**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.**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 2373 |  | Басова К. В. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Создание и сортировка двумерного массива, заполнение квадратичной матрицы с помощью указателей, изменение и перестановка элементов квадратичной матрицы.

**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.

Память представляет собой непрерывную последовательность ячеек (байтов), каждая из которых имеет свой номер – адрес. Для получения адреса какого-либо программного объекта используется оператор **&**.

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 \*.

Например:

int \*p1;

double \*p2;

Переменная p1 предназначена для хранения адресов участков памяти, размер которых соответствует типу int (4 байта), а переменная p2 - типу double (8 байт).

Формально указатели представляют собой обычные целые значения типа int и занимают в памяти 4 байта не зависимо от базового типа указателя. Значения указателей при их выводе на экран представляются как целые значения в шестнадцатеричном формате.

Указателю можно присвоить либо адрес объекта того же типа, либо значение другого указателя.

int a = 1917;

int \*pa = &a; // указатель pa хранит адрес переменной a

При этом указатель и переменная должны иметь один и тот же тип, в данном случае это тип int.

Когда указателю присваивается другой указатель, то фактически первый указатель начинает также указывать на тот же адрес, на который указывает второй указатель.

Нулевой указатель (null pointer) - это указатель, который не указывает ни на какой объект. Для создания нулевого указателя можно применять различные способы:

int \*p1 = nullptr;

int \*p2 = NULL;

int \*p3 = 0;

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:

Операция разыменования указателя представляет выражение в виде \*имя\_указателя. Эта операция позволяет получить объект по адресу, который хранится в указателе.

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.

Иногда требуется присвоить указателю одного типа значение указателя другого типа. В этом случае следует выполнить операцию приведения типов с помощью операции (тип\_указателя \*). Для преобразования указателя к другому типу в скобках перед указателем ставится тип, к которому надо преобразовать. Причем если мы не можем просто создать объект, например, переменную типа void, то для указателя это вполне будет работать. То есть можно создать указатель типа void.

К указателям можно применять некоторые арифметические операции. К таким операциям относятся:  **+**,**-**, **++**, **--**. Добавлять к указателям или вычитать из указателей можно только целые значения.

Кода мы определяем в программе некоторый массив, например,

int Arr[10];

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

cout << Arr:

мы увидим некоторое целое значение в шестнадцатеричном формате, соответствующее адресу первого элемента этого массива.

Указателю, имеющему такой же базовый тип, как и элементы массива, можно присвоить массив следующим образом:

int Arr[10];

int \*p;

p = Arr;

Переменная массива является указателем на первый элемент массива. Чтобы получить 5–й элемент массива Arr, можно воспользоваться одним из следующих выражений:

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

Поскольку указатель и имя массива, в большой степени, взаимозаменяемы, указатели можно индексировать, как обычные массивы:

int A[10],  \*P  =  A;

for (int i = 0; i < 10; ++ i)

cout << P[i] << “  “;

**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.

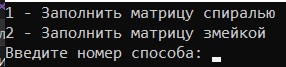
4)  Уменьшает, увеличивает, умножает или делит все элементы матрицы на введенное пользователем число.

**Выполнение работы.**

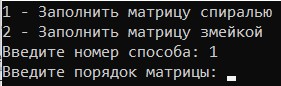
Код программы представлен в приложении А.

Описание кода и использованных алгоритмов.

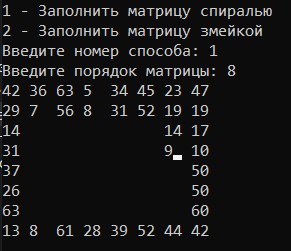
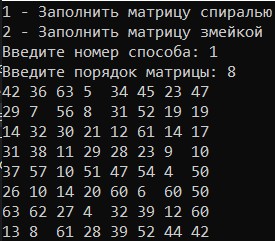
1. При запуске программы пользователю предлагается ввести номер способа, с помощью которого будет заполняться квадратичная матрица.



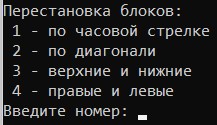
2. После ввода номера пользователь определяет порядок матрицы.



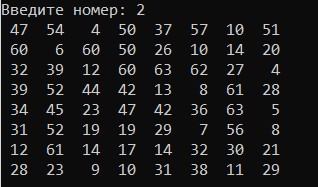
3. Матрица заполняется выбранным способом. Процесс заполнения выводится на экран.

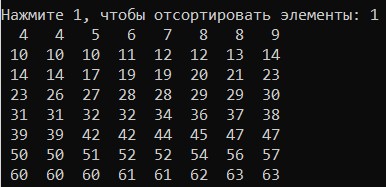
4. Пользователю предлагается выбрать способ перестановки блоков матрицы.



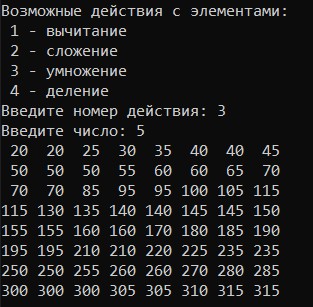
5. Блоки матрицы переставляются в соответствии с выбранным способом. Полученная матрица выводится на экран.



6. Чтобы отсортировать элементы матрицы, необходимо ввести 1. Отсортированный массив выводится на экран.



7. Пользователь выбирает одно из возможных действий над элементами и вводит число, с которым оно будет выполняться. Изменённый массив выводится на экран.



**Выводы.**

Использование указателей позволяет задать двумерный массив, размерность которого может вводиться пользователем с клавиатуры. Работа с таким массивом производится так же, как и с обычным массивом, размерность которого определяется константой.

Одним из способов сортировки двумерного массива является создание одномерного массива, элементам которого построчно присваиваются значения элементов двумерного массива. Сортировка такого массива производится одним из уже известных способов.

**Приложение А**

**код Программы**

#include <iostream>

#include <cstring>

#include <iomanip>

#include <Windows.h>

using namespace std;

int main() {

setlocale(0, "Rus");

int swap;

cout << "1 - Заполнить матрицу спиралью\n";

cout << "2 - Заполнить матрицу змейкой\n";

cout << "Введите номер способа: ";

int k, N;

cin >> k;

cout << "Введите порядок матрицы: ";

cin >> N;

int\*\* arr = new int\*[N];

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

arr[i] = new int[N];

int cx = 0, cy = 4;

int max\_i = N - 1, max\_j = N - 1;

int min\_i = 0, min\_j = 0;

HANDLE hStdout;

COORD destCoord;

hStdout = GetStdHandle(STD\_OUTPUT\_HANDLE);

if (k == 1) {

while (min\_i < max\_i) {

for (int i = min\_i; i <= max\_i; i++) {

int j = min\_j;

destCoord.X = cx;

destCoord.Y = cy;

SetConsoleCursorPosition(hStdout, destCoord);

arr[j][i] = (rand() % (N \* N)) + 1;

cout << arr[j][i];

cout.flush();

Sleep(100);

if (i != max\_i)

cx += 3;

else cy += 1;

}

for (int j = min\_j + 1; j <= max\_j; j++) {

int i = max\_i;

destCoord.X = cx;

destCoord.Y = cy;

SetConsoleCursorPosition(hStdout, destCoord);

arr[j][i] = (rand() % (N \* N)) + 1;

cout << arr[j][i];

cout.flush();

Sleep(100);

if (j != max\_j)

cy += 1;

else

cx -= 3;

}

for (int i = max\_i - 1; i >= min\_i; i--) {

int j = max\_j;

destCoord.X = cx;

destCoord.Y = cy;

SetConsoleCursorPosition(hStdout, destCoord);

arr[j][i] = (rand() % (N \* N)) + 1;

cout << arr[j][i];

cout.flush();

Sleep(100);

if (i != min\_i)

cx -= 3;

else cy -= 1;

}

for (int j = max\_j - 1; j > min\_j; j--) {

int i = min\_i;

destCoord.X = cx;

destCoord.Y = cy;

SetConsoleCursorPosition(hStdout, destCoord);

arr[j][i] = (rand() % (N \* N)) + 1;

cout << arr[j][i];

cout.flush();

Sleep(100);

if (j != min\_j + 1)

cy -= 1;

else cx += 3;

}

min\_i++;

min\_j++;

max\_i--;

max\_j--;

}

}

if (k == 2) {

int i = 0;

while (i < N) {

for (int j = min\_j; j <= max\_j; j++) {

destCoord.X = cx;

destCoord.Y = cy;

SetConsoleCursorPosition(hStdout, destCoord);

arr[j][i] = (rand() % (N \* N)) + 1;

cout << arr[j][i];

cout.flush();

Sleep(100);

if (j != max\_j)

cy += 1;

else

cx += 3;

}

i++;

for (int j = max\_j; j >= min\_j; j--) {

destCoord.X = cx;

destCoord.Y = cy;

SetConsoleCursorPosition(hStdout, destCoord);

arr[j][i] = (rand() % (N \* N)) + 1;

cout << arr[j][i];

cout.flush();

Sleep(100);

if (j != min\_j)

cy -= 1;

else

cx += 3;

}

i++;

}

}

destCoord.X = 0;

destCoord.Y = N + 4;

SetConsoleCursorPosition(hStdout, destCoord);

cout << "Перестановка блоков:\n ";

cout << "1 - по часовой стрелке\n 2 - по диагонали\n 3 - верхние и нижние\n 4 - правые и левые\n";

cout << "Введите номер: ";

int k\_per;

cin >> k\_per;

if (k\_per == 1) {

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

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

swap = arr[i][j];

arr[i][j] = arr[i + N / 2][j];

arr[i + N / 2][j] = swap;

swap = arr[i + N / 2][j];

arr[i + N / 2][j] = arr[i + N / 2][j + N / 2];

arr[i + N / 2][j + N / 2] = swap;

swap = arr[i + N / 2][j + N / 2];

arr[i + N / 2][j + N / 2] = arr[i][j + N / 2];

arr[i][j + N / 2] = swap;

}

}

}

if (k\_per == 2) {

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

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

swap = arr[i][j];

arr[i][j] = arr[N / 2 + i][N / 2 + j];

arr[N / 2 + i][N / 2 + j] = swap;

}

}

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

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

swap = arr[i][j];

arr[i][j] = arr[i - N / 2][N / 2 + j];

arr[i - N / 2][N / 2 + j] = swap;

}

}

}

if (k\_per == 3) {

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

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

swap = arr[i][j];

arr[i][j] = arr[i + N / 2][j];

arr[i + N / 2][j] = swap;

swap = arr[i + N / 2][j + N / 2];

arr[i + N / 2][j + N / 2] = arr[i][j + N / 2];

arr[i][j + N / 2] = swap;

}

}

}

if (k\_per == 4) {

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

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

swap = arr[i][j];

arr[i][j] = arr[i][j + N / 2];

arr[i][j + N / 2] = swap;

swap = arr[i + N / 2][j + N / 2];

arr[i + N / 2][j + N / 2] = arr[i + N / 2][j];

arr[i + N / 2][j] = swap;

}

}

}

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

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

cout.width(3);

cout << arr[m][l] << " ";

}

cout << "\n";

}

cout << "\nНажмите 1, чтобы отсортировать элементы: ";

int k\_sort;

cin >> k\_sort;

int\* new\_arr = new int[N \* N];

int count = 0;

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

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

new\_arr[count] = arr[i][j];

count++;

}

}

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

for (int j = i; j > 0; j--) {

if (new\_arr[j - 1] > new\_arr[j]) {

swap = new\_arr[j - 1];

new\_arr[j - 1] = new\_arr[j];

new\_arr[j] = swap;

}

}

}

count = 0;

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

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

arr[i][j] = new\_arr[count];

count++;

}

}

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

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

cout.width(3);

cout << arr[m][l] << " ";

}

cout << "\n";

}

cout << "Возможные действия с элементами: ";

cout << "\n 1 - вычитание\n 2 - сложение\n 3 - умножение\n 4 - деление\n";

cout << "Введите номер действия: ";

int op\_n;

int num;

cin >> op\_n;

cout << "Введите число: ";

cin >> num;

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

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

if (op\_n == 1)

arr[i][j] -= num;

else if (op\_n == 2)

arr[i][j] += num;

else if (op\_n == 3)

arr[i][j] \*= num;

else

arr[i][j] /= num;

}

}

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

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

cout.width(3);

cout << arr[m][l] << " ";

}

cout << "\n";

}

return 0;

}