**Тип STRING**

Тип STRING - это строковый тип в Паскале. Строкой называется последовательность символов. Строковыми константами вы уже неоднократно пользовались - это последовательность любых символов, заключенная в апострофы; допускаются и пустые строки, они записываются так: ''. Строковые переменные и типизированные константы описываются в виде

STRING

или

STRING [ максимальная длина ]

Если максимальная длина не задана, то по умолчанию она берется равной 255. Максимальная длина при описании строковых данных задается целочисленным константным выражением и никогда не может превышать 255. Это ограничение обусловлено самой структурой типа STRING : фактически строка - это массив ARRAY [ Byte ] OF Char, причем в 0-м символе закодирована текущая длина строки. Строковые переменные могут иметь любую длину от 0 до максимальной. В программе строки можно использовать и как единый структурированный объект (чуть позже мы познакомимся с разнообразными возможностями обработки строк), и как массив символов, т.е. обращаться к элементам строк следует так же, как к элементам массивов. Для строк определены следующие операции :

- строке можно присвоить строку;

- строки можно вводить процедурой READLN;

- строки можно выводить процедурой WRITE[LN];

- для строк определена операция сцепления +, при этом вторая строка дописывается справа к первой и длина результата становится равной сумме длин операндов (если она не превосходит 255).

Запишем программу, выполняющую простейшие операции со строками:

TYPE ShortString = STRING[80];

VAR s1,s2 : ShortString; s3 : STRING;

BEGIN WRITE('Введите 1-ю строку '); READLN(s1);

WRITE('Введите 2-ю строку '); READLN(s2);

WRITELN('Вы ввели ',s1,' и ',s2); WRITELN('s1+s2=',s1+s2);

s3:=s1+s1+s1; WRITELN('s1,повторенная 3 раза ',s3);

END.

Обратите внимание, что при вводе строк всегда используется READLN, но не READ. Процедура READ в отличие от READLN считывает лишь символы до символа конца строки (клавиша Enter), который остается в буфере клавиатуры. Таким образом, пользуясь процедурой READ можно ввести только одну строку; все строки, вводимые вслед за первой, станут пустыми. Например, программа

VAR s1,s2 : STRING;

BEGIN WRITE('Введите 1-ю строку '); READ(s1);

WRITE('Введите 2-ю строку '); READ(s2);

WRITELN('Вы ввели "',s1,'" и "',s2,'"');

END.

при входном потоке abcdef Enter 123456 Enter выведет : Вы ввели "abcdef" и "". Запишем теперь программу, которая вводит некоторую строку, заменяет в ней все цифры на пробелы и дописывает в конец строки символы "???":

VAR s : STRING; L,i : Byte;

BEGIN WRITE('Введите строку '); READLN(s);

L:=ORD(s[0]);

FOR i:=1 TO L DO IF s[i] IN ['0'..'9'] THEN s[i]:=' ';

FOR i:=L+1 TO L+3 DO s[i]:='?';

WRITELN('Вот что получилось : ',s);

END.

Наша программа заменила цифры, но никаких "?" не добавила. Дело в том, что, обращаясь к элементам строки, невозможно изменить текущую длину строки. Второй цикл нашей программы сработал правильно, записав символы "?" в соответствующие элементы строки, но длина строки осталась прежней, и процедура WRITELN вывела только символы с 1-го по L-й. Чтобы решить задачу корректно, мы могли бы добавить в программу один оператор INC(s[0],3); но, конечно, лучше всего просто записать: s:=s+'???'; .

Для обработки строк в Паскале существует несколько стандартных функций и процедур :

1. FUNCTION Length(S: String): Integer; - возвращает длину строки.

2. FUNCTION Concat(S1[,S2,...,Sn]: String): String; - возвращает строку, полученную сцеплением аргументов, может использоваться вместо операции "+".

3. FUNCTION Pos(Substr: String; S: String): Byte; - возвращает номер первого слева символа строки S, начиная с которого строка Substr входит в S, если Substr не входит в S, то значение функции равно 0.

4. FUNCTION Copy(S: String; Index: Integer; Count: Integer): String; - возвращает подстроку строки S, которая начинается с символа с номером Index и имеет длину Count.

5. PROCEDURE Delete(VAR S: String; Index: Integer; Count:Integer); - удаляет из строки S подстроку, начинающуюся с символа с номером Index и имеющую длину Count.

6. PROCEDURE Insert(Substr: String; VAR S: String; Index: Integer); - вставляет в строку S подстроку Substr начиная с символа с номером Index.

Из вышеизложенного понятно, что процедуры и функции могут иметь параметры типа STRING (что неудивительно), но также допустимы функции типа STRING, хотя это и не скалярный тип. Еще две стандартные процедуры предназначены для перевода строки в число и числа в строку:

7. PROCEDURE Val(S: STRING;VAR V; VAR Code: Integer); - преобразует строку S в число V (если это возможно); V - любая переменная арифметического типа, переменная Code возвращает 0, если преобразование прошло успешно, или номер первого неправильного символа строки.

8. PROCEDURE Str(X [: Width [: Decimals ]];VAR S:STRING); - преобразует произвольное арифметическое выражение X в строку S, параметры Width и Decimals позволяют форматировать строку и имеют такой же смысл, как и в процедуре WRITE[LN] .

Теперь, зная процедуру Val, вы можете организовать надежный ввод числовых данных в любой своей программе. Предположим, что программа должна вводить вещественное значение F. Мы можем записать это так :

VAR F : Real; ... BEGIN WRITE('Введите F '); READ(F);

Если пользователь правильно введет число, то все будет в порядке, но если он ошибочно нажмет не ту клавишу (например, запятую вместо точки и т.п.), то произойдет аварийное прерывание, программа завершится, и на экране появится сообщение "Run-time error ...". Программы, таким образом реагирующие на неверный ввод, - плохие! Хорошая программа обязана обрабатывать нажатие практически любых клавиш в любых комбинациях. Мы вполне способны написать такую программу :

VAR F : Real; S : STRING; Code : Integer; ...

BEGIN REPEAT

WRITE('Введите F '); READLN(S);

Val(S,F,Code); IF Code=0 THEN Break;

WRITELN('Ошибка ввода!');

UNTIL FALSE;

Решим часто встречающуюся задачу о распаковке текста: дана строка, содержащая текст на русском языке (или на любом другом языке, в том числе и искусственном - вы увидите, что это не принципиально); нужно выделить слова, содержащиеся в этом тексте. Хотя эта задача и элементарна, ее решение не столь тривиально и требует предварительной разработки алгоритма. Сначала уясним, что такое текст. Текстом будем называть последовательность слов, разделенных любым количеством "пробелов". Слова - это последовательности букв языка (в нашем случае - русских букв), "пробелы" - любые символы, не являющиеся буквами. Итак, наш текст в общем случае имеет вид : \*X\*X...\*X\* , где X - слово, \* - "пробел". Можно предложить следующий алгоритм распаковки:

1) удалим завершающие пробелы, после чего текст примет регулярный вид \*X\*X...\*X;

2) удалим лидирующие пробелы;

3) выделим первое слово и удалим его из текста.

После выполнения пунктов 2 и 3 мы получили одно слово и текст стал короче на одно слово, сохранив при этом свою структуру. Очевидно, что пункты 2 и 3 следует выполнять до тех пор, пока текст не пуст. Запишем программу, реализующую этот алгоритм.

VAR s : STRING; i : Byte;

CONST Letters : SET OF Char = ['а'..'п','р'..'я','А'..'Я']; {это алфавит}

BEGIN WRITE('Введите текст '); READLN(s);

{ удалим завершающие пробелы, здесь есть 1 ОШИБКА! }

WHILE NOT(s[Length(s)] IN Letters) DO Delete(s,Length(s),1);

WRITELN('Слова текста :');

{ организуем цикл ПО СЛОВАМ }

WHILE s<>'' DO BEGIN

{ удалим лидирующие пробелы }

WHILE NOT(s[1] IN Letters) DO Delete(s,1,1);

{ найдем границу первого слова, здесь есть 1 ОШИБКА! }

i:=1; WHILE s[i] IN Letters DO INC(i);

{ i - номер первого пробела }

Dec(i);

{ выведем слово }

WRITELN(Copy(s,1,i));

{ удалим слово из текста }

Delete(s,1,i);

END;

END.

На первый взгляд наша программа работает правильно (мы ввели фразу на русском языке и получили все слова из нее), но тестирование программы обязательно должно включать все предельные, или особенные, случаи. Введем, например, строку, не содержащую никаких слов, и программа зациклится! Это результат ошибки в первом цикле: если в тексте нет букв, все символы из него будут удалены, длина строки станет равной нулю, и в дальнейшем станет проверяться символ с номером 0, который равен #0 и, естественно, не является буквой. Еще одна ошибка подобного рода может произойти при выделении последнего слова: мы увеличиваем индекс i, пока i-й символ - буква, и в конце концов дойдем до конца строки. Но переменная s всегда содержит 255 символов, символы с номерами Length(s)+1, Length(s)+2 и т.д. существуют, и нет никаких гарантий, что они не являются русскими буквами. В этом случае мы можем получить последнее слово с "хвостом". Исправим нашу программу:

VAR s : STRING; i : Byte;

CONST Letters : SET OF Char = ['а'..'п','р'..'я','А'..'Я']; {это алфавит}

BEGIN WRITE('Введите текст '); READLN(s);

{ удалим завершающие пробелы }

WHILE NOT(s[Length(s)] IN Letters)AND(s<>'') DO Delete(s,Length(s),1);

IF s='' THEN BEGIN WRITELN('текст пуст'); Halt; END;

WRITELN('Слова текста :');

{ организуем цикл ПО СЛОВАМ }

WHILE s<>'' DO BEGIN

{ удалим лидирующие пробелы }

WHILE NOT(s[1] IN Letters) DO Delete(s,1,1);

{ найдем границу первого слова }

i:=1; WHILE (s[i] IN Letters)AND(i<=Length(s)) DO INC(i);

{ i - номер первого пробела }

Dec(i);

{ выведем слово }

WRITELN(Copy(s,1,i));

{ удалим слово из текста }

Delete(s,1,i);

END;

END.

Теперь запишем то же самое, используя функции и процедуры :

VAR s : STRING; i : Byte;

CONST Letters : SET OF Char = ['а'..'п','р'..'я','А'..'Я']; {это алфавит}

PROCEDURE DelTail(VAR s:STRING);

BEGIN WHILE NOT(s[Length(s)] IN Letters)AND(s<>'') DO Delete(s,Length(s),1); END;

PROCEDURE DelLead(VAR s:STRING);

BEGIN WHILE NOT(s[1] IN Letters) DO Delete(s,1,1); END;

FUNCTION MakeWord(s:STRING; VAR Bound:Byte):STRING;

BEGIN Bound:=1;

WHILE (s[Bound] IN Letters)AND(Bound<=Length(s)) DO INC(Bound);

Dec(Bound); MakeWord:=Copy(s,1,i); END;

BEGIN WRITE('Введите текст '); READLN(s);

{ удалим завершающие пробелы }

DelTail(s);

IF s='' THEN BEGIN WRITELN('текст пуст'); Halt; END;

WRITELN('Слова текста :');

{ организуем цикл ПО СЛОВАМ }

WHILE s<>'' DO BEGIN

{ удалим лидирующие пробелы } DelLead(s);

{ выведем слово } WRITELN(MakeWord(s,i));

{ удалим слово из текста } Delete(s,1,i);

END;

END.