Objekter i pseudokode

For nogle datastrukture er det nødvendigt at gemme mere data, end blot den værdi vi umiddelbart er interesseret i. Til dette formål bruger vi objekter. Tag som eksempel en hægtet liste. Hver knude/element, x, i listen skal indeholde en peger, x.next, til næste element i listen, samt selve værdien x.key. Et objekt er blot en slags kasse hvori vi kan indpakke en værdi, samt al den ekstra data vi ønsker. Den data vi gemmer i et objekt x kalder vi felter/fields.

En knude i en dobbelthægtet liste kan f.eks have felterne next, prev og key. Selve listen L, indeholder feltet head, hvor L.head er en peger til første element i listen. L kunne også gemme størrelsen på hele listen i et size felt, eller noget mere eksotisk, som hvor mange elementer der er blevet slettet fra L. Hvilke felter et objekt indeholder, kommer an på hvad vores datastruktur skal kunne.

Men hvordan vil vi skrive med pseudokode, at vi har oprettet et nyt objekt, med diverse felter? Her følger et par eksempler på hvordan det kan gøres.

Eksempler

Lad os sige vi vil have en dobbelthægtet liste af tal, hvor vi gemmer størrelsen af listen. Så selve listen har felterne head og size. En knude i listen har felterne prev, key og next. Hvor key er et tal, og next/prev er pegere. Her er en funktion som laver en tom liste:

```
emptyList()
   L = new List{head = nil, size = 0}
   return L
```

Her er en funktion, som givet en værdi, returnerer en knude x indeholdende vores værdi:

```
makeNode(val)
  node = new Node{prev = NIL, key = val, next = NIL}
  return node
```

Vi kunne også være interesseret i at lave et binært træ, hvor hver knude gemmer sin egen højde. Så en knude i træet skal have felterne: left, right, parent, key og height. Selve træet har blot feltet root. Vi kan igen lave et tomt træ og en knude.

```
emptyTree()
   T = new Tree{root = NIL}
   return T

makeTNode(val)
  node = new Node{left = NIL, right = NIL, parent = NIL, key = val, height = 0}
  return node
```