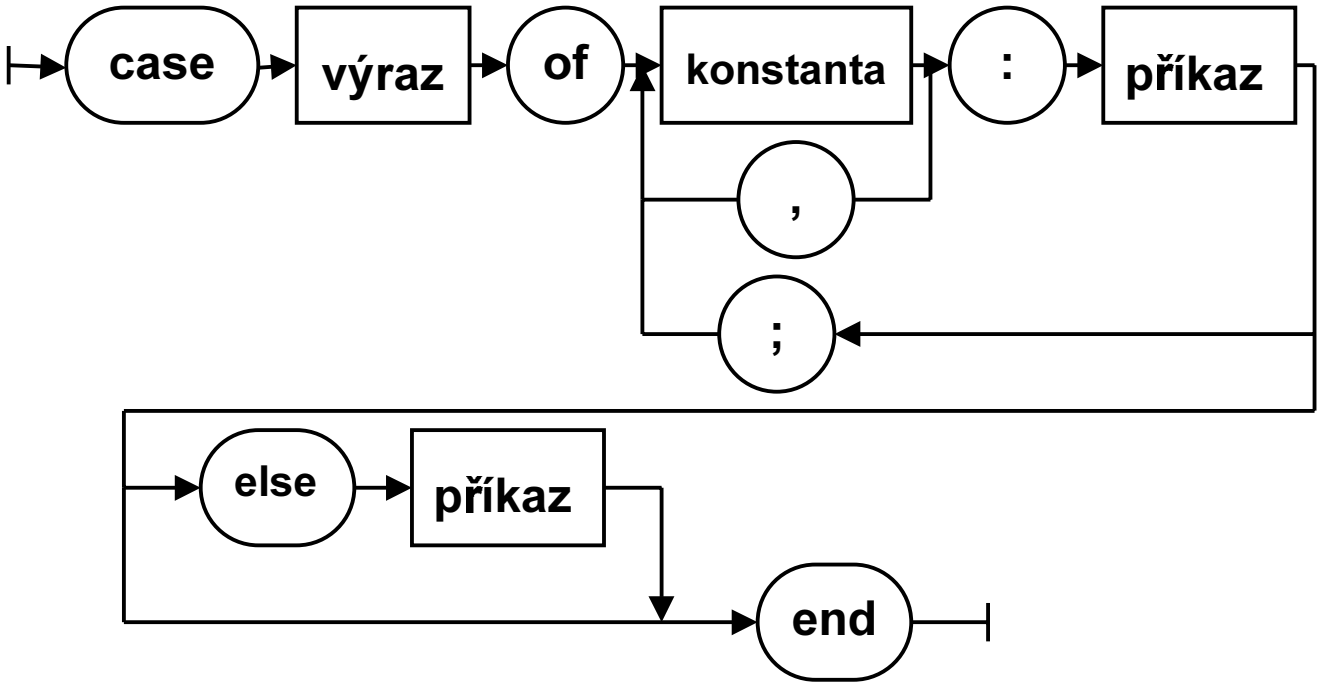
Příkaz case

– **umožňuje provedení jedné z několika možných alternativ na základě hodnoty výrazu ordinálního typu**



selektor větvení (obecně výraz ordinálního typu, rozsah hodnot maximálně -32768 .. 32767)

case* V *of
A1: P1;
A2: P2
….
Ai: Pi
….
An: Pn;

Ai návěští alternativy (stejný typ jako selektor větvení)

Pi příkaz který se vykoná pokud Ai=V

***else* Pe;** ← **vykoná se pokud není splněna žádná alternativa (nepovinná část)**

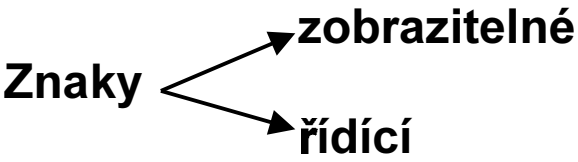
end;

Př.

```
var l : shortint;
begin
  …
case l of
  0,2,4,6,8 : writeln('suda cislice');
  1,3,5,7,9 : writeln('licha cislice');
  else        writeln(' Cislo zaporne nebo vetsi nez 10');
end;
 …
end.
```

Typ char

Proměnné typu char umožňují uložení znaků v paměti počítače.



Znaky jsou v paměti zobrazeny v podobě čísel, každému znaku odpovídá určité ordinální číslo (znakový kód).

Znakové kódy

- **ASCII (*American Standard Code for Information Interchange*) - původně 7 bitový (128 znaků - 95 zobrazitelných znaků, 33 řídících znaků) s rozvojem osobních počítačů rozšířen na 8 bitů (256 znaků, znaky s ordinálním číslem > 127 vyhrazeny pro znaky národních abeced, semigrafické symboly atd.)**
- **EBCDIC (Extended Binary Coded Decimal Interchange Code) - 8 bitový (256 znaků, ne každému ordinálnímu číslu přísluší znak)**

Důležité vlastnosti kódu pro zobrazení znaků:

1. **Podmnožina znaků vyjadřující číslice musí být uspořádána podle velikosti číslic tj.**
   '0' < '1' < '2' ….. < '9'
   a souvislá,
2. **Podmnožina písmen velké a malé abecedy musí být abecedně uspořádaná tj.**
   'a' < 'b' < 'c' … 'z' a 'A' < 'B' … 'Z'

Operace s typem char

Literály typu char : 'A' , ' ',

Konstanty a p roměnné - deklarace:

```
Const    Znak='*';   deklarace znakové konstanty

var   A, B:  char;   deklarace proměnné (1 byte)
```

Relační operace - oba operandy musí být typu *char*, výsledek je typu *boolean*

= <> < <= > >=

Znaky se porovnávají podle hodnoty ordinálního čísla.

Standardní funkce:

Chr(i) - *i* je typu *integer*, výsledek je typu *char.*

Výsledkem je znak odpovídající ordinálnímu číslu s hodnotou *i* .

Ord(zn) - *zn* je typu *char*, výsledek je typ *integer*.

Funkce vrací ordinální číslo znaku *zn*

Platí: Ord(Chr(i)) = i
Chr(Ord(zn)) = zn

Čtení znaku: (var ch : char)

```
Read( ch )
Readln(ch)
```

Zápis znaku:

```
Write(ch)
Writeln(ch)
```

Pokud proměnná *Ch* obsahuje znak reprezentující číslici, lze určit hodnotu číslice na základě následujícího vztahu

C := ord(Ch) - ord('0');

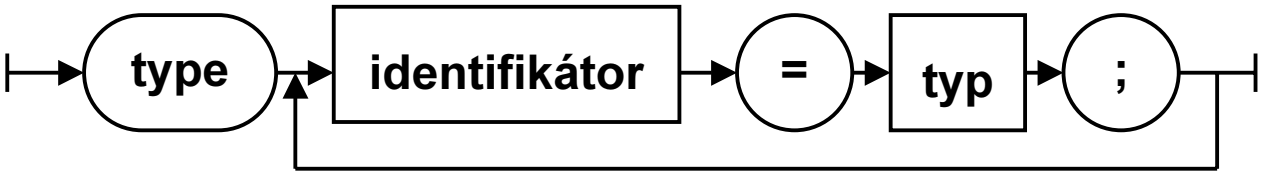
Příklad: Napište program, který přečte posloupnost znaků ukončenou tečkou a vytiskne

- **celkový počet písmen**
- **celkový počet číslic**
- **celkový počet ostatních znaků.**

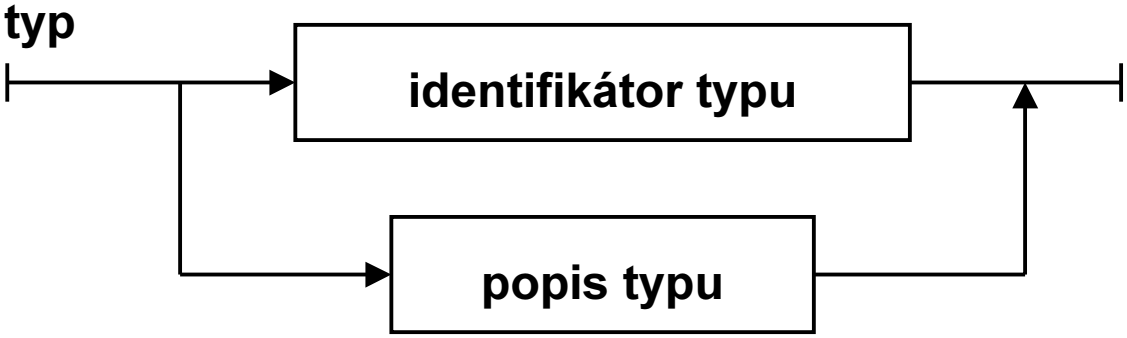
```
program pocty;
const TECKA='.';
var  PISMENA,CISLICE,OSTATNI:integer;
     CH : char;
begin
PISMENA:=0; CISLICE:=0; OSTATNI:=0;
writeln('Zadej vetu ukoncenou teckou');
repeat
read(CH);
case CH of
'0' .. '9' : inc(CISLICE);     { CISLICE:=CISLICE+1 }
'a' .. 'z',
'A' .. 'Z' : inc(PISMENA);
else inc(OSTATNI)
end;
until CH='.';
writeln('Pocet pismen  = ',PISMENA);
writeln('Pocet cislic  = ',CISLICE);
writeln('Ostatni znaky = ',OSTATNI);
end.
```

Typy definované uživatelem

deklarace typů



typ

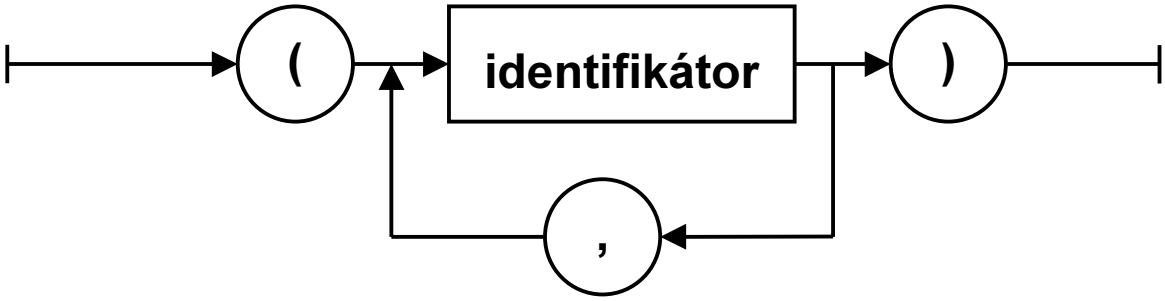



```
type    identifikátor = popis typu;
                  ....
        identifikátor = popis typu;
```

- **uživatelsky definovaný typ umožňuje vytvářet nové typy úpravou typů již existujících. Nově definovaný typ je možné používat v úseku deklarací proměnných, popř. k deklaraci parametrů procedur a funkcí**

Výčtový typ

popis výčtového typu



type T = (c_1,c_2, ..., c_n); c_i – i-tá hodnota typu

Výčtový typ:

- **ordinální typ, slouží k reprezentaci kvalitativních údajů (popis datových objektů, které nabývají malého počtu hodnot a kterými se rozlišují jejich určité stavy a rysy).**
- **je tvořen seznamem identifikátorů reprezentujících všechnu hodnoty tohoto typu.**
- **pořadím výčtu hodnot je stanoveno implicitní uspořádání množiny hodnot**
- **nesmí obsahovat číslo, znak (v apostrofech), popř. řetězec znaků;**

Př.

```
type rok= (leden, unor, brezen, duben, kveten, cerven,
          cervenec, srpen, zari, rijen, listopad,
          prosinec);
   barva= ( bila, zluta, cervena, modra, zelena );
```

Platí: leden < unor < …. < prosinec
bila < zluta < cervena

Pro funkci *ord* platí u výčtového typu platí:

$$Ord(c_k) = k - 1 \quad pro \quad k = 1, 2, \ldots, n$$

Typ *boolean* lze požadovat za speciální výčtový typ definovaný jako

type boolean = (false, true);

Operace vstupu a výstupu není pro výčtový typ definována !!!

Funkce *pred* a *succ*:

Succ(a)* - následující hodnota proměnné *a
Před(a)* - předchozí hodnota proměnné *a

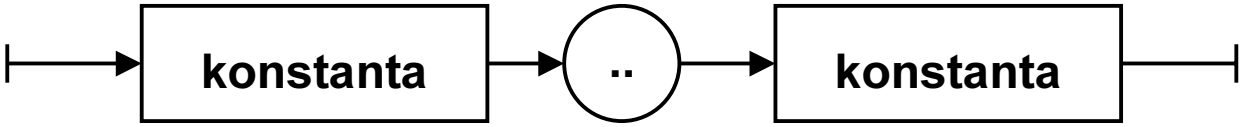
Funkce *Succ* a *Pred* nejsou definovány pro pravý resp. levý krajní prvek (tj. pro meze vyjmenovaného typu).
Př.

Pred(brezen) = unor
Succ(cervena) = modra

Typ interval

- **definuje podmnožinu hodnot již deklarovaného ordinálního typu**

popis typu interval



type t = min .. max;

min, ***max*** **jsou konstanty téhož ordinálního typu (tzv. hostitelského typu) pro které platí:**

min ≤ max

Hostitelskými typy intervalu jsou obvykle typ *integer*, *char* nebo *výčtový* typ. Interval z typu *boolean* nemá smysl.

Příklady:

```
type   pismeno = 'A' .. 'Z';
                                  } intervaly typu char
       Cislice   = '0'  .. '9';

       kladne  =  1 .. maxint ;
                                  } intervaly typu integer
       index   =  10 .. 30;
```

skolni_rok = zari .. cerven; interval výčtového typu

```
var  M: kladne;

     A : -10 ..100;

    SR: skolni_rok;
```

```
program prevod_znaky;

const MEZERA = ' ';
var CH : char;
   SGN,HODNOTA : integer;
   CHYBA,S: boolean;

begin
writeln(' Zadej posloupnost znaku ukoncenou mezerou');
repeat
read(CH);
until CH<>MEZERA;
CHYBA:=false; S:=false;
SGN:=1;
case CH of
'+'       : S:=true ;
'-'       : begin S:=true; SGN:=-1 end;
'0' .. '9' : begin
             SGN:=1; HODNOTA:=ord(CH)-ord('0');
            end;
else CHYBA:=true;
end;
read(ch);
while (not CHYBA) and (CH <> MEZERA) do begin
if (CH>='0') and (CH<='9') then begin
                          HODNOTA := HODNOTA*10+ord(CH)-
ord('0');
                           if S then S:=false;
                          end
                     else CHYBA:=true;
read(CH);
end;
if CHYBA or S then writeln('Neni cislo !!')
       else writeln('Je cislo s hodnotou H=',SGN*HODNOTA);
end.
```