Строки (с точки зрения Стандартного Паскаля)

Строка – это особая разновидность массива со следующим описанием:

```
packed array [1..n] of char;
```

В этом описании все важно: 1) **packed** – обязательно нужно писать; 2) *тип индексов* – только целочисленный, нумерация от единицы, верхняя граница либо в виде явного целого числа, либо в виде целочисленной константы, причем нижняя граница должна быть строго меньше верхней; 3) *тип элементов* – только **char**, ни в коем случае – не диапазон!

Дополнительные возможности при работе со строками (не распространяются на обычные массивы):

1) Две строки считаются **однотипными**, если их длины совпадают.

```
const n=4;
var    S1: packed array [1..4] of char;    S1: packed array [1..n] of char;
...
S1 := S2; {можно, т.к. длины строк равны}
```

2) Две строки **равной длины** можно сравнивать между собой за 1 операцию сравнения: =, <>, <. <=, >, >= (с учетом *лексикографической упорядоченности*, т.е. все строки равной длины упорядочены в соответствии с внутренней кодировкой входящих в них символов, сравнение строк при этом идет слева направо до первого несовпадения, по результату которого и принимается решение, оставшиеся символы уже не рассматриваются).
например, **'abcde' < 'abfdb'**, т.к. **'c'<'f'**

3) Строку можно печатать за 1 обращение к writeln: **writeln(S1)**

4) Разрешены **строки-константы**:

```
const S='abcd';
```

{этой строке транслятор автоматически припишет тип **packed array[1..4] of char** }
Внимание! Конструкция **S[2]** – незаконна, если **S** – имя строки-константы (в Стандарте, а также во Фри-Паскале), т.к. по синтаксису Паскаля слева от *селектора элемента* (т.е. слева от **[2]**) может стоять только *имя переменной*, но не *имя константы*.

Замечание

В Стандартном Паскале неудачно решен вопрос со строковым типом, так как строки рассматриваются как **массивы** с дополнительными свойствами. Отсюда и фиксированная длина строки. В реальных же задачах по обработке строк длина строки, как правило, динамически меняется (за счёт удаления и добавления символов). Решать такие реальные задачи средствами Стандартного Паскаля крайне неудобно, так как приходится искусственно подгонять все строки под одну длину (в расчёте на максимально возможную длину строки, что не всегда известно заранее). Это неминуемо приводит к накладным расходам как по памяти, так и по времени выполнения.
В языке Фри-Паскаль представлено несколько разновидностей строковых типов (String, PChar, AnsiString, WideString), которые избавлены от указанных выше недостатков и позволяют гибко и эффективно решать задачи по обработке строк.