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 1. kola kategórie Z

Milé deti !

Veľmi nás potešilo, koľkí ste sa zapojili do prvého kola a pevne dúfame, že v aspoň rovnakom počte budete riešiť aj ďalšie kolá. Teraz si už urobte pohodlie, vyložte nohy na stôl, zapáľte si svetlo, aby ste si nepokazili oči a začítajte sa do vzorových riešení, ktoré práve držíte v rukách.

KSPci

Na úvod by sme vám radi povedali (rovnako ako minulý a predminulý rok) niekoľko slov o tom, ako si predstavujeme ideálne riešenie.

Nechceme, aby ste sa zbytočne hrali so vstupom a výstupom. Samozrejme, zakazovať vám to nebudeme, ale bodíky navyše za to nedostanete a ešte môžete znechutiť opravovateľa, ktorý musí čítať množstvo kódu nesúvisiaceho s riešením samotnej úlohy. Pri načítavaní nemusíte zachovať formát vstupu uvedený v príklade, ale môžte si ho pozmeniť tak, aby sa vám jednoduchšie načítaval. Vo svojom riešení sa nemusíte zaoberať kontrolovaním správnosti vstupných údajov. Skôr by sme ocenili, keby ste inicializovli (napr. vynulovali) všetky premenné (a najmä polia), ktoré v programe použijete, aby sme videli, že ste na ne nezabudli.

Pod pojmom **popis algoritmu** *nerozumieme* preloženie programu z Pascalu, prípadne C-čka do slovenčiny. Popis by mal zhruba obsahovať: **Popis myšlienky**, na ktorej je založený algoritmus (bez detailov ako sú názvy premenných, procedúr...). **Popis dátových štruktúr**. Čo si vlastne chceme počas výpočtu pamätať a aké štruktúry sú na to vhodné. **Popis algoritmu**. Tu môžeme využívať už popísané dátové štruktúry a spomenúť dôležité procedúry a funkcie, ako aj najdôležitejšie premenné. **Zdôvodnenie správnosti**. Nemusí to byť formálne presný ani detailný dôkaz, avšak mal by obsahovať argumenty zdôvodňujúce každý dôležitý aspekt programu, ktorý nie je na prvý pohľad zrejmý. Sem zaradíme aj dôkaz konečnosti v tých prípadoch, keď nie je zrejmé, že sa program (resp. niektorá operácia, ktorú vykonáva) nezacyklí. Na koniec príde **odhad** časovej a pamäťovej zložitosti programu.

Takáto štruktúra popisu nie je univerzálna. Niektoré časti možno vynechať, niektoré zlúčiť, prípadne nejaké pridať. Poradie by však malo byť zhruba zachované. Popis by sa v žiadnom prípade nemal detailne venovať spracovávaniu vstupu, ani výpisu výstupu.

Niekoľko slov o odhade zložitosti. Pod odhadom časovej zložitosti rozumieme odhad času, ktorý bude program trvať, v závislosti od vstupných premenných. Tento čas je priamo úmerný počtu elementárnych operácií, ktoré program vykonáva – priradenie jednoduchej premennej, porovnanie jednoduchých premenných, aritmetické operácie, atď. Väčšinou sa zaujímame o to, ako dlho beží program v priemernom, alebo najhoršom prípade (čiže si všímame priemerný, alebo maximálny čas vykonávania programu). Analogicky odhad pamäťovej zložitosti je odhad veľkosti použitej pamäti v bajtoch v závislosti od vstupných premenných.

Väčšinou nás nezaujímajú presné hodnoty (niekto má rýchlejší počítač, na niektorých počítačoch má integer 2 a na iných 4 bajty a pod.), preto používame tzv. O-notáciu. Hovoríme, že funkcia $f(n)$ (napr. čas behu programu v milisekundách v závislosti od veľkosti

vstupu n) je $O(g(n))$, ak existuje taká konštanta c, že pre všetky dostatočne veľké n ($n > n_0$) je $f(n) \leq c \cdot g(n)$.

Napríklad, ak program pre vstup veľkosti n nebeží nikdy viac ako $T(n) = 0.5n^3 + 5n \log n + 83$ mikrosekúnd, môžeme povedať, že má časovú zložitosť $O(n^3)$. Ak program potrebuje najviac $M(n) = 3n \log_2 n + 150n + 15$ bajtov pamäti, povieme, že má pamäťovú zložitosť $O(n \log n)$. [1]

Viacero príkladov odhadu časovej i pamäťovej zložitosti nájdete, ak si pozorne prečítate vzoráky.

A ešte jedno upozornenie pre **vás** – riešiteľov: uvedením nepostačujúceho popisu prípadne jeho vynechaním riskujete, že opravovateľ vaše riešenie nepochopí a dostanete napr. 0 (slovom **nula**) bodov.

1. Zo života informačnej kancelárie

opravoval Braňo
(max. 15 bodov)

Dnes je opäť hmla... ale nie o tom som chcel. Vaše riešenia ohodnotiť treba. Väčšina z nich fungovala, len boli trochu pomalé. Podľa času potrebného na jednu operáciu som ich hodnotil takto:

- časová zložitosť $O(1)$ – 13 bodov
- časová zložitosť $O(N)$ – 8 bodov
- nefunkčné riešenia – max 3 body
- popis a elegantnosť programu – 2 body

Na úvod chcem upozorniť, že si môžete formát vstupu trochu pozmeniť tak, aby sa vám vstup lepšie načítaval.

Najväčšia časť vašich riešení bola zaťažená sledovaním podobného úkazu v reálnom svete. Ľudia boli uložení v poli. Keď prišiel nový občan, zaradil sa do radu. Keď sa okienko uvoľnilo, posunul sa celý rad dopredu. A práve tento jav spomaľuje, lebo čím viac ľudí bude stáť v rade, tým dlhšie táto operácia bude trvať. Preto nebudeme posúvať ľudí v rade, ale rad s ľuďmi.

Vzorové riešenie: Vysvetlíme si princíp činnosti na jednom okienku. Pred okienkom stojí niekoľko ľudí. Títo ľudia tvoria rad. A čo od takého radu požadujeme?

- pridať človeka na koniec
- vybrať človeka zo začiatku
- zistiť, či je rad prázdny

Tieto operácie poskytuje dátová štruktúra fronta (ktorá sa nazýva aj rad alebo FIFO[2]). Fronta sa dá implementovať (t.j. napísať v programovacom jazyku) rôznymi spôsobmi, pričom efektívne sú tieto dva:

1. Majme pole A. Budeme si pamätať indexy na začiatok a koniec fronty v A. Začiatok ukazuje na prvý údaj fronty, koniec ukazuje na prvý voľný údaj za frontou. Jednotlivé operácie sa napíšu takto:

- *pridaj(x)* : `A[koniec]:=x` a posuň koniec o jedna.
- *vyber*: vráť `A[zaciatok]` a posuň začiatok o jedna.
- *prázdnosť* : fronta je prázdna, ak `zaciatok = koniec`.

[1] Poznamenávame, že ak je nejaká funkcia $O(n^2)$, tak je aj $O(n^3)$ atď. Vždy sa však snažíme nájsť čo najmenšie horné ohraničenie, t.j. funkciu, ktorá čo najpomalšie rastie.

[2] FIFO = First In, First Out

Takto budeme frontu posúvať ďalej a ďalej, až raz prídeme na koniec poľa A. A čo teraz? Jednoducho pokračujeme od začiatku poľa – `if koniec>max then koniec:=1`. A to isté aj so začiatkom. Pokiaľ sa nebudeme snažiť vložiť do fronty aspoň toľko položiek, ako má pole A prvkov, všetko funguje.

2. Majme spájaný zoznam (predpokladám, že vám to niečo hovorí). Budeme si pamätať pointer na prvú položku fronty a na poslednú.

- *pridaj(x)* : vytvor nový prvok, ulož do neho x, zaraď za posledného a urob ho posledným.
- *vyber* : vráť hodnotu z prvého prvku, prvok za ním urob novým prvým a uvoľni pamäť
- *prázdnosť* : fronta je prázdna, ak je prázdny spájaný zoznam

Ostatné: Obe riešenia majú svoje výhody a nevýhody. Prvé sa jednoduchšie napíše, avšak musíme vedieť, koľko ľudí bude stáť maximálne v rade. V druhom so spájaným zoznamom treba dynamicky alokovať pamäť – ibaže by sme použili malú fintu.

Pre každého človeka si budeme v poli pamätať index na človeka stojaceho za ním. Keď to bude 0, tak za ním nikto nestojí. Tým sa zbavíme dynamickej alokácie pamäte. Má to však jednu nevýhodu – jeden človek môže stáť súčasne len v jednom rade.

Odhad zložitosti: Spracovanie každej operácie (`prišiel`, `uvoľnilo sa`) nezávisí od veľkosti vstupu, preto časová zložitosť jednej operácie je $O(1)$. Označme si max. počet ľudí v jednom rade n, počet všetkých ľudí l a počet okienok o. Potom bude pamäťová zložitosť prvého riešenia $O(o \cdot n)$ a druhého $O(l + o)$.

Listing programu:

```
program Zo_zivota_informacnej_kancelarie;
const max_okien= 20;    {maximalny pocet okien {}
      max_ludi = 10000; {maximalny pocet ludi (aj cisel ludi){}

var ludia    : array[1..max_ludi] of integer;
    zac,kon  : array[1..max_okien] of integer;
    c:char;

procedure prisiel;
var ob,ok:integer;         { ob - obcan, ok - okienko {}
begin
  repeat
     read(c);
  until c=' ';             {precitaj string PRISIEL{}
  readln(ob,ok);

  if zac[ok]=0 then begin  {prazdne okienko {}
    zac[ok]:=ob;             {novy je na zaciatku{}
    kon[ok]:=ob;             {aj na konci{}
    writeln('Obcan ',ob,' k okienku ',ok);
  end else begin           {pridaj obcana do radu {}
    ludia[kon[ok]]:=ob;      {zarad noveho za posledneho{}
    kon[ok]:=ob;             {novy koniec radu{}
  end;
end;

procedure uvolnilo;
var ok,x:integer;         {ok-okienko{}
begin
  x:=0;
  repeat
    read(c);
```

```
    if c=' ' then inc(x);
  until x=2;              {precitala sa druha medzera stringu UVOLNILO SA}
  readln(ok);

  x:=zac[ok];             {stary zaciatok{}
  zac[ok]:=ludia[x];      {novy zaciatok - clovek stojaci za prvym{}
  ludia[x]:=0;            {kedze uz nie je v rade, nik za nim nestoji{}
  if zac[ok]<>0 then begin  {stoji este niekto v rade{}
    writeln('Obcan ',zac[ok],' k okienku ',ok);
  end;
end;

begin
  fillchar(ludia,sizeof(ludia),0);
  fillchar(zac,sizeof(zac),0);
  fillchar(kon,sizeof(kon),0);
  repeat
    read(c);
    case upcase(c) of
      'P' : prisiel;
      'U' : uvolnilo;
    end;
  until upcase(c)='K';
end.
```

Listing programu:

```
program Zo_zivota_informacnej_kancelarie;
const max_okien= 20;    {maximalny pocet okien {}
      max_rad = 100;    {maximalny pocet ludi v rade(aj cisel ludi){}

var A        : array[1..max_okien, 0..max_rad] of integer;
    zac,kon  : array[1..max_okien] of integer;
    c:char;               { kon[i] - ukazuje na prve volne miesto{}

procedure prisiel;
var ob,ok:integer;         { ob - obcan, ok - okienko {}
begin
  repeat
     read(c);
  until c=' ';             {precitaj string PRISIEL{}
  readln(ob,ok);

  if kon[ok]=zac[ok] then  {nikto nestoji v rade, tak moze ist k oknu{}
    writeln('Obcan ',ob,' k okienku ',ok);

  A[ok][kon[ok]]:=ob;      {uloz noveho cloveka{}
  inc(kon[ok]);            {koniec radu sa posunie{}
  if kon[ok]>max_rad then
    kon[ok]:=0;
end;

procedure uvolnilo;
var ok,x:integer;         {ok-okienko{}
begin
  x:=0;
  repeat
    read(c);
    if c=' ' then inc(x);
  until x=2;              {precitala sa druha medzera stringu UVOLNILO SA}
```

```
  readln(ok);

  inc(zac[ok]);           {koniec radu sa posunie{}
  if zac[ok]>max_rad then
    zac[ok]:=0;
  if zac[ok]<>kon[ok] then
    writeln('Obcan ',A[ok,zac[ok]],' k okienku ',ok);
end;

begin
  fillchar(A,sizeof(A),0);
  fillchar(zac,sizeof(zac),0);
  fillchar(kon,sizeof(kon),0);
  repeat
    read(c);
    case upcase(c) of
      'P' : prisiel;
      'U' : uvolnilo;
    end;
  until upcase(c)='K';
end.
```

opravovala Ňaňka
(max. 15 bodov)

2. Zaoceánske krížniky, škunery a plavby

Tento príklad vôbec nebol náročný. Myslím, že po krátkom zamyslení sa mohol každý poslať aspoň nejaké riešenie. Tie vaše by sa dali rozdeliť asi takto:

- vzorové riešenia používajúce maticu susednosti – 15 bodov
- riešenia, ktoré pre každý prístav majú zoznam s ním spojených prístavov – 12 bodov
- riešenia ošetrujúce násobnosť liniek kontrolovaním už načítaných – 9 bodov
- nefunkčné riešenia – 1–3 body
- za nejaké drobné chyby ste mohli stratiť nejaký ten bod.

„Asi najväčším problémom bol vstup.“ dozvedela som sa z jedného riešenia. Tu je asi namieste poznamenať, že u väčšiny z vás tomu tak bolo. Ale nemuselo byť. Príklad vstupu v zadaniach je písaný tak, aby z neho bolo jasné, čo vlastne chceme. Formát vstupu si každý môže spraviť aký chce. Napríklad načítavať linky ako dvojice čísel a predpokladať, že na konci je dvojica 0 0.

Na rozdiel od vás, pre mňa boli najväčšími problémami neprehľadné a hlavne slabo alebo vôbec nepopísané programy. Medzi tieto rátam aj niekoľko programov so siahodlhými popismi s maličkou vadou: popis bol iba prekladom zdrojáku do slovenčiny. Toľko zvládneme k vášmu programu napísať aj my. V popise má byť hlavne idea (ako vás vôbec napadlo to takto robiť), dôkaz alebo zdôvodnenie správnosti (prečo by to vlastne malo fungovať) a nejaký odhad zložitosti (pozri úvod letáku). Samozrejme, chýbajúci alebo slabý popis vás mnohých stál nejaké tie bodíky.

Ďalšou chybou, s ktorou som sa často stretávala, bolo nevypísanie všetkých prístavov s maximálnym počtom liniek. Uznávam, táto chyba sa vyskytla aj vo vzorovom výstupe.[3] Pre vás by však mala byť záväzná špecifikácia úlohy, tj. to, čo sa nachádza v zadaniach za slovom „Úloha:“. A teda aj za túto chybu sa strhávali bodíky.

No a nemôžem nespomenúť opisovanie. Keď prídu z jednej školy riešenia, ktoré sú proti svetlu rovnaké, alebo majú úplne identický popis, či chýbajúci znak v zdrojáku, musím deliť body počtom členov kolektívu.

[3] Neradi si to priznávame, ale občas nám prekĺzne nejaká tá chybička

A teraz k vzorovému riešeniu. Hlavnou myšlienkou bolo, že rovno pri načítaní si budeme linky ukladať do dvojrozmerného poľa A – matice. To, že každá linka sa tam uloží iba raz, je zabezpečené tým, že vždy pri načítaní linky uložíme na miesto tú istú hodnotu. Samozrejme, keďže linky sú obojsmerné, budeme si každú linku zapisovať aj v opačnom smere. Znamená to, že pri načítaní linky z i do j priradíme $A[i,j]$ aj $A[j,i]$ našu zvolenú hodnotu (najlepšie jednotku alebo true). Po skončení načítania už len sčítame, koľkokrát sa v jednotlivých stĺpcoch nachádza naša hodnota. Z toho spravíme maximum a nájdeme všetky prístavy, kde je počet liniek rovný tomuto maximu.

Časová zložitosť je $O(l+n^2)$, kde l je počet liniek. Za predpokladu, že úradníci sú naozaj sklerotickí a že počet načítavaných liniek l je výrazne viac ako n^2, je teda časová zložitosť $O(l)$.

Pamäťová zložitosť je kvadratická ($O(n^2)$), pretože používame jedno dvojrozmerné pole.

Listing programu:

```
program z1812;
const nmax=200;
var A:array[1..nmax,1..nmax] of 0..nmax; {matica liniek medzi prístavmi}
    i,j,k,n,pom,max:0..nmax;

begin
 write('Pocet pristavov: '); readln(n);
 for i:=1 to n do                    {inicializácia}
  for j:=1 to n do A[i,j]:=0;
 pom:=0; max:=0;
 writeln('Linky:'); readln(i,j);     {načítanie, ukončené načítaním 0,0}
 while (i<>0) and (j<>0) do
  begin
   A[i,j]:=1; A[j,i]:=1;
   readln(i,j);
  end;
 for i:=1 to n do                    {zisťovanie maxima}
  begin
   for j:=1 to n do pom:=pom+A[i,j];
   if pom>max then max:=pom;
   A[i,i]:=pom; pom:=0;
  end;
 writeln('Najviac liniek je ',max); {výpis všetkých, čo majú max liniek}
 writeln('Maximum je v pristave/pristavoch');
 for k:=1 to n do
  if A[k,k]=max then write(k,' ');
end.
```

3. Mínus dvojková sústava

opravoval YoYo
(max. 17 bodov)

Tak tento priklad bol podľa niektorých riešiteľov najťažší, aj keď rozsahom bol možno najmenší. Niektorí z vás si ale nevšimli, že má dve časti, a to nielen previesť číslo z desiatkovej sústavy do mínus dvojkovej, ale aj naopak, presne podľa vzorca v zadaní.

Tááák, poďme pekne k bodovaniu:

- Prevod z mínus dvojkovej sústavy do desiatkovej – 7b
- Prevod z desiatkovej do mínus dvojkovej – 8b

- Dôkaz jednoznačnosti zápisu čísla v mínus dvojkovej sústave; túto časť malo len málo riešení – 2b

V prvej časti sa ešte strhávali 2b za zhoršenú časovú zložitosť (teda $O(\log^2 N)$ namiesto $O(\log N)$). No a samozrejme za ďalšie chybičky ste prišli o ďalšie body.

Poďme ale k vzorovému riešeniu. Jedna možnosť je postupovať presne podľa zadania. Vždy cifru vynásobíme príslušnou mocninou -2 a pripočítame k nášmu číslu. Netreba ale mocninu -2 počítať zakaždým odznova (čo je prípad riešení zo zložitosťou $O(\log^2 N)$). Stačí ak si najprv mocninu nastavíme na 1, začneme číslo spracovávať odzadu a zakaždým mocninu prenásobíme -2. Existuje ale ešte jeden spôsob, ktorým môžeme číslo spracovávať odpredu a nepotrebujeme rátať mocniny. Stačí si náš výraz zo zadaní trošku upraviť:

$$x_{n-1}\,(-2)^{n-1} + x_{n-2}\,(-2)^{n-2} + \ldots + x_0\,(-2)^0 = x_0 + (-2)\,(x_1 + (-2)\,(\ldots(x_{n-2} + (-2)\,(x_{n-1}))))$$

Algoritmus už bude teraz jednoduchý. Najprv vezmeme prvú cifru čísla v mínus dvojkovej sústave (x_{n-1}). A teraz pokračujeme jednoducho. Kým ešte máme nejaké cifry, tak naše číslo vynásobíme -2 a pripočítame ďalšiu cifru.

Prevod z desiatkovej sústavy do mínus dvojkovej je už trochu ťažší (možno preto, že v zadaní nebol návod...). Zase použijeme náš upravený vzorec z predchádzajúcej časti. Vieme teda, že naše číslo sa dá napísať ako $c = x_{n-1} + (-2) \cdot z_1$, kde c je naše číslo a z_1 je nejaké iné čislo, pričom určite $x_{n-1} < 2$, teda vlastne x_{n-1} je zvyšok po vydelení čísla c mínus dvomi. Ako dostaneme ďalšiu cifru? No keď vydelíme c bez zvyšku číslom -2, dostaneme obsah prvej zatvorky v predchadzajúcej časti, teda $x_{n-2}+(-2)\cdot z_2$. No a pokračujeme presne rovnako, až kým nedostaneme nulu. V pascale to má ešte malý háčik, totiž funkcie `div` a `mod` fungujú tak, že $(-5)/(-2) = 2$ zv -1. My ale potrebujeme aby zvyšok bol 0 alebo 1, teda $(-5)/(-2) = 3$ zv 1. Doteraz uvedené algoritmy fungujú pre sústavu s ľubovoľným základom. Samozrejme nie je ťažké vysporiadať sa aj s týmto posledným problémom tak, aby to stále fungovalo. Pre jednoduchosť sa tu ale obmedzíme vo vzorovom riešení iba na mínus dvojkovú sústavu[4].

No a ešte dôkaz, že ľubovoľné celé číslo sa dá v mínus dvojkovej sústave napísať práve jedným spôsobom. Dokážeme to sporom. Predpokladajme teda, že existujú dve čísla c_1 a c_2, $c_1 \neq c_2$. Nech $c_1 = (a_n a_{n-1} \ldots a_1 a_0)_{(-2)}$ a $c_2 = (b_m b_{m-1} \ldots b_1 b_0)_{(-2)}$. Bez ujmy na všeobecnosti predpokladjme, že $n \geq m$. Ak $n > m$, tak doplňme $b_n = 0, b_{n-1} = 0, \ldots b_{m+1} = 0$. Teraz nech $q \leq n$ je najväčšie také číslo, že $a_q \neq b_q$. Teda, že prvá cifra, v ktorej sa tieto dve čísla líšia, je q. Cifry pred ňou nás nemusia zaujímať, ich vynechaním dostaneme čísla d_1 a d_2, $d_1 = d_2$, ktorých zápis v mínus dvojkovej sústave je rôzny (konkretne sa určite líšia prvou cifrou). Vieme ale, že $d_1 - d_2 = 0$, teda

$$\left(a_q(-2)^q + a_{q-1}(-2)^{q-1} + \ldots + a_0(-2)^0\right) - \left(b_q(-2)^q + b_{q-1}(-2)^{q-1} + \ldots + b_0(-2)^0\right) = 0$$

$$(a_q - b_q)\,(-2)^q + (a_{q-1} - b_{q-1})\,(-2)^{q-1} + \ldots + (a_0 - b_0)\,(-2)^0 = 0$$

Vieme, že $a_q \neq b_q$. To znamená, že jedno z a_q, b_q je jednotka a druhé nula. Ich rozdiel je teda 1 alebo -1. Teda $|(a_q - b_q)\,(-2)^q| = 2^q$. Zoberme si zvyšok ľavej strany, ktorý je $\sum_{i=0}^{q-1} \left|(a_i - b_i)(-2)^i\right| \leq \sum_{i=0}^{q-1} 2^i = 2^q - 1$. Nerovnosť vychádza z faktu, že $|a_i - b_i| \leq 1$. Prvý sčítanec našej rovnice je teda buď 2^q alebo -2^q. Absolútna hodnota zvyšku ľavej strany je ale bohužiaľ menšia alebo rovná ako $2^q - 1$ [(5)]. To znamená,že ich súčet bude určite rôzny od nuly, čo je náš hľadaný spor. □

[4] Aj keď vo vzoráku je aj riadok, ktorý funguje pre sústavu s temer ľubovoľným základom (aj záporným).

[5] Toto ešte nie je take jasné, treba si uvedomiť napríklad, že $|a + b| \leq |a| + |b|$

Listing programu:

```
Program Zabudnute_cisla;
Const Base=-2;

var Cislo1:String;
    Cislo2:LongInt;


Function Do10(c:string):longint;
var i,b,d:integer; a:longint;
Begin
  Val(C[1],a,d);;
  For I:=2 to Length(c) do Begin
    Val(C[i],b,d);
    a:=a*Base;
    a:=a+b;
  end;
  Do10:=a;
end;


Function Z10(c:longint):String;
var i,j:integer; s,s1:string;
Begin
  S:='';
  while c<>0 do begin
    If c mod (-2) = -1 then Begin
      S:=S+'1';
      c:= (c-1) div (-2);
    end else Begin
      Str(c mod (-2),S1);
      S:=S+S1;
      c:=c div (-2);
    end;
(*     {Tu su tri riadky pre tych, co by chceli univerzalny algoritmus }
       { pre ABS(Base)<=10, to kvoli oznac. cislic (11=A atd.)}
    Str(Abs(c mod Base),S1);
    S:=S+S1;
    c:=(c - abs (c mod Base) ) div Base;
*)
  end;
  If S='' then S:='0';
  S1:=S;
  J:=Length(S);
  For I:=1 to J do S1[J-I+1]:=S[I];
  Z10:=S1;
end;


Begin
  Write('Zadaj cislo v minus dvojkovej sustave: ');
```

```
  Readln(cislo1);
  Writeln('V desiatkovej je to: ',Do10(Cislo1));
  Writeln;

  Write('Zadaj cislo v desiatkovej sustave: ');
  Readln(cislo2);
  Writeln('V minus dvojkovej sustave je to: ',Z10(Cislo2));
  Writeln;

end.
```

4. Zauzlení študenti

opravoval JanoVah
(max. 15 bodov)

Tento príklad riešilo pomerne veľa riešiteľov, okolo 70. Vaše riešenia sa dajú zhrnúť asi do dvoch kategórií:

- „vzorové“ riešenia (v čase $O(n)$). Tých bolo asi najviac, čo svedčí o tom, že príklad bol pre vás pomerne ľahký – max 15 bodov
- riešenia založené na myšlienke vzorového riešenia, avšak s nešťastnými zmenami, ktoré ich robia ich časovú zložitosť kvadratickú ($O(n^2)$) – max 12 bodov

V podstate všetky riešenia boli založené na rovnakej myšlienke ako naše vzorové, až na niektoré detaily, ako je spôsob označovania študentov, prechádzania poľa a podobne. Hoci sme od vás chceli, aby ste našli iba počet kruhov, mnohí z vás sa snažili aj vypisovať študentov v jednotlivých kruhoch. Oceňujem vašu snahu, ale chcel by som vás upozorniť na to, že niekedy sa takýto „rozšírený“ problém nedá riešiť tak efektívne ako ten pôvodný (aj keď v našej úlohe výpis študentov v jednotlivých kruhoch efektivitu nepokazí).

K bodovaniu. Za kvadratickú zložitosť som strhával 3 body. Za nejasný, alebo žiadny popis som strhával 2 body. Pôvodne som chcel strhávať body aj za chýbajúci odhad zložitosti, ale nakoniec som si to rozmyslel. No a teraz už dosť rečí a hor sa do práce.

Základná myšlienka nášho algoritmu je táto: Budeme prechádzať jednotlivými študentmi. Ak narazíme na niekoho, kto ešte nepatrí do nijakého kruhu, zvýšime počet kruhov a celý kruh označíme.

V našom programe N označuje počet študentov. V poli A je na i-tom mieste (označíme ho A_i) číslo študenta, ktorého drží študent i pravou rukou. i-teho študenta budeme považovať za označeného (už patrí do nejakého kruhu), ak $A_i = 0$. Pri tomto si treba uvedomiť, že u označeného študenta nám už nezáleží na tom, koho sa drží, pretože už ho nebudeme nijak ďalej spracovávať. Jediné, čo nám ešte zostáva vyriešiť, je ako označiť celý kruh. Nech teda i je študent, ktorý ešte nie je označný. Označíme ho a presunieme sa na jeho suseda A_i. Jeho sused nemôže byť ešte označený, lebo by bol označený celý kruh, teda aj študent i. Označíme aj jeho a opäť sa presunieme na jeho suseda. V prípade, že narazíme na nejakého označeného študenta, skončíme. Tento študent je určite prvý označený študent v tomto kruhu, [6] teda študent i.

No a teraz k odhadu zložitosti. Ide o odhad, „koľko inštrukcií“ potrebuje program vykonať vzhľadom na N. Keď si pozriete vzorový program, ľahko uvidíte, že na načítanie vstupu je treba $O(N)$ inštrukcií. Vnútro ďalšieho cyklu `for` sa vykoná N-krát. Otázkou zostáva, koľkokrát sa môže vykonať vnútro cyklu while. Ak by sme to odhadli na najviac N-krát v každom z prechodu cyklu `for`, dostávame výslednú zložitosť $O(N^2)$[7]. My však vieme, že môžeme označiť nanajvýš N študentov. Preto hoci je `while` v cykle, jeho vnútro sa môže

[6] Rozmyslite si prečo.

[7] Toto je tiež správny odhad zložitosti programu, ale nie je to najtesnejší odhad. Pozri úvod letáku.

spolu vykonať nanajvýš N-krát, nezávisle od toho, koľkokrát sa vykoná vnútro cyklu `for`. Teda zložitosť nášho algoritmu je $O(N)$.

No a teraz už vzorový program:

Listing programu:

```
program studenti;

var
  N,i,j,p : integer;
  A : array[1..20] of integer;

begin
  { nacitanie vstupnych udajov }
  write('Zadaj pocet studentov: '); readln(N);
  for i := 1 to N do begin
    write('Koho drzi student c.',i,': ');
    readln(A[i]);
  end;
  p := 0;
  for i := 1 to N do begin
    { neoznaceny student este nepatri do ziadneho kruhu }
    if (A[i] <> 0) then inc(p);
    { prejdeme cely kruh a oznacime ho 0-ou }
    while (A[i] <> 0) do begin
      j := i;
      i := A[i];
      A[j]:=0;
    end;
  end;
  writeln('Pocet kruhov: ',p);
end.
```

5. Zeofína sa vracia

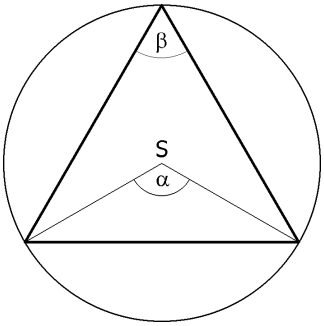
opravovala Meri
(max. 15 bodov)

Zeofínka sa vašim programom potešila, a potom sa s čistou mysľou pobrala do večných lovíšť. Zostalo po nej len pár otázok a stôp v piesku. Nuž, ako sa bodovalo?

- Úplne správne riešenie – 14–15 bodov
- Funkčné riešenie, Zeofínka sa raz presunie – 11–13b
- Funkčné riešenie, Zeofínka potrebuje pole – 8–10b
- Riešenie nefungujúce pre zložené hviezdičky – 6–7b
- Ešte nefunkčnejšie riešenia – max 5b

No, a ako sa to malo riešiť?
Hviezdičky sa dajú rozdeliť do troch skupín: hviezdičky s nepárnym počtom vrcholov, hviezdičky s počtom vrchlov deliteľným 4 a tie ostatné. Označme počet vrcholov hviezdičky n.

Nech má hviezdička nepárny počet vrcholov. Potom sú každé dva susedné spojené cez jeden „skoro opačný“ vrchol. Ak teda začneme z ľubovolného vrcholu, tak sa vieme dostať do jemu susedného vrcholu, odtiaľ do ďalšieho susedného atď., až kým neobeháme celú hviezdičku a vrátime sa späť. Takže takúto hviezdičku vieme nakresliť jedným ťahom. Uhol



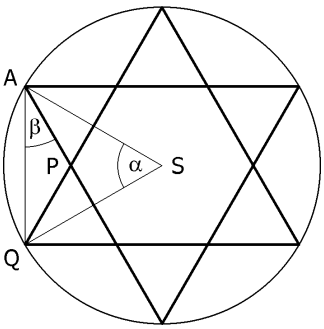
obr. 1

$\alpha = \frac{360^o}{n}$ a kedže $\beta = \frac{\alpha}{2}$ (kvôli vete o stredových a obvodových uhloch), $\beta = \frac{180^o}{n}$ (pozri obr. 1).

Takúto hviezdičku vieme teda nakresliť pomerne jednoducho, stačí n-krát nakresliť rovnú čiaru a potom otočíme korytnačku o $180 - \frac{180}{n}$ stupňov.

Pozrime sa teraz na hviezdičky s párnym počtom vrcholov. Vtedy $\alpha = \frac{360^o}{n/2}$ a teda $\beta = \frac{360^o}{n}$. Najprv si ich vrcholy očíslujme od 1 po n. Z ľubovoľného vrcholu i sa vieme jedným presunom dostať do vrcholu $\left(i + 2 + \frac{n-2}{2}\right) \bmod n$. Ak n nie je deliteľné 4, tak sa z nepárneho vrcholu dostaneme zasa len do nepárneho, a teda hviezda sa skladá z dvoch podhviezd (jedna obsahuje len nepárne vrcholy a druhá len párne vrcholy).

Tieto podhviezdy sa na niekoľkých miestach pretínajú, takže nám stačí nakresliť kúsok prvej hviezdy, otočiť sa, nakresliť druhú hviezdu a potom už len dokresliť tú prvú hviezdičku. Ale ako ďaleko od cípu je spojnica tých dvoch hviezdičiek? (pozri obr. 2)



obr. 2

$\alpha = \frac{360^o}{n}$
$\beta = 90^o - \frac{360^o}{n}$
$|QS| = |AS| = r$, pričom r je polomer kruhu opísaného hviezdičke

Vďaka kosínusovej vete si vieme spočítať $|QA| = \sqrt{2|SA|^2(1 - \cos(360^o/n))}$. V trojuholníku AQP zase z kosínusovej vety vieme vypočítať, že $|QP| = \frac{|QA|}{2\cos\beta}$. Uhol pri vrchole P zase z kosínovej vety je $\cos\gamma = \frac{2|PA|^2 - |QA|^2}{2|PA^2|}$. Pretože $\arccos x = \operatorname{arctg}\sqrt{\frac{1-x^2}{x}}$ [8], možeme vypočítať aj uhol γ a pustiť sa do písania...

Ak je počet vrcholov hviezdičky deliteľný 4, z vrcholu i sa dostaneme do vrcholu $(i + 2 + \frac{n-2}{2}) \bmod n$ (čo je číslo s opačnou paritou ako i), odtiaľ do vrcholu $(i + 2) \bmod n$ (čo má zase rovnakú paritu ako i) atď., až kým neobeháme celú hviezdičku. Teda takáto hviezdička sa dá nakresliť jedným ťahom tak, že korytnačka n-krát nakreslí čiaru a otočí sa o $180 - \frac{360}{n}$ stupňov.

[8] Ak si tento vzorec neviete zapamätať ani odvodiť, je napísaný v Pascalovskom helpe pre arctg.

Listing programu:

```
program korytnacka;
uses crt;
const
  trasa=10;
var
  N:integer;
  uhol:real;
  r:real;

procedure Kresli(kolko,cipov:integer);
var
 uhol2:real;
 mensia_cast,ciara:real;
 i:integer;
begin
  dec(kolko);

  ciara:=sqrt(2*sqr(r)*(1-cos(360/n)));
  mensia_cast:=ciara/2*(cos(360/n));

  writeln('DOPREDU ',mensia_cast:2:2);

  if kolko=1 then begin
    uhol2:=(2*sqr(mensia_cast)-sqr(ciara))/(2*sqr(mensia_cast));
    uhol2:= ArcTan (sqrt (1-sqr (uhol)) /uhol);
    writeln('DOLAVA ',uhol2:2:2);
    Kresli(kolko,cipov);
  end;

  if (kolko=0)and(cipov=n) then writeln('DOPREDU ',trasa-mensia_cast :2:2);
  if kolko=1 then writeln('DOPREDU ',trasa-mensia_cast:2:2);
  for i:=1 to cipov-1 do begin
    writeln('DOPRAVA ',uhol:2:2);
    writeln('DOPREDU ',trasa);
  end;

  if (kolko=0)and(cipov<>n) then begin
    writeln('DOPRAVA ',uhol:2:2);
    writeln('DOPREDU ',trasa-mensia_cast:2:2);
    writeln('DOPRAVA ',uhol2:2:2);
  end;
end;

begin
  write('Kolko cipov ma mat hviezda? ');
  readln(N);

  r:=sqrt(sqr(trasa)/2*(1-cos(90-360/n)));

  writeln('POLOZ CHVOSTIK');
```

```
  if n mod 2= 0 then uhol:=180-360/N
    else uhol:=180-180/N;
  if n mod 4 = 2 then Kresli(2,n div 2)
    else Kresli(1,n);
  writeln('ZDVIHNI CHVOSTIK');
  readkey;
end.
```

Vsledkov listina po 1. srii kateg¡=rie KSP

	Meno a priezvisko		řkola	11	12	13	14	15	Σ
1.	Mravec Pavol	3	Gym. K.tra Modra	15	15	16	15	14	75
2.	Koula JiŠ		.....	14	15	15	15	14	73
	Katreni Jn	3B	Gym. Spi. Nov Ves, kolsk	15	15	15	15	13	73
4.	Vozr Oto	2	Gym. Matky Alexie Bratislava	15	15	14	15	8	67
5.	TruchlÀ Peter	3	Gym. Nitra, Provsk 1	10	12	15	15	13	65
	Choma Martin	2	Gym. Star -ubova	10	15	15	15	10	65
7.	Hanakovi Tom	4A	Gym. sv.Frantika ilina	10	15	11	15	13	64
	Steinov Monika	3A	Gym. Bratislava, Einsteinova	10	15	11	15	13	64
9.	Havlj Frantiek	4	.....	15	15	8	15	10	63
10.	rmek Ras»o	3A	Gym. Bratislava, Tilgnerova	10	15	15	15	7	62
	Teke Michal	3E	Gym. Preov, Kontantnova	14	15	9	15	9	62
12.	omlo Ivan	1	Gym. ahy, Mldencka	10	9	14	15	12	60
13.	Novk Andrej	3B	Gym. ilina, Hlinsk	15	15	12	15	2	59
	Palenr Jan		Gym. V.Paulnyho-T Martin	14	15	15	15		59
15.	Minrik Gabriel		SP Komrno, Petofiho	10	14	8	12	13	57
	Kavka Michal	3B	Gym. Koice, Alejov	10	12	15	7	13	57
17.	Rusnk Pavol	3E	Gym. Preov, Kontantnova	10	15	7	12	10	54
18.	Topor Peter	4	Gym. Povask Bystrica	10	9	7	14	13	53
	Karas Filip	1B	Gym. Nitra, Golianova 68	9	8	11	14	11	53
20.	Baran P	2B	Gym. J. Hronca Bratislava	10	8	13	15	6	52
21.	Trenkler Pavol	1A	Gym. Koice, Zbrojnin	14	12		12	12	50
	Frlika Martin	2B	Gym. J. Hronca Bratislava	10	13	9	12	6	50
23.	Kansz Rbert	2	Gym. Koice, Dnepersk	8	15	13	13		49
	LuanskÀ Jn	3B	Gym. Trebiov, Komenskho	9	15	13	12		49
	Ivan Michal	3B	Gym. Nedoerskho Prievidza	10	14	5	15	5	49
26.	bodk Michal	4	Gym. Povask Bystrica	10	15		15	8	48
	Libi Peter	3C	Gym. -.tra, Trenn	10	9	12	10	7	48
	Kvasnika Martin	3A	Gym. A.Markua Bratislava	10	13	7	15	3	48
	Jirsek Joko	6	Gym. Koice, Zbrojnin	10	15	4	15	4	48
30.	Studva Tom	3	Gym. A.Markua Bratislava	10	14	3	15	5	47
31.	Pirchala Martin	2B	Gym. Koice, Alejov	10	2	15	7	11	45
32.	Klimo Peter	3B	Gym. Bratislava, Hubenho	1	8	11	13	11	44
33.	tefanek Anton	1C	Gym. J. Hronca Bratislava	8	11	11	13		43
34.	Urban Jaroslav	3A	Gym. LiptovskÀ Hrdok	10	9		15	7	41
	Mark Lszl	2	Gym. H.Selyeho ma. Komrno		15	12	10	4	41
	KonenÀ tefan	2D	Gym. J. Hronca Bratislava	9	8	9	10	5	41
37.	VeselÀ Jn	1B	Gym. Nitra, Golianova 68	7	7	11	13		38
38.	Burda Milan	1B	Gym. Nitra, Golianova 68	8	8	10	10		36
	Vranec Maro	2A	Gym. Koice, Alejov		15	15		6	36
40.	Szarkov Eva	4C	Gym. J. Hronca Bratislava	10	13		12		35
41.	Lovek Peter	4	Gym. Povask Bystrica	10	9		12	3	34
42.	Olejnk tefan	1A	Gym. J. Hronca Bratislava	9	5	11	0	8	33
43.	Vrbel Tom	3A	Gym. V.Paulnyho-T Martin	10	nn	nn	15	6	31
44.	Dovjak Marek	3	Gym. P.O.H. Kemarok	10	2	4	11	2	29
	ChudÀ Michal	3B	Gym. Levice	2	4	8	15		29
	Bal Miroslav	1A	Gym. J. Hronca Bratislava	8		11	10		29
47.	Kko Andrej	4D	Gym. Bratislava, Metodova		14		13		27
48.	Fila Michal	1B	Gym. Nitra, Golianova 68	7	7	11		1	26
49.	HodermarskÀ Ladislav	3A	SPE Star Tur	10	9			6	25
	Psztor Tom	4A	Gym. ahy, Mldencka			13		12	25

	Meno a priezvisko		řkola	11	12	13	14	15	Σ
51.	Schmotzer Marin	1C	Gym. Bratislava, I.Horvtha	10	8	4	2		24
	ulk Radovan	1B	Gym. Nitra, Golianova 68	2	8	13		1	24
	Lajdov Jana		Gym. ilina-Vlince	10		7		7	24
	ille Alexander	4B	Gym. ahy, Mldencka			13		11	24
	Luk Michal	3A	Gym. Koice, Alejov	1		13	10		24
56.	RakovskÀ Martin	3B	Gym. Nitra, Golianova 68			11		12	23
	Szab Zsolt	3	Gym. ahy, Mldencka			13		10	23
58.	imrk Martin	3B	Gym. Nitra, Golianova 68	5	2	11	2		20
	Gregor Rastislav	2E	Gym. ilina, Vek Okrun	8	5	0		7	20
60.	Jank Oliver	1C	Gym. L.Stockela Bardejov	15		4			19
61.	Krchniakov Korina		Gym. J. Hronca Bratislava		3		15		18
	Bartov Dominika	1B	Gym. J. Hronca Bratislava				13	5	18
	Chriateov -ubica	1B	Gym. J. Hronca Bratislava				13	5	18
64.	Demko Martin		Gym. Koice, Alejov	4		13			17
65.	Hutr Jn	2C	SPE Pie»any, SNP	10	6				16
	Kritofk tefan	2C	SP Levice, F.Heku		9		7		16
	Klnai Peter	3A	Gym. Levice			3	13		16
	Dojr Jn		Gym. Turianske Teplice	15	1				16
69.	Zachar Michal	2A	Gym. Koice, Trebiovsk	3	3	7		2	15
70.	Deknek Mat	1A	Gym. J. Hronca Bratislava	0		7		7	14
71.	JanovickÀ Stanislav	2C	SP Levice, F.Heku					13	13
	Marek Pavol	2B	Gym. A.Merici Trnava			7		6	13
	Vataha Martin	8	Gym. Snina	5	8				13
	Gergely Jakab	4A	Gym. ahy, Mldencka			13			13
	Heimlich Dezider	4A	Gym. ahy, Mldencka			13			13
76.	Zemk Luk	8	Gym. Poprad, Popradsk nbr.					11	11
	underlk Peter	1B	Gym. J. Hronca Bratislava		11				11
	Max Matej		Gym. Koice, Alejov	3	5	3			11
	Kocsis Pavol		Gym. Koice, Alejov	4	7				11
	Sihelnk Slavomr		Gym. Koice, Alejov					11	11
81.	Andrejko Maro	7	Z Koice, Park Angelinum		5			5	10
	Hantke Richard	9A	Z Koice, Charkovsk 1		5			5	10
	Viravec Martin	3B	Gym. Svidnk		10				10
	Kuis Jn		Gym. Koice, Alejov	4	2		4		10
	Kamr Jaroslav		Gym. Koice, Alejov	4	2		4		10
	Sabadosov Oga		Gym. Koice, Alejov	2	5	3			10
87.	Vojt Peter	9A	Z Koice, Charkovsk 1			4		5	9
	Molnr Tom	9A	Z Koice, Charkovsk 1		5	4			9
89.	Alvk Jakub	8	Gym. Snina		8				8
	Sipos Robert		Gym. Koice, Alejov		5	3			8
91.	volik Peter	2C	SP Levice, F.Heku				7		7
92.	Marko Peter		Gym. Koice, Alejov		2		4		6
	Tak Karol		Gym. Koice, Alejov	3		3			6
	Trejbal Ivan		Gym. Koice, robrova		nn			6	6
	Bal Daniel		Gym. Koice, Alejov					6	6
96.	OpaternÀ Marek	3A	Gym. Levice					5	5
	Sask Peter		Gym. Koice, Alejov			5	0		5
98.	Juliny Mrio		Gym. Pie»any, Nm.SNP 9	4	nn				4
	bert Tibor	3A	Gym. Levice					4	4
100.	Beck Patrik		Gym. .Moyzesa B.Bystrica	3					3
	Rojko Pavol		Gym. Koice, robrova	3	nn				3
	Jasa Maro	1B	Gym. Koice, robrova	3	nn				3
	Brilla Pavol	1D	Gym. Koice, robrova	3	nn				3

	Meno a priezvisko		řkola	11	12	13	14	15	Σ
	StrmenskÀ Tom		Gym. Koice, robrova	3	nn				3
105.	Pogny Peter	1B	Gym. H.Selyeho ma. Komrno		2				2
106.	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
110.	Sege Andrej	3B	Gym. Nitra, Golianova 68						0
	Harmao Dominik	8	Gym. Snina						0
	Kardy		Gym. Koice, robrova		nn				0