



Mygrep-projekti

C++

Matias Sydänmaa

Mygrep
Maaliskuu 2025

Tietotekniikan Insinööri

SISÄLLYS

| | | |
|-----|--|---|
| 1 | OHJELMAN SUUNNITTELURATKAISU | 3 |
| 2 | OHJELMAN TOIMINTA | 4 |
| 2.1 | Todisteet onnistuneesta kääntämisestä ja linkittämisestä | 4 |
| 2.2 | Todisteet ohjelman toiminnasta | 4 |
| 3 | TYÖAIKAKIRJANPITO | 6 |
| 4 | OPPIMINEN JA INKREMENTIT | 7 |
| 4.1 | OPPIMINEN | 7 |
| 4.2 | INKREMENTIT | 7 |
| 5 | OSOITE GIT-REPOON JA YHTEYSTIEDOT | 8 |
| | LÄHDEKOODI | 9 |

1 OHJELMAN SUUNNITTELURATKAISU

Ohjelma toteuttaa grep-tyylisen haun joko käyttäjän antamasta syötteestä tai tiedostosta. Se tukee optioita, kuten

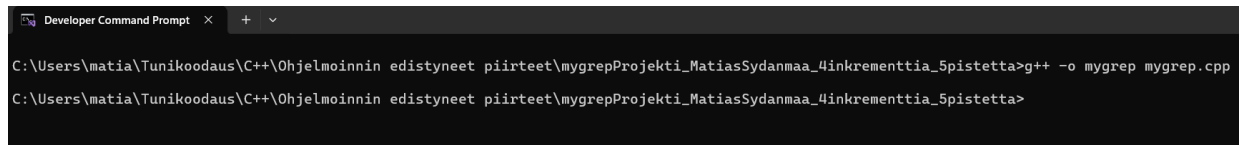
- **-l** – tulostaa rivin numeron, jos hakusana löytyy
- **-o** – tulostaa hakusanan esiintymismäärän
- **-r** – kääntää haun (näyttää rivit, joissa hakusanaa ei esiinny)
- **-i** – hakee kirjainkoosta riippumatta hakusanan

Ohjelma hyödyntää tiedostonkäsittelyä ja komentoriviparametreja hakutoiminnon toteuttamiseksi. Se sisältää myös poikkeuskäsittelyn esimerkiksi silloin, kun tiedoston avaaminen epäonnistuu.

2 OHJELMAN TOIMINTA

2.1 Todisteet onnistuneesta kääntämisestä ja linkittämisestä

Ohjelmassa käytetään G++-kääntäjää, joka on osa GNU Compiler Collectionia (GCC). GCC:stä käytetään MinGW-versiota. Kääntäminen suoritetaan `g++ -o mygrep mygrep.cpp` komennolla.



```

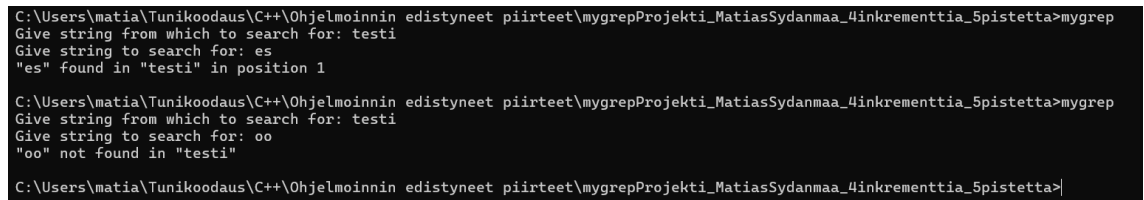
C:\Users\matia\Tunikoodaus\C++\Ohjelmoinnin edistyneet piirteet\mygrepProjekti_MatiasSydanmaa_4inkrementtia_5pistetta>g++ -o mygrep mygrep.cpp
C:\Users\matia\Tunikoodaus\C++\Ohjelmoinnin edistyneet piirteet\mygrepProjekti_MatiasSydanmaa_4inkrementtia_5pistetta>

```

KUVA 1. Ohjelman onnistunut kääntäminen

2.2 Todisteet ohjelman toiminnasta

Ohjelman toimivuuden testaamiseen käytetään liitteenä olevaa `man_grep_plain_ASCII.txt` tiedostoa. Ohjelmassa toimivat kaikki 4. inkrementtiä. Seuraavat näyttökuvat todistavat ohjelman toimivuuden, koska tulostus on identtinen tehtävänantotiedostossa olevien esimerkkien kanssa ja ohjelma lukee samaa tiedostoa molemmissa.



```

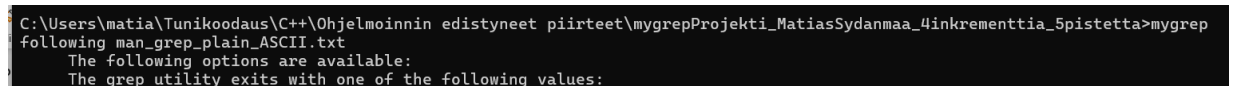
C:\Users\matia\Tunikoodaus\C++\Ohjelmoinnin edistyneet piirteet\mygrepProjekti_MatiasSydanmaa_4inkrementtia_5pistetta>mygrep
Give string from which to search for: testi
Give string to search for: es
"es" found in "testi" in position 1

C:\Users\matia\Tunikoodaus\C++\Ohjelmoinnin edistyneet piirteet\mygrepProjekti_MatiasSydanmaa_4inkrementtia_5pistetta>mygrep
Give string from which to search for: testi
Give string to search for: oo
"oo" not found in "testi"

C:\Users\matia\Tunikoodaus\C++\Ohjelmoinnin edistyneet piirteet\mygrepProjekti_MatiasSydanmaa_4inkrementtia_5pistetta>

```

KUVA 2. 1. Inkrementin toimivuus todistettu

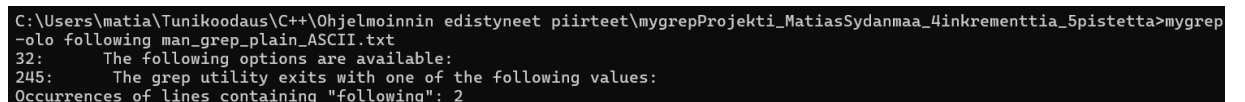


```

C:\Users\matia\Tunikoodaus\C++\Ohjelmoinnin edistyneet piirteet\mygrepProjekti_MatiasSydanmaa_4inkrementtia_5pistetta>mygrep
following man_grep_plain_ASCII.txt
The following options are available:
The grep utility exits with one of the following values:

```

KUVA 3. 2. Inkrementin toimivuus todistettu



```

C:\Users\matia\Tunikoodaus\C++\Ohjelmoinnin edistyneet piirteet\mygrepProjekti_MatiasSydanmaa_4inkrementtia_5pistetta>mygrep
-o following man_grep_plain_ASCII.txt
32: The following options are available:
245: The grep utility exits with one of the following values:
Occurrences of lines containing "following": 2

```

KUVA 4. 3. Inkrementin toimivuus todistettu

```
C:\Users\matia\Tunikoodaus\C++\Ohjelmoinnin edistyneet piirteet\mygrepProjekti_MatiasSydanmaa_4inkrementtia_5pistetta>mygrep -or following man_grep_plain_ASCII.txt

GREP(1)                                BSD General Commands Manual                                GREP(1)

NAME
  grep, egrep, fgrep, zgrep, zegrep, zfgrep -- file pattern searcher

SYNOPSIS
  grep [-abcdDEFGHhIiJlLnOopqRSsUVvwXZ] [-A num] [-B num] [-C[num]]
    [-e pattern] [-f file] [--binary-files=value] [--color[=when]]
    [--colour[=when]] [--context[=num]] [--label] [--line-buffered]
    [--null] [pattern] [file ...]

DESCRIPTION
  The grep utility searches any given input files, selecting lines that
  match one or more patterns. By default, a pattern matches an input line
  if the regular expression (RE) in the pattern matches the input line
  without its trailing newline. An empty expression matches every line.
  Each input line that matches at least one of the patterns is written to
  the standard output.

  grep is used for simple patterns and basic regular expressions (BREs);
  egrep can handle extended regular expressions (EREs). See re_format(7).
```

KUVA 5. 4. Inkrementin toimivuus todistettu

```
C:\Users\matia\Tunikoodaus\C++\Ohjelmoinnin edistyneet piirteet\mygrepProjekti_MatiasSydanmaa_4inkrementtia_5pistetta>mygrep -oi folloWING man_grep_plain_ASCII.txt
  The following options are available:
  The grep utility exits with one of the following values:
Occurrences of lines containing "folloWING": 2
```

KUVA 6. 4. Inkrementin toimivuus todistettu

```
C:\Users\matia\Tunikoodaus\C++\Ohjelmoinnin edistyneet piirteet\mygrepProjekti_MatiasSydanmaa_4inkrementtia_5pistetta>mygrep -olori folloWING man_grep_plain_ASCII.txt
1:
2: GREP(1)                                BSD General Commands Manual                                GREP(1)
3:
4: NAME
5:   grep, egrep, fgrep, zgrep, zegrep, zfgrep -- file pattern searcher
6:
7: SYNOPSIS
8:   grep [-abcdDEFGHhIiJlLnOopqRSsUVvwXZ] [-A num] [-B num] [-C[num]]
9:     [-e pattern] [-f file] [--binary-files=value] [--color[=when]]
10:    [--colour[=when]] [--context[=num]] [--label] [--line-buffered]
11:    [--null] [pattern] [file ...]
12:
13: DESCRIPTION
14:   The grep utility searches any given input files, selecting lines that
15:   match one or more patterns. By default, a pattern matches an input line
16:   if the regular expression (RE) in the pattern matches the input line
17:   without its trailing newline. An empty expression matches every line.
18:   Each input line that matches at least one of the patterns is written to
19:   the standard output.
20:
21:   grep is used for simple patterns and basic regular expressions (BREs);
```

KUVA 7. 4. Inkrementin toimivuus todistettu

```
C:\Users\matia\Tunikoodaus\C++\Ohjelmoinnin edistyneet piirteet\mygrepProjekti_MatiasSydanmaa_4inkrementtia_5pistetta>move man_grep_plain_ASCII.txt man_grep_plain_ASCII.txt2
  1 file(s) moved.

C:\Users\matia\Tunikoodaus\C++\Ohjelmoinnin edistyneet piirteet\mygrepProjekti_MatiasSydanmaa_4inkrementtia_5pistetta>mygrep -olori folloWING man_grep_plain_ASCII.txt
An exception occurred. Exception Nr. -1
Error: file could not be opened
```

KUVA 8. Ajoesimerkki poikkeuskäsittelystä

3 TYÖAIKAKIRJANPITO

| Päivämäärä | Tunnit | Mitä tehty |
|------------|--------|---------------|
| 26.2.2025 | 2 | Suunnittelu |
| 26.2.2025 | 2 | Koodaaminen |
| 27.2.2025 | 4 | Koodaaminen |
| 27.2.2025 | 1 | Testaus |
| 3.3.2025 | 4 | Dokumentointi |

4 OPPIMINEN JA INKREMENTIT

4.1 OPPIMINEN

Projektin aikana opin tiedostojen käsittelystä C++:ssa. Harjoittelin tiedostojen avaamista ja lukemista ifstream-olion avulla ja opin tunnistamaan yleiset virheet, kuten tilanteet, joissa tiedostoa ei löydy tai sitä ei voida avata.

Poikkeusten käsittelyssä opin, kuinka try-catch-rakenteella voidaan hallita virhetilanteita ja antaa käyttäjälle selkeämpiä virheilmoituksia. Tämä auttoi ohjelman luotettavuuden parantamisessa ja teki siitä käyttäjäystävällisemmän.

Ohjelman modularisointi ja optioiden käsittely olivat myös tärkeitä oppeja. Opin, kuinka komentoriviparametreja voidaan lukea ja käsitellä joustavasti. Optioiden tarkistaminen ja niiden vaikutusten toteuttaminen oli mielenkiintoinen osa projektia, ja sain paremman käsityksen siitä, miten ohjelmille voi lisätä erilaisia käyttömahdollisuuksia.

4.2 INKREMENTIT

Tavoittelen 5 pistettä, koska tein inkrementit 1, 2, 3 ja 4.

5 OSOITE GIT-REPOON JA YHTEYSTIEDOT

| | |
|------------|---|
| Sähköposti | matias.sydanmaa@tuni.fi |
| Git-repo | https://github.com/maddys1/Mygrep-C--Tuni/tree/main |

LÄHDEKOODI

mygrep.cpp

```
#include <iostream>
#include <string>
#include <fstream>
#include <algorithm>
#include <exception>
using namespace std;

// Function to convert a string to lower case
string toLowerCase(const string& str) {
    string lowerStr = str;
    transform(lowerStr.begin(), lowerStr.end(), lowerStr.begin(), ::tolower);
    return lowerStr;
}

int main(int argc, char* argv[]) {
    try {
        if (argc == 1) {
            string inputString, searchString;

            // Ask user for input
            cout << "Give string from which to search for: ";
            getline(cin, inputString);

            cout << "Give string to search for: ";
            getline(cin, searchString);

            // Look for the string in the input string
            size_t position = inputString.find(searchString);
            if (position != inputString.npos) {
                cout << "\"" << searchString << "\" found in \"" << inputString << "\" in
position " << position << endl;
            } else {
                cout << "\"" << searchString << "\" not found in \"" << inputString << "\""
<< endl;
            }
        } else if (argc >= 3) {
            string searchString, filename;
            string options = "";
            bool lineNumbering = false;
            bool countOccurrences = false;
            bool reverseSearch = false;
            bool ignoreCase = false;

            // Handle optional argument order
            int argIndex = 1;
            if (argv[argIndex][0] == '-') {
                options = argv[argIndex];
```

```

        argIndex++;
    }
    if (argc - argIndex < 2) {
        throw runtime_error("Error: Missing search string or filename");
    }
    searchString = argv[argIndex];
    filename = argv[argIndex + 1];

    // Parse options
    lineNumbering = options.find('l') != string::npos;
    countOccurrences = options.find('o') != string::npos;
    reverseSearch = options.find('r') != string::npos;
    ignoreCase = options.find('i') != string::npos;

    // Try to open the file
    ifstream file(filename);
    if (!file) {
        throw runtime_error("Error: file could not be opened");
    }

    string line;
    int lineNumber = 0;
    int occurrences = 0;
    while (getline(file, line)) {
        lineNumber++;
        string searchLine = line;
        string searchStr = searchString;

        // Convert to lower case if ignoreCase option is set
        if (ignoreCase) {
            searchLine = toLowerCase(line);
            searchStr = toLowerCase(searchString);
        }

        // Check if the search string is found in the line
        bool found = searchLine.find(searchStr) != string::npos;
        if (reverseSearch) {
            found = !found;
        }

        // Print the line if the search string is found (or not found if reverseSearch
is set)
        if (found) {
            if (lineNumbering) {
                cout << lineNumber << ": ";
            }
            cout << line << endl;
            occurrences++;
        }
    }

    // Print the number of occurrences if countOccurrences option is set

```

```

        if (countOccurrences) {
            if (reverseSearch) {
                cout << "Occurrences of lines NOT containing \"" << searchString <<
"\": " << occurrences << endl;
            } else {
                cout << "Occurrences of lines containing \"" << searchString << "\":
" << occurrences << endl;
            }
        }
    } else {
        // Print usage information if the arguments are incorrect
        cerr << "Usage: \n" << " ./mygrep (interactive mode) \n" << " ./mygrep [-
options] <search> <filename> (search in file)" << endl;
    }
} catch (const exception& e) {
    // Handle exceptions and print error message
    cerr << "An exception occurred. Exception Nr. -1\n" << e.what() << endl;
}
return 0;

```