**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 |  | Алексеев Г. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**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> |
|  | using namespace std; |
|  |  |
|  | const int n = 6; |
|  | void FillingArrRand(int Arr[n][n], const int N); |
|  | void OutPut(int Arr[n][n], const int N); |
|  | void FillingNewElements\_A(int Arr[n][n], const int N, int\* P); |
|  | void FillingNewElements\_B(int Arr[n][n], const int N, int\* P); |
|  | void Permutation\_A(int Arr[n][n], int FinalArr[n][n], const int N); |
|  | void Permutation\_B(int Arr[n][n], int FinalArr[n][n], const int N); |
|  | void Permutation\_C(int Arr[n][n], int FinalArr[n][n], const int N); |
|  | void Permutation\_D(int Arr[n][n], int FinalArr[n][n], const int N); |
|  | void Sorting(int Arr[n][n], int SortArr[n][n], const int N, int\* P); |
|  | void Calculation(int Arr[n][n], const int N, int\* P); |
|  |  |
|  | int main() |
|  | { |
|  | setlocale(0, ""); |
|  | int Mas[n][n], MasOfPermutation[n][n], MasSort[n][n], Number, \*p = \*Mas, \* pSortMas = \*MasSort; |
|  | cout << "Меню заданий:" << "\n" << "1 - Подстановка элементов в матрицу" << "\n" << "2 - Перестановка блоков матрицу" << "\n" << "3 - Сортировка матрицы" << "\n" << "4 - Взаимодействие числа с матрицей" << "\n"; |
|  | cout << "Введите номер задания: "; |
|  | cin >> Number; |
|  | switch (Number) |
|  | { |
|  | case(1): |
|  | system("cls"); |
|  | Number = 0; |
|  | cout << "Меню задания:" << "\n" << "1 - способ (а) заполнения элемента" << "\n" << "2 - способ (b) заполнения элемента" << "\n" << "Введите число: "; |
|  | cin >> Number; |
|  | if ((Number >= 1) and (Number <= 2)) |
|  | { |
|  | if (Number == 1) |
|  | { |
|  | system("cls"); |
|  | FillingNewElements\_A(Mas, n, p); |
|  | } |
|  | if (Number == 2) |
|  | { |
|  | system("cls"); |
|  | FillingNewElements\_B(Mas, n, p); |
|  | } |
|  | } |
|  |  |
|  | else |
|  | cout << "Ошибка: введена неправильна переменная" << "\n"; |
|  | break; |
|  | case(2): |
|  | system("cls"); |
|  | FillingArrRand(Mas, n); |
|  | Number = 0; |
|  | cout << "Меню задания:" << "\n" << "1 - способ (а) перестановки блоков массива" << "\n" << "2 - способ (b) перестановки блоков массива" << "\n" << "3 - способ (c) перестановки блоков массива" << "\n" << "4 - способ (d) перестановки блоков массива" << "\n" << "Введите число: "; |
|  | cin >> Number; |
|  | system("cls"); |
|  | cout << "Массив: " << "\n"; |
|  | OutPut(Mas, n); |
|  | cout << "Результат работы:" << "\n"; |
|  | if ((Number >= 1) and (Number <= 4)) |
|  | { |
|  | if (Number == 1) |
|  | Permutation\_A(Mas, MasOfPermutation, n); |
|  | if (Number == 2) |
|  | Permutation\_B(Mas, MasOfPermutation, n); |
|  | if (Number == 3) |
|  | Permutation\_C(Mas, MasOfPermutation, n); |
|  | if (Number == 4) |
|  | Permutation\_D(Mas, MasOfPermutation, n); |
|  | } |
|  | else |
|  | cout << "Ошибка: введена неправильна переменная" << "\n"; |
|  | break; |
|  | case(3): |
|  | FillingArrRand(Mas, n); |
|  | cout << "Массив:" << "\n"; |
|  | OutPut(Mas, n); |
|  | cout << "Отсортированный массив:" << "\n"; |
|  | Sorting(Mas, MasSort, n, pSortMas); |
|  | break; |
|  | case(4): |
|  | system("cls"); |
|  | Calculation(Mas, n, p); |
|  | break; |
|  | default: |
|  | cout << "Ошибка: введена неправильна переменная" << "\n"; |
|  | break; |
|  | } |
|  | } |
|  | void FillingArr(int Arr[n][n], const int N) |
|  | { |
|  | for (int i = 0; i < N; ++i) |
|  | { |
|  | for (int j = 0; j < N; ++j) |
|  | { |
|  | Arr[i][j] = 0; |
|  | } |
|  | } |
|  | } |
|  | void FillingArrRand(int Arr[n][n], const int N) |
|  | { |
|  | for (int i = 0; i < N; ++i) |
|  | { |
|  | for (int j = 0; j < N; ++j) |
|  | { |
|  | Arr[i][j] = rand() % (N\*N) - 1; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | void OutPut(int Arr[n][n], const int N) |
|  | { |
|  | for (int i = 0; i < N; ++i) |
|  | { |
|  | for (int j = 0; j < N; ++j) |
|  | { |
|  | cout << Arr[i][j] << " "; |
|  | } |
|  | cout << "\n"; |
|  | } |
|  | } |
|  | //Задание 1 |
|  | void FillingNewElements\_A(int Arr[n][n], const int N, int\* P) |
|  | { |
|  | FillingArr(Arr, N); |
|  | int count = 0, count\_index = 0; |
|  | while (count\_index < n \* n) |
|  | { |
|  | for (int i = count; i < n; ++i) |
|  | { |
|  | if (count\_index < n) |
|  | { |
|  | \*P = rand() % (n \* n) + 1; |
|  | OutPut(Arr, N); |
|  | if (count\_index != n - 1) |
|  | P = P + 1; |
|  | count\_index++; |
|  | Sleep(1000); |
|  | system("cls"); |
|  | } |
|  | else |
|  | { |
|  | P = P + 1; |
|  | \*P = rand() % (n \* n) + 1; |
|  | OutPut(Arr, N); |
|  | count\_index++; |
|  | Sleep(1000); |
|  | system("cls"); |
|  | } |
|  |  |
|  | } |
|  | count++; |
|  | for (int i = count; i < n; ++i) |
|  | { |
|  | P = P + n; |
|  | \*P = rand() % (n \* n) + 1; |
|  | OutPut(Arr, N); |
|  | count\_index++; |
|  | Sleep(1000); |
|  | system("cls"); |
|  |  |
|  | } |
|  | for (int i = count; i < n; ++i) |
|  | { |
|  | P = P - 1; |
|  | \*P = rand() % (n \* n) + 1; |
|  | OutPut(Arr, N); |
|  | count\_index++; |
|  | Sleep(1000); |
|  | system("cls"); |
|  |  |
|  | } |
|  | count++; |
|  | for (int i = count; i < n; ++i) |
|  | { |
|  | P = P - (n); |
|  | \*P = rand() % (n \* n) + 1; |
|  | OutPut(Arr, N); |
|  | count\_index++; |
|  | Sleep(1000); |
|  | system("cls"); |
|  | } |
|  | } |
|  | OutPut(Arr, N); |
|  | cout << "\n"; |
|  | } |
|  | void FillingNewElements\_B(int Arr[n][n], const int N, int\* P) |
|  | { |
|  | FillingArr(Arr, N); |
|  | int count\_index = 0; |
|  | while (count\_index < n \* n) |
|  | { |
|  | for (int i = 0; i < n; ++i) |
|  | { |
|  | \*P = rand() % (n \* n) + 1; |
|  | OutPut(Arr, N); |
|  | count\_index++; |
|  | if ((count\_index % n != 0) and (count\_index != 0)) |
|  | P = P + n; |
|  | Sleep(100); |
|  | system("cls"); |
|  | } |
|  | if (count\_index < N \* N) |
|  | P = P + 1; |
|  | for (int i = 0; i < n; ++i) |
|  | { |
|  | \*P = rand() % (n \* n) + 1; |
|  | OutPut(Arr, N); |
|  | count\_index++; |
|  | if (count\_index % n != 0) |
|  | P = P - n; |
|  | Sleep(100); |
|  | system("cls"); |
|  | } |
|  | if (count\_index < N \* N) |
|  | P = P + 1; |
|  | } |
|  | OutPut(Arr, N); |
|  | cout << "\n"; |
|  | } |
|  | //Задание 2 |
|  | void Permutation\_A(int Arr[n][n], int FinalArr[n][n], const int N) |
|  | { |
|  | for (int i = 0; i < N / 2; ++i) |
|  | { |
|  | for (int j = 0; j < N / 2; ++j) |
|  | { |
|  | FinalArr[i][j] = Arr[i + (N / 2)][j]; |
|  | FinalArr[i + (N / 2)][j] = Arr[i + (N / 2)][j + (N / 2)]; |
|  | FinalArr[i + (N / 2)][j + (N / 2)] = Arr[i][j + (N / 2)]; |
|  | FinalArr[i][j + (N / 2)] = Arr[i][j]; |
|  | } |
|  | } |
|  | OutPut(FinalArr, N); |
|  | cout << "\n"; |
|  | } |
|  | void Permutation\_B(int Arr[n][n], int FinalArr[n][n], const int N) |
|  | { |
|  | for (int i = 0; i < N / 2; ++i) |
|  | { |
|  | for (int j = 0; j < N / 2; ++j) |
|  | { |
|  | FinalArr[i][j] = Arr[i + (N / 2)][j + (N / 2)]; |
|  | FinalArr[i + (N / 2)][j + (N / 2)] = Arr[i][j]; |
|  | FinalArr[i + (N / 2)][j] = Arr[i][j + (N / 2)]; |
|  | FinalArr[i][j + (N / 2)] = Arr[i + (N / 2)][j]; |
|  | } |
|  | } |
|  | OutPut(FinalArr, N); |
|  | cout << "\n"; |
|  | } |
|  | void Permutation\_C(int Arr[n][n], int FinalArr[n][n], const int N) |
|  | { |
|  | for (int i = 0; i < N / 2; ++i) |
|  | { |
|  | for (int j = 0; j < N / 2; ++j) |
|  | { |
|  | FinalArr[i][j] = Arr[i + (N / 2)][j]; |
|  | FinalArr[i + (N / 2)][j] = Arr[i][j]; |
|  | FinalArr[i + (N / 2)][j + (N / 2)] = Arr[i][j + (N / 2)]; |
|  | FinalArr[i][j + (N / 2)] = Arr[i + (N / 2)][j + (N / 2)]; |
|  | } |
|  | } |
|  | OutPut(FinalArr, N); |
|  | cout << "\n"; |
|  | } |
|  | void Permutation\_D(int Arr[n][n], int FinalArr[n][n], const int N) |
|  | { |
|  | for (int i = 0; i < N / 2; ++i) |
|  | { |
|  | for (int j = 0; j < N / 2; ++j) |
|  | { |
|  | FinalArr[i][j] = Arr[i][j + (N / 2)]; |
|  | FinalArr[i][j + (N / 2)] = Arr[i][j]; |
|  | FinalArr[i + (N / 2)][j + (N / 2)] = Arr[i + (N / 2)][j]; |
|  | FinalArr[i + (N / 2)][j] = Arr[i + (N / |
|  | 2)][j + (N / 2)]; |
|  | } |
|  | } |
|  | OutPut(FinalArr, N); |
|  | cout << "\n"; |
|  | } |
|  | //Задание 3 |
|  | void Sorting(int Arr[n][n], int SortArr[n][n], const int N, int\* PSort) |
|  | { |
|  | FillingArrRand(Arr, N); |
|  | int Element = 0; |
|  | for (int i = 0; i < N; ++i) |
|  | { |
|  | for (int j = 0; j < N; ++j) |
|  | { |
|  | SortArr[i][j] = Arr[i][j]; |
|  | } |
|  | } |
|  | for (int i = 0; i < (N \* N); ++i) |
|  | { |
|  | for (int j = 0; j < (N \* N) - 1; ++j) |
|  | { |
|  | if (\*PSort > \*(PSort + 1)) |
|  | { |
|  | Element = \*(PSort + 1); |
|  | \*(PSort + 1) = \*PSort; |
|  | \*PSort = Element; |
|  | } |
|  | PSort = PSort + 1; |
|  | } |
|  | for (int j = 0; j < (N \* N) - 1; ++j) |
|  | { |
|  | if (\*PSort < \*(PSort - 1)) |
|  | { |
|  | Element = \*(PSort - 1); |
|  | \*(PSort - 1) = \*PSort; |
|  | \*PSort = Element; |
|  | } |
|  | PSort = PSort - 1; |
|  | } |
|  | } |
|  | OutPut(SortArr, N); |
|  | cout << "\n"; |
|  | } |
|  | //Задание 4 |
|  | void Calculation(int Arr[n][n], const int N, int\* P) |
|  | { |
|  | FillingArrRand(Arr, N); |
|  | int Number, Flag; |
|  | cout << "Введите число, с которым будут взаимодействовать элементы матрицы: "; |
|  | cin >> Number; |
|  | cout << "Меню:" << "\n" << "0 - Умножение" << "\n" << "1 - Деление" << "\n" << "2 - Сложение" << "\n" << "3 - Вычитание" << "\n"; |
|  | cout << "Введите число: "; |
|  | cin >> Flag; |
|  | system("cls"); |
|  | cout << "Матрица:" << "\n"; |
|  | OutPut(Arr, N); |
|  | switch(Flag) |
|  | { |
|  | case(0): |
|  | for (int i = 0; i < N\*N; ++i) |
|  | { |
|  | \*P = \*P \* Number; |
|  | P = P + 1; |
|  | } |
|  | cout << "Итог:" << "\n"; |
|  | OutPut(Arr, N); |
|  | break; |
|  | case(1): |
|  | for (int i = 0; i < N \* N; ++i) |
|  | { |
|  | \*P = \*P / Number; |
|  | P = P + 1; |
|  | } |
|  | cout << "Итог:" << "\n"; |
|  | OutPut(Arr, N); |
|  | break; |
|  | case(2): |
|  | for (int i = 0; i < N \* N; ++i) |
|  | { |
|  | \*P = \*P + Number; |
|  | P = P + 1; |
|  | } |
|  | cout << "Итог:" << "\n"; |
|  | OutPut(Arr, N); |
|  | break; |
|  | case(3): |
|  | for (int i = 0; i < N \* N; ++i) |
|  | { |
|  | \*P = \*P - Number; |
|  | P = P + 1; |
|  | } |
|  | cout << "Итог:" << "\n"; |
|  | OutPut(Arr, N); |
|  | break; |
|  | default: |
|  | cout << "Ошибка:введена неправильная переменная"; |
|  | break; |
|  | } |
|  | } |