PRESEK
List za mlade matematike, fizike, astronome in računalnikarje

ISSN 0351-6652
Letnik **26** (1998/1999)
Številka 1
Strani 14–17

Martin Juvan:

SREČNA ŠTEVILA Z RAČUNALNIKOM

Ključne besede: računalništvo, programiranje, pascal, Ulamovo rešeto.

Elektronska verzija: http://www.presek.si/26/1358-Juvan.pdf

SREČNA ŠTEVILA Z RAČUNALNIKOM

V predzadnji številki lanskega letnika Preseka nas je profesor Grasselli "poskušal" prepričati, da je število 13 srečno (glej prispevek J. Grasselli, Nesrečno število 13 je srečno, Presek 25 (97/98), str. 258–260). O tem, ali mu je to uspelo ali ne, ne bomo razpravljali, pogledali pa bomo, kako lahko s pomočjo računalnika rešimo nalogo, ki jo je zastavil na koncu prispevka. Predlagal je, da poiščemo vsa srečna števila do 1000.

Predno pa se lotimo naloge, se spomnimo, kako sploh pridemo do srečnih števil. Začnemo z zaporedjem vseh naravnih števil

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, ...

Na prvem koraku iz zaporedja odvržemo vsa soda števila. Ostane zaporedje

1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, ...

Na drugem koraku v (že zmanjšanem) zaporedju pogledamo drugo število, to je 3, in iz zaporedja odvržemo vsako tretje število. Ostane

1, 3, 7, 9, 13, 15, 19, 21, 25, 27, ...

Na tretjem koraku pogledamo tretji element trenutnega zaporedja, to je 7, in iz zaporedja odvržemo vsak sedmi člen. Ostane

1, 3, 7, 9, 13, 15, 21, 25, 27, ...

Gotovo ste uganili nadaljevanje. Na četrtem koraku pogledamo četrti člen, ta je enak 9, in izločimo vsako deveto število. Nato sledi peti korak, pa šesti itn. Števila, ki "preživijo" prav vse korake, imenujemo *srečna*.

Postopek, ki smo ga opisali, imenujemo *Ulamovo rešeto*. Močno spominja na znano Eratostenovo rešeto, s katerim lahko določimo praštevila do neke vnaprej izbrane meje. Prepričan sem, da ste program za iskanje praštevil z Eratostenovim rešetom že kdaj napisali, saj gre za programerski izziv, s katerim se v mladosti sreča vsak "resen" programer. Ulamovo rešeto je sicer podobno, a vendarle nekoliko drugačno. Pri Eratostenovem rešetu je odločitev o tem, ali bomo število izločili ali ne, odvisna od ostanka pri deljenju, pri Ulamovem rešetu pa je kriterij mesto števila v že zmanjšanem zaporedju. Zato bo program za sejanje z Ulamovim rešetom malce bolj zapleten kot program za sejanje z Eratostenovim rešetom.

Program bomo zapisali v programskem jeziku pascal. Števila, ki so kandidati za srečna števila, bomo v programu hranili v tabeli `tab`. Ker bomo prvi korak sejanja opravili kar sami, bomo na začetku v tabelo vpisali liha števila. Pri tem bodo indeksi v tabeli tekli od 1 do 500, vrednost `tab[i]` pa bo $2 \cdot i - 1$. V spremenljivki `n` bomo ves čas hranili podatek, koliko kandidatov za srečna števila je še v tabeli `tab`. Na začetku bo `n` enako 500 (toliko je lihih števil do 1000). S števcem `k` bomo šteli korake. Začetna vrednost bo enaka 2.

Jedro programa bo zanka `while`. V vsaki ponovitvi zanke bomo opravili en (`k`-ti) korak sejanja. Zanko bomo ponavljali toliko časa, dokler bo `k`-ti element tabele (tega bomo hranili tudi v spremenljivki `d`) manjši ali enak `n`. Vemo, da nadaljnji koraki ne morejo več izločiti števil iz tabele, saj je v tabeli premalo števil. V vsaki ponovitvi zanke bomo iz tabele odstranili števila, ki so na mestih `d`, $2 \cdot$ `d`, $3 \cdot$ `d`, ... To storimo tako, da števila, ki so na mestih `d` + 1, `d` + 2, ..., $2 \cdot$ `d` − 1 pomaknemo za eno mesto v levo (v tabeli smo pred njimi odstranili eno število, in sicer tisto na mestu `d`), števila na mestih $2 \cdot$ `d` + 1, ..., $3 \cdot$ `d` − 1 za dve mesti v levo itn. Premikanje izvedemo v zanki `for`, pri čemer bomo poleg števca zanke, tega bomo označili z `j`, uporabili še pomožni števec `i`. Ko `j` teče od vrednosti `d` + 1 do `n`, vsakič pogledamo, ali je deljiv z `d`. Če je deljiv, ne storimo ničesar (pripadajoči element tabele tako preskočimo in ga s tem izločimo). Kadar pa `j` ni deljiv z `d`, pripadajoči element tabele `tab` premaknemo za nekaj mest naprej. Kam ga moramo prestaviti, nam pove števec `i`. Vsakič, ko premaknemo kak element, ta števec povečamo za ena. Kadar pa elementa ne premaknemo, to pa je vsakič, ko je `j` deljiv z `d`, števec `i` dodatno zaostane še za eno mesto za števcem `j`. Začetna vrednost števca `i` bo `d` (na tem mestu v tabeli `tab` nastane prva "luknja", ki jo je treba zapolniti). Po koncu zanke `for` seveda ne smemo pozabiti na popravek vrednosti spremenljivke `n` in na pripravo spremenljivk `k` in `d` za novi korak izločanja.

```
program SrecnaStevila;
{ Poišče srečna števila do konstante meja. }
  const
    meja = 1000;
    vel = (meja + 1) div 2;  { 500 }
  var
    tab: array [1..vel] of integer;  { tabela s kandidati }
    n: integer;  { koliko kandidatov je še v tabeli }
    k: integer;  { števec korakov }
    d, i, j: integer;
```

```
begin
  { Zaporedje lihih naravnih števil do konstante meja. }
  for i:=1 to vel do tab[i] := 2 * i - 1;
  n := vel;  { Po prvem koraku ostane toliko kandidatov. }
  k := 2;
  d := tab[k];
  while d <= n do begin
    i := d;  { indeks prvega števila, ki ga odvržemo }
    for j:=d + 1 to n do
      if j mod d <> 0 then begin
        tab[i] := tab[j];
        i := i + 1;
      end;
    n := i - 1;  { toliko kandidatov je "preživelo" k-ti korak }
    k := k + 1;  { priprava na naslednji korak }
    d := tab[k];
  end;  { while }
  { Izpis srečnih števil. }
  for i:=1 to n do write(tab[i]:5);
  writeln;
end.
```

Pravijo, da je znanje čim več različnih jezikov v življenju zelo pomembno. No, morda pri tem res niso mišljeni programski jeziki, a vseeno poskusimo gornjo rešitev zapisati še v programskem jeziku C.

Čeprav sta tako pascal kot C postopkovna programska jezika, pa je med njima vseeno kar nekaj razlik. Omenimo samo nekatere, ki bodo vplivale na zapis našega programa. V C-ju konstante običajno definiramo s pomočjo predprocesorjevega ukaza `#define`. C tudi loči male in velike črke, v navadi pa je, da s predprocesorjem definirane simbole pišemo z velikimi črkami. Tako bosta `meja` in `vel` postali `MEJA` in `VEL`. Zanka `for` je v C-ju precej splošnejša kot v pascalu in pravzaprav ustreza pascalski zanki `while` (skupaj z inicializacijo spremenljivk). Namesto operatorjev `mod` in `div` iz pascala imamo v C-ju operatorja `%` in `/` (operator `/` pomeni celoštevilsko deljenje le tedaj, kadar sta oba operanda celoštevilska izraza; če je vsaj eden od operandov realno število, potem se tudi deljenje izvede v realnem). Za prirejanje namesto `:=` uporabljamo `=`, v pogojih pa za primerjanje namesto pascalskih `=` in `<>` uporabljamo `==` in `!=`. Ena od bolj zoprnih lastnosti C-ja je, da se v tabelah elementi vedno začno z indeksom 0. To bo vplivalo tudi na zapis našega programa. Pri inicializaciji tabele `tab` bo števec `i` tekel od 0 do 499, element `tab[i]` pa bo na začetku dobil

vrednost $2 \cdot \mathtt{i} + 1$. Tudi pri izločanju števil bomo morali narediti ustrezni premik indeksov, saj `d`-ti element zaporedja ne bo več `tab[d]`, ampak `tab[d - 1]`. Tako bomo izločali elemente z indeksi $\mathtt{d}-1, 2\cdot\mathtt{d}-1, \ldots$ Tudi števec korakov `k` bo imel za ena manjšo vrednost. Z uporabo operatorja `++` bomo zapis programa v C-ju še nekoliko "zgostili". Vrednost izraza *ime*`++` je (stara) vrednost spremenljivke *ime*, "stranski" učinek pa, da se vrednost te spremenljivke poveča za ena.

```
#include <stdio.h>

/* Poišče srečna števila do konstante MEJA. */

#define MEJA 1000
#define VEL ((MEJA + 1) / 2) /* 500 */

int main(void)
{
  int tab[VEL]; /* tabela s kandidati */
  int n; /* koliko kandidatov je še v tabeli */
  int k; /* števec korakov */
  int d, i, j;

  /* Zaporedje lihih naravnih števil do konstante meja. */
  for (i = 0; i < VEL; i++) tab[i] = 2 * i + 1;
  n = VEL; /* Po prvem koraku ostane toliko kandidatov. */
  for (k = 1, d = tab[k]; d <= n; k++, d = tab[k]) {
    i = d - 1; /* indeks prvega števila, ki ga odvržemo */
    for (j = d; j < n; j++)
      if (j % d != d - 1) tab[i++] = tab[j];
    n = i; /* toliko kandidatov je preživelo k-ti korak */
  }
  /* Izpis srečnih števil. */
  for (i = 0; i < n; i++) printf("%5d", tab[i]);
  printf("\n");

  return 0;
}
```

Naj tudi sam zaključim z nalogo. Ugotovite, katero je bilo (oziroma bo) zadnje "srečno" leto v tem tisočletju in katero bo prvo "srečno" leto v prihodnjem tisočletju. Z majhno spremembo enega od gornjih programov tega ne bo težko storiti.

Martin Juvan