Строки, записи, множества

---

1. Строки

Тип "**array**[M..N] **of** Char" , т.е. массив литер естественно интерпретировать как тип строк фиксированного размера. Вирт предусмотрел специальный упакованный тип "**packed array**[M..N] **of** Char", строки которого можно включать в список вывода. Они также совместимы по присваиванию со строками-константами строго равной длины. Пример.

```
program StrTest;
type TStr: packed array[1..4] of Char;
var s1,s2: TStr;
begin
s1 := 'ABCD';
s2 := s1;
WriteLn('s2 = ', s2, ' s1[3] = ', s1[3]);
end.
```

В АЯ TPascal определён очень удобный стандартный строковый тип **sring**. Описание: **string**[Lm], где Lm - целочисленная константа из диапазона 1..255. Это описание определяет массив из Lm+1-ой литеры типа Char, для обращения к которым можно использовать индексы из диапазона 0..Lm. Элемент с индексом 0 хранит текущую длину строки. Для получения её значения достаточно записать выражение Ord(str[0]), где str - строка типа **string**. Но на практике для определения длины строки этого типа чаще используют стандартную функцию Length. Элементы строки с индексами из диапазона 1..Lm хранят соответствующие литеры строки.

Строка, текущая длина которой равна нулю, называется пустой. `''` - строковая константа нулевой длины.

Тип **string** совместим с типом Char и с упакованным массивом литер. Строки этого типа можно включать не только в список вывода, но и в список ввода.

Для типа **string** определена операция сцепления строк - +.

Пример: `s2 := '<' + s1 + '>' .`

Кроме того, для работы со строками типа **string** определены стандартные процедуры и функции:

Вид вызова (для функций указан тип возвращаемого значения)	Алгоритм (префиксы имён параметров указывают на их тип)

Length(sS) -> i	Возвращает длину sS
Copy(sS, iInd, iCnt) -> s	Возвращает подстроку строки sS, которая начинается с позиции iInd и содержит iCnt символов (или менее, если sS заканчивается раньше)
Pos(sP,sS) -> i	Позиция в sS подстроки sP (0, если sS не содержит sP)
Insert(sP, svS, iInd)	Вставляет sP в svS с позиции iInd
Delete(sS, iInd, iCnt)	Удаляет из sS iCnt символов, начиная с позиции iInd (или меньше, чем iCnt, если sS заканчивается раньше)

Пример. Программа, которая заменяет все вхождения в строку символа '*' заданной подстрокой.

```
program Macro;
var sText: string[80];
    sArg:  string[8];
    p: Integer;
begin
WriteLn('Введите sText и sArg:');
ReadLn(sText);
ReadLn(sArg);
p := Pos('*', sext);
while p <> 0 do
    begin
    Delete(sText, p, 1);
    Insert(sArg, sText, p);
    p := Pos('*', sText);
    end;
WriteLn('Выход:');
WriteLn(sText);
end.
```

2. Записи

Описание типа:

record
<Имя поля> *{*, <Имя поля>*}*: <Имя или описание типа>*{* ;
<Имя поля> *{*, <Имя поля>*}* : <Имя или описание типа> *}*
end

Пример.

```
type TDate = record
             Day, Mon, Year: Integer
             end;
     TName = record
             FName, LNname: string[20]
             end;
     TStd  = record
             Name: TName;
             Bd:   TDate;
             Notes: array[1..5] of Integer
             end;
var  Std: TStd;
```

Обращение:

```
Std
Std.Name  Std.Name.FName Std.Name.LName
Std.Bd  Std.Bd.Day Std.Bd.Mon Std.Bd.Year
Std.Notes  Std.Notes[2]
```

3. Оператор with

Синтаксис:

with <Имя старшей структуры> **do** <Оператор>

Пример.

```
with Std.Bd do
    begin Day := 8; Mon := 4; Year := 1975 end
```

Применение оператора **with** может привести к некоторому сокращению записи программы.

4. Множества

Описание типа:

set of <Имя или описание ординального типа>

В TPascal на базовый тип наложено ограничение (Ord(Min) >= 0) and (Ord(Max) < 256), где Min и Max минимальное и максимальное значения типа.

<Конструктор множества> ::= [ ] | [<Элемент> *{*, <Элемент>*}*]

<Элемент> ::= <Константа> *[* ..<Константа> *]*

Операции:

a **in** M	Элемент a принадлежит множесту M
M1 <= M2	M1 содержится в M2
M1 >= M2	M1 включает M2
M1 = M2	M1 и M2 эквивалентны
M1 <> M2	M1 и M2 неэквивалентны
M1 + M2	Объединение множеств M1 и M2
M1 * M2	Пересечение множеств M1 и M2
M1 – M2	Разность множеств M1 и M2 (множество элементов из M1, которых нет в M2)

Пример. Программа определяет является ли строка именем.

```
program TstIsNam;
const LETTERS = ['A'..'Z','a'..'z'];
      FIGURES = ['0'..'9'];
var Str:    string[10];
    i:      Integer;
    IsName: Boolean;
begin
```

```
Write('Str: '); ReadLn(Str);
IsName := Str[1] in LETTERS;
if IsName then
    for i := 2 to Length(Str) do
        if not(Str[i] in LETTERS + FIGURES) then
            begin IsName := False; Break end;
WriteLn('IsName = ', Ord(IsName));
end.
```

Пример. Программа выводит элементы множества.

```
program WrSet;
const MIN = 1; MAX = 12;
var i: Integer;
    S: set of MIN..MAX;
begin
S := [2, 5..8];
for i := MIN to MAX do
    if i in S then Write(i:3);
end.
```

5. Оператор case

Этот оператор относится к операторам выбора, но в отличие от оператора **if** количество вариантов выбора может быть больше двух. Синтаксис:

case <Выражение> **of**
<Выбор>: <Оператор> *{*;
<Выбор>: <Оператор> *}*
[**else** <Оператор> *]*
end

<Выбор> ::= <Элемент> *{*, <Элемент> *}*

<Элемент> ::= <Константа> *[* .. <Константа> *]*

Выражение и константы дожны быть одного ординального типа. Часть **else** можно опустить. В этом случае оператор не выполняет никаких действий.

Пример. Фрагмент программы, который выводит количество дней в каждом месяце.

```
for i := 1 to 12 do
    begin
    case i of
    1,3,5,7,8,10,12: begin m := 31; d := 0 end;
    2:               begin m := 28; d := 1 end
    else
        begin m := 30; d := 0 end
    end; {case}
    WriteLn('m[', i, '] = ', m);
    end;
```

d - количество дополнительных дней в месяце в високосном году.
