Цель работы.

Разработать алгоритм и написать программу на языке C++, которая позволяет: создать квадратичную целочисленную матрицу. Используя арифметику указателей, выполнить сортировку элементов.

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. Все данные (переменные, константы и др.) хранятся в памяти. Память представляет собой непрерывную последовательность ячеек (байтов), каждая из которых имеет свой номер – адрес:

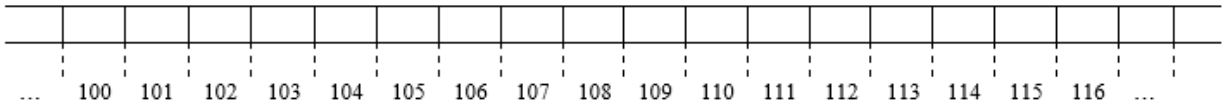

Рисунок 1 Представление памяти.

При определении, например, некоторой переменной, она располагается в памяти по определенному адресу и занимает столько ячеек, сколько требует тип этой переменной. Говорят, что переменная **A** располагается по адресу 101 и занимает 4 байта, а переменная **B** имеет адрес 105 и занимает 8 байт памяти.

Указатели – это тоже обычные переменные, но они **служат для хранения адресов памяти**. Указатели определяются в программе следующим образом:

<тип данных> *<имя переменной>

Здесь <**тип данных**> определяет так называемый **базовый тип указателя**.

<Имя переменной> является идентификатором переменной-указателя.

Признаком того, что это переменная указатель, является символ *, располагающийся между базовым типом указателя и именем переменной-указателя. Указатели поддерживают ряд операций: присваивание, получение адреса указателя, получение значения по указателю, некоторые арифметические операции и операции сравнения.

Так как ссылка не является объектом, то нельзя определить указатель на ссылку, однако можно определить ссылку на указатель. Через подобную ссылку можно изменять значение, на которое указывает указатель или изменять адрес самого указателя.

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. Этот адрес можно получить также через операцию &.

К указателям могут применяться операции сравнения>,> =, <, <=, ==,! =. Операции сравнения применяются только к указателям одного типа и к значениям NULL и nullptr. Для сравнения используются номера адресов:

```
int a = 10;
    int b = 20;
    int *pa = &a;
    int *pb = &b;

    if (pa > pb)
```

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.

Указатели – это очень мощное, полезное, но и очень опасное средство. Ошибки, которые возникают при неправильном использовании указателей, кроме того, что могут приводить к серьезным и непредсказуемым ошибкам в работе программы, еще и очень трудно диагностировать (обнаруживать).

Основная и наиболее часто встречающаяся ошибка при работе с указателями связана с использованием **неинициализированных указателей**.

Использование арифметики указателей при работе с массивами приводит обычно к уменьшению объема генерируемого кода программы и к уменьшению времени ее выполнения, то есть к увеличению быстродействия.

Постановка задачи.

Необходимо написать программу, которая:

1) Используя арифметику указателей, заполняет квадратичную целочисленную матрицу порядка *N* (6) случайными числами от 1 до N*N согласно схеме, приведенной на рис2. Пользователь должен видеть процесс заполнения квадратичной матрицы.

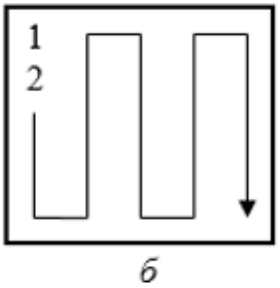

Рисунок 2 Тип вывода.

2) Получает новую матрицу, из матрицы п. 1, переставляя ее блоки в соответствии со схемами рис 3.

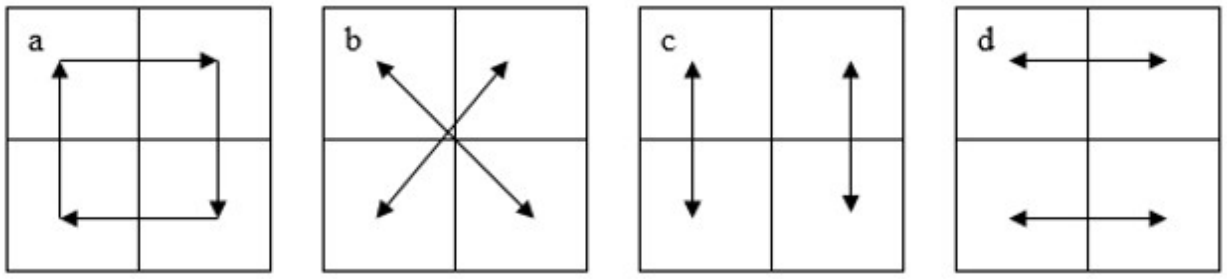

Рисунок 3 Схемы перестановки блоков.

3) Используя арифметику указателей, сортирует элементы любой сортировкой.

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

Выполнение работы.

Используя арифметику указателей, заполняет квадратичную целочисленную матрицу порядка *N* (6) случайными числами от 1 до N*N согласно схеме, приведенной на рис2.

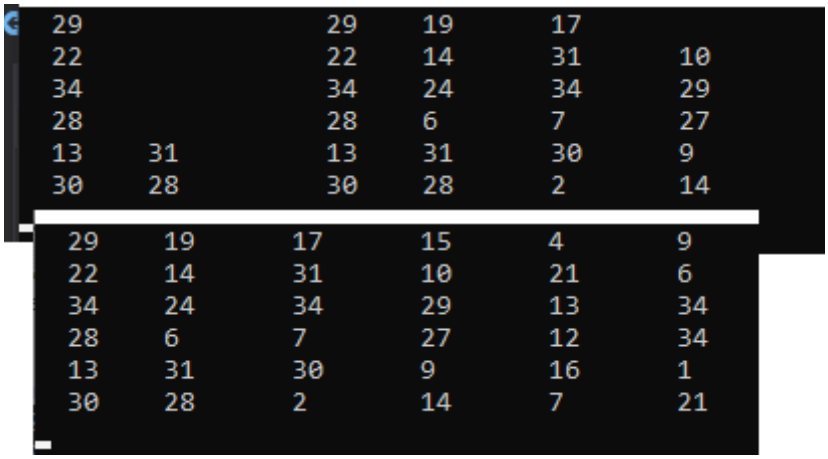

Рисунок 4 Пример работы программы вывода.

Получает новую матрицу, из матрицы п. 1, переставляя ее блоки в соответствии со схемами рис 3.

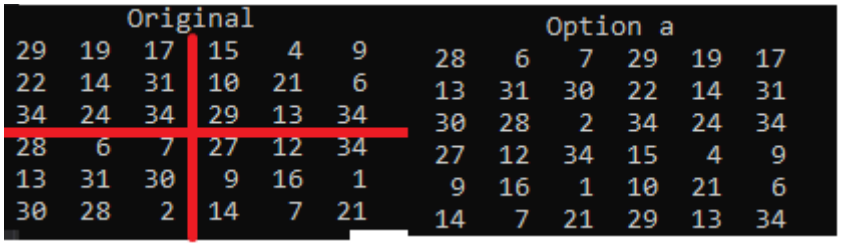

Рисунок 5 Перестановка блоков по схеме а.

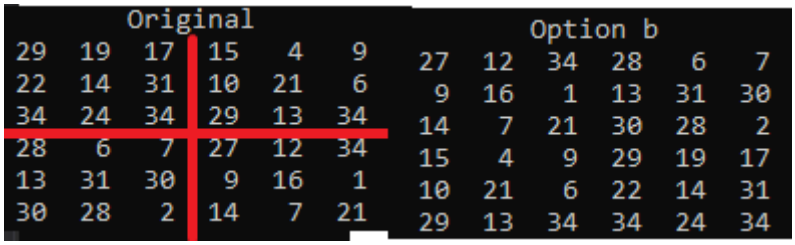

Рисунок 6 Перестановка блоков по схеме b.

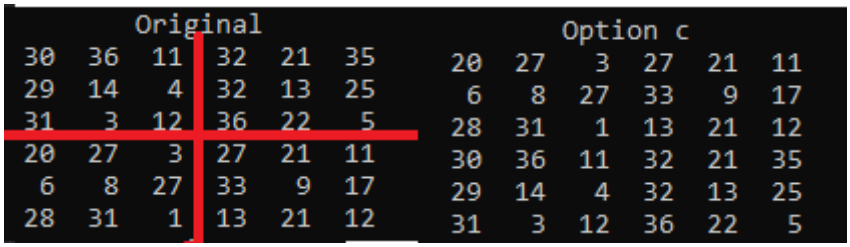

Рисунок 7 Перестановка блоков по схеме c.

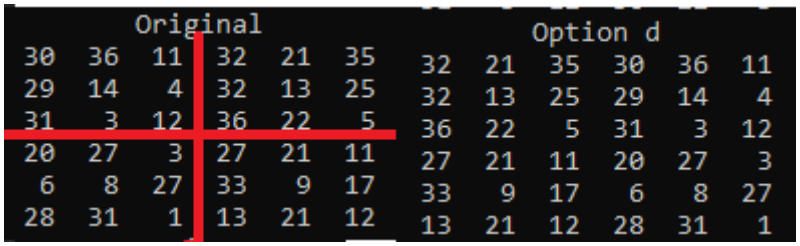

Рисунок 8 Перестановка блоков по схеме d.

Используя арифметику указателей, сортируем элементы.

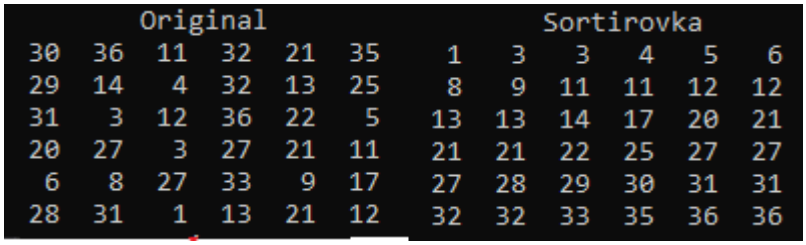

Рисунок 9 Сортировка массива.

Сортировка массива был применен алгоритм сортировки пузырьковой сортировкой. Принцип работы пузырьковой сортировки можно описать в три пункта:

1. Прохождение по всему массиву;

2. Сравнивание между собой пар соседних ячеек;
3. Если при сравнении оказывается, что значение ячейки j меньше, чем значение ячейки j - 1, то мы меняем значения этих ячеек местами;

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

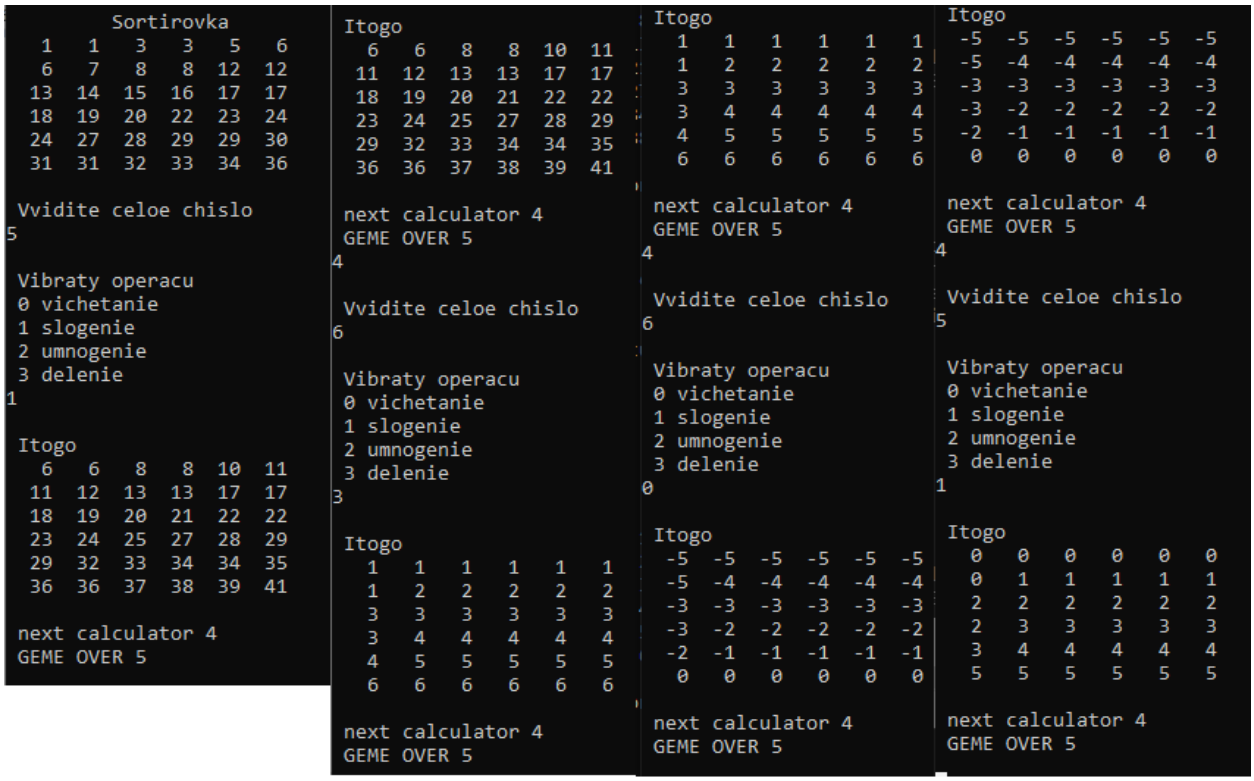

Рисунок 10 Работа программы с элементами массива.

Выводы.

Данная практическая работа позволяет разобраться с работой двумерных массивов. Двумерный массив отличается от одномерного тем же чем линия от плоскости, т.е. соответствием параметров, однозначно определяющих конкретную сущность. В одномерном массиве для однозначного определения элемента хватит одного параметра(как в случае с прямой для определения точки на ней достаточно одной цифры), в то время

как в двумерном массиве необходимо два параметра для идентификации текущего элемента(для точки на плоскости x, y).

Также познакомился с указателями главное различие кроется именно в работе с указателями и ссылками:

1. Указатель можно переназначать неограниченное количество раз (ссылка после привязки не может быть перемещена на другую ячейку памяти).
2. Указатель может быть равен NULL (нулевой указатель - указывает в никуда). Ссылка всегда указывает на определенный объект.
3. Нельзя получить адрес ссылки таким же образом, как с указателями.
4. И самое главное - не существует арифметики ссылок, а к указателям можно применять некоторые арифметические операции. Конечно, можно получить адрес объекта по ссылке и применить к этому адресу арифметику указателей, но это немного другой подход.

ПРИЛОЖЕНИЕ А

ПОЛНЫЙ КОД ПРОГРАММЫ

ПРАКТИЧЕСКАЯРАБОТА