Cvičení Programování I

Cvičící: **Pavel Surynek, KTIML**
surynek@ktiml.mff.cuni.cz
http://ktiml.mff.cuni.cz/~surynek
Semestr: **Zima 2007/2008**
Kroužek: **Matematika/53**
Rozvrh: **Středa 12:20-13:50 (učebna K7)**

Stručné poznámky ke cvičení z 7.11.2007

0. Organizační záležitosti. Za domácí úkol byly úlohy **jak dlouho může běžet program**, **pigeonhole principle**, dále byly programovací úlohy (to, co už jsme vyřešili slovně) - **sedlový bod**, **největší společný dělitel** a **hledání prvku v setříděné posloupnosti**.

1. NSD pascalovsky. Naprogramujte v Pascalu program na výpočet největšího společného dělitele dvou čísel.

2. Sedlový bod pascalovsky. Je dána čtvercová matice nad celými čísly. *Napište program v Pascalu*, který co nejrychleji v této matici nalézt sedlový bod. Sedlový bod je místo v matici, které obsahuje největší hodnotu ve svém řádku a nejmenší hodnotu ve svém sloupci nebo nejmenší hodnotu ve svém řádku a největší hodnotu ve svém sloupci. Snažte se minimalizovat počet kroků, přičemž za krok považujeme každé podívání se na políčko matice. Nápověda: předvýpočet.

3. Vyhledávání s setříděné posloupnosti pascalovsky. Je dána setříděná posloupnost N přirozených čísel (například: 5, 20, 27, 35, 60, 66, 91). *Napište program v Pascalu*, který pro dané přirozené číslo x (například: 26) a zmiňovanou posloupnost a rozhodne, zda se zadané přirozené číslo nachází v zadané posloupnosti (v uvedeném příkladě bude odpověď ne). Algoritmus může provádět operace porovnání a v každém kroku se „může podívat“ na jedno číslo na libovolné pozici v posloupnosti. Snažte se minimalizovat počet kroků algoritmu (porovnání, podívání se, ... se považuje za krok).

> **Řešení.** Program bude zadané číslo hledat pomocí půlení intervalu. V každém kroku algoritmus vyhledává zadané číslo mezi d-tým a h-tým prvkem vstupní setříděné posloupnosti (na začátku je $d = 1$ a $h = N$). Interval $[d, h]$ posloupnosti rozdělí na dvě poloviny a podle hodnoty prostředního prvku tohoto intervalu rozhodne, ve které polovině intervalu se hledané číslo nachází. Potom algoritmus stejným způsobem pokračuje v hledání, tentokrát ale na intervalu poloviční velikosti. Toto půlení intervalu algoritmus provádí tak dlouho, dokud se velikost intervalu nezmenší na 1 (0), pak už lze snadno rozhodnout.

```
program binarni_vyhledavani;

const N = 7; { delka vstupni posloupnosti }
const posl: array[1..10] of integer = (5,20,27,35,60,66,91);
   { vstupni setridena posloupnost }
var x: integer; { hledane cislo }
var s: integer; { prostredni index intervalu }

procedure vyhledej(d,h,x: integer);
   { d - dolni mez, h - horni mez, x - hledani cislo }
begin
   if h-d = 0 then begin
```

```
     if posl[h] = x then begin
        writeln('Nalezeno.');
     end
     else begin
        writeln('Nenalezeno.');
     end;
    end
   else begin
     if h-d>0 then begin
        s := (h+d) div 2;
        if x >= posl[s] then begin
           vyhledej(s, h, x);
        end
        else begin
           vyhledej(d, s-1, x);
        end;
     end
     else begin
        writeln('Nenalezeno.');
     end;
   end;
end;

begin
   read(x);
   vyhledej(1,N,x);
end.
```