**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. | Карпачёв А. С. |  |
| Преподаватель | Глущенко А. Г. |  |

Санкт-Петербург

2022

**Цель работы.**

Изучение двумерных статических массивов, указателей, арифметики указателей.

**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. Массив представляет собой индексированную последовательность однотипных элементов с заранее определенным количеством элементов. Наглядно одномерный массив можно представить, как набор пронумерованных ячеек, в каждой из которых содержится определенное значение.

Элементы массива нумеруются с нуля. При описании массива используются те же модификаторы (класс памяти, const и инициализатор), что и для простых переменных.

Сортировка – процесс размещения элементов заданного множества объектов в определенном порядке. Когда элементы отсортированы, их проще найти, производить с ними различные операции. Сортировка напрямую влияет на скорость алгоритма, в котором нужно обратиться к определенному элементу массива.

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

Нулевой указатель (null pointer) — это указатель, который не указывает ни на какой объект. Если мы не хотим, чтобы указатель указывал на какой-то конкретный адрес, то можно присвоить ему условное нулевое значение.

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.

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.

**Постановка задачи.**

Необходимо написать программу, которая:

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.**

Код программы представлен в приложении А.

Блок описания кода и использованных алгоритмов

Используя арифметику указателей, производится заполнение матрицы двумя способами. При перемещении блоков матрицы, каждый блок сохраняется в отдельный массив, после чего исходная матрицы собирается заново, но уже в нужном порядке. При сортировке матрицы, все элементы переписываются в одномерный массив, сортируются, после чего заново строится матрица. При увеличении или уменьшении всех элементов, происходит последовательный проход по всем элементам матрицы и изменяются их элементы.

**Выводы.**

Изучен навык работы с двумерными статическими массивами. Изучен навык работы с указателями, основы арифметики указателей. Изучена базовая связь между массивами и указателями, способы обращения к элементам массивов через указатели.

Приложение А

рабочий код

#include <iostream>

#include <chrono>

#include <utility>

#include "windows.h"

using namespace std;

using namespace chrono;

int main()

{

setlocale(LC\_ALL, "russian");

srand(time(0));

int i, max, min, j, N, M, a[20][20], b[20][20];

// N количества срок

N = 6;

// M количества столбцов

M = 6;

//cout << "Рандомное заполнене матрицы выгледит так " << "\n";

min = 1;

max = 99;

int\* pN = &N, \* pM = &M;//указатели

//цикл по переменной i, в которой перебираем строки матрицы

for (i = 0; i < \*pN; i++) { // заполнение матрицы

//цикл по переменной j, в котором перебираем столбцы

for (j = 0; j < \*pM; j++) {

//"рандомное" заполнение матрицы

a[i][j] = rand() % (max - min + 1) + min;

//нулевая матрица

b[i][j] = 0;

}

}

for (int i = 0; i < M; i++) { // Вывод матрицы

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

cout << " " << b[i][j];

Sleep(100);

}

cout << endl;

}

HANDLE hStdout;

COORD destCoord;

hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

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

{

destCoord.X = 0 + 4;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][0];

cout.flush();

Sleep(200);

}

for (int i = 5; i >= 0; i--)

{

destCoord.X = 1 + 8;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][1];

cout.flush();

Sleep(200);

}

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

{

destCoord.X = 2 + 12;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][2];

cout.flush();

Sleep(200);

}

for (int i = 5; i >= 0; i--)

{

destCoord.X = 3 + 16;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][3];

cout.flush();

Sleep(200);

}

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

{

destCoord.X = 4 + 20;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][4];

cout.flush();

Sleep(200);

}

for (int i = 5; i >= 0; i--)

{

destCoord.X = 5 + 24;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][5];

cout.flush();

Sleep(200);

}

{

destCoord.X = 5 + 28;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << "";

cout.flush();

Sleep(200);

}

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

system("pause");

{

//задание два

//часть а

system("cls");

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

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

swap(a[i][j], a[i + (N / 2)][j]);

}

}

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

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

swap(a[i][j], a[i + (N / 2)][j + (N / 2)]);

}

}

//вывод Матрицы

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

{

destCoord.X = 0 + 4;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][0];

cout.flush();

Sleep(50);

}

for (int i = 5; i >= 0; i--)

{

destCoord.X = 1 + 8;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][1];

cout.flush();

Sleep(50);

}

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

{

destCoord.X = 2 + 12;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][2];

cout.flush();

Sleep(50);

}

for (int i = 5; i >= 0; i--)

{

destCoord.X = 3 + 16;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][3];

cout.flush();

Sleep(50);

}

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

{

destCoord.X = 4 + 20;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][4];

cout.flush();

Sleep(50);

}

for (int i = 5; i >= 0; i--)

{

destCoord.X = 5 + 24;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][5];

cout.flush();

Sleep(50);

}

{

destCoord.X = 5 + 28;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << "";

cout.flush();

Sleep(50);

}

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

system("pause");

}

{//часть Б

system("cls");

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

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

if (j < (N / 2)) {

swap(a[i][j], a[i + (N / 2)][j + (N / 2)]);

}

else {

swap(a[i][j], a[i + (N / 2)][j - (N / 2)]);

}

}

}

//вывод матрицы

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

{

destCoord.X = 0 + 4;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][0];

cout.flush();

Sleep(50);

}

for (int i = 5; i >= 0; i--)

{

destCoord.X = 1 + 8;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][1];

cout.flush();

Sleep(50);

}

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

{

destCoord.X = 2 + 12;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][2];

cout.flush();

Sleep(50);

}

for (int i = 5; i >= 0; i--)

{

destCoord.X = 3 + 16;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][3];

cout.flush();

Sleep(50);

}

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

{

destCoord.X = 4 + 20;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][4];

cout.flush();

Sleep(50);

}

for (int i = 5; i >= 0; i--)

{

destCoord.X = 5 + 24;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][5];

cout.flush();

Sleep(50);

}

{

destCoord.X = 5 + 28;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << "";

cout.flush();

Sleep(50);

}

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

system("pause");

}

{//часть С

system("cls");

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

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

swap(a[i][j], a[i][j + (N / 2)]);

}

}

//вывод матрицы

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

{

destCoord.X = 0 + 4;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][0];

cout.flush();

Sleep(50);

}

for (int i = 5; i >= 0; i--)

{

destCoord.X = 1 + 8;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][1];

cout.flush();

Sleep(50);

}

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

{

destCoord.X = 2 + 12;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][2];

cout.flush();

Sleep(50);

}

for (int i = 5; i >= 0; i--)

{

destCoord.X = 3 + 16;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][3];

cout.flush();

Sleep(50);

}

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

{

destCoord.X = 4 + 20;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][4];

cout.flush();

Sleep(50);

}

for (int i = 5; i >= 0; i--)

{

destCoord.X = 5 + 24;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][5];

cout.flush();

Sleep(50);

}

{

destCoord.X = 5 + 28;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << "";

cout.flush();

Sleep(50);

}

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

system("pause");

}

{//часть D

system("cls");

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

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

swap(a[i][j], a[i + (N / 2)][j]);

}

}

//вывод матрицы

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

{

destCoord.X = 0 + 4;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][0];

cout.flush();

Sleep(50);

}

for (int i = 5; i >= 0; i--)

{

destCoord.X = 1 + 8;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][1];

cout.flush();

Sleep(50);

}

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

{

destCoord.X = 2 + 12;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][2];

cout.flush();

Sleep(50);

}

for (int i = 5; i >= 0; i--)

{

destCoord.X = 3 + 16;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][3];

cout.flush();

Sleep(50);

}

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

{

destCoord.X = 4 + 20;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][4];

cout.flush();

Sleep(50);

}

for (int i = 5; i >= 0; i--)

{

destCoord.X = 5 + 24;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][5];

cout.flush();

Sleep(50);

}

{

destCoord.X = 5 + 28;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << "";

cout.flush();

Sleep(50);

}

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

system("pause");

}

{

}

{//задание 4

{

int value0;

string znak;

Menu: {

cout << "Введите действие ('+', '-', '\*', '/')\n";

cin >> znak;

cout << "Введите значение)\n";

cin >> value0;

}

if (znak == "+") {

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

for (int j = 0; j < N; j++)

a[i][j] += value0;

}

else if (znak == "-") {

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

for (int j = 0; j < N; j++)

a[i][j] -= value0;

}

else if (znak == "\*") {

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

for (int j = 0; j < N; j++)

a[i][j] \*= value0;

}

else if (znak == "/") {

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

for (int j = 0; j < N; j++)

a[i][j] /= value0;

}

else {

goto Menu;

}

system("cls");

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

{

destCoord.X = 0 + 4;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][0];

cout.flush();

Sleep(50);

}

for (int i = 5; i >= 0; i--)

{

destCoord.X = 1 + 8;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][1];

cout.flush();

Sleep(50);

}

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

{

destCoord.X = 2 + 12;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][2];

cout.flush();

Sleep(50);

}

for (int i = 5; i >= 0; i--)

{

destCoord.X = 3 + 16;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][3];

cout.flush();

Sleep(50);

}

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

{

destCoord.X = 4 + 20;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][4];

cout.flush();

Sleep(50);

}

for (int i = 5; i >= 0; i--)

{

destCoord.X = 5 + 24;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << a[i][5];

cout.flush();

Sleep(50);

}

{

destCoord.X = 5 + 28;

destCoord.Y = i;

SetConsoleCursorPosition(hStdout, destCoord);

cout << "";

cout.flush();

Sleep(50);

}

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

system("pause");

cout << "\n";

}

}

{

cout << " Спасибо за внимание";

}

return 0;

}