**1 ЛАБОРАТОРНАЯ РАБОТА №1**

**«Исследование способов анализа областей эквивалентности и**

**построения тестовых последовательностей»**

**1.1 Цель работы**

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

**1.2 Вариант задания**

Задача 1. Дана целочисленная прямоугольная матрица. Определить номер строки, в которой находится самая длинная серия нулевых элементов.

Задача 2. Дана строка. Преобразовать строку, заменив в ней все вопросительные знаки амперсантами.

Задача 3. Программа, которая считывает текст из файла и выводит на экран только строки, содержащие трехзначные числа.

**1.3 Ход выполнения работы**

1.3.1 В начале выполнения лабораторной работы была написана программа, которая определяет номер строки, в которой находится самая длинная серия нулевых элементов. Код программы представлен в листинге 1.1.

Листинг 1.1 – Текст первой программы

#include <iostream>

using namespace std;

int cntStr(int \*\*a, int n, int m);

int main() {

int n, m;

int \*\*a;

cout << "Enter N\*M: ";

cin >> n >> m;

a = new int \*[n];

for (int i = 0; i < n; i++)

a[i] = new int[m];

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

cout << "[" << i << "][" << j << "] = ";

cin >> a[i][j];

}

}

cout << cntStr(a, n, m);

for (int i = 0; i < n; i++)

delete[] a[i];

delete[] a;

return 0;

}

int cntStr(int \*\*a, int n, int m) {

int max = 0, maxCur = 0, maxTmp = 0, maxStr = -1;

for (int i = 0; i < n; i++) {

maxTmp = 0;

for (int j = 0; j < m; j++) {

if (a[i][j] == 0)

maxTmp++;

else {

if (maxTmp > maxCur) {

maxCur = maxTmp;

}

maxTmp = 0;

}

}

if (maxTmp > maxCur) {

maxCur = maxTmp;

}

if (maxCur > max) {

max = maxCur;

maxStr = i;

}

maxCur = 0;

}

if (maxStr == -1) return 0;

return maxStr + 1;

}

Для этой программы были определены области эквивалентности, которые показаны в таблице 1.1.

Таблица 1.1 – Области эквивалентности для первой программы

|  |  |
| --- | --- |
| Размер матрицы | Последовательность нулей |
| 1×1 | Нет |
| 1×1 | Есть |
| N×M | Нет |
| N×M | В начале |
| N×M | В середине |
| N×M | В конце |
| N×M | Несколько в строке |

Далее для этой программы по выявленным областям эквивалентности были разработаны примеры тестовых последовательностей, что показано в таблице 1.2.

Таблица 1.2 – Тестовые последовательности для первой программы

|  |
| --- |
| Входная матрица |
| 3 |
| 0 |
| [1, 2, 3]  [4, 5, 6]  [7, 8, 9] |
| [1, 0, 0, 0]  [2, 3, 4, 5] |
| [1, 2, 3, 4]  [0, 0, 0, 0]  [6, 7, 8, 9]  [1, 2, 3, 4] |
| [1, 2, 3]  [4, 5, 6]  [7, 0, 9] |
| [1, 0, 0, 1, 0, 0]  [2, 3, 4, 5, 6, 7]  [0, 0, 1, 0, 0, 0] |

На рисунках 1.1 – 1.3 показано правильное выполнение нескольких тестовых примеров из разработанной последовательности.

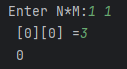


Рисунок 1.1 – Результат выполнения первого тестового примера

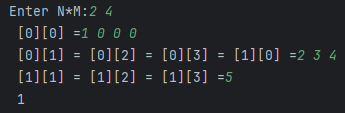


Рисунок 1.2 – Результат выполнения третьего тестового примера

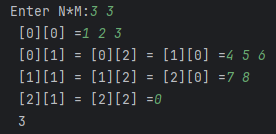


Рисунок 1.3 – Результат выполнения шестого тестового примера

1.3.2 Далее была написана программа, которая преобразует строку, заменив в ней все вопросительные знаки амперсантами. Код программы представлен в листинге 1.2.

Листинг 1.2 – Текст второй программы

#include <iostream>

#include <string>

using namespace std;

string replace(string str);

int main(){

string str;

cout<<"Enter string: ";

cin>>str;

cout<<replace(str);

return 0;

}

string replace(string str){

for(int i = 0; i < str.length(); i++){

if(str[i] == '?')

str[i]='&';

}

return str;

}

Для этой программы были определены области эквивалентности, которые показаны в таблице 1.3.

Таблица 1.3 – Области эквивалентности для второй программы

|  |  |
| --- | --- |
| Размер строки | Амперсант |
| 1 | Нет |
| 1 | Есть |
| >1 | Нет |
| >1 | В начале |
| >1 | В середине |
| >1 | В конце |
| >1 | Несколько в строке |

Далее для этой программы по выявленным областям эквивалентности были разработаны примеры тестовых последовательностей, что показано в таблице 1.4.

Таблица 1.4 – Тестовые последовательности для второй программы

|  |  |
| --- | --- |
| Входная строка | Выходная строка |
| a | a |
| ? | & |
| abcde | abcde |
| ?ab23ef | &ab23ef |
| 1bc?defg | 1bc&defg |
| 123bcdefg? | 123bcdefg& |
| ??ab???c1d??? | &&ab&&&c1d&&& |

На рисунках 1.4 – 1.6 показано правильное выполнение нескольких тестовых примеров из разработанной последовательности.



Рисунок 1.4 – Результат выполнения второго тестового примера



Рисунок 1.5 – Результат выполнения пятого тестового примера



Рисунок 1.6 – Результат выполнения седьмого тестового примера

1.3.3 Последней была написана программа, которая считывает текст из файла и выводит на экран только строки, содержащие трехзначные числа. Код программы представлен в листинге 1.3.

Листинг 1.3 – Текст третьей программы

/\* Задача 3. Программа, которая считывает текст из файла и выводит на

экран только строки, содержащие четырехзначные числа.\*/

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main(){

string str, line;

cout<<"Enter file name: ";

cin>>str;

string filename = str + ".txt";

ifstream fin(filename);

if(!fin){

cout<<"Error opening file";

return 1;

}

bool lineFlag = false;

while(getline(fin, line)){

for(int i = 0; i<line.length()-2; i++){

if (isdigit(line[i]) && isdigit(line[i + 1]) && isdigit(line[i + 2])) {

if ((i == 0 || !isdigit(line[i - 1])) && (!isdigit(line[i + 3]) || line.length() == i + 3)){

cout << line << endl;

lineFlag = true;

break;

}

}

}

}

if(!lineFlag) cout<<"No suitable lines";

return 0;

}

Для этой программы были определены области эквивалентности, которые показаны в таблице 1.5.

Таблица 1.5 – Области эквивалентности для второй программы

|  |  |
| --- | --- |
| Количество строк | Трехзначное число |
| 0 | - |
| 1 | Есть |
| 1 | Нет |
| >1 | В начале |
| >1 | В середине |
| >1 | В конце |
| >1 | Несколько в строке |

Далее для этой программы по выявленным областям эквивалентности были разработаны примеры тестовых последовательностей, что показано в таблице 1.6.

Таблица 1.6 – Тестовые последовательности для третьей программы

|  |  |
| --- | --- |
| Входной файл | Выходные строки |
| - | - |
| abc123de12 | abc123de12 |
| a45bcde12 | - |
| 123abcdefghijk  ab1%cdefghij78k  abc!3defgh+ijk | 123abcdefghijk |
| abcdefghijk  ab1%cdefghij78k  abc!3de123fghijk  abcdef-gijk | 1bc&defg |
| 12abcdefghijk  ab1%cde23fghij78k  abc!d3ef-gh55ijk  abc!d3ef-gh55ij123k | abc!d3ef-gh55ij123k |
| 123abcdef123ghijk  ab1%cde23fghij78k  ab123c!d3ef-gh55ijk456  abc!d3ef-gh55ijk | 123abcdef123ghijk  ab123c!d3ef-gh55ijk456 |

На рисунках 1.7 – 1.9 показано правильное выполнение нескольких тестовых примеров из разработанной последовательности.



Рисунок 1.7 – Результат выполнения третьего тестового примера



Рисунок 1.8 – Результат выполнения шестого тестового примера

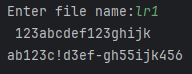


Рисунок 1.9 – Результат выполнения седьмого тестового примера

**Выводы**

В ходе лабораторной работы были исследованы способы анализа областей эквивалентности входных данных для тестирования программного обеспечения. Также были приобретены практические навыки составления построения тестовых последовательностей. В конце выполнения лабораторной работы был написан отчет.