**Тема: Массивы.**

План занятия:

1. Понятие массива

2. Описание массива

3. Одномерные и двумерные массивы

До сих пор мы рассматривали переменные, которые имели только одно значение, могли содержать в себе только одну величину определенного типа. Исключением являлись лишь строковые переменные, которые представляют собой совокупность данных символьного типа, но и при этом мы говорили о строке, как об отдельной величине.

Вы знаете, что компьютер предназначен в основном для облегчения работы человека с большими информационными объемами. Как же, используя только переменные известных вам типов, сохранить в памяти и обработать данные, содержащие десяток, сотню, тысячу чисел или, к примеру, строк? А ведь такие задачи встречаются в любой области знания. Конечно, можно завести столько переменных, сколько данных, можно даже занести в них значения, но только представьте, какой величины будет текст такой программы, сколько времени потребуется для его составления, как много места для возможных ошибок? Естественно, об этом задумывались и авторы языков программирования. Поэтому во всех существующих языках имеются типы переменных, отвечающие за хранение больших массивов данных. В языке Паскаль они так и называются: "массивы".

Массивом будем называть упорядоченную последовательность данных одного типа, объединенных под одним именем. Кстати, под это определение подходит множество объектов из реального мира: словарь (последовательность слов), мультфильм (последовательность картинок) и т. д. Проще всего представить себе массив в виде таблицы, где каждая величина находится в собственной ячейке. Положение ячейки в таблице должно однозначно определяться набором координат (индексов). Самой простой является линейная таблица, в которой для точного указания на элемент данных достаточно знания только одного числа (индекса). Мы с вами пока будем заниматься только линейными массивами, так как более сложные структуры строятся на их основе.

Описание типа линейного массива выглядит так:
Type <Имя типа>=Array [<Диапазон индексов>] Of <Тип элементов>;

В качестве индексов могут выступать переменные любых порядковых типов. При указании диапазона начальный индекс не должен превышать конечный. Тип элементов массива может быть любым (стандартным или описанным ранее).

Описать переменную-массив можно и сразу (без предварительного описания типа) в разделе описания переменных:
Var <Переменная-массив> : Array [<Диапазон индексов>] Of <Тип элементов>;

Примеры описания массивов:
*Var
S, BB : Array [1..40] Of Real;
N : Array ['A'..'Z'] Of Integer;
R : Array [-20..20] Of Word;
T : Array [1..40] Of Real;*

Теперь переменные S, BB и T представляют собой массивы из сорока вещественных чисел; массив N имеет индексы символьного типа и целочисленные элементы; массив R может хранить в себе 41 число типа Word.

Единственным действием, которое возможно произвести с массивом целиком - присваивание. Для данного примера описания впоследствии допустима следующая запись:
S:=BB;

Однако, присваивать можно только массивы одинаковых типов. Даже массиву T присвоить массив S нельзя, хотя, казалось бы, их описания совпадают, произведены они в различных записях раздела описания.

Никаких других операций с массивами целиком произвести невозможно, но с элементами массивов можно работать точно также, как с простыми переменными соответствующего типа. Обращение к отдельному элементу массива производится при помощи указания имени всего массива и в квадратных скобках - индекса конкретного элемента. Например:
R[10] - элемент массива R с индексом 10.

Фундаментальное отличие компонента массива от простой переменной состоит в том, что для элемента массива в квадратных скобках может стоять не только непосредственное значение индекса, но и выражение, приводящее к значению индексного типа. Таким образом реализуется косвенная адресация:
BB[15] - прямая адресация;
BB[K] - косвенная адресация через переменную K, значение которой будет использовано в качестве индекса элемента массива BB.

Такая организация работы с такой структурой данных, как массив, позволяет использовать цикл для заполнения, обработки и распечатки его содержимого.

Если вы помните, с такой формой организации данных мы встречались, когда изучали строковые переменные. Действительно, переменные типа String очень близки по своим свойствам массивам типа Char. Отличия в следующем: строковые переменные можно было вводить с клавиатуры и распечатывать на экране (с обычным массивом это не проходит); длина строковой переменной была ограничена 255 символами (255 B), а для размера массива критическим объемом информации является 64 KB.

Теперь рассмотрим несколько способов заполнения массивов и вывода их содержимого на экран. В основном мы будем пользоваться числовыми типами компонент, но приведенные примеры будут справедливы и для других типов (если они допускают указанные действия).

*Program M1;
Var
A : Array [1..20] Of Integer;
Begin
A[1]:=7; {Заполняем массив значениями (отдельно каждый компонент)}
A[2]:=32;
A[3]:=-70;
.............. {Трудоемкая задача?}
A[20]:=56;
Writeln(A[1],A[2],A[3], ?,A[20])
End.*

Как бы ни был примитивен приведенный пример, он все же иллюстрирует возможность непосредственного обращения к каждому элементу массива отдельно. Правда, никакого преимущества массива перед несколькими простыми переменными здесь не видно. Поэтому - другой способ:

*Program M2;
Var A : Array [1..20] Of Integer;
I : Integer;
Begin
For I:=1 To 20 Do {Организуем цикл с параметром I по всем возможным}
Readln(A[I]); {значениям индексов и вводим A[I] с клавиатуры }
For I:=20 Downto 1 Do {Распечатываем массив в обратном порядке}
Write(A[I],'VVV')
End.*

Эта программа вводит с клавиатуры 20 целых чисел, а затем распечатывает их в обратном порядке. Теперь попробуйте написать такую же программу, но без использования структуры массива. Во сколько раз она станет длиннее? Кстати, введение язык Паскаль цикла с параметром было обусловлено во многом необходимостью обработки информационных последовательностей, т. е. массивов.

Следующая программа заполняет массив значениям квадратов индексов элементов:

*Program M3;
Const
N=50; {Константа N будет содержать количество элементов массива}
Var A : Array [1..N] Of Integer;
I : Integer;
Begin
For I:=1 To N Do
A[I]:=I\*I
For I:=1 To N Do
Write(A[I],'VVV')
End.*

В дальнейшем для учебных целей мы будем использовать массивы, заданные с помощью генератора случайных чисел. В языке Паскаль случайные числа формирует функция Random. Числа получаются дробными, равномерно расположенными в интервале от 0 до 1. Выражение, дающее целое случайное число в интервале [-50,50] будет выглядеть так:
Trunc(Random\*101)-50

Зададим и распечатаем случайный массив из сорока целых чисел:

*Program M4;
Const
N=40; {Константа N будет содержать количество элементов массива}
Var
A : Array [1..N] Of Integer;
I : Integer;
Begin
For I:=1 To N Do
Begin
A[I]:= Trunc(Random\*101)-50
Write(A[I],'VVV')
End
End.*

С обработкой линейных массивов связано множество задач. Их мы рассмотрим на практических занятиях.

**Двумерные и многомерные массивы**
Представьте себе таблицу, состоящую из нескольких строк. Каждая строка состоит из нескольких ячеек. Тогда для точного определения положения ячейки нам потребуется знать не одно число (как в случае таблицы линейной), а два: номер строки и номер столбца. Структура данных в языке Паскаль для хранения такой таблицы называется двумерным массивом. Описать такой массив можно двумя способами:
I.
*Var A : Array [1..20] Of Array [1..30] Of Integer;*

II. *Var A : Array [1..20,1..30] Of Integer;*

В обоих случаях описан двумерный массив, соответствующий таблице, состоящей из 20 строк и 30 столбцов. Приведенные описания совершенно равноправны.

Отдельный элемент двумерного массива адресуется, естественно, двумя индексами. Например, ячейка, находящаяся в 5-й строке и 6-м столбце будет называться A[5][6] или A[5,6].

Для иллюстрации способов работы с двумерными массивами решим задачу: "Задать и распечатать массив 10X10, состоящий из целых случайных чисел в интервале [1,100]. Найти сумму элементов, лежащих выше главной диагонали."

При отсчете, начиная с левого верхнего угла таблицы, главной будем считать диагональ из левого верхнего угла таблицы в правый нижний. При этом получается, что элементы, лежащие на главной диагонали будут иметь одинаковые индексы, а для элементов выше главной диагонали номер столбца будет всегда превышать номер строки. Договоримся также сначала указывать номер строки, а затем - номер столбца.

*Program M5;
Var A : Array[1..10,1..10] Of Integer;
I, K : Byte;
S : Integer;
Begin
S:=0;
For I:=1 To 10 Do
Begin
For K:=1 To 10 Do
Begin
A[I,K]:=Trunc(Random\*100)+1;
Write(A[I,K]:6);
If K>I Then S:=S+A[I,K]
End;
Writeln
End;
Writeln('Сумма элементов выше гл. диагонали равнаV',S)
End.*

Если модель данных в какой-либо задаче не может свестись к линейной или плоской таблице, то могут использоваться массивы произвольной размерности. N-мерный массив характеризуется N индексами. Формат описания такого типа данных:
Type

<Имя типа>=Array[<диапазон индекса1>,<диапазон индекса2>,...
<диапазон индекса N>] Of <тип компонент>;

Отдельный элемент именуется так:
<Имя массива>[<Индекс 1>,<Индекс 2>,...,<Индекс N>]

Список источников:

1. Абрамов В.Г. Введение в язык Паскаль: учебник для ССУЗов / В.Г. Абрамов, Н.П. Трифонов, Г.Н. Трифонова – М.:Наука, 2015. – 384 с.
2. Златопольский Д.М. Сборник задач по программированию / Д.М. Златопольский– СПб.: БХВ - Петербург, 2011. – 304 с.
3. Окулов С.М. Программирование в алгоритмах. – М.:Бином. Лаборатория знаний, 2013. – 384 с.
4. Электронный ресурс:  <http://vplaksina.narod.ru/uchebnik/algoritm.htm>
5. Электронный ресурс:  <http://www.maksakov-sa.ru/Algol/RazAlgol/index.html>