PRESEK
List za mlade matematike, fizike, astronome in računalnikarje

ISSN 0351-6652
Letnik **22** (1994/1995)
Številka 1
Strani 10–16

Ciril Pezdir:

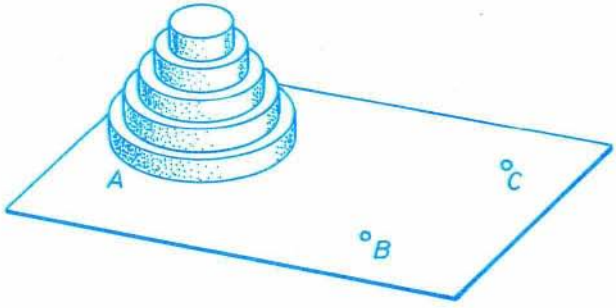
POSPLOŠENI HANOJSKI STOLP

Ključne besede: matematika, računalništvo, programiranje, rekurzije.

Elektronska verzija: http://www.presek.si/22/1208-Pezdir.pdf

POSPLOŠENI HANOJSKI STOLP

Hanojski stolp verjetno večina bralcev že pozna (slika spodaj). Vendar se v prispevku ne bomo ukvarjali le z osnovno verzijo hanojskega stolpa, temveč bomo nalogo nekoliko posplošili.



Najprej pa nekaj uvodnih besed za tiste, ki se boste s tem stolpom srečali prvič. Kot je razvidno z zgornje slike, hanojski stolp sestavljajo okrogle ploščice različnih velikosti, zložene na položaju z oznako A. Naloga zahteva, da prestavimo stolp z n ploščicami na položaj C, pri čemer si lahko pomagamo s pomožnim položajem B. Pri prestavljanju pa moramo upoštevati naslednji pravili:

1. na vsaki ploščici lahko ležijo le manjše ploščice;
2. hkrati lahko prestavimo le po eno vrhnjo ploščico.

Najprej poskusimo prestaviti stolp z majhnim številom ploščic, potem pa število ploščic povečujmo. Pri večjem številu ploščic izgubimo pregled in nalogi nismo več kos brez papirja in svinčnika. Program, napisan v pascalu, nam pomaga prestaviti stolp z veliko ploščicami:

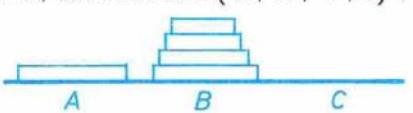
```
program Hanojski_stolp;
  var n: integer;                                    { velikost stolpa }

procedure Prestavi(izvor, pomozni, cilj: char; n: integer);
{ Prestavi n ploščic s položaja izvor na položaj cilj, }
{ pri čemer si pomaga s pomožnim položajem pomozni. }
begin
  if n>0 then begin
    Prestavi(izvor, cilj, pomozni, n-1);
    writeln('Prestavi ploscico s polozaja ',izvor,' na polozaj ',cilj,'.');
    Prestavi(pomozni, izvor, cilj, n-1);
  end;
end;
```

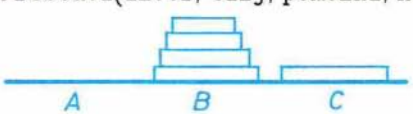
```
begin { glavni program }
  write('Vnesi velikost stolpa: '); readln(n);
  Prestavi('A', 'B', 'C', n);
end.
```

Način reševanja naloge je rekurziven. Vidimo, da podprogram `Prestavi` kliče sam sebe, vendar z drugačnimi parametri. Delovanje podprograma lahko razložimo takole:

Prestaviti želimo stolp z n ploščicami s položaja A na položaj C preko pomožnega položaja B; klic `Prestavi('A', 'B', 'C', n)` v glavnem programu.



To storimo takole: Najprej prestavimo stolp z $n-1$ ploščicami na pomožni položaj; klic `Prestavi(izvor, cilj, pomozni, n-1)` v podprogramu.



Potem prestavimo spodnjo ploščico na ciljni položaj; izpis `writeln('Prestavi ploscico s polozaja ',izvor,' na polozaj ',cilj,'.')` v podprogramu.

Nazadnje stolp z $n-1$ ploščicami prestavimo s pomožnega na ciljni položaj; klic `Prestavi(pomozni, izvor, cilj, n-1)` v podprogramu.

Premik stolpa z $n-1$ ploščicami seveda opravimo rekurzivno na enak način, le izvorni, ciljni in pomožni položaj so drugače razporejeni. Rekurzijo končamo, ko nam ni več potrebno prestaviti nobene ploščice.

Izračunajmo še število potrebnih premikov za prestavitev stolpa n ploščic. Označimo to število z $N(n)$. Iz podprograma `Prestavi` zlahka razberemo $N(1) = 1$ in rekurzivno zvezo

$$N(n) = N(n-1) + 1 + N(n-1) = 2 \cdot N(n-1) + 1.$$

Tako je

$$\begin{aligned} N(n) &= 2 \cdot N(n-1) + 1 = 2 \cdot (2 \cdot N(n-2) + 1) + 1 \\ &= 2 \cdot (2 \cdot (2 \cdot N(n-3) + 1) + 1) + 1 \\ &= 2 \cdot (2 \cdot (2 \cdot \ldots \cdot (2 \cdot (2 \cdot N(1) + 1) + 1) + \ldots + 1) + 1) + 1 \\ &= 2^{n-1} + 2^{n-2} + \ldots + 8 + 4 + 2 + 1 = 2^n - 1 \end{aligned}$$

Zadnjo enakost dobimo, če v zvezo $a^n - b^n = (a-b)(a^{n-1} + a^{n-2}b + \ldots + ab^{n-2} + b^{n-1})$ vstavimo $a = 2$ in $b = 1$. To je tudi najmanjše možno število premikov, kar se da lepo dokazati z matematično indukcijo.

Pa povečajmo število položajev. Kako si lahko sedaj, z večjim številom pomožnih položajev pomagamo pri prestavljanju stolpa? Navedeni program poišče rešitev, ki zaradi uporabe dodatnih pomožnih položajev potrebuje precej manj premikov kot rešitev, ki jo poišče prejšnji program.

```
program Posploseni_hanojski_stolp;
  var
    n: integer;                                   { velikost stolpa }
    k: integer;                                   { skupno število položajev }
    izvor, cilj: integer;                         { začetni in končni položaj }

function Pomozni(i, izvor, cilj: integer): integer;
{ Poišče številko i-tega pomožnega položaja, }
{ če izvzamemo položaja izvor in cilj. }
begin
  if i>=izvor then i:=i+1;
  if i>=cilj  then i:=i+1;
  if i=izvor then i:=i+1;
  Pomozni:=i;
end;

procedure Prestavi(izvor, cilj, n, k: integer);
{ Prestavi n ploščic s položaja izvor na položaj cilj, }
{ pri čemer si pomaga s k-2 pomožnimi položaji. }
  var i, maxPom: integer;
begin
  if n>0 then begin
    Prestavi(izvor, Pomozni(1, izvor, cilj), n-(k-2), k);
    if n<(k-2) then maxPom:=n else maxPom:=k-2;
    for i:=2 to maxPom do
      writeln('Prestavi ploscico s polozaja ',izvor,' na polozaj ',
              Pomozni(i,izvor,cilj),'.');
    writeln('Prestavi ploscico s polozaja ',izvor,' na polozaj ',cilj,'.');
    for i:=maxPom downto 2 do
      writeln('Prestavi ploscico s polozaja ',Pomozni(i,izvor,cilj),
              ' na polozaj ',cilj,'.');
    Prestavi(Pomozni(1, izvor, cilj), cilj, n-(k-2), k);
  end;
end;
```

```
begin { glavni program }
  write('Vnesi stevilo polozajev: '); readln(k);
  write('Vnesi stevilko zacetnega polozaja: '); readln(izvor);
  write('Vnesi stevilko koncnega polozaja: '); readln(cilj);
  write('Vnesi velikost stolpa: '); readln(n);
  Prestavi(izvor, cilj, n, k);
end.
```

Položaje oštevilčimo s števili od 1 do k. Funkcija `Pomozni(i, izvor, cilj)` vrne številko i-tega pomožnega položaja. Če je na primer $izvor = 2$, $cilj = 4$, potem z indeksom i takole naslovimo ostale, to je pomožne, položaje:

		izvor		cilj				
položaji	1	2	3	4	5	...	$k-1$	k
i	1		2		3	...	$k-3$	$k-2$

Spremenljivka $maxPom$ v podprogramu `Prestavi` predstavlja največji indeks pomožnega položaja, ki ga bomo uporabili. Če je ploščic manj kot je vseh pomožnih položajev ($n < k-2$), potem je $maxPom = n$, sicer pa je $maxPom = k-2$.

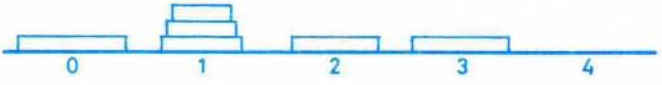
Delovanje podprograma `Prestavi` lahko prikažemo na podoben način kot v prejšnjem primeru.

Prestaviti želimo stolp z n ploščicami z izvora na cilj; klic

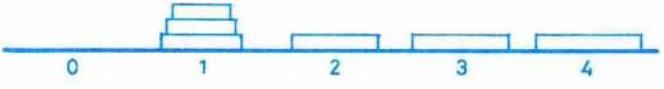
`Prestavi(izvor,cilj,n,k)`

v glavnem programu.

To naredimo takole: Najprej prestavimo stolp z $n-(k-2)$ ploščicami na prvi pomožni položaj; klic `Prestavi(izvor, Pomozni(1, izvor, cilj), n-(k-2), k)` v podprogramu. Na začetnem položaju ostane še $k-2$ (oziroma n, če je $n < k-2$) ploščic.



V drugem koraku na ostale pomožne položaje (teh je ravno $k-3$) razporedimo po eno ploščico z izvora; zanka **for** `i:=2` **to** `maxPom` **do** `writeln('Prestavi ploscico s polozaja ',izvor,' na polozaj ', Pomozni(i,izvor,cilj),'.')` v podprogramu. Na začetnem položaju tako ostane le še ena ploščica, ciljni položaj pa je še vedno prost.



Z izvora prestavimo največjo ploščico na cilj; izpis `writeln('Prestavi ploscico s polozaja ',izvor,' na polozaj ',cilj,'.')` v podprogramu.

Z ostalih pomožnih položajev v obratnem vrstnem redu prestavljamo ploščice na cilj; zanka **for** `i:=maxPom` **downto** `2` **do** `writeln('Prestavi ploscico s polozaja ',Pomozni(i,izvor,cilj),' na polozaj ',cilj, '.')` v podprogramu.

Stolp s prvega pomožnega položaja prestavimo na cilj; klic `Prestavi(Pomozni(1, izvor, cilj), cilj, n-(k-2), k)` v podprogramu.

Premik stolpa z $n-(k-2)$ ploščicami zopet opravimo na enak način, le da so izvor, cilj in pomožni položaj drugače razporejeni.

Pri vrednostih $k=3$, $izvor=1$ in $cilj=3$ je delovanje drugega programa enako prvemu.

Preštejmo še število potez, ki jih potrebuje gornji program za rešitev posplošene naloge hanojskega stolpa. Z nekaj sistematičnega dela, pri katerem nam je računalnik v pomoč, sestavimo naslednjo tabelo, v katero vpišemo število potez $N(k, n)$ v odvisnosti od števila ploščic n in števila položajev k:

$n\backslash k$	3	4	5	6
1	1	1	1	1
2	3	3	3	3
3	7	5	5	5
4	15	9	7	7
5	31	13	11	9
6	63	21	15	13
7	127	29	19	17
8	255	45	27	21

Iz tabele lahko uganemo tele formule, ki veljajo za poljuben n:

$$N(3,n)=\sum_{i=1}^{n}2^{i-1},\quad N(4,n)=\sum_{i=1}^{n}2^{i\ div\ 2},$$
$$N(5,n)=\sum_{i=1}^{n}2^{(i+1)\ div\ 3},\quad N(6,n)=\sum_{i=1}^{n}2^{(i+2)\ div\ 4}.$$

Z natančnim pregledom programa lahko razberemo tudi splošno formulo, ki velja za poljubna k in n:

$$\begin{aligned}N(k,n)&=\sum_{i=1}^{n}2^{(i+k-4)\,div\,(k-2)}\\&=2^{1+(n-1)\,div\,(k-2)}(k-2+(n-1)\,mod\,(k-2))-(2k-5).\end{aligned}$$

Če vstavimo $k=3$, dobimo ravno 2^n-1.

Seveda se lahko vprašamo, ali je postopek optimalen glede števila potez, ali pa mogoče obstaja postopek, s katerim bi prestavili stolp z manjšim številom potez. Žal odgovora ne poznam.