



# **СИМВОЛЬНЫЕ И СТРОКОВЫЕ ВЕЛИЧИНЫ**

# ОГЛАВЛЕНИЕ

- Символьные и Строковые величины
- Сравнение переменных
- Сложение переменных
- Функция Функция Concat
- Функция Length
- Функция Pos
- Функция Copy
- Функции Ord и Функции Ord и Chr Функции Ord и Chr
- Процедура Delete
- Процедура Insert
- Процедура Val
- Процедура Str
- Цикл с символьной переменной



# СИМВОЛЬНЫЕ И СТРОКОВЫЕ ВЕЛИЧИНЫ

Для работы с символами, словами и предложениями в Паскале существуют типы данных – **CHAR** и **STRING**, представляющие собой символы в кодировке Windows.

Тип **CHAR** (**символьный**) - значения занимают 1 байт и представляют собой 1 символ.

Тип **STRING** (**строковый**) – значением может быть любая последовательность символов длиной не более 255.

Переменные *должны быть* описаны в разделе описания переменных или констант:

*Имя:* **char**;

*Имя:* **string**;

или

*Имя:* **string**[*длина*];

где *Имя* – имя переменной;

**string**, **char** – строковый или символьный тип;

*длина* – константа типа **INTEGER**, определяющая максимальную длину строки.



Объявления:

**var**

**fio: string[30];**//строковая переменная fio длиной 30

**buff: string;** //строковая переменная buff длиной до 255

**СИМВОЛОВ**

**a:char;** //строковая переменная a длиной в 1 символ

**const**

**name='Иванов Петр';**

**s=' b';**

Если при объявлении строковой переменной длина не указывается, то она может быть до 255 символов.

К символам в строке можно обращаться, используя индекс:  
**stroka[5]** обозначает 5-тый символ в строке.

# СРАВНЕНИЕ СИМВОЛЬНЫХ ВЕЛИЧИН

Последовательность символов, являющаяся строкой, заключается в одинарные кавычки.

Строковую переменную можно сравнить с другой переменной или константой типа `STRING`, используя операторы `=`, `<`, `>`, `<=`, `>=`, `<>`.

Строки сравниваются посимвольно от первого символа.

1. Если все символы сравниваемых строк одинаковые, то такие строки считаются равными.

`'abcd' = 'abcd'`



# СРАВНЕНИЕ СИМВОЛЬНЫХ ВЕЛИЧИН

2. Если в одинаковых позициях строк находятся разные символы, большей считается та строка, у которой в этой позиции находится символ с большим кодом.

'abd' > 'abc'

3. Если строки имеют различную длину, но в общей части символы совпадают, то короткая строка меньше, чем длинная.

'abcd' > 'abc'



# СЛОЖЕНИЕ ПЕРЕМЕННЫХ

Объединение нескольких строк в одну: к концу первой строки присоединяется начало второй и т. д.

Длина результирующей строки должна быть  $\leq 255$  символов.

Пример:

```
c:= 'Иван' + ' ' + 'Петров';
```

```
write(c); //будет выведено 'Иван Петров'
```



## Функция *LENGTH*

Возвращает длину строки.

*Length (параметр)*

*Параметр* – переменная или константа строкового типа. Возвращаемое значение (целое число) – количество символов, из которых состоит строка.

**Пример,**  
**n:=length ('Иванов');** - значение  
переменной **n** равно 6.



# ЗАДАНИЯ

1. Ввести строку символов. Определить и вывести на печать ее длину. Напечатать первый и последний символы строки.

2. Ввести две фамилии. Определить какая фамилия имеет большую длину. Результат вывести в виде:

*Фамилия «Иваненко» имеет больше символов чем «Петров»*

3. Описать строковую константу «Преобразование» и вывести символы с 4-го по 8-ой

4. Описать строковую константу  $k$ =«Кабинет» и строковую переменную **pred** для хранения названия предмета. Ввести с клавиатуры название предмета (в нужном падеже). Получить в переменной **res** полное название кабинета.



# ФУНКЦИЯ CONCAT

**Concat(s1,...,sn)**

возвращает строку, являющуюся результатом слияния строк  $s_1, \dots, s_n$ .

Результат тот же, что у выражения  $s_1+s_2+\dots+s_n$ .

## Пример

**S1 := 'город '; S2:='Киев';**

**S:=CONCAT(S1,S2);**

**S3:=S1+S2;**

значением переменных **S** и **S3** будет 'город Киев'.

## ЗАДАНИЯ

Решить задачу 4 с использованием функции Concat

Описать строковую константу `k='Кабинет'` и строковую переменную `pred`. Ввести с клавиатуры название предмета (в нужном падеже). Получить в переменной `res` полное название кабинета.

Объявить константу `slovo1='волна'`. Сравнить ее с переменной `slovo2`, присваивая ей значения: `'волга'`, `'вол'`, `'ворон'`.



# ФУНКЦИЯ POS

Позволяет определить положение подстроки в строке.  
Если подстрока не найдена возвращается 0. Общий вид:

**Pos**(*Подстрока*, *Строка*);

где *Подстрока* – строковая константа или переменная, которую надо найти в строковой константе или переменной *Строка*.

Пример

```
p:=pos('Петербург' , 'Санкт-Петербург');  
Значение переменной p будет равно 7.
```

# ФУНКЦИЯ СОРУ

Позволяет выделить фрагмент строки. Общий вид:

**Сору**(*Строка*, *p*, *n*);

где *Строка* – переменная строкового типа, содержащая строку, фрагмент которой надо получить;

*p* – номер символа, с которого начинается выделяемая подстрока;

*n* – длина выделяемой подстроки.

## Пример

```
St := 'инженер Иванов';
```

```
Fam := сору(st,9,6);
```

значением переменной **fam** будет строка 'Иванов'.

# ФУНКЦИЯ ORD

Возвращает символ с кодом X. Общий вид: Ord(x)

## Пример

```
m:=ord(8);  
m1:=ord('8');  
Writeln(m,' ',m1);
```

Результат: 8 56

# ФУНКЦИЯ CHR

Общий вид: Chr(x).

Для символов **x** - **char** возвращает их код.

Для **целых x** возвращает само значение.

## Пример

```
Writeln(chr(243),' ',chr(222));
```

Результат: у Ю

# СТРОКОВЫЕ ПРОЦЕДУРЫ

## ПРОЦЕДУРА DELETE

Позволяет удалить часть строки. Общий вид:

**DELETE**(*Строка*,*p*,*n*);

где *Строка* – переменная строкового типа;

*p* – номер символа, с которого начинается удаляемая подстрока;

*n* – длина удаляемой подстроки.

Пример

**S**:= 'город Санкт-Петербург';

**delete**(**S**,7,6);

значением переменной **S** будет строка 'город Петербург'.

# ПРОЦЕДУРА INSERT

Позволяет вставить подстроку в строку символов.

Строка раздвигается. В общем виде обращение к процедуре выглядит так:

**Insert (subs, s, index);**

вставляет подстроку **subs** в строку **S** с позиции **index**.

## Пример

```
S := 'город Петербург';
```

```
INSERT('Санкт-', S, 7);
```

значением переменной **S** будет строка 'город Санкт-Петербург'.



# ПРОЦЕДУРА VAL

Преобразует строку *S* к числовому представлению и записывает результат в переменную *V*.

## Val(*S*, *V*, CODE)

где *S* – строковая константа или переменная, содержащая изображение числа;

*V* – переменная, которой должно быть присвоено значение числа, изображенного в строке;

*CODE* – возвращаемый процедурой код ошибки. Если строка может быть преобразована в число, то код ошибки равен нулю.

### Пример

```
S:='1234';
```

```
VAL(S, X, A);
```

значение переменной *X* будет равно 1234, а переменной *A* - ноль.

# ПРОЦЕДУРА STR

Преобразует число в строку.

**Str**(*N*, *S*) ;

где *N*– переменная, которая должна быть преобразована в строку;

*S*– строковая переменная.

## Пример

В результате выполнения строк программы:

`X:=1234;`

`str(X,S);`

значением переменной *S* будет строка '1234'.

## ЦИКЛ С СИМВОЛЬНОЙ ПЕРЕМЕННОЙ

*Переменная* - параметр цикла может иметь любой порядковый тип (*целый, символьный, перечисляемый или интервальный*). При этом типы начального и конечного значения должны соответствовать типу параметра цикла.

### Пример

```
program prim;
var    i:integer;
en: (red, green, blue, white); {перечисляемый тип}
c: char;           {символьный тип}
a: 0..10;         {интервальный тип для чисел}
a: 'с..z'; {интервальный тип для символов}
begin
for en:=red to blue do write(Ord(en):2); {выводится 0 1 2}
for c:='a' to 'z' do write(c); {выводится символы 'abcd ...xyz'}
for i:=0 to 10 do begin a:=i; write(a); end; {выводится символы '01...910'}
end.
```