**Статические массивы, матрицы. Передача массивов как параметров в подпрограммы**

*Статический массив*

Объявление статического массива отличается от объявления обычной переменной только указанием количества элементов массива. Например, следующее объявление означает, что именем points называется массив из 100 действительных чисел.

**double** points[100];

В некотором смысле можно считать, что такое объявление переменной points создает 100 переменных, которые называются points[0], points[1], ..., points[99]. Плюс к этому, "имена" этих переменных можно вычислять: points[1], points[0+1] или points[k-1] имеют одно значение (если k=2).

В реальных программах следует избегать явного использования числовых констант в объявлениях массива (и других частях программы). Если нам нужно объявить два массива, которые теоретически могут иметь разный размер, например,

**double** points[100];

**int** students[100];

то в дальнейшем, если возникнет необходимость увеличить один из массивов, будет сложно отличить одну константу от другой. Особенно это верно при обработке элементов массива (см. ниже). Правильным считается использование директив препроцессора для присвоения константам "говорящих" имен. Например:

#define NPOINTS 100

#define NSTUDENTS 100

...

**double** points[NPOINTS];

**int** students[NSTUDENTS];

Объявление массива может быть совмещено с присвоением значений его элементам. Например,

**double** points[] = {1.0, 3.14, -1.2, 12.65};

создает массив из четырех действительных чисел с указанными значениями. Заметим, что в данном случае число элементов массива в квадратных скобках не указывается. Компилятор самостоятельно вычисляет длину по списку начальных значений. В программе можно вычислить длину такого массива, разделив его размер на размер одного элемента

*Статические матрицы*

Матрица – это двумерный массив, каждый элемент которого имеет два индекса: номер строки – i; номер столбца – j.

Статический двумерный массив (матрицу) можно объявить так:

тип имя\_переменной [n][m];

где тип определяет тип элементов массива, имя\_переменной – имя матрицы, n – количество строк, m – количество столбцов в матрице. Строки нумеруются от 0 до n-1, столбцы – от 0 до m-1.

Например,

double x[20][35];

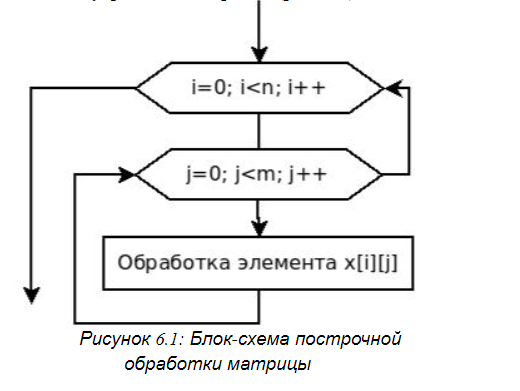
Описана матрица вещественных чисел x, состоящая из 20 строк и 35 столбцов

(строки нумеруются от 0 до 19, столбцы от 0 до 34).

Для обращения к элементу матрицы необходимо указать ее имя, и в квадратных скобках номер строки, а затем в квадратных скобках – номер столбца. Например,

x[2][4] – элемент матрицы x, находящийся в третьей строке и пятом столбце38.

Для работы с элементами матрицы необходимо использовать два цикла. Для построчной обработки матрицы значениями параметра первого (внешнего) цикла будут номера строк матрицы, значениями параметра второго (внутреннего) цикла – номера столбцов (рис. 6.1). При построчной обработке матрицы вначале поочередно рассматриваются элементы первой строки (столбца), затем второй и т.д. до последней. Если необходимо обрабатывать матрицу по столбцам, то необходимо организовать внешний цикл по столбцам, а внутренний по строкам



*Передача массивов как параметров в подпрограммы*

Первый способ – безразмерный (динамический) массив. Записывается он так:  
Procedure One (Arr: array of integer);  
Как Вы видите, размерность массива здесь не указана и таким образом, она будет подобрана автоматически, в зависимости от передаваемого массива. Это очень удобно, поскольку позволяет использовать этот способ для передачи в одну процедуру массивов разного размера.  
Недостатком такого способа является то, что его нельзя использовать для многомерных массивов, этим способом можно передать лишь одномерный массив. Другой недостаток – это сбивание нумерации элементов. Поскольку номера ни первого ни последнего элемента не заданы, Pascal назначает их автоматически и делает он это начиная с нуля. Таким образом, если нумерация Вашего массива была от 1 до N, то после передачи в подпрограмму она станет от 0 до N-1. Это нужно не забывать и учитывать.  
Второй способ – это создание нового типа переменной! В параметрах основной программы (после const но до var), можно создавать свой тип переменной, сделав его массивом заданного размера. Делается это так:  
Type Имя\_типа = array [1..N,1..M] of real  
После этого Вы объявляете массив присваивая ему уже этот, новый тип и не указывая размерность:  
Mas: Имя\_типа  
Если представить, что имя типа мы задали Massiv, то объявление подпрограммы будет выглядеть так:  
Procedure One (Arr: massiv);  
Такой способ позволяет передавать в качестве параметра массивы с любым количеством измерений и многомерные массивы можно передать только так. Недостатком же этого способа является то, что количество измерений и размеры передаваемых массива должны совпадать с принимаемым массивом, т.е. этим способом в подпрограмму можно передать лишь одинаковые массивы.

Литература

<https://studfile.net/preview/5680538/page:34/>

<https://www.cyberforum.ru/pascal/thread1575880.html>