Язык программирования Pascal
Массивы

- До сих пор мы работали с простыми типами данных – логический (boolean), целый (integer , word , byte , longint), вещественный (real), символьный (char). Любой алгоритм можно запрограммировать с помощью этих четырех базовых типов. Но для обработки информации о многообразном реальном мире требуются данные, имеющие более сложное строение. Такие сложные конструкции, основанные на простейших скалярных типах, называются структурами.

- **Структура – некоторый составной тип данных, составленный из базовых скалярных.**

Если структура не изменяет своего строения на протяжении всего выполнения программы, в которой она описана, то такую структуру называют статической.

- **Массив** – однородная совокупность элементов

Самой распространенной структурой, реализованной практически во всех языках программирования, является массив.

- Массивы состоят из ограниченного числа компонент, причем все компоненты массива имеют один и тот же тип, называемый базовым. Структура массива всегда однородна. Массив может состоять из элементов типа integer, real или char, либо других однотипных элементов. Из этого, правда, не следует делать вывод, что компоненты массива могут иметь только скалярный тип.
- Другая особенность массива состоит в том, что к любой его компоненте можно обращаться произвольным образом. Что это значит? Программа может сразу получить нужный ей элемент по его порядковому номеру (индексу).

Индекс массива

- Номер элемента массива называется **индексом**. Индекс – это значение порядкового типа, определенного, как **тип индекса** данного массива. Очень часто это целочисленный тип (integer, word или byte), но может быть и логический и символьный.

Описание массива в Паскале

- В языке Паскаль тип массива задается с использованием специального слова **array** (англ. – массив), и его объявление в программе выглядит следующим образом:

Type < имя _ типа >= **array** [I] **of** T;

где I – тип индекса массива, T – тип его элементов.

- Можно описывать сразу переменные типа массив, т.е. в разделе описания переменных:

Var a, b: **array** [I] **of** T;

- Обычно тип индекса характеризуется некоторым диапазоном значений любого порядкового типа: I 1 .. I n . Например, индексы могут изменяться в диапазоне 1..20 или ' a '..' n '.

- При этом длину массива Паскаля характеризует выражение:

ord (I n)- **ord** (I 1)+1.

- Вот, например, объявление двух типов: vector в виде массива Паскаля из 10 целых чисел и stroka в виде массива из 256 символов:

```
Type
    Vector=array [1..10] of integer;
    Stroka=array [0..255] of char;
```

- С помощью индекса массива можно обращаться к отдельным элементам любого массива, как к обычной переменной: можно получать значение этого элемента, отдельно присваивать ему значение, использовать его в выражениях.

- Опишем переменные типа vector и stroka:

```
Var a: vector;
    c: stroka;
```

- далее в программе мы можем обращаться к отдельным элементам массива *a* или *c*.

Например, a [5]:=23; c [1]:=' w '; a [7]:= a [5]*2; writeln (c [1], c [3]).

Вычисление индекса массива Паскаля

- Индекс массива в Паскале не обязательно задавать в явном виде. В качестве индекса массива можно использовать переменную или выражение, соответствующее индексному типу. Иначе говоря, индексы можно вычислять.
- Этот механизм – весьма мощное средство программирования. Но он порождает распространенную ошибку: результат вычислений может оказаться за пределами интервала допустимых значений индекса, то есть будет произведена попытка обратиться к элементу, которого не существует. Эта типичная ошибка называется «выход за пределы массива».

Пример программы с ошибкой массива Паскаля

```
Program primer _ error;
Type
vector=array [1..80] of word;
var
    n: integer;
    a: vector;
begin
    n:=45;
    a[n*2]:=25;
end.
```

- Хотя данная программа полностью соответствует синтаксису языка, и транслятор «пропустит» ее, на стадии выполнения произойдет ошибка выхода за пределы массива Паскаля. При n =45 выражение n *2=90, компьютер сделает попытку обратиться к элементу массива a [90], но такого элемента нет, поскольку описан массив размерностью 80.
- Будем считать, что хорошая программа должна выдавать предупреждающее сообщение в случае попытки обращения к несуществующим элементам массива. Не лишним будет проверять возможный выход как за правую, так и за левую границы массива, ведь не исключено, что в результате вычисления значения выражения получится число, находящееся левее границы массива Паскаля.
- Из всего этого следует сделать вывод: программисту надо быть очень аккуратным при работе с индексами массива.

- Как известно, определение типа данных означает ограничение области допустимых значений, внутреннее представление в ЭВМ, а также набор допустимых операций над данными этого типа. Мы определили тип данных как массив Паскаля. Какие же операции определены над этим типом данных? Единственное действие, которое можно выполнять над массивами целиком, причем только при условии, что массивы однотипны, – это присваивание. Если в программе описаны две переменные одного типа, например,

```
Var
   a, b: array [1..10] of real;
```

- то можно переменной *a* присвоить значение переменной *b* (a :=b). При этом каждому элементу массива *a* будет присвоено соответствующее значение из массива *b*. **Все остальные действия над массивами Паскаля производятся поэлементно *(это важно!)***.

Ввод массива Паскаля

- Для того чтобы ввести значения элементов массива, необходимо последовательно изменять значение индекса, начиная с первого до последнего, и вводить соответствующий элемент. Для реализации этих действий удобно использовать цикл с заданным числом повторений, т.е. простой арифметический цикл, где параметром цикла будет выступать переменная – индекс массива Паскаля. Значения элементов могут быть введены с клавиатуры или определены с помощью оператора присваивания.

Пример фрагмента программы ввода массива Паскаля

```
Var
    A : array [1..10] of integer;
    I : byte; {переменная I вводится как индекс массива}
Begin
    For i:=1 to 10 do
        Readln (a[i]); {ввод i- го элемента производится с клавиатуры}
```

Пример фрагмента программы заполнения массива Паскаля случайными числами

- Рассмотрим теперь случай, когда массив Паскаля заполняется автоматически случайными числами, для этого будем использовать функцию **random (N).**

```
Var
    A: array [1..10] of integer;
    I : byte; {переменная I вводится как индекс массива}
Begin
    For i :=1 to 10 do
        A [ i ]:= random (10); {i -му элементу массива присваивается «случайное» целое число в диапазоне от 0 до 10}
```

Вывод массива Паскаля

- Вывод массива в Паскале осуществляется также поэлементно, в цикле, где параметром выступает индекс массива, принимая последовательно все значения от первого до последнего.

Пример фрагмента программы вывода массива Паскаля

```
Var
    A: array [1..10] of integer;
    I : byte; {переменная I вводится как индекс массива}
Begin
    For i :=1 to 10 do
        Write ( a [ i ],' '); {вывод массива осуществляется в строку, после каждого элемента печатается пробел}
```

- Вывод можно осуществить и в столбик с указанием соответствующего индекса. Но в таком случае нужно учитывать, что при большой размерности массива все элементы могут не поместиться на экране и будет происходить скроллинг, т.е. при заполнении всех строк экрана будет печататься очередной элемент, а верхний смещаться за пределы экрана.

Пример фрагмента программы вывода массива Паскаля в столбик

```
Var
    A: array [1..10] of integer;
    I : byte; {переменная I вводится как индекс массива}
Begin
    For i:=1 to 10 do
        Writeln ('a[', i,']=', a[i]); {вывод элементов массива
в столбик}
```

- На экране мы увидим, к примеру, следующие значения:

```
a [1]=2
a [2]=4
a [3]=1 и т.д.
```

Пример решения задачи с использованием массивов Паскаля

- **Задача:** даны два n -мерных вектора. Найти сумму этих векторов.
- **Решение задачи:**

Входными данными в этой задаче будут являться два одномерных массива. Размер этих массивов может быть произвольным, но определенным. Т.е. мы можем описать заведомо большой массив, а в программе определить, сколько элементов реально будет использоваться. Элементы этих массивов могут быть целочисленными. Тогда описание будет выглядеть следующим образом:

```
var a, b: array [1..100] of integer;
```

Выходными данными будут элементы результирующего массива, назовем его c. Тип результирующего массива также должен быть целочисленным.

Кроме трех массивов нам потребуется переменная – параметр цикла и индекс массива, назовем ее i, а также переменная n для определения количества элементов в каждом массиве.

Ход решения задачи:

определим количество элементов (размерность) массивов, введем значение n;

введем массив a;

введем массив b;

в цикле, перебирая значения индекса i от 1 до n, вычислим последовательно значения элементов массива c по формуле:

```
c [ i ]= a [ i ]+ b [ i ];
```

выведем на экран полученный массив.

Пример программы суммирования векторов

```
Program summa;
Var
   a, b, c: array [1..100] of integer;
   I, n: byte;
Begin
   Write ('введите размерность массивов:');
   Readln(n);
   For i:=1 to n do
      Readln (a[i]); {ввод массива a}
   For i:=1 to n do
      Readln (b[i]); {ввод массива b}
   For i:=1 to n do
      C[i]:=a[i]+b[i]; {вычисление суммы массивов}
   For i:=1 to n do
      write (c[i],' '); {вывод массива c}
end.
```

Домашнее задание

- Задача 1.

Написать программу, которая вводит с клавиатуры одномерный массив из **10** целых чисел и выводит количество ненулевых элементов.

- Задача 2

Написать программу, которая определяет, сколько раз в массиве случайных чисел **A[1 ..100]** встречается число X, введенное с клавиатуры.