**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(ка) гр. | Тимошенков И.М. |  |
| Преподаватель | Глущенко А. Г. |  |

Санкт-Петербург

2022

**Цель работы.**

Научиться работе с указателями и двумерными массивами.

**Основные теоретические положения.**

В изучаемых нами языках программирования между массивами и указателями имеется очень тесная связь.

Когда мы определяем в программе некоторый массив, например, переменная **Arr** без индексов представляет собой указатель на первый элемент массива в данном случае из 10 целых чисел (содержит адрес первого элемента массива). Если вывести на экран значение переменной **Arr**, мы увидим некоторое целое значение в шестнадцатеричном формате, соответствующее адресу первого элемента этого массива.

**Замечание**. Именно по этой причине в языке C++ отсутствует операция присвоения сразу всех значений одного массива другому (в некоторых других языках, например, в Pascal такая возможность имеется). Действительно, если имеются два массива

то попытка выполнить присвоение **A1 = A2**привела бы к тому, что переменная **A1** стала бы указывать на ту же область памяти, что и переменная **A2** (мы скопировали адрес из **A2**в **A1**, а не содержимое одного массива в другой).  Адрес, который хранился ранее в переменной **A1,** был бы утерян, что привело бы к утечке памяти (для десяти элементов массива **A1** в памяти было выделено место, но теперь мы “забыли”, где оно находится, то есть потеряли память). По этой причине подобные операции с массивами в языке C++ запрещены. Более того, запрещены любые изменения значения переменной массива.

Указателю, имеющему такой же базовый тип, как и элементы массива, можно присвоить массив следующим образом. Но обратное присвоение выполнить невозможно.

Такое присвоение невозможно, поскольку переменная массива – это константа, изменение которой запрещено.

Так как переменная массива является указателем на первый элемент массива, появляются дополнительные возможности по работе с массивами на основе использования арифметики указателей. Например, чтобы получить 5–й элемент массива **Arr**можно воспользоваться одним из следующих выражений:

**Arr[4]**или   \*(**Arr + 4)**или**\*( p + 4)**

Первое выражение – это пример обычной индексации элементов массива. Во втором и третьем выражениях мы использовали арифметику указателей и с помощью операции + получили адрес пятого элемента массива. Затем с помощью операции \* взяли значение по этому адресу и получили значение 5-го элемента массива. Обратите внимание на скобки в этих выражениях, если их не поставить и написать \***Arr + 4**или **\*p + 4**, то эти выражения будут равны значению первого элемента массива увеличенного на 4, так как операция \* имеет больший приоритет, чем операция +.

Вот пример фрагмента программы для работы с массивом с помощью обычной индексации элементов массива. Этот фрагмент обеспечивает ввод элементов целочисленного массива с клавиатуры, вычисление квадратов значений элементов массива, а затем вывод элементов массива на экран:

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.

Поскольку указатель и имя массива, в большой степени, взаимозаменяемы, указатели можно индексировать, как обычные массивы. Можно создавать и массивы указателей.

Массив **M** – это трехэлементный массив указателей на целые значения, то есть каждый элемент этого массива представляет собой указатель на целое.

С помощью массивов указателей можно моделировать различные интересные конструкции данных. Например, пусть имеется квадратная матрица размерности 5 х 5 симметричная относительно главной диагонали. Для ее однозначного представления достаточно хранить в памяти  не все 25 элементов этой матрицы, а только 15 (например, элементы под главной диагональю вместе с элементами главной диагонали). Для этого можно предложить следующую конструкцию:

А это пятиэлементный массив указателей на символы, инициализированный некоторыми текстовыми строками:

Как это работает: когда компилятор встречает в программе некоторый текст, заключенный в кавычки, в памяти создается символьный массив соответствующей этому тексту длины и адрес этого символьного массива присваивается соответствующему элементу – указателю массива **Words**.

**Выполнение работы.**

Код программы представлен в приложении А.

#include <iomanip>

#include <iostream>

#include <Windows.h>

using namespace std;

// функция, которая вывводит массив зигзагом

void zigzag(int arr3[10][10], int arr[10][10], int size) { //слева пустой массив

HANDLE hStdout;

COORD destCoord;

hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

destCoord.X = 0;

destCoord.Y = 0;

SetConsoleCursorPosition(hStdout, destCoord);

int columnChanges = 0;

int stepZigzag = 0;

int\* ppp = &arr[0][0];

int spacex = 0;

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

if (stepZigzag % 2 == 0) {

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

arr3[g][columnChanges] = \*(ppp);

destCoord.X = columnChanges + spacex;

destCoord.Y = g + g + 4;

SetConsoleCursorPosition(hStdout, destCoord);

Sleep(10);

cout << \*ppp;

ppp += 1;

}

spacex += 3;

}

else {

for (int jj = size - 1; jj > -1; jj--) {

arr3[jj][columnChanges] = \*(ppp);

destCoord.X = columnChanges + spacex;

destCoord.Y = jj + jj + 4;

SetConsoleCursorPosition(hStdout, destCoord);

Sleep(10);

cout << \*ppp;

ppp += 1;

}

spacex += 3;

}

columnChanges += 1;

stepZigzag += 1;

}

destCoord.X = 0;

destCoord.Y = 18;

SetConsoleCursorPosition(hStdout, destCoord);

}

// изменение положения блоков матрицы, которое производится через индексы, меняет левые и правые блоки

void left\_rightSwap (int arr[10][10], int size) {

for (int h = 0; h < size / 2; h++) {

for (int k = 0; k < size / 2; k++) {

swap(arr[k][h], arr[k][h + (size / 2)]);

swap(arr[k + (size / 2)][h], arr[k + (size / 2)][h + (size / 2)]);

}

}

}

// меняет верхние и нижние блоки

void up\_downSwap(int arr[10][10], int size) {

for (int h = 0; h < size / 2; h++) {

for (int k = 0; k < size / 2; k++) {

swap(arr[h][k], arr[h + (size / 2)][k]);

swap(arr[h][k + (size / 2)], arr[h + (size / 2)][k + (size / 2)]);

}

}

}

// меняет блоки крест-накрест

void left\_down\_rightSwap(int arr2[10][10], int size) {

for (int h = 0; h < size / 2; h++) {

for (int k = 0; k < size / 2; k++) {

swap(arr2[k][h], arr2[k + (size / 2)][h + (size / 2)]);

swap(arr2[k + (size / 2)][h], arr2[k][h + (size / 2)]);

}

}

}

// меняет блоки по-спирали (пункт а)

void swapBySpiral(int arr3[10][10], int nn) {

for (int h = 0; h < nn / 2; h++) {

for (int k = 0; k < nn / 2; k++) {

swap(arr3[h][k + (nn / 2)], arr3[h + (nn / 2)][k + (nn / 2)]);

}

}

for (int h = 0; h < nn / 2; h++) {

for (int k = 0; k < nn / 2; k++) {

swap(arr3[k][h], arr3[k][h + (nn / 2)]);

}

}

for (int h = 0; h < nn / 2; h++) {

for (int k = 0; k < nn / 2; k++) {

swap(arr3[h][k], arr3[h + (nn / 2)][k]);

}

}

}

//выводит матрицу на экран

void matrixOutPut(int arr[10][10], int size) {

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

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

cout << arr[i][j] << '\t';

}

cout << '\n';

}

cout << '\n';

cout << '\n';

cout << '\n';

}

// сортировка 2мерного массива

void bubbleSort(int arr[10][10], int size)

{

if (size == 6) {

int arr2[6][6];

int\* s = &arr2[0][0];

int\* end = &arr[9][9];

for (int\* i = &arr[0][0]; i < end; (++i, s++)) {

if (\*i < 0) {

s--;

}

else {

\*s = \*i;

}

}

int\* end2 = &arr2[size - 1][size - 1];

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

for (int\* j = &arr2[0][0]; j < end2; j++) {

if (\*j > \*(j + 1))

swap(\*j, \*(j + 1));

}

}

s = &arr2[0][0];

int k = 1;

for (int\* i = &arr[0][0]; i < end + 1; (i++, s++)) {

if (\*i < 0) {

s--;

}

else {

\*i = \*s;

}

}

}

if (size == 8) {

int arr2[8][8];

int\* s = &arr2[0][0];

int\* end = &arr[9][9];

for (int\* i = &arr[0][0]; i < end; (++i, s++)) {

if (\*i < 0) {

s--;

}

else {

\*s = \*i;

}

}

int\* end2 = &arr2[size - 1][size - 1];

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

for (int\* j = &arr2[0][0]; j < end2; j++) {

if (\*j > \*(j + 1))

swap(\*j, \*(j + 1));

}

}

s = &arr2[0][0];

int k = 1;

for (int\* i = &arr[0][0]; i < end + 1; (i++, s++)) {

if (\*i < 0) {

s--;

}

else {

\*i = \*s;

}

}

}

if (size == 10) {

int\* end = &arr[size - 1][size - 1];

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

for (int\* j = &arr[0][0]; j < end; j++) {

if (\*j > \*(j + 1))

swap(\*j, \*(j + 1));

}

}

}

}

// создает матрицу

void creatMatrix(int arr2[10][10], int size) {

int\* pointerOnMat = &arr2[0][0];

int\* end = &arr2[size-1][size-1];

for (int\* pointerOnMat = &arr2[0][0]; pointerOnMat <= end; pointerOnMat++)

{

\*pointerOnMat = 0 + rand() % (size \* size) + 1;

}

}

// умножает матрицу на заданное число

void multiplyMatrix(int arr[10][10], int size) {

cout << "Введите число на которое хотите умножить матрицу" << '\n';

int number;

cin >> number;

int\* pointerOnMat = &arr[0][0];

for (int\* end = pointerOnMat + (size \* size); pointerOnMat != end; pointerOnMat++)

{

\*pointerOnMat \*= number;

}

}

// прибавляет к матрице число

void addMatrix(int arr[10][10], int size) {

cout << "Введите число которое хотите прибавить к матрице" << '\n';

int number;

cin >> number;

int\* pointerOnMat = &arr[0][0];

for (int\* end = pointerOnMat + (size \* size); pointerOnMat != end; pointerOnMat++)

{

\*pointerOnMat += number;

}

cout << '\n';

cout << '\n';

cout << '\n';

}

// делит матрицу на число

void divideMatrix(int arr[10][10], int size) {

cout << "Введите число на которое хотите разделить матрицу" << '\n';

int number;

cin >> number;

int\* pointerOnMat = &arr[0][0];

for (int\* end = pointerOnMat + (size \* size); pointerOnMat != end; pointerOnMat++)

{

\*pointerOnMat /= number;

}

}

// вычитает из матрицы число

void subtractMatrix(int arr[10][10], int size) {

cout << "Введите число на которое хотите уменьшить матрицу" << '\n';

int number;

cin >> number;

int\* pointerOnMat = &arr[0][0];

for (int\* end = pointerOnMat + (size \* size); pointerOnMat != end; pointerOnMat++)

{

\*pointerOnMat -= number;

}

}

int main() {

setlocale(0, "Rus");

srand(time(0));

int size;

cout << "Введите размер матрицы (6, 8, 10): ";

cin >> size;

cout << '\n';

if (size != 8 and size != 6 and size != 10) {

cout << "Вы ввели неправильный размер матрицы";

return 1;

}

int arr[10][10];

int arr2[10][10];

creatMatrix(arr, size);

zigzag(arr2, arr, size);

cout << "Выберете, как хотите поменять блоки матрицы: \n";

cout << "1 - По спирали\n";

cout << "2 - Верхине и нижние\n";

cout << "3 - Левые и правые\n";

cout << "4 - Крест - накрест\n";

int s;

cin >> s;

switch (s)

{

case 1:

swapBySpiral(arr2, size);

matrixOutPut(arr2, size);

break;

case 2:

up\_downSwap(arr2, size);

matrixOutPut(arr2, size);

break;

case 3:

left\_rightSwap(arr2, size);

matrixOutPut(arr2, size);

break;

case 4:

left\_down\_rightSwap(arr2, size);

matrixOutPut(arr2, size);

break;

}

cout << "Сортировка матрицы \n";

bubbleSort(arr2, size);

matrixOutPut(arr2, size);

cout << "Выберете операцию над матрицей: \n";

cout << "1 - Прибавление числа\n";

cout << "2 - Умножение на другое число\n";

cout << "3 - Вычитание\n";

cout << "4 - Деление\n";

int ss;

cin >> ss;

switch (ss)

{

case 1:

addMatrix(arr2, size);

matrixOutPut(arr2, size);

break;

case 2:

multiplyMatrix(arr2, size);

matrixOutPut(arr2, size);

break;

case 3:

subtractMatrix(arr2, size);

matrixOutPut(arr2, size);

break;

case 4:

divideMatrix(arr2, size);

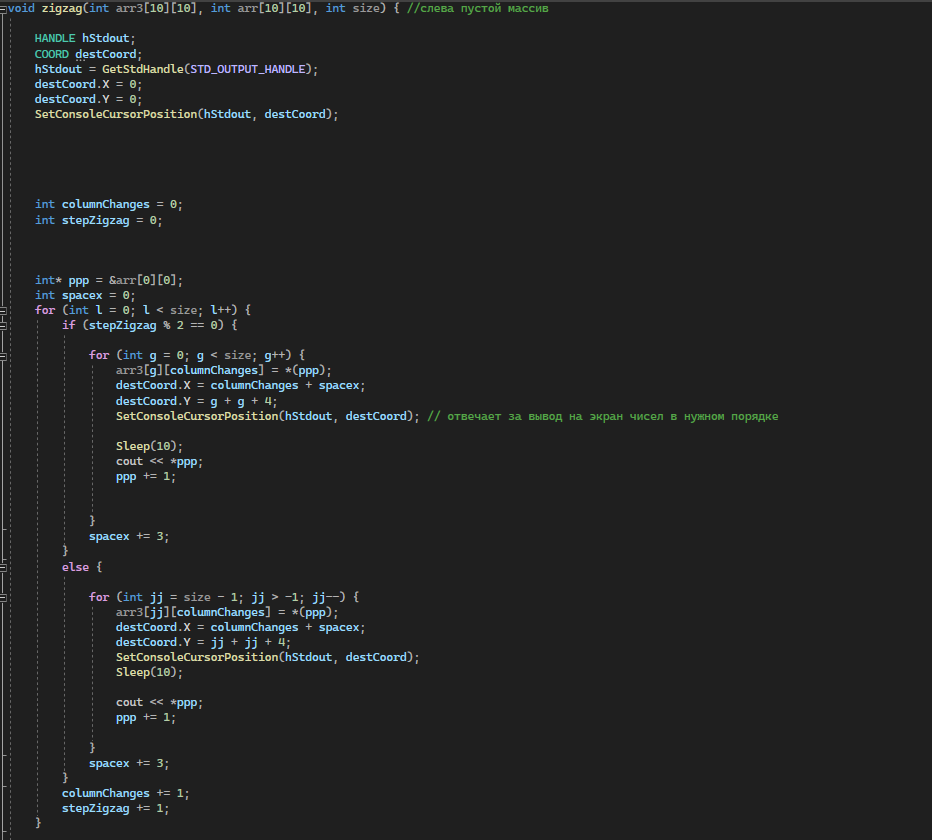
matrixOutPut(arr2, size);

break;

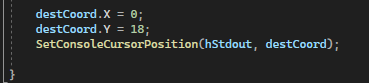
}

}

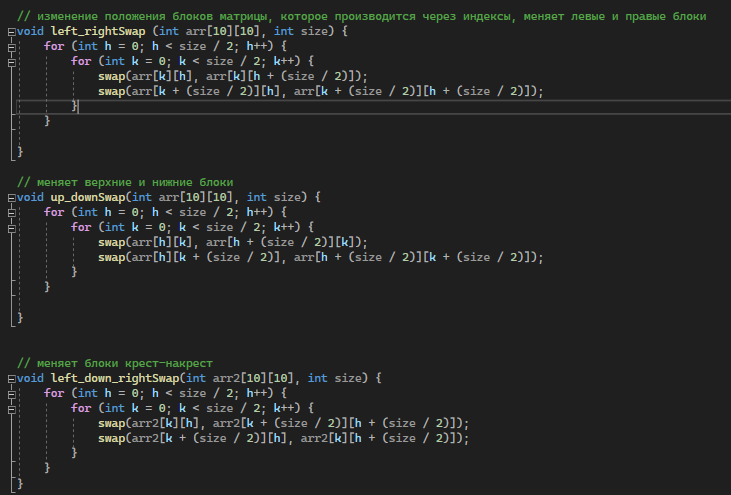
**Блок описания кода и использованных алгоритмов**

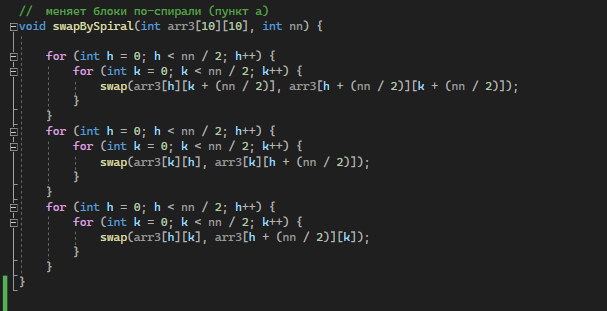
****

Перемещение каретки вниз, чтобы выводить информацию дальше

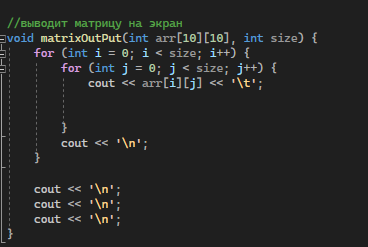


Алгоритмы перемещения блоков

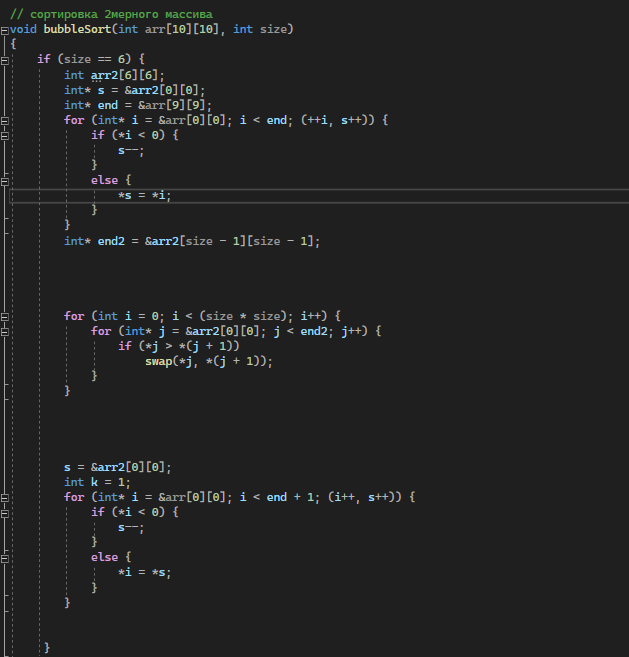




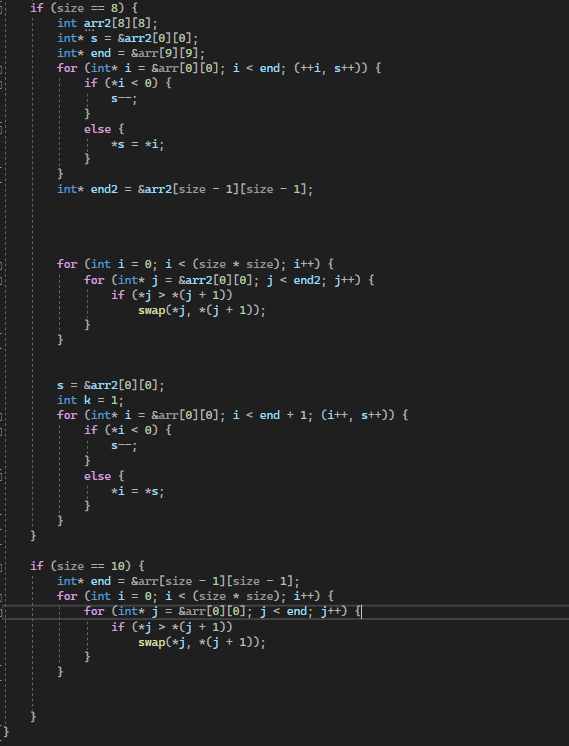
Алгоритм вывода матрицы на экран

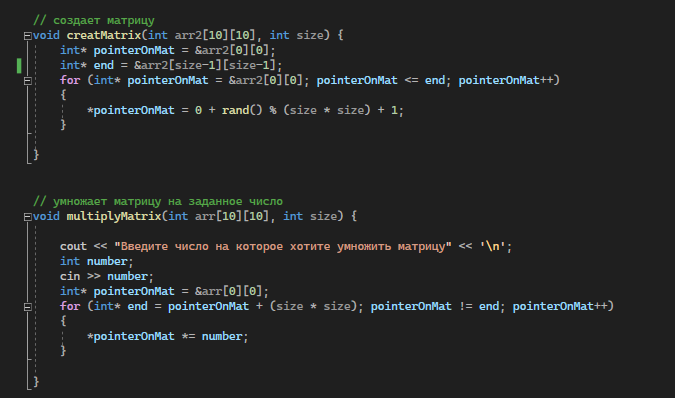
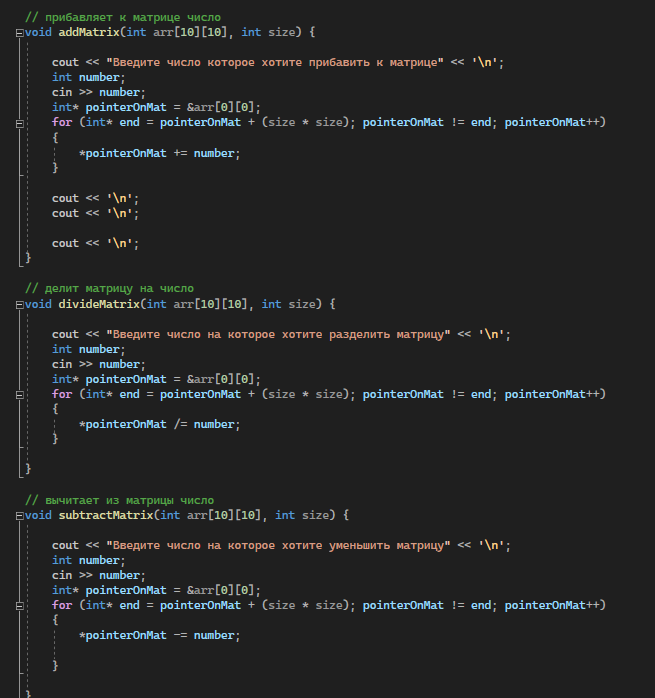


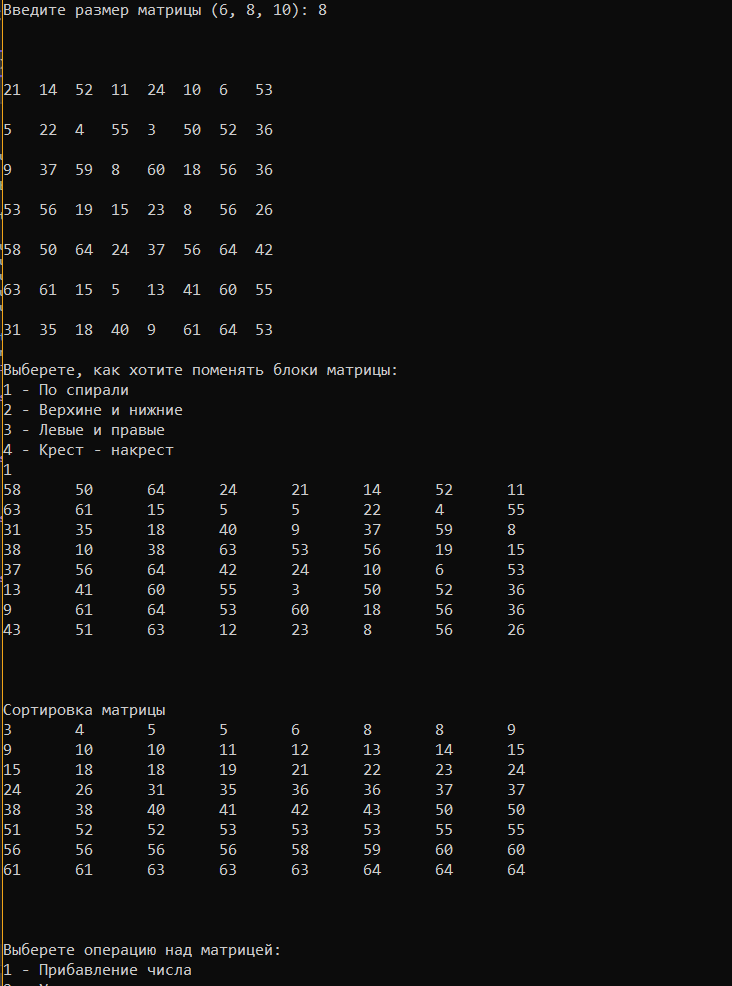
Ниже представлен кусок кода для сортировки 2 мерного массива n = 6;

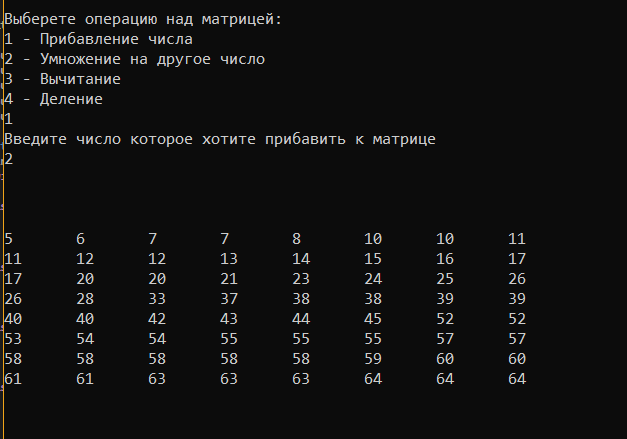


Так как во всех функциях изначально поступает массив именно 10 на 10, то при фактическом размере 6 на 6, я не смог оптимизировать сортировку при таких значениях, так как часть ячеек не хранит значений, из-за этого адекватно отсортировать и отобразить массив 6 на 6 или 8 на 8 у меня не получилось. Поэтому в алгоритме сортировки, если фактически массив имеет размер 6 на 6 или 8 на 8, то сначала двумерный массив 10 на 10, где есть только 36 или 64 значений, преобразуется в массив 6 на 6 или 8 на 8, в зависимости от выбора пользователя в начале программы, сортируется при помощи указателей и опять упаковывается в массив 10 на 10.

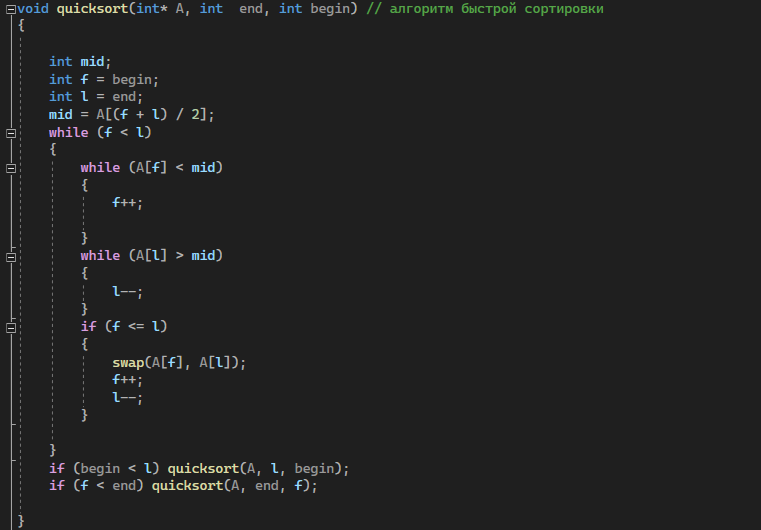


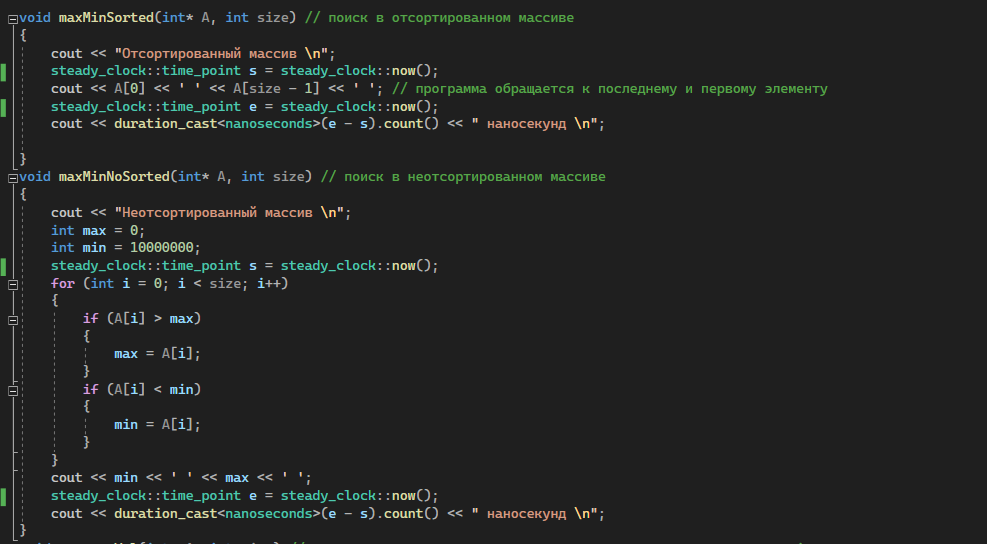
Работа программы



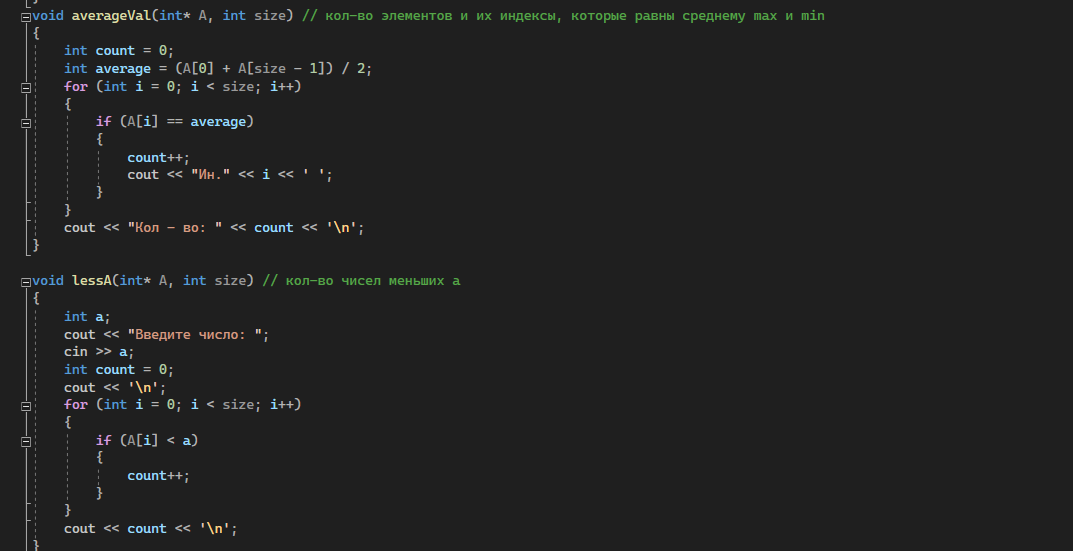
Алгоритм быстрой сортировки (рекурсивное решение)



Алгоритмы поиска минимального и максимального значения



Алгоритм поиска числа **< а или > b** и числа равному среднему максимального и минимального.



Основная программа с формированием меню.