**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №1**

**по дисциплине «Программирование»**

Тема: Двумерные статические массивы. Указатели.

|  |  |  |
| --- | --- | --- |
| Студент(ка) гр. 2372 |  | Котлова П. В. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Изучение двумерных статических массивов; применение ввода, вывода, сортировка двумерных массивов, используя арифметику указателей; работа с условной инструкцией, инструкцией множественного выбора и циклами.

**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.

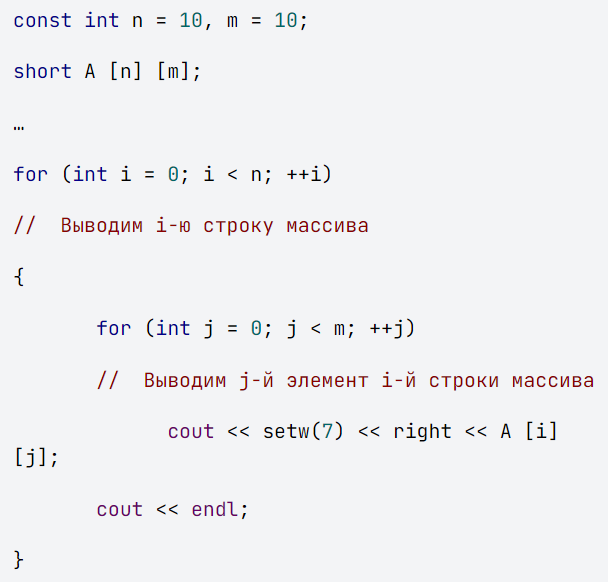
Все массивы можно разделить на две группы: одномерные и многомерные. Описание массива в программе отличается от объявления обычной переменной наличием размерности массива, которая задается в квадратных скобках после имени.

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

Многомерные массивы определяются аналогично одномерным массивам. Количество элементов по каждому измерению указывается отдельно в квадратных скобках: int A1 [5] [3].

Общее число элементов в многомерном массиве определяется как произведение количества элементов по каждому измерению. Так, например, массив **D** содержит **10 \* 15 \* 3 = 450** элементов типа **double**, а объем памяти, требующийся для этого массива, будет равен **450 \* 8 = 3600** байтам.

Вывод двумерных массивов, как правило, осуществляется в табличной форме. Реализация такого алгоритма может быть, например, такой:



**Указатели** – это тоже обычные переменные, но они **служат для хранения адресов памяти**.

Указатели определяются в программе следующим образом:

**<тип данных> \*<имя переменной>**

К указателям можно применять некоторые арифметические операции. К таким операциям относятся: +,**-**, **++**, **--**. Результаты выполнения этих операций по отношению к указателям существенно отличаются от результатов соответствующих арифметических операций, выполняющихся с обычными числовыми данными.

int A = 20, B = 30;

Указатель **p1** содержит адрес переменной **A**, который равен 100 и \***p1** будет равно значению переменной **A**, то есть 20. Выполним следующую операцию:

p1 = p1 + 1;

int \*p1 = &A;

значение указателя увеличится на **sizeof(<базовый тип указателя>) \* N** и в нашем случае это приращение будет равно **sizeof(int) \* N = 4 \* N**. Так, если N = 4, а p1= 100, то значение указателя **p1** увеличится на 16 и станет равно 116, и указатель будет  ссылаться на данные, расположенные по адресу 116.

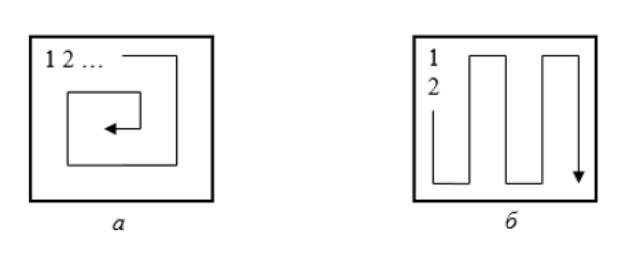
Так как переменная массива является указателем на первый элемент массива, появляются дополнительные возможности по работе с массивами на основе использования арифметики указателей. Например, чтобы получить 5–й элемент массива **Arr**можно воспользоваться одним из следующих выражений:

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

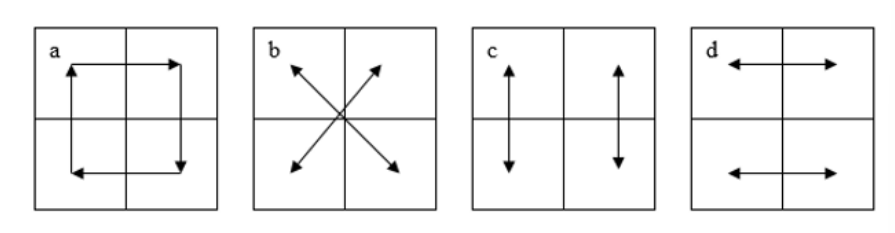
**Постановка задачи.**

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:



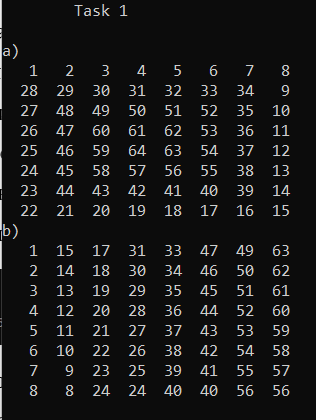
1. Используя арифметику указателей, сортирует элементы любой сортировкой.
2. Уменьшает, увеличивает, умножает или делит все элементы матрицы на введенное пользователем число.

**Выполнение работы.**

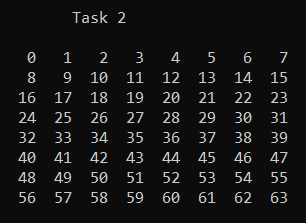
Код программы представлен в приложении А.

Описание кода и использованных алгоритмов.

1. При запуске программы появляется окно, в котором выводятся две матрицы порядка N=8. Для удобства значения элементов были выведены по возрастанию. В пункте а на экран выведена матрица, заполненная по спирали, а в пункте b ­– змейкой:

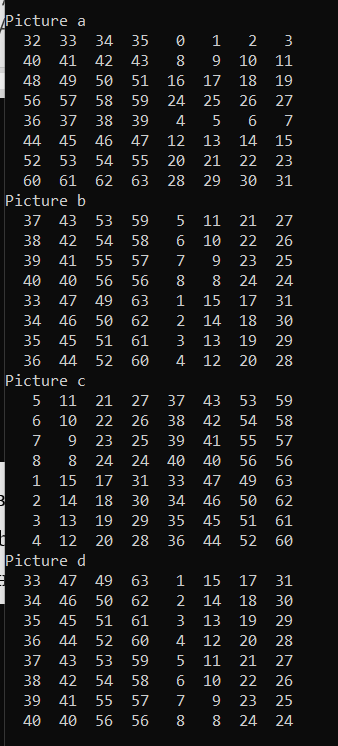


1. В следующем пункте программы выводится матрица порядка N=8, элементы опять же выведены по возрастанию значений.

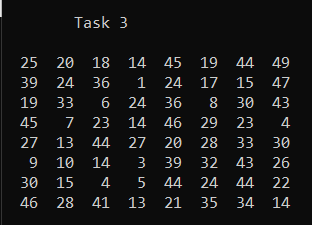
****

С ней программа будет работать дальше.

1. Затем на экране выводятся 4 матрицы с переставленными блоками предыдущей матрицы, как показано на рисунках в задании.

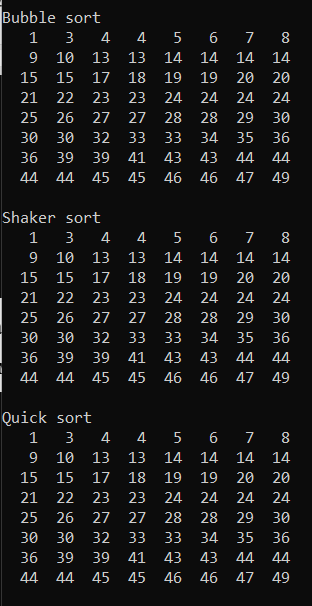


1. Затем выводится матрица, значения которой вводятся рандомно.

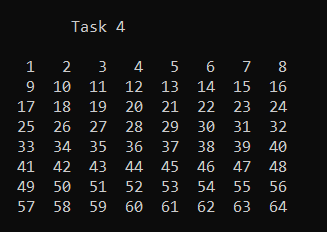


С ней программа будет работать дальше.

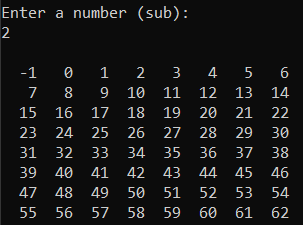
1. Данная матрица сортируется компилятором следующими сортировками:

****

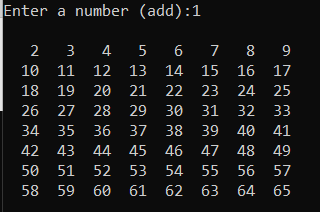
1. Далее программа выводит матрицу порядка N=8, с которой дальше будет работать программа.

****

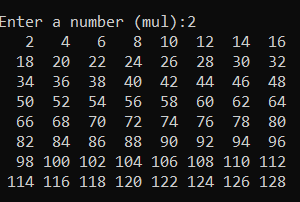
1. Программа уменьшает все элементы матрицы на введенное пользователем число:

****

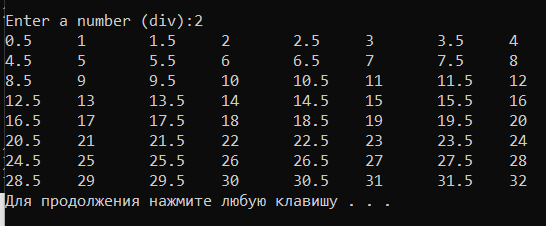
1. Программа увеличивает все элементы матрицы на введенное пользователем число:

****

1. Программа умножает все элементы матрицы на введенное пользователем число:

****

1. Программа делит все элементы матрицы на введенное пользователем число:

****

**Выводы.**

В ходе работы были изучены двумерные статические массивы, ввод, вывод, сортировка двумерных массивов, функции, указатели и работа с ними. В ходе работы была повторена работа с условной инструкцией и инструкцией множественного выбора и циклами.

Приложение А

рабочий код

#include <stdio.h>

#include <iostream>

#include "windows.h"

#include <iomanip>

#include <cstdlib>

#include <conio.h>

#include <ctime>

using namespace std;

const int n = 8;

void printmas(int (\*arr)[n], int n)

{

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

{

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

{

cout << setw(4) << arr[i][j];

if (j == (n - 1))

cout << "\n";

}

}

}

void print(int\* arr[], int n)

{

int i, j;

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

{

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

cout << setw(4) << arr[i][j];

if (j == (n - 1))

cout << "\n";

}

}

}

void cols\_by\_snake(int (\*a)[n], int n) {

int i, j, k = 1;

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

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

a[i][j] = (j % 2 ? (j + 1) \* n - i - 1 : k);

k++;

}

}

void output\_matrix(int (\*a)[n], int n) {

int i, j;

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

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

Sleep(30);

cout << setw(4) << a[i][j];

if (j == (n - 1))

cout << "\n";

}

}

}

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

{

int mid;

int f = begin;

int l = endn;

mid = arr[(f + l) / 2];

while (f < l)

{

while (arr[f] < mid) f++; //В левой части массива пропускаем(оставляем на месте) элементы, которые меньше центрального

while (arr[l] > mid) l--; // В правой части пропускаем элементы, которые больше центрального

if (f <= l)

{

swap(arr[f], arr[l]); ////Меняем элементы местами

f++;

l--;

}

}

////Рекурсивные вызовы, если осталось, что сортировать

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

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

}

void bubblesort(int (\*arr)[n], int n)

{

for (int k = 0; k < n; ++k)

for (int l = 0; l < n; ++l)

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

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

if ((i + 1 == n) && (j + 1 == n))

continue;

else {

if (j + 1 == n && arr[i][j] > arr[i + 1][0])

swap(arr[i][j], arr[i + 1][0]);

else {

if (arr[i][j] > arr[i][j + 1])

swap(arr[i][j], arr[i][j + 1]);

}

}

}

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

{

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

{

cout << setw(4) << arr[i][j];

if (j == (n - 1))

cout << "\n";

}

}

}

void shakersort(int(\*mas)[n], int n)

{

int left = 0, right = n - 1;

int flag = 1;

while ((left < right) && flag > 0)

{

flag = 0;

for (int i = left; i < right; i++)

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

{

if (mas[i][j] > mas[i + 1][j])

{

swap(mas[i][j], mas[i + 1][j]);

flag = 1; // перемещения в этом цикле были

}

}

right--;

for (int i = right; i > left; i--)

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

{

if (mas[i - 1][j] > mas[i][j]) {

swap(mas[i - 1][j], mas[i][j]);

flag = 1; // перемещения в этом цикле были

}

}

left++; // сдвигаем левую границу на следующий элемент

}

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

{

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

{

cout << setw(4)<< mas[i][j];

if (j == (n - 1))

cout << "\n";

}

}

}

void qsort(int \*arr[], int endl, int begin)

{

int mid;

int f = begin;

int l = endl;

mid = \*arr[(f + l) / 2];

while (f < l)

{

while (\*arr[f] < mid) f++; //В левой части массива пропускаем(оставляем на месте) элементы, которые меньше центрального

while (\*arr[l] > mid) l--; // В правой части пропускаем элементы, которые больше центрального

if (f <= l)

{

swap(arr[f], arr[l]); ////Меняем элементы местами

f++;

l--;

}

}

////Рекурсивные вызовы, если осталось, что сортировать

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

if (l < endl) qsort(arr, endl, f);

}

#define N 4

int main() {

srand(time(0));

cout << "\tTask 1\n\n";

cout << "a)\n";

int const n = 8;

int i = 1, I, j, k,

p = n / 2;

int M[n][n], \*ptr[n \* n];

// Обнуление матрицы и инициализация указателя

for (I = 0; I < n; ++I)

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

{

M[I][j] = 0; ptr[I \* n + j] = &M[I][j];

}

for (k = 1; k <= p; k++) // Число спиралей

{

//Верхний горизонтальный столбец

for (j = (k-1); j < (n-k+1); j++) \*ptr[(k-1)\*n + j] = i++;

//Правый верхний столбец

for (j = k; j < (n - k + 1); j++)

\*ptr[j \* n + (n - k)] = i++;

//Нижний горизонтальный столбец

for (j = (n-k-1); j >= (k-1); --j)

\*ptr[(n - k) \* n + j] = i++;

//Левый верхний столбец

for (j = (n - k - 1); j >= k; j--)

\*ptr[j \* n + (k - 1)] = i++;

}

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

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

if (n \* n < 20 \* 20)

{

Sleep(30);

cout << setw(4)<< \* ptr[i \* n + j];

if (j == (n - 1))

cout<< "\n";

}

}

cout << "b)\n";

int arr[n][n];

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

{

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

{

arr[i][j] = i \* n + j;

}

}

cols\_by\_snake(&arr[0],n);

output\_matrix(&arr[0],n);

cout << "\n\n\tTask 2\n\n";

int arr1[n][n];

int arr2[n][n];

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

{

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

{

arr2[i][j] = i \* n + j;

}

}

printmas(&arr2[0],n);

cout << "\n\n";

cout << "Picture a\n";

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

{

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

{

arr1[i][j] = arr2[i + p][j];

arr1[i][j + p] = arr2[i][j];

arr1[i + p][j + p] = arr2[i][j + p];

arr1[i + p][j] = arr2[i + p][j + p];

}

}

printmas(&arr1[0],n);

cout << "Picture b\n";

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

{

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

{

arr1[i][j] = arr[i + p][j + p];

arr1[i][j + p] = arr[i+p][j];

arr1[i + p][j + p] = arr[i][j];

arr1[i + p][j] = arr[i][j + p];

}

}

printmas(&arr1[0], n);

cout << "Picture c\n";

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

{

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

{

arr1[i][j] = arr[i + p][j];

arr1[i][j + p] = arr[i + p][j + p];

arr1[i + p][j] = arr[i][j];

arr1[i + p][j + p] = arr[i][j + p];

}

}

printmas(&arr1[0], n);

cout << "Picture d\n";

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

{

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

{

arr1[i][j] = arr[i][j + p];

arr1[i][j + p] = arr[i][j];

arr1[i + p][j] = arr[i + p][j + p];

arr1[i + p][j + p] = arr[i + p][j];

}

}

printmas(&arr1[0], n);

cout << "\n\n\tTask 3\n\n";

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

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

arr[i][j] = rand() % 49 + 1;

printmas(&arr[0], n);

cout << "\nBubble sort\n";

bubblesort(&arr[0], n);

cout << "\nShaker sort\n";

shakersort(&arr[0], n);

cout << "\nQuick sort\n";

int\* pa[n];

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

pa[i] = arr[i];

qsort(pa, 0, n);

print(pa, n);

cout << "\n\n\tTask 4\n\n";

signed int mas[n][n];

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

{

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

{

mas[i][j] = i \* n + j+1;

cout << setw(4) << mas[i][j];

if (j == (n - 1))

cout << "\n";

}

}

cout << "\nEnter a number (sub):";

int l, d, o, s;

cin >> l;

cout << "\n";

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

{

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

{

mas[i][j] -= l;

cout << setw(4) << mas[i][j];

if (j == (n - 1))

cout << "\n";

}

}

cout << "Enter a number (add):";

cin >> d;

cout << endl;

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

{

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

{

mas[i][j] = i \* n + j + 1;

}

}

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

{

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

{

mas[i][j] += d;

cout << setw(4) << mas[i][j];

if (j == (n - 1))

cout << "\n";

}

}

cout << endl;

cout << "Enter a number (mul):";

cin >> o;

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

{

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

{

mas[i][j] = i \* n + j + 1;

}

}

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

{

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

{

mas[i][j] \*= o;

cout << setw(4) << mas[i][j];

if (j == (n - 1))

cout << "\n";

}

}

cout << endl;

cout << "Enter a number (div):";

cin >> s;

float w[n][n];

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

{

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

{

w[i][j] = i \* n + j + 1;

}

}

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

{

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

{

w[i][j] /= s;

cout << w[i][j] <<'\t';

if (j == (n - 1))

cout << "\n";

}

}

system("pause");

}