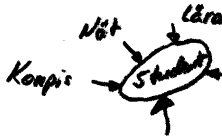
Programmeringsmetodik, 5p

Gunnar.Joki@tech.oru.se

www.oru.se/tech – kurssida – veckoplanering
– föreläsningsanteckningar
– tentor

Mål – konstruera större program i C++

Metod – Learning by doing (jfr ta körkort för lastbil)

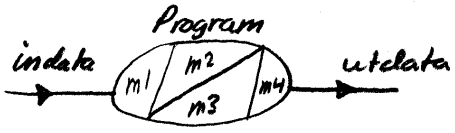


– Litteratur - Kompendium på nätet
– Vägen till C

Verktyg – PC, win 2000
– MicrosoftVisualC++.net

Krav – delkurs 1, 3.5p - tentamen
– delkurs 2, 1.5p - närvaro på 6 datorövningar
– inl 1 och inl 2 godkända
– för att få tentamen rättad ska inl 1 vara godkänd och inl 2 inlämnad senast tentamensdagen

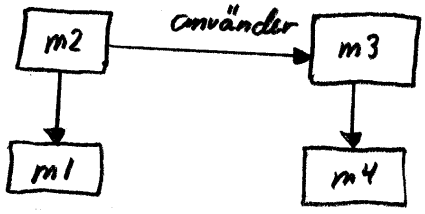
1) Modulär programutveckling.



Stora program måste delas in i hanterbara delar (enheter, moduler).

Modul – separatkompilerbar och testbar fil

Beroendediagram – diagram som visar vilka moduler använder vilka tjänster i form av data och funktioner.



Hur delar man in i moduler?

- Huvudprogrammet utgör en modul
- Samla liknande funktioner i en modul
- Datatyp och funktioner på denna datatyp bildar en modul

(E-x) Program som beräknar medelvärde, min och max för en reell vektor.

a) allt i samma modul (fil)

```
/* Rvektest.c */
float rvekmin(float rvek[], int nr)
{
    int i;
    float min = rvek[0];
    for (i=1; i<nr; i++)
    {
        if (rvek[i] < min)
        {
            min = rvek[i];
        }
    }
    return min;
}

float rvekmax(float rvek[], int nr)
{
  //se ovan
}
float rvekmed(float rvek[], int nr)
{
   int i;
   float sum = 0.0;
```

```
    for (i=0; i<nr; i++)
    {
        sum += rvek[i];
    }
    if (nr>0)
    {   return sum/nr;
    }
    return 0;
}
#include <stdio.h>
#include <conio.h>
void main()
{
    float rv[5] = {4.19, 4.28, 4.24, 4.35, 4.18};

    printf("Min = %.2f\n", rvekmin(rv,5));
    printf("Max = %.2f\n", rvekmax(rv,5));
    printf("Medel = %.2f\n", rvekmed(rv,5));
    getch();
}
```

b) dela upp i 2 moduler med funktioner för reell vektor i en modul och huvudprogrammet i en annan.

```
/*Rvek.c */
float rvekmin ( float rvek[], int nr)
{
  // Se ovan!
}
float rvekmax ( float rvek[], int nr)
{
  //Se ovan!
}
float rvekmed ( float rvek[], int nr)
{
  //Se ovan
}
```

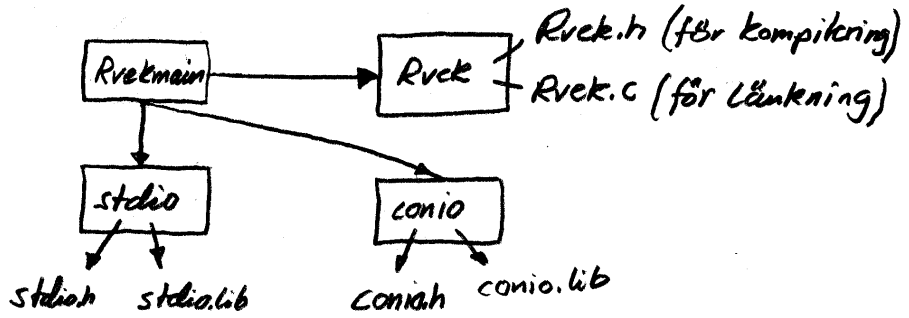
Modul 1 som kan kompileras separat!

```
/* Rvekmain.c */
float rvekmin (float rvek [], int nr);
float rvekmax( float rvek [], int nr);
float rvekmed( float rvek [], int nr);

#include <stdio.h>
#include <conio.h>

void main()
{
    float rv[5] ={7.43, 7.51, 7.48, 7.56, 7.49};
    //Se ovan!
}
```

Funktionsprototyper som behövs för kompilering av huvudprogrammet!

Bäst att skriva dem i headerfilen Rvek.h!

Beroendediagram

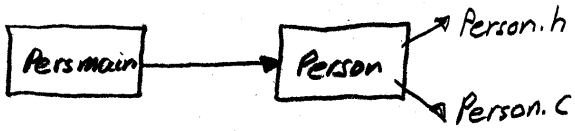


Headerfilerna innehåller funktionsprototyper och datatyper som behövs för kompilering av modul.

För att kunna länka ihop modulerna till en körbar fil måste även själva funktionen vara med, alltså .c filen (eller .lib när de gäller standardbiblioteken)

Abstrakt datatyp – datatyp med tillhörande
– funktioner

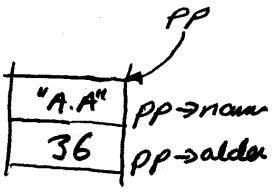
(Ex) Program som läser in namn och ålder för personer och skriver ut dessa i ordning med yngsta först.



```
/* Person.h */
typedef struct
{
   char namn[30];
   int alder;
}person;
void las-person(person *pp);
/* Läser en person */
void skriv-person(person p);
/* Skriver en person */
int yngre-person(person p1, person p2);
/* Sant om p1 är yngre än p2 */
```

OBS! Typedef ger ett aliasnamn till structen nämligen person!

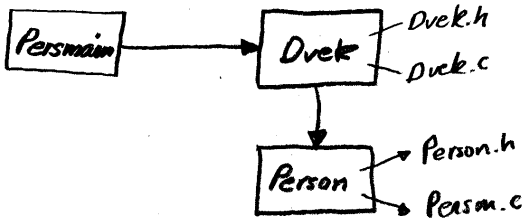
```
/* Person.c */
#include "Person.h"
#include <stdio.h>
void las-person(person *pp)
{
  printf("Ge namn: ");
  gets(pp->namn);
  printf("Ge ålder: ");
  scanf("%d", &pp->alder);
  getchar();
}
```



```
void skriv_person(person p)
{
   printf("Namn: %s\n", p.namn);
   printf("Ålder: %d\n", p.alder);
}

int yngre_person(person p1, person p2)
{
   return p1.alder < p2.alder;
}

#include "Person.h"
void main()
{
    person a, b;  //Två personer a och b

   las_person(&a);
   las_person(&b);
   if( yngre_person(a, b))
   {
      skriv_person(a);
      skriv_person(b);
   }
   else
   {
      skriv_person(b);
      skriv_person(a);
   }
}
```

Funktionsbibliotek – generella funktioner som kan användas av flera program

(Ex) Tillverka ett generellt bibliotek av funktioner för beräkning av min, max och medel för vektorer samt använd biblioteket för att bestämma yngsta och äldsta personen och medelåldern för ett antal personer i en personvektor.




```
/* Dvek.h */
#include "Person.h"
typedef person dtyp;
```

Här anpassas biblioteket till aktuell data genom att person ges aliasnamnet dtyp!

```
dtyp dvekmin(dtyp dvek[], int nr, int (*jfr)(dtyp d1, dtyp d2));
dtyp dvekmax(dtyp dvek[], int nr, int (*jfr)(dtyp d1, dtyp d2));
float dvekmed(dtyp dvek[], int nr, float (*dsum)(dtyp d));
```

Pekare till funktion!

```
/* Dvek.c */
#include "Dvek.h"
dtyp dvekmin(dtyp dvek[], int nr, int (*jfr)(dtyp d1, dtyp d2))
{
    int i;
    dtyp min = dvek[0];
    for(i=1; i<nr; i++)
    {
        if (jfr(dvek[i], min))
        {
            min = dvek[i];
        }
    }
    return min;
}
dtyp dvekmax(dtyp dvek[], int nr, int (*jfr)(dtyp d1, dtyp d2))
{
    //se ovan med if(jfr(max, dvek[i])
}
float dvekmed(dtyp dvek[], int nr, float (*dsum)(dtyp d))
{
    int i;
    float sum = 0.0;
    for(i=0; i<nr; i++)
    {
        sum = sum + dsum(dvek[i]);
    }
    return sum/nr;
}
```

```
/* Persmain.c */
#include "Dvek.h"
void main()
{
    person pvek[5] = {{"A.A", 36}, {"B.B", 25}, {"C.C", 42},
                      {"D.D", 28}, {"E.E", 32}};
    printf("Yngsta : \n");
    skriv_person(dvekmin(pvek, 5, yngre_person));
    printf("Äldsta : \n");
    skriv_person(dvekmax(pvek, 5, yngre_person));
    printf("Medelålder: %6.1f \n", dvekmed(pvek, 5, psum));
}
```

Modulen Person måste kompletteras med summafunktionen

```
float psum(person p)
{
   return p.alder;
}
```

OBS! #include "Dvek.h" inkluderar även "Person.h" och ska alltså ej inkluderas separat. Vill man inkludera "Person.h" i huvudprogrammet måste man sätta villkorlig kompilering av "Person.h" för att ej få dubbla definitioner av datatypen person.

Villkorlig kompilering

(Ex)
```
#ifndef PERSONH
#define PERSONH
typedef struct
{
    char namn[30];
    int alder;
} person;
:
#endif
```

Villkorlig kompilering ser till att kocken endast kompileras en gång oberoende av hur många gånger Person.h inkluderas!

Regel: Sätt in villkorlig kompikring i alla abstakta datatyper!

Hemuppgift: Använd sorteringsbiblioteket i kompendiet kap 7 till att sortera vektorn pvek ovan. Anpassa först biblioteket till datatypen person genom

```
/*Sort.h */
#include "Person.h"
typedef person dtyp;
:
```