**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра «Информационные системы»**

отчет

**по практической работе №3**

**по дисциплине «Программирование»**

Тема: указатели и многомерные статические массивы

|  |  |  |
| --- | --- | --- |
| Студент гр. 0323 |  | Балашевич К.Д. |
| Преподаватель |  | Глущенко А.Г. |

Санкт-Петербург

2020

**Цель работы.**

Изучение структуры многомерных статических массивов, обработка данных многомерных массивов. Получение практических навыков работы с указателями. Изучение простейшей арифметики указателей.

**Основные теоретические положения.**

В языке c++ возможна обращение к данным в памяти как по имени переменной, так и по адресу в памяти, по которому эти данные располагаются. Для упрощения работы с адресами предусмотрен особый тип переменных, называемый указателем. Такая переменная хранит адрес в памяти, по которому располагаются некоторые данные.

В памяти ЭВМ элементы массивов (в языке c++) располагаются последовательно. Многомерные же массивы представляются в памяти ЭВМ как массивы массивов, т.е. в виде непрерывной последовательности. Объявление таких массивов производится с использованием записи размера каждого измерения в отдельных квадратных скобках.

Подобное расположение массивов в памяти позволяет работать с массивами зная только адрес некоторого элемента, индекс этого элемента в массиве и размер массива. На практике в большинстве случаев используют первый элемент массива.

Удобно использовать и ту особенность указателей, что при прибавлении единицы к значению указателя, оно увеличивается на размер типа данных, который использовался при объявлении указателя.

Учитывая вышеизложенное, работа с элементом массива несколько упрощается при использовании указателя на массив и увеличения его значения на индекс элемента.

**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.



1. Получает новую матрицу, из матрицы п. 1, переставляя ее блоки в соответствии со схемами:



1. Используя арифметику указателей, сортирует элементы любой сортировкой.
2. Уменьшает, увеличивает, умножает или делит все элементы матрицы на введенное пользователем число.

**Выполнение работы.**

Для решения поставленной задачи была написана программа на языке C++. Итоговый код программы представлен в приложении А.

Было проведено тестирование программы с использование различных компиляторов. Результаты тестирования представлены в приложении Б. При этом были получены результаты, соответствующие расчётным.

**Выводы.**

Использование указателей значительно упрощает работу с массивами.

Приложение А

Полный код программы

#include <iostream>

#include <windows.h>

#include <ctime>

using namespace std;

int mainMenu (int \*matrixPointer, int matrixRank);

void generateZeroMatrix (int \*matrixPointer, int matrixRank);

void generateMatrixA (int \*matrixPointer, int matrixRank);

void generateMatrixB (int \*matrixPointer, int matrixRank);

void printMatrix (int \*matrixPointer, int matrixRank);

void changeMatrixA (int \*matrixPointer, int matrixRank);

void changeMatrixB (int \*matrixPointer, int matrixRank);

void changeMatrixC (int \*matrixPointer, int matrixRank);

void changeMatrixD (int \*matrixPointer, int matrixRank);

void sortMatrix (int \*matrixPointer, int matrixRank);

int arithmeticMatrix (int \*matrixPointer, int matrixRank);

int main ()

{

int matrixRank;

int \*matrixPointer;

setlocale(LC\_ALL, "ru\_RU.utf8");

//setlocale(0, "");

srand(time(0));

cout << "Введите ранг матрицы: ";

cin >> matrixRank;

int Matrix[10][10];

matrixPointer=&Matrix[0][0];

generateZeroMatrix(matrixPointer, matrixRank);

int exit=1;

while (exit) exit=mainMenu (matrixPointer, matrixRank);

return 0;

}

int mainMenu (int \*matrixPointer, int matrixRank)

{

system("cls");

cout << "Текущее состояние матрицы:" << endl << endl;

printMatrix(matrixPointer, matrixRank);

cout << endl << "Главное меню:" << endl << "1) Заполнить матрицу случайными"

" числами по схеме a." << endl << "2) Заполнить матрицу случайными "

"числами по схеме b." << endl << "3) Преобразовать матрицу в соотве"

"тствии со схемой a." << endl << "4) Преобразовать матрицу в соотве"

"тствии со схемой b." << endl << "5) Преобразовать матрицу в соотве"

"тствии со схемой c." << endl << "6) Преобразовать матрицу в соотве"

"тствии со схемой d." << endl << "7) Отсортировать матрицу." << endl

<< "8) Произвести арифметические действия над матрицей." << endl <<

"0) Выход из программы." << endl << "Введите желаемый вариант: ";

int choise;

cin >> choise;

switch (choise)

{

case 0:

return 0;

case 1:

{

generateMatrixA(matrixPointer, matrixRank);

return 1;

}

case 2:

{

generateMatrixB(matrixPointer, matrixRank);

return 1;

}

case 3:

{

changeMatrixA(matrixPointer, matrixRank);

return 1;

}

case 4:

{

changeMatrixB(matrixPointer, matrixRank);

return 1;

}

case 5:

{

changeMatrixC(matrixPointer, matrixRank);

return 1;

}

case 6:

{

changeMatrixD(matrixPointer, matrixRank);

return 1;

}

case 7:

{

sortMatrix(matrixPointer, matrixRank);

return 1;

}

case 8:

{

int choise=1;

while (choise) choise=arithmeticMatrix(matrixPointer, matrixRank);

return 1;

}

default:

{

system("cls");

cout << "Некоррекный ввод!";

Sleep(1000);

return 1;

}

}

}

void generateZeroMatrix (int \*matrixPointer, int matrixRank)

{

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

}

void generateMatrixA (int \*matrixPointer, int matrixRank)

{

system("cls");

for (int stop=0; stop<matrixRank/2; stop++)

{

for (int i=0; i<matrixRank-2\*stop; i++)

{

\*(matrixPointer+((matrixRank+1)\*stop)+i)=rand()%(matrixRank\*matrixRank)+1;

Sleep(500);

system("cls");

cout << "Матрица заполняется:" << endl;

printMatrix(matrixPointer, matrixRank);

cout << endl;

}

for (int i=1+stop; i<matrixRank-stop; i++)

{

\*(matrixPointer+(i+1)\*matrixRank-stop-1)=rand()%(matrixRank\*matrixRank)+1;

Sleep(500);

system("cls");

cout << "Матрица заполняется:" << endl;

printMatrix(matrixPointer, matrixRank);

cout << endl;

}

for (int i=matrixRank-1; i>2\*stop; i--)

{

\*(matrixPointer+(matrixRank-1-stop)\*matrixRank+i-stop-1)=rand()%(matrixRank\*matrixRank)+1;

Sleep(500);

system("cls");

cout << "Матрица заполняется:" << endl;

printMatrix(matrixPointer, matrixRank);

cout << endl;

}

for (int i=matrixRank-2-stop; i>stop; i--)

{

\*(matrixPointer+matrixRank\*i+stop)=rand()%(matrixRank\*matrixRank)+1;

Sleep(500);

system("cls");

cout << "Матрица заполняется:" << endl;

printMatrix(matrixPointer, matrixRank);

cout << endl;

}

}

system("cls");

cout << "Выполнено!";

Sleep(1000);

}

void generateMatrixB (int \*matrixPointer, int matrixRank)

{

system("cls");

for (int column=0; column<matrixRank; column++)

{

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

{

\*(matrixPointer+matrixRank\*i+column)=rand()%(matrixRank\*matrixRank)+1;

Sleep(500);

system("cls");

cout << "Матрица заполняется:" << endl;

printMatrix(matrixPointer, matrixRank);

cout << endl;

}

column++;

for (int i=matrixRank-1; i>-1; i--)

{

\*(matrixPointer+matrixRank\*i+column)=rand()%(matrixRank\*matrixRank)+1;

Sleep(500);

system("cls");

cout << "Матрица заполняется:" << endl;

printMatrix(matrixPointer, matrixRank);

cout << endl;

}

}

system("cls");

cout << "Выполнено!";

Sleep(1000);

}

void printMatrix (int \*matrixPointer, int matrixRank)

{

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

{

cout << \*(matrixPointer+i) << "\t";

if (i%matrixRank==matrixRank-1) cout << endl;

}

}

void changeMatrixA (int \*matrixPointer, int matrixRank)

{

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

{

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

{

\*(matrixPointer+i+matrixRank\*j) = \*(matrixPointer+i+matrixRank\*j) + \*(matrixPointer+i+matrixRank/2+matrixRank\*j);

\*(matrixPointer+i+matrixRank/2+matrixRank\*j) = \*(matrixPointer+i+matrixRank\*j) - \*(matrixPointer+i+matrixRank/2+matrixRank\*j);

\*(matrixPointer+i+matrixRank\*j) = \*(matrixPointer+i+matrixRank\*j) - \*(matrixPointer+i+matrixRank/2+matrixRank\*j);

}

}

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

{

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

{

\*(matrixPointer+i+matrixRank\*j) = \*(matrixPointer+i+matrixRank\*j) + \*(matrixPointer+i+matrixRank/2\*matrixRank+matrixRank\*j);

\*(matrixPointer+i+matrixRank/2\*matrixRank+matrixRank\*j) = \*(matrixPointer+i+matrixRank\*j) - \*(matrixPointer+i+matrixRank/2\*matrixRank+matrixRank\*j);

\*(matrixPointer+i+matrixRank\*j) = \*(matrixPointer+i+matrixRank\*j) - \*(matrixPointer+i+matrixRank/2\*matrixRank+matrixRank\*j);

}

}

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

{

for (int j=matrixRank/2; j<matrixRank; j++)

{

\*(matrixPointer+i+matrixRank\*j) = \*(matrixPointer+i+matrixRank\*j) + \*(matrixPointer+i+matrixRank/2+matrixRank\*j);

\*(matrixPointer+i+matrixRank/2+matrixRank\*j) = \*(matrixPointer+i+matrixRank\*j) - \*(matrixPointer+i+matrixRank/2+matrixRank\*j);

\*(matrixPointer+i+matrixRank\*j) = \*(matrixPointer+i+matrixRank\*j) - \*(matrixPointer+i+matrixRank/2+matrixRank\*j);

}

}

}

void changeMatrixB (int \*matrixPointer, int matrixRank)

{

int buffer;

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

{

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

{

buffer=\*(matrixPointer+i+matrixRank\*j);

\*(matrixPointer+i+matrixRank\*j)=\*(matrixPointer+i+matrixRank\*j+matrixRank/2\*matrixRank+matrixRank/2);

\*(matrixPointer+i+matrixRank\*j+matrixRank/2\*matrixRank+matrixRank/2)=buffer;

}

}

for (int i=matrixRank/2; i<matrixRank; i++)

{

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

{

buffer=\*(matrixPointer+i+matrixRank\*j);

\*(matrixPointer+i+matrixRank\*j)=\*(matrixPointer+i+matrixRank\*j+matrixRank/2\*matrixRank-matrixRank/2);

\*(matrixPointer+i+matrixRank\*j+matrixRank/2\*matrixRank-matrixRank/2)=buffer;

}

}

}

void changeMatrixC (int \*matrixPointer, int matrixRank)

{

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

{

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

{

int buffer;

buffer=\*(matrixPointer+i+matrixRank\*j);

\*(matrixPointer+i+matrixRank\*j)=\*(matrixPointer+i+matrixRank/2\*matrixRank+matrixRank\*j);

\*(matrixPointer+i+matrixRank/2\*matrixRank+matrixRank\*j)=buffer;

}

}

for (int i=matrixRank/2; i<matrixRank; i++)

{

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

{

int buffer;

buffer=\*(matrixPointer+i+matrixRank\*j);

\*(matrixPointer+i+matrixRank\*j)=\*(matrixPointer+i+matrixRank/2\*matrixRank+matrixRank\*j);

\*(matrixPointer+i+matrixRank/2\*matrixRank+matrixRank\*j)=buffer;

}

}

}

void changeMatrixD (int \*matrixPointer, int matrixRank)

{

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

{

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

{

int buffer;

buffer=\*(matrixPointer+i+matrixRank\*j);

\*(matrixPointer+i+matrixRank\*j)=\*(matrixPointer+i+matrixRank/2+matrixRank\*j);

\*(matrixPointer+i+matrixRank/2+matrixRank\*j)=buffer;

}

}

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

{

for (int j=matrixRank/2; j<matrixRank; j++)

{

int buffer;

buffer=\*(matrixPointer+i+matrixRank\*j);

\*(matrixPointer+i+matrixRank\*j)=\*(matrixPointer+i+matrixRank/2+matrixRank\*j);

\*(matrixPointer+i+matrixRank/2+matrixRank\*j)=buffer;

}

}

}

void sortMatrix (int \*matrixPointer, int matrixRank)

{

int sorted=0;

int currentMaxUnsorted=matrixRank\*matrixRank-1;

while (!sorted)

{

sorted=1;

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

{

if (\*(matrixPointer+i)-\*(matrixPointer+i+1)>0)

{

\*(matrixPointer+i) = \*(matrixPointer+i) + \*(matrixPointer+i+1);

\*(matrixPointer+i+1) = \*(matrixPointer+i) - \*(matrixPointer+i+1);

\*(matrixPointer+i) = \*(matrixPointer+i) - \*(matrixPointer+i+1);

sorted=0;

}

}

currentMaxUnsorted--;

}

}

int arithmeticMatrix (int \*matrixPointer, int matrixRank)

{

system("cls");

cout << "Текущее состояние матрицы:" << endl << endl;

printMatrix(matrixPointer, matrixRank);

cout << "Доступные действия:" << endl << "1) Вычитание." << endl << "2) Сло"

"жение." << endl << "3) Умножение." << endl << "4) Деление." << endl

<< "Чтобы выйти в главное меню введите 0." << endl << endl << "Введ"

"ите номер желаемоо действия: ";

int choise;

cin >> choise;

switch (choise)

{

case 1:

{

cout << "Введите число, которое вы хотите вычесть из матрицы: ";

int number;

cin >> number;

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

{

\*(matrixPointer+i)=\*(matrixPointer+i)-number;

}

return 1;

}

case 2:

{

cout << "Введите число, которое вы хотите прибавить к матрице: ";

int number;

cin >> number;

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

{

\*(matrixPointer+i)=\*(matrixPointer+i)+number;

}

return 1;

}

case 3:

{

cout << "Введите число, на которое вы хотите умножить матрицу: ";

int number;

cin >> number;

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

{

\*(matrixPointer+i)=\*(matrixPointer+i)\*number;

}

return 1;

}

case 4:

{

cout << "Введите число, на которое вы хотите разделить матрицу: ";

int number;

cin >> number;

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

{

\*(matrixPointer+i)=\*(matrixPointer+i)/number;

}

return 1;

}

default:

{

return 0;

}

}

}

Приложение Б

результаты тестирования

Тестирование программы осуществлялось на компьютере с процессором x64 под управлением ОС Windows 10 Pro (версия 2004, 64-бит) с использованием следующих компиляторов:

* Оптимизирующий компилятор Microsoft (R) C/C++ версии 19.16.27043 для x86

Результаты тестирования программы представлены в табл. 1.

Таблица 1 – Результаты тестирования программы

|  |  |
| --- | --- |
| Ожидаемый результат | Полученный результат |
| Матрица ранга 8, заполненная числами от 1 до 64 по спирали. | 60 54 54 34 8 6 39 34  11 49 30 45 44 21 24 8  50 43 21 37 54 49 10 63  30 61 21 19 61 40 55 19  27 20 26 20 3 61 23 31  61 40 52 7 43 62 46 13  5 54 23 22 42 52 52 62  26 34 14 7 2 51 20 44 |
| Матрица ранга 8, заполненная числами от 1 до 64 «змейкой». | 36 46 51 61 50 6 18 14  23 41 26 24 28 57 8 53  6 54 50 28 52 12 3 35  18 19 46 10 26 24 44 30  22 28 33 2 11 58 22 58  7 21 7 32 58 22 30 14  43 5 4 22 19 51 33 42  8 4 48 48 7 30 51 30 |
| 22 28 33 2 36 46 51 61  7 21 7 32 23 41 26 24  43 5 4 22 6 54 50 28  8 4 48 48 18 19 46 10  11 58 22 58 50 6 18 14  58 22 30 14 28 57 8 53  19 51 33 42 52 12 3 35  7 30 51 30 26 24 44 30 | 22 28 33 2 36 46 51 61  7 21 7 32 23 41 26 24  43 5 4 22 6 54 50 28  8 4 48 48 18 19 46 10  11 58 22 58 50 6 18 14  58 22 30 14 28 57 8 53  19 51 33 42 52 12 3 35  7 30 51 30 26 24 44 30 |
| 50 6 18 14 11 58 22 58  28 57 8 53 58 22 30 14  52 12 3 35 19 51 33 42  26 24 44 30 7 30 51 30  36 46 51 61 22 28 33 2  23 41 26 24 7 21 7 32  6 54 50 28 43 5 4 22  18 19 46 10 8 4 48 48 | 50 6 18 14 11 58 22 58  28 57 8 53 58 22 30 14  52 12 3 35 19 51 33 42  26 24 44 30 7 30 51 30  36 46 51 61 22 28 33 2  23 41 26 24 7 21 7 32  6 54 50 28 43 5 4 22  18 19 46 10 8 4 48 48 |
| 36 46 51 61 22 28 33 2  23 41 26 24 7 21 7 32  6 54 50 28 43 5 4 22  18 19 46 10 8 4 48 48  50 6 18 14 11 58 22 58  28 57 8 53 58 22 30 14  52 12 3 35 19 51 33 42  26 24 44 30 7 30 51 3 | 36 46 51 61 22 28 33 2  23 41 26 24 7 21 7 32  6 54 50 28 43 5 4 22  18 19 46 10 8 4 48 48  50 6 18 14 11 58 22 58  28 57 8 53 58 22 30 14  52 12 3 35 19 51 33 42  26 24 44 30 7 30 51 3 |
| 22 28 33 2 36 46 51 61  7 21 7 32 23 41 26 24  43 5 4 22 6 54 50 28  8 4 48 48 18 19 46 10  11 58 22 58 50 6 18 14  58 22 30 14 28 57 8 53  19 51 33 42 52 12 3 35  7 30 51 30 26 24 44 30 | 22 28 33 2 36 46 51 61  7 21 7 32 23 41 26 24  43 5 4 22 6 54 50 28  8 4 48 48 18 19 46 10  11 58 22 58 50 6 18 14  58 22 30 14 28 57 8 53  19 51 33 42 52 12 3 35  7 30 51 30 26 24 44 30 |
| 2 3 4 4 5 6 6 7  7 7 8 8 10 11 12 14  14 18 18 19 19 21 22 22  22 22 23 24 24 26 26 28  28 28 30 30 30 30 32 33  33 35 36 41 42 43 44 46  46 48 48 50 50 51 51 51  52 53 54 57 58 58 58 61 | 2 3 4 4 5 6 6 7  7 7 8 8 10 11 12 14  14 18 18 19 19 21 22 22  22 22 23 24 24 26 26 28  28 28 30 30 30 30 32 33  33 35 36 41 42 43 44 46  46 48 48 50 50 51 51 51  52 53 54 57 58 58 58 61 |
| Вычитание 2:  2 3 4 4 5 6 6 7  7 7 8 8 10 11 12 14  14 18 18 19 19 21 22 22  22 22 23 24 24 26 26 28  28 28 30 30 30 30 32 33  33 35 36 41 42 43 44 46  46 48 48 50 50 51 51 51  52 53 54 57 58 58 58 61 | Вычитание 2:  2 3 4 4 5 6 6 7  7 7 8 8 10 11 12 14  14 18 18 19 19 21 22 22  22 22 23 24 24 26 26 28  28 28 30 30 30 30 32 33  33 35 36 41 42 43 44 46  46 48 48 50 50 51 51 51  52 53 54 57 58 58 58 61 |
| Деление на 5:  0 0 0 0 0 0 0 1  1 1 1 1 1 1 2 2  2 3 3 3 3 3 4 4  4 4 4 4 4 4 4 5  5 5 5 5 5 5 6 6  6 6 6 7 8 8 8 8  8 9 9 9 9 9 9 9  10 10 10 11 11 11 11 11 | Деление на 5:  0 0 0 0 0 0 0 1  1 1 1 1 1 1 2 2  2 3 3 3 3 3 4 4  4 4 4 4 4 4 4 5  5 5 5 5 5 5 6 6  6 6 6 7 8 8 8 8  8 9 9 9 9 9 9 9  10 10 10 11 11 11 11 11 |
| Сложение -100:  -100 -100 -100 -100 -100 -100 -100 -99  -99 -99 -99 -99 -99 -99 -98 -98  -98 -97 -97 -97 -97 -97 -96 -96  -96 -96 -96 -96 -96 -96 -96 -95  -95 -95 -95 -95 -95 -95 -94 -94  -94 -94 -94 -93 -92 -92 -92 -92  -92 -91 -91 -91 -91 -91 -91 -91  -90 -90 -90 -89 -89 -89 -89 -89 | Сложение -100:  -100 -100 -100 -100 -100 -100 -100 -99  -99 -99 -99 -99 -99 -99 -98 -98  -98 -97 -97 -97 -97 -97 -96 -96  -96 -96 -96 -96 -96 -96 -96 -95  -95 -95 -95 -95 -95 -95 -94 -94  -94 -94 -94 -93 -92 -92 -92 -92  -92 -91 -91 -91 -91 -91 -91 -91  -90 -90 -90 -89 -89 -89 -89 -89 |
| Умножение на 0:  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 | Умножение на 0:  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 |