Korešpondenn seminr z programovania
XVIII. ronk, 2000/2001
Katedra vyuovania informatiky MFF UK,
Mlynsk Dolina, 842 48 Bratislava

KSP finanne podporuje nadcia Open Society Fund Bratislava a IUVENTA – zariadenie pre voľn as det, mldeže a dospelch

Kateg¡=ria Z

Vzorové riešenia 2. kola kategórie Z

Milí riešitelia,
Dostávajú sa vám do rúk vzorové riešenia. Prajeme vám veľa úspechov pri ich čítaní a dúfame, že ich pochopíte. S najúspešnejšími z vás sa stretneme na jarnom sústredení.

KSPci

1. Zaneprázdnený knihovník

opravovalo YoYo–δ
(max. 14 bodov)

Aj keď tento príklad bol najľahší, tým prísnejšie bolo bodovanie. Popis bol cenený za dva body, musel ale obsahovať aspoň pokus o odhad časovej zložitosti. Za správnu implementáciu každej z operácií `polož`, `navrchu`, `zober` sa dalo získať po 3 body. Za výpis kníh na konci sa dali získať tiež 3 body. Za prípadnú zhoršenú zložitosť niektorých operácií sa samozrejme strhávali body.

Bodovanie je jasné, povedzme si niečo o vašich riešeniach. Ako väčšina[1] z vás konštatovala v popise, išlo o to implementovať zásobník. Riešenia sa dali rozdeliť na dve skupiny podľa spôsobu implementácie. Tie v prvej používali statické pole a pointer na koniec zásobníka. Riešenia z druhej skupiny dynamicky alokovali jednotlivé prvky a spájali ich do zoznamu. Najčastejšou chybou bolo neošetrenie prázdneho zásobníka pri operáciách `navrchu` a `zober`. Ďalej ste boli upozorňovaní na vymazávanie celého poľa, ktoré je pri správnej implementácii zbytočné.

Poďme teda k vzoráku. Mená kníh budeme ukladať do poľa. Prosím? Že máme dve kopy? No nič, tak budeme mať dve polia. Alebo bude len jedno, dvojrozmerné. Prvý rozmer bude od 1 po 2. Prvá kniha (teda kniha na spodu kopy) sa uloží na prvú pozíciu, ďalšia na druhú, ... A kde je posledná? Potrebujeme ešte teda ukazovateľ na poslednú knihu. Bude to premenná typu integer a bude obsahovať index poslednej vloženej knihy alebo 0 v prípade, že žiadne knihy na kope nie sú. Na začiatku ju teda nastavíme na 0.

Ako bude vyzerať `polož`? Procedúru si nazveme `PUSH`, aby bolo jasné, že implementujeme zásobník. Premenná `Vrch[kopa]` nám ukazuje na poslednú knihu. Zvýšime ju teda o jedna a ukazuje nám na miesto za ňou. Na toto miesto teraz zapíšeme názov novej knihy. `Vrch[kopa]` nám teda opäť ukazuje na poslednú knihu na kope. Pozrime sa na `navrchu`. Anglický programátorský ekvivalent: `TOP`. Ak `Vrch[kopa]=0`, tak je zásobník prázdny. Ak `Vrch[kopa]>0`, tak potom `zasobnik[kopa,vrch[kopa]]` je najvrchnejšia kniha.

Zostáva nám `zober`. Procedúra `POP` vo vzoráku aj vracia názov odobranej knihy. Aký je ale postup? Opäť, ak `Vrch[kopa]=0`, tak je kopa prázdna a nemôžeme robiť nič. Ak nie, vrátime názov najvrchnejšej knihy (uložíme ho do premennej `s`) a znížime `Vrch[kopa]` o jedna. Tá nám teda ukazuje na teraz už poslednú knihu. Ak predtým `Vrch[kopa]` bola 1, tak je teraz 0, to znamená, že kopa je prázdna.

Na konci programu je ešte treba vypísať všetky zostávajúce knihy na kope. Väčšina riešení prešla cyklom od `Vrch[kopa]` po 1 a vypisovala hodnoty poľa `zasobnik`. Vzorové riešenie používa procedúru `POP`, až kým sa zásobník nevyprázdni. Výhoda takéhoto riešenia?

[1] No, väčšina... Väčšina nemala žiadny popis.

Predstavte si, že sa rozhodnete prepísať túto implementáciu na tú s dynamickou alokáciou. Stačí vám prepísať procedúry `PUSH`, `TOP`, `POP` a zvyšku programu sa nemusíte ani dotknúť.

Dobre, ešte potrebujeme odhad zložitosti. Všetky tri operácie `PUSH`, `TOP`, `POP` sa vykonajú v konštantnom čase ($O(1)$). Pri každej operácii sa totiž vykoná iba konštantný počet operácií, nezávislý na vstupných údajoch. Výpis kníh na konci závisí od počtu kníh na kope. Nech N je počet príkazov, ktoré sa vykonajú. Aký je maximálny počet kníh na oboch kopách po ich vykonaní? Dosiahneme ho, ak všetky príkazy budú polož a je to teda N. Výpis všetkých názvov kníh má teda zložitosť $O(N)$. Celý program má teda lineárnu zložitosť $O(N)$. Vykoná sa totiž N príkazov s konštantnou zložitosťou a potom ešte výpis, celkovo teda $2N$ operácií.

Listing programu:

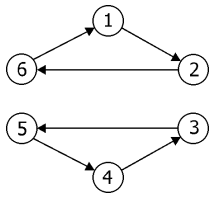
```
Program Knihovnik;
Const MaxKnih=100;
      MaxKopa=2;
Type TNazov=String[20];
Var Zasobnik:Array[1..MaxKopa,1..MaxKnih] of  TNazov;
    Vrch:Array[1..MaxKopa] of word;
    kopa:byte;
    nazov,prikaz:TNazov;

Procedure Init(i:byte);
Begin
  Vrch[i]:=0;          {staci nam nastavit ukazovatel na vrch zasobnika}
end;
Function Empty(i:byte):Boolean;
Begin
  Empty:=(Vrch[i]=0);                    {Ak je vrch 0, tak je prazdny}
end;
Procedure Push(i:byte; S:TNazov);
Begin
  Inc(Vrch[i]);                            {Zvysime ukazovatel o jedna}
  Zasobnik[i,Vrch[i]]:=S;                              {a ulozime nazov}
end;
Procedure Pop(i:byte; var  S:TNazov);
Begin
  If Not Empty(i) then Begin                                {Ak mozeme,}
    S:=Zasobnik[i,Vrch[i]];
    Dec(Vrch[i]);             {tak znizime ukazovatel na vrch zasobnika}
  end;
end;
Procedure Top(i:byte; var S:TNazov);
Begin
  If Not Empty(i) then                    {Ak nie je zasobnik prazdny,}
    S:=Zasobnik[i,Vrch[i]];                 {tak vratime vrchny zaznam}
end;

Begin
  For kopa:=1 to 2 do Init(kopa);      {obidva zasobniky zincializujeme}
  repeat
    write('>'); Readln(prikaz);        {nacitame prikaz}
    If prikaz='poloz' then Begin
```

```
      write('------->kopa: '); Readln(kopa);      {nacitame cislo kopy}
      write('------->nazov: '); Readln(nazov);             {  a nazov }
      Push(kopa,nazov);                        { a vlozime do zasobnika}
    end    {poloz}
    else
    If prikaz='navrchu' then Begin
      write('------->kopa: '); Readln(kopa);      {nacitame cislo kopy}
      If not Empty(kopa) do Begin
        Top(kopa,nazov);                   {pozrieme prvok na zasobniku}
        Writeln('---> Navrchu ',kopa,'. kopy je kniha: ',nazov,'.');
      else
        Writeln('---> ',kopa,'. kopa je prazdna.');
    end   {navrchu}
    else
    If prikaz='zober' then Begin
      write('------->kopa: '); Readln(kopa);      {nacitame cislo kopy}
      If not Empty(kopa) then                       {ak nie je prazdna}
        Pop(kopa,nazov);                {vyberieme najvrchnejsiu knihu}
    end; {zober}
  until prikaz='koniec';
                                              {na konci uz len vypiseme }
  For kopa:=1 to 2 do Begin                   {zvysne knihy na kopach   }
    If Not Empty(kopa) then Begin                {ak nie je prazdna kopa}
      Writeln;
      Writeln('Na ',kopa,'. kope su knihy:');
      While not Empty(kopa) do Begin            {tak kym nebude prazdna }
        Pop(kopa,nazov);                        {odvrchu vyberame knihy }
        Writeln(Nazov);                         {a vypisdujeme ich      }
      end;
    end;  {empty kopa }
  end;  {for kopa}
end.
```

2. Zo zapadnutej dedinky

opravoval JanoS
(max. 15 bodov)

Chúďa Janka. Jej problém s klebetami bol asi väčší, ako predpokladala, keď si na ňom vylámalo zuby toľko dobrých programátorov. Mnohí z vás sa dali oklamať poznámkou, nepremýšľali toľko a svorne si mysleli, že ak každému obyvateľovi niekto verí, tak sa klebeta rozšíri. Nie je to však pravda, ako sa ľahko presvedčíte pohľadom na obrázok. Takéto riešenie bolo zväčša ohodnotené **jedným bodom** a vyskytlo sa asi u 40% riešiteľov.



Ďalšia metóda, ako riešiť túto úlohu, bola postupným iterovaním. Taktiež by sa dala nazvať simuláciou. Takéto riešenia si pamätali, ku komu sa dostala klebeta a v každom kroku ju rozšírili všetkým dôveryhodným ľuďom. Takéto riešenia používali N až N^2 krokov, a každý krok im trval M až N^2 operácií, kde N je počet ľudí v dedine a M je počet vzťahov *kto komu dôveruje*. Celková časová zložitosť takýchto riešení sa teda pohybovala medzi $O(MN)$ a $O(N^4)$. Podľa prefíkanosti riešení som udeľoval **5 až 9 bodov**.

- Nefunkčné a zlé riešenia – 0–1 bod
- Iteratívne riešenia $O(N^4)$ až $O(M^2)$ – 5–7 bodov

- Riešenia s časmi $O(N^3)$ až $O(NM)$ – 8–9 bodov
- Vzorové riešenia $O(N^2)$ až $O(M)$ – 13–14 bodov

Tak a teraz prichádzame ku samotnému vzorovému riešeniu, za ktoré bolo **13 až 14 bodov**. Úloha sa optimálne dala riešiť dvoma prístupmi, rekurzívne alebo frontou. My si ukážeme jednoduchšie z nich, a to rekurzívne.

Majme pole `klebeta`, do ktorého si uložíme, či daný občan vie klebetu, alebo nie. Toto pole má na začiatku nulový obsah, iba na prvom políčku má jednotku. Ďalej si vytvoríme jednu procedúru, `klebeť`, ktorá dostane ako parameter číslo občana X a rozšíri klebetu známym občana X. `Klebeť` bude predpokladať, že občan X už klebetu vie (teda že `klebeta[X]=1`) a urobí to, že všetkým známym občana X, ktorí ešte klebetu nevedia (`klebeta[Znamy]=0`), ju povie a opätovne zavolá `klebeť(Znamy)`.

Keď teraz spustíme procedúru `klebeť` s parametrom 1, klebeta sa rozšíri po celej dedine presne tak, ako keby ju rozprávali občania. Takýto prístup nazývame prehľadávanie do hĺbky. Na to, aby takýto algoritmus bol korektný, musí spĺňať dva predpoklady: musí byť správny a nesmie sa zacykliť.

Správnosť ukážeme v dvoch krokoch. Prvý krok: ak program o niekom povie, že sa klebetu dozvie, tak je to naozaj pravda. Z postupu vidno, že program používa iba známe vzťahy dôvery, čiže správa sa naozaj šíri iba tak, ako sa bude šíriť v skutočnosti. Druhý krok: Ak sa správa k niekomu dostane, povie to aj náš program. Ak sa ku niekomu správa dostala, tak existovala cesta od zdroja signálu ku danému občanovi. Naša rekurzívna procedúra sa však naisto volala pre všetkých občanov na danej ceste (keďže si museli dôverovať) a nakoniec označila aj posledného občana. Z tohoto vyplýva, že algoritmus je správny.

Konečnosť algoritmu dokážeme nasledovne. Koľkokrát sa volá procedúra `klebeť`? Na to, aby sa zavolala, musí sa nejaký občan práve dozvedieť klebetu. Keďže je občanov iba N, aj táto procedúra môže byť zavolaná pre každého občana len jedenkrát, a samotná procedúra je evidentne konečná.

Odhad časovej zložitosti: Ako bolo písané vyššie, `klebeť` sa zavolá nanajvýš N–krát, a v každom kroku pozrie všetkých známych občana X, ktorých môže byť nanajvýš N. Výsledná časová zložitosť je teda $O(N^2)$. Jednoduchým trikom sa tento čas dal ešte o chlp znížiť. Problém je, že procedúra `klebeť` vždy kontroluje všetkých ostatných občanov, či im občan X dôveruje. Takýmto prístupom každý krok trvá N operácií. Keď však použijeme prefíkanejšiu dátovú štruktúru na uloženie dôverných vzťahov, tak potom pre každého občana skontrolujeme len tých, ktorým on naozaj verí. V súčte potom vykonáme M operácií, čo môže byť menšie ako N^2. Ako vyzerá naša „prefíkaná“ dátová štruktúra?

Použijeme obyčajné dvojrozmerné pole $N \times (N+1)$, kde však neukladáme na pozíciu i, j fakt, že občan i dôveruje občanovi j, ale do i–teho riadku si uložíme na nulovú pozíciu počet známych občana i, nech je to q, a na prvých q pozícií uložíme jeho známych. Takto ich vieme prezrieť bez toho, aby sme sa zaoberali ľuďmi, ktorým nedôveruje.

Vzorový program má teda zložitosť $O(M)$. Nakoniec som udeľoval **-1 až 1 bod** podľa prítomnosti a kvality popisu, odhadov zložitostí, osobných preferencií a aktuálneho počasia. Nejaký ten bodík som strhol za závažnejšie chyby.

Listing programu:

```
Program Zo_zabudnutej_dedinky;
Const Max=50;
Var N,M,I,Kto,Komu,Pocet:Integer;
    Dovera:Array[1..Max,0..Max] Of Integer;  {V 0 stlpci je pocet znamych}
    Klebeta:Array[1..Max] Of Boolean;

Procedure Klebet(Kto:Integer);              {Rozsiri Klebetu od obcana KTO}
Var Kam:Integer;                            {Kam klebetu sirime...        }
```

```
Begin
   For Kam:=1 To Dovera[Kto,0] Do          {Vsetkym nasim znamym...      }
   If not Klebeta[Dovera[Kto,Kam]] Then    {Ak este klebetu nepoculi...  }
   Begin
      Klebeta[Dovera[Kto,Kam]]:=True;      {tak im ju povieme...         }
      Klebet (Dovera[Kto,Kam]);            {a prikazeme sirit dalej...   }
   End;
End;

Begin
   Readln(N);       {N bude pocet obyvatelov}
   Readln(M);       {M bude pocet dvojic doveryhodnych ludi}
   Fillchar(Dovera,Sizeof(Dovera),0);
   For I:=1 To M Do Begin
      Readln(Kto,Komu);
      Inc(Dovera[Kto,0]);              {Doveruje dalsiemu cloveku}
      Dovera[Kto,Dovera[Kto,0]]:=Komu; {a to je clovek KOMU      }
   End;
   FillChar(Klebeta,Sizeof(Klebeta),False);
   Klebeta[1]:=True;                 {Zdroj klebiet}
   Klebet(1);                        {Rozkazovaci sposob od klebetit}

   Pocet:=0;
   For I:=1 To N Do
      If Klebeta[I] Then Inc(Pocet); {Zratame kolko ludi poculo klebetu}
   If Pocet=N Then Writeln('ANO, klebetu sa kazdy dozvie.')
              Else Writeln('NIE, klebeta sa nerozsiri.');
End.
```

3. Zúfalý tesár

opravoval Vlado
(max. 15 bodov)

Zrejme všetci tí, ktorí sa pokúsili tento príklad riešiť, zistili, že išlo naozaj o ľahučký príklad. Až na zopár výnimiek, ktoré by sa na jednej ruke spočítať dali (no... vlastne na dvoch), mali všetci tento príklad viac–menej dobre. Avšak to viac–menej je tu oprávnené, pretože, ako vám môže prezradiť pohľad do výsledkovej listiny, len trom z vás sa podarilo získať plný počet bodov, teda 15. A že prečo je to tak?

Za riešenie, ktoré bolo funkčné, ste mohli získať 10–12 bodov, pričom vaše riešenia možno rozdeliť do niekoľkých kategórií:

- **CAT 1** – riešenia, ktoré postupne odčitovali od roku hodnoty rímskych cifier v poradí od najvyššej, pričom za cifry berieme aj výnimky, teda `IV`, `IX`, `XL`, `XC`, `CD` a `CM`. Takéto riešenie mohlo získať 12 bodov (pozri prvé vzorové riešenie).
- **CAT 2** – riešenia, ktoré odčitovali len hodnoty základných cifier (teda nie výnimiek) a zároveň generovali rímsky zápis čísla, a až potom hľadali výnimky, čo bolo buď cez vyhľadávanie podreťazcov, alebo (**CAT 2.1**) pomocou počtov jednotlivých rímskych cifier. Riešenia tejto skupiny mohli získať najviac 10 bodov.
- **CAT 3** – riešenia tejto kategórie sa pre každú cifru desiatkového zápisu rozhodli, koľko úderov treba na vytesanie danej cifry a priamo tento počet pripočítali k výsledku, prípadne (**CAT 3.1**) najprv vygenerovali rímsky zápis (aj s výnimkami) a v ňom následne spočítali počet úderov. Tieto riešnia mohli získať takisto 12 bodov (pozri druhé vzorové riešenie).

Ďalej, niekoľko z vás si poriadne neprečítalo zadanie, takže sa napríklad snažilo používať aj iné výnimky, napríklad **IM**. Body som za to nestrhával, ale zbytočne ste robili niečo navyše. V tomto prípade sa vám v riešení za kategóriou objavilo písmenko `A`. Strhnuté body sa dali získať chýbajúcim popisom, či nejakou menšou chybou. Nefunkčné riešenia získavali $\frac{1}{2}$ až $\frac{1}{3}$ bodov z maxima tej-ktorej kategórie.

No, a za čo že sa dali získať tie zvyšné 3 body? Hovorí vám niečo „zdôvodnenie správnosti“? Áno, presne za to...

Vzorové riešenie (1): K uvedenému už niet veľmi čo dodať, jednoducho sa snažíme postupne odčítať najväčšie číslo z možných „cifier“, teda v poradí 1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4 a 1. Pri každom odčítaní vieme, akú „cifru“ sme vlastne použili, a preto si môžeme pripočítať počet úderov potrebných na vytesanie tejto „cifry“. Ak sa dostaneme k nule, máme zistený počet úderov na toto číslo.

No, a teraz zopár slov k tomu zdôvodneniu správnosti. Pozrime sa na 2 prípady, konkrétne na `CM`, teda 900, a na `CD`, teda 400 (ostatné výnimky sú ekvivalentné). Nech n je náš spracúvaný rok, predpokladajme, že už sme v štádiu, kedy $n < 1000$, a nech $n \geq 900$, teda cheme použiť výnimku `CM`. Položme si otázku: „Čo by vytesal tesár, ktorý nepozná výnimky?“ Vytesal by `DLLLL`, v tomto poradí. My však vieme, že to je výnimka `CM`, a navyše, po jej použití sa dostaneme do toho istého stavu (čo sa týka zvyšku n po odčítaní `CM`), ako sa dostal tesár bez výnimiek (po odčítaní `DLLLL`). Teda tesár bez výnimiek by vytesaním `DLLLL` vykonal vlastne to isté, ako vykonáme my vytesaním `CM`, a ďalej budú pokračovať z rovnakej pozície. Preto budeme uvažovať o tejto výnimke práve tu, teda medzi 1000 a 500. Pre výnimku `CD` je situácia podobná, no a pre všetky ostatné prípady vlastne identická.

Listing programu:

```
const tab:array[1..2,1..13] of integer = (
            (1000,900,500,400,100,90,50,40,10,9,5,4,1),
            (   4,  6,  3,  5,  2, 4, 2, 4, 2,3,2,3,1));
var i,n,count:integer;
begin
  readln(n);  count:=0;
  for i:=1 to 13 do
  begin
    count:=count+tab[2,i]*(n div tab[1,i]);
    n:=n mod tab[1,i];
  end;
  writeln(count);
end.
```

Vzorové riešenie (2): Uvažujme zápis roku n v desiatkovej sústave. Tento zápis bude mať zrejme najviac 4 cifry. Je dôležité si všimnúť nasledovnú skutočnosť: naše výnimky sú také, že každá z nich zasahuje **práve** do jedného rádu, t.j. buď do stoviek, do desiatok alebo do jednotiek.[2] Preto ak vezmeme všetky roky také, ktoré majú napríklad na mieste desiatok cifru **4**, bude mať táto štvorka v každom z týchto čísel rovnaký zápis, konkrétne to bude `XL`. Na základe tejto úvahy si môžeme jednoducho vytvoriť tabuľku, ktorá pre každé miesto v desiatkovom zápise a pre každú možnú cifru na tomto mieste určuje, koľko úderov bude mať jej zápis. Teda napríklad cifra `3` na mieste tisícok bude mať vždy zápis `MMM`, teda 12 úderov, `9` na mieste stoviek bude vždy `CM`, teda 6 úderov a podobne. Potom je už úloha naozaj jednoduchá, stačí sa (najviac 4–krát) pozrieť do tabuľky, sčítať a máme výsledok.

[2] Skúste sa pozrieť, čo by sa stalo v prípade, že by boli povolené aj iné výnimky

Listing programu:

```
const tab:array[1..4,0..9] of integer = ((0,1,2,3,3,2,3,4,5,3),
   (0,2,4,6,4,2,4,6,8,4),  (0,2,4,6,5,3,5,7,9,6),  (0,4,8,12,16,0,0,0,0,0));
var i,n,count:integer;
begin
  readln(n);  count:=0;
  for i:=1 to 4 do
  begin
    count:=count+tab[i,n mod 10];
    n:=n div 10;
  end;
  writeln(count);
end.
```

Pamäťová a časová zložitosť: Ako si mnohí všimli, časová, ako aj pamäťová zložitosť oboch uvedených riešení bola $O(1)$, nakoľko si vždy pamätáme len konštantné množstvo údajov a vykonávame konečný počet krokov, ktorý (ak uvažujeme, že vstup bol zhora ohraničený[3]) nezávisí od vstupu.

4. Zvláštne zariadenie

opravovala Ňaňka
(max. 15 bodov)

Tento príklad bol veľmi úspešný. Možno z dôvodu, že nejaké riešenie sa dalo napísať, a nemalo až tak zložitú myšlienku... Proste si to odsimulovať. Niektorí však pochopili, že musí existovať aj iné riešenie, a tak vzali do hrsti rozum[4] a snažili sa to vylepšiť. Vaše riešenia možno rozdeliť takto:

- Vzorové riešenie $O(n)$ – 15 bodov
- Riešenia, kde každá blcha skákala najviac n skokov $O(n\,min(n,k))$ – 13 bodov
- „Simulácia“ zariadenia $O(k\,n)$ – 10 bodov
- Horšia časová zložitosť, napr. $O(k\,n^2)$ – 7–8 bodov
- Nefunkčné riešenia – 1–2 body

Za popis som už tradične strhávala 1 až 2 body. Za prílišnú pamäťovú zložitosť ste mohli prísť o 1 bod.

Ale vráťme sa k vzorovému riešeniu. Z minulého kola vieme, že permutáciu (lebo tento príklad je naozaj o permutácii) možno rozložiť na cykly. Tie majú tú dobrú vlastnosť, že sú disjunktné, čo znamená, že žiadne dva cykly nebudú mať spoločný prvok.

Čo to znamená pre nás? Znamená to, že každá blcha skáče v niektorom takomto cykle (čiže iba po niektorých, možno aj všetkých, vrcholoch) a na iné nikdy neskočí. Vieme tiež, že po nich skáče periodicky. Teda po l skokoch, kde l je dĺžka cyklu, sa vráti na svoje pôvodné miesto a zase skáče ďalej.

Preto, keď si vezmeme ľubovoľný cyklus tejto permutácie, vieme jednoznačne povedať, kam blcha skočí za K skokov. Ak sa cyklus skladá z prvkov c_0, c_1,..., c_{l-1}, čo znamená, že z vrchola c_i sa skáče na vrchol c_{i+1} $(0 \leq i < l-1)$ a z vrchola c_{l-1} na c_0, tak blcha z vrchola c_i bude po K skokoch na vrchole c_j, kde $j = (i+k) \bmod l$.

Teraz už stačí zobrať si vzorák z minulého kola a pozrieť, ako sa cykly hľadajú a pre každý cyklus si pre všetky blchy v ňom zistiť, kam skočia.

[3] akože aj bol

[4] prípadne vzoráky z minulého kola

A nakoniec drobná poznámka. Niektorí ste si všimli a niektorí nie, že sa nám do vzorového výstupu v zadaniach schovala malá chybička. U vás sa tiež v hojnom množstve vyskytovala. Problém bol v tom, že sme/ste si mysleli, že si pre každý vrchol uchovávame informáciu o blche. Ono to však je naopak. My si pre každú blchu pamätáme, kde sa práve nachádza. Preto na opravenie tejto chyby stačí výstup pred vypísaním vhodne zmeniť[5].

Listing programu:

```
program Zvltne_zariadenie;
const nmax=100;                       {maximum b‰ch}
var A,V:array[1..nmax] of integer;    {natanie,vĂsledok }
    C:array[0..nmax] of integer;      {cyklus             }
    N,K,l,i,j:integer;

procedure Spracuj_cyklus(l:integer);{spracovva cykly permutcie}
var i:integer;
begin
  C[0]:=C[l];                         {mod d vĂsledok z intervalu 0..l-1}
  for i:=1 to l do A[C[i]]:=-C[(i+K) mod l];
end;

begin
 readln(N,K);
 for i:=1 to N do readln(j,A[j]);
 for i:=1 to N do                     {had cykly permutcie}
  if A[i]>0 then
  begin
    l:=1; C[l]:=i; j:=i;
    while A[j]<>i do begin            {kĂm sa a prvok nerovn prvmu}
      inc(l); j:=A[j]; C[l]:=j;
    end;
    Spracuj_cyklus(l);
  end;
 for i:=1 to N do C[-A[i]]:=i;        {pozri poznmku v popise...}
 for i:=1 to N do writeln(i,'. vrchol - blcha c. ',C[i]);
end.
```

5. Zeofína v ríši fraktálov

opravovala Meri
(max. 15 bodov)

Iste vás zaujíma, ako sa tento príklad bodoval. Nuž, takto:

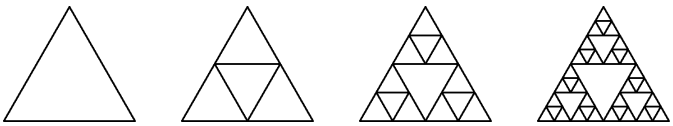
- správne riešenie – 12–15 bodov
- funkčné riešenie, ale Zeofínka sa zbytočne veľa nachodí – 8–11 bodov
- nefunkčné alebo čiastočne funkčné riešenie – najviac 7 bodov

Ďalšie 2 body sa dali stratiť za chýbajúci alebo nedostatočný popis programu alebo odhad zložitosti.

A aké je teda to vzorové riešenie? Nuž, pozrime sa, ako sú trojuholníčky usporiadané. Trojuholník rádu 1 je vlastne úplne obyčajným trojuholníkom. Trojuholník rádu 2 je zložený z trojuholníka rádu 1, do ktorého je vložený menší rovnostranný trojuholník. Trojuholník

[5] pozri vzorové riešenie

rádu 3 pozostáva z trojuholníka rádu 2, ktorému z každej strany vnútorného trojuholníka vyrastá jeden mrňavučký trojuholníček. Trojuholník rádu 4 ... atď, až kým Zeofínka nedostane poriadnu svalovicu.



rád 1 rád 2 rád 3 rád 4

Toto nám priamo dáva návod na konštrukciu programu. Najprv si nakreslíme vonkajší trojuholník tak, aby sme skončili v polovici niektorej jeho strany. Otočíme sa doľava a začneme kresliť výplň. Na to by nám mohla poslúžiť rekurzívna procedúrka. Výplň sa skladá z vloženého trojuholníka a tak ho teda začne kresliť. Keď sa dostane do polovice prvej strany, zistí, že by z nej mohol „vyrastať“ menší trojuholníček. Lenže, ten je (až na veľkosť strany) rovnaký, ako ten, ktorý práve kreslí. Takže, ak z nej vyrastá, tak sa otočí a zavolá sama seba. Po dokončení vnútorného trojuholníka dokreslí stranu, otočí sa a dokreslí aj zvyšné dve strany podobným postupom.

Takáto procedúrka musí vedieť dve veci: dĺžku strany trojuholníka a to, či z neho vyrastá ešte aj nejaký ďalší trojuholníček. Na začiatku si zistíme, koľko krát by sa procedúrka mala vnoriť (toto číslo sa rovná rádu trojuholníka-1 – to vyplýva z jeho popisu) a zadáme jej ho ako parameter `rad`. Keď bude volať sama seba, zavolá sa s parametrom `rad` o 1 menším. Ak by dostala ako parameter jednotku, už by iba nakreslila jednoduchý trojuholník a nevnárala by sa. Pri rekurzívnom volaní musíme navyše upraviť aj dĺžku strany trojuholníka, ktorý má nakresliť (to je parameter `dlzka`), a to na jednu polovicu.

Časová zložitosť algoritmu je $O(3^N)$, lebo Zeofínka musí nakresliť práve toľko úsečiek. Toto uhádla väčšina z vás. Zato s pamäťovou ste mali väčšie problémy. Nie je konštantná ani exponenciálna, ale lineárna. Najviac pamäte zaberajú rekurzívne vnorené procedúrky na zásobníku. Naraz si ich potrebujeme pamätať najviac N. (Skúste si predstaviť, ako sa postupne vnárajú a vynárajú.)

Listing programu:

```
program Zeofina;
const dlzka=300;
var rad:integer;

procedure Sierpinsky(rad:integer,dlzka:real);
var i:integer;
begin
  for i:=1 to 3 do begin
    writeln('dopredu ',dlzka/2);
    if rad>1 then begin
      writeln('doprava 120');
      Sierpinsky(rad-1,dlzka/2);
      writeln('dolava 120');
    end;
    writeln('dopredu ',dlzka/2);
    writeln('dolava 120');
  end;
end;

begin
 write('Zadaj N: ');  readln(N);
```

```
 writeln('dopredu ',dlzka div 2);
 writeln('dolava 120');
 writeln('dopredu ',dlzka);
 writeln('dolava 120');
 writeln('dopredu ',dlzka);
 writeln('dolava 120');
 writeln('dopredu ',dlzka div 2);
 writeln('dolava 60');
 if rad<1 Sierpinsky(rad-1,dlzka/2);
end.
```

Vsledkov listina po 2. srii kateg¡=rie KSP-Z

	Meno a priezvisko		řkola		21	22	23	24	25	Σ
1.	Mravec Pavol	3	Gym. K.tra Modra	75	14	14	12	15	15	145
2.	Katreni Jn	3B	Gym. Spi. Nov Ves, kolsk	73	12	15	10	15	15	140
3.	TruchlÀ Peter	3	Gym. Nitra, Provsk 1	65	14	15	10	15	15	134
4.	Vozr Oto	2	Gym. Matky Alexie Bratislava	67	13	15	12	14	11	132
5.	Teke Michal	3E	Gym. Preov, Kontantnova	62	14	14	12	11	15	128
6.	Steinov Monika	3A	Gym. Bratislava, Einsteinova	64	12	15	11	13	11	126
7.	rmek Ras»o	3A	Gym. Bratislava, Tilgnerova	62	11	13	10	10	15	121
8.	Trenkler Pavol	1A	Gym. Koice, Zbrojnin	50	14	14	12	10	15	115
9.	Minrik Gabriel		SP Komrno, Petofiho	57	14	9	12	8	14	114
10.	Choma Martin	2	Gym. Star -ubova	65	14	1	12	10	9	111
11.	Palenr Jan		Gym. V.Paulnyho-T Martin	59	14	14	12	10		109
12.	Rusnk Pavol	3E	Gym. Preov, Kontantnova	54	12	13	12		15	106
13.	Kansz Rbert	2	Gym. Koice, Dnepersk	49	12	13	10	10	11	105
	Libi Peter	3C	Gym. -.tra, Trenn	48	13	13	12	10	9	105
15.	Kavka Michal	3B	Gym. Koice, Alejov	57	12	9	12	10	2	102
16.	Novk Andrej	3B	Gym. ilina, Hlinsk	59	14	1	15	10	2	101
17.	Kvasnika Martin	3A	Gym. A.Markua Bratislava	48	12	0	10	14	14	98
	Baran P	2B	Gym. J. Hronca Bratislava	52	12	12	12	10		98
19.	Frlika Martin	2B	Gym. J. Hronca Bratislava	50	12	1	9	10	15	97
20.	Vranec Maro	2A	Gym. Koice, Alejov	36	12	8	15	10	11	92
21.	Karas Filip	1B	Gym. Nitra, Golianova 68	53	10	4	8	6	10	91
	Bal Miroslav	1A	Gym. J. Hronca Bratislava	29	13	15	9	10	15	91
23.	Jirsek Joko	6	Gym. Koice, Zbrojnin	48	11		12	8	11	90
	KonenÀ tefan	2D	Gym. J. Hronca Bratislava	41	12	0	12	10	15	90
25.	Mark Lszl	2	Gym. H.Selyeho ma. Komrno	41	12	5	10	10	10	88
26.	tefanek Anton	1C	Gym. J. Hronca Bratislava	43	5	7	12	10	10	87
27.	Burda Milan	1B	Gym. Nitra, Golianova 68	36	11	4	10	6	10	77
28.	Psztor Tom	4A	Gym. ahy, Mldencka	25	12	8	10	6	13	74
29.	Koula JiŠ		.....	73						73
30.	Szarkov Eva	4C	Gym. J. Hronca Bratislava	35	14		12	10		71
	Max Matej	7A	Gym. Koice, Alejov	11	12	7	15	11	15	71
32.	Dovjak Marek	3	Gym. P.O.H. Kemarok	29	14	8	4	11		66
33.	Hanakovi Tom	4A	Gym. sv.Frantika ilina	64						64
34.	Havlj Frantiek	4	.....	63						63
35.	Klnai Peter	3A	Gym. Levice	16	12	14	10	10		62
36.	omlo Ivan	1	Gym. ahy, Mldencka	60						60
37.	Olejnk tefan	1A	Gym. J. Hronca Bratislava	33	5	0	11	10		59
38.	HodermarskÀ Ladislav	3A	SPE Star Tur	25	13		10	10		58
39.	Schmotzer Marin	1C	Gym. Bratislava, I.Horvtha	24	10	1	10	10		55
40.	Topor Peter	4	Gym. Povask Bystrica	53						53
	ChudÀ Michal	3B	Gym. Levice	29	12	0	4	8		53
	Gergely Jakab	4A	Gym. ahy, Mldencka	13	10	8	12	10		53
43.	ulk Radovan	1B	Gym. Nitra, Golianova 68	24	11	5	12			52
	Hutr Jn	2C	SPE Pie»any, SNP	16	13	1	12	10		52
45.	Dojr Jn		Gym. Turianske Teplice	16	12	1	12	10		51
46.	underlk Peter	1B	Gym. J. Hronca Bratislava	11	10	9		10	10	50
	Ambroz Peter	1B	Gym. J. Hronca Bratislava	0	10	9	10	12	9	50
48.	LuanskÀ Jn	3B	Gym. Trebiov, Komenskho	49						49
	Ivan Michal	3B	Gym. Nedoerskho Prievidza	49						49
50.	bodk Michal	4	Gym. Povask Bystrica	48						48

	Meno a priezvisko		řkola		21	22	23	24	25	Σ
	Kov Jakub	1C	Gym. J. Hronca Bratislava	0	7	8	12	10	11	48
52.	Studva Tom	3	Gym. A.Markua Bratislava	47						47
53.	Demko Martin		Gym. Koice, Alejov	17	6	5	3	6	9	46
	Luk Michal	3A	Gym. Koice, Alejov	24		1	12	9		46
55.	Pirchala Martin	2B	Gym. Koice, Alejov	45						45
56.	OpaternÀ Marek	3A	Gym. Levice	5	12	7	10	10		44
	Klimo Peter	3B	Gym. Bratislava, Hubenho	44						44
58.	Cimrk Martin	3B	Gym. Nitra, Golianova 68	20	13			10		43
	Sabadoov Oga	3B	Gym. Koice, Alejov	10	10	1	12	10		43
	Juliny Mrio		Gym. Pie»any, Nm.SNP 9	4	12	5	12	10		43
61.	Jank Oliver	1C	Gym. L.Stockela Bardejov	19	12	1		10		42
62.	Urban Jaroslav	3A	Gym. LiptovskÀ Hrdok	41						41
	Viravec Martin	3B	Gym. Svidnk	10	12	0	10	9		41
64.	Gregor Rastislav	2E	Gym. ilina, Vek Okrun	20	11	1	6		2	40
	Deknek Mat	1A	Gym. J. Hronca Bratislava	14	6		12	8		40
66.	Lajdov Jana		Gym. ilina-Vlince	24		1	6	8		39
67.	VeselÀ Jn	1B	Gym. Nitra, Golianova 68	38						38
68.	Teke Jakub	1C	Gym. J. Hronca Bratislava	0	14		12		11	37
69.	Kritofk tefan	2C	SP Levice, F.Heku	16		1	12	7		36
70.	Lovek Peter	4	Gym. Povask Bystrica	34						34
71.	Szab Zsolt	3	Gym. ahy, Mldencka	23	10	0				33
	Trejbal Ivan	1F	Gym. Koice, robrova	6	6			10	11	33
73.	Filo Michal	1B	Gym. Nitra, Golianova 68	0	12	1	10	9		32
74.	Molnr Tom	9A	Z Koice, Charkovsk 1	9				8	14	31
	Vrbel Tom	3A	Gym. V.Paulnyho-T Martin	31						31
	Trubeov Barbora	1A	Gym. J. Hronca Bratislava	0	10	9	2	10		31
	Kraicov Katarna	1B	Gym. A.Merici Trnava	0		9	10	6	6	31
78.	Kamr Jaroslav	3B	Gym. Koice, Alejov	10	5	4	6	5		30
	Sihelnk Slavomr	3A	Gym. Koice, Alejov	11	11			8		30
	Kardy Gabriel	1A	Gym. Koice, robrova	0	12		10	8		30
81.	Keru Tom		.....	0	12	0	10	7		29
82.	Andrejko Maro	7	Z Koice, Park Angelinum	10				8	10	28
	Kramari Michal	1C	Gym. J. Hronca Bratislava	0	10	0	10	8		28
84.	Kko Andrej	4D	Gym. Bratislava, Metodova	27						27
	Kubo Kamil	1A	Gym. J. Hronca Bratislava	0	5		12	10		27
86.	Fila Michal	1B	Gym. Nitra, Golianova 68	26						26
	Vojt Peter	9A	Z Koice, Charkovsk 1	9	11	6				26
88.	JanovickÀ Stanislav	2C	SP Levice, F.Heku	13			12			25
	Sege Andrej	3B	Gym. Nitra, Golianova 68	0	14		11			25
90.	ille Alexander	4B	Gym. ahy, Mldencka	24						24
	Lipkov Juliana		Gym. J. Hronca Bratislava	0	10		5	9		24
92.	RakovskÀ Martin	3B	Gym. Nitra, Golianova 68	23						23
	Kocsis Pavol	3A	Gym. Koice, Alejov	11		3	3	6		23
	Kov Tom	1A	Gym. J. Hronca Bratislava	0	11	0	2	10		23
	Duai Peter	3B	Gym. Koice, Alejov	0		1	12	10		23
96.	Rojko Pavol		Gym. Koice, robrova	3	8		11			22
	Svolk Peter	2C	SP Levice, F.Heku	0			12	10		22
	Hanulov Anna	1C	Gym. J. Hronca Bratislava	0	6	6	10			22
99.	Berec Ale	1B	Gym. Nitra, Golianova 68	0	11	0	10			21
100.	Hantke Richard	9A	Z Koice, Charkovsk 1	10				8		18
	Krchniakov Korina		Gym. J. Hronca Bratislava	18						18
	Kuis Jn	3B	Gym. Koice, Alejov	10	4	4				18
	Bal Daniel		Gym. Koice, Alejov	6	12					18

	Meno a priezvisko		řkola		21	22	23	24	25	Σ
	Bartov Dominika	1B	Gym. J. Hronca Bratislava	18						18
	Chriateov -ubica	1B	Gym. J. Hronca Bratislava	18						18
106.	Krul	1C	Gym. J. Hronca Bratislava	0	6	0	3	8		17
107.	tefanec Richard	1A	Gym. J. Hronca Bratislava	0	5		3	8		16
	Bartov Laura	3A	Gym. J. Hronca Bratislava	0	10				6	16
	Harvan Peter	1D	Gym. J. Hronca Bratislava	0	11		5			16
110.	Zachar Michal	2A	Gym. Koice, Trebiovsk	15						15
	Brilla Pavol	1D	Gym. Koice, robrova	3	12					15
112.	Vataha Martin	8	Gym. Snina	13		1				14
113.	Marek Pavol	2B	Gym. A.Merici Trnava	13						13
	Heimlich Dezider	4A	Gym. ahy, Mldencka	13						13
115.	Nemek Marek	2B	Gym. A.Merici Trnava	0	12					12
	Kirly Pavol	3A	Gym. Koice, Alejov	0		3	5	4		12
	Fodor Marek	3A	Gym. Koice, Alejov	0		3	5	4		12
118.	Pogny Peter	1B	Gym. H.Selyeho ma. Komrno	2	9					11
	Zemk Luk	8	Gym. Poprad, Popradsk nbr.	11						11
	Mandk Milan	2B	Gym. J. Hronca Bratislava	0	11					11
	Balazovocova Eva	1C	Gym. J. Hronca Bratislava	0	5	6				11
122.	Halamek Rastislav	1A	Gym. J. Hronca Bratislava	0	5	1	3			9
	Blicha Ladislav	3B	Gym. Koice, Alejov	0			5	4		9
124.	Alvk Jakub	8	Gym. Snina	8						8
	Sipos Robert		Gym. Koice, Alejov	8						8
126.	volik Peter	2C	SP Levice, F.Heku	7						7
	Sille Alexander		Gym. ahy, Mldencka	0		0		7		7
128.	Marko Peter		Gym. Koice, Alejov	6						6
	Tak Karol		Gym. Koice, Alejov	6			nn	nn		6
	Balogh Tom	1D	Gym. J. Hronca Bratislava	0			6			6
131.	Sask Peter		Gym. Koice, Alejov	5						5
132.	bert Tibor	3A	Gym. Levice	4						4
133.	Beck Patrik		Gym. .Moyzesa B.Bystrica	3						3
	Jasa Maro	1B	Gym. Koice, robrova	3						3
	StrmenskÀ Tom		Gym. Koice, robrova	3						3
136.	Kovik Vojtech	3	8-r. cirk. gym. ahy	1						1
	Torma Viktor	3	8-r. cirk. gym. ahy	1						1
	Papp Zoltn	3	8-r. cirk. gym. ahy	1						1
	Palko		Gym. Koice, Alejov	1						1
	Ochrnek Martin	1A	Gym. J. Hronca Bratislava	0		1				1
141.	Harmao Dominik	8	Gym. Snina	0						0
	Mazk		.....	0						0
	Trnka Rastislav	1D	Gym. Koice, robrova	0		0				0