Информационные технологии

Лекция №7

Работа с массивами

Массив – упорядоченный набор однотипных переменных , объединенных одним именем.

В качестве типа элементов массива можно использовать все типы: все числовые, символьный, строковый и логический типы.

```
Var
  iArray : array [1..10] of integer;
  rArray : array [1..100] of real;
  cArray : array [0..9] of char;
  sArray : array [1..20] of string[100];
```

Каждый элемент массива имеет свой номер (индекс).

```
  iArray[2] := 10;
  rArray[3] := 3.14;
  sArray[14] := 'Russian Federation';
```

Для индексов массивов подходит любой порядковый тип, то есть такой, который в памяти машины представляется целым числом.

Ограничение состоит в том, что размер массива не должен превышать 64 Кб.

Каждый элемент является переменной, т.е. обладает своим именем и значением.

Массив относится к так называемым структурированным данным, то есть таких, что имеют фиксированную внутреннюю структуру (организацию).

При обращении к отдельному элементу массива необходимо указать его индекс (местонахождение в массиве):

```
Var
  A : array[1..10] of integer;
begin
  i:=7;
  A[i] := 123;
end.
```

Здесь i - индекс элемента массива

Объявление массива

Массивы, как и другие переменные, должны быть объявлены в разделе var

```
var
  Mas: array [1..15] Of real;
  Work: array [(Mon, Tue, Wed)] Of integer;
  B: array ['A'..'Z'] Of boolean;
  C: array [1..3, 1..5] Of real;
```

Ввод массива

Чтобы заполнить массив данными существует несколько способов:

- непосредственное присваивание значений элементам;
- генерация и присваивание значений с помощью функции random;
- ввод значений элементов с клавиатуры;

1) Ввод элементов одномерного массива с клавиатуры:

```
var
  A : array[1..20] of real;
begin
  writeln('Введите элементы массива:');
  for i:=1 to n do readln(A[i]);
  ...

end.
```

2) Заполнение массива случайными числами.

В этом случае необходимо перезапустить генератор случайных чисел. Затем в цикле (например, в цикле с параметром, где в качестве параметра выступает индекс массива) сгенерировать значения для всех элементов.

```
Var
  A: array[1..100] of integer;
  n: integer;
begin
  Randomize;
  N := 100;
  for i:=1 to n do
  a[i]:=random(100);
  ...

end.
```

Двумерные массивы

Двумерный массив можно задать 2-мя способами:

1)

```
Var
 Mas : array [1..3] of array [1..5] of integer;
```

2)

```
Var
  Mas : array [1..3, 1..5] of integer;
```

1	12	25
8	7	2
65	11	9
89	7	15

Ссылка на элемент матрицы Mas, лежащий на пересечении i-той строки ij-ого столбца выглядит следующим образом

```
writeln(Mas[i][j]);
```

Символьный тип данных

Тип данных, переменные которого хранят ровно один символ (букву, цифру, знак препинания и т.п.) называется символьным, а в Паскале — char.

Символы в компьютере сохраняются не в виде букв, а в виде чисел. Так, каждой букве, числу, вообще, любому иероглифу, способному выводится на экран, соответствует число в кодовой таблице.

Кодовая таблица - это специальная система перевода чисел в начертания на мониторе.

Объявить переменную такого типа можно так:

```
var
  ch: char;
```

Для того чтобы положить в эту переменную символ, нужно использовать оператор присваивания, а символ записывать в апострофах, например:

```
ch:='R';
```

Для символьных переменных возможно также использование процедуры readln, например:

```
  write('Exit? (Yes/No)');
  readln(ch);
  if ch='Y' then ...
  else ...;
```

Символьные переменные в памяти компьютера хранятся в виде числовых кодов, иначе говоря, у каждого символа есть порядковый номер.

К примеру, код пробела равен 32, код 'A' - 65, 'B' - 66, 'C' - 67, код символа '1' - 48, '2' - 49, '.' - 46 и т. п.

Некоторые символы (с кодами, меньшими 32) являются управляющими, при выводе таких символов на экран происходит какое либо действие, например, символ с кодом 10 переносит курсор на новую строку, с кодом 7 - вызывает звуковой сигнал, с кодом 8 - сдвигает курсор на одну позицию влево.

Под хранение символа выделяется 1 байт (байт состоит из 8 бит, а бит может принимать значения 0 или 1), поэтому всего можно закодировать 2^8=256 различных символов.

Кодировка символов, которая используется в Паскале, называется ASCII

(American Standard Code for Information Interchange - американский стандартный код для обмена информацией).

Кодовая таблица ASCII

Некоторые коды

Пробел: 32
Цифры от 0 до 9: 48-57
Символ A: 65
Символ tab: 9

Операции над символами
Над символами возможны операции перевода их в числовой эквивалент и обратно.

Как уже говорилось, в Паскале символы связаны с числами в соответствии с кодовой таблицей ASCII.

Обратите на это внимание, поскольку в Windows символы представлены в таблице ANSI, поэтому вы можете обнаружить несоответствие вашей программы, открытой в Windows, например в Блокноте, и в среде Паскаль.

Для получения ASCII-кода любого символа используется функция ord(); . В качестве параметра записывается переменная типа char или же непосредственно в кавычках нужный символ. Например:

```
writeln( Ord('л') );
```

или

```
c3:= ‘л’;
writeln( Ord(c3) );
```

Если в c3 была записана буква л, то на экране появится цифра 171 - ASCII-код строчной русской буквы л.

Обратное действие - получение символа по его коду делает функция

```
chr();
```

В скобках записывается число - от 0 до 255, то есть код необходимого символа или числовая Переменная.

Подобная строчка

```
 writeln( Chr(102) );
```

напишет нам символ f

Если вы используете в функции chr() не числовую переменную, а готовое число, можете вместо самой функции писать символ '#', а после него - число:

```
chr(102)
```

аналогично

```
#102
```

```
program ASCII;
Var
  ch: char;
begin
  for ch:=#32 to #255 do
    write(ord(ch),'->',ch,' ');
  readln;
end.
```

В этой программе в качестве счётчика цикла была использована символьная переменная, это разрешается, поскольку цикл for может использовать в качестве счётчика переменные любого типа, значения которого хранятся в виде целых чисел.

Сравнение символов

Также как и числа, символы можно сравнивать на =, <>, <, >, <=, >=.
В этом случае Паскаль сравнивает не сами символы, а их коды.

Таблица ASCII составлена таким образом, что коды букв (латинских и большинства русских) возрастают при движении в алфавитном порядке, а коды цифр расположены по порядку.

```
Var
  Ch1: char;
  Ch2: char;
Begin
  ch1:= 'A';
  readln(ch2);
  If ch1 > ch2 then
    writeln('ch1 greater than ch2');
end.
```

Строки

Для хранения строк (то есть последовательностей из символов) в Паскале имеется тип `string`.

Значениями строковых переменных могут быть последовательности различной длины (от нуля и более, длине 0 соответствует пустая строка).

Объявить строковую переменную можно двумя способами: либо

```
Var
  s: string; //(максимальная длина строки - 255 символов),
```

либо

```
var
  s: string[n]; //(максимальная длина - n символов, n - константа или конкретное число).
  S2: string[100]; //длина строки 100 символов
  S3: string[10]; //длина 10 символов
```

Для того чтобы положить значение в строковую переменную используются те же приёмы, что и при работе с символами.

В случае присваивания конкретной строки, это строка должна записываться в апострофах

```
Var
  s: string;
begin
  s:= 'Hello, world!';
end.
```

Простейший пример со строками: программа спрашивает имя у пользователя, а затем приветствует его:

```
program Hello;
Var
  s: string;
begin
  write('Как Вас зовут: ');
  readln(s);
  write('Привет, ',s,'!');
  readln;
end.
```

Хранение строк

В памяти компьютера строка хранится в виде последовательности из символьных переменных, у них нет индивидуальных имён, но есть номера, начинающиеся с 1.

Перед первым символом строки имеется ещё и нулевой, в котором хранится символ с кодом, равным длине строки.

```
Var
  S: string;          //занимает в памяти 256 байт=1+255
  S1: string[100];    //занимает в памяти 101 байт
```

Сравнение строк
Строки сравниваются последовательно, по символам.
Сравниваются первые символы строк, если они равны - то вторые, и т. Д.
Если на каком-то этапе появилось различие в символах, то меньшей будет та строка, в которой меньший символ.
Если строки не различались, а затем одна из них закончилась, то она и считается меньшей.

Примеры:

```
'Ананас' < 'кокос'
'свинья' > 'свинина'
' ' < 'A'
'Hell' < 'hello'
```

Склеивание (конкатенация) строк
К строкам можно применять операцию "+", при этом результатом будет строка, состоящая из последовательно записанных "слагаемых".

Пример:

```
s:= 'abc'+'def'+'ghi'; //переменная s будет содержать 'abcdefghi'.
```

Процедуры и функции для работы со строками

`length(s: string): integer`
(после двоеточия записан тип значения, возвращаемого функцией, в нашем случае - целое число).
Эта функция возвращает длину строки `s`.

`copy(s: string; start: integer; len: integer): string`
Возвращает вырезку из строковой переменной `s`, начиная с символа с номером `start`, длина которой `len`

`pos(s1: string; s: string): byte`
Ищет подстроку `s1` в строке `s`. Если находит, то возвращает номер символа, с которого начинается первое вхождение `s1` в `s`; если `s1` не входит в `s`, то функция возвращает 0

`insert(s1: string; s: string; start: integer)`
Вставляет строку `s1` в строковую переменную `s` начиная с символа с номером `start`.

`delete(s: string; start: integer; len: integer)`
Удаляет из строковой переменной `s` фрагмент, начинающийся с символа с номером `start` и длиной `len`.

```
str(x, st);
```
преобразует число, записаное в `x` (целого типа) в строковой тип и записывает его в `st`; Например, если в `x` будет записано число 132, то в `st` мы получим строчку `'132'`.

```
val(st,x,er);
```
обратная процедура к `str`, то-есть, переводит строковое значение `st` в числовое `x`. В переменную `er` (`Integer`) записывается код ошибки.
Код ошибки необходим, поскольку в строке могут содержатся не только цифры, но и буквы, которые не преобразуются в числа. Поэтому, если в `st` были не только числа, то в `x` ничего не запишется, а в `er` будет не ноль, а какое-то число.

Обычно, после этой процедуры проверяют, ноль ли записан в `er` чтобы убедиться в правильности операции.