**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ИС**

**отчет**

**по практической работе №3**

**по дисциплине «Программирование»**

**Тема: Двумерные статические массивы. Указатели**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 2373 |  | Суднищиков М.К. |
| Преподаватель |  | Глущенко А.Г. |

Санкт-Петербург

2022

**Цель работы.**

Познакомиться с двумерными статическими массивами, их объявлением, научимся обрабатывать данные двумерных статических массивов с помощью арифметики указателей, применим сортировку массива, реализованную с указателями. Написать программу, создающую матрицу размера N = (6, 8, 10) и заполняющую её рандомными значениями по заданным схемам, меняющую блоки матрицы местами, увеличивающую, уменьшающую, умножающую и делящею элементы матрицы на введенное пользователем число.

**Основные теоретические положения.**

Массив – это индексированная последовательность однотипных элементов с заранее определенным количеством элементов. Массивы используются для обработки большого количества однотипных данных.

Имя массива является указателем.

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

Многомерный массив – это массив, характеризованный двумя или более измерениями, причем доступ к каждому элементу такого массива осуществляется с помощью определенной комбинации двух или более индексов.

Двумерный массив – форма многомерного массива, элементы определяются двумя индексами.

Указатель – переменная служащая для хранения адреса.

Указатель определяется:

<тип данных> \*<имя переменной>

Тип данных указателя должен совпадать с типом данных объекта, на который он указывает. Значения указателей при их выводе на экран представляются как целые значения в шестнадцатеричном формате.

К указателям можно применять некоторые арифметические операции. К таким операциям относятся: сложение, вычитание, инкремент и декремент (+, -, ++, --). Результат применения этих операций к указателям отличается от результата применения их к разыменованным указателям или числам, так как изменяет не значение, а адрес.

**Постановка задачи.**

Необходимо разработать алгоритм и написать программу, которая:

1) Используя арифметику указателей, заполняет квадратичную целочисленную матрицу порядка N случайными числами от 1 до N\*N по столбцам попеременно по направлению вниз вверх и по спирали от левого верхнего края к центру. Пользователь должен видеть процесс заполнения квадратичной матрицы.

2) Получает новую матрицу, переставляя блоки исходной матрицы.

3) Используя арифметику указателей, сортирует элементы матрицы.

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

**Выполнение работы.**

Для выполнения поставленной задачи работы использован язык программирования С++. Итоговый код программы представлен в приложении.

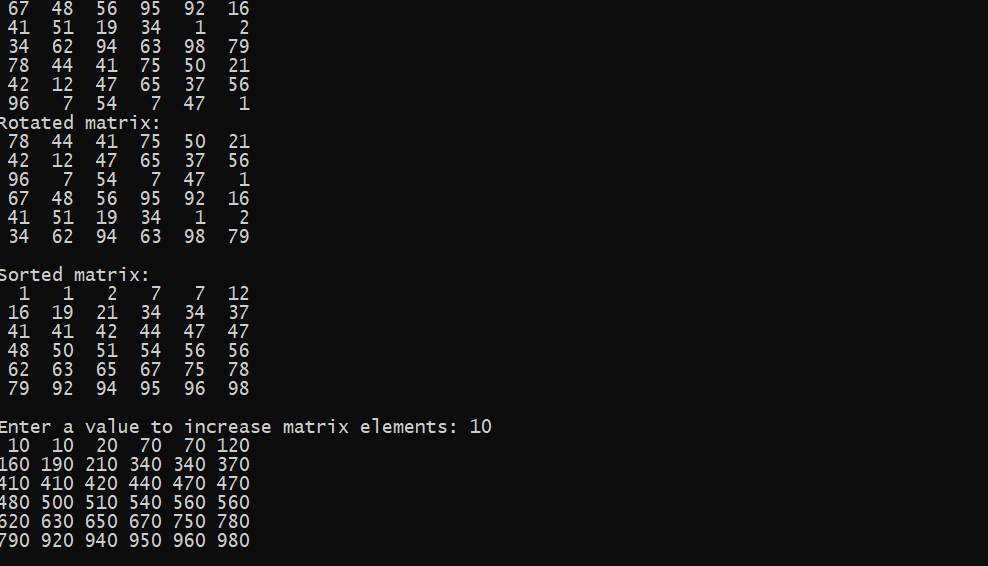
Для создания массива случайных значений в заданном диапазоне используем функцию srand().

Для вывода массива в консоль соблюдая правильный порядок значений подключаем библиотеку <Windows.h>, используем функцию, управляющую курсором.

Для выполнения заданий используем арифметику указателей, передаём указатель на первый элемент массива и остальные необходимые в конкретном задании значения.

**Блок скриншотов работы программы**





**Выводы.**

В процессе работы было получено представление о работе с двумерными статическими массивами, сортировками, с использованием арифметики указателей. Написана программа, заполняющая двумя способами случайными элементами двумерный статический массив, отображающая в консоли процесс заполнения массива поэтапно, изменяющая этот массив.

Приложение А

рабочий код

#include <iostream>

#include <thread>

#include <iomanip>

#include <Windows.h>

#include <ctime>

using namespace std;

void printMatrix(int\*\*, int, bool = 1);

void printMatrix(int\*\* matrix, int N, bool zero)

{

if (zero) system("cls");

for (int\*\* elemJ = matrix, \*\*lastJ = elemJ + N - 1; elemJ <= lastJ; elemJ++)

{

for (int\* elemI = \*elemJ, \*lastI = elemI + N - 1; elemI <= lastI; elemI++)

{

cout << setw(3) << \*elemI;

if (elemI != lastI) cout << " ";

}

cout << "\n";

}

std::this\_thread::sleep\_for(0.1s);

}

void zeroMatrix(int\*\* matrix, int N) {

for (int\*\* elemJ = matrix, \*\*lastJ = elemJ + N - 1; elemJ <= lastJ; elemJ++)

{

for (int\* elemI = \*elemJ, \*lastI = elemI + N - 1; elemI <= lastI; elemI++)

{

\*elemI = 0;

}

}

}

void snakeMatrix(int\*\* matrix, int N) {

printMatrix(matrix, N);

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

for (int\*\* elemJ = matrix, \*\*lastJ = matrix + N - 1; elemJ <= lastJ; elemJ++)

{

\*(\*elemJ + lineNum \* 2) = rand() % 101;

printMatrix(matrix, N);

}

for (int\*\* elemJ = matrix + N - 1, \*\*lastJ = matrix; elemJ >= lastJ; elemJ--)

{

\*(\*elemJ + lineNum \* 2 + 1) = rand() % 101;

printMatrix(matrix, N);

}

}

}

void rotateMatrix(int\*\* matrix, int N) {

int\*\* a = new int\* [N / 2], \*\* b = new int\* [N / 2], \*\* c = new int\* [N / 2], \*\* d = new int\* [N / 2], \*\*\* arr = new int\*\* [4];

arr[0] = a; arr[1] = b; arr[2] = c; arr[3] = d;

for (int\*\*\* i = arr, \*\*\*ilast = arr + 3; i <= ilast; i++)

{

for (int\*\* elem = \*i, \*\*last = elem + N / 2 - 1; elem <= last; elem++)

{

\*elem = new int[N / 2];

}

}

for (int\*\* elemJ = matrix, \*\*lastJ = elemJ + N / 2 - 1, \*\*elemArray = a; elemJ <= lastJ; elemJ++, elemArray++)

{

for (int\* elemI = \*elemJ, \*lastI = elemI + N / 2 - 1, \*elemArrayJ = \*elemArray; elemI <= lastI; elemI++, elemArrayJ++)

{

\*elemArrayJ = \*elemI;

}

}

for (int\*\* elemJ = matrix, \*\*lastJ = elemJ + N / 2 - 1, \*\*elemArray = b; elemJ <= lastJ; elemJ++, elemArray++)

{

for (int\* elemI = \*elemJ + N / 2, \*lastI = elemI + N / 2 - 1, \*elemArrayJ = \*elemArray; elemI <= lastI; elemI++, elemArrayJ++)

{

\*elemArrayJ = \*elemI;

}

}

for (int\*\* elemJ = matrix + N / 2, \*\*lastJ = elemJ + N / 2 - 1, \*\*elemArray = c; elemJ <= lastJ; elemJ++, elemArray++)

{

for (int\* elemI = \*elemJ, \*lastI = elemI + N / 2 - 1, \*elemArrayJ = \*elemArray; elemI <= lastI; elemI++, elemArrayJ++)

{

\*elemArrayJ = \*elemI;

}

}

for (int\*\* elemJ = matrix + N / 2, \*\*lastJ = elemJ + N / 2 - 1, \*\*elemArray = d; elemJ <= lastJ; elemJ++, elemArray++)

{

for (int\* elemI = \*elemJ + N / 2, \*lastI = elemI + N / 2 - 1, \*elemArrayJ = \*elemArray; elemI <= lastI; elemI++, elemArrayJ++)

{

\*elemArrayJ = \*elemI;

}

}

int\*\* pa, \*\* pb, \*\* pc, \*\* pd;

pa = c; pb = d; pc = a; pd = b;

for (int\*\* elemJ = matrix, \*\*lastJ = elemJ + N / 2 - 1, \*\*elemArray = pa; elemJ <= lastJ; elemJ++, elemArray++)

{

for (int\* elemI = \*elemJ, \*lastI = elemI + N / 2 - 1, \*elemArrayJ = \*elemArray; elemI <= lastI; elemI++, elemArrayJ++)

{

\*elemI = \*elemArrayJ;

}

}

for (int\*\* elemJ = matrix, \*\*lastJ = elemJ + N / 2 - 1, \*\*elemArray = pb; elemJ <= lastJ; elemJ++, elemArray++)

{

for (int\* elemI = \*elemJ + N / 2, \*lastI = elemI + N / 2 - 1, \*elemArrayJ = \*elemArray; elemI <= lastI; elemI++, elemArrayJ++)

{

\*elemI = \*elemArrayJ;

}

}

for (int\*\* elemJ = matrix + N / 2, \*\*lastJ = elemJ + N / 2 - 1, \*\*elemArray = pc; elemJ <= lastJ; elemJ++, elemArray++)

{

for (int\* elemI = \*elemJ, \*lastI = elemI + N / 2 - 1, \*elemArrayJ = \*elemArray; elemI <= lastI; elemI++, elemArrayJ++)

{

\*elemI = \*elemArrayJ;

}

}

for (int\*\* elemJ = matrix + N / 2, \*\*lastJ = elemJ + N / 2 - 1, \*\*elemArray = pd; elemJ <= lastJ; elemJ++, elemArray++)

{

for (int\* elemI = \*elemJ + N / 2, \*lastI = elemI + N / 2 - 1, \*elemArrayJ = \*elemArray; elemI <= lastI; elemI++, elemArrayJ++)

{

\*elemI = \*elemArrayJ;

}

}

printMatrix(matrix, N, 0);

}

void sortMatrix(int\*\* matrix, int N) {

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

int i = 0;

for (int\*\* elemJ = matrix, \*\*lastJ = elemJ + N - 1; elemJ <= lastJ; elemJ++)

{

for (int\* elemI = \*elemJ, \*lastI = elemI + N - 1; elemI <= lastI; elemI++)

{

\*(arr + i) = \*elemI;

i++;

}

}

bool reverse = 1;

while (reverse) {

reverse = 0;

for (int\* j = arr, \*jlast = j + (N \* N) - 2; j <= jlast; j++)

{

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

{

reverse = 1;

int t = \*(j + 1);

\*(j + 1) = \*j;

\*j = t;

}

}

}

i = 0;

for (int\*\* elemJ = matrix, \*\*lastJ = elemJ + N - 1; elemJ <= lastJ; elemJ++)

{

for (int\* elemI = \*elemJ, \*lastI = elemI + N - 1; elemI <= lastI; elemI++)

{

\*elemI = \*(arr + i);

i++;

}

}

printMatrix(matrix, N, 0);

}

void increaseMatrix(int\*\* matrix, int N, int k) {

for (int\*\* elemJ = matrix, \*\*lastJ = elemJ + N - 1; elemJ <= lastJ; elemJ++)

{

for (int\* elemI = \*elemJ, \*lastI = elemI + N - 1; elemI <= lastI; elemI++)

{

\*elemI \*= k;

}

}

printMatrix(matrix, N, 0);

}

int main()

{

srand(time(NULL));

int N;

cout << "Enter N(6, 8 or 10):" << "\n";

cin >> N;

while (N != 6 && N != 8 && N != 10) {

cout << "Wrong N value, try again" << "\n";

cin >> N;

}

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

for (int\*\* elem = matrix, \*\*last = elem + N - 1; elem <= last; elem++)

{

\*elem = new int[N];

}

zeroMatrix(matrix, N);

snakeMatrix(matrix, N);

cout << "Rotated matrix: " << '\n';

rotateMatrix(matrix, N);

cout << '\n' << "Sorted matrix: " << '\n';

sortMatrix(matrix, N);

cout << '\n' << "Enter a value to increase matrix elements: ";

int k;

cin >> k;

increaseMatrix(matrix, N, k);

}