**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедраа ИС**

отчет

**по практической работе №3**

**по дисциплине «Программирование»**

Тема: **Двумерные статические массивы. Указатели**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 2372 |  | Мельникова М.А. |
| 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 делящеую 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:

void XY(short x, short y, int n = 0)

{

COORD p = { x, y }; //Структура, определяющая координаты в буфере консоли

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

}

Для задержки во времени между выводом значений подключаем библиотеку <thread> и прописываем следующую строку в функции:

this\_thread::sleep\_for(chrono::milliseconds(100));

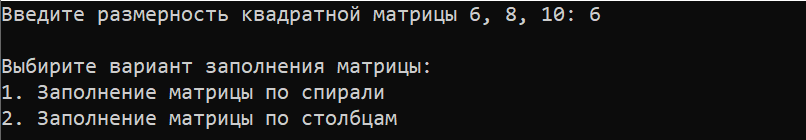
Для выполнения заданий пишем функции, куда передаём указатель на первый элемент массива, его размерность и остальные необходимые в конкретном задании значения.

При запуске программа выводит следующее:



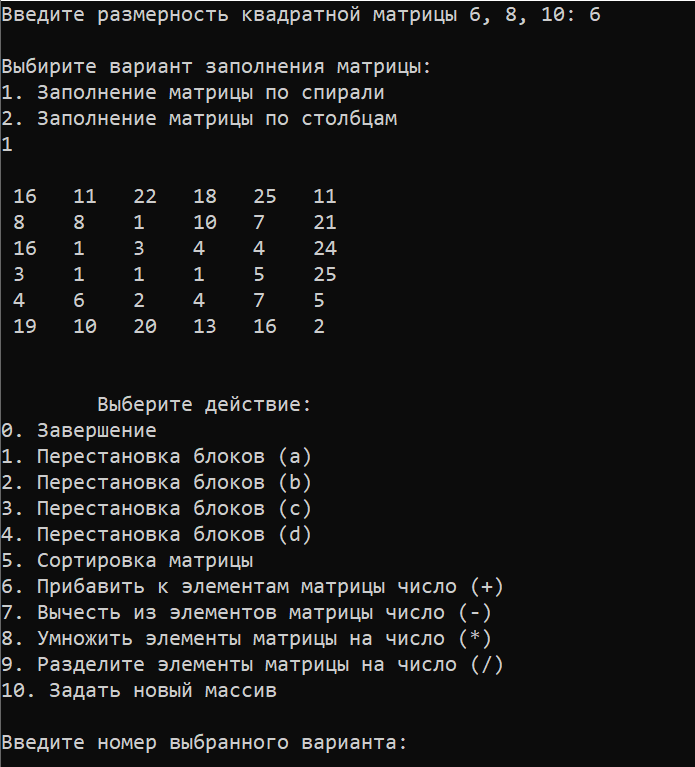
*Рис.1*

После выбора размерности предлагается выбор варианта заполнения матрицы:



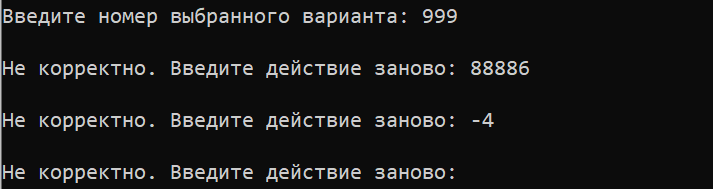
*Рис.2*

Когда значение введено матрица выводится в консоль с задержкой во времени между элементами и разворачивается полное меню с оставшимися заданиями



*Рис.3*

В случае введения номера несуществующего задания выводится предупреждение о некорректном вводе и возможность выбрать задание заново или выйти из программы



*Рис.4*

**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

Полный код программы

#include <iostream>

#include <string>

#include <thread> // sleep\_for

#include <Windows.h> //COORD

using namespace std;

/\*

"\x1B[31mtext\n" //красный

"\x1B[37mtext\n"; //белый

\*/

//Функция для перемещения по координатам консоли

void XY(short x, short y, int n = 0)

{

COORD p = { x, y }; //Структура, определяющая координаты в буфере консоли

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

this\_thread::sleep\_for(chrono::milliseconds(100));

}

void out(int N, int\* pArr)

{

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

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

if (\*pArr < 10 && \*pArr >= 0) {

cout << \*(pArr++) << " ";

}

else {

cout << \*(pArr++) << " ";

}

}

cout << "\n";

}

cout << "\n";

}

void spiral(int N, int\* pArr, short x, short y)

{

int n = N--; // n = N, N = N - 1

x = -4;

pArr--;

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

pArr++;

\*pArr = 1 + rand() % (N \* N);

x += 5; XY(x, y);

cout << \*pArr;

}

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

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

pArr += n;

\*pArr = 1 + rand() % (N \* N);

y += 1; XY(x, y);

cout << \*pArr;

}

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

pArr--;

\*pArr = 1 + rand() % (N \* N);

x -= 5; XY(x, y);

cout << \*pArr;

}

N--;

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

pArr -= n;

\*pArr = 1 + rand() % (N \* N);

y -= 1; XY(x, y);

cout << \*pArr;

}

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

pArr++;

\*pArr = 1 + rand() % (N \* N);

x += 5; XY(x, y);

cout << \*pArr;

}

N--;

}

y += (n / 2 + 1); x = 0; XY(x, y);

}

void snake(const int N, int\* pArr, short x, short y)

{

x = 1;

\*pArr = 1 + rand() % (N \* N);

XY(x, y); cout << \*pArr;

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

for (int j = 1; j < N; j++) {

pArr += N;

\*pArr = 1 + rand() % (N \* N);

y += 1; XY(x, y);

cout << \*pArr;

}

pArr++; \*pArr = 1 + rand() % (N \* N);

x += 5; XY(x, y);

cout << \*pArr;

for (int j = 1; j < N; j++) {

pArr -= N;

\*pArr = 1 + rand() % (N \* N);

y -= 1; XY(x, y);

cout << \*pArr;

}

pArr++;

if (i != N / 2 - 1) {

\*pArr = 1 + rand() % (N \* N);

x += 5; XY(x, y);

cout << \*pArr;

}

}

y += N + 1; x = 0; XY(x, y);

}

void transpositionA(const int N, int\* pArr) //Меняет подматрицы по часовой стрелке

{

int\* p1Arr = pArr, \* p2Arr = pArr + N / 2, \* p3Arr = pArr, \* p4Arr = pArr + N \* N / 2, \* p5Arr = pArr + N \* N / 2, \* p6Arr = p4Arr + N / 2;

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

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

swap(\*(p1Arr++), \*(p2Arr++));

}

p1Arr += N / 2, p2Arr += N / 2;

}

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

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

swap(\*(p3Arr++), \*(p4Arr++));

}

p3Arr += N / 2, p4Arr += N / 2;

}

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

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

swap(\*(p5Arr++), \*(p6Arr++));

}

p5Arr += N / 2, p6Arr += N / 2;

}

cout << "Новый массив:\n"; out(N, pArr);

}

void transpositionB(const int N, int\* pArr) //Меняет диагональные подматрицы

{

int\* p1Arr = pArr, \* p3Arr = pArr + N / 2, \* p4Arr = pArr + N \* N / 2, \* p2Arr = p4Arr + N / 2;

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

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

swap(\*(p1Arr++), \*(p2Arr++));

}

p1Arr += N / 2, p2Arr += N / 2;

}

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

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

swap(\*(p3Arr++), \*(p4Arr++));

}

p3Arr += N / 2, p4Arr += N / 2;

}

cout << "Новый массив:\n"; out(N, pArr);

}

void transpositionC(const int N, int\* pArr) //Меняет верхние и нижние подматрицы

{

int\* p1Arr = pArr, \* p2Arr = pArr + N \* N / 2;

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

swap(\*(p1Arr++), \*(p2Arr++));

}

cout << "Новый массив:\n"; out(N, pArr);

}

void transpositionD(const int N, int\* pArr) //Меняет правые и левые подматрицы

{

int\* p1Arr = pArr, \* p2Arr = pArr + N / 2;

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

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

swap(\*(p1Arr++), \*(p2Arr++));

}

p1Arr += N / 2, p2Arr += N / 2;

}

cout << "Новый массив:\n"; out(N, pArr);

}

void \_plus(int N, int\* pArr, int value) {

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

\*(pArr + i) += value;

}

cout << "\n"; out(N, pArr);

}

void \_minus(int N, int\* pArr, int value) {

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

\*(pArr + i) -= value;

}

cout << "\n"; out(N, pArr);

}

void \_multiply(int N, int\* pArr, int value) {

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

\*(pArr + i) \*= value;

}

cout << "\n"; out(N, pArr);

}

void \_divide(int N, int\* pArr, int value) {

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

\*(pArr + i) /= value;

}

cout << "\n"; out(N, pArr);

}

void sort(const int N, int\* pArr) //Сортирует массив сортировкой bubble sort

{

int t;

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

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

if (\*(pArr + j) < \*(pArr + i)) {

swap(\*(pArr + j), \*(pArr + i));

}

}

}

cout << "Отсортированный массив:\n"; out(N, pArr);

}

int main()

{

setlocale(0, "Rus");

srand(time(0));

bool flag = true;

while (flag) {

short x = 0, y = 0;

int Arr[100];

int\* pArr = Arr;

int N = 1;

cout << "Введите размерность квадратной матрицы 6, 8, 10: ";

y += 1;

cin >> N;

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

{

cout << "\n\x1B[31mНе корректно. Введите размерность матрицы заново:\x1B[37m ";

cin >> N;

y += 2;

}

cout << "\nВыбирите вариант заполнения матрицы:\n"

"1. Заполнение матрицы по спирали\n"

"2. Заполнение матрицы по столбцам\n";

y += 4;

int operation = 0;

cin >> operation;

while (operation != 1 && operation != 2)

{

cout << "\n\x1B[31mНе корректно. Введите вариант заполнения заново:\x1B[37m ";

cin >> operation;

y += 2;

}

y += 2;

switch (operation)

{

case 1:

{

spiral(N, pArr, x, y);

break;

}

case 2:

{

snake(N, pArr, x, y);

break;

}

default:

break;

}

cout <<

"\n\tВыберите действие:\n"

"0. Завершение\n"

"1. Перестановка блоков (a)\n"

"2. Перестановка блоков (b)\n"

"3. Перестановка блоков (c)\n"

"4. Перестановка блоков (d)\n"

"5. Сортировка матрицы\n"

"6. Прибавить к элементам матрицы число (+)\n"

"7. Вычесть из элементов матрицы число (-)\n"

"8. Умножить элементы матрицы на число (\*)\n"

"9. Разделите элементы матрицы на число (/)\n"

"10. Задать новый массив\n"

"11. Индивидуальное задание\n\n";

cout << "Введите номер выбранного варианта: ";

bool flag\_ = true;

while (flag\_) {

int option = 111;

cin >> option;

switch (option)

{

case 1:

{

transpositionA(N, pArr);

cout << "\nДля прололжения введите действие: ";

break;

}

case 2:

{

transpositionB(N, pArr);

cout << "\nДля прололжения введите действие: ";

break;

}

case 3:

{

transpositionC(N, pArr);

cout << "\nДля прололжения введите действие: ";

break;

}

case 4:

{

transpositionD(N, pArr);

cout << "\nДля прололжения введите действие: ";

break;

}

case 5:

{

sort(N, pArr);

cout << "\nДля прололжения введите действие: ";

break;

}

case 6:

{

int value;

cout << "(+) Введите число: ";

cin >> value;

\_plus(N, pArr, value);

cout << "\nДля прололжения введите действие: ";

break;

}

case 7:

{

int value;

cout << "(-) Введите число: ";

cin >> value;

\_minus(N, pArr, value);

cout << "\nДля прололжения введите действие: ";

break;

}

case 8:

{

int value;

cout << "(\*) Введите число: ";

cin >> value;

\_multiply(N, pArr, value);

cout << "\nДля прололжения введите действие: ";

break;

}

case 9:

{

int value;

cout << "(/) Введите число: ";

cin >> value;

\_divide(N, pArr, value);

cout << "\nДля прололжения введите действие: ";

break;

}

case 10:

{

flag\_ = false;

system("cls");

break;

}

case 11:

{

//Individual task

cout << "\nДля прололжения введите действие: ";

break;

}

case 0:

{

flag\_ = false;

flag = false;

break;

}

default:

cout << "\n\x1B[31mНе корректно. Введите действие заново:\x1B[37m ";

break;

}

}

}

}