Лабораторная работа №1

Исследование способов анализа областей эквивалентности и построения тестовых последовательностей

1. Цель работы

Исследовать способы анализа областей эквивалентности входных данных для тестирования программного обеспечения. Приобрести практические навыки составления построения тестовых последовательностей.

2. Постановка задачи

Требуется написать программу, выполняющую заданные действия согласно варианту, определить области эквивалентности входных данных, составить примеры тестовых последовательностей.

Вариант — 22

1) Дана квадратная матрица 3x3. Определить является ли заданная матрица положительно определённой.
2) Дана строка. Преобразовать строку: если нет символа #, то оставить её без изменения, иначе заменить каждый символ, встречающийся после первого вхождения # на символ @.
3) Программа, которая находит минимальную длину строки текстового файла и печатает эту строку.

3. Ход работы

3.1 Была написана программа на языке программирования C++ выполняющая требуемые действия и представлена в приложении А.

3.2 Были определены области эквивалентности входных данных и составлены примеры тестовых последовательностей.

Примечание: в программе не учтены обработки исключительных ситуаций.

Так как размер матрицы фиксирован, области эквивалентности для проверки является ли заданная матрица положительно определённой следующие:

1) Элементы матрицы соответствуют тому что матрица положительно определена;

2) Элементы матрицы не соответствуют тому что матрица положительно определена.

Области эквивалентности для обработки строки:

1) По наличию решётки:

   а) Решётка есть в строке;

   б) Решётки нет в строке.

2) Позиция решётки в строке:

   а) Решётка – это первый символ строки;

   б) Решётка – это последний символ строки;

   в) Решётка – средний символ строки.

3) Уникальный случай – решётка единственный символ в строке.

Для задания 3 были определены области эквивалентности входных данных:

1) Наличие строк в тексте:

   а) В текстовом файле есть строки;

   б) В текстовом файле отсутствуют строки.

2) Местоположение минимальной строки:

   а) Минимальная строка – первая

   б) Минимальная строка – последняя

   в) Минимальная строка – где-то в середине текста

В соответствии с областями эквивалентности были сделаны тесты программы. Результаты тестов представлены на рисунках 1 – 10.

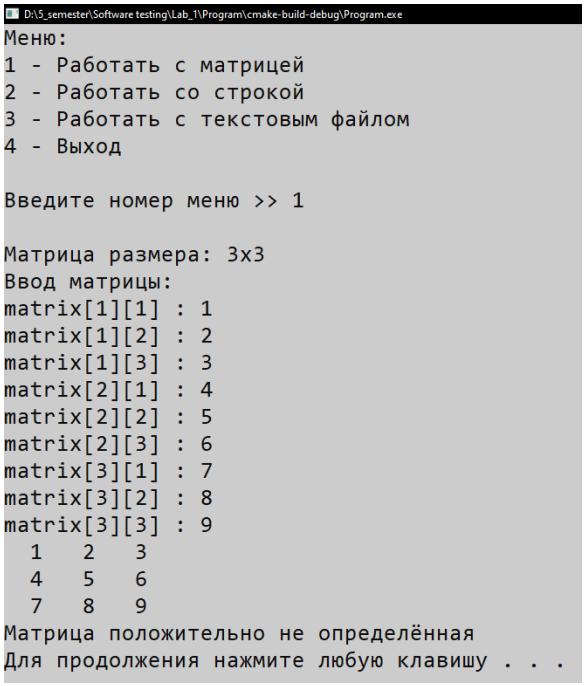

Рисунок 1 – Первый тест при работе с матрицей, она положительно не определена

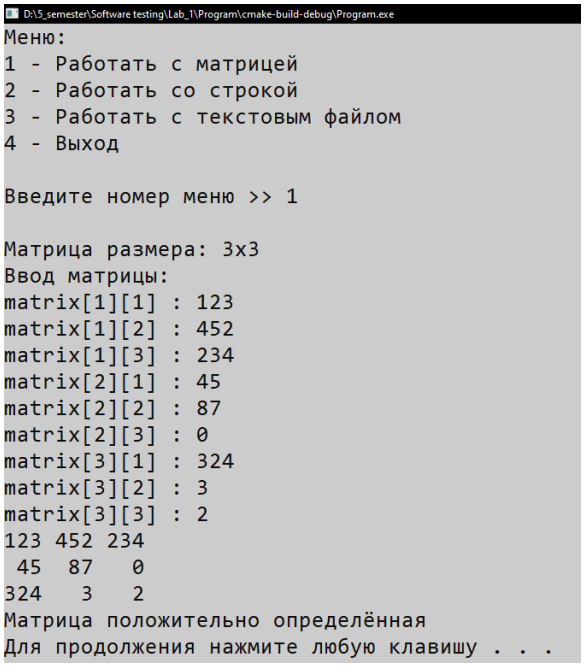

Рисунок 2 – Второй тест при работе с матрицей, она положительно определена

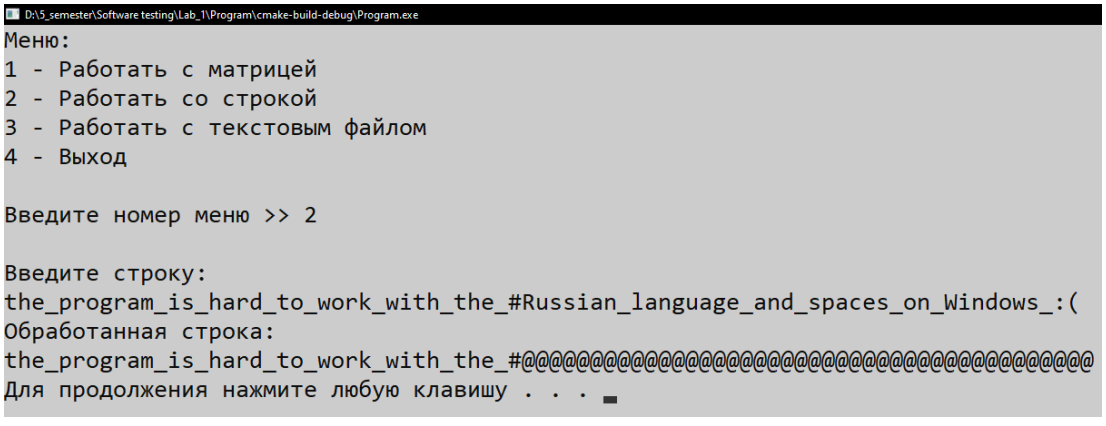

Рисунок 3 – В строке есть ‘#’ и она в середине строки

```
D:\5_semester\Software testing\Lab_1\Program\cmake-build-debug\Program.exe
Меню:
1 - Работать с матрицей
2 - Работать со строкой
3 - Работать с текстовым файлом
4 - Выход

Введите номер меню >> 2

Введите строку:
the_program_is_hard_to_work_with_the_Russian_language_and_spaces_on_Windows_:(#
Обработанная строка:
the_program_is_hard_to_work_with_the_Russian_language_and_spaces_on_Windows_:(#@@
Для продолжения нажмите любую клавишу . . .
```

Рисунок 4 – ‘#’ в конце строки

```
D:\5_semester\Software testing\Lab_1\Program\cmake-build-debug\Program.exe
Меню:
1 - Работать с матрицей
2 - Работать со строкой
3 - Работать с текстовым файлом
4 - Выход

Введите номер меню >> 2

Введите строку:
#the_program_is_hard_to_work_#with_the_Russian_langua#ge_and_spaces_on_Windows_:(
Обработанная строка:
#@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
Для продолжения нажмите любую клавишу . . .
```

Рисунок 5 – ‘#’ в начале строки и ещё несколько на протяжении строки

```
D:\5_semester\Software testing\Lab_1\Program\cmake-build-debug\Program.exe
Меню:
1 - Работать с матрицей
2 - Работать со строкой
3 - Работать с текстовым файлом
4 - Выход

Введите номер меню >> 2

Введите строку:
#
Обработанная строка:
#@dIю
Для продолжения нажмите любую клавишу . . .
```

Рисунок 6 – Решётка единственный символ в строке

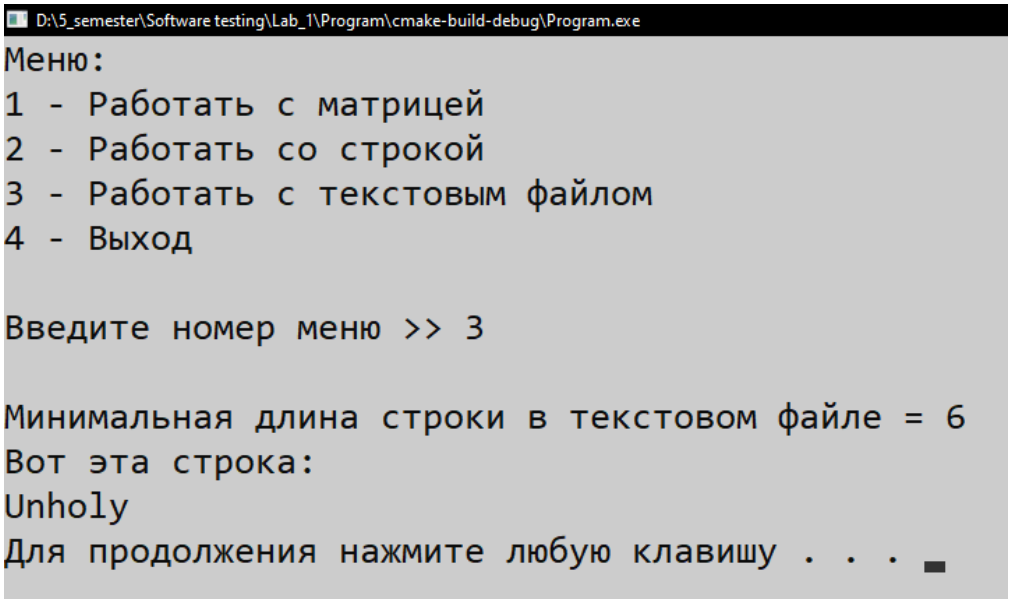

Рисунок 7 – В текстовом файле есть текст и минимальная строка в середине текста

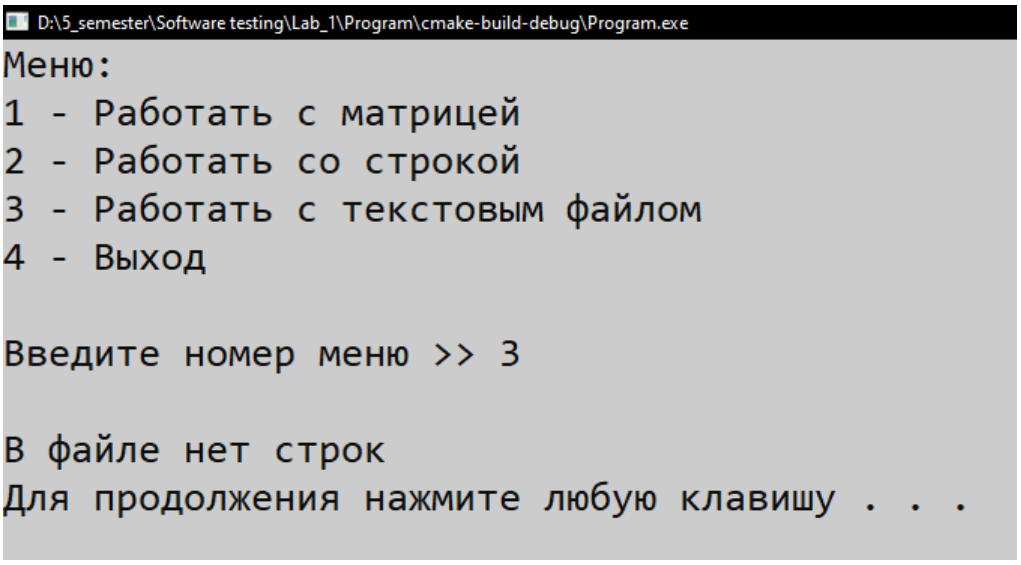

Рисунок 8 – В текстовом файле нет строк

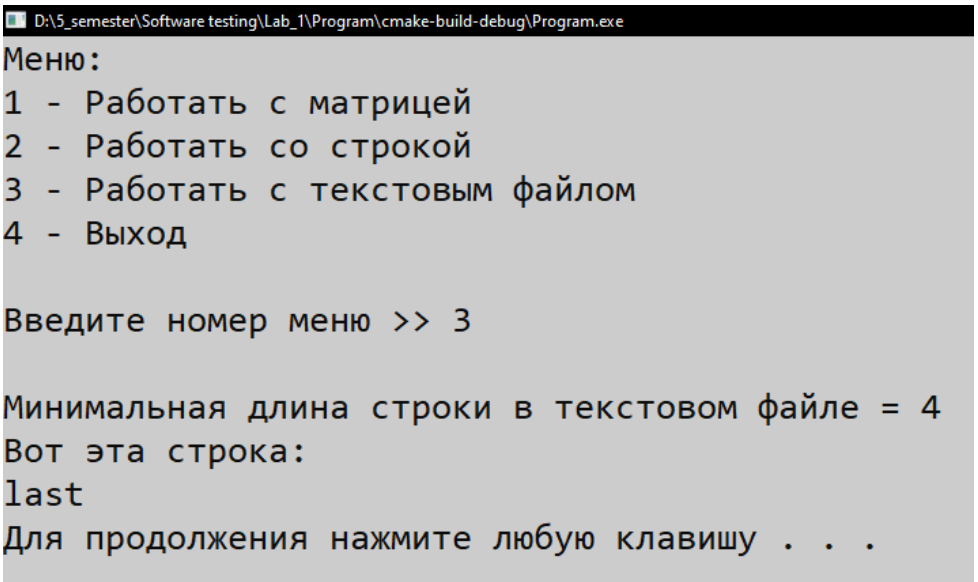

Рисунок 9 – Минимальная строка последняя в текстовом файле

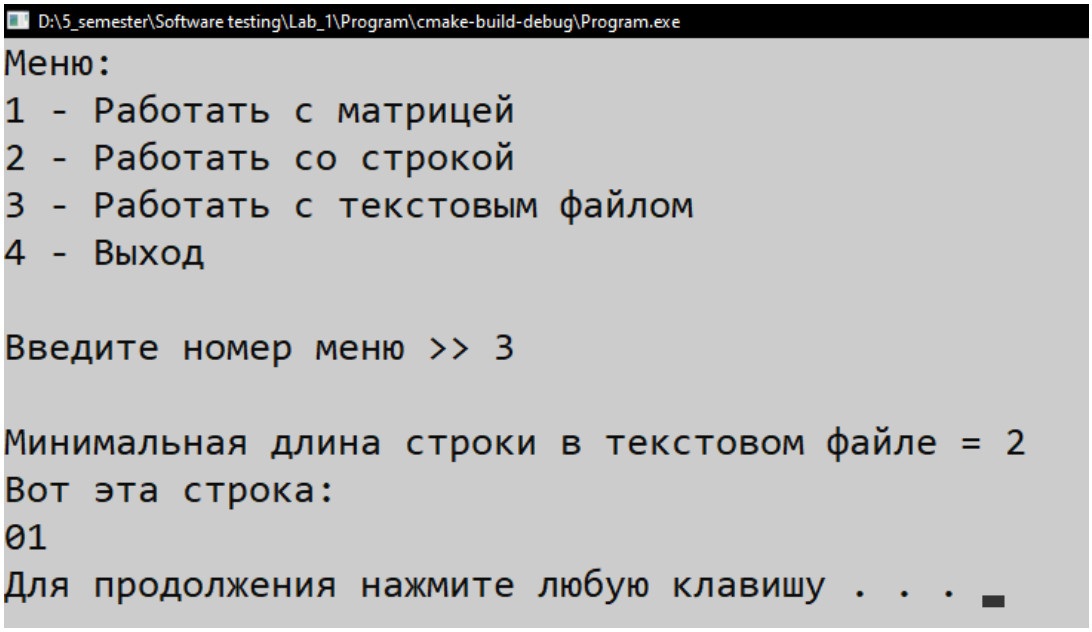

Рисунок 10 – Минимальная строка первая в текстовом файле

Все результаты тестов соответствуют ожидаемым результатам, кроме теста со строкой, где символ решётки находится в конце строки. В этом исключительном случае после символа решётки появляются дополнительные символы в связи с несостыковками в кодировках страниц и в связи с тем, что после конца строки идут такие невидимые символы как конец строки и перевод на новую строку.

Выводы:

В ходе выполнения данной лабораторной работы были исследованы способы анализа областей эквивалентности входных данных для тестирования программного обеспечения. Приобретены практические навыки составления построения тестовых последовательностей. Сделан вывод что данные процедуры крайне полезны и действительно помогают выявить проблемы в программе на граничных значениях и в областях эквивалентности. Был повторен материал, связанный со строками, массивами и файлами. Полученные навыки и опыт помогут при дальнейшей разработке и тестировании более сложных программ.

ПРИЛОЖЕНИЕ А

Листинг программы

```
#include <iostream>
#include <iomanip>
#include <algorithm>
#include <fstream>

int print_menu() {
    int num;
    system("cls");
    std::cout << "Меню:" << std::endl;
    std::cout << "1 - Работать с матрицей" << std::endl;
    std::cout << "2 - Работать со строкой" << std::endl;
    std::cout << "3 - Работать с текстовым файлом" << std::endl;
    std::cout << "4 - Выход" << std::endl << std::endl;
    std::cout << "Введите номер меню >> ";
    std::cin >> num;
    std::cout << std::endl;
    return num;
}

int **entering_a_square_matrix(size_t N) {
    //выделение памяти двумерному массиву
    int **matrix = new int * [N];
    for (size_t i = 0; i < N; i++) {
        matrix[i] = new int [N];
    }

    std::cout << "Ввод матрицы:" << std::endl;
    for (size_t i = 0; i < N; i++) {
        for (size_t j = 0; j < N; j++) {
            std::cout << "matrix[" << i+1 << "][" << j+1 << "] : ";
            std::cin >> matrix[i][j];
        }
    }
    return matrix;
}
```

```
void matrix_output(int **matrix, size_t N) {
    for(size_t i = 0; i < N; i++){
        for(size_t j = 0; j < N; j++)
            std::cout << std::setw(3) << matrix[i][j] << ' ';
        std::cout << std::endl;
    }
}


bool is_the_matrix_positive_definite(int **matrix, size_t N) {
    //положительно определённая матрица: невырожденная матрица B, что A = B^T·B
    //невырожденная матрица - матрица определитель которой != 0
    //B^T - транспонирование матрица - строки меняются на столбцы (матрицу как будто перевернули дном вверх)
    //не могу понять как работает положительно определенная матрица поэтому просто узнаю невырожденная она или нет
    bool isMPV = false;
    size_t determinant = matrix[0][0]*matrix[1][1]*matrix[2][2] + matrix[0][1]*matrix[1][2]*matrix[2][0] + matrix[0][2]*matrix[1][0]*matrix[2][1]
                         - matrix[0][2]*matrix[1][1]*matrix[2][0] - matrix[0][1]*matrix[1][0]*matrix[2][2] - matrix[0][0]*matrix[1][2]*matrix[2][1];
    //если матрица вырожденная то (условно) матрица положительно определённая
    if (determinant != 0) isMPV = true;
    (isMPV == true)? std::cout << "Матрица положительно определённая" << std::endl : std::cout << "Матрица положительно не определённая" << std::endl;
    return isMPV;
}



void matrix_destroyer(int **matrix, size_t N) {
    for (size_t i = 0; i < N; i++) {
        delete [] matrix[i];
    }
    delete [] matrix;
}


void work_with_matrix() {
    size_t N = 3;
    int **matrix;
    std::cout << "Матрица размера: " << N << "x" << N << std::endl;
    //Ввод матрицы
    matrix = entering_a_square_matrix(N);
    //Вывод матрицы
```

```
    matrix_output(matrix, N);
    //определение того, является ли матрица положительно определённой
    bool isMPV = is_the_matrix_positive_definite(matrix, N);
    //очистка памяти
    matrix_destroyer(matrix, N);
    system("pause");
}

void work_with_string() {
    getchar();
    const int length = 100;
    char str[length] = "";
    std::cout << "Введите строку:" << std::endl;
    std::cin >> str;
    //обработка строки
    int str_length = 0;
    for (int i = 0; str[i] != '\0'; i++) {
        str_length++;
    }
    bool flag = false;
    for(int i = 0; i < str_length; i++) {
        if (!flag) {
            if(str[i] == '#') flag = true;
        }
        else {
            str[i] = '@';
        }
    }
    std::cout << "Обработанная строка:" << std::endl << str << std::endl;
    system("pause");
}

//поиск минимальной длины строки текстового файла и печать этой строки на экран
void work_with_text_file() {
    std::ifstream file("D:\\5_semester\\Software testing\\Lab_1\\Program\\text.txt");
    std::string temp_str;
    std::getline(file, temp_str);
    std::string min_str = temp_str;
    int min_str_length = temp_str.length();
    // пока не достигнут конец файла класть очередную строку в переменную str
```

```
    while(getline(file, temp_str)) {
        if (temp_str.length() < min_str_length) {
            min_str = temp_str;
            min_str_length = temp_str.length();
        }
    }
    file.close();
    std::cout << "Минимальная длина строки в текстовом файле = " << min_str_length
              << std::endl << "Вот эта строка:" << std::endl << min_str << std::endl;
    system("pause");
}

int main() {
    setlocale(LC_ALL, "Rus");
    system("color 70");
    size_t num;
    do {
        switch (num = print_menu()) {
            case 1:
                work_with_matrix();
                break;
            case 2:
                work_with_string();
                break;
            case 3:
                work_with_text_file();
                break;
            case 4:
                system("pause");
                break;
            default:
                std::cout << "Введите корректное значение" << std::endl;
                system("pause");
        }
    } while (num != 4);
    return 0;
}
```