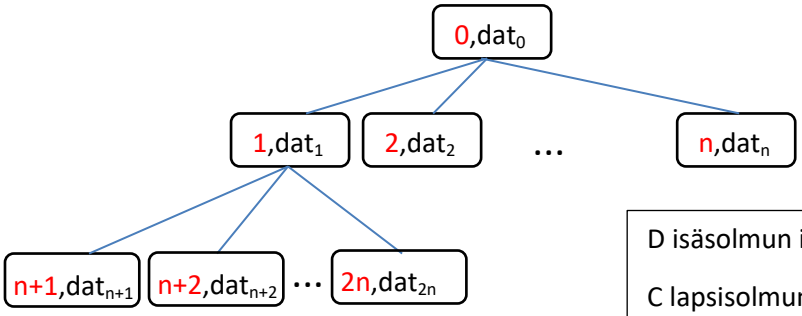
Määrittely: n-ääri puu on **täydellinen**, kun kaikilla isäsolmuilla on kaikki lapset (viimeinen rivi ei välttämättä ole täysi). Silloin puu voidaan sijoittaa C/C++ taulukkoon. Liikkuminen puussa tapahtuu indeksien avulla seuraavan kaavion mukaisesti.

n-ääripuu C/C++ taulukossa

(indeksit taulukossa 0,1,2,3,… sekä m/n=floor(m/n), kun m,n kokonaislukuja)



D isäsolmun indeksi

C lapsisolmun indeksi

dat solmun data

dat$_0$	dat$_1$	dat$_2$	...						

n-puussa isän indeksistä lasten indekseihin:

C_1=n*D+1
C_2=n*D+2
⋮
C_n=n*D+n

n-puussa lapsen indekseistä isän indeksiin (C>0):

```
switch (C % n) {
case 1: D=(C-1)/n; break;
case 2: D=(C-2)/n; break;
 ⋮
case n-1: D= (C-(n-1))/n;break;
case 0: D=(C-n)/n; break;
}
```

Huom! C-kielessä switch -lauseessa voidaan tapaukset case 1 → case n-1 yhdistää default: D=C/n
N alkiota muodostaa täydellisen n-ääripuun, mikäli (N-1)%n == 0. Jos kokonaismäärää rajoitetaan tarvittaessa N=N-(N-1)%n, saadaan täydellinen n-ääripuu.
Täydellisen n-ääripuun, jossa on N solmua, viimeisen isä(sisä)solmun indeksi saadaan lausekkeella

D=(N-n)/n (on viimeisen lapsen isä)

N alkion täydellisen n-ääripuun konstruointi, kun lasten data lasketaan isäsolmun datasta:

```
indexTree[0]=data0;
for (i=0; i<=(N-n)/n; i++){
  for (j=1; j<=n; j++){
    // f funktio: solmun data → solmun data
    indexTree[n*i+j]=f(indexTree[i],j);
  }
}
```

N-alkion täydellisen n-ääripuun läpikäynti juuresta lähtien:

```
for (i=0; i<N;i++){
  /* tee jotain alkiolla indexTree[i] */
}
```