**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №3**

**по дисциплине «Программирование»**

Тема: Двумерные статические массивы. Указатели

|  |  |  |
| --- | --- | --- |
| Студент гр. 2372 |  | Соколовский В.Д. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Изучение двумерных статических массив и основных алгоритмов работы с ними. Изучение ссылок и указателей, арифметики указателей. Написание программы, заполняющей матрицу размера N (N = 6, 8, 10) и заполняющий его паттернами «змейка» и «спираль». Программа должна так же менять местами блоки (подмассивы) матрицы в соответствии с четырьмя паттернами обмена из задания. Сортировать матрицу с использованием арифметики указателей и увеличивать, уменьшать, умножать и делить все элементы матрицы на число, введенное пользователем.

**Основные теоретические положения.**

Указатели и ссылки 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 другое.

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.

Операция разыменования указателя позволяет получить объект по адресу, который хранится в указателе.

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.

**Постановка задачи.**

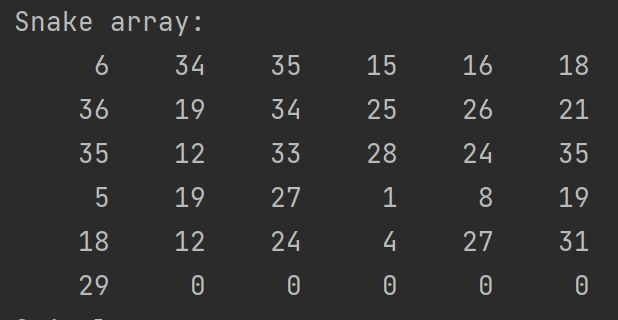
Разработать алгоритм и написать программу, которая:

1. Создает целочисленную матрицу размерности N\*N и заполняет её нулями.
2. Заполняет матрицу случайными числами от 1 до N^2 змейкой
3. Заполняет матрицу случайными числами от 1 до N^2 по спирали
4. Меняет местами блоки матрицы четырьмя методами, метод выбирает пользователь
5. Сортирует матрицу, используя арифметику указателей
6. Увеличивает, уменьшает, умножает или делит все элементы матрицы на число, которое вводит пользователь. Операция также выбирается пользователем.

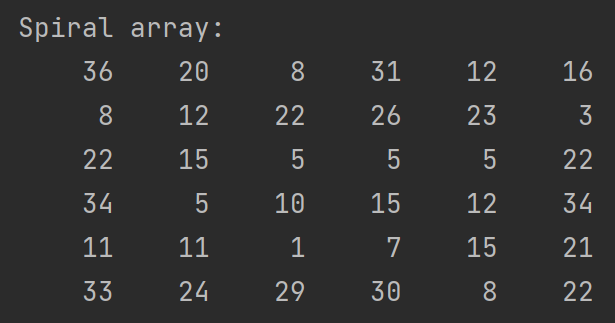
**Выполнение работы.**

Код программы представлен в приложении А.

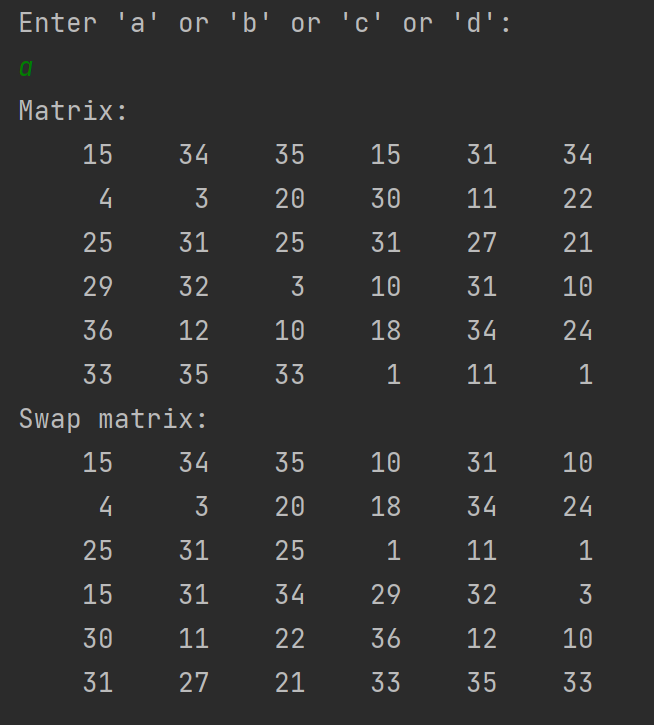
1. массив заполняется <Змейкой>



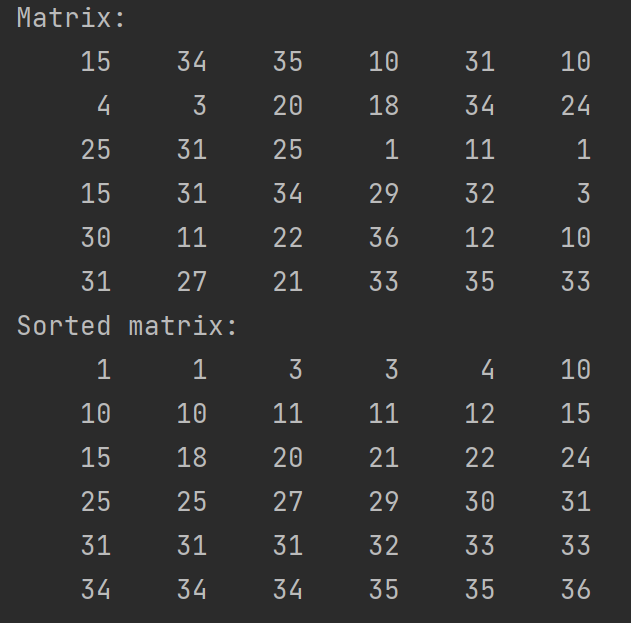
1. массив заполняется <Спиралью>



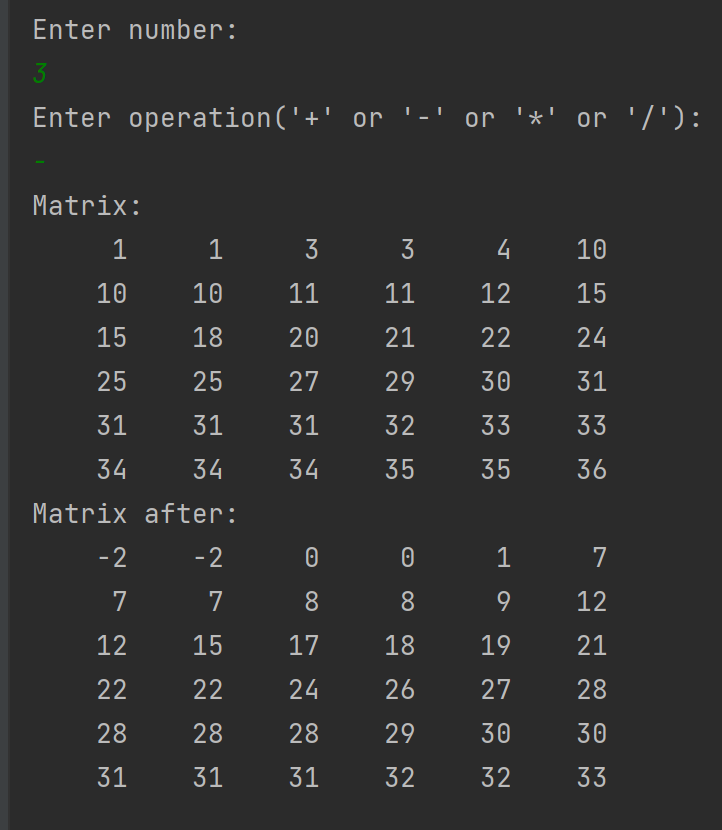
1. ожидается пользовательский выбор паттерна обмена блоков матрицы. После выбора производится обмен и выводятся значения матрицы до и после обмена.



1. выполняется сортировка матрицы сортировкой вставками с использованием арифметики указателей



1. ожидается пользовательский выбор операции (сложение, вычитание, умножение или деление) и все элементы увеличиваются, уменьшаются, умножаются или делятся (в зависимости от выбранной операции) на число, которое так же вводит пользовать.



**Выводы.**

В ходе работы были изучены двумерные статические массивы и методы работы с ними. Изучены ссылки и указатели, арифметика указателей.

Приложение А

рабочий код

#include <iostream>

#include <cstdlib>

using namespace std;

void matr(int \*arr, int N, int wdt = 6){ // matrix declaration(объявление матрицы)

for(int i = 0; i < N \* N; i = i + N){

for(int j = i; j < i + N; j++){

cout.width(wdt); // set width(задать ширину)

cout << arr[j];

}

putchar('\n');

}

}

int randInt(int a, int b) {

return a + rand() % b;

}

void fillZr(int \*arr, int N){ // zero fill(нулевое поле)

for(int i = 0; i < N \* N; i++){

\*arr++ = 0;

}

}

void fillSnk(int \*arr, int N){ // creating a snake(создание змейки)

matr(arr, N);

int \*pArr = arr;

int c = N;

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

int \*pR;

if(c > 0){

pR = pArr + N \* N - N;

}

else{

pR = pArr - N \* N + N;

}

while(pArr - c != pR){

\*pArr = randInt(1, N \* N);

pArr = pArr + c;

}

pArr = pArr - c - 1;

c = -c;

}

}

void fillSpr(int \*arr, int N){

matr(arr, N);

int \*pArr = arr;

int size = N;

for(int sq = 0; sq < N / 2; sq++){

int c = 1;

for(int dir = 0; dir < 4; dir++){

for(int i = 0; i < size - 1; i++){

\*pArr = randInt(1, N \* N);

pArr = pArr + c;

}

c = abs(c) == 1 ? N : 1;

c = dir < 1 ? + c : -c;

}

size = size - 2;

pArr = pArr + N + 1;

}

}

int \*pointQ(int \*arr, int N, int quaN) {

switch(quaN){

case 1:{

return (arr);

}

case 2:{

return (arr + N / 2);

}

case 3:{

return (arr + N \* N / 2);

}

case 4:{

return (arr + N \* N / 2 + N / 2);

}

}

}

void swapQ(int \*arr, int N, int q1, int q2){

int \*pQ1 = pointQ(arr, N, q1);

int \*pQ2 = pointQ(arr, N, q2);

for(int i = 0; i < N / 2; i++){ // swap(меняем)

for(int j = 0; j < N / 2; j++){

swap(\*(pQ1++), \*(pQ2++));

}

pQ1 = pQ1 + N - N / 2;

pQ2 = pQ2 + N - N / 2;

}

}

void insertSort(int \*arr, int N){

for(int \*p1 = arr + 1; p1 < arr + N \* N; p1++){

for(int \*p2 = p1; p2 > arr && \*(p2 - 1) > \*p2; p2--){

swap(\*(p2 - 1), \*p2);

}

}

}

void matrV(int \*arr, int N, int op, int df){

for(int \*pArr = arr; pArr < arr + N \* N; pArr++){

switch(op){

case '+':{

\*pArr = \*pArr + df;

break;

}

case '-':{

\*pArr = \*pArr - df;

break;

}

case '\*':{

\*pArr = \*pArr \* df;

break;

}

case '/':{

\*pArr = \*pArr / df;

break;

}

}

}

}

int main() {

const int N = 6;

int mtrx[N][N];

int \*ptrMatr = &mtrx[0][0];

cout << "Zero fill array: \n";

fillZr(ptrMatr, N);

fillSnk(ptrMatr, N);

cout << "Snake array: \n";

fillSpr(ptrMatr, N);

cout << "Spiral array: \n"; // output spiral array

fillSpr(ptrMatr, N);

cout << "\n" << "\n";

char us;

cout << "Enter 'a' or 'b' or 'c' or 'd': \n";

cin >> us;

cin.sync();

cout << "Matrix:\n";

matr(ptrMatr, N);

switch(us){

case 'a':{

swapQ(ptrMatr, N, 1, 2);

swapQ(ptrMatr, N, 2, 4);

swapQ(ptrMatr, N, 4, 3);

swapQ(ptrMatr, N, 3, 1);

break;

}

case 'b':{

swapQ(ptrMatr, N, 1, 4);

swapQ(ptrMatr, N, 2, 3);

break;

}

case 'c':{

swapQ(ptrMatr, N, 1, 3);

swapQ(ptrMatr, N, 2, 4);

break;

}

case 'd':{

swapQ(ptrMatr, N, 1, 2);

swapQ(ptrMatr, N, 3, 4);

break;

}

}

cout << "Swap matrix:\n";

matr(ptrMatr, N);

cout << "\n" << "\n";

// Insertion sort

cout << "Matrix:\n";

matr(ptrMatr, N);

insertSort(ptrMatr, N);

cout << "Sorted matrix:\n";

matr(ptrMatr, N);

// Decreasing, increasing, multiplying, dividing a matrix by the entered number

int num;

cout << "Enter number:\n";

cin >> num;

cin.sync(); // removes unread characters from the input stream(убирает непрочитанные символы из входного потока)

char oper;

cout << "Enter operation('+' or '-' or '\*' or '/'):\n";

cin >> oper;

cin.sync();

cout << "Matrix:\n";

matr(ptrMatr, N);

matrV(ptrMatr, N, oper, num);

cout << "Matrix after:\n";

matr(ptrMatr, N);

}