**MyGrep Project**

Petteri Laukkanen

Email: [petteri.laukkanen@tuni.fi](mailto:petteri.laukkanen@tuni.fi)

20.2.2023

This project can be found at

MyGrep is a command-line based utility written in order to show what I have learned during the course and the state if my skills in writing C++ and it meets the requirements of the highest score possible for the project. The project is a simplified version of the grep utility that is found on Linux. The utility is capable of performing a string search within a given string, searching for text within a text file and searching for text within a text file with additional starting arguments.

The starting arguments that can be used with the MyGrep utility are as follows:

'-o’ - base for options to be added to. If this argument is used alone, it shows the  
 variables list.

'-ol' - shows line numbering.

'-oo' - prints the number of occurrences of the search string.

'-or' - performs a reverse search.

'-oi' - makes the search case-insensitive.

If the MyGrep utility is used with no arguments, the user is prompted to input a string to search from and a string to search for. The program then searches for the string and outputs the position of the first occurrence of the search string in the target string.

If the MyGrep utility is used with two arguments and the second argument is '-o', a help page is outputted to the console.

If the MyGrep utility is used with more than two arguments, the program will search for the search string in a text file. The text file to search from is specified as the third argument. If the file could not be opened, the program informs the user. If the file was opened, the program goes through every line of the file and outputs the lines that contain the search string.

If the MyGrep utility is used with four arguments and the second argument starts with '-o', the program will search for the search string in a text file with additional arguments. The third argument specifies the file to search from, while the fourth argument is the search string. If the search string is found within the file, the program outputs the line that contains the search string. The starting arguments '-ol', '-oo', '-or', and '-oi' can be used to further modify the search.

**Compiling and using the utility**

Starting with compiling the project

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

As you can see the build succeeded with nothing failed or skipped

Next, we place our file that will be used to test the utility in the same folder that mygrep is in.

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

Then, we will open a command prompt window and navigate to the folder above, using the cd command.

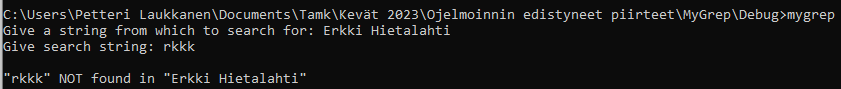
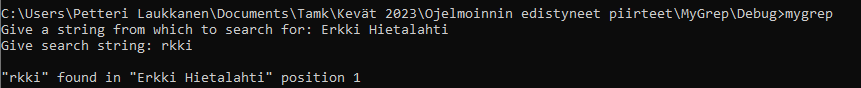
Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

Now we can start testing the utility.

**Increment one**

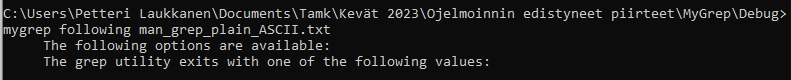
We will use mygrep without parameters and test two cases: search string was found and search string wasn’t found.



Here we can see that the one with “rkki” the text was found and with “rkkk” the text wasn’t found

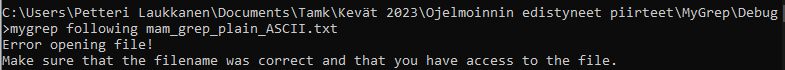
**Increment two**

We will use mygrep with two parameter: string to search for “following” and the file to search from “man\_grep\_plain\_ASCII.txt



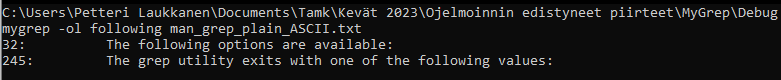
We see that mygrep returns the two lines containing the word “following”.

To display handling of special cases if for example the file is not present, we will use the same command with “man\_grep” mis-spelled “mam-grep”



**Increment three**

We will use mygrep with three parameters: starting arguments -o(l for line numbering, o for number of occurrences or both for line numbering and number of occurrences) The program checks for both parameters separately so the order doesn’t matter.

  
Only with line numbering

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti  
Only with number of occurrences

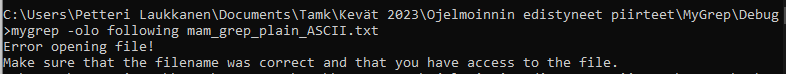
Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti  
With both line numbering and occurrences

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti  
Same command with arguments in different order to show that the order of arguments doesn’t matter.

To display handling of special cases if for example the file is not present, we will use the same command with “man\_grep” mis-spelled “mam-grep”



**Increment four**

We will use mygrep with the newly introduced starting arguments: i (case insensitive) and r (reverse search)

The same procedure for argument handling is used, meaning that argument order does not matter.

With the argument reverse search, we can see that the program prints out the entire file with the two lines containing following missing:

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

…

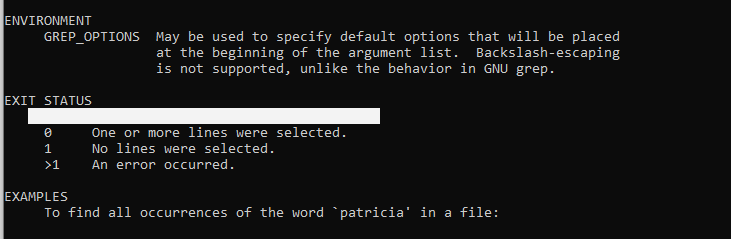
Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti  
Utility

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti  
Original File

…

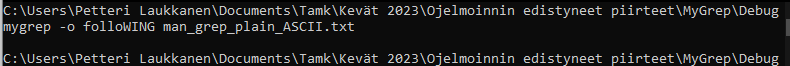
  
Utility

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti  
Original File

Using argument case insensitive:

We will spell following -> folloWING and as you can see:

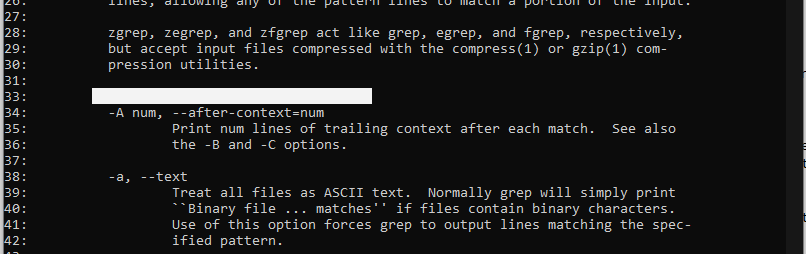
  
Without the argument i, no lines were found.

Kuva, joka sisältää kohteen teksti, sisä, näyttökuva, sulje

Kuvaus luotu automaattisesti  
Using the argument i and the same spelling we can see that the two lines are found.

Lastly, we will be using all of the arguments at once and in two different orders, again to show that: 1- multiple arguments can be combined and 2- the order of arguments does not matter:

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti  
…  
  
…  
Kuva, joka sisältää kohteen teksti, näyttö, musta, valkokangas

Kuvaus luotu automaattisesti  
…

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

As we can see, all of the arguments have worked:

Reverse search: all but two of the lines were printed out.

Line numbering: as seen on the left side of the output.

Case insensitive: the line 32 and 245 are missing because they contained “following”

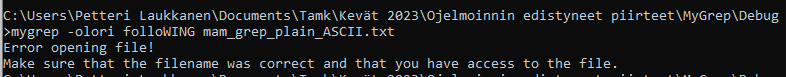
Occurrences: At the end we can see that there were 299 occurrences of lines NOT containing folloWING.

Here is an example with the arguments l, o and i:

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

And lastly to display handling of special cases if for example the file is not present, we will use the same command with “man\_grep” mis-spelled “mam-grep”



In addition, I added a small help page to this project which can be accessed via the command mygrep -o, this will display explanations and syntax for the commands the same way most terminal/command prompt utilities do. It was not mentioned in the instructions for this project, but since there was no outcome required for “mygrep -o”, it does not impact the functionality of the required parts of the program.

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

**After word**

All around this project took me around 7 hours to make:

14.2 0.5hours Increment 1

15.2 2.0hours Increment 2

16.2 1.5hours Increment 3

17.2 1.0hours Increment 4

20.2 2.0hours Documentation

During this project I learned a lot about handling text files using C++ and I also just got generally better at using and documenting C++.

I successfully implemented all four increments, and I am aiming for 5 points from this project.

Mygrep code:

Mygrep.cpp:

/\* MyGrep Project

Made by Petteri Laukkanen

This is a simple version of the grep utility found in linux.

This project meets the reguirements of the highest score available for this project.

With mygrep you can:

search for text in a given string.

Usage:

mygrep [string to search for] [string to search from]

search for text in a textfile.

Usage: mygrep [string to search for] [file to search from]

search for text in a textfile with added starting arguments:

Usage: mygrep -o[arguments to be used] [string to search for] [file to search from]

For example: mygrep - olo following man\_grep\_plain\_ASCII.txt

list possible arguments and their syntax (kind of a help page)

Usage mygrep -o

Possible variables:

-o base for options to be added to. (If alone shows this Variables list)

-ol Show line numbering

-oo Print amount of occurences

-or Reverse search

-oi Make search case insensitive

\*/

#include <iostream>

#include <fstream>

#include <string>

#include<clocale>

#include <algorithm>

using namespace std;

//Giving main two arguments, argc (how many arguments are present) and argv (what those arguments are)

int main(int argc, char\* argv[]) {

setlocale(LC\_ALL, "finnish");

string search; //Storing string that user wants to search for

string target; //Storing string that user wants to search from

//if the value of argc is one (mygrep command was executed with no parameters)

if (argc == 1) {

//Asking user for string from which to search from and using get line to get the entire string that was inputted.

cout << "Give a string from which to search for: ";

getline(cin, target);

//Asking user for string to find and using get line to get the entire string that was inputted.

cout << "Give search string: ";

getline(cin, search);

size\_t found = target.find(search); //Finding the the position of the first occurence of search in target and storing it in found.

//If the value of found is not npos meaning that the search string was found in the target.

if (found != string::npos) {

//Output the what was found and in what position.

cout << "\n\"" << search << "\" found in \"" << target << "\" position " << found;

}

//If the string wasn't found, inform user that the search string wasn't found in the target string.

else {

cout << "\n\"" << search << "\" NOT found in \"" << target << "\"";

}

}

//else if the value of argc c is 2 and the second argument is -o (mygrep -o), output help "page"

else if (argc == 2 && string(argv[1]) == "-o") {

cout << "\nVariables\n\n";

cout << "Usage:\n";

cout << "mygrep -o[Variables to be used] [text to search for] [file to search from]\n";

cout << "For example: mygrep - olo following man\_grep\_plain\_ASCII.txt\n\n";

cout << "Variables can be stacked!\n";

cout << "List of possible variables:\n";

cout << "-o base for options to be added to.\n If alone shows this Variables list.\n\n";

cout << "-ol Show line numbering\n";

cout << "-oo Print amount of occurences\n";

cout << "-or Reverse search\n";

cout << "-oi Make search case insensitive";

}

//else (There are more than 2 arguments)

else {

string filename; //Filename to be opened

ifstream file; //ifstream variable for storing the file where to search from.

string variables = argv[1]; //Storing the second argument from argv to string variables (array starts from 0)

//If valua of argc is 3

if (argc == 3) {

search = argv[1]; //Store second variable in search

filename = argv[2]; //Store third variable in filename

file.open(filename); //Open file with the name stored in filename

//(Special case) //If the file couldn't be opened (Name was incorrect or the file doesn't exist / is corrupted, inform the user

if (file.fail()) {

cerr << "Error opening file!\nMake sure that the filename was correct and that you have access to the file.";

return 0;

}

string line; //String to store current line that is being compared to string search

//While there are new lines to go through in file (Goes through every line of the file)

while (getline(file, line)) {

//If the search finds the string search in the file, output the line that contained the search string

if (line.find(search) != string::npos) {

cout << line << endl;

}

}

}

//If value of argc is 4 and the variable called variables starts with -o (Testing for correct syntax)

if (argc == 4 && variables.substr(0, 2) == "-o") {

//Create booleans for all possible starting arguments

bool l = false;

bool o = false;

bool r = false;

bool i = false;

size\_t pos = 1; //Variable for storing position while searching for the occurence of the letter "o" starting from 1 because postion 0 never contains an "o"

int count\_o = 0; //Store how many time o has been found

//While pos is not npos (the current position doesn't have an "o")

while (pos != string::npos) {

count\_o++; //Add 1 to count\_o

pos = variables.find('o', pos + 1); //find "o" in the argument starting from position pos + 1 (Starting search after first "o" that is always there

}

//Testing for which variables should be active.

//if the previous while loop found a second "o" in the argument set boolean o to true

if (count\_o > 1) {

o = true;

}

//If there is an "l" in the variables string, set l to true

if (variables.find('l') != string::npos) {

l = true;

}

//If there is an "r" in the variables string, set r to true

if (variables.find('r') != string::npos) {

r = true;

}

//If there is an "i" in the variables string, set i to true

if (variables.find('i') != string::npos) {

i = true;

}

int line\_number = 1; //Store line where the search is currently

int line\_count = 0; //Store how many times the search string has been found

search = argv[2]; //String to search for is the third argument given

filename = argv[3]; //Filename of file to search from is the fourth argument given

file.open(filename); //Open file with the name in "filename"

//(Special case) If the file couldn't be opened (Name was incorrect or the file doesn't exist / is corrupted, inform the user

if (file.fail()) {

cerr << "Error opening file!\nMake sure that the filename was correct and that you have access to the file.";

return 0;

}

string line; //Store current line

string line\_copy; //Store copy of current line to restore original format

string search\_copy = search; //Store copy of the search string to restore original format

//If r (reverse search) was not in the parameters

if (r == false) {

//Get every line in the file one by one until at the end of the file

while (getline(file, line)) {

line\_copy = line; //Store the current line into line\_copy

//If i (case insensitive) was in the parameters

if (i == true) {

//Make every character of the search and line strings lowercase (enables searching case insensitively)

transform(search.begin(), search.end(), search.begin(), ::tolower);

transform(line.begin(), line.end(), line.begin(), ::tolower);

}

//if the current line contains the search string

if (line.find(search) != string::npos) {

//If l (line numbering) was in the parameters

if (l == true) {

//Output the current line number

cout << line\_number << ": ";

}

cout << line\_copy << endl; //Output the current line from line\_copy so upper case letters are shown correctly

line\_count++; // Add one the the amount of linus that have included the search string

}

line\_number++; //Add one to the current line number

}

//If o (Occurences) was in the parameters

if (o == true) {

//Output the amount of lines that contained the search string

cout << "\nOccurences of lines containing \"" << search\_copy << "\": " << line\_count;

}

}

//If r (reverse search) was in the parameters

else if (r == true) {

//Get every line in the file one by one until at the end of the file

while (getline(file, line)) {

line\_copy = line; //Store the current line into line\_copy

//Make every character of the search and line strings lowercase (enables searching case insensitively)

if (i == true) {

transform(search.begin(), search.end(), search.begin(), ::tolower);

transform(line.begin(), line.end(), line.begin(), ::tolower);

}

//If current line include search string

if (line.find(search) == string::npos) {

//If l (line numbering) was in the parameters

if (l == true) {

//Output line number before every line

cout << line\_number << ": ";

}

cout << line\_copy << endl; //Output the current line from line\_copy so upper case letters are shown correctly

line\_count++; //Add one the the amount of lines found containing the search string

}

line\_number++; //Add one to the current line number

}

//Output how many rows NOT containing the search string were in the file

cout << "\nOccurences of lines NOT containing \"" << search\_copy << "\": " << line\_count;

}

}

}

}