

Порядковые типы данных

Порядковые (ordinal) типы

Таким типам соответствуют данные, которые поддерживают понятия «предшествующее» и «последующее» значения

Например:

Для целого числа 5 можно определенно сказать, что ему предшествует число 4, а следующее за ним - число 6.

С другой стороны невозможно сказать, какое число непосредственно предшествует вещественному числу 5.0.

Порядковые типы в Pascal

- целые типы;
- символьные типы;
- булевы типы;
- ограниченные типы.

Функции для порядковых типов

Succ(<значение >)

Возвращает следующее значение.

Например:

$\text{Succ}(123) = 124$

$\text{Succ}(\text{false}) = \text{true}$

$\text{Succ}(\text{'a'}) = \text{'b'}$

$\text{Succ}(\text{'я'}) = \text{\#0}$

Функции для порядковых типов

Pred(<значение >)

Возвращает предыдущее значение.

Например:

Pred(123) = 122

Pred(false) = true

Pred('я') = 'ю'

Pred(#0) = 'я'

Функции для порядковых типов

Ord(<значение >)

Возвращает целое число,
соответствующее номеру <значения> во
множестве возможных значений типа.

Например:

$\text{Ord}(123) = 123$

$\text{Ord}(-123) = -123$

$\text{Ord}('я') = 255$

$\text{Ord}(\#0) = 0$

Функции для порядковых типов

High(<имя переменной>)

High(<значение>)

High(<тип>)

Возвращает максимальное значение.

Например:

High('x') = 'я'

High(char) = 'я'

High(10) = 2147483647

High(integer) = 2147483647

Функции для порядковых типов

Low(<имя переменной>)

Low(<значение>)

Low(<тип>)

Возвращает минимальное значение.

Например:

Low('a') = #0

Low(integer) = -2147483648

Функции для порядковых типов

Inc(var x [; n: integer]);

Dec(var x [; n: integer]);

Увеличивает(уменьшает) значение x
на величину n.

Например:

a:=5; inc(a); → a станет равно 5

b:=5; dec(b,2); -> b станет равно 3

Целые типы данных

Integer от -2147483648 до 2147483647

Cardinal от 0 до 4294967295

Оба типа занимают в памяти 4 байта

Int64

Byte от 0 до 255

Операции над целыми числами

Арифметические + - * div mod

Логические побитовые and or xor

10 and 5

```
      1010
      and
    0101
    -----
    0000
```

Операции сдвигов shr shl

11 shr 2 1011 -> 10 (2)

5 shl 2 101 -> 10100 (20)

Чему равно \$125?

$$\begin{aligned} \$125 &= 1 \cdot 16^2 + 2 \cdot 16^1 + 5 \cdot 16^0 = \\ &256 + 32 + 5 = 293 = \end{aligned}$$

(0001 0010 0101 в двоичном коде)

$$\begin{aligned} \$FF &= 15 \cdot 16^1 + 15 \cdot 16^0 = \\ &240 + 15 = 255 \end{aligned}$$

(1111 1111 в двоичном коде)

Символьные типы данных

Тип Char - множество из 256 символов.

Каждый символ этого типа занимает 1 байт. Символы упорядочены в соответствии с таблицей кодировки ANSI (American National Standard Code for Information Interchange).

Тип WideChar - множество из 65536 символов Unicode.

Каждый символ Unicode занимает 2 байта. Первые 256 символов Unicode соответствуют символам ANSI

Символ	Код	Двоичное представление	16-ричное представление
Нет символа	0	0000 0000	\$00
Пробел	32	0010 0000	\$20
!	33	0010 0001	\$21
...			
0	48	0011 0000	\$30
1	49	0011 0001	\$31
2	50	0011 0010	\$32
3	51	0011 0011	\$33
...			
A	65	0100 0001	\$41
B	66	0100 0010	\$42
C	67	0100 0011	\$43
a	97	0110 0001	\$61
b	98	0110 0010	\$62

Запись СИМВОЛЬНЫХ КОНСТАНТ

- '1' 'z' 'я' '-'
- #8 #13 #49 #255
- chr(65) chr(255)

Функции и операции

Chr(код символа)

Возвращает символ по заданному коду

Ord(символ)

Возвращает код символа

Операции \leq \lt \gt $>$

(Сравниваются коды символов)

Операция $+$

допустима при сложении со строкой

Полезные особенности таблицы кодировки

Разница в кодировке больших и маленьких букв и латинского и русского алфавита равна 20 или 32 .

Следовательно, чтобы превратить строчную букву в прописную, достаточно из ее кода вычесть 32 .

Число 32 можно и не помнить, так как его можно получить так:

$\text{ord}('z') - \text{ord}('Z')$, или $\text{ord}('я') - \text{ord}('Я')$.

Полезные особенности таблицы кодировки

У символов цифр младший полубайт соответствует ее числовому значению, а старшему полубайту соответствует десятичное число 48.

Чтобы преобразовать символ цифры в число, достаточно из кода символа цифры вычесть 48.

...

```
var c : char; n : integer;
```

...

```
n := ord(c) – 48; // равносильно ord(c) – ord('0');
```

Ограниченный тип данных

Это разновидность порядкового типа, когда диапазон возможных значений искусственно сужен.

Например:

```
type    Caps = 'A'..'Z';  
var     bigLetter : Caps;  
        month : 1..12;
```

Множество и операция in

Интервальный тип можно использовать для задания множества данных порядкового типа, например ['a'..'z'] задает множество строчных латинских букв.

Для проверки принадлежности некоторого значения множеству можно использовать операцию in, например:

5 in [0..9] вернет true

'я' in ['a'..'z'] вернет false

Цикл For

For <параметр цикла> := <начальное значение> **to** <конечное значение> **do**
begin
 <инструкции, называемые телом цикла,>
end;

For<параметр цикла>:= начальное значение>**downto**<конечное значение> **do**
begin
 <инструкции, называемые телом цикла,>
end;

Порядок выполнения инструкции **for...to..do**

1. Вычисляются <начальное значение> и <конечное значение >.
2. Переменной < параметр цикла > присваивается <начальное значение>.
3. Если значение <параметра цикла> превышает <конечное значение>, то цикл заканчивается.
4. Выполняется тело цикла.
5. Переменной < параметр цикла > присваивается следующее значение.
6. Выполнение продолжается с пункта 3.



Пример использования цикла for

```
function factorial(n:integer):int64;  
    var i:integer;  
begin  
    result:=1;  
    for i:=1 to n do  
        result:=result*i  
    end;
```


Пример использования цикла for

```
function power(x:real; n:integer):real;  
    var i:integer;  
begin  
    result:=1;  
    for i:=1 to n do  
        result:=result*x;  
end;
```

Среднее арифметическое (Цикл с while)

```
function average( n : integer ) : real;  
  var sum:real; count: integer;  
begin  
  count := n; sum := 0;  
  while count > 0 do  
  begin  
    sum := sum + random();  
    count := count - 1;  
  end;  
  result := sum / n;  
end;
```

Среднее арифметическое (Цикл с for)

```
function average( n : integer ) : real;  
  var sum:real; count: integer;  
begin  
  sum := 0;  
  for count :=n downTo 1 do  
  begin  
    sum := sum + random();  
  end;  
  result := sum / n;  
end;
```

Какой будет result?

```
...  
b:=10; result :=0;  
for i:=1 to b do  
begin  
    result := result +1;  
    b:=b-1;  
end;  
...
```