**Практическое задание 13**

**Двумерные массивы. Функции работы с двумерными массивами.   
Обмен данными с текстовыми файлами**

Цель. Изучение представления двумерных массивов. Реализация алгоритмов работы с двумерными массивами на основе функций. Обмен данными с текстовыми файлами.

**ПЛАН**

[1. Краткое теоретическое введение 1](#_Toc26108124)

[2. Использование функций при обработке матриц 2](#_Toc26108125)

[3. Обмен данными с текстовыми файлами 3](#_Toc26108126)

1. Краткое теоретическое введение

Синтаксис С++ разрешает использовать только одномерные массивы. Двумерные массивы трактуются как массивы массивов.

**Объявление** двумерного массива требует, чтобы были указаны два размера: число строк и число столбцов матрицы в константном выражении, например:

int a[**8**][**10**]; // Массив из 8-ми одномерных массивов,

// в каждом из которых 10 элементов.

При объявлении массива в памяти выделяется место под размещение его элементов. Элементы двумерного массива размещаются по строкам линейно по возрастанию индексов. Индексы нумеруются с нуля.

**Инициализация** двумерных массивов синтаксически выглядит как инициализация нескольких одномерных массивов, например:

int a[3][4] ={

{1, 2, 3, 4}, // Значения элементов нулевой строки.

{2, 4, 6, 8}, // Значения элементов первой строки.

{9, 8, 7, 6} // Значения элементов второй строки.

};

При **размещении** в памяти будет выделено место под запись 3 \* 4 = 12 элементов, и они будут размещены линейно в следующем порядке:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **2** | **4** | **6** | **8** | **9** | **8** | **7** | **6** |
| *Нулевая строка* | | | | *Первая строка* | | | | *Вторая строка* | | | |

Как и одномерные массивы, матрицы могут быть **условно переменной длины**. При описании матрицы число строк и число столбцов должно быть константой: целочисленные значения. Они определяют механизм выделения памяти для матрицы.

Реально из выделенного пространства можно использовать меньшее число данных. Для матрицы условно переменной длины следует описать размер по максимуму, и ввести переменные, реально определяющие размер матрицы.

int a[25][25]; // Максимальное число строк, столбцов.

int n, m; // Реальное число строк, столбцов.

…

cout << "Введите размер матрицы (строк, столбцов <25)\n";

cin >> n >> m;

**Обращение к элементу** массива aij можно выполнить по индексу (прямая адресация) или указателю (косвенная адресация):

a[i][j] // Адресуется элемент массива, стоящий на

// пересечении i - того столбца и j - той строки.

**Алгоритмы** работы с двумерными масивами, в которых требуется обращение ко всем элементам матрицы, основаны на использовании вложенного цикла, у которого во внешнем цикле управляющей переменной служит номер строки матрицы, а во внутреннем номер столбца. Тогда просмотр элементов происходит построчно.

Для обращения к элементам матрицы по строкам цикл записывается так:

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

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

{

// В теле цикла обращение к переменной a[i][j].

}

Если переменить эти циклы местами, просмотр будет происходить по столбцам:

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

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

{

// В теле цикла обращение к переменной a[i][j].

}

2. Использование функций при обработке матриц

При разработке функций обработки матриц можно использовать два различных подхода: матрица как матрица или матрица как массив строк.

В первом случае матрица существует и рассматривается как самостоятельная структура данных, к которой необходимо применить какой-либо алгоритм обработки. В качестве параметра функции передается вся матрица целиком. Чтобы задача обработки решалась в общем виде, в функцию через параметры следует передать имя матрицы и ее размеры, при этом функция получает матрицу как двумерный массив. С++ должен знать, каков способ разбиения этой структуры на строки, поэтому число строк данных можно опустить, а число данных в каждой строке опускать нельзя. Оно должно быть указано обязательно как константное выражение в квадратных скобках при передаче матрицы. Так, прототип функции, получающей матрицу в качестве входного данного, может выглядеть так:

int function (int a[][**M**], int n, int m); // **M** – константное выражение.

**Пример.** Функции для ввода и вывода матрицы на экран описаны в заголовочном файле M\_Array.h. Предполагается, что матрица условно переменного размера, поэтому число строк и столбцов матрицы по описанию определено константами N = 4 и M = 5. Реальный размер матрицы определяют переменные n, m.

void input\_matr (int a[][**M**], int &n, int &m);

void print\_matr (int a[][**M**], int n, int m);

Функция main описывает входные данные и передает управление в функции обработки.

int n, m; // Реальные размеры матрицы.

int matr [N][M]; // Описан двумерный массив.

input\_matr(matr, n, m); // Передан в функцию ввода.

print\_matr(matr, n, m); // Предан в функцию вывода.

Во втором случае, если матрица, это массив из одномерных массивов, то для обработки отдельных строк матрицы можно использовать функции обработки одномерных массивов. Зная, что элементы матрицы, это одномерные массивы, каждый из них можно по очереди передавать в функцию, которая умеет работать с одномерным массивом.

В заголовочном файле M\_Array.h описаны функция вывода одномерного массива, и функция преобразования, которая находит наименьшее значение и переставляет его на первое место.

void print\_arr(int arr[], int len)

void Change (int arr[], int len)

К этим функциям можно обратиться в цикле, поочередно передавая строки матрицы как одномерные массивы. Цикл управления находится в основной программе.

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

Change (matr[i], m); // matr[i], это i – тая строка матрицы.

Функция вывода матрицы вызывается после завершения обработки.

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

print\_matr (matr, n, m);

3. Обмен данными с текстовыми файлами

Для работы с файлами необходимо подключить заголовочный файл <fstream>, в котором определены классы ввода и вывода, и подключены заголовочные файлы <ifstream> − файловый ввод и <ofstream> − файловый вывод.

Для записи в файл используется объект ofstream, для чтения – объект ifstream.

На консоль ввод и вывод выполняются с помощью объектов cin и cout. Файловый ввод/вывод полностью аналогичен вводу/выводу на консоль, если создать объекты для переназначения ввода/вывода в файл:

* создать объект класса ofstream/ifstream**;**
* связать объект класса с файлом, в который будет производиться запись/чтение;
* записать/прочитать данные в файл;
* закрыть файл.

**Запись в файл**

Для записи в файл к объекту fstream применяется оператор << , как и при выводе на консоль:

#include <fstream>

…

ofstream out; // out - поток для записи.

out.open("Data.txt"); // Открыть файл для записи.

// Вывести данные в поток out

out.close(); // Чтобы сохранить.

Если физически файл с именем Data.txt не существует, он будет создан. Если такой файл есть, данные будут записаны заново.

**Чтение из файла**

Для чтения данных из файла для объектов ifstream может применяться оператор >> , как и при чтении с консоли:

#include <fstream>

 …

ifstream in("Data.txt"); // Открыть файл для чтения

if (in.is\_open()) {

// Прочитать данные из потока: файл Data.txt.

}

in.close(); // Закрыть файл.

**Задание**

Заголовочный файл M\_array.h скопируйте в папку проекта и присоедините к проекту, откройте и ознакомьтесь с содержимым.

**Упражнение 1. Чтение матрицы из файла**

Создайте в IDE текстовый файл такого содержания:

3 4

1 2 3 4

11 22 33 44

9 8 7 6

Сохраните с именем «Data.txt». В первой строке зписаны число строк и столбцов матрицы.

В main объявите двумерный массив. Прочитайте данные функцией чтения. Выведите на консоль функцией вывода.

**Упражнение 2. Запись матрицы в файл**

Полученную матрицу выведите в файл с именем «Data\_Out.txt», используя функцию вывода. Откройте файл в IDE, убедитесь, что данные записаны.

**Упражнение 3. Алгоритмы обработки двумерных массивов**

Напишите функцию Change\_Matrix\_One, изменяющую матрицу по следующему алгоритму: найти среднее арифметическое значение в каждой строке и дописать его в конец каждой строки. Выведите матрицу в файл.

**Упражнение 4. Алгоритмы обработки двумерных массивов**

Напишите функцию Change\_Matrix\_Two, изменяющую матрицу по следующему алгоритму: найти наибольшее значение в каждом столбце и дописать его в конец каждого столбца. Выведите матрицу в файл.

**Выводы**

1. Двумерные массивы, это массивы одномерных массивов. Хранятся по строкам в линейной области адресного пространства. Реальный размер может быть меньше, чем тот, что указан в объявлении матрицы.

2. Функция, получающая двумерный массив, должна знать, каким образом в линейной последовательности хранимых значений опознать одну строку. Для этого в списке формальных параметров ей передается константа = длине строки:

void print\_matr(int matr[][**M**], int n, int m) // **M**, это длина строки-константа

3. При вызове функции на стороне main ей передаются реальные данные:

input\_matr(matr,n,m); // matr – имя матрицы, n,m – ее реальные размеры.