Anotace

- Ordinalni typy - typ char, funkce ord, chr, succ, prev, inc, dec, Motivace: Máme dlouhé číslo (nebo číslo ve stringu).
- Zapis cisla v pozicni soustave, jeho vyhodnoceni Hornerovym schematem,
- Evaluace polynomu Hornerovym schematem,
- Umocnovani cisla,
- k rekurzi.

Ordinální datové typy

- Typy, jejichž hodnoty tvoří lineárně uspořádanou množinu (tedy pro každou dvojici je jasné, který prvek je větší).
- Z pascalských typů jsou to: `integer`, `longint`, `byte`, `char`, `word`, `boolean` a `shortint` (a další definované uživatelem, zejména výčtový typ a interval).
- Pro ordinální typy jsou definovány funkce "Ord", "Pred" a "Succ".

Funkce pro ordinální typy

- Funkce `Ord` vrátí ordinální hodnotu (tedy hodnotu pro část typů, konkrétně pro `integer`, `longint`, `byte`, `word` a `shortint`).
- Pro typ `char` vrátí ASCII hodnotu příslušného znaku. Umíme tedy zjistit ASCII-hodnotu jednotlivých znaků.
- Co naopak? Použijeme funkci `Chr`, která vrátí znak s příslušným číslem.
- Úloha: Jak převedeme číslo uložené v integeru na řetězec znaků?
- Řetězec je reprezentován jako pole znaků (až na drobná omezení).

Příklad

```
var a,i,j:integer; text,pom:string[255];
begin pom:='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';
      readln(a); i:=1; j:=1;
      while a<> 0 do
      begin pom[i]:=chr(a mod 10+48);
            a:=a div 10; i:=i+1;
      end;
      delete(pom,i,255-i); i:=i-1;
      text:=copy(pom,1,length(pom));
      while i>0 do
      begin text[j]:=pom[i];
            i:=i-1; j:=j+1;
      end;
      writeln(text);
end
```

Hornerovo schéma

- Co když chceme konvertovat obráceně? (řetězec na integer)
- Použijeme tzv. Hornerovo schéma, tedy začneme od začátku řetězce (nejvyšší číslice).
- Zjistíme její hodnotu a postupujeme indukcí:
  Dosavadní výsledek vynásobíme deseti a přičteme číslici na dalším řádu.

číslo $a_n a_{n-1} a_{n-2} ... a_0$ zapsané v desítkovém zápisu je vlastně $a_n 10^n + a_{n-1} 10^{n-1} + ... + a_0$. A platí:

$$a_n 10^n + a_{n-1} 10^{n-1} + ... + a_0 = (...((a_n * 10) + a_{n-1} * 10) + ... + a_1) * 10 + a_0$$

Tímto způsobem můžeme vyhodnocovat čísla zapsaná v různých (pozičních) soustavách (dvojkové, čtyřkové, pětkové, šestkové...).

Příklad

```
program x;
var a:string;
    i,hod:longint;
begin
     readln(a); i:=1; hod:=0;
     while i<=length(a) do
     begin
     hod:=10*hod+ord(a[i])-ord('0');
          i:=i+1;
     end;
     writeln(hod);
end.
```

Evaluace polynomu

- Máme zadán polynom $a_n x^n + a_{n-1} x^{n-1} + ... + a_0$.
- Chceme tento polynom vyhodnotit (evaluovat), tedy určit jeho hodnotu v zadaném bodě x.
- Možnosti?
- Hrubá síla (počítat $a_n x^n$, $a_{n-1} x^{n-1}$,... a sečíst)
- nebo Hornerovo schéma

$$\sum_{i=0}^{n} a_i x^i = ((...(a_n x + a_{n-1})x + ... + a_1)x + a_0).$$

Evaluace polynomu Hornerovým schématem

- 1: Načti koeficient u nejvyššího (dosud neprošlého) stupně,
- dosud načtené hodnoty vynásob hodnotou x,
- přičti hodnotu nově načteného koeficientu,
- GOTO 1;

Evaluace polynomu Hornerovým schématem

```
program nic;
var i,a,souc,stupen,x:integer;
{Vyhodnot polynom stupne stupen v bode x, do a
nacitej koeficienty}
begin
      readln(stupen); readln(x);
      souc:=0;
      for i:=0 to stupen do
      begin souc:=souc*x;
            readln(a);
            souc:=souc+a;
      end;
      writeln('Hodnota polynomu je: ',souc);
end.
```

Umocňování na vysoké exponenty

- Naivní přístup je lineární vůči exponentu.
- Také lze provést Hornerovým schématem,
- exponent převedeme do dvojkové soustavy, zaznamenáme si základ do průběžné proměnné a opakujeme pro každý bit exponentu (dokud nejsme na konci exponentu):
  - průběžný výsledek umocni na druhou, pokud je zpracovávaný bit exponentu 1, vynásob průběžný výsledek ještě základem.

Hornerovo schéma

mravní naučení

- Jde o velmi obecnou metodu, kterou lze použít ke všelijaké práci s čísly.
- Neukázali jsme si zdaleka všechny aplikace, očekává se, že budete případně Hornerovo schéma schopni (v případě potřeby) aplikovat samostatně!

Odbočka – labely a GOTO

- V Pascalu je možno provádět poměrně nekontrolované skoky po programu.
- Za sekcí globálních proměnných (`var`) vytvoříme sekci `label`, kde vyjmenujeme použité labely,
- následně můžeme těmito labely označit vybraná místa programu
- a pomocí `goto label;` udělat nepodmíněný skok.
- GOTO nepoužívejte, používám ho jen já v pseudokódu k pedagogickým účelům.

Vnořené funkce a procedury

Proceduru či funkci lze definovat i uvnitř jiné procedury nebo funkce:

```
procedure f(a:integer);
    procedure g(b:integer);
    begin
        writeln('Proc. g v procedure f s par-em ',b);
    end;
begin
    writeln('Procedura f s parametrem ',a);
    g(2);{Volame vnorenou proc. g}
end;
```

Scope resolution

- Procedura/funkce vidí jen funkce (procedury) vnořené přímo do sebe (ne do svých vnořených funkcí/procedur)!
- Vnořené funkce vnášejí další zmatek do významu proměnných (a dokonce procedur a funkcí).
- Identifikátor má takový význam, jaký je v danou chvíli "nejlokálnější".

Příklad

```
procedure f(h:integer);
    procedure g(b:integer);
        procedure h(c:integer);
        begin
            writeln('Procedura h s par. ',c);
        end;
    begin
        writeln('Procedura g s parametrem ',b);
        h(5);
    end;
begin
    writeln('Funkce f s parametrem ',h);
    g(3); f(5); {zatim OK, ale h(4) udela chybu!}
end;
```

Rekurze

- Může mít dobrý smysl volat z funkce sebe sama.
- Této metodě říkáme **rekurze.**
- Rekurze není nic jiného, než vtipně pojmenovaná indukce!
- Příklady: Úředníci na úřadech, Faktoriál, Caesarův kód...

Úředníci na úřadech

- Člověk chce nechat provést úřední výkon.
- Úředník požaduje vyplnění papírů vyžadující návštěvy dalších úřadů.
- Řešení:

```
procedure vypln(musime_vyplnit:seznam_papiru);
var x:seznam_papiru;
for papir in musime_vyplnit do
begin
    x:=zeptej_se_urednika(papir);
    if nonempty(x) then
       vypln(x);
end;
```

Faktoriál

- $n! = 1 \cdot 2 \cdot \ldots \cdot n$
- Jak tuto funkci naprogramovat?
- Cyklem:

```
fakt:=1;
for i:=1 to n do
    fakt:=fakt*i;
```

- nebo rekurzí.

Faktoriál rekurzí:

```
function faktorial(a:integer):integer;
begin
    if a<2 then
          faktorial:=1
    else  faktorial:=a*faktorial(a-1);
end;
```

Jaká je složitost výpočtu funkce faktoriál?