МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ИНСТИТУТ НЕПРЕРЫВНОГО И ДИСТАНЦИОННОГО ОБРАЗОВАНИЯ

|  |
| --- |
| КАФЕДРА компьютерных технологий и программной инженерии |

ОЦЕНКА

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ |
| «ОБРАБОТКА ЧИСЛОВЫХ МАТРИЦ» |
| по дисциплине: Основы программирования |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | Z9431 |  |  |  | Андреев Д.И. |
|  | номер группы |  | подпись, дата |  | инициалы, фамилия |
| Студенческий билет № | 2019/3781 | |  |  |  |

Санкт-Петербург 2020

Оглавление

[1. Цель работы 3](#_Toc55577038)

[2. Задание 3](#_Toc55577039)

[3. Описание созданных функций 3](#_Toc55577040)

[4. Листинг программы 3](#_Toc55577041)

[5. Пример выполнения программы 3](#_Toc55577042)

[6. Анализ результатов и выводы 3](#_Toc55577043)

1. Цель работы

Целью работы является изучение структуры данных двумерный массив.

1. Задание

Вариант 1

Дана целочисленная прямоугольная матрица. Определить:

1. количество строк, не содержащих ни одного нулевого элемента;
2. максимальное из чисел, встречающихся в заданном матрице более одного раза.
3. Описание созданных функций

Для решения поставленной в лабораторной работе задачи потребуются следующие функции:

Имя: maxNonUnique

Назначение: вычисляет максимальное из чисел, встречающихся матрице более одного раза и выводит результат в терминал

Входные данные:

* mat – указатель на двумерный массив элементов (матрицу);
* size – размерность матрицы.

Выходные данные: отсутствуют.

Побочный эффект: отсутствует.

Тестовые данные:

Матрица:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 2 | 5 | 6 |
| 15 | 5 | 1 | 0 |
| 6 | 15 | 2 | 3 |
| 21 | 8 | 1 | 6 |

Результат выполнения функции:

"Максимальное из чисел, встречающихся матрице более одного раза: 15"

Прототип: void maxNonUnique(int \*\*mat, const unsigned short size);

Алгоритм:

Псевдокод:

Объявить переменную max для хранения текущего максимального повторяющегося элемента

Для каждой строки

Для каждого элемента в строке

Объявить переменную для хранения текущего искомого числа (searchNum) и присвоить ей значение данного элемента

Объявить логическую переменную для завершения поиска exitFlag и присвоить ей false

Для каждой строки или пока exitFlag == false

Для каждого элемента в строке

Если данный элемент равен значению переменной searchNum , searchNum > max и элемент не сравнивается сам с собой

Присвоить переменной max значение переменной searchNum

Установить флаг exitFlag = true

Выйти из цикла

Блок-схема:



Имя: notNullStrings

Назначение: Количество строк, не содержащих ни одного нулевого элемента и выводит результат в терминал

Входные данные:

* mat – указатель на двумерный массив элементов (матрицу);
* size – размерность матрицы.

Выходные данные: отсутствуют.

Побочный эффект: отсутствует.

Тестовые данные:

Матрица:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 2 | 5 | 10 |
| 1 | 1 | 3 | 2 |
| 0 | 5 | 0 | 0 |
| 1 | 2 | 8 | 9 |

Результат выполнения функции:

"Количество строк, не содержащих ни одного нулевого элемента: 2"

Прототип: void notNullStrings (int \*\*mat, const unsigned short size);

Алгоритм:

Псевдокод:

Объявить переменную count для подсчёта строк

Для каждой строки

Для каждого элемента в строке

Если данный элемент равен 0

Увеличить значение переменной count на 1

Завершить цикл

Блок-схема:



1. Листинг программы

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <iomanip>

// проверка на утечки памяти

#define \_CRTDBG\_MAP\_ALLOC

#include <cstdlib>

#include <crtdbg.h>

#define DBG\_NEW new(\_NORMAL\_BLOCK, \_\_FILE\_\_, \_\_LINE\_\_)

#define new DBG\_NEW

using namespace std;

//определения функций

void showHint();

short getCommand();

void showMatrix(int \*\* mat, const unsigned short size);

void randomGenerate(int\*\* mat, const unsigned short size);

void maxNotUnique(int\*\* mat, const unsigned short size);

void notNullStrings(int\*\* mat, const unsigned short size);

void fillFromKeyboard(int\*\* mat, const unsigned short size);

int main()

{

setlocale(0, "Russian");

cout << "Лабораторная работа №6.\n";

cout << Обработка числовых матриц.\nВариант " << (1 - 1) % 20 + 1 << endl;

short command = -1;

int\*\* matrix = nullptr;

unsigned short size = 0;

//Для тестов, чтобы можно было тестировать на случайном размере

cout << "Введите размерность матрицы: ";

cin >> size;

while (!cin || !(size > 0)) {

cout << "Введено неверное значение. Повторите ввод: ";

std::cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cin >> size;

}

matrix = new int\* [size];

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

matrix[i] = new int[size];

}

do {

command = getCommand();

switch (command)

{

case 0:

fillFromKeyboard(matrix, size);

break;

case 1:

randomGenerate(matrix, size);

break;

case 2:

showMatrix(matrix, size);

break;

case 3:

notNullStrings(matrix, size);

break;

case 4:

maxNotUnique(matrix, size);

break;

case 5:

showHint();

break;

case 6:

break;

default:

cout << "Неизвестная команда\n";

break;

}

} while (command != 6);

for (unsigned short i = 0; i < size; ++i)

delete[] matrix[i];

delete[] matrix;

// Для обнаружения утечек памяти

\_CrtSetReportMode(\_CRT\_WARN, \_CRTDBG\_MODE\_FILE);

\_CrtSetReportFile(\_CRT\_WARN, \_CRTDBG\_FILE\_STDOUT);

\_CrtSetReportMode(\_CRT\_ERROR, \_CRTDBG\_MODE\_FILE);

\_CrtSetReportFile(\_CRT\_ERROR, \_CRTDBG\_FILE\_STDOUT);

\_CrtSetReportMode(\_CRT\_ASSERT, \_CRTDBG\_MODE\_FILE);

\_CrtSetReportFile(\_CRT\_ASSERT, \_CRTDBG\_FILE\_STDOUT);

\_CrtDumpMemoryLeaks();

return 0;

}

short getCommand()

{

short com;

cout << "Введите команду для продолжения (5 - показать все команды): ";

cin >> com;

while (!cin) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Введите корректное значение (6 для выхода): ";

cin >> com;

}

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

return com;

}

void showMatrix(int \*\* mat, const unsigned short size)

{

for (unsigned short i = 0; i < size; ++i) {

for (unsigned short j = 0; j < size; ++j) {

cout << setw(2) << mat[i][j] << ' ';

}

cout << '\n';

}

}

void randomGenerate(int\*\* mat, const unsigned short size)

{

srand(clock());

for (unsigned short i = 0; i < size; ++i) {

for (unsigned short j = 0; j < size; ++j) {

mat[i][j] = rand() % 100;

}

}

}

void maxNonUnique(int\*\* mat, const unsigned short size)

{

int max = 0;

for (unsigned short i = 0; i < size; ++i) {

for (unsigned short j = 0; j < size; ++j) {

int searchNum = mat[i][j];

bool exitFlag = false;

for (unsigned short k = 0; k < size && !exitFlag; ++k) {

for (unsigned short n = 0; n < size; ++n) {

int el = mat[k][n];

if (searchNum == mat[k][n] && !(searchNum < max) && n != j) {

max = searchNum;

exitFlag = true;

break;

}

}

}

}

}

cout << "Максимальное из чисел, встречающихся матрице более одного раза: " << max << '\n';

}

void notNullStrings(int\*\* mat, const unsigned short size)

{

unsigned int count = 0;

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

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

if (mat[i][j] == 0) {

++count;

break;

}

}

}

cout << "Количество строк, не содержащих ни одного нулевого элемента: " << count << '\n';

}

void fillFromKeyboard(int\*\* mat, const unsigned short size)

{

int elem = 0;

cout << "Введите значение элементов для вставки:\n";

for (int row = 0; row < size; ++row)

{

for (int col = 0; col < size; ++col) {

cout << "Элемент [" << row << "][" << col << "]: ";

cin >> elem;

while (!cin) {

std::cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Введено недопустимое значение. Повторите ввод.\n";

cout << "Элемент [" << row << "][" << col << "]: ";

cin >> elem;

}

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

mat[row][col] = elem;

}

}

}

void showHint()

{

cout << "0 - Заполнить матрицу с клавиатуры\n";

cout << "1 - Заполнить матрицу случайными числами\n";

cout << "2 - Вывести матрицу на экран\n";

cout << "3 - Показать количество строк, не содержащих ни одного нулевого элемента\n";

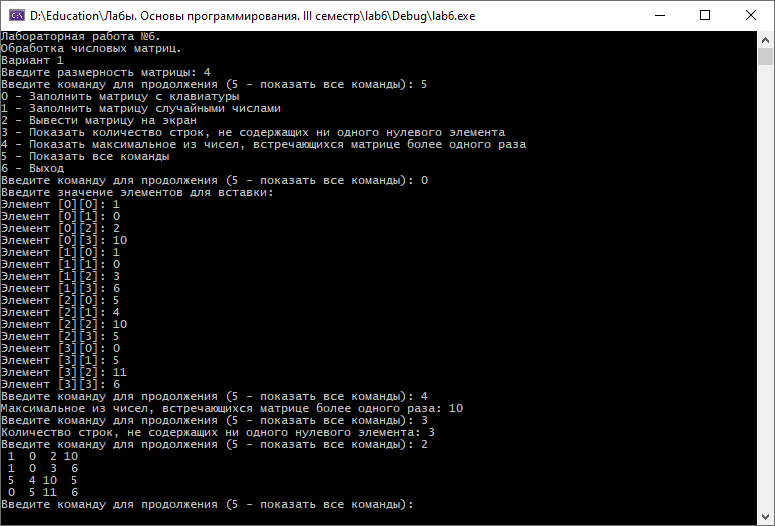
cout << "4 - Показать максимальное из чисел, встречающихся матрице более одного раза\n";

cout << "5 - Показать все команды\n";

cout << "6 - Выход\n";

}

1. Пример выполнения программы



1. Анализ результатов и выводы

К достоинствам программы можно отнести:

Программа работает без ошибок и верно выполняет поставленную задачу

Простота реализации и легко воспринимаемый код программы, а также реализация заданий в виде функций дает возможность повторного использования кода в других проектах

За счет динамического выделения памяти есть возможность выполнения поставленной задачи для матриц с различной размерностью без изменения кода программы

К недостаткам можно отнести:

Линейный поиск, реализованный в данной программе не дает хороших результатов при большой размерности матриц