**Основные понятия**

Массив - структурированный тип данных с элементами одного и того же типа, имеющий одно имя и определенное количество элементов, расположенных в памяти строго последовательно. Количество элементов определяет размер массива. Порядковый номер элемента массива называется его индексом. Число индексов называется размерностью массива, например, массив с двумя индексами называется двумерным массивом. Вектор – одномерный массив элементов, матрица – массив векторов.

Обработка массивов выполняется обычно следующим образом: объявление, ввод или инициализация элементов массива, преобразование и вывод.

**Объявление массива**

Объявить массив – выделить место в памяти компьютера, объём которой зависит от количества элементов и типа массива. Тип массива - это тип входящих в него элементов. Массивы могут быть разных типов: int, float, char, и т.д. Массив объявляют так же, как и обычные переменные, но после имени массива в квадратных скобках записывается его размер.

int A[10], B[20]; // 2 массива по 10 и 20 целых чисел

float C[12]; // массив из 12 вещественных чисел

При объявлении массива можно сразу заполнить его начальными значениями (проинициализировать), перечисляя их внутри фигурных скобок:

int A[4] = { 2, 3, 12, 76 };

Если в списке в фигурных скобках записано меньше чисел, чем элементов в массиве, то оставшиеся элементы заполняются нулями. Если чисел больше, чем надо, компилятор сообщает об ошибке. Например,

int A[4] = { 2 }; // последние три элемента равны 0

int A[4] = { 2, 3, 12, 76, 0 }; // ошибка, должно быть не более 4-х элементов

Для повышения универсальности программы размер массива лучше определять через константу. В этом случае для переделки программы для массива другого размера надо только поменять значение этой константы:

const int N = 20; //N - константа

main()

{

int A[N]; // размер массива задан через константу

...

}

**Обращение к элементу массива**

Каждый элемент массива имеет свой порядковый номер. Чтобы обратиться к элементу массива, надо написать имя массива и затем в квадратных скобках номер нужного элемента. Важно запомнить правило: элементы массивов в языке Си нумеруются с нуля. Поэтому индекс последнего элемента массива на 1 меньше числа элементов в данном массиве.

Таким образом, если в массиве 10 элементов, он содержит элементы:

**A[0], A[1], A[2], ..., A[9]**

Номер элемента массива также называется его **индексом**. Вот примеры обращения к массиву ***A***:

**x = (A[3] + 5)\*A[1]; // прочитать значения A[3] и A[1]**

**A[0] = x + 6; // записать новое значение в A[0]**

В языке Си не контролируется **выход за границы массива,** то есть формально вы можете записать что-то в элемент с несуществующим индексом, например в **A[345]** или в **A[-12]**. Однако при этом вы стираете какую-то ячейку в памяти, не относящуюся к массиву, поэтому последствия такого шага непредсказуемы и во многих случаях программа «зависает» или «падает» с ошибкой по памяти.

**Ввод с клавиатуры и вывод на экран**

Существует много способов ввода в зависимости от вашей задачи:

• элементы массива вводятся с клавиатуры вручную;

• массив заполняется случайными числами (например, для моделирования случайных процессов);

• элементы массива читаются из файла;

• массив заполняется в процессе вычислений.

Чтобы ввести массив в память, надо каждый его элемент обработать отдельно (например, вызвав для него функцию ввода **scanf**). Для ввода и вывода массива обычно используется цикл **for**.

**Пример обработки одномерного массива**

Дан массив из 50 целых чисел. Найти наибольший элемент в массиве и его порядковый номер.

# include <stdio.h>

#include <conio.h>

# define n 50 // определение константы n=50

int i, max, nom, a[n]; //описание массива целых чисел из n элементов

main()

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

{ printf( “\n Введите элемент массива ”);

scanf ("%d", &a[i]) };

for (i=1,max=a[0],nom=0; i<n; i++)

if (max<a[i])

{nom=i; max=a[i];}

printf("\n Вывод элементов исходного массива : \n");

for (i=0; i<n; i++) printf ( "%6d", a[i] );

printf ("\n Максимальное число в массиве %4d, его индекс %4d " , max, nom+1);

getch();

}

**Заполнение случайными числами**

Пусть требуется заполнить массив равномерно распределенными случайными числами в интервале **[a,b]**. Поскольку для целых и вещественных чисел способы вычисления случайного числа в заданном интервале отличаются, рассмотрим оба варианта.

Описание функции-датчика случайных чисел находится в заголовочном файле **stdlib.h**. Получить случайные числа с равномерным распределением в интервале **[0,N-1]**:

**rand() % N;**

Для получения случайных целых чисел с равномерным распределением в интервале **[a,b]** надо использовать формулу

**k = rand % ( b – a + 1 ) + a;**

Для вещественных чисел формула несколько другая:

**x = rand()\*(b - a)/RAND\_MAX + a;**

Здесь константа **RAND\_MAX** – это максимальное случайное число, которое выдает стандартная функция **rand**.

В приведенном ниже примере массив A заполняется случайными целыми числами в интервале [-5,10], а массив X - случайными вещественными числами в том же интервале.

#include <stdlib.h>

#include <stdio.h>

const int N = 10;

int random (int N) { return rand()%N; }

main()

{ system("CLS");

int i, A[N], a = -5, b = 10;

float X[N];

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

A[i] = random(b-a+1) + a;

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

X[i] = (float)rand()\*(b-a)/RAND\_MAX + a;

printf("\nЦелые\n");

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

printf("%4d",A[i]);

printf("\nВещественные\n");

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

printf("%6.2f",X[i]);

system("PAUSE");

return 0;

}

Многомерные массивы задаются указанием каждого измерения в квадратных скобках. Например, оператор

int matr [6] [8];

задает описание матрицы из 6 строк и 8 столбцов. Нумерация строк и столбцов начинается с 0. При инициализации двумерного массива он представляется как массив из массивов, при этом каждый массив заключается в свои фигурные скобки, либо задается общий список в том порядке, в котором элементы располагаются в памяти:

int mas [] [2]= { {1,2}, {0,2}, {1,0}};

int mas [3][2]={1,2,0,2,1,0};

Если задать не все элементы, то остальные заполнятся нулями:

float X[2][3] = {{1., 3.},{6.}};

Здесь элементы X[1][2], X[2][1] и X[2][2] будут нулевыми.

Расположение двумерных массивов в памяти.

Иногда, бывает полезно знать, как матрицы располагаются в памяти ЭВМ. Оказывается, во всех современных языках программирования (кроме Фортрана) элементы матрицы располагаются по строкам, то есть сначала изменяется последний индекс. Объявленная выше матрица X расположена так:

X[0][0] X[0][1] X[0][2] X[1][0] X[1][1] X[1][2]

Ввод/вывод многомерных массивов

Единственная проблема состоит в том, чтобы не перепутать переменные в двух циклах и пределы их изменения.

const int M = 5; // число строк

const int N = 4; // число столбцов

main() {

int i, j, A[M][N];

for ( i = 0; i < M; i ++ ) // цикл по строкам

for ( j = 0; j < N; j ++ ) // цикл по столбцам строки

{

printf ("A[%d][%d]=", i, j); // подсказка для ввода

scanf ("%d", & A[i][j]); // ввод A[i][j]

}

// работа с матрицей

}

Заметьте, что при изменении порядка циклов (если поменять местами два оператора for) изменится и порядок ввода элементов в память.

При выводе матрицы ее элементы желательно расположить в привычном виде – по строкам. Напрашивается такой прием: вывели одну строку матрицы, перешли на новую строку экрана, и т.д. Надо учитывать, что для красивого вывода на каждый элемент матрицы надо отвести равное количество символов (иначе столбцы будут неровные). Делается это с помощью форматирования – цифра после знака процента задает количество символов, отводимое на данное число.

printf("Матрица A\n");

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

{ // цикл по строкам

for ( j = 0; j < N; j ++ ) // вывод одной строки (в цикле)

printf ( "%4d", A[i][j] ); // 4 символа на число

printf("\n"); // переход на другую строку

}