



Podmínka

= něco, co JE, nebo NENÍ splněno

typ Boolean

hodnoty: **TRUE** pravda

FALSE lež

domluva (optimistická):

FALSE < TRUE

když X , Y jsou (číselné) výrazy, potom

$X = Y$

$X \langle \rangle Y$

$X < Y$

$X > Y$

$X \leq Y$

$X \geq Y$

jsou výrazy typu boolean.

$2 < 3$

$ABS(x-3) < ABS(y-3)$

typ Boolean - operace

NOT negace

AND konjunkce

OR disjunkce

not P P neplatí

P and Q platí P i Q současně

P or Q platí P nebo Q (nebo oboje)

? jak zapsat VYLUČUJÍCÍ NEBO ?

? a implikaci ?

typ Boolean - priority operací

1) negace

2) konjunkce

3) disjunkce

POZOR! POZOR! POZOR! POZOR!

Logické operátory mají vyšší prioritu než relační operátory.

Takže $x < y$ and $y < z$

se vyhodnocuje jako

$x < (y \text{ and } y) < z$

= CHYBA!

Zpět k definici IF, WHILE, REPEAT:

„podmínka“

znamená

„výraz typu boolean“

příklad MONOTONIE...

Zkrácené vyhodnocování

výrazů typu `boolean`

P or Q pokud `P=TRUE`,
výsledek bude `TRUE`
=> nezávisí na Q

R and S pokud `R=FALSE`,
výsledek bude `FALSE`
=> nezávisí na S

Pro určení hodnoty výrazu (v takovém případě)
už nemusíme vyhodnocovat ostatní jeho členy.

ÚPLNÉ VYHODNOCOVÁNÍ

vyhodnotí všechny podvýrazy

ZKRÁCENÉ VYHODNOCOVÁNÍ

skončí vyhodnocování výrazu nebo i podvýrazu
ve chvíli, kdy dokáže určit výsledek

příklad hledání čísla s logaritmem...

příklad test prvočíselnosti...

```
... while PRVOCISLO and (D<=MEZ) do
```

práce se znaky: typ CHAR = písmeno

konstanty typu CHAR: '@', ''''

příklad

```
var c: char;  
begin  
    c := '?';  
    repeat  
        read( c )  
    until c='*';  
    read( c );  
    write( c )  
end.
```

kód znaku

každému znaku, který patří do znakové sady, je přiřazeno určité číslo, tzv. **Ordinální číslo znaku**

standardní funkce:

ORD (c) char -> integer
CHR (x) integer -> char

příklad

ord ('A') = 65
chr (33) = '!'
chr (-5) = **CHYBA!**

Kódy znaků závisí na počítači a překladači, norma požaduje:

- '0', '1', ..., '9'

v tomto pořadí a bezprostředně za sebou

- 'A', 'B', ..., 'Z'

v tomto pořadí

- jsou-li k dispozici i malá písmena, musí pro ně platit stejná podmínka jako pro velká písmena

Otázka:

jak zapsat podmínku

„znak Z je číslice“

?

Odpořěd' 1:

`(Z='0') or (z='1') or...or (Z='9')`

Odpořed' 2:

(z >= '0') and (z <= '9')

Příklad:

Vstup celých čísel znak po znaku

$C_0C_1C_2C_3 \dots C_n$

(zápis čísla bez znaménka)

$$10^n * C_0 + 10^{n-1} * C_1 + \dots + 10^0 * C_n$$

$$= (\dots (c_0 * 10 + c_1) \\ * 10 + c_2) \\ * 10 + c_3) \\ * \dots) \\ * 10 + c_n$$

HORNEROVO SCHEMA

Hodnota číslice

$$n = \text{ord}(n) - \text{ord}('0')$$

příklad čtení čísla, se znaménkem a mezerami

Textový soubor

posloupnost znaků rozdělená do řádek

READ (x) načtení x

READLN (x) načtení x
a potom přechod na další řádku

oba příkazy čtou ze STANDARDNÍHO
VSTUPNÍHO TEXTOVÉHO SOUBORU

WRITE (x) výstup x

WRITELN (x) výstup x

a potom přechod na další řádku

oba příkazy píší na **STANDARDNÍ VÝSTUPNÍ
TEXTOVÝ SOUBOR**

Lze použít i pro jiné textové soubory.

Představa o textovém souboru

- . na konci každého řádku je zvláštní znak
ODDĚLOVAČ ŘÁDEK
- . na konci každého textového souboru je zvláštní znak
UKONČOVACÍ ZNAK SOUBORU
- . tyto zvláštní znaky **NEPATŘÍ** od množiny hodnot typu **CHAR**
- . jejich smysl je dávat souboru **STRUKTURU**

Základní akce = vstup/výstup jednoho znaku.

`Read(c)` naplní `c` a posune se v souboru o jeden znak dále. **Zpátky NELZE.**

Po přečtení posledního znaku nelze číst dál, pokus o čtení způsobí chybu.

Standardní funkce

- `eof` end of file
- `eoln` end of line

```
read( x1, ..., xn )
```

```
begin  
    read( x1 );  
    ...  
    read( xn )  
end;
```

```
readln( x1, ..., xn )
```

```
begin  
    read( x1 );  
    ...  
    read( xn );  
    readln  
end;
```


Skutečná implementace (nejen v BP):

- . řádky textových souborů jsou odděleny dvojicí znaků CR a LF (`chr(13)` a `chr(10)`)
(Inebo jenom CR nebo jenom LF!)
- . textový soubor ukončen znakem EOF (`chr(26)`)
(pokud je ukončen)
- . čtení čísla končí až na bílém znaku za číslem
=> vstup

123AB

způsobí chybu.

Výstup do standardního výstupního souboru

vystupovat mohou hodnoty typu

`char`, `integer`, `real`, `boolean`

...a znakové řetězce

Příkaz `writeln` zapíše jen oddělovač řádek.

```
write( x1, ..., xn )
```

```
begin  
    write( x1 );  
    ...  
    write( xn )  
end;
```

```
writeln( x1, ..., xn )
```

```
begin  
    write( x1 );  
    ...  
    write( xn );  
    writeln  
end;
```

Formátování výstupu

výraz : délka

Význam:

Hodnota výraz má být zapsána na výstup tolika znaky, kolik je hodnota délka (to může být také výraz!).

Ne vždy to lze splnit.

Pokud to splnit lze, doplňuje se mezerami zleva.

Boolean se tiskne jako TRUE nebo FALSE.

Real:

- **v semilogaritmickém tvaru**
- **jedna číslice před desetinou tečkou**
- **nejméně jedna číslice za desetinou tečkou**
- **E nebo e**
- **znaménko exponentu**
- **exponent**

výraz : délka : míst

Textové soubory v TP

1. deklarace

```
var f: text;
```

2. přiřazení

```
assign( f, 'c:\vstup.txt' );
```

3. otevření

```
reset( f ) NEBO rewrite( f )
```

4. vstup

```
read( f, ...)
```

```
readln( f, ...)
```

výstup

```
write( f, ...)
```

```
writeln( f, ...)
```

zavření

```
close( f )
```

