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 25.11.2005

1. Organizační záležitosti. První **série témat na zápočtový program** je zhruba následující: hry - piškvorky, karty, sudoku, kalendáře, textové editory, formátovaní textu, prohlížeče souborů se zalamováním řádků, simulátor procesoru, interpret jazyka, kalkulačka výrazová, kalkulačka na velká čísla, kreslítka na funkce, grafika - raytracing kuliček, vektorové kreslítko, bitmapové kreslítko, simulace - gravitace, aerodynamika, galaktické srážky, neuronové sítě - rozpoznávání písmen.

Dodatek k tématům na zápočtový program: dáma, šachy, procházení bludiště, skládání Rubikovy kostky, zobrazování 3d objektů, fraktály, simulace populací (kořist-predátor), šifrování-dešifrování, hádaní dalšího členu posloupnosti.

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

1. Halda v poli. Datovou strukturu halda je možné reprezentovat v poli. Nechť kořen haldy (nejmenší prvek) je v poli na prvním místě. Má-li uzel u v haldě nějakého potomka (nebo dva potomky), nechť jsou to uzly v (levý potomek) a w (pravý potomek), pak když u je v poli na místě i, je v v poli na místě $2*i$ a w je v poli na místě $2*i+1$. Napište v Pascalu program na vložení nového prvku do takto reprezentované haldy. Řešení pomocí obrázku jsme již ukazovali na jednom z minulých cvičení, nyní je třeba vše reprezentovat v rámci popsaného pole.

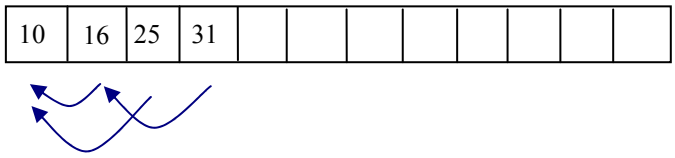
Řešení. Vložení nového prvku do hlady reprezentované v poli probíhá tak, že nový prvek vložíme do reprezentujícího pole na konec a poté opravujeme uspořádání pro prvky nacházející se na předcházejících pozicích. Když vložíme prvek do reprezentujícího pole na pozici i, pak je nutné prověřit podmínku na uspořádání pro prvek na pozicích i a $\lfloor i/2 \rfloor$. Pokud je podmínka na uspořádání porušena, prohodíme prvky na daných pozicích a pokračujeme v opravách podmínky na uspořádání (nyní ale nikoli pro pozici i ale pro pozici $\lfloor i/2 \rfloor$). Jinak operace vložení nové prvku končí.

```
program Halda_v_poli;

const MAX = 100; { maximalni pocet prvku v halde }

var halda: array[1..MAX] of integer; { reprezentujici pole }
var pocet: integer; { pocet prvku v halde }

procedure inicializuj;
begin
  pocet := 0;
end;

procedure vypis;
var i: integer;
begin
  for i:=1 to pocet do begin
    write(halda[i], ' ');
```

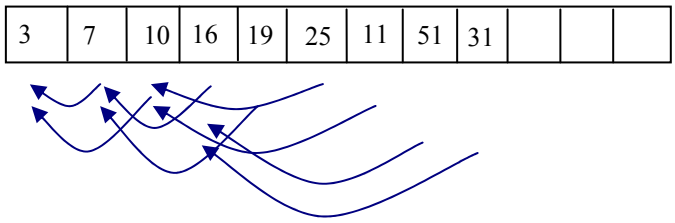
```
  end;
  writeln;
end;

procedure vloz(novy: integer);
var i, j, p: integer;
begin
  pocet := pocet + 1;
  halda[pocet] := novy;
  i := pocet;
  while i > 1 do begin
    j := i div 2;
    if halda[i] < halda[j] then begin
      p := halda[i];
      halda[i] := halda[j];
      halda[j] := p;
    end
    else begin
      break;
    end;
    i := j;
  end;
end;

begin
  inicializuj;
  vloz(10);
  vloz(16);
  vloz(25);
  vloz(31);
  vypis;
  vloz(19);
  vloz( 3);
  vloz(11);
  vloz(51);
  vloz( 7);
  vypis;
end.
```

Šipky znázorňují podmínky na uspořádání

Obsah pole reprezentujícího haldu v okamžiku prvního výpisu pomocí procedury „vypis“

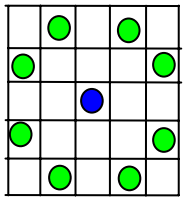


Obsah pole reprezentujícího haldu v okamžiku druhého výpisu pomocí procedury „vypis“



Asymptotická časová složitost vložení nového prvku do hlady je $O(\log_2 N)$, kde N aktuální velikost haldy. Asymptotická prostorová složitost přesně odpovídá počtu uložených prvků, tedy $O(N)$.

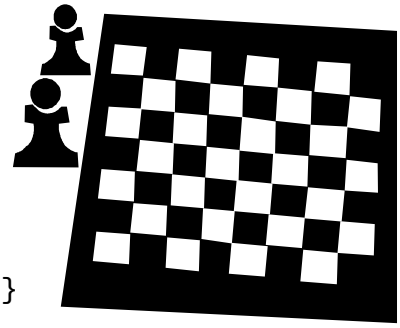
3. Koník. Je dána šachovnice o velikosti $n \times n$. Dále jsou zadány souřadnice políčka, na kterém stojí šachová figurka kůň, a souřadnice políčka, kam chceme koně pomocí dovolených tahů přesunout. Napište v Pascalu program, který nalezne nejkratší posloupnost tahů, kterými lze koně přesunou ze startovního políčka do cílového. Připomenutí dovolených tahů (modré kolečko značí výchozí pozici, zelená kolečka označují dovolené tahy):



Řešení. Použijeme algoritmus vlny (odsimuluj a dívej se na sachovnici - po šachovnici se šíří vlna). Jinak se též tento algoritmus nazývá prohledávání do šířky. Algoritmus vlny je řízen pomocí datové struktury fronta. Fronta je struktura, která umožňuje vložení a odebrání prvku. Množina uložených prvků je reprezentována jako posloupnost, nový prvek se vkládá vždy na konec této posloupnosti, odebírání prvku se provádí na začátku. Fronta lze snadno implementovat v poli, tak že používáme „klouzavý“ konec a začátek.

Řízení pomocí fronty vypadá tak, že algoritmus postupně prozkoumává možné pozice a kdykoli narazí na novou možnou pozici, vloží si tuto do fronty (na konec). Ve frontě jsou vždy pozice jejichž okolí (vzhledem k figurce kůň) je třeba prozkoumat. V každém kroku tedy algoritmus vybere jeden záznam (pozici) z počátku fronty, provede průzkum okolí a případné nové pozice (nenavštívené) poskládá na konec fronty.

```
program Konik;

const MAX = 8; { velikost sachovnice }
const MAX_FRONTA = 100; { maximalni velikost pomocene fronty }

var sachovnice:array[1..MAX,1..MAX] of integer;
  { pole, kde si pamatujeme navstivena policka,
    0 - nenavstiveno, jinak pocet kroku, za jak dlouho
    navstiveno }

type POZICE = record
                x, y: integer;
              end;

type TAH = record
             poz: POZICE;
             pred: integer; { predchudce
                              tohoto tahu }
             delka: integer;
           end;

var fronta: array[1..MAX_FRONTA] of TAH; { pomocna fronta }
var zacatek, konec: integer; { zacatek a konec fronty }
var tahy: array[1..8] of POZICE; { mozne tahy konem }

procedure inicializuj;
var i, j: integer;
begin
  for i:=1 to MAX do begin
    for j:=1 to MAX do begin
      sachovnice[i][j] := 0;
    end;
  end;
  zacatek := 1;
  konec := 1;

  tahy[1].x :=  1;  tahy[1].y :=  2;
  tahy[2].x :=  2;  tahy[2].y :=  1;
  tahy[3].x :=  2;  tahy[3].y := -1;
  tahy[4].x :=  1;  tahy[4].y := -2;
```



```
  tahy[5].x := -1;  tahy[5].y := -2;
  tahy[6].x := -2;  tahy[6].y := -1;
  tahy[7].x := -2;  tahy[7].y :=  1;
  tahy[8].x := -1;  tahy[8].y :=  2;
end;

procedure vloz(var novy: TAH);
begin
  fronta[konec] := novy;
  konec := konec + 1;
end;

procedure odeber;
begin
  zacatek := zacatek + 1;
end;

procedure vypis;
var i: integer;
begin
  i := konec - 1;
  while i > 0 do begin
    writeln('x:', fronta[i].poz.x, ' y:', fronta[i].poz.y);
    i := fronta[i].pred;
  end;
end;

function kroky(var cil: POZICE): boolean;
var i: integer;
var nova: POZICE;
var dalsi: TAH;
var pokracuj: boolean;
begin
  pokracuj := true;
  while pokracuj do begin
    if zacatek >= MAX_FRONTA then begin
      kroky := false;
      pokracuj := false;
      break;
    end
    else begin
      for i:=1 to 8 do begin
        nova.x := fronta[zacatek].poz.x + tahy[i].x;
        nova.y := fronta[zacatek].poz.y + tahy[i].y;
        if (nova.x >= 1) and (nova.x <= 8) and
           (nova.y >= 1) and (nova.y <= 8) then begin
          if sachovnice[nova.x][nova.y] = 0 then begin
            sachovnice[nova.x][nova.y] :=
              fronta[zacatek].delka + 1;
            dalsi.poz := nova;
            dalsi.pred := zacatek;
            dalsi.delka := fronta[zacatek].delka + 1;
            vloz(dalsi);
            if (nova.x = cil.x) and (nova.y = cil.y) then begin
              kroky := true;
              pokracuj := false;
              break;
            end;
```

```
          end;
        end;
      end;
      odeber;
    end
  end;
end;

procedure hledej(var start: POZICE; var cil: POZICE);
var prvni: TAH;
begin
  prvni.poz := start;
  prvni.pred := -1;
  prvni.delka := 0;
  vloz(prvni);
  if kroky(cil) then begin
    writeln('Uspech.');
    vypis;
  end
  else begin
    writeln('Neuspech.');
  end;
end;

var start, cil: POZICE;

begin
  inicializuj;
  writeln('Zadej pocatecni pozici:');
  write('x:');
  read(start.x);
  write('y:');
  read(start.y);
  writeln('Zadej cilovou pozici:');
  write('x:');
  read(cil.x);
  write('y:');
  read(cil.y);
  hledej(start, cil);
end.
```

Jak je to s asymptotickou časovou a prostorovou složitostí (prostorová složitost odpovídá počtu použitých paměťových míst, zjednodušeně počtu symbolů, které je třeba si zapsat, kdybychom úlohu simulovali na papíře) tohoto algoritmu ? Rozmyslete (ukažte), že algoritmus najde nejkratší možnou cestu mezi zadanými pozicemi. Popřípadě jednu z nejkratších cest, pokud existuje více nejkratších cest.