**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №1**

**по дисциплине «Программирование»**

Тема: **«Арифметика указателей»**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 0324 |  | Костюхина Э.В. |
| Преподаватель |  | Глущенко А.Г |

Санкт-Петербург

2020

**Цель работы.**

Знакомство с указателями. Научиться применять арифметику указателий при работе с матрицами.

**Основные теоретические положения.**

Указатели и ссылки являются одними из самых важных и достаточно сложных для понимания и использования средств языка программирования. Они ориентированы на прямую работу с памятью компьютера. С помощью этих средств реализуется работа с динамической памятью и динамическими объектами, возвращение из функций измененных данных и многое другое.

Указатели – это тоже обычные переменные, но они служат для хранения адресов памяти.

Указатели поддерживают ряд операций: присваивание, получение адреса указателя, получение значения по указателю, некоторые арифметические операции и операции сравнения. Указателю можно присвоить либо адрес объекта того же типа, либо значение другого указателя. Когда указателю присваивается другой указатель, то фактически первый указатель начинает также указывать на тот же адрес, на который указывает второй указатель. Указатель хранит адрес переменной, и по этому адресу мы можем получить значение этой переменной. Но кроме того, указатель, как и любая переменная, сам имеет адрес, по которому он располагается в памяти.

К указателям можно применять некоторые арифметические операции. К таким операциям относятся: +, -, ++, --. Результаты выполнения этих операций по отношению к указателям существенно отличаются от результатов соответствующих арифметических операций, выполняющихся с обычными числовыми данными. Добавлять к указателям или вычитать из указателей можно только целые значения.

Основная и наиболее часто встречающаяся ошибка при работе с указателями связана с использованием неинициализированных указателей. Вторая группа ошибок может быть связана с некорректным использованием арифметики указателей. При некорректном выполнении наращивания или уменьшения указателей с помощью операций + или - можно выйти за пределы предполагаемого объекта (например, массива) и в результате получить неверные данные или модифицировать не те значения.

**Постановка задачи.**

1)    Используя арифметику указателей, заполняет квадратичную целочисленную матрицу порядка N (6,8,10) случайными числами от 1 до N\*N согласно схемам, приведенным на рисунках. Пользователь должен видеть процесс заполнения квадратичной матрицы.



2)    Получает новую матрицу, из матрицы п. 1, переставляя ее блоки в соответствии со схемами:



3)    Используя арифметику указателей, сортирует элементы любой сортировкой.

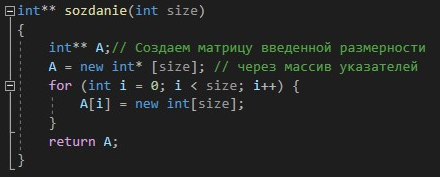
4)    Уменьшает, увеличивает, умножает или делит все элементы матрицы на введенное пользователем число.

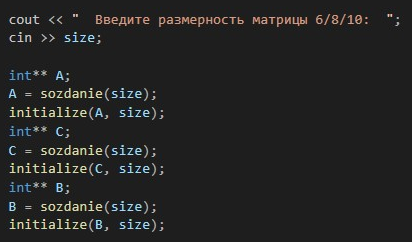
**Выполнение работы.**

Для выполнения поставленной задачи была написана программа на языке C++. Итоговый код программы представлен ниже.

В программе я использовала динамические двумерные массивы, чтобы в памяти выделялось место именно под ту размерность матрицы, которую вводит пользователь. Так же программа разбита на функции для упрощения.

Сначала я создаю три динамических массива, с которыми в последствии буду работать (с помощью функции sozdanie(int size), аргументом является размер матрицы, который вводит пользователь).





Далее для выполнения первого пункта я получаю положение каретки и вызываю функцию spiral(int\*\* a, int size, int num, int strStart, int strEnd, int colStart, int colEnd, int x, int y). Аргументами являются: динамический двумерный массив, его размер, четыре счетчика, которые нам нужны для выполнения функции и координаты, с которых будет начинаться рисунок. Функция состоит из четырех циклов, которые по очереди заполняют двумерный массив по определенной схеме, а пользователь может увидеть процесс заполнения, благодаря перемещению каретки по заданным координатам. Так как функция является рекурсивной, она подходит под любые четные размерности матрицы.

Чтобы заполнить матрицу по схеме б), была создана другая рекурсивная функция: snake(int\*\* a, int size, int str, int col, int x, int y). Аргументами так же являются динамический двумерный массив, размер массива, два счетчика и координаты. Функция состоит из двух циклов, которые по очереди заполняют двумерный массив.

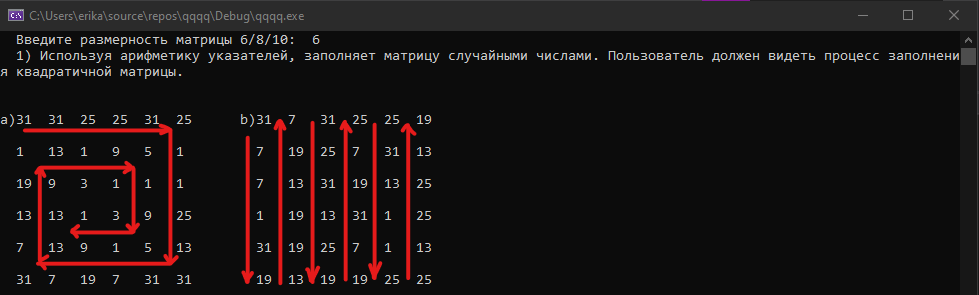
Для выполнения второго пункта используется функция zadanie\_2(int\*\* a, int\*\* C, int size, int X, int Y). Функция переставляет блоки матрицы в соответствии с указанными схемами.

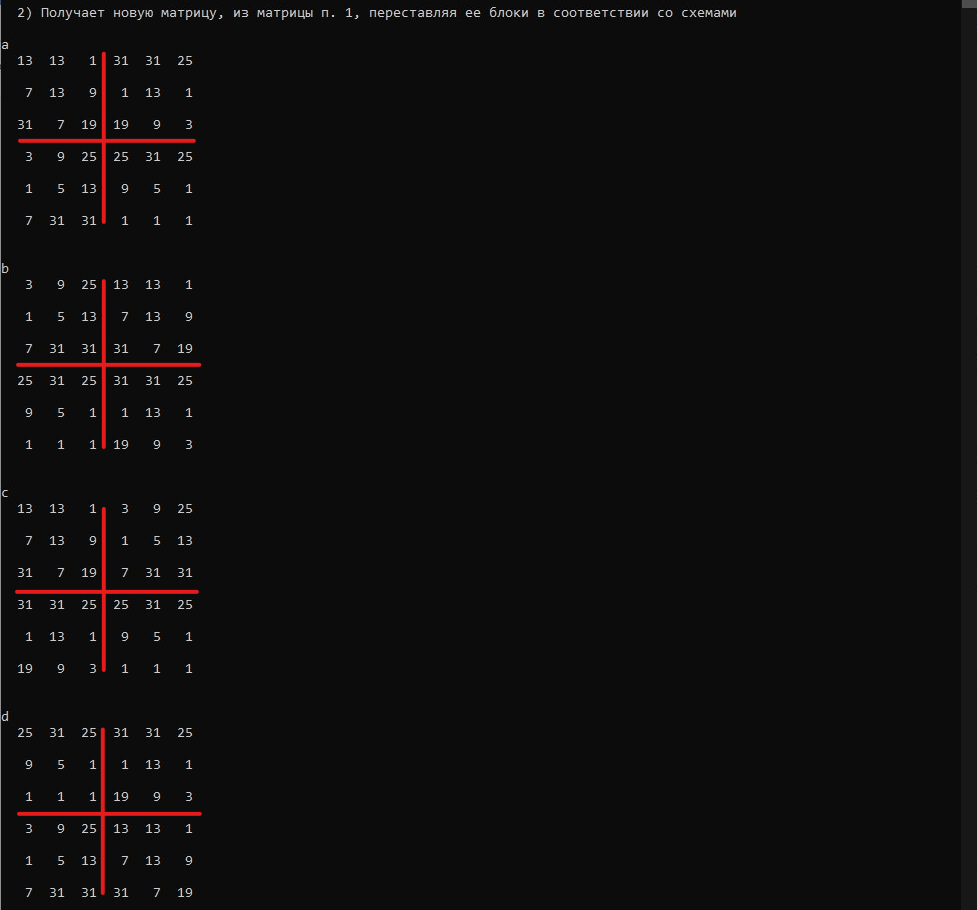
Для выполнения третьего пункта была написана функция сортировки для двумерного массива sort(int\*\* a, int size). Сначала значения упорядочиваются по строкам и столбцам, далее если последний элемент в стоке больше первого элемента в следующей строке, то они меняются местами, и значения опять упорядочиваются по строкам и столбцам, и так продолжается пока массив не будет полностью отсортирован.

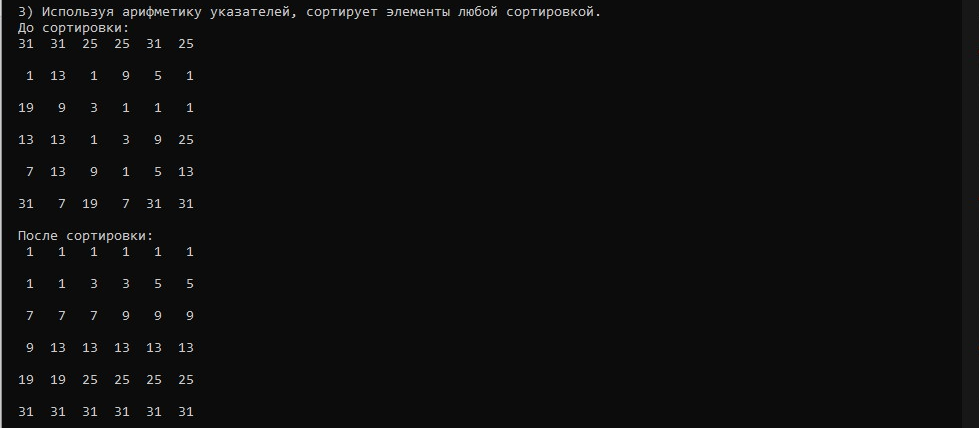
Для выполнения четвертого пункта используется функция actions(int\*\* A, int\*\* C, int N, int znach, int size). Которая принимает на два вход массива, число, введенное пользователем и размер массива. Функция копирует массив A в массив С и далее работает с массивом С.

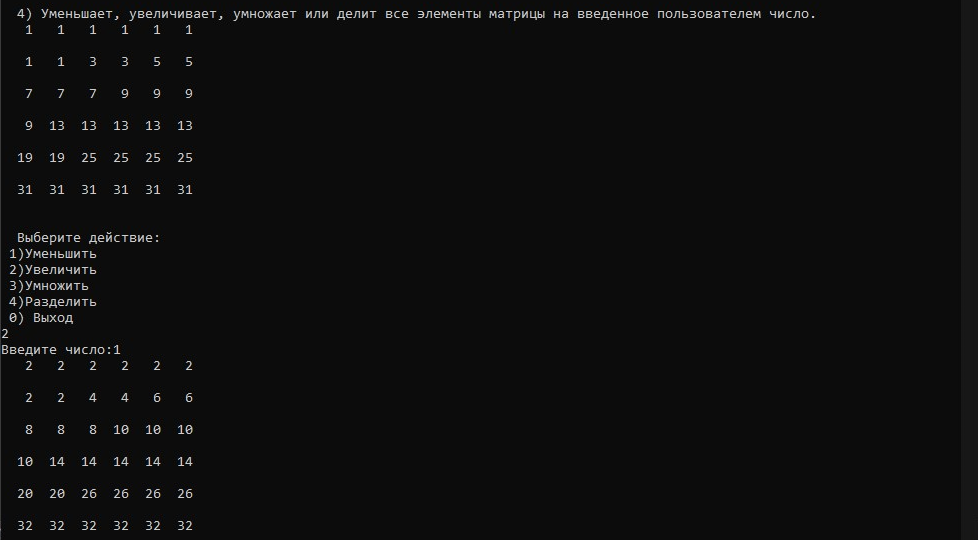
Как вы могли заметить в программе присутствует функция copy(int\*\* A, int\*\* C, int size). Она создана для того, чтобы работать с двумя массивами, заполненными в первом пункте, посредством создания их копий.

**Результаты работа программы.**

****

****

****

****

**Выводы.**

Программа работает правильно. Все полученные результаты соответствуют теоретическим данным.