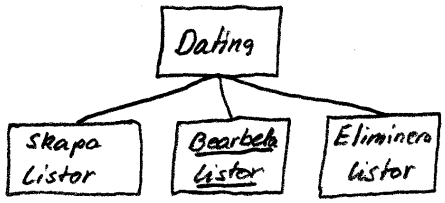
4) Programkonstruktion

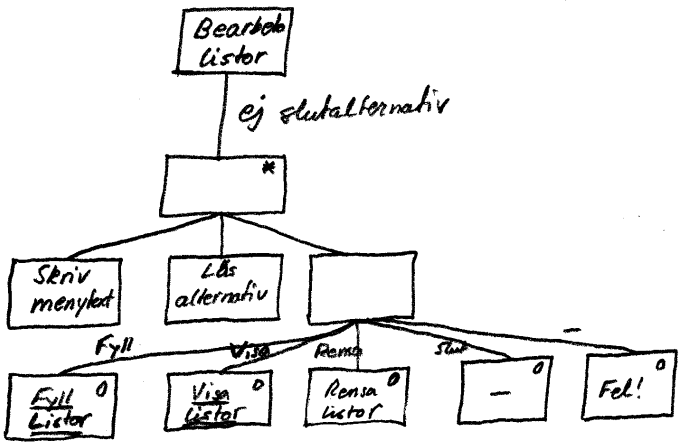
Program bearbetar data!

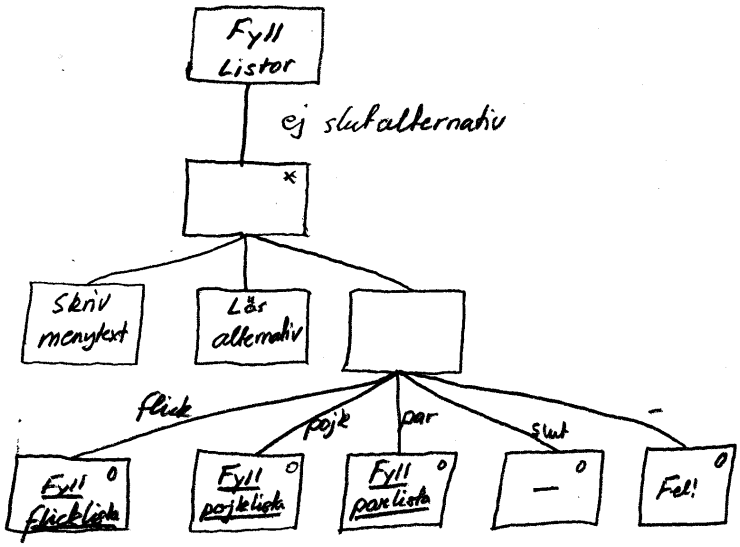
- bearbetar: Stegvis förfining
- data: Abstrakta datatyper, Dataflöden

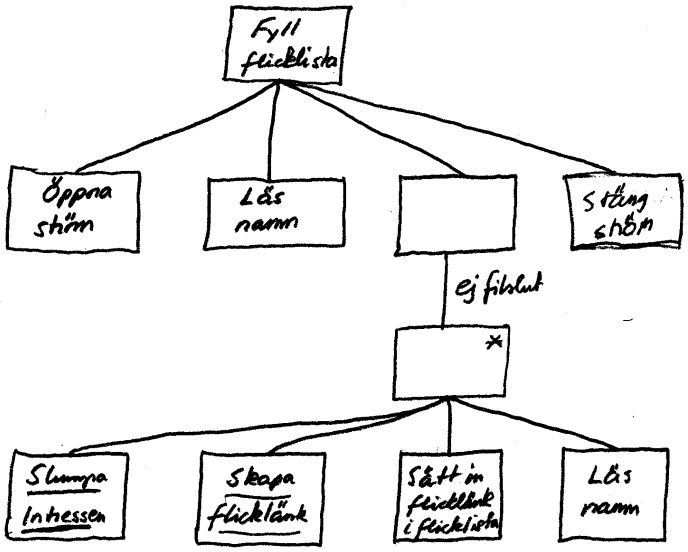
(Ex) Konstruera Dating med

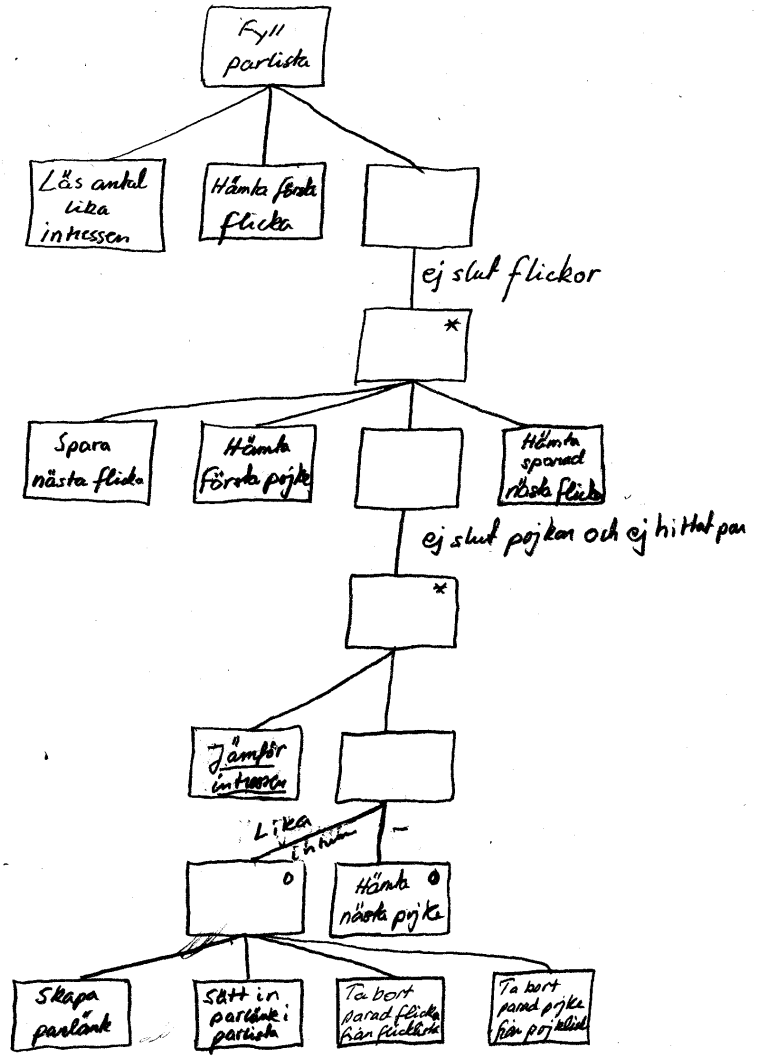
a) Stegvis förfining – vad ska programmet göra?
– Lämna detaljer till senare!






Fyll Listor
ej slutalternativ
*
Skriv menytext
Läs alternativ
flick
pojk
par
slut
Fyll flicklista
Fyll pojklista
Fyll parlista
—
Fel!
0

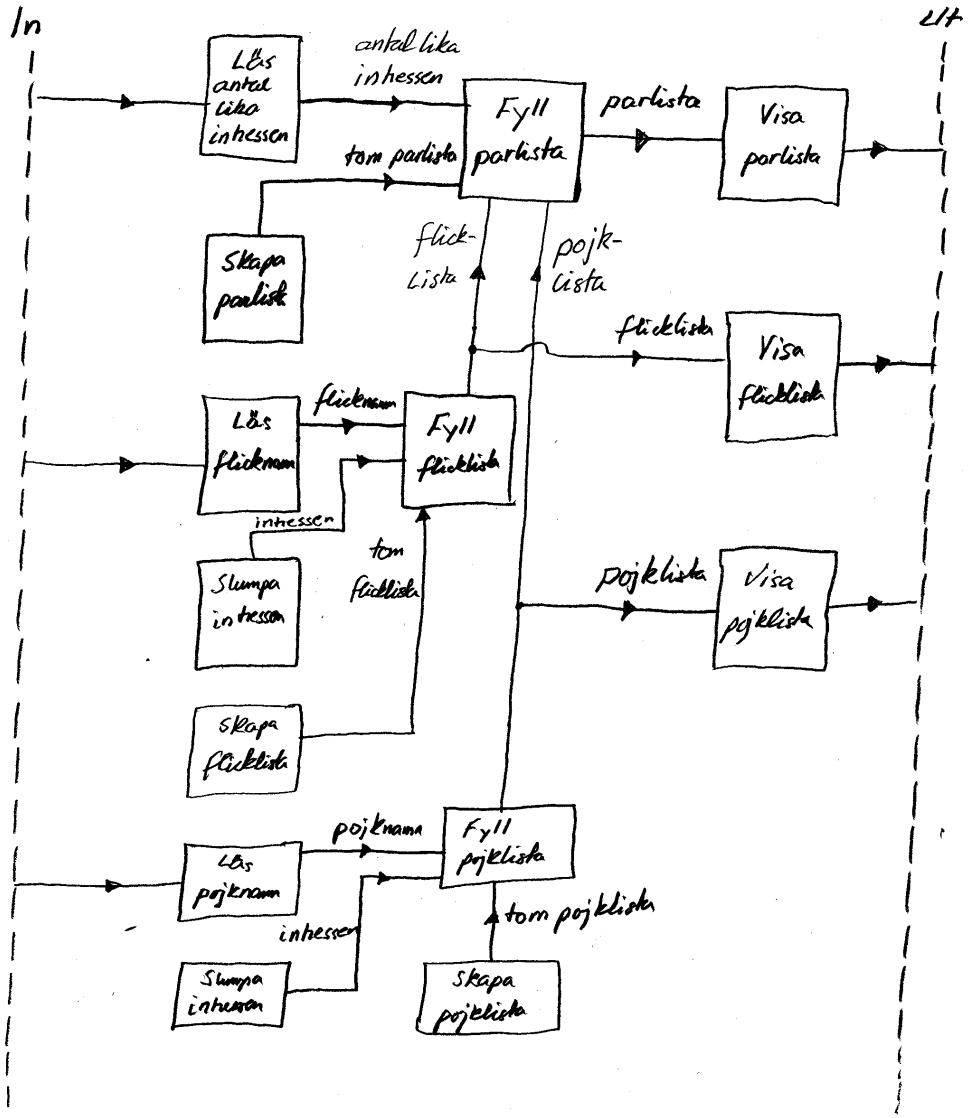

Fyll flicklista
Öppna ström
Läs namn
Stäng ström
ej filslut
*
Slumpa Intressen
Skapa flicklänk
Sätt in flicklänk i flicklista
Läs namn



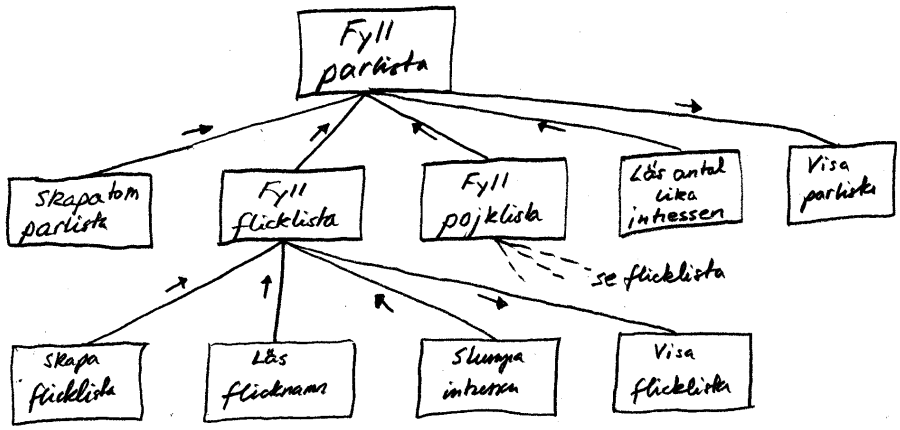
Fördelar-Ett bra sätt att lösa problem på!

Nackdelar-Fri komposition ger utspridd data ex. intressetabell vilket försvårar underhåll!

b) Dataflöden – vilken indata behövs för önskad utdata?
– varje databearbetning blir en funktion!



Häng upp "burkarna" i tänkta "snören"
av dataflöden och låt den burk
där dataflödet vänder från att ha varit
inflöde till att vara utflöde, bli huvudprogram.



Fördelar – dataflödet representerar verkligheten!

Nackdelar – utspridd data ex intressen
försvårar underhåll!

c) Abstrakta datatyper – vad ska programmet hantera?
– leta efter objekt

1) Leta objekt

Listor, flickor, pojkar, par, intressen
↑ ↑ ↑
DList Ding Itab

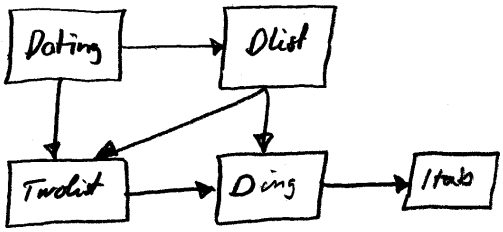
2) Operationer

DList – skapa, fylla, visa, ronsa, eliminera

Ding – skapa, visa, jfr

Itab – slumpa, visa, jfr

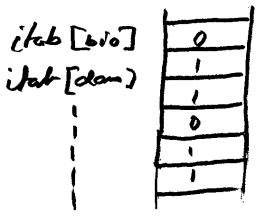
3) Beroenden



4) Implementera

Börja längst ner i beroendediagrammet, alltså Bottom Up, och testa varje modul

```
/* Itab.h */
typedef enum {bio, dans, musik, idrott, resor, djur} intressetyp;
typedef int itabtyp[6];
  :
/* Itab.c */
void slumpa (itabtyp itab)
{
    intressetyp intresse;
    for (intresse = bio; intresse <= djur; intresse++)
    {
        itab[intresse] = rand() % 2;
    }
}
```

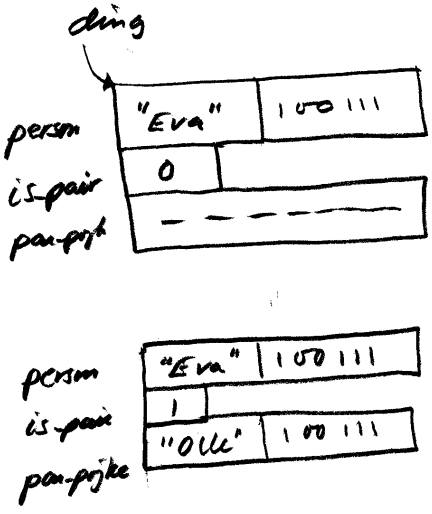



```
void visa_itab (itabtyp itab)
{
    intressetyp intresse;
    for (intresse = bio; intresse <= djur; intresse++)
    {   if (itab[intresse])
        {   switch (intresse)
            {
```

```
            case bio:
                printf("Bio   ");
                break;
            }
        }
        else
        {   printf("      ");
        }
    }
}
int antal_lika(itabtyp itab1, itabtyp itab2)
{
    /* Räkna antal lika intressen i itab1 och itab2 */
    /* OBS! Båda skall vara 1 */
}
```

```
/* ding.h */
typedef struct
{ char namn[30];
  itabtyp itab;
} persontyp;

typedef struct
{ persontyp person;
  int is_pair;
  persontyp par_pojke;
} dingtyp;
```



```
/* Ding.c */

void skapa_ding(dingtyp *ding, char *namn)
{
    strcpy(ding->person.namn, namn);
    slumpa(ding->persm.itab);
    ding->is_pair = 0;
}
void skapa_par(dingtyp *par, dingtyp flicka, dingtyp pojke
{
    par->person = flicka.person;
    par->par_pojke = pojke.person;
    par->is_pair = 1;
}
void visa_ding(dingtyp ding)
{
    printf("%-12s", ding.person.namn);
    visa_itab(ding.person.itab);
    if (ding.is_pair)
    {
        printf("%-12s", ding.par_pojke.namn);
        visa_itab(ding.par_pojke.itab);
    }
}
```

```
/* Dlist.c */
void fyll_lista(headtyp *lista, char *filnamn)
{
    FILE *tsin;
    char *namn[30];
    linktyp *lp;
    dingtyp d;
    tsin=fopen(filnamn, "rt");
    while (fscanf(tsin, "%s", namn) != EOF)
    {
        newlink(&lp);
        skapa_ding(&d, namn);
        putlink(d, lp);
        inlast(lp, lista);
    }
    fclose(tsin);
}
;
```

```
void fyll_parlista(headtyp *parlista, headtyp *flicklista,
                   headtyp *pojklista, int antal_gem_intresse)
{
    linktyp *flicklink, *pojklink, *parlink, *nextflicklink;
    dingtyp flickding, pojkding, parding;
    int hittat_par;

    flicklink = firstlink(flicklista);
    while (flicklink != NULL)
    {
        nextflicklink = succlink(flicklink);
        flickding = getlink(flicklink);
        pojklink = firstlink(pojklista);
        while (pojklink != NULL)
        {   pojkding = getlink(pojklink);
            if (passar_ihop(flickding, pojkding, gem_intresse))
            {   newlink(&parlink);
                skapa_par(&parding, flickding, pojkding);
                putlink(parding, parlink);
```

```
            insort(parlink, parlista, is_less_ding);
            elimlink(&pojklink);
            elimlink(&flicklink);
            hittat_par = 1;
        }
        else
        {
            pojklink = succlink(pojklink);
        }
    }
    flicklink = nextflicklink;
  }
}
```

Fördelar ÷ Lätt att förändra, underhålla!

– Bra bild av verkligheten

Nackdelar – Normalt tänker man inte i objekt utan i funktioner.

Hemuppgift: Du vill införa en förändring i programmet eftersom du vill ha ett nytt intresse datorspel infört. Var i programmet behöver man ändra?