How to Build Your Own grep In C++

Part 1 – Introduction: Why CLI Tools Matter

What are CLI tools?

Command Line interface (CLI) tools are programs that we run in a terminal by typing the commands. For example, instead of clicking buttons in the specific app or in any interface, we can use the commands to run the program. They are so useful because they let the users quickly perform the tasks, and process. Many developers use them to get things done quickly just by interacting with terminal to run things.

Why is *grep* important?

grep is one of the most useful CLI tools. It helps us search for words or phrases in text files. For example, when we trying to find where the word "error" appears in the files with millions of texts, *grep* can search that right away. It's so useful for programmers, especially who have to work with text data to find specific patterns or words, and even data analysis.

How does building one teach you about real-world?

Making my own version of grep in C++ helped me understand how real tools work behind the real process. We can know how to

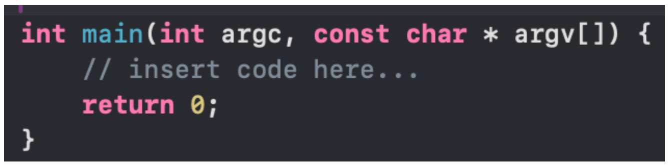
- use command-line arguments, for example like -i or -n
- read files line by line
- search keyword in each line
- add extra features like showing line numbers or ignoring uppercase or lowercase (caseInsensitive)

It can also give us good practice writing clean and useful programs

Part 2 – Core Concepts Explained

How argc and argv work in main()

In C++, when we run a program from the terminal, the words we type after the programs name will act like arguments.



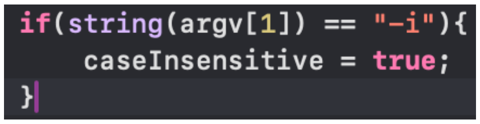
```cpp
int main(int argc, const char * argv[]) {
    // insert code here...
    return 0;
}
```

- argc is the number of arguments we passed in
- argv is the array of C-style strings that hold the value of arguments

If we run this command line ‘./mygrep -i -n error logs.txt’, argc will be 5 and argv[4] (array)

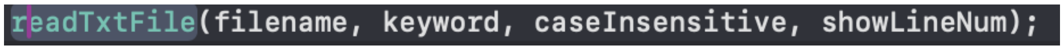
How to process command-line flags manually

To process command-line flags manually, we can just check what’s the flag is and then call the related functions depending on the flag.



```
if(string(argv[1]) == "-i"){
    caseInsensitive = true;
}
```

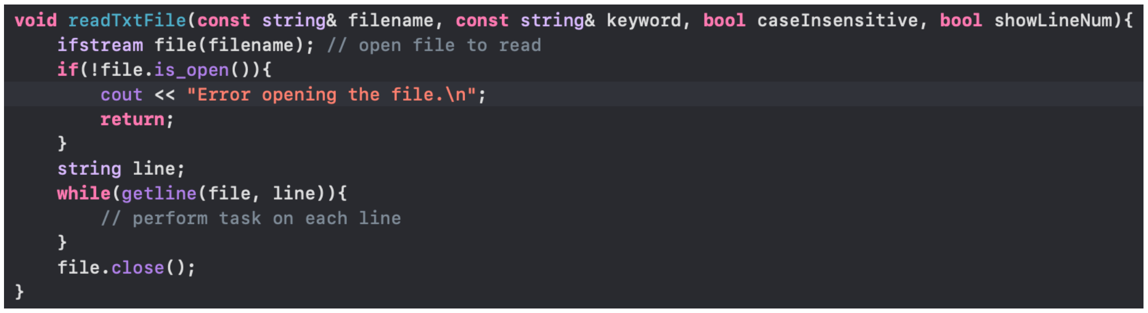
if the flag is “-i”, which is for caseInsensitive, we can use Boolean. And then call the function.



```
readTxtFile(filename, keyword, caseInsensitive, showLineNum);
```

As we set it to true, it will not consider that what the case is lower or upper.

How to read from a file using std::ifstream



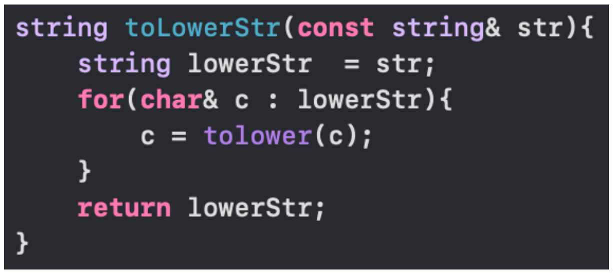
```
void readTxtFile(const string& filename, const string& keyword, bool caseInsensitive, bool showLineNum){
    ifstream file(filename); // open file to read
    if(!file.is_open()){
        cout << "Error opening the file.\n";
        return;
    }
    string line;
    while(getline(file, line)){
        // perform task on each line
    }
    file.close();
}
```

First, we open and read a text file line by line, and perform each line at a time.

How to compare strings in a case-sensitive and insensitive way

By default, C++ compares the strings in case sensitive way, where uppercase and lowercase seems not the same.

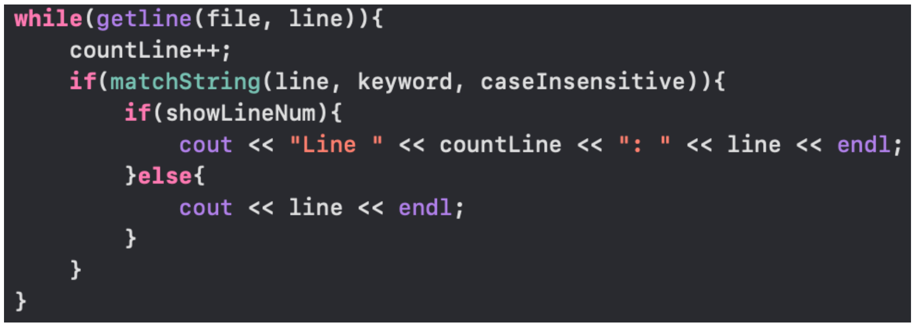
To compare insensitive way, we can do it like this.



```
string toLowerStr(const string& str){
    string lowerStr  = str;
    for(char& c : lowerStr){
        c = tolower(c);
    }
    return lowerStr;
}
```

Manually convert strings to lower case, and then compare them.

How to print clean formatted output with line numbers

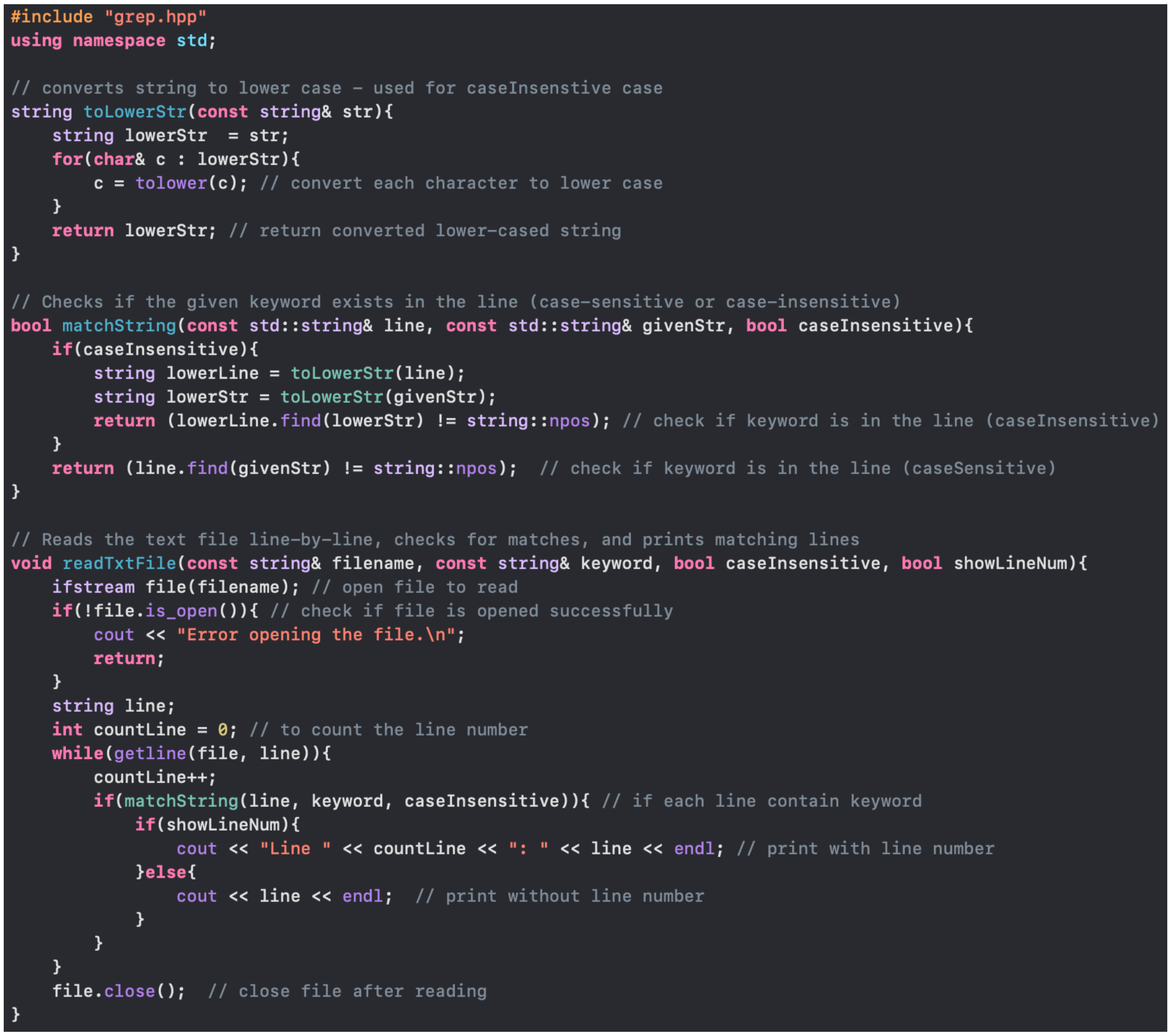


```cpp
while(getline(file, line)){
    countLine++;
    if(matchString(line, keyword, caseInsensitive)){
        if(showLineNum){
            cout << "Line " << countLine << ": " << line << endl;
        }else{
            cout << line << endl;
        }
    }
}
```

This way we can print each line, based on the flag '-i' for caseInsensitive and '-n' for showLineNum.

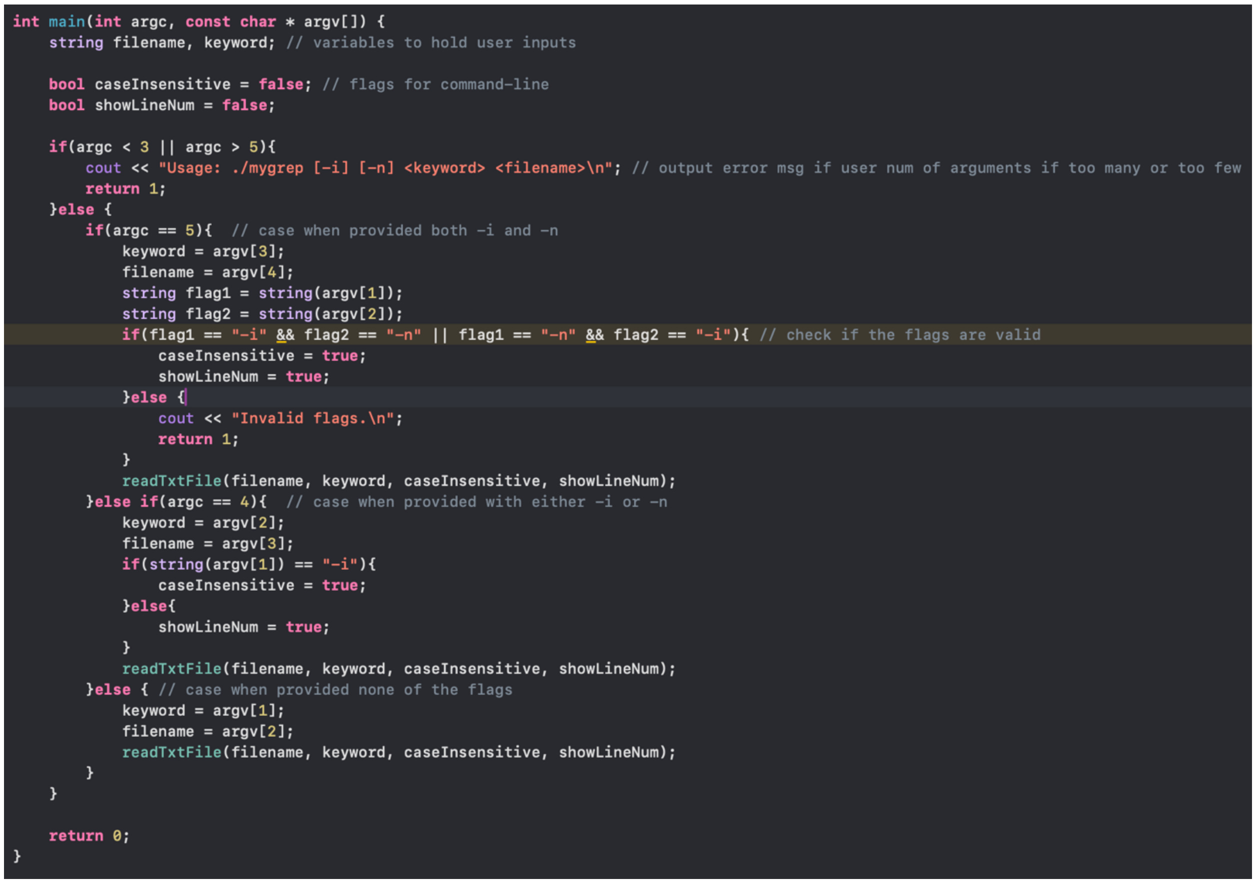
Part 3 – Code and Walkthrough

grep.cpp (helper functions)



```cpp
#include "grep.hpp"
using namespace std;

// converts string to lower case - used for caseInsenstive case
string toLowerStr(const string& str){
    string lowerStr  = str;
    for(char& c : lowerStr){
        c = tolower(c); // convert each character to lower case
    }
    return lowerStr; // return converted lower-cased string
}

// Checks if the given keyword exists in the line (case-sensitive or case-insensitive)
bool matchString(const std::string& line, const std::string& givenStr, bool caseInsensitive){
    if(caseInsensitive){
        string lowerLine = toLowerStr(line);
        string lowerStr = toLowerStr(givenStr);
        return (lowerLine.find(lowerStr) != string::npos); // check if keyword is in the line (caseInsensitive)
    }
    return (line.find(givenStr) != string::npos);  // check if keyword is in the line (caseSensitive)
}

// Reads the text file line-by-line, checks for matches, and prints matching lines
void readTxtFile(const string& filename, const string& keyword, bool caseInsensitive, bool showLineNum){
    ifstream file(filename); // open file to read
    if(!file.is_open()){ // check if file is opened successfully
        cout << "Error opening the file.\n";
        return;
    }
    string line;
    int countLine = 0; // to count the line number
    while(getline(file, line)){
        countLine++;
        if(matchString(line, keyword, caseInsensitive)){ // if each line contain keyword
            if(showLineNum){
                cout << "Line " << countLine << ": " << line << endl; // print with line number
            }else{
                cout << line << endl;  // print without line number
            }
        }
    }
    file.close();  // close file after reading
}
```

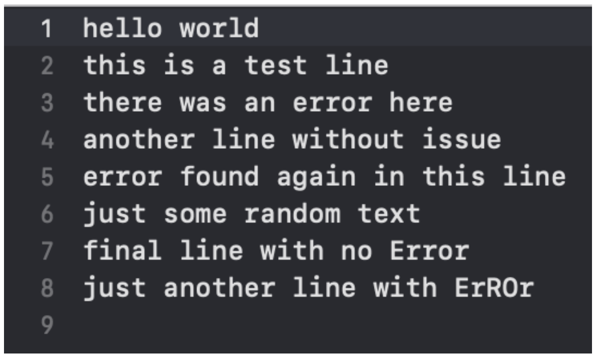
main.cpp



```cpp
int main(int argc, const char * argv[]) {
    string filename, keyword; // variables to hold user inputs

    bool caseInsensitive = false; // flags for command-line
    bool showLineNum = false;

    if(argc < 3 || argc > 5){
        cout << "Usage: ./mygrep [-i] [-n] <keyword> <filename>\n"; // output error msg if user num of arguments if too many or too few
        return 1;
    }else {
        if(argc == 5){  // case when provided both -i and -n
            keyword = argv[3];
            filename = argv[4];
            string flag1 = string(argv[1]);
            string flag2 = string(argv[2]);
            if(flag1 == "-i" && flag2 == "-n" || flag1 == "-n" && flag2 == "-i"){ // check if the flags are valid
                caseInsensitive = true;
                showLineNum = true;
            }else {
                cout << "Invalid flags.\n";
                return 1;
            }
            readTxtFile(filename, keyword, caseInsensitive, showLineNum);
        }else if(argc == 4){  // case when provided with either -i or -n
            keyword = argv[2];
            filename = argv[3];
            if(string(argv[1]) == "-i"){
                caseInsensitive = true;
            }else{
                showLineNum = true;
            }
            readTxtFile(filename, keyword, caseInsensitive, showLineNum);
        }else { // case when provided none of the flags
            keyword = argv[1];
            filename = argv[2];
            readTxtFile(filename, keyword, caseInsensitive, showLineNum);
        }
    }

    return 0;
}
```

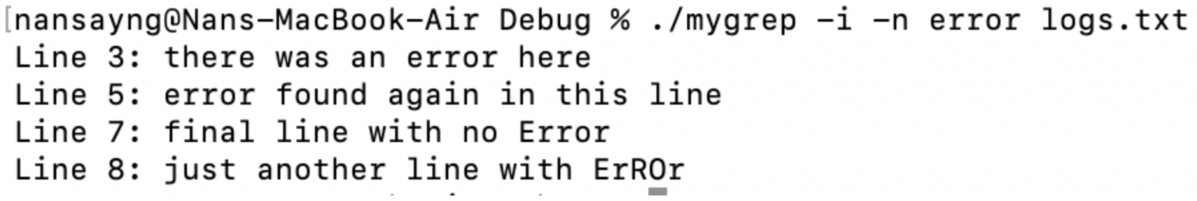
Part 4 – Demo Output

logs.txt file



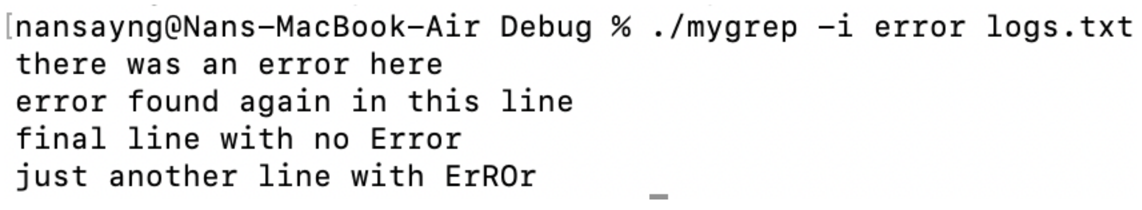
```
hello world
this is a test line
there was an error here
another line without issue
error found again in this line
just some random text
final line with no Error
just another line with ErROr

```

Ran with both -i and -n flags (both for caseInsensitive and showLineNum)



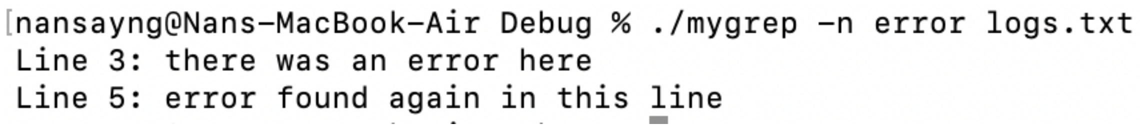
```
[nansayng@Nans-MacBook-Air Debug % ./mygrep -i -n error logs.txt
 Line 3: there was an error here
 Line 5: error found again in this line
 Line 7: final line with no Error
 Line 8: just another line with ErROr
```

Ran with only -i (caseInsensitive)



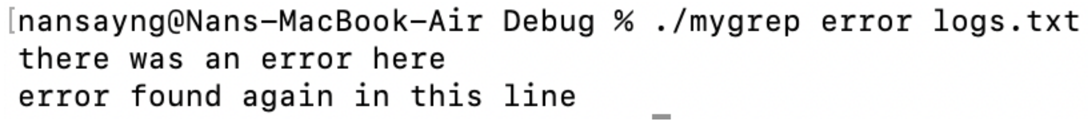
```
[nansayng@Nans-MacBook-Air Debug % ./mygrep -i error logs.txt
 there was an error here
 error found again in this line
 final line with no Error
 just another line with ErROr
```

Ran with only -n (showLineNum)



```
[nansayng@Nans-MacBook-Air Debug % ./mygrep -n error logs.txt
 Line 3: there was an error here
 Line 5: error found again in this line
```

Ran with neither -i or -n



```
[nansayng@Nans-MacBook-Air Debug % ./mygrep error logs.txt
 there was an error here
 error found again in this line
```

<u>Part 5 – What I Learned</u>

What was harder than expected?

One thing that harder than I thought would be handling the command-line arguments which is (***argc*** and ***argv***). At first, I was consumed that I can handle the argv array as the strings, but it was an array of C-style string which is character pointers, So I had to convert them to string to make string comparisons.

Another thing is I had to carefully organize my if statements to make sure the program picked the right functions based on how many arguments or what kind of flags are passed in.

What new C++ feature or behavior surprised you?

I was surprised by how useful the *grep* is in general. And also how useful getline() is when reading the txt file line by line. I thought reading files would be hard but using ifstream and getline(), it made it so much easier. And also find() to check if keyword is in the line was useful, another thing that I've learned and surprised me the most is **string::npos** , which was very helpful for string matching.

Why can't AI just write this perfectly?

I think AI can somewhat help us get started, fix the errors, or for explanation that we can't understand sometimes. But I think that when I built this myself, I had to test it, fix bugs and make sure it handled different inputs or flags correctly. That's something AI can't always do because it doesn't know exactly how should setup or logic is supposed to work.