**Лабораторная работа №2**

**Исследование способов анализа областей эквивалентности и построения тестовых последовательностей**

1. **Цель работы**

Исследовать основные подходы к структурному тестированию программного обеспечения. Приобрести практические навыки построения графа потоков управления и определения независимых ветвей программы.

1. **Постановка задачи**

Вариант — 22

Варианты заданий соответствуют заданиям по лабораторной работе №1. По варианту задаются требования к программам. Для каждой из них необходимо:

1) Написать программу, выполняющую заданные действия.

2) Построить граф потоков управления.

3) Вычислить цикломатическое число для построенного графа потоков

управления.

4) Определить независимые ветви программы.

1. **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 –

Рисунок 2 –

Рисунок 3 –

Рисунок 4 –

Рисунок 5 –

Рисунок 6 –

Рисунок 7 –

Рисунок 8 –

Рисунок 9 –

Рисунок 10 –

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 A**

**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;

}