

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут
імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи №2 з дисципліни
«Алгоритми та структури даних 2. Структури даних»

«Метод декомпозиції. Пошук
інверсій»
Варіант 3

Виконав студент ІП-15, Борисик Владислав Тарасович
(шифр, прізвище, ім'я, по батькові)

Перевірів Соколовський Владислав Володимирович
(прізвище, ім'я, по батькові)

Код програми:

main.cpp:

```
#include "functions.h"

int main(int arg_count, char *arg_values[]) {

    string file_name = get_input_file_name(arg_values);
    int person_number = get_person_number(arg_values);

    int users_amount = get_users_amount(file_name);
    int films_amount = get_films_amount(file_name);

    string text = read_input_file(file_name);
    string matrix_string = parse_matrix_string(text);
    vector<vector<int>> matrix = parse_matrix(matrix_string);

    cout << "Matrix on input:\n";
    print_matrix(matrix, users_amount, films_amount);

    cout << "\nSliced matrix (sliced user index in first
column):\n";
    vector<vector<int>> sliced_matrix = slice_matrix(matrix,
users_amount);
    print_matrix(sliced_matrix, users_amount, films_amount - 1);

    vector<Result> results = find_preferences(sliced_matrix,
users_amount, person_number - 1);

    sort_results(results);

    write_output_file(results, person_number);
}
```

functions.h:

```
#ifndef ASD_LABWORK_2_FUNCTIONS_H
#define ASD_LABWORK_2_FUNCTIONS_H

#endif //ASD_LABWORK_2_FUNCTIONS_H

#include <iostream>
#include <string>
#include <fstream>
#include <vector>
#include "structs.h"
using namespace std;

string get_input_file_name(char *arg_values[]);
int get_person_number(char *arg_values[]);
int get_users_amount(string file_name);
int get_films_amount(string file_name);
string read_input_file(string file_name);
string parse_matrix_string(string text);
vector<vector<int>> parse_matrix(string matrix_string);
void print_matrix(vector<vector<int>> matrix, int rows, int
columns);
void merge_sorting(vector<int> &matrix_row_main, vector<int>
&matrix_row_secondary);
vector<vector<int>> slice_matrix(vector<vector<int>> matrix, int
rows);
vector<Result> find_preferences(vector<vector<int>> matrix, int
rows, int person_index);
void write_output_file(vector<Result> results, int person_number);
void sort_results(vector<Result> &results);
```

functions.cpp:

```
#include "functions.h"
```

```
//отримуємо назву вхідного файлу із командного рядка (перший аргумент)
```

```
string get_input_file_name(char *arg_values[]){  
    string file_name = arg_values[1];  
  
    return file_name;  
}
```

```
//отримуємо номер людини, за якою будемо рахувати вподобання, із командного рядка (другий аргумент)
```

```
int get_person_number(char *arg_values[]){  
    int person_number = stoi(arg_values[2]);  
  
    return person_number;  
}
```

```
//отримуємо перший рядок із вхідного файлу (містить кількість користувачів і фільмів)
```

```
string read_first_line(string file_name){  
    // потік read_file (тільки для читання)  
    ifstream read_file(file_name);  
    string line;  
  
    getline(read_file, line);  
  
    return line;  
}
```

```
//отримуємо кількість користувачів із першого рядку
```

```
int get_users_amount(string file_name){  
    string line = read_first_line(file_name);  
    cout << "line: " << line << '\n';  
  
    int space_position = line.find(' ');  
    cout << "space_position: " << space_position << '\n';  
  
    string users_amount_string = line.substr(0, space_position);  
    cout << "users_amount_string: " << users_amount_string << '\n';  
  
    int users_amount = stoi(users_amount_string);  
    cout << "users_amount: " << users_amount << '\n';  
}
```

```
        return users_amount;
    }

    //отримуємо кількість фільмів із першого рядку
    int get_films_amount(string file_name){
        string line = read_first_line(file_name);

        int space_position = line.find(' ');

        string films_amount_string = line.substr(space_position + 1,
line.size());

        int films_amount = stoi(films_amount_string);

        return films_amount;
    }

    //отримуємо дані з вхідного файлу
    string read_input_file(string file_name){
        ifstream read_file(file_name);
        string text;
        string line;

        while (getline(read_file, line)){
            text += line + '\n';
        }
        read_file.close();

        text = text.substr(0, text.size()-1);

        return text;
    }

    //конвертуємо матрицю в рядковий формат
    string parse_matrix_string(string text){
        int newline_position = text.find('\n');

        string new_text = text.substr(newline_position + 1,
text.size());

        return new_text;
    }

    //розділяємо рядки матриці
    vector<string> split_matrix_rows(const string& text){
```

```

// вектор з рядками
vector<string> lines;
// позиція минулого символу нового рядку
int previous_position = 0;
// лічильник (рахує символи тексту)
int counter = 0;

while(counter <= text.size()){
    // якщо символ дорівнює символу нового рядка або counter
    // дорівнює останньому символу тексту
    if(text[counter] == '\n' || counter == text.size()){
        // рядок (підрядок, який починається з позиції минулого
        // символу нового рядка довжиною counter-previous_position)
        string line = text.substr(previous_position, counter-
previous_position);
        // додаємо цей рядок у вектор
        lines.push_back(line);
        // оновлюємо позицію крапки
        previous_position = counter+1;
    }
    counter++;
}

return lines;
}

```

```

//переводимо матрицю з рядкового формату в двовимірний вектор
vector<vector<int>> parse_matrix(string matrix_string){
    vector<vector<int>> matrix;
    vector<string> matrix_rows = split_matrix_rows(matrix_string);

    for(int i = 0; i < matrix_rows.size(); i++){
        vector<int> matrix_rows_int;
        int space_position = 0;

        while (space_position != string::npos) {
            space_position = matrix_rows[i].find(' ');

            string num_str = matrix_rows[i].substr(0,
space_position);
            int num = stoi(num_str);

            matrix_rows[i].erase(0, space_position + 1);
            matrix_rows_int.push_back(num);
        }
    }
}

```

```

    }

    matrix.push_back(matrix_rows_int);
}

return matrix;
}

//виводимо матрицю в консоль
void print_matrix(vector<vector<int>> matrix, int rows, int
columns){
    for(int i = 0; i < rows; i++) {
        for (int j = 0; j < columns + 1; j++){
            cout << matrix[i][j] << " ";
        }
        cout << '\n';
    }
}

//"обрізаємо" вектор з позиції from до позиції to
vector<int> slice_vector(vector<int> vec, int from, int to){
    vector<int> sliced_vector{};

    for(int i = from; i < to; i++){
        sliced_vector.push_back(vec[i]);
    }

    return sliced_vector;
}

//сортуємо два вектори методом merge_sort (метод декомпозиції)
void merge_sorting(vector<int> &matrix_row_main, vector<int>
&matrix_row_secondary){
    if (matrix_row_main.size() > 1){
        int middle = matrix_row_main.size()/2;

        vector<int> left_vector_main =
slice_vector(matrix_row_main, 0, middle);
        vector<int> right_vector_main =
slice_vector(matrix_row_main, middle, matrix_row_main.size());
        vector<int> left_vector_secondary =
slice_vector(matrix_row_secondary, 0, middle);
        vector<int> right_vector_secondary =
slice_vector(matrix_row_secondary, middle,

```

```

matrix_row_secondary.size());

    merge_sorting(left_vector_main, left_vector_secondary);
    merge_sorting(right_vector_main, right_vector_secondary);

    int i = 0, j = 0, k = 0;

    while (i < left_vector_main.size() and j <
right_vector_main.size()){
        if (left_vector_main[i] < right_vector_main[j]){
            matrix_row_main[k] = left_vector_main[i];
            matrix_row_secondary[k] = left_vector_secondary[i];
            i++;
        }
        else{
            matrix_row_main[k] = right_vector_main[j];
            matrix_row_secondary[k] =
right_vector_secondary[j];
            j += 1;
        }
        k += 1;
    }

    while(i < left_vector_main.size()) {
        matrix_row_main[k] = left_vector_main[i];
        matrix_row_secondary[k] = left_vector_secondary[i];
        i += 1;
        k += 1;
    }

    while (j < right_vector_main.size()){
        matrix_row_main[k] = right_vector_main[j];
        matrix_row_secondary[k] = right_vector_secondary[j];
        j += 1;
        k += 1;
    }
}

}

// "обрізаємо" матрицю (а саме, обрізаємо перший стовпець(який
містить номер користувача))
vector<vector<int>> slice_matrix(vector<vector<int>> matrix, int
rows){
    vector<vector<int>> sliced_matrix{};

```



```

    for(int i = 0; i < rows; i++){
        vector<int> row = matrix[i];
        vector<int> sliced_row = slice_vector(row, 1, row.size());

        sliced_matrix.push_back(sliced_row);
    }

    return sliced_matrix;
}

//сортуємо вподобання користувача
void sort_preferences(vector<vector<int>> &matrix, int rows, int
person_index){
    for(int i = 0; i < rows; i++){
        if(i == person_index)
            continue;
        else{
            vector<int> row_to_sort = matrix[person_index];
            merge_sorting(row_to_sort, matrix[i]);
        }
    }
}

// рахуємо інверсії у векторі
int count_inversions_in_vector(vector<int> vec){
    int inversions = 0;

    for(int i = 0; i < vec.size() - 1; i++){
        for(int j = i + 1; j < vec.size(); j++){
            if(vec[i] > vec[j])
                inversions++;
        }
    }

    return inversions;
}

//рахуємо інверсії у матриці
vector<int> count_inversions_in_matrix(vector<vector<int>> matrix,
int rows){
    vector<int> inversions{};

    for(int i = 0; i < rows; i++){
        int inversions_amount =

```

```

count_inversions_in_vector(matrix[i]);

    inversions.push_back(inversions_amount);
}

return inversions;
}

//генеруємо результати
vector<Result> generate_results(vector<int> inversions, int
person_index){
    vector<Result> results{};

    for(int i = 0; i < inversions.size(); i++){
        Result res{};
        res.person_number = i+1;

        if(i == person_index){
            res.reverses_amount = 0;
        }
        else{
            res.reverses_amount = inversions[i];
        }

        results.push_back(res);
    }

    return results;
}

//знаходимо вподобання
vector<Result> find_preferences(vector<vector<int>> matrix, int
rows, int person_index){
    sort_preferences(matrix, rows, person_index);
    vector<int> inversions = count_inversions_in_matrix(matrix,
rows);
    vector<Result> results = generate_results(inversions,
person_index);

    return results;
}

//сортуюмо результати за зростанням степені подібності вподобань
void sort_results(vector<Result> &results){
    //bubble sort

```

```
    for (int i = 0; i < results.size() - 1; i++)
        for (int j = 0; j < results.size() - i - 1; j++)
            if (results[j].reverses_amount > results[j +
1].reverses_amount)
                swap(results[j], results[j + 1]);
}

//записуємо результати у вихідний файл
void write_output_file(vector<Result> results, int person_number){
    ofstream write_file("ip15_Borysyk_3.txt");

    string text = to_string(person_number) + "\n";

    for(int i = 0; i < results.size(); i++){
        if(results[i].person_number == person_number){
            continue;
        }
        else{
            text += to_string(results[i].person_number) + " " +
to_string(results[i].reverses_amount) + "\n";
        }
    }

    write_file << text;

    write_file.close();
}
```


structs.h:

```
struct Result{  
    int person_number;  
    int reverses_amount;  
};
```

Результат виконання програми

Для тестування оберемо такі ж вхідні дані, як у прикладі завдання. Таким чином можна повністю впевнитись у правильності виконання завдання.

На вхід подамо файл input.txt:

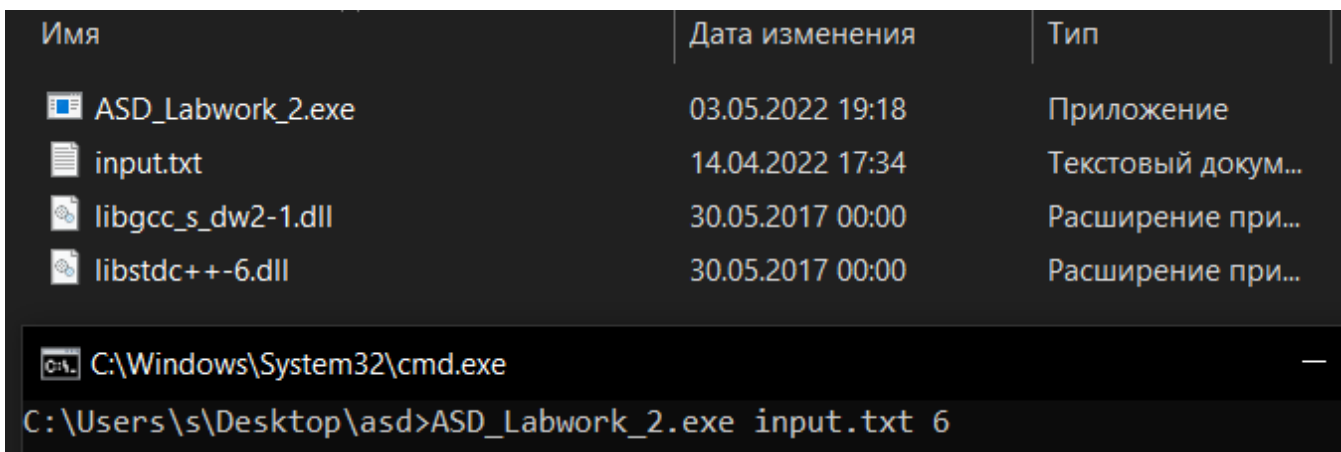
 input.txt – Блокнот

Файл Правка Формат Вид

```
10 5
1 5 2 1 3 4
2 3 2 4 1 5
3 4 5 3 2 1
4 5 1 4 3 2
5 1 2 5 4 3
6 2 5 4 1 3
7 2 4 5 3 1
8 5 3 1 4 2
9 4 5 2 3 1
10 3 1 2 4 5
```

Запускати програму потрібно з консолі. Також необхідно надати два аргументи:

1. назва вхідного файлу (з розширенням)
2. номер людини, з якою будуть порівнюватись інші користувачі



Имя	Дата изменения	Тип	Размер
ASD_Labwork_2.exe			
input.txt			
ip15_Borysyk_3.txt			
libgcc_s_dw2-1.dll			
libstdc++-6.dll			

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.1526]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\s\Desktop\asd>ASD_Labwork_2 input.txt 6
Matrix on input:
1 5 2 1 3 4
2 3 2 4 1 5
3 4 5 3 2 1
4 5 1 4 3 2
5 1 2 5 4 3
6 2 5 4 1 3
7 2 4 5 3 1
8 5 3 1 4 2
9 4 5 2 3 1
10 3 1 2 4 5

Sliced matrix (sliced user index in first column):
5 2 1 3 4
3 2 4 1 5
4 5 3 2 1
5 1 4 3 2
1 2 5 4 3
2 5 4 1 3
2 4 5 3 1
5 3 1 4 2
4 5 2 3 1
3 1 2 4 5
```

Після виконання програми створився файл ip15_Borysyk_3.txt, там зберігаються результати:

Имя	Дата изменения	Тип	Размер
ASD_Labwork_2.exe			
input.txt			
ip15_Borysyk_3.txt			
libgcc_s_dw2-1.dll			
libstdc++-6.dll			

ip15_Borysyk_3.txt – Блокнот

Файл Правка Формат Вид Справка

6
3 3
2 4
7 4
9 4
5 5
1 7
4 7
8 7
10 8

Результати виконання програми збіглися із тим, що дані в прикладі завдання. Отже, програма працює правильно.

Тестування алгоритму

Для тестування роботи алгоритму, перевіримо його вручну. Наприклад, візьмемо користувача 6 і 7. За прикладом, повинно вийти 4 інверсії.

6: 2 5 4 1 3

7: 2 4 5 3 1

① $\begin{array}{ccccc} \downarrow & & & & \\ 2 & 5 & 4 & 1 & 3 \\ \uparrow & & & & \\ & 2 & 4 & 5 & 3 & 1 \end{array}$

② $\begin{array}{ccccc} & & \downarrow & & \\ 1 & 2 & 5 & 4 & 3 \\ & & \uparrow & & \\ 3 & 2 & 4 & 5 & 1 \end{array}$

③ $\begin{array}{ccccc} & & & \downarrow & \\ 1 & 2 & 3 & 5 & 4 \\ & & & \uparrow & \\ 3 & 2 & 1 & 4 & 5 \end{array}$

④ $\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ 3 & 2 & 1 & 5 & 4 \end{array}$

Інверсії:

$(1, 2); (1, 3); (2, 3); (5, 4)$

В результаті отримали 4 інверсії. Програмно також отримали 4 інверсії:

7 4

Отже, алгоритм працює правильно.

Висновок

Під час виконання цієї лабораторної роботи я ознайомився з методом декомпозиції і навчився рахувати кількість інверсій в масиві. В результаті виконання я отримав програму, яка реалізує рекомендаційний алгоритм для сервісу перегляду фільмів.