Регулярный тип (массив).

1. Определение массива.

В ряде случаев приходится сталкиваться с ситуацией, когда должно использоваться относительно много переменных одного типа.

Представим себе программу, которая вычисляет метеорологические данные (температура, влажность, давление, осадки и т.п.) за год. Т.е. в этой программе всегда будет не меньше 365 переменных. Можно описать эти 365 (366) переменных следующим образом:

var day1, day2, day3, …, day365: real;

но уже только одна такая запись будет занимать больше страницы, не говоря о том, чтобы посчитать, например, среднюю температуру за год.

Для таких и подобных этому случаев в языке Паскаль предусмотрена возможность введения большого числа переменных одного и того же типа за счет использования структурированного типа массив. Следует отметить, что подобный тип данных есть практически во всех языках программирования.

Массив – это упорядоченный набор перенумерованных элементов одного типа.

Элементы массива характеризуются следующими *свойствами*:

1) все элементы упорядочены;
2) обращение к любому элементу массива осуществляется по его индексу;
3) количество элементов определяется один раз при описании массива и далее не может быть превышено;
4) все элементы массива принадлежат к одному типу данных, называемому ***базовым типом массива***.

Все элементы массива имеют общее имя – ***имя массива***, а указание на конкретный элемент осуществляется с помощью ***индекса*** – порядкового номера элемента.

Для описания объекта типа массив в языке Паскаль используется следующая конструкция:

array [<тип1>] of <тип2>;

где тип1 – тип индекса;

тип2 – тип элементов (базовый тип).

2. Типы индексов.

В качестве типа индекса могут быть использованы такие типы, как:

1. Ограниченный.

Пример.

array [1..n] of integer;
array [1146..2004] of real;
array [-754..-1] of integer;

В этом случае обращение к элементам массива осуществляется через указание имени массива и в квадратных скобках его индекса – любого числа из представленного интервала.

Пример.

При описании следующего вида

```
type tula = array [1146..2004] of real;
var stat: tula;
```

возможны обращения к элементам массива:

```
stat [1675]        stat [1980]        stat [2000]
```

Описать массив можно и другим способом:

```
var stat1: array [2000..2004] of real;
```

2. Перечислимый.

Пример.

```
type m = (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec);
var t: array [m] of integer;
```

Тогда обращение к элементам массива будет следующим:

```
t[Jan]                t[Oct]
```

т.е. в качестве индекса выступает любое значение из определенного ранее набора.

3. Булевский.

В этом случае индексные выражения могут представлять собой логические выражения.

Пример.

```
var k: array [boolean] of integer; a,b: boolean;
```

Обращение к элементам массива может быть следующим:

```
k[false]              k[true]              k[a or b]
```

4. Символьный.

Пример.

```
type s = array [char] of integer;
var m: s; c: char;
```

К элементам массива можно обратиться следующим образом:

```
m[‘d’]                m[‘z’]                m[pred(c)]
```

В качестве типа индекса не могут быть использованы типы integer и real, т.к. множества значений этих типов в языке Паскаль не ограничены (тип integer ограничен лишь возможностями техники), а real еще и не упорядочено.

При объявлении типа индексов массива удобно использовать константы:

```
const n = 10;
type m = array[1..n] of integer;
```

3. Размерность массива.

Количество индексов в обозначении элемента массива определяет ***размерность*** массива. Массив может быть одномерным, двумерным, трехмерным и т.д. При описании таких массивов должен быть описан тип каждого индекса.

Пример.

```
var m: array[1..10, 1..15] of real;
m[5,8]                m[1,10]
```

В этом случае мы имеем дело с прямоугольной таблицей, состоящей из 15 столбцов и 10 строк.

В языке Паскаль число размерностей неограниченно. Например, выражение

```
var a: array[1..1000, 1..35, 1..20, 1..70] of real;
```

совершенно правильно с точки зрения синтаксиса языка, хотя можно сомневаться, хватит ли у компьютера памяти для такого количества переменных.

Замечание.

Размерность массивов на практике ограничена размером памяти компьютера.

4. Организация работы с массивами.

Для организации работы с массивом необходимо:

1) *объявить* (или описать) массив, т.е. задать имя, типы индексов и элементов;

2) *сгенерировать* (или сформировать) массив, т.е. задать значения элементов массива. Среди способов генерации массивов можно выделить следующие: с клавиатуры, из памяти, с помощью датчика случайных чисел, по какому-либо закону (формуле) и др.;

3) *визуализировать* массив, т.е. вывести на экран первоначальные значения элементов массива;

4) *обработать* элементы массива в соответствии с задачей;

5) вывести на экран *результат* выполненных действий.

Действия по формированию и визуализации массивов удобнее оформить в виде отдельных *процедур*.

5. Массивы констант.

В объявлении массива констант указываются значения его элементов, заключенные в круглые скобки и отделены друг от друга запятыми.

Пример.

```
type color = (Red, Blue, Green);
     palette = array [color] of string[10];
const m: palette = (‘Красный’, ‘Синий’, ‘Зеленый’);
```

В результате элементы массива m приобретают следующие значения:

```
m[Red] = ‘Красный’;
m[Blue] = ‘Синий’;
```

m[Green] = ‘Зеленый’;

Элементы такого массива удобно задавать не как одиночные символы, а в виде строковой константы. Например, объявление константы

const digits: array [0..9] of char = (‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’)

можно представить в более компактной форме:

const digits: array [0..9] of char = ‘0123456789’;

Чтобы обратиться к элементу константы типа массив необходимо указать ее имя и затем в квадратных скобках – требуемое число индексов, которые могут задаваться выражениями соответствующих типов.