



Ordinální typy

standardní: integer, char, boolean

Vlastnosti ordinálních typů:

1. hodnot je konečný počet
a hodnoty jsou uspořádány
2. ke každé hodnotě (kromě té největší) existuje
bezprostřední následník
a ke každé hodnotě (kromě té nejmenší) existuje
bezprostřední předchůdce

Funkce

ORD ordinální hodnota

boolean

ord(FALSE)=0 ord(TRUE)=1

integer

ord(N)=N

char

ord(c)= *podle tabulky znaků*

PRED předchůdce

```
pred( -123 )=-124  
pred( 0 )=-1  
pred( TRUE )=FALSE  
pred( '3' )='2'
```

SUCC následník

```
succ( -123 )=-122  
succ( 9 )=10  
succ( FALSE )=TRUE //POZOR BP, FPC...  
succ( '3' )='4'
```

Příklad

```
var zn: char;  
begin  
  zn := 'A';  
  repeat  
    writeln( 'Znak ', zn, ' má číslo ', ord(zn) );  
    zn := succ( zn )  
  until zn > 'Z'  
end.
```



**Jak jednoduše zjistit,
jsou-li v naší znakové sadě písmena
bezprostředně za sebou**



Výčtové typy

Definice typu:

`identifikátor = popis typu`

Popis výčtového typu:

`(identifikátor { ,identifikátor })`

Příklad

```
den_tydne = (pondeli, utory,  streda,
             ctvrtek,  patek,  sobota,  nedele);
var den1, den2: den_tydne;
...
den1 := ctvrtek;
den2 := pondeli;
den1 := den2;
den1 := succ( den1 );
```

? pondeli vs. 'pondeli' ?

Typ interval

popis typu interval

`<konstanta>..<konstanta>`

např.

```
male_cislo = 0..255;
```

```
cislice    = '0'..'9';
```

```
pracovni_dny = pondeli..patek;
```

```
cisla_od_m_do_n = m..n;
```

Typ interval

obě konstanty:

- . stejný typ
- . ordinální typ
- . první \leq druhá

Interval je podmnožinou hostitelského typu

=> všechny jeho hodnoty jsou zároveň hodnotami hostitelského typu

definice typů jsou uvozeny klíčovým slovem **TYPE**

Příkaz cyklu FOR

```
for <identifikátor> := <výraz>  
  to/downto <výraz> do <příkaz>
```

= pro cykly, kde **předem** známe počet opakování

Např.

```
for i:=1 to N do  
  soucet := soucet + i;  
for zn := 'a' to 'z' do  
  writeln( 'znak ', zn,  
          ' má číslo ', ord(zn) );
```

```
for i:=1 to N do ...
```

```
  i := 1;
```

```
  while i<=N do
```

```
  begin
```

```
    ...
```

```
    i := succ(i)
```

```
  end;
```

```
for i:=N downto 1 do ...
```

```
  i := N;
```

```
  while i>=1 do
```

```
  begin
```

```
    ...
```

```
    i := pred(i)
```

```
  end;
```

`continue`

ukončení těla cyklu

`break`

vyskočení z cyklu

!! Nedosazujte do řídicí proměnné cyklu !!

Strukturované datové typy

umíme strukturovat příkazy,
můžeme strukturovat i data

POLE

slovo

1	2	3	4	5	6	7	8	9	10	11	12
P	R	O	G	R	A	M	O	V	Á	N	Í

popis typu:

`array[typ{ , typ}] of typ`

`slovo: array[1..12] of char;`

Pole

indexovaná proměnná

proměnná [výraz { , výraz }]

slovo[3] = 'O'

slovo[9] = 'V'

příklad: obracet slova

příklad: frekvence znaků

Eratosthenovo síto

(Έρατοσθένης,

276/272 - 194 př. n. l. v Alexandrii)

Vícerozměrné pole

= podle definice

```
slovník = array[1..VelikostSlovníku] of  
          array[jazyky] of  
            array[1..MaxDelka] of char
```

lze zapsat i jako

```
slovník = array[1..VelikostSlovníku,  
                jazyky,  
                1..MaxDelka] of char
```

podobně

```
sl[CisloHesla][jazyk][pismeno]
```

lze zapsat i jako

```
sl[CisloHesla, jazyk, pismeno]
```

? Co znamená zápis

```
sl[CisloHesla[jazyk]][pismeno] ?
```

? Kolik prvků typu integer má pole

A: array[-2..2, 3..8, boolean] of integer ?

Vyhledávání v poli

Úloha:

V neuspořádaném poli najít prvek s jednou konkrétní hodnotou

```
1:  for i:=zac to kon do
      if P[i]=HLEDANY then...
```

```
2:  i:=zac;
      while (i<=kon) and (P[i]<>HLEDANY] do
          i := i+1;
```

! Pokud vyhodnocuje booleovské výrazy nezkráceně, dojde k chybě ! (RangeCheck!)

3: trik: „vyhledávání se zarážkou“

pole o 1 prvek delší a tam umístíme hledanou hodnotu

```
P[kon+1] := HLEDANY;  
i := zac;  
while P[i] <> HLEDANY do  
    i := i+1;  
if i = kon+1 then... { nebyl tam }
```

Strukturované konstanty v BP

const

```
<jmeno>: <typ> = <hodnota>;
```

Příklad:

const

```
A: array[1..10] of integer
  = (2,3,5,7,11, 13,17,19,23,29);
AA: array[boolean, 1..5] of char
  = ( ('f','a','l','s','e'),
      (' ','t','r','u','e') );
```

**Ve skutečnosti to jsou jen proměnné
s počáteční hodnotou!**

Representace znakových řetězců (nejen v pascalu)

pole

- a) ukládat délku
- b) ukončovací znak

V TP:

typ string

var

```
a: string;  
b, c: string[10];  
d: string[50];
```

`string[MaxDelka]`

`string[N]` je representováno jako

`array[0..N]` of `char`,

0-tý prvek obsahuje délku řetězce
(jako `char`)

`length(s)`

`copy(s, odkud, kolik)`

`pos(co, kde)`, vrací 0, pokud neobsahuje

Příklad: nalezení všech výskytů podřetězce

Příklad: náhrada podřetězce

Těžká úloha:

**Napište program, který vytiskne
svůj vlastní zdrojový kód
(aniž by ho odněkud četl).**

Ten program nemusí dělat nic jiného.

Pokud se vám to podaří, pošlete mailem, Subj.: „SELF“.

