Тоббо Кирилл Вадимович

Группа: 3530904/10005

Лабораторная работа №5. Индивидуальные задания.

**Задание 1.10**

**Общая постановка задачи.**

Напишите функцию для работы со встроенным массивом, которая будет выводить Количество столбцов, содержащих повторяющиеся элементы. Функция, должна иметь три параметра – адрес матрицы, количество строк и количество столбцов и выполнять действия в соответствии с указанным вариантом. Элементы матрицы – целые числа.

**Детальные требования, тест план.**

1. Создаем встроенный массив.
2. Пишем функцию, которая проходится по массиву и проверяет есть ли в строках хотя бы один нулевой элемент. Функция возвращает Количество столбцов, содержащих повторяющиеся элементы.
3. В main проверяем открыт ли файл вывода и выводим туда значение количества строк, имеющих хотя бы один нулевой элемент.

**Код:**

#include <iostream>

#include <fstream>

const char\* ERROR\_INPUT\_FILE\_NOT\_FOUND = "Error: input.txt file not found.";

const char\* ERROR\_OUTPUT\_FILE\_NOT\_FOUND = "Error: output.txt file not found.";

const char\* ERROR\_INVALID\_ARRAY\_INPUT = "Error: invalid array input.txt.";

const int sizeX = 3;

const int sizeY = 3;

void setArray(int(&array)[sizeX][sizeY], std::ifstream& input);

int getDuplNumber(int(&array)[sizeX][sizeY]);

bool isColumnNumsUniq(int(&array)[sizeX][sizeY], int& columnNumber);

int main()

{

try {

int array[sizeX][sizeY];

std::ifstream input(R"(C:\Users\orcva\source\repos\LAB1\LAB5\input.txt)");

if (!input.is\_open())

{

throw ERROR\_INPUT\_FILE\_NOT\_FOUND;

}

setArray(array, input);

input.close();

int answer = getDuplNumber(array);

std::ofstream output(R"(C:\Users\orcva\source\repos\LAB1\LAB5\output.txt)");

if (!output.is\_open())

{

throw ERROR\_OUTPUT\_FILE\_NOT\_FOUND;

}

output << answer;

output.close();

}

catch (const char\* error)

{

system("cls");

std::cerr << std::endl << error << std::endl;

exit(1);

}

}

void setArray(int(&array)[sizeX][sizeY], std::ifstream& input)

{

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

{

for (int j = 0; j < sizeY; ++j)

{

if (!input.eof()) {

input >> array[i][j];

/\*if (input.fail() || (input.peek() != 10 && input.peek() != 32))

{

throw ERROR\_INVALID\_ARRAY\_INPUT;

}\*/

}

else

{

throw ERROR\_INVALID\_ARRAY\_INPUT;

}

/\*if (!input.eof()) {

throw ERROR\_INVALID\_ARRAY\_INPUT;

}\*/

}

}

}

int getDuplNumber(int(&array)[sizeX][sizeY])

{

int ans = 0;

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

{

if (!isColumnNumsUniq(array, i)) ans++;

}

return ans;

}

bool isColumnNumsUniq(int(&array)[sizeX][sizeY], int& columnNumber)

{

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

{

for (int j = 0; j < sizeY; ++j)

{

if (j != i && array[j][columnNumber] == array[i][columnNumber])

{

return false;

}

}

}

return true;

}

**Задание 2.10**

**Общая постановка задачи.**

Напишите функцию для работы с двумерным массивом, которая будет выводить номер столбца, в котором находится самая длинная серия подряд идущих равных элементов. Функция, должна иметь два параметра – адрес матрицы (целые числа), размерность матрицы и выполнять действия в соответствии с указанным вариантом. Элементы матрицы – целые числа

**Детальные требования, тест план.**

1. Массив по условию двумерный. Проверяем открыт ли файл с данными для массива.
2. Пишем функцию, создающую массив.
3. Создаем функцию для вывода данных из файла в массив.
4. Создаем функцию, которая ищет номер столбца, в котором находится самая длинная серия подряд идущих равных элементов.
5. Полученное значение выводим в файл в main.

**Код:**

#include <iostream>

#include <fstream>

const char\* ERROR\_INPUT\_FILE\_NOT\_FOUND = "Error: input.txt file not found.";

const char\* ERROR\_OUTPUT\_FILE\_NOT\_FOUND = "Error: output.txt file not found.";

const char\* ERROR\_INVALID\_ARRAY\_INPUT = "Error: invalid array input.txt.";

const char\* ERROR\_X\_NOT\_FOUND = "Error: array x size not found.";

const char\* ERROR\_Y\_NOT\_FOUND = "Error: array y size not found.";

const char\* ERROR\_INVALID\_X\_INPUT = "Error: x must be integer greater then 0.";

const char\* ERROR\_INVALID\_Y\_INPUT = "Error: y must be integer greater then 0.";

void setArray(int\*\*& array, int sizeX, int sizeY, std::ifstream& input);

int getDuplNumber(int\*\*& array, int sizeX, int sizeY);

bool isColumnNumsUniq(int\*\*& array, int sizeY, int& columnNumber);

int main()

{

try

{

std::ifstream input(R"(C:\Users\orcva\source\repos\LAB1\LAB5-2\input.txt)");

if (!input.is\_open())

{

throw ERROR\_INPUT\_FILE\_NOT\_FOUND;

}

int sizeX = -5;

int sizeY = -5;

int\*\* array;

if (!input.eof())

{

input >> sizeX;

if (input.fail() || sizeX < 1 || (input.peek() != 10 && input.peek() != 32))

{

throw ERROR\_INVALID\_X\_INPUT;

}

}

else

{

throw ERROR\_X\_NOT\_FOUND;

}

if (!input.eof())

{

input >> sizeY;

if (input.fail() || sizeY < 1 || (input.peek() != 10 && input.peek() != 32))

{

throw ERROR\_INVALID\_Y\_INPUT;

}

}

else

{

throw ERROR\_Y\_NOT\_FOUND;

}

setArray(array, sizeX, sizeY, input);

input.close();

int answer = getColWithMaxRepeats(array, sizeX, sizeY);

std::ofstream output(R"(C:\Users\orcva\source\repos\LAB1\LAB5-2\output.txt)");

if (!output.is\_open()){

throw ERROR\_OUTPUT\_FILE\_NOT\_FOUND;

}

output << answer;

output.close();

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

{

delete[] array[i];

}

delete[] array;

}

catch (const char\* error)

{

system("cls");

std::cerr << std::endl << error << std::endl;

exit(1);

}

}

void setArray(int\*\*& array, int sizeX, int sizeY, std::ifstream& input)

{

array = new int\* [sizeX];

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

{

int\* line = new int[sizeY];

for (int j = 0; j < sizeY; ++j)

{

if (!input.eof()) {

input >> line[j];

}

else

{

throw ERROR\_INVALID\_ARRAY\_INPUT;

}

}

array[i] = line;

}

}

int getColumnNumsRepeats(int\*\*& array, int sizeY, int& columnNumber)

{

int maxReps = 0;

int curReps = 0;

for (int i = 1; i < sizeY; ++i)

{

if (array[i-1][columnNumber] == array[i][columnNumber]) {

curReps++;

if (curReps > maxReps) {

maxReps = curReps;

}

}

else {

curReps = 0;

}

}

return maxReps;

}

int getColWithMaxRepeats(int\*\*& array, int sizeX, int sizeY)

{

int ans = -1;

int prevReps = 0;

for (int i = 0; i < 3; ++i)

{

int reps = getColumnNumsRepeats(array, sizeY, i);

if (reps > prevReps) {

ans = i;

prevReps = reps;

}

}

return ans;

}