**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: **Двумерные статические массивы. Указатели.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3372 |  | Надточаев Ф.А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цель работы.**

Изучить двумерныестатические массивы и арифметику указателей на практике; получить навыки работы с двумерными массивами, различными способами их заполнения и вывода, использования библиотеки ncureses для заданного вывода массивов на экран.

**Основные теоретические положения.**

В изучаемых нами языках программирования между массивами и указателями имеется очень тесная связь.

Кода мы определяем в программе некоторый массив, например,

int Arr[10];

переменная **Arr** без индексов представляет собой указатель на первый элемент массива в данном случае из 10 целых чисел (содержит адрес первого элемента массива). Если вывести на экран значение переменной **Arr**

cout << Arr:

мы увидим некоторое целое значение в шестнадцатеричном формате, соответствующее адресу первого элемента этого массива.

**Замечание**. Именно по этой причине в языке C++ отсутствует операция присвоения сразу всех значений одного массива другому (в некоторых других языках, например, в Pascal такая возможность имеется). Действительно, если имеются два массива

int A1[10], A2[10];

то попытка выполнить присвоение **A1 = A2** привела бы к тому, что переменная **A1** стала бы указывать на ту же область памяти, что и переменная **A2** (мы скопировали адрес из **A2** в **A1**, а не содержимое одного массива в другой). Адрес, который хранился ранее в переменной **A1,** был бы утерян, что привело бы к утечке памяти (для десяти элементов массива **A1** в памяти было выделено место, но теперь мы “забыли”, где оно находится, то есть потеряли память). По этой причине подобные операции с массивами в языке C++ запрещены. Более того, запрещены любые изменения значения переменной массива.

Указателю, имеющему такой же базовый тип, как и элементы массива, можно присвоить массив следующим образом:

int Arr[10];

int \*p;

p = Arr;

Но обратное присвоение выполнить невозможно:

Arr = p; // Ошибка

Такое присвоение невозможно, поскольку переменная массива – это константа, изменение которой запрещено.

Так как переменная массива является указателем на первый элемент массива, появляются дополнительные возможности по работе с массивами на основе использования арифметики указателей. Например, чтобы получить 5–й элемент массива **Arr** можно воспользоваться одним из следующих выражений:

**Arr[4]** или \*(**Arr + 4)** или **\*( p + 4)**

Первое выражение – это пример обычной индексации элементов массива. Во втором и третьем выражениях мы использовали арифметику указателей и с помощью операции + получили адрес пятого элемента массива. Затем с помощью операции \* взяли значение по этому адресу и получили значение 5-го элемента массива. Обратите внимание на скобки в этих выражениях, если их не поставить и написать \***Arr + 4** или **\*p + 4**, то эти выражения будут равны значению первого элемента массива увеличенного на 4, так как операция \* имеет больший приоритет, чем операция +.

Вот пример фрагмента программы для работы с массивом с помощью обычной индексации элементов массива. Этот фрагмент обеспечивает ввод элементов целочисленного массива с клавиатуры, вычисление квадратов значений элементов массива, а затем вывод элементов массива на экран:

int A[10];

for (int i = 0; i < 10; ++ i)

{

cin >> A[i];

A[i] = A[i] \* A[i];

}

for (int i = 0; i < 10; ++ i)

cout << A[i] << “ “;

cout << endl;

…..

А вот тот же фрагмент, но с использованием арифметики указателей:

int A[10];

for (int \*Next = A, \*End = Next + 9; Next <= End; ++ Next)

{

cin >> \*Next;

\*Next = \*Next \* \*Next; // \*Next = (\*Next) \* (\*Next);

}

for (int \*Next = A, \*End = Next + 9; Next <= End; ++ Next)

cout << \*Next << “ “;

cout << endl;

…

Использование арифметики указателей при работе с массивами приводит обычно к уменьшению объема генерируемого кода программы и к уменьшению времени ее выполнения, то есть к увеличению быстродействия.

Поскольку указатель и имя массива, в большой степени, взаимозаменяемы, указатели можно индексировать, как обычные массивы:

int A[10], \*P = A;

for (int i = 0; i < 10; ++ i)

cout << P[i] << “ “;

Можно создавать и массивы указателей. Например:

int a = 1, b = 2, c = 3, \*M[3];

M[0] = & a; // Элементам массива М присваиваются адреса переменных a, b и c

M[1] = & b;

M[2] = & c;

for (int i = 0; i < 3; ++ i)

cout << \*M[i] << “ ”;

cout << endl;

Массив **M** – это трехэлементный массив указателей на целые значения, то есть каждый элемент этого массива представляет собой указатель на целое.

С помощью массивов указателей можно моделировать различные интересные конструкции данных. Например, пусть имеется квадратная матрица размерности 5 х 5 симметричная относительно главной диагонали. Для ее однозначного представления достаточно хранить в памяти  не все 25 элементов этой матрицы, а только 15 (например, элементы под главной диагональю вместе с элементами главной диагонали).

**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 (\*Для манипуляции с элементами используйте только арифметику указателей):



2) Получает новую матрицу, из матрицы п. 1, переставляя ее блоки в соответствии со схемами (\*Для манипуляции с элементами используйте только арифметику указателей):



3) Используя арифметику указателей, сортирует элементы любой сортировкой из списка ниже (рекомендуется переиспользовать написанный код в работе №2 и модернизировать его для работы с указателями).   
Варианты сортировок:

* Shaker sort;
* Comb sort;
* Insert sort;
* Quick sort;

4) Уменьшает, увеличивает, умножает или делит все элементы матрицы на введенное пользователем число (\*Для манипуляции с элементами использовать только арифметику указателей).

**Выполнение работы.**

Код программы представлен в приложении А.

|  |  |
| --- | --- |
| Ввод пользователем и обработка данных | Работа алгоритма и вывод на экран |
| Выбор размерности матрицы |  |
| Пользователь вводит размерность матрицы (6,8,10). Число устанавливается как длина и ширина массива. |  |
| Старт меню | |
| На экран выводится меню из пунктов программы, каждый из которых выполняет одну из функций, описанных в пункте постановки задачи. |  |
| Пункт 1.1 Заполнение массива случайными числами до N\*N спиралью. | |
| Пользователь вводит 11. На экран выводится заполненный случайными числами массив с анимацией заполнения по спирали. |  |
| Пункт 1.2. Заполнение массива случайными числами до N\*N змейкой. | |
| Пользователь вводит 12. На экран выводится заполненный случайными числами массив с анимацией заполнения змейкой. |  |
| Пункт 2.1. Перестановка блоков матрицы по часовой стрелке. | |
| На экран выводится матрица, блоки которой поменяны местами по часовой стрелке относительно исходной матрицы. |  |
| Пункт 2.2. Перестановка блоков матрицы по диагонали. | |
| На экран выводится матрица, блоки которой поменяны местами по главной и побочной диагоналям относительно исходной матрицы. |  |
| Пункт 2.3. Перестановка верхние-нижние | |
| На экран выводится матрица, блоки которой поменяны местами по схеме верхние на нижние относительно исходной матрицы. |  |
| Пункт 2.4. Перестановка левые-правые | |
| На экран выводится матрица, блоки которой поменяны местами по схеме левые на правые относительно исходной матрицы. |  |
| Пункт 3. Сортировка массива | |
| На экран выводится отсортированная матрица, используется алгоритм shaker sort. |  |
| Пункт 4. Операция над каждым элементом массива | |
| На экран выводится список из операций, пользователь выбирает одну из опций, далее вводит операнд, на экран выводится матрица, с каждым элементом которой была выполнена соответствующая операция. |  |
|  |  |

**Выводы.**

По итогам работы мы изучили двумерные массивы и работу с ними через арифметику указателей , получили практические навыки работы с арифметикой указателей и библиотекой вывода ncurses. Результатом проделанной работы является программа, принцип работы которой представлен в пункте «Выполнение работы» и код которой находится в приложении А.

Приложение А

рабочий код

#include <algorithm>

#include <curses.h>

#include <iostream>

#include <ncurses.h>

#include <unistd.h>

using namespace std;

int getNumber ()

{

int a;

cin >> a;

while (cin.fail())

{

cin.clear();

cin.ignore();

cin >> a;

}

return a;

}

int lenOfNumber(int a)

{

int i = 0;

while (a > 0)

{

a /= 10;

i +=1;

}

return i;

}

void swapPointers(int \*a, int \*b)

{

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void shakerSort(int \*array, int length)

{

int start = 0;

int end = length - 2;

while (start <= end)

{

for (int i = start; i < length - 1 - start; ++i)

{

if (\*(array+i) > \*(array+i+1))

swapPointers(array+i, array+i+1);

}

for (int i = end ; i > 0 + start; --i)

{

if (\*(array+i) < \*(array+i-1))

swapPointers((array+i), (array+i-1));

}

start += 1;

end -= 1;

}

}

int main()

{

cout << "Введите размерность матрицы: " << endl;

int scale;

cin >> scale;

const int N = scale;

int A[N][N];

srand(time(0));

while (true)

{

cout << "#########################" << endl;

cout << "Выберите опцию программы: " << endl

<< "0 ИДЗ #10, является ли матрица волшебным квадратом" << endl

<< "1.1 Заполнение спиралью" << endl

<< "1.2 Заполнение змейкой" << endl

<< "2.1 Перестановка блоков по часовой стрелке" << endl

<< "2.2 Перестановка по диагонали" << endl

<< "2.3 Перестановка верхние-нижние" << endl

<< "2.4 Перестановка левые-правые" << endl

<< "3 Отсортировать массив" << endl

<< "4 Выполнить операцию над каждым элементом матрицы" << endl

<< "#################################################"<< endl;

int userChoice;

userChoice = getNumber();

switch (userChoice)

{

case 11:

{

// #1.1

initscr();

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

{

for (int j = 0; j < N-2\*i; ++j)

{

\*((\*A+j)+i\*N+i) = rand() % (N\*N+1);

move((j+i\*N+i)/N, (j+i\*N+i)%N\*(lenOfNumber(N\*N)+1));

printw("%0\*d", lenOfNumber(N\*N), \*((\*A+j)+i\*N+i));

refresh();

usleep(100000);

}

for (int j = 0; j < N-2 - 2\*i ; ++j)

{

\*((\*A+((i+2)\*N-(i+1))+j\*N)) = rand() % (N\*N+1);

move(((i+2)\*N-(i+1)+j\*N)/N, (((i+2)\*N-(i+1)+j\*N)%N)\*(lenOfNumber(N\*N)+1));

printw("%0\*d", lenOfNumber(N\*N), \*((\*A+((i+2)\*N-(i+1))+j\*N)));

refresh();

usleep(100000);

}

for (int j = 0; j < N - 2\*i; ++j)

{

\*((\*A+((N-i)\*N-(i+1)) - j)) = rand() % (N\*N+1);

move(((N-i)\*N-(i+1)-j)/N, (((N-i)\*N-(i+1)-j)%N)\*(lenOfNumber(N\*N)+1));

printw("%0\*d", lenOfNumber(N\*N), \*((\*A+((N-i)\*N-(i+1)) - j)));

refresh();

usleep(100000);

}

for (int j = 0; j < N-2 - 2\*i; ++j)

{

\*((\*A+((N-2-i)\*N+i) - j\*N)) = rand() % (N\*N+1);

move(((N-2-i)\*N+i - j\*N)/N, (((N-2-i)\*N+i - j\*N)%N)\*(lenOfNumber(N\*N)+1));

printw("%0\*d", lenOfNumber(N\*N), \*((\*A+((N-2-i)\*N+i) - j\*N)));

refresh();

usleep(100000);

}

}

endwin();

clear();

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

{

printf("%0\*d", lenOfNumber(N\*N), \*(\*A+i));

cout << (((i + 1) % N == 0) ? '\n':' ');

}

break;

}

case 12:

{

// #1.2

initscr();

int i = -1;

for (; i < N-1;)

{

i += 1;

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

{

\*(\*A+j\*N+i) = rand() % (N\*N+1);

move((j\*N+i)/N, ((j\*N+i)%N)\*(lenOfNumber(N\*N)+1));

printw("%0\*d", lenOfNumber(N\*N), \*(\*A+j\*N+i));

refresh();

usleep(100000);

}

i+=1;

for (int j = N - 1; j >= 0; --j)

{

\*(\*A+j\*N+i) = rand() % (N\*N+1);

move((j\*N+i)/N, ((j\*N+i)%N)\*(lenOfNumber(N\*N)+1));

printw("%0\*d", lenOfNumber(N\*N), \*(\*A+j\*N+i));

refresh();

usleep(100000);

}

}

endwin();

clear();

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

{

printf("%0\*d", lenOfNumber(N\*N), \*(\*A+i));

cout << (((i + 1) % N == 0) ? '\n':' ');

}

break;

}

case 21:

{

// #2.1

int B[N][N];

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

{

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

{

if (i < N/2)

{

\*(\*B+i\*N+N/2+j) = \*(\*A+i\*N+j); // из левой части в правую

\*(\*B+i+j\*N) = \*(\*A+i+N/2\*N+j\*N); // из нижней части в верхнюю

}

else

{

\*(\*B+i\*N+j) = \*(\*A+i\*N+N/2+j); // из правой части в левую

\*(\*B+i+N/2\*N+j\*N) = \*(\*A+i+j\*N); // из верхней части в нижнюю

}

}

}

cout << "После преобразования: " << endl;

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

{

printf("%0\*d", lenOfNumber(N\*N), \*(\*B+i));

cout << (((i + 1) % N == 0) ? '\n':' ');

}

break;

}

case 22:

{

// #2.2

int B[N][N];

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

{

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

{

\*(\*B+N/2\*N+N/2+i+j\*N) = \*(\*A+i+j\*N); // правый нижний <- левый верхний

\*(\*B+i+j\*N) = \*(\*A+N/2\*N+N/2+i+j\*N); // левый верхний <- правый нижний

\*(\*B+N/2\*N+i+j\*N) = \*(\*A+N/2+i+j\*N); // левый нижний <- правый верхний

\*(\*B+N/2+i+j\*N) = \*(\*A+N/2\*N+i+j\*N); // праый верхний <-левый нижний

}

}

cout << "После преобразования: " << endl;

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

{

printf("%0\*d", lenOfNumber(N\*N), \*(\*B+i));

cout << (((i + 1) % N == 0) ? '\n':' ');

}

break;

}

case 23:

{

// #2.3

int B[N][N];

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

{

\*(\*B+i) = \*(\*A+N\*N/2+i);

}

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

{

\*(\*B+i) = \*(\*A+i-N\*N/2);

}

cout << "После преобразования: " << endl;

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

{

printf("%0\*d", lenOfNumber(N\*N), \*(\*B+i));

cout << (((i + 1) % N == 0) ? '\n':' ');

}

break;

}

case 24:

{

// #2.4

int B[N][N];

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

{

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

{

\*(\*B+j+i\*N) = \*(\*A+j+N/2+i\*N);

\*(\*B+j+N/2+i\*N) = \*(\*A+j+i\*N);

}

}

cout << "После преобразования: " << endl;

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

{

printf("%0\*d", lenOfNumber(N\*N), \*(\*B+i));

cout << (((i + 1) % N == 0) ? '\n':' ');

}

break;

}

case 3:

{

// #3

cout << "Отсортируем матрицу:" << endl;

shakerSort(\*A, N\*N);

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

{

printf("%0\*d", lenOfNumber(N\*N), \*(\*A+i));

cout << (((i + 1) % N == 0) ? '\n':' ');

}

break;

}

case 4:

{

// #4

cout << "Выберите операцию для действий с матрицей:\n"

<< "0. Прибавить \n"

<< "1. Вычесть \n"

<< "2. Умножить \n"

<< "3. Поделить \n";

int choice;

choice = getNumber();

int userNumber;

cout << "Введите число, с которым хотите произвести операцию: ";

userNumber = getNumber();

switch (choice)

{

case 0:

{

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

{

\*(\*A+i) += userNumber;

}

break;

}

case 1:

{

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

{

\*(\*A+i) -= userNumber;

}

break;

}

case 2:

{

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

{

\*(\*A+i) \*= userNumber;

}

break;

}

case 3:

{

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

{

\*(\*A+i) /= userNumber;

}

break;

}

default:

{

cout << "Введите номер операции из предложенных" << endl;

}

}

cout << "После преобразования: " << endl;

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

{

printf("%0\*d", lenOfNumber(N\*N), \*(\*A+i));

cout << (((i + 1) % N == 0) ? '\n':' ');

}

break;

}

}

}

}