Cvičení Programování I

Cvičící: **Pavel Surynek, KTIML,**
pavel.surynek@seznam.cz
Semestr: **Zima 2005/2006**
Kroužek: **Matematika/59**
Rozvrh: **Pátek 10:40-12:10 (učebna K2)**

Stručné poznámky ke cvičení ze 6.1.2006

1. Organizační záležitosti. Termín pro odevzdání zápočtových programů je konec zkouškového. K zápočtovým programům je nutno dodat dokumentaci. Dokumentace by měla popisovat, jak se program ovládá z uživatelského hlediska a měla by poskytovat popis samotného programu tak, aby se v něm kdokoli, kdo zná Pascal, dokázal zorientovat.

Příští cvičení bude věnováno dotazům, nejasnostem, zápočtovým programům atd. podle přání studentů.

Jakékoli **dotazy** ke cvičení lze posílat na uvedenou e-mailovou adresu. Osobní konzultace ke cvičení lze dohodnout e-mailem (alespoň den předem).

2. Numerické výpočty. Mějme křivku v rovině zadanou parametricky, například:

$$x(t) = a_x t^4 + b_x t^3 + c_x t^2 + d_x t + e_x$$

$$y(t) = a_y t^4 + b_y t^3 + c_y t^2 + d_y t + e_y, \text{ kde } t \in [0,1] \text{ a } a_x,\ b_x,\ c_x,\ d_x,$$

e_x, a_y, b_y, c_y, d_y a e_y jsou reálné konstanty. Úkolem je křivku „bod po bodu“ nakreslit, tj. zvolíme nějaký krok pro parametr t, například 0.001, a postupně počítáme $x(t)$ a $y(t)$ pro $t = 0.0$, $t = 0.001$, $t = 0.002$, $t = 0.003$ atd. a výsledné body vykreslujeme. Musíme tedy $1000\times$ vypočítat hodnotu výrazů pro $x(t)$ a $y(t)$. Navrhněte schéma pro výpočet $x(t)$ a $y(t)$, které minimalizuje počet aritmetických operací.

Řešení. Nejdříve se podívejme na hloupé řešení.

```
const Ax = ...;
const Bx = ...;
...
const Ey = ...;

var xt, yt, t: real;

begin
  t := 0.0;
  while t < 1.0 do begin
    xt := Ax*t*t*t*t + Bx*t*t*t + Cx*t*t + Dx*t + Ex;
    yt := Ay*t*t*t*t + By*t*t*t + Cy*t*t + Dy*t + Ey;
    nakresli(xt, xy);
  end;
end.
```

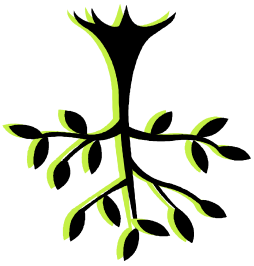
V každém opakování cyklu se použije 20 operací násobení a 8 operací sčítání. Zkusme chytřejší řešení, budeme vytýkat parametr t.

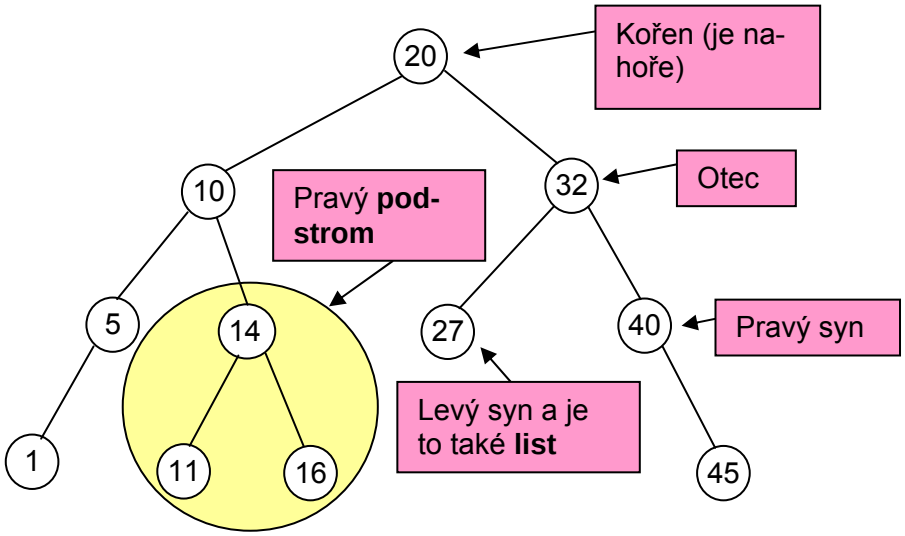
```
const Ax = ...;
const Bx = ...;
...
```

```
const Ey = ...;

var xt, yt, t: real;

begin
  t := 0.0;
  while t < 1.0 do begin
    xt := t*(t*(t*(t*Ax + Bx) + Cx) + Dx) + Ex;
    yt := t*(t*(t*(t*Ay + By) + Cy) + Dy) + Ey;
    nakresli(xt, xy);
  end;
end.
```

V každém opakování cyklu se použije 8 operací násobení a 8 operace sčítání. Tím jsme obdrželi více než dvakrát rychlejší program.
DCV (pro znalce diferenciálního počtu). Pokuste se program ještě urychlit. Zkuste co nejvíce eliminovat operaci násobení a zkuste ji nahradit operací sčítání.

3. Binární vyhledávací strom. Binární vyhledávací strom je datová struktura, která vypadá podobně jako na následujícím obrázku:





Jedná se tedy o strom, kde v každém uzlu je nějaká hodnota. Každý uzel stromu, kromě kořene, má otce. Každý uzel stromu má jednoho, nebo dva syny, nebo nemusí mít žádného syna. Platí, že když je nějaký vrchol ohodnocen hodnotou h, pak všechny vrcholy v jeho levém podstromu mají hodnoty menší než h. To samé symetricky, všechny vrcholy v jeho pravém podstromu mají hodnoty větší než h. Navrhněte reprezentaci binárního vyhledávacího stromu a naprogramujte v Pascalu procedury na vyhledání zadané hodnoty ve stromu, vložení nové hodnoty do stromu a vymazání hodnoty ze stromu.

Řešení. Strom budeme reprezentovat v poli, kde každý uzel stromu se bude nacházet v jedné buňce pole. U každého uzlu stromu si budeme pamatovat hodnotu v něm uloženou a jakého má otce a syny - to budou indexy do tohoto pole.

```
program BVS; { Binarni vyhledavaci strom }

const MAX = 100;

type UZEL = record
              otec: integer;
                { index otce, -1 pro koren }
```

```
              levy, pravy: integer;
                { indexy synu, -1, kdyz syn chybi }
              hod: integer;
                { hodnota v uzlu }
            end;

var pocet, koren: integer;
var strom:array[1..MAX] of UZEL;

procedure inicializuj;
begin
  koren := 1;
  pocet := 0;
end;

procedure vypis;
var i: integer;
begin
  for i:=1 to pocet do begin
    write('index:', i, ' ');
    write('hodnota:', strom[i].hod, ' ');
    write('index otce:', strom[i].otec, ' ');
    write('index l.syna:', strom[i].levy, ' ');
    write('index p.syna:', strom[i].pravy);
    writeln;
  end;
end;

function vyhledej(x: integer): boolean;
var index: integer;
begin
  if pocet = 0 then begin
    writeln('Nenalezeno.');
    vyhledej := false;
  end
  else begin
    index := koren;
    while index <> -1 do begin
      if x = strom[index].hod then begin
        writeln('Nalezeno.');
        vyhledej := true;
        break;
      end
      else begin
        if x > strom[index].hod then begin
          index := strom[index].pravy;
        end
        else begin { jiste plati, ze x < strom[index].hod }
          index := strom[index].levy;
        end;
      end;
    end;
    if index = -1 then begin
      writeln('Nenalezeno.');
      vyhledej := false;
    end;
  end;
end;
```

```
procedure vloz(x: integer);
var index, otec: integer;
begin
  if pocet = 0 then begin
    strom[1].otec := -1;
    strom[1].levy := -1;
    strom[1].pravy := -1;
    strom[1].hod := x;
    pocet := 1;
  end
  else begin
    index := koren;
    while true do begin
      if x = strom[index].hod then begin
        writeln('Nalezeno.');
        break;
      end
      else begin
        otec := index;
        if x > strom[index].hod then begin
          if strom[index].pravy = -1 then begin
            pocet := pocet + 1;
            strom[index].pravy := pocet;
            strom[pocet].otec := index;
            strom[pocet].levy := -1;
            strom[pocet].pravy := -1;
            strom[pocet].hod := x;
            break;
          end
          else begin
            index := strom[index].pravy;
          end;
        end
        else begin { jiste plati, ze x < strom[index].hod }
          if strom[index].levy = -1 then begin
            pocet := pocet + 1;
            strom[index].levy := pocet;
            strom[pocet].otec := index;
            strom[pocet].levy := -1;
            strom[pocet].pravy := -1;
            strom[pocet].hod := x;
            break;
          end
          else begin
            index := strom[index].levy;
          end;
        end;
      end;
    end;
  end;
end;

begin
  inicializuj;
  vloz(10);
  vloz(20);
```

```
  vloz(7);
  vloz(11);
  vloz(3);
  vloz(17);
  vypis;
end.
```

Operace na odstranění hodnoty z binárního vyhledávacího stromu je za DCV.

4. Generování kombinací a další. Stále nikdo nevyřešil generování všech možných kombinací dané třídy ze zadané množiny (to samé pro variace a permutace plus opakování). Je to stále DCV.

5. Prvočísla. Napište program pro nalezení prvních N prvočísel. N bude vstup programu.

Řešení. DCV (stačí si vyhledat příslušný algoritmus).