Ohjelmointitehtävä: Pascalin kolmio - ratkaisuesimerkkejä

Pascalin kolmio voidaan laskea riveittäin ylhäältä alas. Koska rivin 0 jälkeen seuraavan rivin laskemiseen tarvitaan tietoja vain edelliseltä, muita rivejä on turha pitää muistissa laskennan aikana.

Kolmio on mukava ajatella suorakulmaiseksi:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

Jos kullakin rivillä lasketaan alkioita oikealta vasemmalle, voidaan parhaillaan laskettava rivi tallettaa samaan tilaan kuin edellinen rivi.

Toiseksi tarpeetonta laskentaa voidaan vähentää käyttämällä hyväksi kolmion symmetrisyyttä ylimmän kulman puolittajan suhteen. Kustakin rivistä tarvitsee laskea vain korkeintaan puolet ja lisäksi parillisilla riveillä yksi alkio. Jos lasketaan kolmion oikea puolisko, edellä olevasta kolmiosta tulee siis laskea seuraava osa:

```
1
1
2 1
3 1
6 4 1
```

Jotta tässäkin riittäisi yhden rivin pitäminen muistissa, tulee parilliset rivit laskea oikealta vasemmalle ja parittomat vasemmalta oikealle. Ylin rivi on parillinen.

Tässä on ensin Pascal-kielinen ratkaisu, jossa on symmetrisyyttä ei ole otettu huomioon, sitten Java-kielinen ratkaisu, jossa symmetrisyyskin on otettu huomioon. Näistä ensimmäisestä saisi 23 pistettä, jälkimmäisestä täydet 25 pistettä.

Tehtävän voi ratkaista oikein myös käyttämällä hyväksi erästä kertomafunktion sisältävää lauseketta. Sellaista perusratkaisua voidaan parantaa ainakin ottamalla huomioon symmetria, välttämällä toistuvien osalausekkeiden laskeminen useaan kertaan ja laskemalla kertoma tehokkaasti.

Yhden rivin ratkaisu Pascalilla (23 pistettä)

```
program PascalsTriangle(input,output);
var n, i, j: integer;
    T: array [0..99] of integer; {Taulukon alkiotyyppi ei vaikuta arvosteluun}
begin
    readln(n);

    T[0]:=1;
    for i:=1 to n do
    begin T[i]:=1;
        for j:=i-1 downto 1 do T[j]:=T[j]+T[j-1]
    end;

    for j:=0 to n do write(T[j],' ');
    writeln
end.
```

Symmetriaa käyttävä puolen rivin ratkaisu Javalla (25 pistettä)

```
import java.io.*;
class PascalsHalf{

   public static void main(String[] args)
             throws IOException, NumberFormatException{

      int n=Integer.parseInt(new BufferedReader(
                          new InputStreamReader(System.in)).readLine());

      int[] T=new int[n/2+1]; /* Tehtävän määrittelyn mukaan int riittää */
      boolean evenLine=n==0;  /* Erikoistapaus: rivi 0 on parillinen, 1 pariton
                                 mutta molemmat 1/2-rivit ovat sisällöltään samoja */
      int k=0;                /* Rivin viimeisen alkion indeksi                    */

      T[0]=1;
      for (int i=2; i<=n; i++){
         evenLine=!evenLine;
         if (evenLine){
            T[++k]=1;
            for (int j=k-1; j>0; j--) T[j]+=T[j-1];
            T[0]*=2;
         }
         else for (int j=0; j<k; j++) T[j]+=T[j+1];
      }

      for (int j=k; j>=0; j--) System.out.print(T[j]+" ");
      for (int j=evenLine?1:0; j<=k; j++) System.out.print(T[j]+" ");
      System.out.println();
   }
}
```