**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. 2372 |  | Сарайкин А. А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2021

**Цель работы.**

Изучение двумерных статических массив и основных алгоритмов работы с ними. Изучение ссылок и указателей, арифметики указателей. Написание программы, заполняющей матрицу размера N (N = 6, 8, 10) и заполняющий его паттернами «змейка» и «спираль». Программа должна так же менять местами блоки (подмассивы) матрицы в соответствии с четырьмя паттернами обмена из задания. Сортировать матрицу с использованием арифметики указателей и увеличивать, уменьшать, умножать и делить все элементы матрицы на число, введенное пользователем

**Основные теоретические положения.**

Указатели поддерживают ряд операций: присваивание, получение адреса указателя, получение значения по указателю, некоторые арифметические операции и операции сравнения.

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.

Добавлять к указателям или вычитать из указателей можно только целые значения.

Поскольку упомянутые арифметические операции выполняются по-разному при их применении к указателям и обычным арифметическим типам данных, а также учитывая высший приоритет операции \*, при использовании указателей в составе выражений следует внимательно обращаться со скобками.

Указатели – это очень мощное, полезное, но и очень опасное средство. Ошибки, которые возникают при неправильном использовании указателей, кроме того, что могут приводить к серьезным и непредсказуемым ошибкам в работе программы, еще и очень трудно диагностировать (обнаруживать).

**Постановка задач.**

1. Написать для каждого задания функцию, структурировать код.
2. Вывести двумерный массив двумя способами, используя работу с кареткой.
3. Отсортировать двумерный массив сортировкой «Вставками», не обращаясь к индексам элементов. (используя арифметику указателей)

**Выполнение работы.**

Код программы представлен в приложении А.

1. При запуске программы пользователь вводит размер квадратной матрицы.
2. Пользователю предлагается выбрать способ вывода массива.
3. Массив выводится змейкой или спиралью
4. Пользователю предлагается выбрать один из 4 методов перестановки блоков
5. Пользователю предлагается ввести число и операцию. Массив умножается/делится/уменьшается/увеличивается на введенное пользователем число.

**Выводы.**

Код написан, и работает корректно. Все задачи выполнены.

Приложение А

рабочий код

include <iostream>  
#include <windows.h>  
#include <chrono>  
#include <thread>  
using namespace std;  
void XY(int x, int y){  
 COORD coord;  
 coord.X = x;  
 coord.Y = y;  
 SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coord);  
 this\_thread :: sleep\_for(chrono :: milliseconds (400));  
}  
void null\_arr (int n, int \*arr){  
 for(int \*parr = arr, i = 1; i <= n \* n; i++, parr++){  
 \*parr = 0;  
 }  
}  
void arr\_out (int n, int \*arr){  
 for(int \*parr = arr, i = 1; i <= n \* n; i++, parr++){  
 if (\*parr < 10){  
 cout <<" ";  
 }  
  
 cout << \*parr <<" ";  
  
 if (i % n == 0){  
 cout << "\n";  
 }  
 }  
}  
void spiral (int n, int \*arr){  
 int \*parr = arr, i = 1 ,pop = 1, x = 0, y = 0;  
 while (i <= n \* n){  
 for(int j = 1; j <= n - pop; j++){  
 \*arr = i;  
 cout << \*arr;  
 if (j != n) XY(x += 3, y);  
 i++;  
 arr++;  
 }  
 for(int j = 1; j <= n - pop; j++){  
 \*arr = i;  
 cout << \*arr;  
 XY(x, ++y);  
 i++;  
 arr += n;  
 }  
 for(int j = 1; j <= n - pop; j++){  
 \*arr = i;  
 cout << \*arr;  
 XY(x -= 3, y);  
 i++;  
 arr--;  
 }  
 for(int j = 1; j <= n - pop; j++){  
 \*arr = i;  
 i++;  
 if(j == n - pop){  
 cout << \*arr;  
 XY(x += 3,y);  
 arr++;  
 }  
 else{  
 cout << \*arr;  
 XY(x,--y);  
 arr -= n;  
 }  
 }  
 pop += 2 ;  
 }  
 cout << "\n" << parr;  
 cout << "\n" << arr - ((n \* n) - 1);  
 XY(0,n + 2);  
}  
void snake(int n, int \*arr){  
 int \*parr = arr, i = 1, x = 0, y = 0;  
 while(i <= n \* n){  
 for(int j = 1; j <= n; j++){  
 \*parr = i;  
 i++;  
 if(j == n){  
 cout << \*parr;  
 XY(x += 3,y);  
 parr++;  
 }  
 else{  
 cout << \*parr;  
 XY(x,++y);  
 parr += n;  
 }  
 }  
 for(int j = 1; j <= n; j++){  
 \*parr = i;  
 i++;  
 if(j == n){  
 cout << \*parr;  
 XY(x += 3,y);  
 parr++;  
 }  
 else{  
 cout << \*parr;  
 XY(x,--y);  
 parr -= n;  
 }  
 }  
 }  
 XY(0,n + 2);  
}  
void permutationA(int n, int \*arr){  
 int buff\_1,buff\_2,buff\_3, \*parr = arr;  
 for(int j = 0; j < n / 2;j++){  
 for (int g = 0; g < n / 2; g++){  
 buff\_1 = \*(parr + n / 2);  
 \*(parr + n / 2) = \*parr;  
 parr += n / 2;  
 buff\_2 = \*(parr + (n / 2) \* n);  
 \*(parr + (n / 2) \* n) = buff\_1;  
 parr += (n / 2) \* n;  
 buff\_3 = \*(parr - n / 2);  
 \*(parr - n / 2) = buff\_2;  
 parr -= n / 2;  
 \*(parr - (n / 2) \* n) = buff\_3;  
 parr -= (n / 2) \* n;  
 if (g != n / 2 - 1) parr++;  
 }  
 parr += (n / 2 + 1);  
 }  
 arr\_out(n, arr);  
}  
void permutationB(int n, int \*arr){  
 int \*parr = arr;  
 for(int j = 0; j < n / 2;j++){  
 for (int g = 0; g < n / 2; g++){  
 swap(\*parr,\*(parr + (n / 2) \* n + n / 2 ));  
 if (g != n / 2 - 1) parr++;  
 }  
 parr += (n / 2 + 1);  
 }  
 parr = arr + n / 2;  
 for(int j = 0; j < n / 2;j++){  
 for (int g = 0; g < n / 2; g++){  
 swap(\*parr,\*(parr + (n / 2) \* n - n / 2 ));  
 if (g != n / 2 - 1) parr++;  
 }  
 parr += (n / 2 + 1);  
 }  
 arr\_out(n, arr);  
}  
void permutationC(int n, int \*arr) {  
 int \*parr = arr;  
 for(int j = 0; j < n / 2;j++){  
 for (int g = 0; g < n / 2; g++){  
 swap(\*parr,\*(parr + (n / 2) \* n ));  
 if (g != n / 2 - 1) parr++;  
 }  
 parr += (n / 2 + 1);  
 }  
 parr = arr + n / 2;  
 for(int j = 0; j < n / 2;j++){  
 for (int g = 0; g < n / 2; g++){  
 swap(\*parr,\*(parr + (n / 2) \* n ));  
 if (g != n / 2 - 1) parr++;  
 }  
 parr += (n / 2 + 1);  
 }  
 arr\_out(n, arr);  
}  
void permutationD(int n, int \*arr) {  
 int \*parr = arr;  
 for(int j = 0; j < n / 2;j++){  
 for (int g = 0; g < n / 2; g++){  
 swap(\*parr,\*(parr + (n / 2)));  
 if (g != n / 2 - 1) parr++;  
 }  
 parr += (n / 2 + 1);  
 }  
 parr = arr + (n / 2) \* n;  
 for(int j = 0; j < n / 2;j++){  
 for (int g = 0; g < n / 2; g++){  
 swap(\*parr,\*(parr + (n / 2) ));  
 if (g != n / 2 - 1) parr++;  
 }  
 parr += (n / 2 + 1);  
 }  
 arr\_out(n, arr);  
}  
  
void arr\_sort(int \*arr, int \*end, int n){  
  
 for (int \*parr\_1 = arr + 1; parr\_1 <= end; parr\_1++){  
 for (int \*parr = parr\_1;parr > arr;parr-- ){  
 if (\*parr < \*(parr - 1)) {  
 swap(\*parr,\*(parr - 1));  
 }  
 }  
  
 }  
 arr\_out(n,arr);  
}  
void op\_plus(int \*arr,int \*end, int n, int value){  
 for (int \*parr = arr; parr <= end; parr++){  
 \*parr += value;  
 }  
 arr\_out(n, arr);  
}  
void op\_minus(int \*arr,int \*end, int n, int value){  
 for (int \*parr = arr; parr <= end; parr++){  
 \*parr -= value;  
 }  
 arr\_out(n, arr);  
}  
void op\_multiplication(int \*arr,int \*end, int n, int value){  
 for (int \*parr = arr; parr <= end; parr++){  
 \*parr \*= value;  
 }  
 arr\_out(n, arr);  
}  
void op\_division(int \*arr,int \*end, int n, int value){  
 for (int \*parr = arr; parr <= end; parr++){  
 \*parr /= value;  
 }  
 arr\_out(n, arr);  
}  
void transposeMatrix(int \*arr, int N) {  
 int size = N;  
 for (int \*pStart = arr; pStart < arr + N\*N; pStart += N + 1) {  
 for (int \*pRow = pStart, \*pCol = pStart; pCol < pStart + size; pRow += N, ++pCol)  
 std::swap(\*pRow, \*pCol);  
 --size;  
 }  
 arr\_out(N, arr);  
}  
int main(){  
 int n;  
 char letter;  
 cin >> n;  
 int arr[n][n];  
 null\_arr(n, &arr[0][0]);  
  
 repeat1:  
 cout << "Choose output format :" <<"\n" <<"a) Spiral"<<"\n" <<"b) Sneak"<<"\n";  
 cin >> letter;  
 switch (letter){  
 case 'a':  
 if (system("CLS")) system("clear");  
 spiral(n, &arr[0][0]);  
 cout << "\n" << "\n";  
 break;  
 case 'b':  
 if (system("CLS")) system("clear");  
 snake(n, &arr[0][0]);  
 cout << "\n" << "\n";  
 break;  
 default:  
 cout <<"\t" << "Error" << "\n";  
 cin.clear();  
 while(cin.get() != '\n');  
 goto repeat1;  
 }  
 repeat2:  
 cout << "Choose permutation :" << "\n" << "1) a"<< "\n"<< "2) b"<< "\n"<< "3) c"<< "\n"<< "4) d"<< "\n";  
 cin >> letter;  
 switch (letter){  
 case 'a':  
 cout <<'\n';  
 permutationA(n,&arr[0][0]);  
 cout <<'\n';  
 break;  
 case 'b':  
 cout <<'\n';  
 permutationB(n,&arr[0][0]);  
 cout <<'\n';  
 break;  
 case 'c':  
 cout <<'\n';  
 permutationC(n,&arr[0][0]);  
 cout <<'\n';  
 break;  
 case 'd':  
 cout <<'\n';  
 permutationD(n,&arr[0][0]);  
 cout <<'\n';  
 break;  
 default:  
 cout <<'\n';  
 cout <<"\t" << "Error" << "\n";  
 cout <<'\n';  
 goto repeat2;  
 }  
 int \*end = &arr[0][0] + n \* n - 1;  
 cout << "Sorted array :" << '\n'<< '\n';  
 arr\_sort(&arr[0][0],end,n);  
  
 cout << "\n";  
  
 int value;  
 repeat3 :  
 cout << "Enter value"<<"\n";  
 cin >> value;  
 cout << "Enter operation"<< "\n";  
 cin >> letter;  
 switch (letter){  
 case '+':  
 op\_plus(&arr[0][0],end , n ,value);  
 break;  
 case '-':  
 op\_minus(&arr[0][0],end , n ,value);  
 break;  
 case '\*':  
 op\_multiplication(&arr[0][0],end , n ,value);  
 break;  
 case '/':  
 op\_division(&arr[0][0],end , n ,value);  
 break;  
 default:  
 cout <<'\n';  
 cout <<"\t" << "Error" << "\n";  
 cout <<'\n';  
 goto repeat3;  
 }  
 cout << "\n";  
 transposeMatrix(&arr[0][0],n);  
 system("pause");  
 return 0;  
}