**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: "Указатели и матрицы"

|  |  |  |
| --- | --- | --- |
| Студент гр. 0324 |  | Гордиенко Т.Е. |
| Преподаватель |  | Глущенко А.Г |

Санкт-Петербург

2020

**Цель работы:** научиться использовать указатели.

**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:**

#include <iostream>

#include <Windows.h>

#include <chrono>

#include <random>

using namespace std;

void gotoxy(int x, int y) {//перемещение курсора консоли на координату {x,y}

COORD p = { x, y };

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), p);

}

void spiral(int size, int \*num, int \*matrix) {//присваивание и вывод спиралькой

system("cls");

long seed = chrono::system\_clock::now().time\_since\_epoch().count();

default\_random\_engine rand(seed);

uniform\_int\_distribution<int> d(1, size \* size);

int s = 1, left = -1, up = -1, right, down, x, y, temp;//счётчики заполненых граней, координаты и временная переменная

down = right = size;

x = 0;

y = 0;

gotoxy(x, y);

while (s <= size \* size) {

for (int i = left + 1; i < right; i++) {//Заполнение верхней строки от левой границы до провой

temp = d(rand);

cout << temp;

\*num = temp;//присваивание значения элементу матрицы

num++;//сдвиг указателя на сл элемент

s++;

x += 5;

gotoxy(x, y);//сдвиг каретки вправо

Sleep(100);

}

up++;//+ к счётчику заполнения

if (s > size\*size) {

break;

}

x -= 5;

num--;

y += 3;

num += 100;

gotoxy(x, y);

for (int i = up + 1; i < down; i++) {//заполнение правого столбца от верхней границы до нижней

temp = d(rand);

cout << temp;

\*num = temp;

s++;

y += 3;//сдвиг каретки вниз

num += 100;//сдвиг указателя на сл элемент стобца (+100, потомучто в одной строке 100 эелементов)

gotoxy(x, y);

Sleep(100);

}

right--;

if (s > size\*size) {

break;

}

y -= 3;

num -= 100;

x -= 5;

num--;

gotoxy(x, y);

for (int i = right - 1; i > left; i--) {//заполнение нижней строки от правой границы до левой

temp = d(rand);

cout << temp;

\*num = temp;

s++;

x -= 5;

num--;

gotoxy(x, y);

Sleep(100);

}

down--;

if (s > size\*size) {

break;

}

x += 5;

num++;

y -= 3;

num -= 100;

gotoxy(x, y);

for (int i = down - 1; i > up; i--) {//заполнение левого столбца от нижней границы до верхней

temp = d(rand);

cout << temp;

\*num = temp;

s++;

y -= 3;

num -= 100;

gotoxy(x, y);

Sleep(100);

}

left++;

y += 3;

num += 100;

x += 5;

num++;

gotoxy(x, y);

}

num = matrix;

x = 0;

y += ((size / 2) + 1) \* 3;

gotoxy(x, y);//сдвиг каретки на строку под матрицей

}

void serpentine(int size, int \*num, int \*matrix) {//присваивание и вывод серпантином

system("cls");

int s = 1, x = 0, y = 0, temp;

long seed = chrono::system\_clock::now().time\_since\_epoch().count();

default\_random\_engine rand(seed);

uniform\_int\_distribution<int> d(1, size \* size);

for (int i = 0; i != size; i++) {//повтаряется n раз для матрицы n\*n

if (i % 2 == 0) {

for (int j = 0; j != size; j++) {

temp = d(rand);

cout << temp;

\*num = temp;

s++;

y += 3;

num += 100;

gotoxy(x, y);

Sleep(100);

}

y -= 3;

num -= 100;

x += 5;

num++;

gotoxy(x, y);

}

else {

for (int j = 0; j != size; j++) {

temp = d(rand);

cout << temp;

\*num = temp;

s++;

y -= 3;

num -= 100;

gotoxy(x, y);

Sleep(100);

}

y += 3;

num += 100;

x += 5;

num++;

gotoxy(x, y);

}

}

num = matrix;

if (size % 2 == 0) {

y += 3 \* size + 3;

x = 0;

}

else {

y += 3;

x = 0;

}

gotoxy(x, y);

}

void swap\_clock(int size, int \*num, int \*matrix) {//смена по часовой стрелке

int temp;

for (int i = 0; i != size / 2; i++) {

for (int j = 0; j != size / 2; j++) {

temp = \*num;

\*num = \*(num + (100 \* (size / 2)));

\*(num + (100 \* (size / 2))) = \*(num + (100 \* (size / 2)) + (size / 2));

\*(num + (100 \* (size / 2)) + (size / 2)) = \*(num + (size / 2));

\*(num + (size / 2)) = temp;

num++;

}

num -= size / 2;

num += 100;

}

num = matrix;

}

void swap\_x(int size, int \*num, int \*matrix) {//смена крест на крест

int temp;

for (int i = 0; i != size / 2; i++) {

for (int j = 0; j != size / 2; j++) {

temp = \*num;

\*num = \*(num + 100 \* (size / 2) + (size / 2));

\*(num + 100 \* (size / 2) + (size / 2)) = temp;

num++;

}

num -= size / 2;

num += 100;

}

for (int i = size / 2; i != size; i++) {

for (int j = 0; j != size / 2; j++) {

temp = \*num;

\*num = \*(num + (size / 2) - 100 \* (size / 2));

\*(num + (size / 2) - 100 \* (size / 2)) = temp;

num++;

}

num -= size / 2;

num += 100;

}

num = matrix;

}

void swap\_up\_down(int size, int \*num, int \*matrix) {//смена верх и низ

int temp;

for (int i = 0; i != size / 2; i++) {

for (int j = 0; j != size; j++) {

temp = \*num;

\*num = \*(num + 100 \* (size / 2));

\*(num + 100 \* (size / 2)) = temp;

num++;

}

num -= size;

num += 100;

}

num = matrix;

}

void swap\_left\_right(int size, int \*num, int \*matrix) {//смена лево и право

int temp;

for (int i = 0; i != size; i++) {

for (int j = 0; j != size / 2; j++) {

temp = \*num;

\*num = \*(num + (size / 2));

\*(num + (size / 2)) = temp;

num++;

}

num -= size / 2;

num += 100;

}

num = matrix;

}

void sort(int size, int \*num, int \*matrix) {//сортировка

int temp, range = 100 - size + 1;

for (int j = size \* size; j != 0; j--) {

for (int i = 1; i != j; i++) {

if (i % size != 0) {

if (\*num > \*(num + 1)) {

temp = \*num;

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

\*(num + 1) = temp;

}

num++;

}

else {

if (\*num > \*(num + range)) {

temp = \*num;

\*num = \*(num + range);

\*(num + range) = temp;

}

num += range;

}

}

num = matrix;

}

num = matrix;

}

void change(int size, int \*num, int \*matrix) {// + - / \*

char action;

int chislo;

tryAgain:

cout << "Знак действия ";

cin >> action;

cout << "Число ";

cin >> chislo;

switch (action) {

case '\*':

for (int i = 0; i != size; i++) {

\*num \*= chislo;

num++;

}

break;

case '/':

for (int i = 0; i != size; i++) {

\*num /= chislo;

num++;

}

break;

case '+':

for (int i = 0; i != size; i++) {

\*num += chislo;

num++;

}

break;

case '-':

for (int i = 0; i != size; i++) {

\*num -= chislo;

num++;

}

break;

default:

cout << "try again";

goto tryAgain;

}

num = matrix;

}

void print(int size, int \*num, int \*matrix) {//вывод матрицы

for (int i = 0; i != size; i++) {

for (int j = 0; j != size; j++) {

if (\*num < 10) {

cout << \*num << " ";

}

else{

cout << \*num << " ";

}

num++;

}

num -= size;

num += 100;

cout << '\n';

}

num = matrix;

}

int main() {

setlocale(0, "");

int size, action;

int matrix[100][100];

int \*num = \*matrix;

cout << "Введите размер массива (n) ";

cin >> size;

tryAgain1:

cout << "Выбеирте действие: " << '\n';

cout << "1) Вывести массив спиралькой и присвоить рандомные значения" << '\n';

cout << "2) Вывести массив серпантином и присвоить рандомные значения" << '\n';

cout << "3) Поменять блоки по часовой стрелке" << '\n';

cout << "4) Поменять блоки крест на крест" << '\n';

cout << "5) Поменять верхний и нижний блок" << '\n';

cout << "6) Поменять левый и правый блок" << '\n';

cout << "7) Отсортировать матрицу по возрастанию" << '\n';

cout << "8) Умножить/разделить/увеличить/уменьшить элементы на число А" << '\n';

cout << "9) Вывести матрицу" << '\n';

cout << "0) Выход" << '\n' << '\n';

cin >> action;

switch (action) {

case 1:

spiral(size, num, \*matrix);

break;

case 2:

serpentine(size, num, \*matrix);

break;

case 3:

swap\_clock(size, num, \*matrix);

break;

case 4:

swap\_x(size, num, \*matrix);

break;

case 5:

swap\_up\_down(size, num, \*matrix);

break;

case 6:

swap\_left\_right(size, num, \*matrix);

break;

case 7:

sort(size, num, \*matrix);

break;

case 8:

change(size, num, \*matrix);

break;

case 9:

print(size, num, \*matrix);

break;

case 0:

return 0;

break;

default:

cout << "try again" << '\n';

goto tryAgain1;

}

goto tryAgain1;

}