**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Информационных систем**

отчет

**по лабораторной работе №3**

**по дисциплине «Программирование»**

Тема: Указатели

|  |  |  |
| --- | --- | --- |
| Студент гр. 0323 |  | Кольцов К.Э |
| Преподаватель |  | Глущенко А.Г. |

Санкт-Петербург

2020

**Цель работы.**

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 (смю рис. 1), приведенным на рисунках. Пользователь должен видеть процесс заполнения квадратичной матрицы.



Рисунок 1 – Схемы заполнения матриц

2)    Получает новую матрицу, из матрицы п. 1, переставляя ее блоки в соответствии со схемами (см. рис. 2).



Рисунок 2 – Схемы перестановки матрицы по квадратам 2 на 2

3)    Используя арифметику указателей, сортирует элементы любой сортировкой.

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

**Основные теоретические положения.**

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.

Все массивы можно разделить на две группы: одномерные и многомерные. Описание массива в программе отличается от объявления обычной переменной наличием размерности массива, которая задается в квадратных скобках после имени.

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

Аналогом одномерного массива из математики может служить последовательность некоторых элементов с одним индексом: a\_i*ai*​ при  i = 0, 1, 2, … n – одномерный вектор. Каждый элемент такой последовательности представляет собой некоторое значение определенного типа данных.

Быстрая сортировка (quick sort) – одна из самых быстрых сортировок. Эта сортировка по сути является существенно улучшенной версией алгоритма пузырьковой сортировки.

Общая идея алгоритма состоит в том, что сначала выбирается из массива элемент, который называется опорным. От выбора опорного элемента не зависит корректность алгоритма, но в отдельных случаях может сильно зависеть его эффективность. Затем необходимо сравнить все остальные элементы с опорным и переставить их в массиве так, чтобы разбить массив на три непрерывных отрезка, следующие друг за другом: меньше опорного, раны опорному и больше опорного. Для меньших и больших значений необходимо выполнить рекурсивно ту же последовательность операций, если длина отрезка больше единицы.

На практике массив обычно делят на две части: «меньше опорного» и «равные и большие» или «меньше опорного или равные» и «большие». Такой поход в общем случае эффективнее, ведь упрощается алгоритм разделения.

Указатели хранят адрес в памяти компьютера, в котором находятся переменная, от типа переменной зависит размер этой памяти, так для указателя \*int размер 4, и при операции увеличении оператора int \*p = 140, операция p + 1 выдаст 144, а не 140. Для получения указателя используется оператор &, для доступа к данным указателя операция \*. Указатели являются переменными, то есть имеют адрес в памяти, поэтому можно создавать указатели на указатели и т.д.

**Экспериментальные результаты.**

Программа выполняет команды, которые пишет пользователь, всего команд 6:

h: список команд;

m: заново создать матрицу;

c: увеличить, уменьшить, умножить или поделить матрицу на число;

s: отсортировать матрицу, используя порядок создания;

p: перестановка элементов;

e: завершить работу программы.

Команда h выводит список команд, команда e завершает программу, остальные команды вызывают соответствующие методы в коде программы.

Команда m позволяет заново сгенерировать матрицу.

Команда с считывает оператор (+, -, \* или :) и затем число, для изменения всей матрицы.

Команда s сортируют матрицу быстрой сортировкой по схеме создания.

Команда p переставляет элементы матрицы по схемам квадратов 2 на 2.

Работу каждой функции можно просмотреть отдельно (см. рис. 3, 4, 5)

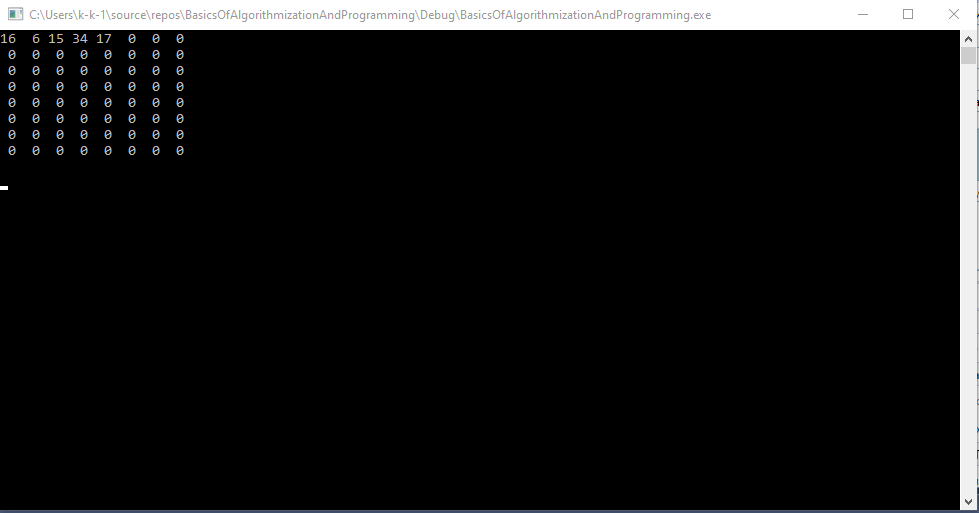


Рисунок 3 – Процесс заполнения матрицы

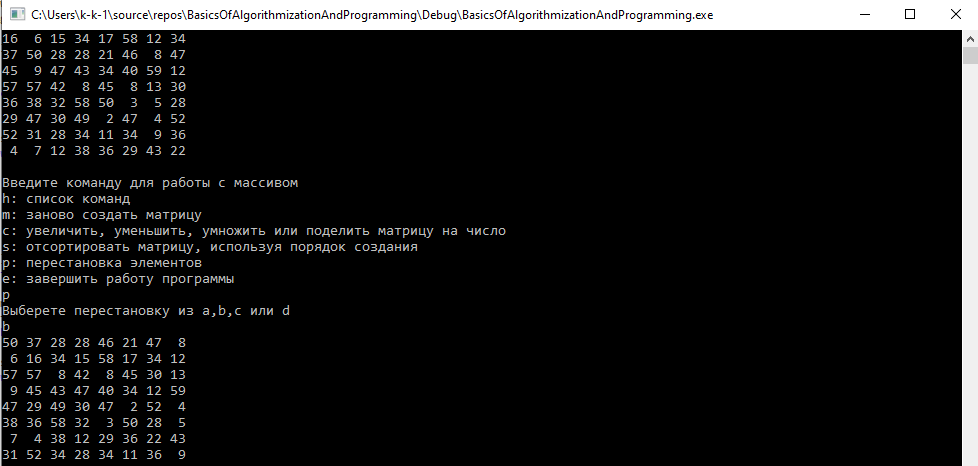


Рисунок 4 – Готовая матрица и перестановка элементов

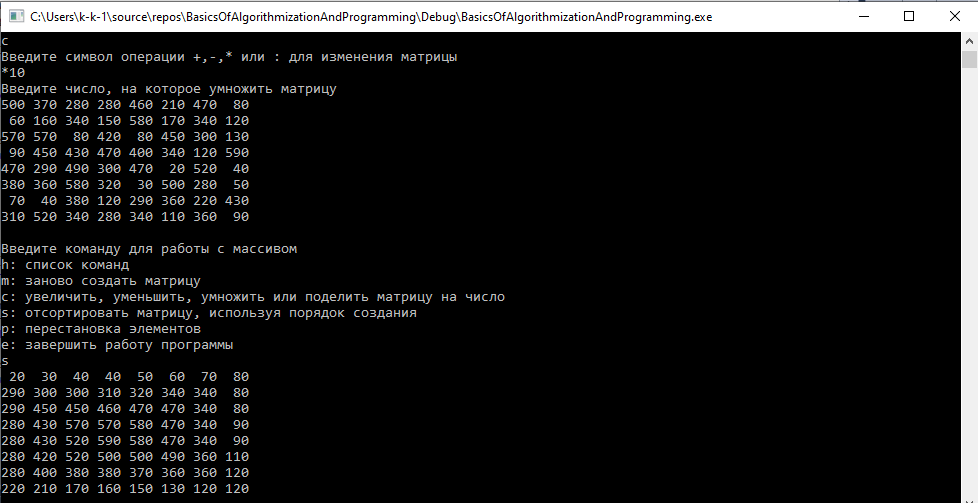


Рисунок 5 – Умножение матрицы на число и сортировка по схеме создания

**Обработка результатов эксперимента.**

Программа корректно работает с двумерным массивом с помощью указателей, сортирует и заполняет.

**Выводы.**

В ходе данной лабораторной работы я изучил указатели и их арифметику.

ПРИЛОЖЕНИЕ А

Листинг программы

﻿#include <iostream>

#include <iomanip>

#include <ctime>

#include <windows.h>

#define N 8

void printMatrix(int [N][N]);

void makeMatrixA(int[N][N], int\*[N\*N]);

void makeMatrixB(int[N][N], int\*[N\*N]);

void changeMatrixA(int[N][N]);

void changeMatrixB(int[N][N]);

void changeMatrixC(int[N][N]);

void changeMatrixD(int[N][N]);

void quicksort(int\*\*, int, int);

void userChangeMatrix(int[N][N]);

int maxValue;

int main()

{

std::srand(time(0));

setlocale(LC\_ALL, "Russian");

int \*sort[N\*N], mat[N][N];

char answer;

bool flag = true;

while (flag) {

std::cout << "Выберете вариант создания матрицы: a или b\n";

std::cin >> answer;

switch (answer)

{

case 'a':

makeMatrixA(mat, sort);

flag = false;

break;

case 'b':

makeMatrixB(mat, sort);

flag = false;

break;

default:

std::cout << "Вы выбрали несуществующую схему\n";

break;

}

}

flag = true;

char answer1;

bool flag1;

while (flag) {

std::cout << "Введите команду для работы с массивом\n" \

"h: список команд\n" \

"m: заново создать матрицу\n" \

"c: увеличить, уменьшить, умножить или поделить матрицу на число\n" \

"s: отсортировать матрицу, используя порядок создания\n" \

"p: перестановка элементов\n" \

"e: завершить работу программы\n";

std::cin >> answer;

switch (answer)

{

case 'h':

std::cout << "h: список команд\n" \

"m: заново создать матрицу\n" \

"c: увеличить, уменьшить, умножить или поделить матрицу на число\n" \

"s: отсортировать матрицу, используя порядок создания\n" \

"p: перестановка элементов\n" \

"e: завершить работу программы\n";

break;

case 'm':

flag1 = true;

while (flag1) {

std::cout << "Выберете вариант создания матрицы: a или b\n";

std::cin >> answer1;

switch (answer1)

{

case 'a':

makeMatrixA(mat, sort);

flag1 = false;

break;

case 'b':

makeMatrixB(mat, sort);

flag1 = false;

break;

default:

std::cout << "Вы выбрали несуществующую схему\n";

break;

}

}

break;

case 'c':

userChangeMatrix(mat);

break;

case 's':

quicksort(sort, 0, N\*N - 1);

printMatrix(mat);

break;

case 'p':

flag1 = true;

while (flag1) {

std::cout << "Выберете перестановку из a,b,c или d\n";

std::cin >> answer1;

switch (answer1)

{

case 'a':

changeMatrixA(mat);

flag1 = false;

break;

case 'b':

changeMatrixB(mat);

flag1 = false;

break;

case 'c':

changeMatrixC(mat);

flag1 = false;

break;

case 'd':

changeMatrixD(mat);

flag1 = false;

break;

default:

std::cout << "Вы выбрали несуществующую перестановку\n";

break;

}

}

break;

case'e':

flag = false;

break;

default:

std::cout << "Вы указали несуществующую команду\n";

break;

}

}

system("pause");

return 0;

}

void printMatrix(int mat[N][N]) {

int p = maxValue, d = 0;

while (p) {

d++;

p /= 10;

}

for (int \*i = mat[0];i < (mat[0] + N \* N);i += N) {

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

std::cout << std::setw(d) << \*j << ' ';

std::cout << '\n';

}

std::cout << '\n';

}

void makeMatrixA(int mat[N][N], int \*sort[N\*N]) {

maxValue = 0;

for (int \*i = mat[0];i < (mat[0] + N \* N);i += N) {

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

\*j = 0;

}

printMatrix(mat);

Sleep(250);

int count = 0;

for (int \*i[4] = { mat[0], (mat[0] + N - 1), (mat[0] + N \* N - 1), (mat[0] + N \* (N - 1)) };i[0] < i[1];i[0] += N + 1, i[1] += N - 1, i[2] -= N + 1, i[3] -= N - 1) {

for (int \*j = i[0];j < i[1];j++) {

\*j = (std::rand() % (N\*N)) + 1;

if (\*j > maxValue)

maxValue = \*j;

sort[count++] = j;

system("cls");

printMatrix(mat);

Sleep(250);

}

for (int \*j = i[1];j < i[2];j += N) {

\*j = (std::rand() % (N\*N)) + 1;

if (\*j > maxValue)

maxValue = \*j;

sort[count++] = j;

system("cls");

printMatrix(mat);

Sleep(250);

}

for (int \*j = i[2];j > i[3];j--) {

\*j = (std::rand() % (N\*N)) + 1;

if (\*j > maxValue)

maxValue = \*j;

sort[count++] = j;

system("cls");

printMatrix(mat);

Sleep(250);

}

for (int \*j = i[3];j > i[0];j -= N) {

\*j = (std::rand() % (N\*N)) + 1;

if (\*j > maxValue)

maxValue = \*j;

sort[count++] = j;

system("cls");

printMatrix(mat);

Sleep(250);

}

}

}

void makeMatrixB(int mat[N][N], int \*sort[N\*N]) {

maxValue = 0;

for (int \*i = mat[0];i < (mat[0] + N \* N);i += N) {

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

\*j = 0;

}

printMatrix(mat);

Sleep(250);

int count = 0;

for (int \*i[2] = { mat[0],(mat[0] + N \* (N - 1) + 1) };i[0] < mat[0] + N - 1;i[0] += 2, i[1] += 2) {

for (int \*j = i[0];j <= i[0] + N \* (N - 1);j += N) {

\*j = (std::rand() % (N\*N)) + 1;

if (\*j > maxValue)

maxValue = \*j;

sort[count++] = j;

system("cls");

printMatrix(mat);

Sleep(250);

}

for (int \*j = i[1];j >= i[0] + 1;j -= N) {

\*j = (std::rand() % (N\*N)) + 1;

if (\*j > maxValue)

maxValue = \*j;

sort[count++] = j;

system("cls");

printMatrix(mat);

Sleep(250);

}

}

}

void changeMatrixA(int mat[N][N]) {

for (int \*i = mat[0];i < mat[0] + N \* N;i += 2 \* N)

for (int \*j = i;j < i + N;j += 2) {

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

std::swap(\*(j + N), \*(j + N + 1));

std::swap(\*j, \*(j + N + 1));

}

printMatrix(mat);

}

void changeMatrixB(int mat[N][N]) {

for (int \*i = mat[0];i < mat[0] + N \* N;i += 2 \* N)

for (int \*j = i;j < i + N;j += 2) {

std::swap(\*(j + N + 1), \*j);

std::swap(\*(j + N), \*(j + 1));

}

printMatrix(mat);

}

void changeMatrixC(int mat[N][N]) {

for (int \*i = mat[0];i < mat[0] + N \* N;i += 2 \* N)

for (int \*j = i;j < i + N;j += 2) {

std::swap(\*(j + N), \*j);

std::swap(\*(j + 1), \*(j + N + 1));

}

printMatrix(mat);

}

void changeMatrixD(int mat[N][N]) {

for (int \*i = mat[0];i < mat[0] + N \* N;i += 2 \* N)

for (int \*j = i;j < i + N;j += 2) {

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

std::swap(\*(j + N), \*(j + N + 1));

}

printMatrix(mat);

}

void quicksort(int\*\* arr, int begin, int end)

{

int f = begin, l = end, mid = \*arr[(f + l) / 2];

while (f <= l)

{

while (\*arr[f] < mid) f++;

while (\*arr[l] > mid) l--;

if (f <= l)

{

std::swap(\*arr[f], \*arr[l]);

f++;

l--;

}

}

if (begin < l) quicksort(arr, begin, l);

if (f < end) quicksort(arr, f, end);

}

void userChangeMatrix(int mat[N][N]) {

char answer;

bool flag = true;

int ab;

while (flag) {//цикл для приема команд от пользователя

std::cout << "Введите символ операции +,-,\* или : для изменения матрицы\n";

std::cin >> answer;

switch (answer)

{

case '+':

std::cout << "Введите число, на которое увеличить матрицу\n";

while (true) {

std::cin >> ab;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

for (int\* i = mat[0];i < mat[0] + N \* N;i++)

\*i += ab;

maxValue += ab;

flag = false;

break;

case '-':

std::cout << "Введите число, на которое уменьшить матрицу\n";

while (true) {

std::cin >> ab;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

for (int\* i = mat[0];i < mat[0] + N \* N;i++)

\*i -= ab;

maxValue -= ab;

flag = false;

break;

case '\*':

std::cout << "Введите число, на которое умножить матрицу\n";

while (true) {

std::cin >> ab;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

for (int\* i = mat[0];i < mat[0] + N \* N;i++)

\*i \*= ab;

maxValue \*= ab;

flag = false;

break;

case ':':

std::cout << "Введите число, на которое поделить матрицу\n";

while (true) {

std::cin >> ab;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

for (int\* i = mat[0];i < mat[0] + N \* N;i++)

\*i /= ab;

maxValue /= ab;

flag = false;

break;

default:

std::cout << "Вы указали несуществующую команду";

break;

}

}

printMatrix(mat);

}