy = stoi(num);

    if (cy > 0) {
        class_year = cy;
    }
    else {
        throw invalid_argument("Class year cannot be negative");
    }
}
```

Example of throwing an exception.

Conclusion

In this project, we implemented a hashmap-based word counter and explored user input, file i/o, and exception handling in C++.

While working on the project, we found the following resources useful:

https://www.tutorialspoint.com/cplusplus/cpp_files_streams.htm

https://isocpp.org/wiki/faq/exceptions

https://stackoverflow.com/a/2393389/10005441