**Korešpondenčný seminár z programovania**
**XXI. ročník, 2003/04**
**Katedra vyučovania informatiky FMFI UK,**
**Mlynská Dolina, 842 48 Bratislava**

*KSP finančne podporujú:* aSc – Applied Software Consultants spol. s r.o.
*MICROSTEP-MIS spol. s r.o.*

# Vzorové riešenia 2. kola letnej časti, kat. Z

Milí riešitelia!
Ďalší ročník KSP je úspešne za nami (a vami). Všetkým vám blahoželáme k dosiahnutým výsledkom a dúfame, že ste sa čo-to naučili a bavilo vás to. S najlepšími z vás sa na jeseň stretneme na týždňovom sústredení, pozvánky očakávajte tak koncom septembra. Ale ako to už chodí, ešte pred tým vás čakajú zaslúžené prázdniny plné slnka, vody a skvelých kamarátov. Poriadne si ich užite! Keď sa oddýchnutí vrátite späť do škôl, vráti sa aj váš obľúbený seminár.

War doesn't show who's right, just who's left.

KSPáci

## 1. Zo života informačnej kancelárie II

opravovalo YoYo $\delta$
(max. 15 bodov)

Kocúrkovský systém kancelárií sa vám očividne zapáčil a niektoré vaše riešenia boli naozaj kocúrkovské. Poďme ale pekne po poriadku. Hlavná idea bola skoro vo všetkých vašich riešeniach rovnaká: najprv spočítať prvých $K$ prvočísel a potom simulovať príkazy s použitím jedného spoločného „frontu“[1].

Ak prišiel nejaký človek, tak ste vyskúšali všetky okienka, či nemôže k niektorému voľnému ísť, ináč ste ho zaradili do frontu. Ak sa nejaké okienko uvoľnilo, prešli ste vždy od začiatku frontu a hľadali ste niekoho, kto k okienku môže ísť.

A akú mal takýto prístup časovú zložitosť? Na začiatku bolo treba vypočítať prvých $K$ prvočísel. To ste všetci zvládli prinajhoršom v kvadratickom čase od $K$-teho prvočísla (a s tým sa uspokojí aj vzorové riešenie). Príkaz `PRISIEL` potreboval očividne $O(K)$ operácií. Pre $N$ ľudí to teda bolo $O(KN)$. S príkazom `UVOLNILO SA` to už bolo horšie. Zvyčajne ste prehľadávali celý front, teda $O(N)$ a ak by mala dôjsť rad na každého na koho môže[2] tak to máme $O(N^2)$.

Vzorové riešenie sa veľmi líšiť nebude od tohto postupu. Akurát trošičku vylepšíme uvoľnenie okienka. A ako? No vcelku jednoducho. Ak sme už raz prezreli časť frontu, načo ju kontrolovať znova?

Začnime teda vygenerovaním prvočísel. Najjednoduchšie a najvýhodnejšie je začať ich generovať pekne zaradom a to tak, že prechádzame jednotlivé čísla, pre každé zistíme či je to prvočíslo a ak áno, tak si ho zapamätáme. A ako overíme, či je nejaké číslo prvočíslo? Jednoducho ho zaradom skúsime deliť menšími číslami a ak nájdeme nejakého deliteľa, tak to prvočíslo nebude. Dôležité je ale uvedomiť si, že pre číslo $x$ nám stačia iba čísla menšie ako $\sqrt{x}+1$. No a ešte drobné urýchlenie získame tým, že ho budeme skúšať deliť iba prvočíslami. Takýto postup má časovú zložitosť $O(X \cdot \sqrt{x})$ kde $X$ je $K$-te prvočíslo.

Budeme si ale trošku nezvyčajne pamätať ľudí, ktorí čakajú. Namiesto zapisovania ich čísel do poľa si budeme pre každého z nich pamätať, či čaká alebo nie. Ak niekto príde,

[1] tieto fronty mali všelijaké podoby, aj keď vo všetkých prípadoch by ste najlepšie pochodili s obyčajným poľom

[2] všimli ste si snáď že exsistujú nešťastníci na ktorých nikdy rad nedojde. Napríklad už len taký chudák prvý...

zistíme si jeho poradové číslo a prezrieme okienka či nie je voľné niektoré z tých ktoré delia jeho poradové číslo. Ak je, vypíšeme správu, nastavíme, že okienko nie je voľné a poznačíme si, že pri tom okienku bol naposledy on. Ak nie je žiadne voľné, jednoducho si poznačíme pre jeho poradové číslo, že čaká v rade.

Keď sa uvoľní okienko, pozrieme sa kto pri ňom naposledy bol a začneme prechádzať ľudí ktorí prišli po ňom. Toto spravíme ešte trošičku prefíkanejšie, nemusíme ich prechádzať po jednom ale rovno po násobkoch prvočísla patriaceho tomuto okienku – t.j. stále pripočítavame prvočíslo až kým, alebo neprejdeme za posledného človeka, alebo nenájdeme čakajúceho.

Časová zložitosť takéhoto algoritmu je $O(KN)$ bez predvýpočtu prvočísel[3]. Všetky operácie `PRISIEL` sú ako sme už povedali $O(KN)$. Pre každé okienko pri operáciách `UVOLNILO SA` prinajhoršom raz prejdeme všetkých ľudi (dokonca iba násobky nejakého prvočísla) čo pre $K$ okienok znamená $O(KN)$ operácií.

Pamäťová zložitosť vzorového riešenia je lineárna vzhľadom na počet ľudí, ktorý do kancelárie prídu. To sa ešte dá zlepšiť na lineárnu vzhľadom na maximálny počet ľudí, ktorí budú v čakárni naraz.

A ako boli vaše riešenia bodované? Riešenia ekvivalentné so vzorovým by dostali 15 bodov. Riešenia s časom $O(N^2)$ dostávali 13 a menej bodov. No a horšie dostali ešte menej.

## Listing programu:

```
Program Kocurkovo;

const maxK=500;
      maxN=500;

var pr:array [1..maxK] of integer;                          {prvocisla}
    pr_inv:array [1..maxK] of integer;          {prevedenie prvocisla na okienko}
    okno:array[1..maxK] of integer;          {kto bol naposledy pri danom okienku}
    volne:array[1..maxK] of boolean;                        {ci je okienko volne}
    clovek:array[0..maxN] of boolean;                {ci i-ty clovek caka v rade}
    max_clovek: integer;                        {najposlednejsi clovek co prisiel}
    K:integer;
    i:integer;
    prikaz:string[20];

procedure prvocisla( k:integer );                  {vypocita prvych k prvocisel}
var i,j,a:integer;
begin
    i:=3;  j:=2;
    pr[1]:=2;
    pr_inv[2]:=1;
    repeat
        a:=1;  {vyskusame delit po odmocninu}
        while (pr[a]*pr[a]<i) and (a<j) and ( i mod pr[a]<>0) do inc(a);
        if (a>=j) or (pr[a]*pr[a]>i) then begin  {mame prvocislo}
            pr[j] := i;
            pr_inv[i] := j;
            inc(j);
        end;
        inc(i);
    until j>k
end;

procedure k_oknu(i,j:integer);
```

[3] a výpočet prvočísel sa pod to schová tiež, ak $N$ je vačšie ako $K$-te prvočíslo

```
begin writeln('Obcan s poradovym cislom ',i,' prosim k okienku ',j); end;

procedure prisiel;
var i:integer;
begin
   inc(max_clovek);   {jeho cislo}
   writeln( 'Prisiel obcan cislo ',max_clovek,'.' );
   i:=1; {skusime najst volne okienko, ktoreho prvocislo deli max_clovek}
   while (i<=K) and ((max_clovek mod pr[i] <>0) or (not volne[i]))  do inc(i);
   if i<=K then begin            {mame volne}
      k_oknu(max_clovek,pr[i]);
      okno[i]:=max_clovek;
      volne[i]:=false;
   end else clovek[max_clovek]:=true;   {caka v rade}
end;

procedure uvolnilo;
var i:integer;
begin
   readln(i);
   i:=pr_inv[i];
   if volne[i] then writeln('vsak je uz volne...'); {hmmm, uvolnilo sa volne okienko?}
{prechadzame nasobky prvocisla zacinajuc za clovekom, ktory tu bol naposledy}
   while (okno[i]<=max_clovek) and (not clovek[okno[i]]) do okno[i] :=
okno[i]+pr[i];
   if (okno[i]<=max_clovek) then begin {nasli sme cakajuceho}
      k_oknu( okno[i], pr[i] );
   clovek[okno[i]]:=false;    {dotycny uz necaka}
      volne[i]:=false; {zbytocne, ale pre istotu}
   end else begin  {inac je okienko volne}
      okno[i]:=0;
      volne[i]:=true;
   end;
end;

begin
   readln(K);
   prvocisla(K);
   for i:=1 to K do writeln(pr[i]);
   clovek[0]:=false;
   max_clovek:=0;
   { pri kazdom okne je 0-ty clovek (nikto)}
   for i:=1 to K do begin okno[i] := 0; volne[i] := true; end;

   while 0=0 do begin
      readln(prikaz);
      if prikaz='PRISIEL' then prisiel
      else if prikaz='UVOLNILO SA' then uvolnilo
      else writeln('Zly prikaz!');
   end;
end.
```

## 2. Zaba neblázni

opravovala Ňaňka
(max. 15 bodov)

Opäť raz som opravovala najobľúbenejší príklad série. Jeho zadanie malo v sebe drobnú nepresnosť. Úlohou bolo: „Napísať program, ktorý zistí, či ešte niekedy budú žabky sedieť

tak, ako sedeli na začiatku. Ak áno, zistiť, koľkokrát musí Zaba tlesknúť, aby sa tak stalo.“ Ako neskôr ukážem, pokiaľ je zadanie korektné, vždy sa vie každá žabka dostať na svoje miesto. Toto jej nemôže trvať dlhšie ako $N$ skokov (preskáče postupne všetky lavičky). Ak pohyb naspäť na miesto trvá žabke $s$ skokov, bude na svojom mieste aj po $2.s$,$3.s$ ... skokoch. Stačí teda nájsť číslo, ktoré je násobkom ľubovoľného čísla medzi 1 a $N$. A to je $N!$. Nikto však túto chybičku do dôsledkov nevyužil.

Upravené zadanie teda znie: „Napíšte program, ktorý zistí, či ešte niekedy budú žabky sedieť tak, ako sedeli na začiatku. Ak áno, zistite, koľkokrát najmenej musí Zaba tlesknúť, aby sa tak stalo.“

V prvom rade ukážeme, že každá žaba sa dostane na svoje miesto. Je zrejmé, že keď žabka skáče, po určitom počte skokov sa dostane na miesto, kde už bola. Tvrdím, že prvé také miesto je kameň na ktorom pôvodne sedela. Ak by to tak nebolo, jej cesta by viedla po kameňoch $k_1, \ldots k_m$ a $k_{m+1} = k_i$ by bolo prvé miesto, kde sa kamene opakujú. Potom by buď bola v zadaní chyba, lebo na kameňoch $k_m$ a $k_{i-1}$ by museli byť rovnaké čísla alebo by $k_m$ a $k_{i-1}$ museli byť ten istý kameň (spor s predpokladom, že $k_{m+1} = k_i$ je prvé miesto, kde sa kamene opakujú). Na svoje miesto sa tiež dostane na ľubovoľný násobok tohto počtu skokov.

Teraz sa pozrieme na jednotlivé žabky. Na svojej ceste žabka *Ž* skáče vždy dokola po niekoľkých kameňoch. Žabky, ktoré na týchto kameňoch pôvodne sedeli sa na svoje miesta dostanú po rovnakom počte skokov ako naša žabka *Ž*. Nazvime si postupnosť týchto žabiek cyklom a dĺžkou cyklu nazvime počet žabiek (vrátane žabky *Ž*). Dĺžka cyklu zároveň hovorí, koľko skokov žabky v cykle potrebujú, aby sa dostali na svoje miesta. Na svoje miesta sa dostanú aj na počet skokov, ktoré získame, ak prenásobíme dĺžku cyklu ľubovoľným celým kladným číslom.

Pôvodné žabky sa nám teda rozložili na niekoľko cyklov. Ak sú dĺžky jednotlivých cyklov $c_1, c_2 \ldots c_k$ a potrebujeme nájsť číslo, ktoré je násobkom každého z týchto čísel a zároveň je najmenším takým číslom. Takémuto číslu sa hovorí najmenší spoločný násobok (nsn). Pre dve čísla $a, b$ je $nsn(a, b) = (a.b)/NSD(a, b)$. Pre viac čísel (v našom prípade $c_1, c_2 \ldots c_k$ počítame rekurzívne ako $nsn(c_1 \ldots c_k) = nsn(nsn(c_1, \ldots c_{k-1}), c_k)$ pre $k > 2$. Špeciálne (triviálne) prípady sú $nsn(c_1, c_2) = (c_1.c_2)/NSD(c_1, c_2)$, $nsn(c_1) = c_1$ a $nsn() = 1$.

Teraz sa pozrime bližšie na vzorové riešenie. Bude postupne prechádzať žabky. V prípade, že žabka ešte nie je v žiadnom cykle, je žabkou Ž a môže začať skákať. Postupne označuje žabky, ktoré sú s ňou v cykle a počíta dĺžku cyklu, až kým nepríde na svoje pôvodné miesto, resp. na nejaké miesto, kde už bola (vieme, že pri korektnom zadaní je to práve jej miesto). Priebežne si uchovávame $nsn$ dĺžok už prejdených cyklov. Ak teda máme $nsn(c_1 \ldots c_{k-1})$ a $c_k$, vieme vypočítať $nsn(c_1 \ldots c_k)$. Všimnite si, že každú žabku spracovávame iba jeden krát, teda časová zložitosť bude *O(N.(čas na spracovanie žabky))*

Ostáva ešte povedať, ako počítame najväčšieho spoločného deliteľa dvoch čísel $NSD$. Používame na to Euklidov algoritmus. Základná vlastnosť $NSD$, ktorú používa je, že $NSD(a, b) = NSD(b \ mod \ a, a)$, ak $0 < a \leq b$ a $NSD(0, b) = b$. Tento algoritmus pracuje v logaritmickom čase. Stručne zdôvodním prečo. Majme teda čísla $a, b$ také, že $a < b$. Po prvom kroku budeme mať čísla $c, a$ (vidno, že platí $c < a$). Platí, že $c \leq b/2$ (ak by totiž $a < b/2$, potom aj $c < a < b/2$, inak sa $c = b - a < b/2$) a preto po dvoch krokoch sa oba parametre zmenšia na polovicu. Odtiaľ už logaritmický čas intuitívne vidno.

Modifikovaný Euklidov algoritmus ($NSD(a, b) = NSD(b - a, a)$ pre $0 < a \leq b$), ktorý ste niektorí používali má čas lineárny, čo vidno napríklad pri rátaní $NSD(1, N)$.

Keď si to zhrnieme, časová zložitosť vzorového riešenia *O(N.(čas na spracovanie žabky))* $= O(N. \log N)$ a pamäťová zložitosť je *O(N)* (používame jedno pole veľkosti $N$).

Poďme teraz k vašim riešeniam. Dali by sa rozdeliť do týchto kategórií.

- simulácie skákania žabiek – maximálne 8 bodov
- hľadanie cyklov pre každú žabku – maximálne 12 bodov

- vzorové riešenie s rozkladom na prvočinitele – maximálne 12 bodov
- vzorové riešenie s použitím modifikovaného Euklidovho algoritmu – maximálne 12 bodov
- vzorové riešenie – maximálne 15 bodov

Nejaké bodíky sa dali stratiť napríklad za slabý alebo žiadny popis a tradične sa body delili aj pri kolektívnych riešeniach.

A nakoniec to najlepšie – vzorové riešenie.

### Listing programu:

```
const Nmax=100;                  {maximalny pocet zabiek}
var a:array[1..Nmax] of integer; {urcuje, kam zabky maju skakat}
   N,i,j,nsn,cyklus,pom:integer;

function NSD(a,b:integer):integer;
 {najvacsi spolocny delitel cisel a,b}
begin
 while (a*b<>0) do
  if a>b then a:=a mod b
        else b:=b mod a;
 NSD:=a+b;
end;

begin
 {nacitanie a inicializacia}
 readln(N);
 for i:=1 to N do readln(a[i]);
 nsn:=1;

 {spracovanie cyklov premutacie}
 for i:=1 to N do
  if a[i]>0 then begin      {pokial zabka doteray nebola v ziadnom cykle}
   cyklus:=0; j:=i;
   repeat                  {prejdeme cez tento cyklus}
    pom:=a[j]; a[j]:=0; j:=pom;
    inc(cyklus);
   until a[j]=0;           {az pokial nenarazime na zabku, ktoru sme uz videli}
   nsn:=(nsn*cyklus) div NSD(nsn,cyklus); {a vyratame nsn(a,b)=(a*b)/NSD(a,b)}
  end;

 {vysledok je nsn dlzok cyklov permutacie}
 writeln('Pocet tlesknuti Zaby je ',nsn);

end.
```

## 3. Zo sveta Motovodu

opravovala Anička
(max. 15 bodov)

Väčšina z vás zisťovala, či treba kúpiť mapu s pomerne veľkou časovou zložitosťou: napr. v čase až $O(M^2)$ (max. 8 bodov), v čase $O(MN)$ alebo $O(N^2)$ (max. 11 bodov). No našli sa aj takí, čo úlohu vyriešili v čase $O(M+N)$, teda v optimálnom (15 bodov). Za popis a odhady zložitosti ste mohli získať/stratiť max. 5 bodov.

Tak sa teda na to pozrime. Idea vám bola jasná. Ak sú nejaké dve zastávky spojené cestou, nemôžu ležať na jednom brehu. Pokúsime sa teda porozhadzovať zastávky ľavý a pravý breh. Ako rozumný prístup vyzerá toto: zoberme si prvú zastávku prvej načítanej

dvojice. Bez problémov môžme usúdiť, že leží na ľavom brehu. Ak teda nájdeme zastávku, s ktorou je prepojená nejakou cestou, táto bude ležať na pravom brehu. A naopak, zastávka prepojená s touto zastávkou na pravom brehu musí zas ležať naľavo. Problém nastane, ak nájdeme cestu medzi dvomi zastávkami, ktoré už obe majú svoj breh určený a je to rovnaký breh. V takom prípade môžeme rovno vypísať, že mapu kupovať netreba a skončiť. Ak sa takýto problém počas celého behu programu nevyskytne, mapu kúpiť treba, lebo je možné, že toto mesto je vhodné pre motovod.

Tento postup ma aj pekný názov: prehľadávanie do hĺbky (depth first search, DFS). Predstavme si, že zastávky sú vrcholy grafu a spoje medzi zastávkami sú jeho hranami. Zoberieme si jeden vrchol, napr. $v_1$, priradíme ho na breh. Nájdeme vrchol, s ktorym je spojený, nazvime ho $v_2$ a jemu priradíme opačný breh. Potom to isté spustíme pre vrchol $v_2$. Nájdeme vrchol, s ktorým je spojený a priradíme mu opačný breh, než na akom je ... atď. Samozrejme, skôr než vrcholu priradíme nejaký breh, skontrolujeme, či už nemá priradený opačný breh, ako mu chceme dať.

Takýto algoritmus sa však dá napísať so zložitosťou $O(N^2)$, ale aj $O(M+N)$. Kde je pes zakopaný? Dôležité je, ako si uložíme údaje o cestách medzi zastávkami. Ak použijeme maticu $N \times N$, v ktorej si zapamätáme, či medzi danými dvomi mestami je alebo nie je cesta, musíme ju pri hľadaní ciest celú prechádzať. Výsledkom je kvadratická zložitosť. Môžme to však spraviť inak. Budeme mať n riadkov, pre každý vrchol jeden. Načítame dvojicu zastávok $i, j$. Následne zapíšeme j na pozíciu $[i, 1]$ ($[i, n]$). Okrem toho si zapamätáme (v listingu programu v poli 'index'), že najbližšie budeme v i-tom riadku zapisovať na druhú poziciu ($[i, n+1]$). Okrem toho je fajn si pamätať, koľko hodnôt máme zapísaných v danom riadku (viď. pole 'pocetC'- počet ciest). V samotnom prehľadávaní sa teda nikdy nepozeráme dvakrát na tú istú cestu. Po skončení načítavania nastavíme 'ukazovátko' (index) opäť na prvú pozíciu v riadku. Ak teda hľadáme cestu z vrcholu v1, pozrieme sa v riadku v1 na pozíciu, na ktorú ukazuje hodnota v poli 'index' a následne túto hodnotu zvýšime. Nabudúce sa teda pozrieme až na nasledujúcu pozíciu. Ak si pamätáme počet údajov v danom riadku, tak si ľahko okontrolujeme, či má zmysel z daného vrcholu ešte hľadať cestu. Výsledkom je čas $O(M+N)$.

Na záver už len: nie je rozumné predpokladať, že graf, vytvorený zastávkami a cestami medzi nimi, sa nemôže skladať z viacerých cyklov, ak to v zadaní nie je napísané. Tí z vás, ktorí rátali len s takýmito prípadmi, nedostali veľa bodov. Ak teda prehľadávanie do hĺbky skončí, ale ešte sme neprezreli $m$ ciest, musíme začať odznova s nejakým vrcholom, ktorý ešte nie je zaradený na žiadny breh.

## Listing programu:

```
var m,n,zac,prezrete,i: integer;
    cesty: array[1..10,1..10] of integer;
    pocetC,index,brehy: array[1..100] of integer;
    mapa: boolean;

Procedure nacitanie;
var f:text;
    i,a,b: integer;
begin
  assign(f,'motovod.in');reset(f);   {nacitanie zo suboru}
  read(f,n); readln(f,m);
  for i:= 1 to n do index[i]:= 0;
  for i:= 1 to n do pocetC[i]:= 0;
  read(f,zac);
  inc(pocetC[zac]);                  {vrchol, z ktoreho prvykrat spustim}
  inc(index[zac]);                   {prehladavanie}
  readln(f,cesty[zac,index[zac]]);
  for i:= 2 to m do begin
```

```
    read(f,a);
    inc(pocetC[a]);
    inc(index[a]);
    readln(f,cesty[a,index[a]]);
  end;
  close(f);
end;

Procedure DFS(breh, zast: integer);
var i:integer;
begin
  inc(prezrete);
  if breh= brehy[cesty[zast,index[zast]]] then begin
  mapa:=false; exit; {test, ci dva vrcholy nelezia na jednom brehu}
  end
  else if brehy[cesty[zast,index[zast]]]= 0 then {0: este nezaradeny vrchol}
    case breh of
    1: brehy[cesty[zast,index[zast]]]:= 2;   {1: lavy breh, 2: pravy breh}
    2: brehy[cesty[zast,index[zast]]]:= 1;
    end;
  for i:= 1 to pocetC[zast] do begin  {rekurzivne spustime proceduru}
    DFS(brehy[cesty[zast,i]],cesty[zast,i]); {pre dalsi vrchol}
    inc(index[i]);
  end;
end;

Procedure vypis;
var f:text;
begin
  assign(f,'motovod.out');rewrite(f);  {vypis do suboru}
  if mapa then write(f,'Ano')
  else write(f,'Nie');
  close(f);
end;

Begin
  nacitanie;
  mapa:= true;
  for i:= 1 to n do if index[i]>0 then index[i]:= 1;
  brehy[zac]:=1; {tento vrchol mozme dat na lubovolny breh}
  prezrete:=0;
  repeat
    DFS(brehy[zac],zac);
    if prezrete < m then begin {ak graf nie je spojity}
      i:=1;
      while index[i] = pocetC[i] do inc(i);
      zac:=i;
    end;
  until (prezrete = m) or (not mapa);
  vypis;
End.
```

opravoval Goober
(max. 10 bodov)

## 4. Zeofína v záhrade II

Tento príklad bol vskutku príjemný na opravovanie, lebo riešenia, ktoré prišli, boli zväčša správne a navyše mali aj dobrú časovú i pamäťovú zložitosť (ak vôbec má zmysel o zložitos-

tiach v takomto príklade hovoriť). Bodovanie bolo preto veľmi jednoduché – 10 bodov bolo za správne riešenie s dostatočným popisom. No a keď niečo chýbalo (napríklad poriadny popis, program, ...), tak sa počet bodov patrične znížil.

**Malé ospravedlnenie** Keď sme sa Zeofíny pýtali, akýže vzor jej to vznikol v záhradke, poslala nám svoju odpoveď poštou. Lenže tam zaúradoval akýsi lapikurkár[4] a zmazal určité riadky vo výstupe.

**Ako sa to dalo riešiť?** Zeofínin obrázok sa skladá z niekoľkých vodorovných a niekoľkých zvislých čiar. Aby sme na žiadnu čiaru nezabudli (a žiadnu nekreslili dvakrát), nakreslíme najprv všetky vodorovné a potom všetky zvislé, aj keď to možno bude pre Zeofínu znamenať viac chodenia. Pozrime sa najprv na vodorovné čiary. Aby sa nám o nich ľahšie rozprávalo, očíslujeme ich zdola číslami $0, 1, 2, \ldots, n$ (ak má záhrada $n$ korytnačích metrov na výšku, vyskytuje sa v nej $(n+1)$ vodorovných čiar). Je ľahké vidieť, že čiary s číslami $0, 1, 3, 4, 6, 7, \ldots$ sú prerušované, zatiaľčo čiary $2, 5, 8, \ldots$ idú cez celú šírku záhradky. Inak povedané, čiara je plná vtedy (a len vtedy), keď jej číslo dáva zvyšok 2 po delení tromi. Keďže plné čiary je ľahé nakresliť, pozrieme sa na tie prerušované. Každá taká čiara sa skladá z $m$ úsekov, z ktorých niektoré sú plné a niektoré prázdne. Podobne ako predtým, aj tu sa ukáže jednoduchý vzor – úseky $1, 4, 7, \ldots$ (jednotlivé úseky číslujeme $1 - m$) sú prázdne a úseky $2, 3, 5, 6, 8, \ldots$ sú plné. Teda prázdne úseky sú práve tie, ktorých čísla dávajú po delení tromi zvyšok 1.

Takže vodorovné čiary máme už zmapované detailne. No a keď sa pozrieme na zvislé čiary, s prekvapením zistíme, že vyzerajú presne tak, ako tie vodorovné, akurát sú inak otočené. Takže ich budeme kresliť takmer úplne rovnakým postupom (až na to otočenie).

**No a implementácia?** Jednotlivé čiary budeme kresliť podobne, ako keby Zeofína bola písací stroj[5]. Zeofína vždy nakreslí danú vodorovnú čiaru (plnú/prerušovanú) a po jej nakreslení sa vráti na ľavý okraj záhradky. Potom sa posunie na začiatok ďalšej (čiže sa posunie o riadok vyššie) a opakuje postup. Keď nakreslí všetky vodorovné čiary, otočí sa a podobným postupom nakreslí všetky zvislé čiary. Aby nevznikli nejasnosti, na začiatku je Zeofína v ľavom dolnom rohu a pozerá sa smerom doprava.

**Zložitosť?** Nuž, s odhadom zložitosti je to v tomto prípade jednoduché – naše inštrukcie pre Zeofínu zjavne nejdú po žiadnom úseku viac ako trikrát (dokonca po väčšine čiar prejdú len dvakrát) a celkový počet úsekov je $(m \times n)$. Na každý úsek použijeme nanajvýš dva príkazy a ostatných príkazov (posúvanie sa o riadok, otáčanie sa) je zanedbateľne málo. Teda celkový počet príkazov je $O(mn)$ a keďže na určenie každého z nich potrebujeme len konštantné množstvo času, časová zložitosť je $O(mn)$. Túto časovú zložitosť nemožno asymptoticky zlepšiť, lebo v obrázku sa vyskytuje veľmi veľa (rádovo $m \times n$) úsekov, pričom na každý z nich je potrebné použiť aspoň jeden príkaz.

No a pomocnú pamäť (s výnimkou niekoľkých celočíselných premenných) nepoužívame, a teda pamäťová náročnosť je $O(1)$.

## Listing programu:

```
procedure dole; begin writeln('DOLE'); end;
procedure hore; begin writeln('HORE'); end;
procedure vpred(l:integer); begin writeln('DOPREDU ', l); end;
procedure vlavo(a:integer); begin writeln('DOPRAVA ', a); end;
procedure vpravo(a:integer); begin writeln('DOLAVA ', a); end;

{Nakresli plnu ciaru, na konci je Zeofina opacne ako bola}
procedure plna(dlzka:integer);
begin dole; vpred(dlzka); hore; vlavo(180); vpred(dlzka); end;
```

[4] -a m expr. naničhodník, pobehaj, lapaj [Krátky slovník slovenského jazyka, Veda 1989]

[5] :-)

```
{Nakresli prerusovanu ciaru, na konci je Zeofina opacne ako bola}
procedure prerusovana(dlzka:integer);
var usek:integer;
begin
  for usek:=1 to dlzka do begin
    case usek mod 3 of 1: hore; 2: dole; end; {ak 0, netreba robit nic, lebo uz je
hore};
    vpred(1);
  end;
  hore; vlavo(180); vpred(dlzka);
end;

{Prejde na dalsi riadok}
procedure dalsiriadok;
begin hore; vpravo(90); vpred(1); vpravo(90); end;

{Prejde na dalsi stlpec}
procedure dalsistlpec;
begin hore; vlavo(90); vpred(1); vlavo(90); end;

var n, m:integer;
    riadok, stlpec:integer;

begin
  read(n, m);
  for riadok:=0 to n do begin
    if riadok mod 3 = 2 then plna(m) else prerusovana(m);
    if riadok<n then dalsiriadok; {Z posledneho riadku uz netreba ist}
  end;

  {Otocime sa a zideme do laveho dolneho rohu}
  hore; vlavo(90); vpred(n); vlavo(180);

  for stlpec:=0 to m do begin
    if stlpec mod 3 = 2 then plna(n) else prerusovana(n);
    if stlpec<m then dalsistlpec; {Z posledneho stlpca uz netreba ist}
  end;
end.
```

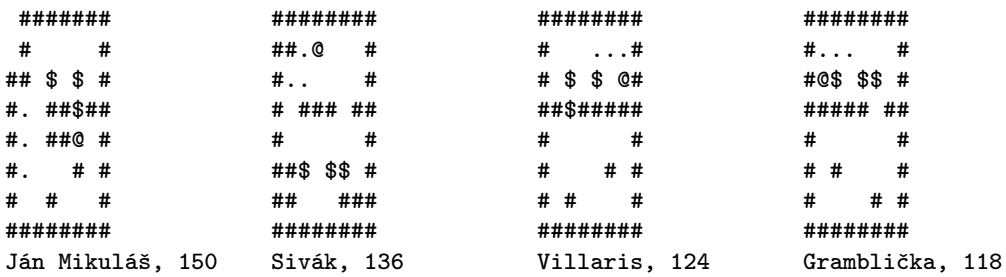
## 5. Z denníčka

opravovala Meri
(max. 15 bodov)

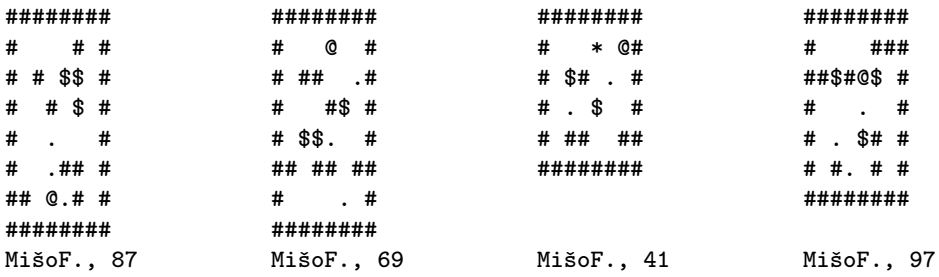
Veľmo ste nás potešili. Levelov ste poslali do bludu. A tak sme sa pohrali do sýtosti (alebo aspoň ja). Každému sme našli najkratšie riešenie (len tak cvične, ako by ste napísali automatizovaný riešič? To len keby ste sa náhodou cez prázdniny nudili...). A pridelili sme im body takto:

```
za 23 - 32 ťahové riešenie ste mohli dostať 8 bodov
za 34 - 44 ťahové riešenie ste mohli dostať 9 bodov
za 46 - 58 ťahové riešenie ste mohli dostať 10 bodov
za 59 - 72 ťahové riešenie ste mohli dostať 11 bodov
za 74 - 86 ťahové riešenie ste mohli dostať 12 bodov
za 91 - 106 ťahové riešenie ste mohli dostať 13 bodov
za 114 - 139 ťahové riešenie ste mohli dostať 14 bodov
za 150 - 164 ťahové riešenie ste mohli dostať 15 bodov
```

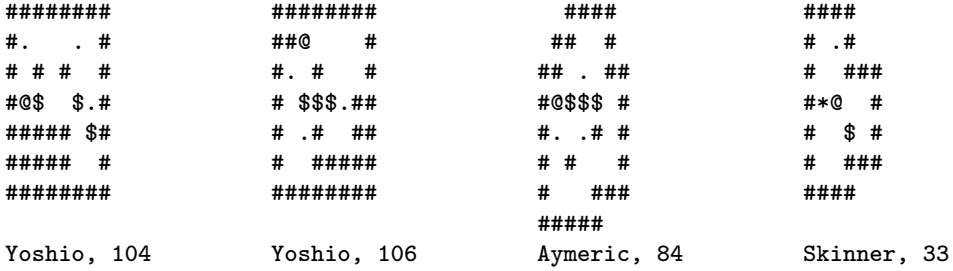
**Výstavka vašich najlepších levelov:**




**Výstavka našich zábavných levelov:**




**A ešte zopár iných**




Príjemnú zábavu pri ich riešení!

# Výsledková listina po 2. kole kategórie KSP-Z

| | Meno a priezvisko | Škola | Ročník | | 21 | 22 | 23 | 24 | 25 | Σ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Takáč Slavomír | Gymnázium | 2 | 61 | 11 | 12 | 15 | 10 | 11 | 120 |
| 2 | Mikuláš Ján | Gym. Haličská Lučenec | 2 | 52 | 2 | 15 | 15 | 10 | 15 | 109 |
| 3 | Herman Peter | Gym. Jura Hronca BA | 1 | 53 | 11 | 14 | 11 | | 14 | 103 |
| 4 | Mikuláš Ondrej | Gym. Haličská Lučenec | 1 | 34 | 13 | 5 | 10 | 5 | 14 | 81 |
| 5 | Ďuďák Juraj | Gym. Golianova Nitra | 2 | 39 | | 12 | 8 | 10 | 10 | 79 |
| 5 | Jerguš Ján | Gym. Alejová Košice | 1 | 24 | 13 | 11 | 9 | 10 | 12 | 79 |
| 7 | Fedák Matúš | Gym. Stará Ľubovňa | 2 | 25 | | 12 | 15 | 10 | 13 | 75 |
| 8 | Dzurenko Miroslav | Gym. Ľ. Štúra Zvolen | 3 | 30 | 11 | 12 | 9 | | 10 | 72 |
| 9 | Holub Michal | Gym. Jura Hronca BA | 3 | 22 | 12 | 12 | 11 | | 11 | 68 |
| 9 | Mindek Peter | Škola pre mim. nadané deti BA | 2 | 19 | 12 | 8 | 11 | 10 | 8 | 68 |
| 11 | Ambrošová Lucia | Gym. Grösslingová BA | 2 | 26 | | 12 | 11 | | 11 | 60 |
| 12 | Danko Juraj | Gym. P. de Coubertina Piešťany | 1 | 14 | 14 | 11 | 8 | | 11 | 58 |
| 12 | Paulis Peter | Gym. Cyrila a Metoda Nitra | 3 | 23 | 10 | 8 | 3 | 3 | 11 | 58 |
| 14 | Kajan Peter | Gym. Jura Hronca BA | 1 | 28 | | 10 | 9 | 9 | | 56 |
| 15 | Buštor Ivan | Gym. Košická BA | 1 | 0 | 11 | 15 | 15 | 3 | 11 | 55 |
| 15 | Rauová Ivana | Gym. Jura Hronca BA | 3 | 35 | | 10 | | | 10 | 55 |
| 17 | Valentíny Martin | Gym. Jura Hronca BA | 3 | 9 | 10 | 12 | 8 | | 14 | 53 |
| 18 | Pančík Andrej | Gym. Tajovského B. Bystrica | 1 | 20 | | 12 | 5 | | 12 | 49 |
| 19 | Šimko Jakub | Gym. Jura Hronca BA | 3 | 19 | 5 | 12 | | | 12 | 48 |
| 20 | Sivák Michal | Gym. Ľudovíta Štúra Trenčín | 2 | 11 | 12 | 8 | | | 15 | 46 |
| 21 | Šuranyi Martin | Gymnázium Levice | 3 | 12 | 12 | 8 | | | 12 | 44 |
| 22 | Villaris Vojtech | Gym. Jura Hronca BA | 3 | 0 | 12 | 12 | 5 | | 14 | 43 |
| 23 | Baumann Martin | Gym. Jura Hronca BA | 1 | 26 | 12 | 3 | | | | 41 |
| 24 | Mazák Tomáš | Gym. Jura Hronca BA | 1 | 17 | | 11 | | | 12 | 40 |
| 24 | Sábo Jozef | Gym. Školská Spiš. Nová Ves | 1 | 21 | | 13 | 4 | 2 | | 40 |
| 26 | Balko Peter | Gym. Jura Hronca BA | 3 | 0 | 11 | 12 | | | 13 | 36 |
| 27 | Žubrietovský Tomáš | Gym. Ľ. Štúra Zvolen | 3 | 7 | 10 | 5 | 4 | | 9 | 35 |
| 28 | Nemjo Martin | SPŠE Michalovce | 2 | 0 | 12 | 11 | | | 10 | 33 |
| 29 | Dobiaš Michal | Gym. Jura Hronca BA | 2 | 32 | | | | | | 32 |
| 30 | Bundala Daniel | Gym. Jura Hronca BA | 2 | 31 | | | | | | 31 |
| 30 | Strizenec Michal | Gym. Jura Hronca BA | 3 | 21 | | 10 | | | | 31 |
| 32 | Korcsok Peter | Gym. maďarské Šahy | 0 | 8 | 12 | 8 | | | | 28 |
| 33 | Bálint Farkaš | Gym. maďarské Šahy | 3 | 9 | | 6 | 1 | | 10 | 26 |
| 33 | Gramblička Martin | Gym. Jura Hronca BA | 3 | 0 | | 12 | | | 14 | 26 |
| 35 | Panák Pavol | Gym. Jura Hronca BA | 2 | 2 | 10 | 11 | | | | 23 |
| 36 | Novák Ján | Gym. Ľudovíta Štúra Trenčín | 2 | 0 | | 5 | 4 | | 12 | 21 |
| 37 | Wertlen Andrej | Gym. Jura Hronca BA | 3 | 0 | | 12 | 8 | | | 20 |
| 38 | Bažík Martin | Gym. Jura Hronca BA | 2008 | 0 | 10 | 9 | | | | 19 |
| 39 | Dekan Martin | Gym. Jura Hronca BA | 3 | 0 | | | 4 | | 13 | 17 |
| 39 | Zachar Lukáš | Gym. A.Sládkoviča B. Bystrica | 3 | 17 | | | | | | 17 |
| 41 | Kováčovský Tomáš | Neznama skola | 2008 | 15 | | | | | | 15 |
| 42 | Bodnár Jozef | Gym. Fiľakovo | 3 | 13 | | | | | | 13 |
| 43 | Majzeľ Tomáš | SPŠE Michalovce | 2 | 0 | | | | | 12 | 12 |
| 44 | Kolesár Štefan | Gym. Alejová Košice | 2 | 5 | | 6 | | | | 11 |
| 44 | Schuster Vladimír | Gym. Jura Hronca BA | 2008 | 0 | | 11 | | | | 11 |
| 46 | Macák Michal | Gym. Giraltovce | 4 | 10 | | | | | | 10 |
| 46 | Vanderka Peter | Gym. Trstená | 3 | 10 | | | | | | 10 |
| 48 | Pandoščák Michal | SPŠE Michalovce | 2 | 3 | | 6 | | | | 9 |
| 48 | Stranaiova Lucia | Gym. Jura Hronca BA | 2008 | 0 | | 9 | | | | 9 |
| 50 | Hankovský Dávid | SPŠE Michalovce | 2 | 0 | | 5 | | 3 | | 8 |

| | Meno a priezvisko | Škola | Ročník | | 21 | 22 | 23 | 24 | 25 | Σ |
|---|---|---|---|---|---|---|---|---|---|---|
| 50 | Kurpel Matej | Gym. Ľudovíta Štúra Trenčín | 2 | 0 | 8 | | | | | 8 |
| 50 | Sivák Vladimír | Gym. Ľudovíta Štúra Trenčín | 2 | 8 | | | | | | 8 |
| 53 | Jacko Vladimír | Gym. Alejová Košice | 2008 | 1 | | 6 | | | | 7 |
| 53 | Oremus Vladimír | Gym. Jura Hronca BA | 2 | 1 | | 6 | | | | 7 |
| 55 | Gazda Tomáš | SPŠE Michalovce | 2 | 0 | | 6 | | | | 6 |
| 56 | Balga Jozef | Gym. maďarské Šahy | 4 | 4 | | | | | | 4 |
| 56 | Kompuš Patrik | Gym. Šrobárova Košice | 1 | 4 | | | | | | 4 |
| 56 | Petrezsél Tibor | Gym. maďarské Šahy | 4 | 4 | | | | | | 4 |
| 59 | Paľko Peter | SPŠE Michalovce | 2 | 3 | | | | | | 3 |
| 60 | Bredová Lucia | Gym. Šrobárova Košice | 1 | 2 | | | | | | 2 |
| 60 | Hopková Dominika | Gym. Šrobárova Košice | 1 | 2 | | | | | | 2 |
| 60 | Mondoková Denisa | Gym. Šrobárova Košice | 1 | 2 | | | | | | 2 |
| 60 | Tokolyova Zuzana | Gym. Šrobárova Košice | 1 | 2 | | | | | | 2 |
| 64 | Lizák Ivan | Gym. Jura Hronca BA | 2 | 2 | | | | | | 2 |
| 65 | Klimo Vladimír | Gym. Jura Hronca BA | 2 | 1 | | | | | | 1 |
| 65 | Verčimák Viliam | SPŠE Michalovce | 2 | 1 | | | | | | 1 |
| 67 | Beran Jakub | Gym. Alejová Košice | 2 | 0 | | | | | | 0 |