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,
- MAXINT, preteceni, dlouha cisla, typ real a jeho presnost.
- Operace s dlouhými čísly.

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 $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0$.

Evaluace polynomu

- Máme zadán polynom $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0$.
- Chceme tento polynom vyhodnotit (evaluovat), tedy určit jeho hodnotu v zadaném bodě x.

Evaluace polynomu

- Máme zadán polynom $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0$.
- Chceme tento polynom vyhodnotit (evaluovat), tedy určit jeho hodnotu v zadaném bodě x.
- Možnosti?

Evaluace polynomu

- Máme zadán polynom $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0$.
- 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)

Evaluace polynomu

- Máme zadán polynom $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0$.
- 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

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

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.
```

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.

Fronta a zásobník

- Fronta je datová struktura osazená operacemi `zařaď` a `vyřaď`,
- `zařaď` zařadí daný prvek na konec fronty,
- `vyřaď` vyřadí daný prvek ze začátku fronty.
- Zásobník je datová struktura osazená operacemi `push` a `pull`.
- `push` přidá na konec zásobníku, `pull` vytáhne z konce zásobníku.

Figurky na šachovnici anebo prohledávání grafu - algoritmus vlny

- Na děravé šachovnici je umístěn král. Určete jak rychle se dostane na určené (cílové) pole?

Figurky na šachovnici anebo prohledávání grafu - algoritmus vlny

- Na děravé šachovnici je umístěn král. Určete jak rychle se dostane na určené (cílové) pole?
- Vlastně jde o úlohu prohledávání grafu.

Figurky na šachovnici anebo prohledávání grafu - algoritmus vlny

- Na děravé šachovnici je umístěn král. Určete jak rychle se dostane na určené (cílové) pole?
- Vlastně jde o úlohu prohledávání grafu.
- Podobné: Thesseus hledá Mínótaura.

Figurky na šachovnici anebo prohledávání grafu - algoritmus vlny

- Na děravé šachovnici je umístěn král. Určete jak rychle se dostane na určené (cílové) pole?
- Vlastně jde o úlohu prohledávání grafu.
- Podobné: Thesseus hledá Mínótaura.
- Rozdíl: Tehdy nám nešlo o nejkratší cestu.

Algoritmus vlny (myšlenky)

- Na šachovnici vysadíme na specifikované políčko mravence,
- mravenci polezou rovnoměrně všemi dostupnými směry,
- mravenci zkusí všechny cesty, rychleji než první mravenec se tam dostat nelze.
- Jiná intuice: Na příslušné poličko vychrstneme vodu,
- voda teče všemi dostupnými cestami, až doteče na cílové políčko.

Algoritmus vlny (implementace)

- Vytvoř prázdnou frontu f.
- Nastav vzdálenost do všech políček kromě startovního na nekonečno, vzdálenost do startovního nastav na 0.
- Přidej do f startovní políčko.
- Dokud není fronta f prázdná, opakuj:
  - a:=vyřaď(f);
  - Pro všechny sousedy z pole a zkus:
  - Pokud vzdálenost do z je větší než vzdálenost do a +1, nastav políčku z vzdálenost $a+1$ a zařaď(z, f).