

PRESEK

List za mlade matematike, fizike, astronome in računalnikarje

ISSN 0351-6652

Letnik 23 (1995/1996)

Številka 5

Strani 282-285

Martin Juvan:

ISKANJE ŠIROKIH ŠTEVIL

Ključne besede: računalništvo, računalniško programiranje, naravna števila.

Elektronska verzija: <http://www.presek.si/23/1268-Juvan.pdf>

© 1996 Društvo matematikov, fizikov in astronomov Slovenije

© 2010 DMFA - založništvo

Vse pravice pridržane. Razmnoževanje ali reproduciranje celote ali posameznih delov brez poprejšnjega dovoljenja založnika ni dovoljeno.

ISKANJE ŠIROKIH ŠTEVIL

Pred časom sem obljubil, da bom napisal nekaj o štetju širokih števil. Predno pa se poglobimo v programiranje, se spomnimo, da je število široko, če je vsota njegovih števk enaka njihovemu produktu.

Ugotoviti sem želel, koliko je širokih števil, ki so manjša od milijarde. Ker ugotavljanje, ali je število široko, ni zapleteno, današnji osebni računalniki pa so že zelo hitri, sem se problema lotil z metodo grobe sile. Sestavlil sem spodnji program, ki za vsako število od 1 do izbrane zgornje meje po definiciji preveri, ali je široko.

```

program SirokaStevilal;
{ Poišče vsa široka števila do 10n z zaporednim preverjanjem po definiciji. }
const
  maxN=9;
var
  n: integer;           { Iščemo široka števila do 10n. }
  n10: longint;        { n10 = 10n }
  maxVsota: integer;   { največja možna vsota števk }
  vsota,produkt: integer; { vsota in produkt števk }
  S: longint;          { števec najdenih širokih števil }
  i,j: longint;
  ost: integer;
begin
  writeln('Iskanje širokih števil do 10↑n.');
```

{ vnos }

```

  write(' Vpisi n (n<',maxN+1,') : '); readln(n);
  n10 := 1; for i:=1 to n do n10 := 10*n10;      { Izračunamo 10n. }
  maxVsota := 9*n;                               { največja možna vsota števk }
  S := 0;
  for i := 1 to n10 do begin
    j := i; vsota := 0; produkt := 1;
    while j>0 do begin                            { Določi vsoto in produkt števk. }
      ost := j mod 10; j := j div 10;
      vsota := vsota+ost; produkt := produkt*ost;
      if produkt>maxVsota then break;
    end; { while }
    if vsota=produkt then S := S+1;                { Ali je število široko? }
  end; { for }
  write('Širokih števil do ',n10,' je ',S,'); readln;
end.
```

V zanko **while**, ki preverja, ali je število široko, sem dodal pogojni stavek **if**, ki z ukazom **break** prekine zanko, če produkt števk prekorači največjo možno vsoto števk. Žal tudi ta majhna izboljšava ni pomagala. Že ko sem gornji program preizkusil pri vrednosti $n = 6$, sem na odgovor čakal dobrih dvajset sekund. Hitro sem ocenil, da bi pri $n = 9$ na rezultat čakal vsaj šest ur, odločno preveč za tako preprosto nalogo.

Nekoliko razočaran zaradi neuspeha sem tako ugotovil, da reševanje naloge zahteva korenite spremembe v pristopu. Za nekaj nasvetov sem poprosil še kolego Marka Petkovška, ki ima vedno na zalogi kakšno uporabno idejo. Tako sem se odločil za reševanje s sestopanjem, ki pa sem ga dopolnil z učinkovitim testom, ki pove, kdaj je nadaljevanje poskušanja še smiselno. Zaradi lažjega programiranja sem uporabil rekurzivni zapis glavnega podprograma. Nastal je naslednji program.

```

program SirokaStevila2;
{ Poišče vsa široka števila do  $10^n$  s sestopanjem. }
const
  maxN=64;
var
  n: integer; { Iščemo široka števila do  $10^n$ . }
  stevke: array[0..maxN] of 1..9; { 0. mesto je pomožno. }
  S: longint; { števec najdenih širokih števil }
  m: integer;

procedure Razziri(i,m: integer; vsota,produkt: integer);
{ Določamo i-to števk, skupaj z njo jih moramo določiti še m;
  vsota je vsota, produkt pa produkt prvih i-1 števk. }
var
  j,k: integer;
  p: longint;
begin
  if m>0 then
    for j:=stevke[i-1] downto 1 do begin
      stevke[j] := j;
      if produkt*j<=vsota+m*j then Razziri(i+1,m-1,vsota+j,produkt*j);
    end
  else if vsota=produkt then begin { Izračuna, koliko števil smo našli. }
    p := 1; k := 1;
    for j:=2 to i-1 do begin
      p := p*j;
      if stevke[j]<>stevke[j-1] then
        k := 1
      else
        begin k := k+1; p := p div k; end;
    end; { for }
    S := S+p; { Zgrajeno široko število predstavlja p širokih števil. }
  end; { else if }
end; { Razziri }

begin
  writeln('Iskanje širokih števil do  $10^n$ .'); { vnos }
  write(' Vpisi n (n<',maxN+1,') : '); readln(n);
  S := 0; stevke[0] := 9; { začetni vrednosti }
  for m:=1 to n do Razziri(1,m,0,1);
  write('Širokih števil do  $10^n$ , n, je ',S,'.'); readln;
end.

```

S podprogramom **Razsiri** postopno, števk po števk, v globalni spremenljivki **stevke** gradimo kandidate za široka števila. Podprogram ima štiri parametre: prvi nam pove, na katero mesto moramo postaviti naslednjo števk, drugi šteje, koliko števk moramo še izbrati, in nam služi v pogoju, s katerim končamo rekurzijo, tretji in četrti pa sta vsota in produkt že izbranih števk. Široka števila, ki so manjša od 10^n , imajo lahko od 1 do n števk. Zato v glavnem programu podprogram **Razsiri** kliče mo n -krat. Vrednosti parametrov pri teh klicih so 1, m , 0 in 1, kjer m zavzame vrednosti med 1 in n . Tako začetno števk postavimo na prvo mesto v tabeli **stevke**, iščemo široka števila z m števki, začetna vsota števk je enaka 0, začetni produkt števk pa je enak 1.

Nobeno široko število ne vsebuje števk 0, zato pri gradnji kandidatov za široka števila uporabljamo le števk od 1 do 9. Da zmanjšamo potrebno delo in omogočimo učinkovito preverjanje, ali je nadaljevanje poskušanja smiselno, gradimo le števila, katerih števk tvorijo nenaraščajoče zaporedje. Tako naslednja števk ne sme biti večja od prejšnje. Če nam manjka še m števk, vemo pa, da nobena ne bo večja od j , se vsota števk lahko poveča kvečjemu za $m \cdot j$. Hkrati vemo, da se produkt števk pri dodajanju neničnih števk ne more zmanjšati. Tako dobimo pogoj, ki ga preverimo, preden naredimo rekurzivni klic.

Ker gradimo le števila z nenaraščajočim zaporedjem števk, nam vsako široko število, ki ga najdemo, če ima vsaj dve različni števk, pravzaprav predstavlja več širokih števil. Tako moramo, na primer, ko najdemo število 4211, šteti še števila 4121, 4112, 1421, 1412 in 1142 ter še šest števil, ki jih dobimo iz naštetih, če zamenjamo števk 2 in 4. Recimo, da ima dobljeno široko število n števk. Vseh različnih permutacij teh števk je $n(n-1) \cdot \dots \cdot 1$. To število označimo z $n!$ in ga imenujemo fakulteta števila n . Vendar pa nekatere permutacije dajo enako število, saj medsebojne zamenjave enakih števk števila ne spremenijo. Da dobimo dejansko število različnih števil, moramo $n!$ deliti še s fakultetami števil tistih števk, ki v dobljenem širokem številu nastopijo večkrat. V zgornjem primeru imamo štirimestno široko število 4211. Ker v njem le števk 1 nastopa večkrat, in sicer dvakrat, nam to število predstavlja $\frac{4!}{2!} = 12$ različnih širokih števil. Ta smo že opisali.

Ko sem preizkusil gornji program, sem bil prijetno presenečen. Naslednjo tabelo je izračunal že v nekaj sekundah.

