ОБРАБОТКА СТРОКОВЫХ ВЕЛИЧИН

В Паскале, как и в других языках программирования, предусмотрена обработка текстов или строк. Для этой цели в языке существует два типа данных: SHAR и STRING.

Тип данных CHAR

Типу данных CHAR соответствуют символьные константы и переменные. Символьная константа есть какой-то символ алфавита, взятый в апострофы. Символьные переменные получают значения символьных констант с помощью оператора присваивания:

ALPFA := 'p'; A := 't'; B := '3'; C := ' '; D := ''.

Все символы алфавита образуют множество литер. Каждый символ имеет свой код в ASCII. Это позволяет использовать булевские сравнения: =, <>, <, <=, >, >= .

Данные этого типа описываются с помощью служебного слова CHAR. Например, переменную ALPFA можно описать как VAR ALPFA : CHAR.

Общая форма записи

VAR <переменная>: CHAR;

При работе с данными типа CHAR, если у нас есть последовательность символов, существует два способа ввода этих символов с клавиатуры.

При первом способе организуется цикл, внутри которого помещается оператор READLN. При этом способе элементы последовательности вводятся поочередно, и после набора на клавиатуре символа необходимо нажать клавишу ввода ENTER. Таким образом, здесь число нажатий клавиши ENTER совпадает с числом вводимых элементов последовательности.

Второй способ характеризуется применением для ввода символов оператора READ. С его помощью можно сразу же ввести всю последовательность символов, которая записывается в буфер клавиатуры. Последующий цикл с оператором READ осуществляет уже выборку элементов из этого буфера в соответствующие переменные, указанные в операторе READ.

П р и м е р 1. С клавиатуры последовательно вводятся символы. Признаком конца ввода является точка. Составить программу выбрасывания групп символов, расположенных между скобками ( ). Сами скобки тоже выбрасываются.

```
program SKOBKI;
var c: char; i: integer;
begin
¦ i := 0; read (c);
¦ while c <> '.' do
¦   begin
¦   ¦ if c = '(' then i : = 1
¦   ¦         else if c = ')' then i := 0
¦   ¦            else if i = 0 then write (c);
¦   ¦ read (c);
¦   ¦   end;
end.
```

Пояснение. I = 1 означает, что ранее была прочитана левая скобка, которой пока еще не нашлось парной правой. В этой ситуации прочитанные символы не выводятся на экран. В результате работы этой программы на экране будет представлена строка символов. Здесь вся последовательность символов вводится сразу по первому оператору READ, а затем в цикле из буфера клавиатуры выбираются, анализируются и выводятся на экран символы вне круглых скобок. Например, если вводится последовательность «asg(zx)ytr.», то экран будет выглядеть так:

asg(zx)ytr. – результат работы оператора READ;
asgytr – результат работы оператора WRITE.

В этой программе можно было бы использовать оператор READLN, но тогда после набора каждого символа необходимо нажимать клавишу ввода. Кроме того, на экран будет выводиться не строка символов, а столбец, состоящий из вводимых и отпечатанных элементов.

Упорядоченность символов языка используется при написании циклов с параметром, где параметр цикла может пробегать буквенные значения.

П р и м е р 2. Программа вывода последовательности букв a, ab, abc, ..., abc ... xyz.

```
program SUITE;
```

```
    var c, d: char;
    begin
       for c := 'a' to 'z' do
       begin
          for d := 'a' to c do write (d);
          writeln (' ');
       end;
end.
```

Массивы литер

В рассмотренных программах все символы вводились последовательно в процессе работы цикла или хранились временно в буфере клавиатуры. Это не всегда удобно, поэтому в языках делают строки как последовательность литер. Строку можно задать как массив литер, при этом в качестве длины строки может выступать верхняя граница массива, например:

```
VAR HAMLET : ARRAY [1..17] OF CHAR.
```

Здесь HAMLET – массив литер, компоненты которого имеют тип CHAR; индекс имеет нижнюю границу, равную 1, верхнюю – 17. Для ввода строки в массив HAMLET необходимо организовать цикл из 17 повторений. При каждом повторе этого цикла с клавиатуры вводится очередной символ строки и нажимается клавиша ввода:

```
for n := 1 to 17 do readln (HAMLET [n]).
```

Этот массив можно образовать и с помощью однократного нажатия клавиши ввода после набора всех элементов строки:

```
for n := 1 to 17 do read (HAMLET [n]).
```

Поскольку массивы литер являются обычными массивами, но их компоненты имеют тип CHAR, то они обладают всеми свойствами регулярных массивов. Для извлечения из массива-строки отдельного символа необходимо использовать индекс этого элемента. Например, можно вывести на экран строку HAMLET в обратном порядке с помощью следующего цикла:

```
for n := 17 downto 1 do write (HAMLET [n]).
```

Тип данных STRING

Наряду с тем положительным, что дают нам массивы литер, они обладают существенным недостатком: их длину нельзя менять во время выполнения программы. Так, описанная переменная HAMLET в п. 7.2 есть массив из 17 элементов, и, следовательно, туда можно поместить только текст, содержащий ровно 17 символов.

Это не всегда удобно. Хотелось бы иметь такую переменную, в которую можно было бы поместить текст произвольной (но ограниченной) длины. Такую возможность предоставляет тип STRING. Так, объявив переменную var HAMLET: string [17], можно путем оператора присваивания (а не через цикл) задать ей значение текста произвольной длины (от 0 до 17), например:

```
HAMLET := 'Быть или не быть';
HAMLET := 'Бедный Йорик';
HAMLET := ' '; HAMLET:= ''.
```

Отметим также, что при компиляции программы в случае объявления строки-массива в памяти ЭВМ резервируется место под массив, который должен быть полностью заполнен в процессе работы программы. Для типа STRING также резервируется место в памяти того же объема, но здесь не обязательно заполнять его целиком. Незаполненные места представлены пробелами. Данный тип представлен следующейобщей формой записи:

Общая форма записи

```
TYPE <имя типа> = STRING [N];
VAR <имя переменной>: <имя типа>;
или
VAR <имя переменной>: STRING [N];
```

Здесь N – целая константа, задающая максимальную длину текста. Доступ к элементам строки производится с помощью индексов, так как в этом типе также все элементы имеют свой (числовой) индекс от 1 до N. В результате получается величина типа CHAR, например:

```
HAMLET := 'ПРОГРАММА';
HAMLET [1] = 'П'; HAMLET [9] = 'А'.
```

Тип STRING и стандартный тип CHAR совместимы. Строки и символы могут употребляться в одних и тех же строковых выражениях.

Строковое выражение состоит из строковых (символьных) констант, переменных, указателей строковых функций и операции конкатенации (склеивания) строк, обозначаемой знаком «+». Строки можно сравнивать. В результате сравнения двух строк истина получается только в том случае, если сравниваемые строки совпадают посимвольно и имеют одинаковую длину (принадлежат одному и тому же типу).

Строковые функции и процедуры

Строковые функции и процедуры введены в систему программирования Turbo Pascal для облегчения манипуляции со строками. Имеется восемь строковых функций и процедур.

1. *Функция CONCAT (склеивание)*

Синтаксис: concat (S1, S2, ..., Sn: string): string. Возвращает строку, полученную конкатенацией строк S1,...,Sn. Если длина результата больше 256, то излишние символы отбрасываются. Эта функция фигурирует в правой части «:=» и в строковых выражениях.

П р и м е р:

NUMBER := concat ('12', '34', '50'); NUMBER = '123450'.

2. *Функция LENGTH (длина)*

Синтаксис: length (S: string): integer. Возвращает длину строки S.

П р и м е р :

N := length ('345'); N = 3.

3. *Функция POS (позиция)*

Синтаксис: pos (S,T: string): integer. Функция POS в качестве аргументов использует две строки и определяет, содержится ли первая строка во второй. Возвращает номер символа, начиная с которого S входит в T. Если вхождения нет, то возвращает 0.

П р и м е р :

N := pos ('E', 'HELLO');

N = 2.

N := pos ('A', 'HELLO');
N = 0 .

4. Функция COPY (вырезка фрагмента)

Синтаксис: copy (S: string; N1, N: integer): string. Возвращает подстроку, полученную из N символов строки S, начиная с позиции N1. Значение переменной S при этом не меняется.

П р и м е р :

```
FRAGMENT := copy ('PROGRAMM', 2, 3);
                    FRAGMENT  = 'ROG'.
```

5. Процедура DELETE (стирание фрагмента)

Синтаксис: delete (var S: string; POS, LEN: integer). Убирает из строки S LEN символов, начиная с POS, при этом длина строки уменьшается на LEN позиций.

П р и м е р :

```
FRAGMENT := 'PROGRAMM';
delete (FRAGMENT, 2, 3);
FRAGMENT = 'PRAMM'.
```

6. Процедура INSERT (вставка)

Синтаксис: insert (S: string; var D: string; POS: integer). Вставляет строку S в строку D перед символом с номером POS, при этом длина строки D увеличивается на LENGTH (S) позиций.

П р и м е р :

```
FRAGMENT := 'PRAMM';
insert ('ROG', FRAGMENT, 2);
FRAGMENT = 'PROGRAMM'.
```

7. *Процедура STR (преобразование в строку)*

Синтаксис: str (I: integer; var S: string); str (R: real; var S: string).

Преобразует I или R из числа в строку и записывает эту строку в S, причем R и I могут записываться форматно, как в процедуре WRITE.

П р и м е р :

а) R := 123.654; str (R:5:2, S); S = '123.65';

б) I := 5683; str (I, S); s = '5683'.

8. *Процедура VAL (преобразование в число)*

Синтаксис: val (S: string; var I, J: integer); val (S: string; var I: real; var J: integer).

Преобразует строковую переменную S в число типа I. Переменная J получает значение 0, если перевод прошел без ошибок. Если же сделана попытка конвертировать в число строку, где есть нецифровые символы, то переменная J принимает значение позиции первого нецифрового символа. При этом работа процедуры прерывается.

П р и м е р :

а) S := '4326';
val (S, I, J);
I = 4326, J = 0;

б) S := '43p8';
val (S, I, J);
I – не определено, J = 3.

Рассмотрим теперь пример на применение указанных функций и процедур обработки строк.

П р и м е р . Изменение порядка слов в строке.

```
program REVERSE;
var OLD_LINE, NEW_LINE: string [50];
PROBEL: integer; WORD: string [50];
begin
    NEW_LINE := ''; readln (OLD_LINE);
    OLD_LINE := concat (OLD_LINE,' ');
    while OLD_LINE <> '' do
    begin
        PROBEL := pos (' ', OLD_LINE);
        word := copy (OLD_LINE, 1, PROBEL);
        NEW_LINE := concat (WORD, NEW_LINE);
        Delete (OLD_LINE, 1, PROBEL);
    end;
    writeln (NEW_LINE)
end.
```

Пояснение. С клавиатуры вводится строка OLD_LINE, и к ней справа подклеивается пробел. Это делается для того, чтобы строка имела одну и ту же структуру: слово плюс пробел. Затем в цикле, признаком конца которого является пустая константа, выделяется очередное по порядку слово и подклеивается слева в переменную NEW_ LINE. После выборки очередного слова из OLD_LINE оно оттуда выбрасывается, что приводит к постепенному уменьшению самой строки. Здесь переменная PROBEL служит

для хранения позиции первого пробела в строке, а WORD – для выбранного из OLD_LINE слова.

Например, строка *'Наша Таня громко плачет'* преобразуется в строку *'плачет громко Таня Наша'*.

Для обработки текстовой информации можно использовать те же методы, что применяют для одномерных массивов, так как структура строкового типа схожа с массивом. Однако для упрощения написания программ по работе с текстами были разработаны стандартные строковые процедуры и функции. Поэтому основной задачей этой лабораторной работы является освоение строковых операций, функций *Length, Pos, Copy* и процедур *Delete, Insert.*

Перед выполнением работы необходимо ознакомиться с теоретическим материалом по теме «Обработка литерных величин. Данные типа *Char* и *String*».

Пример 1. Составить программу обработки данной строки, позволяющую выписать все знаки сравнения и все скобки, сохранив их последовательность.

Решение. Все знаки сравнения и скобки перечислим в строковой константе *srav_sk*. В теле программы последовательно рассмотрим все символы введенной строки *s*, проверяя каждый на вхождение в строку *srav_sk*, выводя на экран содержащиеся в строковой константе символы строки *s*.

```
program string_1;
const srav_sk='<>=(){}[]';
var s:String; i:Integer;
begin
  Writeln('Введите строку:'); Readln(s);
  for i:=1  to Length(s) do
    if Pos(s[i],srav_sk)<>0 then  write(s[i]);
  Readln
end.
```

Пример 2. Дана строка, состоящая из слов, разделенных пробелами (одним или несколькими). Составить программу обработки данной строки, выводящую через запятую слова четной длины, которые при обмене местами левой и правой половины дают то же слово (например, папа, мама, канкан).

Решение. В строке, последовательно перебирая элементы, будем выделять начало *ns* и конец *ks* каждого слова, копировать слово в другую

строковую переменную, а затем проверять четность длины и равенство частей строки *slovo*. Слова, для которых выполнено условие задания будем склеивать в переменной *otv*. После добавления к ответу последнего слова в конце *otv* будет лишняя запятая, которую перед выводом на экран необходимо удалить.

```
program string_2;
var s, slovo, otv:String; ns, ks, i, m: Integer;
begin
  Writeln ('Введите строку:'); Readln(s);
  i:=1; ns:=1; otv:=''; s:=s+' ';
  while i<=Length(s) do
  begin
    while (i<=Length(s)) and (s[i]<>' ') do i:=i+1; {ищем очередной пробел}
    ks:=i; slovo:=Copy(s, ns, ks-ns); {выделяем слово}
    m:=Length(slovo) div 2; {половина длины слова}
    if (Length(slovo) mod 2=0) and (Copy(slovo,1,m)=Copy(slovo,m+1,m))
      {если слово имеет четную длину}
       then otv:= otv+slovo+ ','; {добавляем слово к ответу}
    while (i<=Length(s)) and (s[i]=' ') do i:=i+1; {пропускаем пробелы}
    ns:=i; {начало следующего слова}
  end;
  Delete(otv, Length(otv), 1); {удаляем лишнюю запятую в конце}
  Writeln(otv);
  Readln
end.
```

Вопросы для самоконтроля:

1. Как описываются в языке Паскаль строковые величины?

2. В чем сходство и в чем различие между массивами и строками?

3. Существуют ли ограничения, накладываемые на длину строки?

4. Какие строковые процедуры существуют в языке Паскаль?

5. Для вывода значений каких строковых функций нужны переменные типа string, а для каких – integer?

6. Какие есть возможности извлечения из строки одного символа?

Задание

1. Определить, какое из двух слов длиннее и на сколько.
2. Определить, является ли какое-нибудь из двух слов частью другого.
3. Определить, есть ли в записи квадрата данного числа цифра 1.

4. Поменять в слове первую и последнюю буквы.
5. Если в слове нечетное число букв, то удвоить среднюю.
6. По последнему символу определить тип предложения (повествовательное, вопросительное, восклицательное).
7. Определить, является ли данный символ латинской буквой.
8. Удалить из слова среднюю букву (или две средних).
9. Заменить в арифметическом выражении знаки "+" на знаки "-", а знаки "-" на знаки "+".
10. Удалить все буквы "я" в данном слове.
11. Удвоить все четные буквы слова.
12. Удалить все предлоги "на" в данном предложении.
13. Вставить после каждой буквы данного слова букву "о".
14. Удалить лишние пробелы в данном предложении.
15. Удвоить каждую букву данного слова.
16. Заменить каждую точку многоточием (т.е. тремя точками).
17. Удвоить все согласные буквы.
18. Удалить из данного слова все согласные буквы.
19. Проверить, имеются ли в данном слове одинаковые буквы.
20. Оставить в данном слове из каждого набора одинаковых букв, идущих подряд, только одну букву.