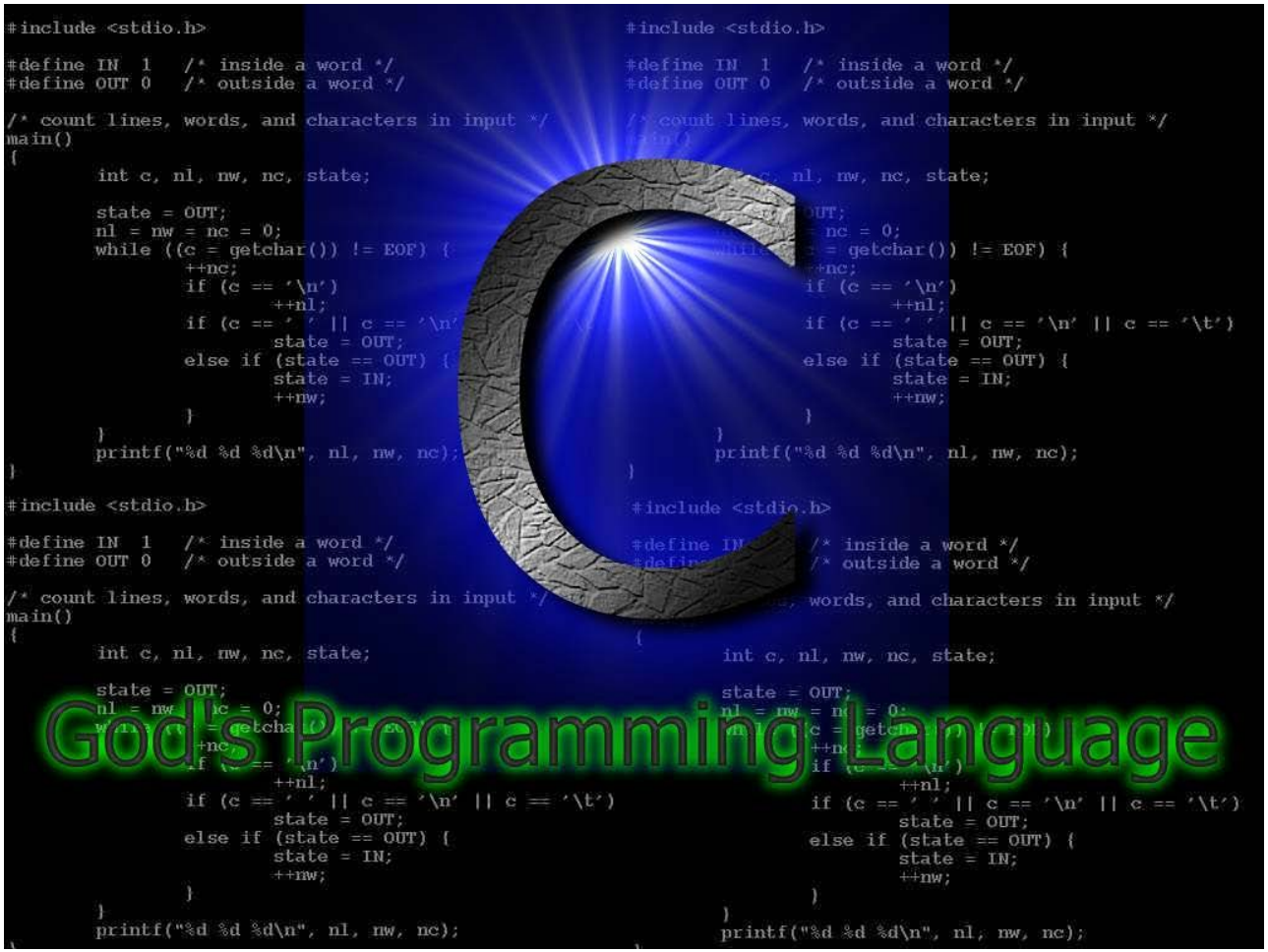
Språket för inbyggda system 2



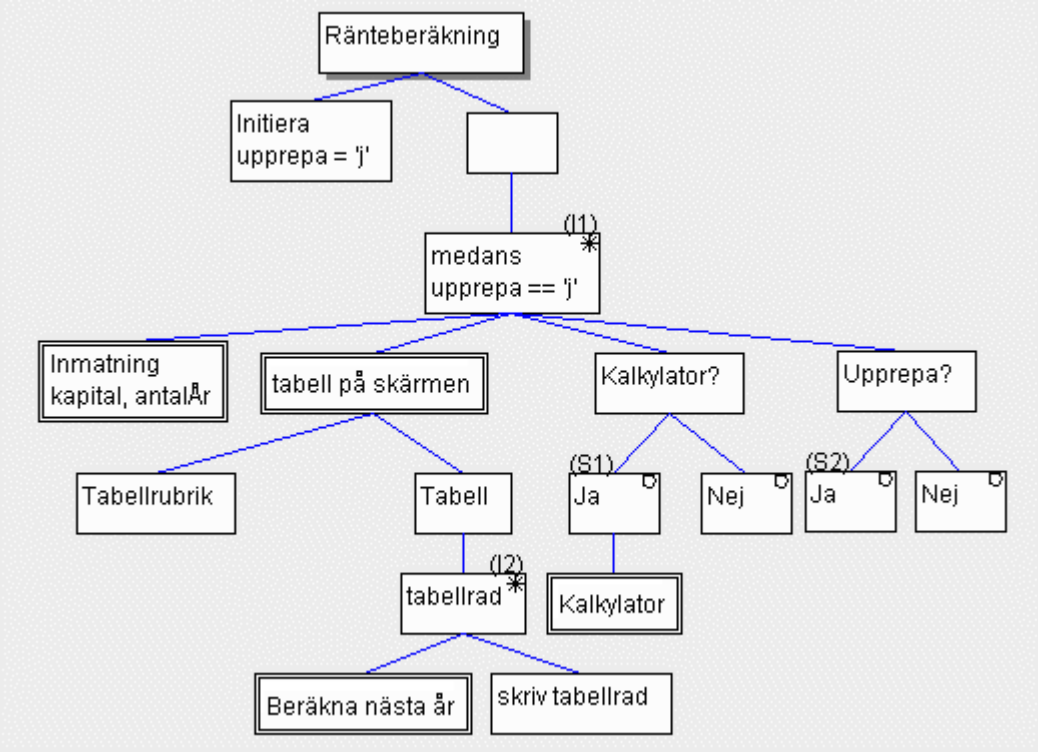
Ett programmeringsprojekt,

flera personer utvecklar ett program

LADYSOFT - programming

Vi ska utveckla ett ränteberäkningsprogram !

Programmets struktur *modulariserad programmering*



Anders Sjögren as@kth.se

Preprocessorn
något om dess användning i projektet

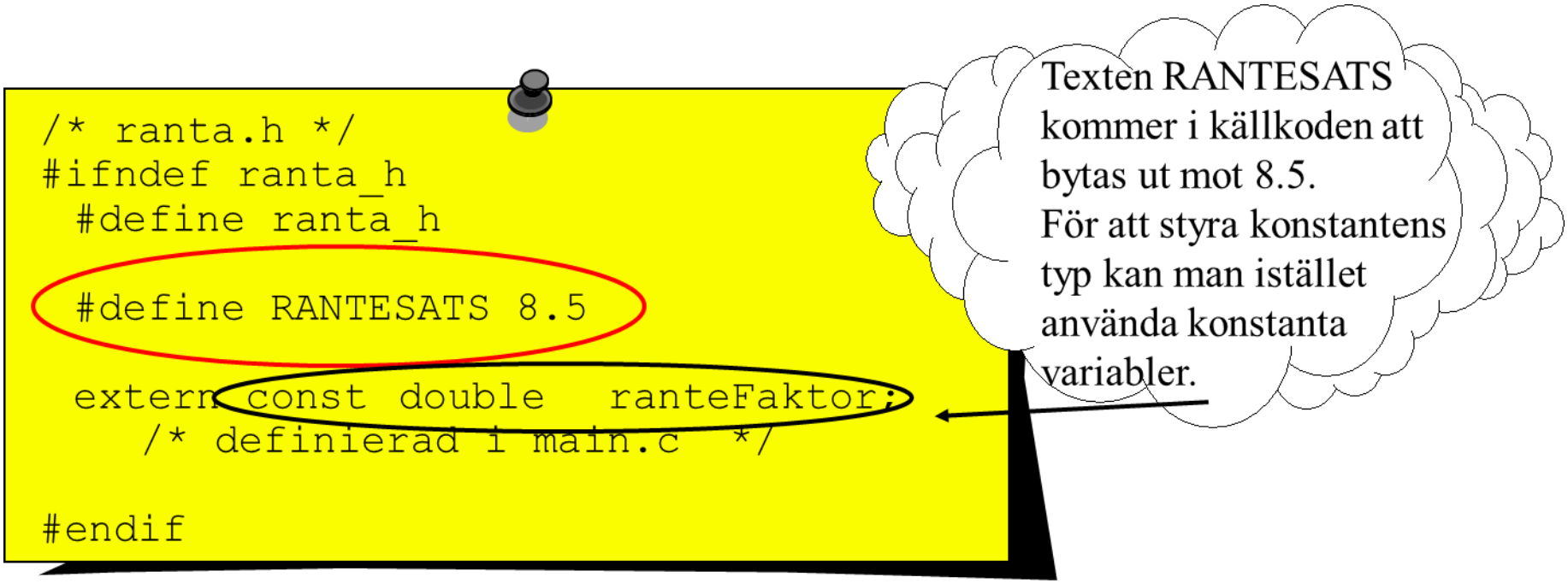
Preprocessorn är ett textmanipuleringsinstrument som används/utförs före kompilering

- alla preprocessordirektiv föregås av # - tecknet
- de vanligaste preprocessovarianterna är
  - makro utan argument - ”konstantdefinition”
  - makro med argument
  - filinkludering
  - villkorligt medtagande/uteslutande av text

Anders Sjögren as@kth.se

Konstantdefinition

- de vanligaste preprocessovarianterna är
  - makro utan argument - ”konstantdefinition”



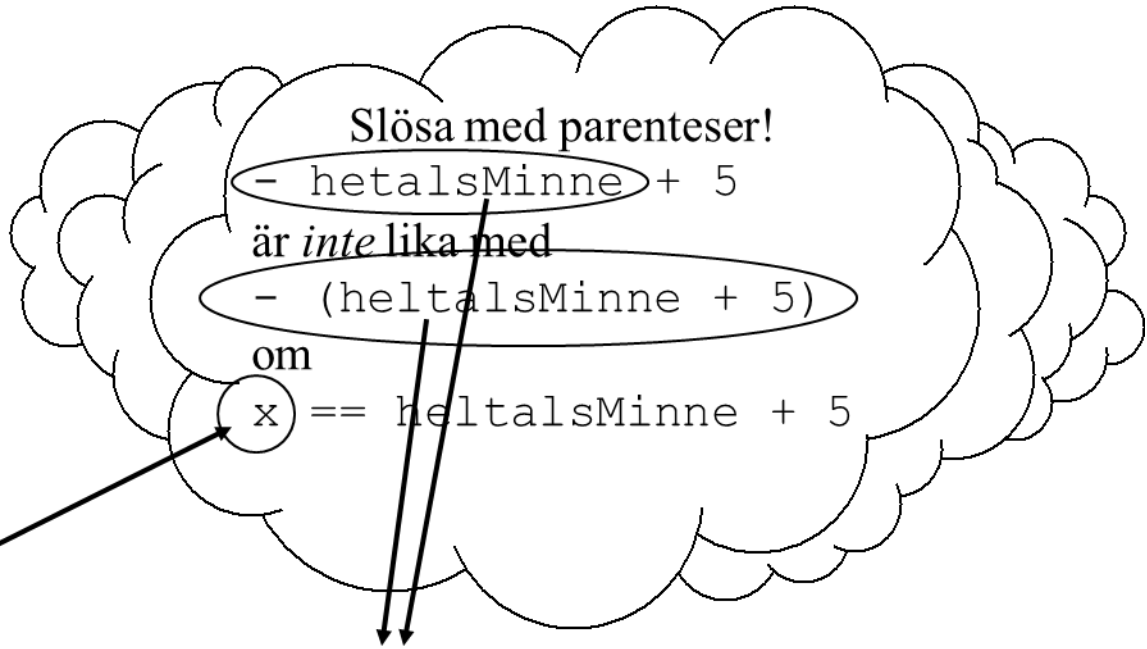
Makro med argument

- de vanligaste preprocessorvarianterna är
  - makro med argument

Makro med argument liknar funktioner. De används för att förenkla uttryck som anses ”för små” för att göra funktion av. Man kortar exekverings-tiden pga att inget funktions-anrop görs ty uttrycket expanderas i källkods-texten.

```
/* tabell.h */
#include "ranta.h"
#include <stdio.h>

#if !defined(tabell_h)
   #define tabell_h

   #define ABS( x )( (x) > 0 ? (x) : -(x) )
   void TabellPaSkarmen( double kapital, int antalAr );

#endif
```

Parenteser!




```
/* tabell.h */
#include "ranta.h"
#include <stdio.h>

#if !defined(tabell_h)
   #define tabell_h

   #define ABS( x )( (x) > 0 ? (x) : -(x) )
   void TabellPaSkarmen( double kapital, int antalAr );

#endif
```

Anders Sjögren as@kth.se

Fil-inkludering

- de vanligaste preprocessorvarianterna är
  - filinkludering

```
/* kalkylat.h */
#include "ranta.h"
#include <stdio.h>

#if !defined(kalkylat_h)
  #define kalkylat_h

  void Kalkylator( void );

#endif
```

Angivna textfilers innehåll kommer att expanderas här. ” ” betyder att filen hämtas från en egen katalog medans < > betyder att filen hämtas från en i utvecklingsmiljön fördefinierad katalog.

villkorligt medtagande/uteslutande

- de vanligaste preprocessovarianterna är
  - villkorligt medtagande/uteslutande av text

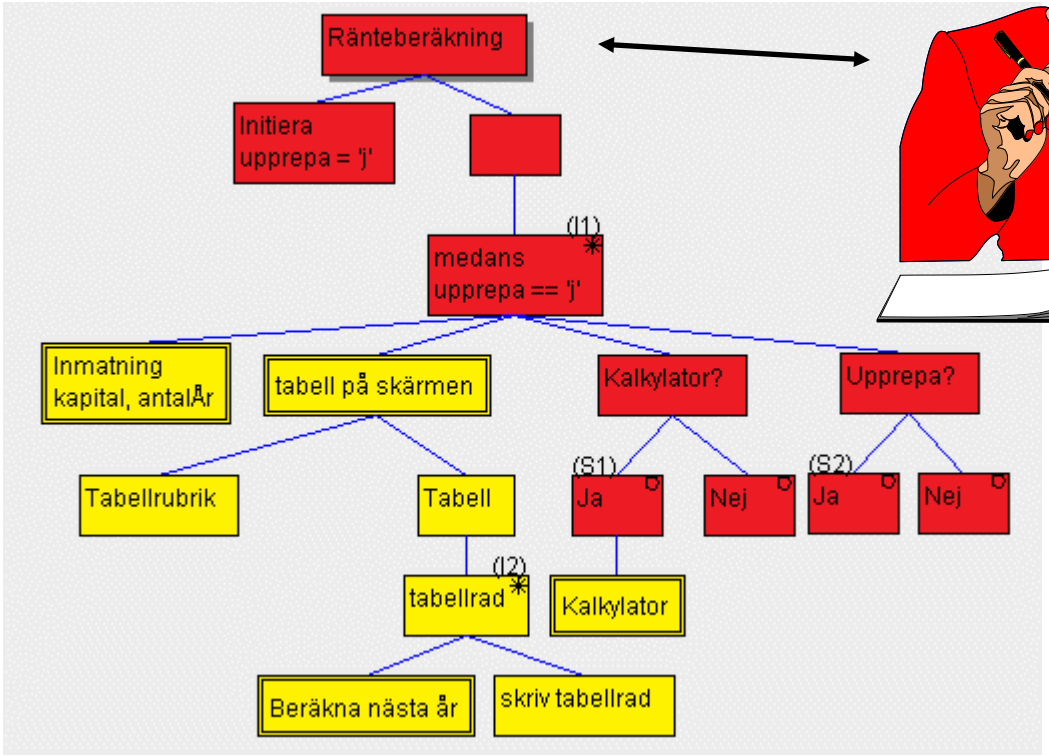
```
/* kalkylat.h */
#include "ranta.h"
#include <stdio.h>

#if !defined(kalkylat_h)
  #define kalkylat_h

  void Kalkylator( void );

#endif
```

För att förhindra att viss text kommer med på flera ställen kan denna konstruktion användas.

kalkylat_h används som en flagga att testa på.

Programmets struktur

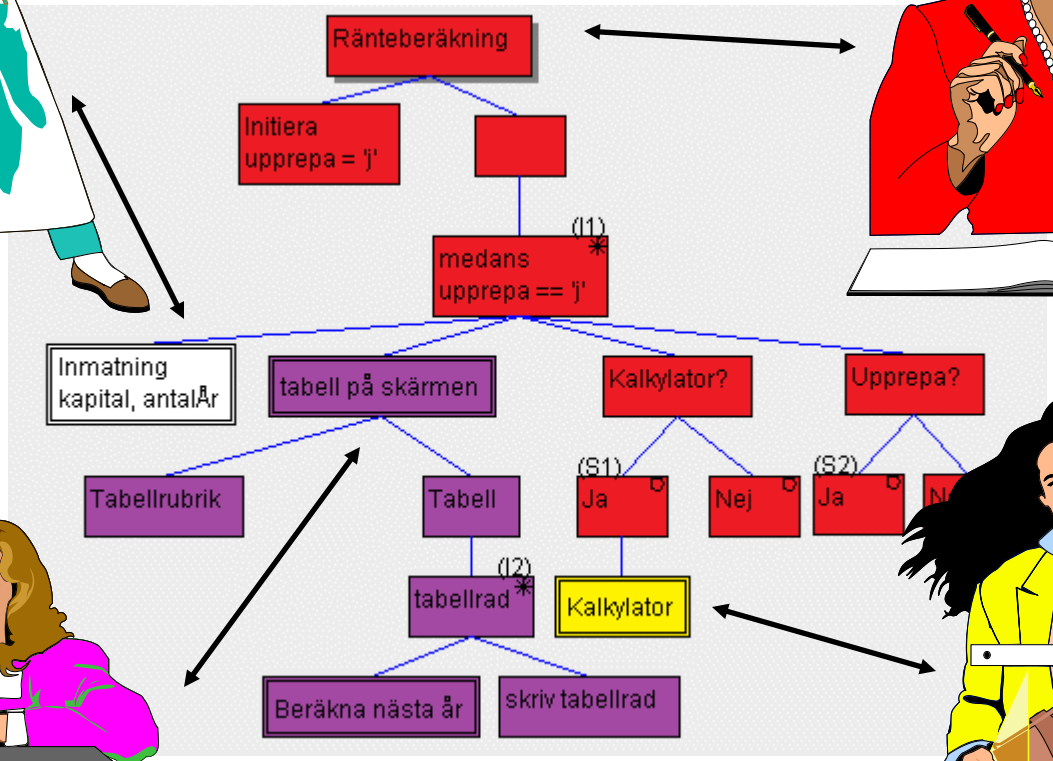
arbetsfördelning

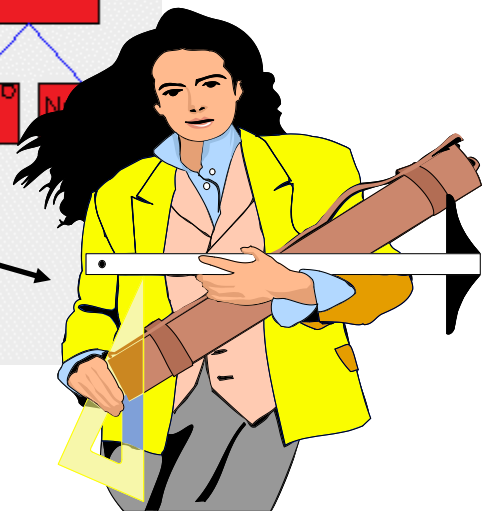


Anders Sjögren as@kth.se

Programmets struktur

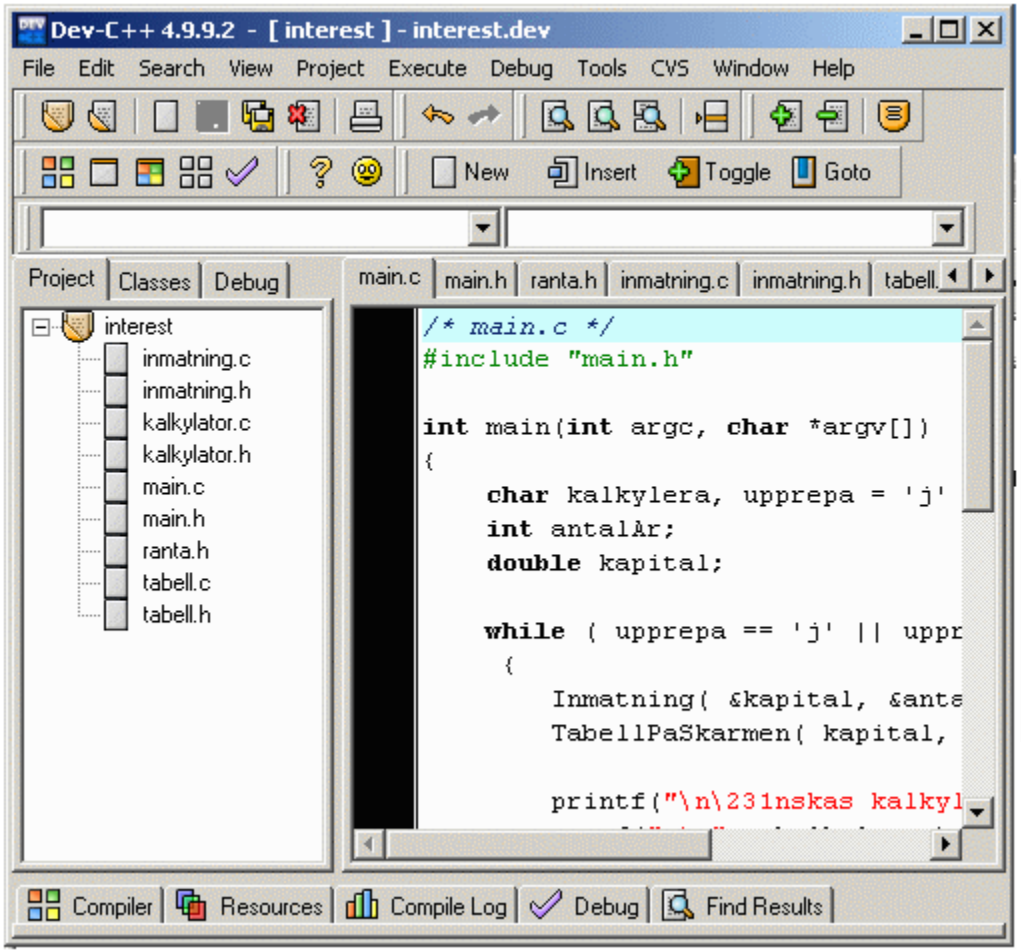
arbetsfördelning





Anders Sjögren as@kth.se

Filer i projektet

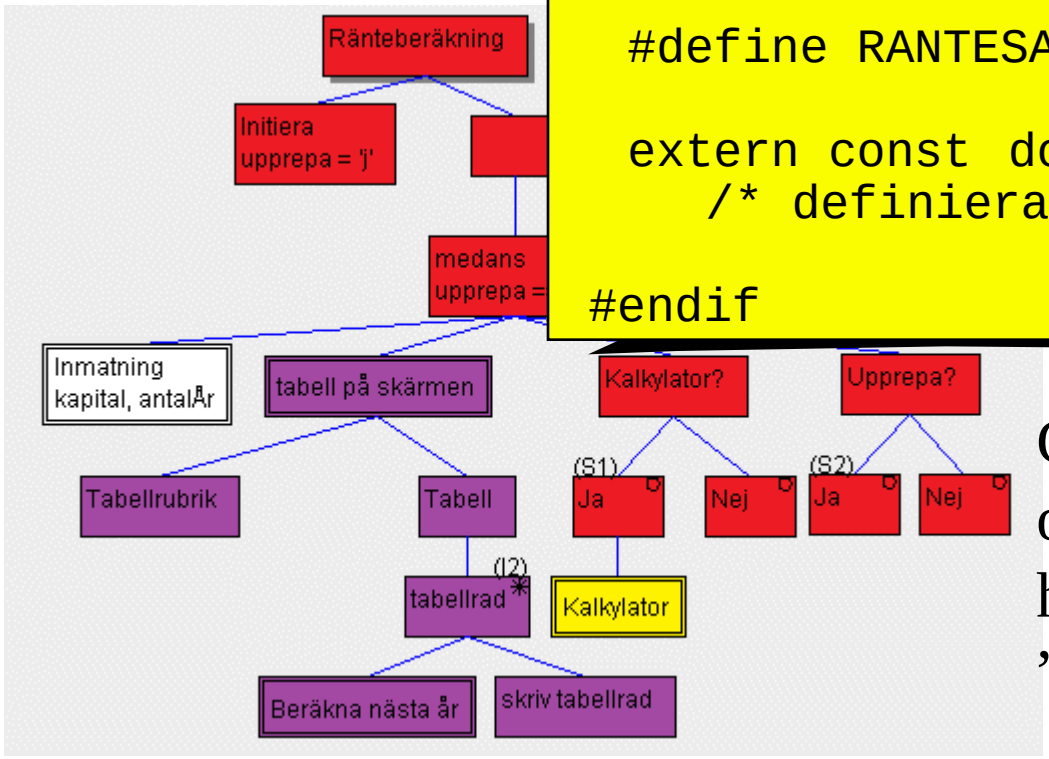


Anders Sjögren as@kth.se

Programmets struktur

gemensam *headerfil*

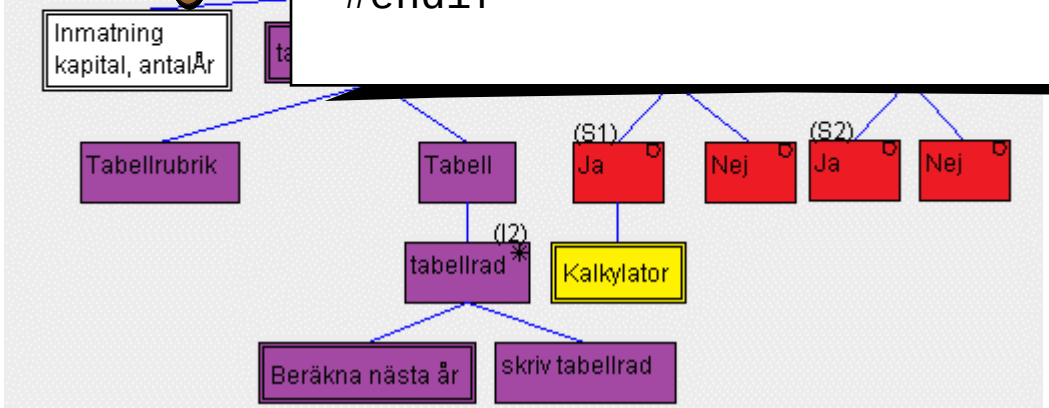
```
/* ranta.h */
#ifndef ranta_h
  #define ranta_h

  #define RANTESATS 8.5

  extern const double  ranteFaktor ;
     /* definierad i main.c  */

#endif
```



Gemensamma deklarationer och definitioner samlas i en header-fil som alla ”inkluderar”.

Arbetsfördelning och filer

```
/* inmatnin.h */
#include "ranta.h"
#include <stdio.h>
#include <conio.h> /* ej ANSI, clrscr() används */

#if !defined(inmatnin_h)
  #define inmatnin_h

  void Inmatning(double* kapitalPek,int*antalArPek);

#endif
```



Varje projektdeltagare skriver sina egna filer som kan kompileras separat för att kontrollera syntaxen i den egna koden.

Funktionsdeklarationer och makron i h-filen(er) och funktionsdefinitioner i c-filen(er).

Arbetsfördelning och filer

```
/* inmatnin.c */
#include "inmatnin.h"

void Inmatning( double* kapitalPek, int* antalArPek ){
  clrscr();   /* ej ANSI-C */
  printf("\nBeräknar kapitaltillväxt vid %0.1f ränta",RANTESATS)
  printf("\n=========================================");
  printf("\nPositivt kapital räknar framåt i tiden.");
  printf("\nNegativt kapital räknar bakåt i tiden.");
  printf("\n\nInsatt kapital och antal år ? ");
  printf("(-->(+/-)1000 10)-->");
  scanf("%lf%d", kapitalPek, antalArPek);

  return;
}
```

Inmatning kapital, antalÅr

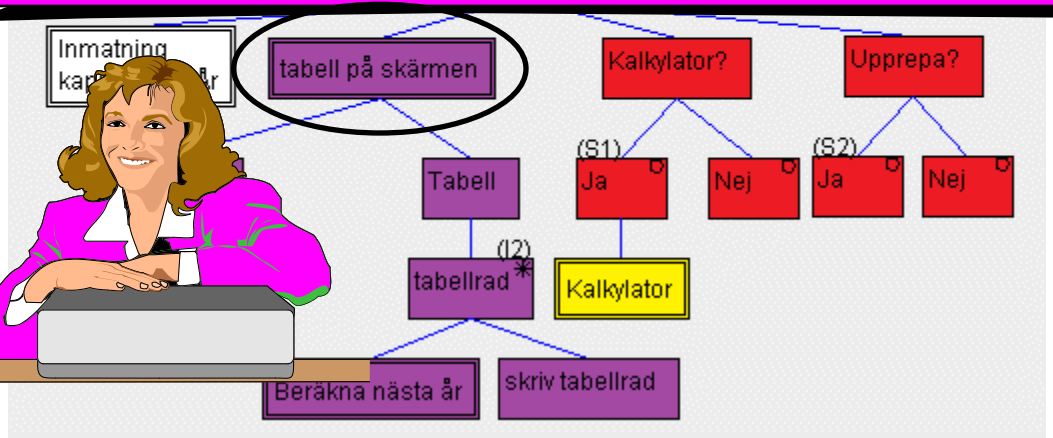
Tabellrubrik

Beräkna nästa år

Anders Sjögren as@kth.se

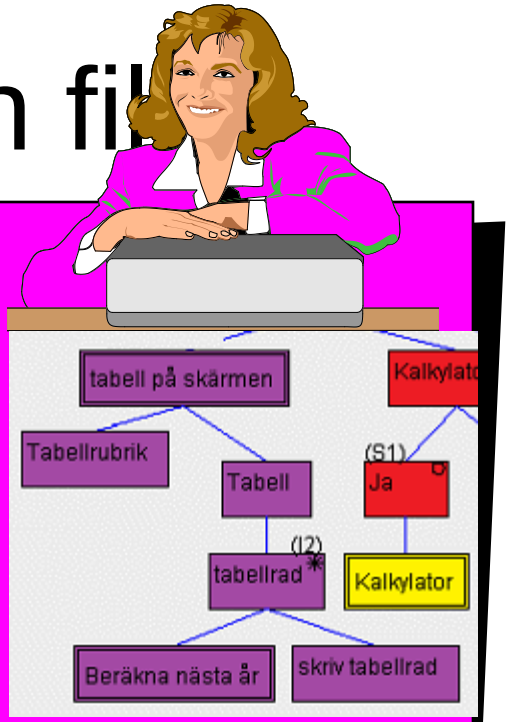
Arbetsfördelning och filer

```
/* tabell.h */
#include "ranta.h"
#include <stdio.h>

#if !defined(tabell_h)
  #define tabell_h

    #define ABS( x )( (x) > 0 ? (x) : -(x) )
  void TabellPaSkarmen( double kapital, int antalAr );

#endif
```



Anders Sjögren as@kth.se

Arbetsfördelning och fil



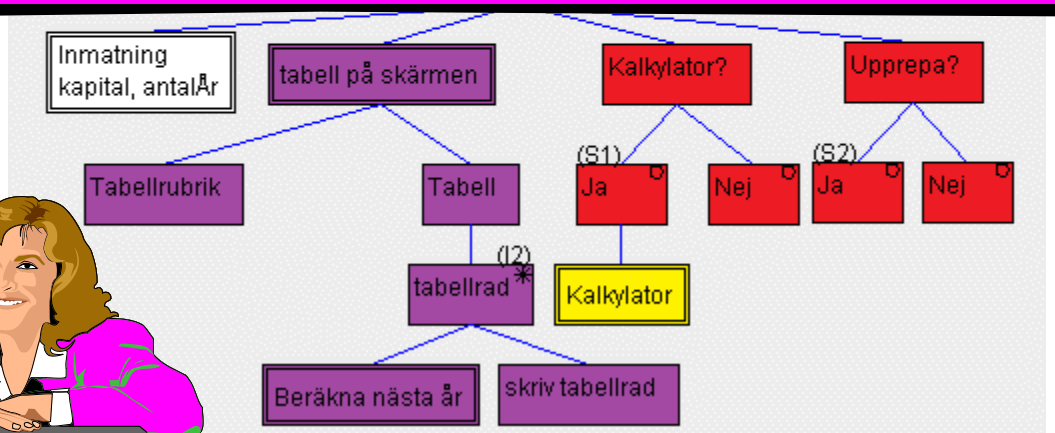

```
#include "tabell.h"

static double NastaAr( double kapital );

void TabellPaSkarmen( double kapital, int antalAr   )
{
  int ar ;

  printf("\n År     Saldo\n ==     =====\n");
  for ( ar = 1; ar <= antalAr; ar++ ) {
    kapital = NastaAr( kapital );

    /* enheter i tabellen */
    if ( -10 < kapital && kapital < 10 )
      printf("%3d%11.2f kr\n", ar, ABS( kapital ));
    else if ( -100 < kapital && kapital < 100 )
      printf("%3d%11.2f da(deka)kr\n", ar, (ABS( kapital ))/10);
    else if ( -1000 < kapital && kapital < 1000 )
      printf("%3d%11.2f h(hekto)kr\n", ar, (ABS( kapital ))/100);
    else
      printf("%3d%11.2f kkr\n", ar, (ABS( kapital ))/1000);
  }
  return;
}
```

Forts ...

Arbetsfördelning och filer

Forts ...

```
static double NastaAr( double x ) {

  if ( x > 0 )
    x = x * ( 1 + ranteFaktor );          /* denna */
  else  x = x * 1/( 1 + RANTESATS/100 ); /* eller denna */

  return x ;
}
```



Anders Sjögren as@kth.se

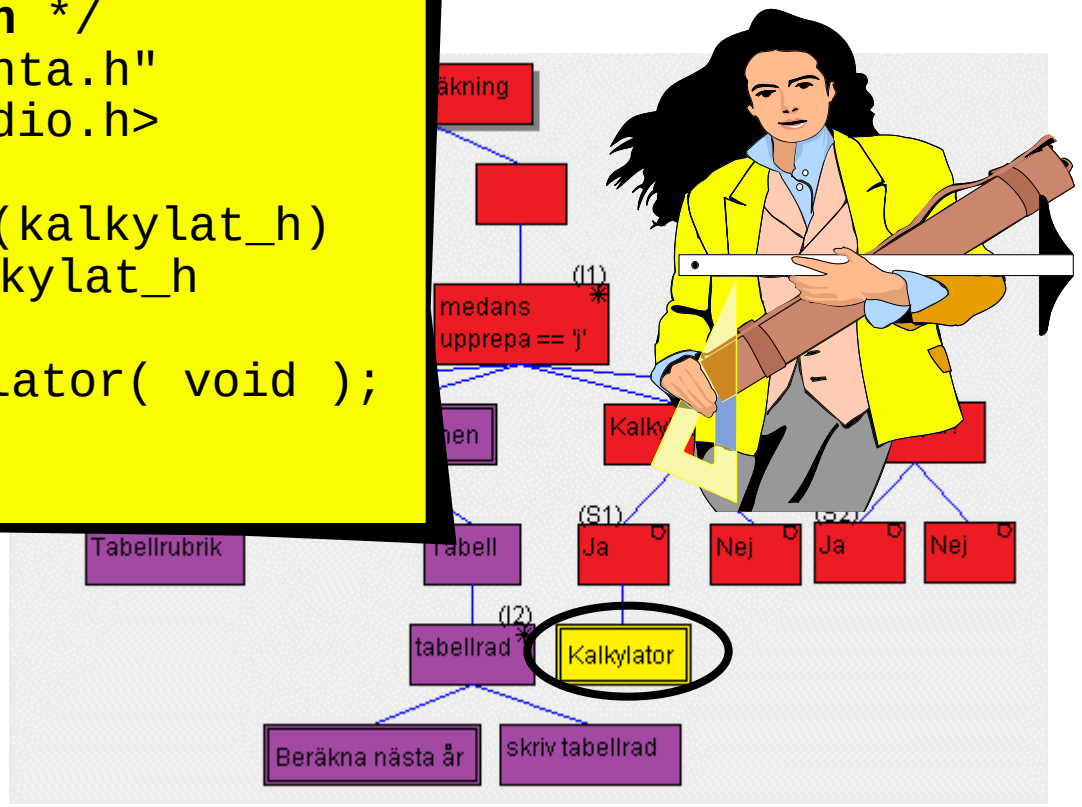
Arbetsfördelning och filer

Forts ...

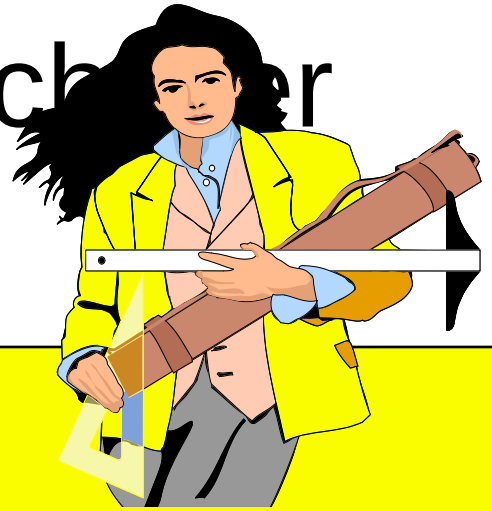
```
static double NastaAr( double x ) {

  if ( x > 0 )
    x = x * ( 1 + ranteFaktor );          /* denna */
  else  x = x * 1/( 1 + RANTESATS/100 ); /* eller denna */

  return x ;
}
```

Arbetsfördelning och filer

```
/* kalkylat.h */
#include "ranta.h"
#include <stdio.h>

#if !defined(kalkylat_h)
  #define kalkylat_h

  void Kalkylator( void );

#endif
```

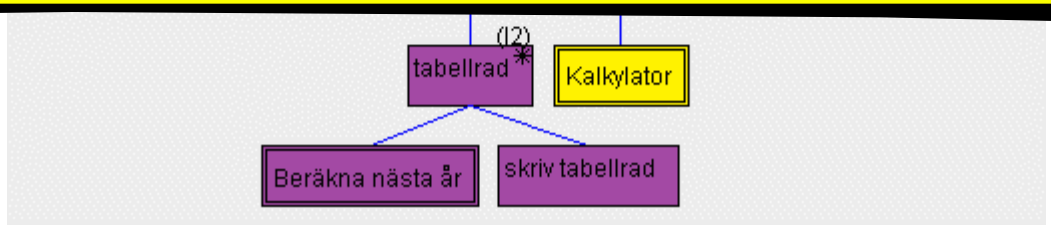


Arbetsfördelning och

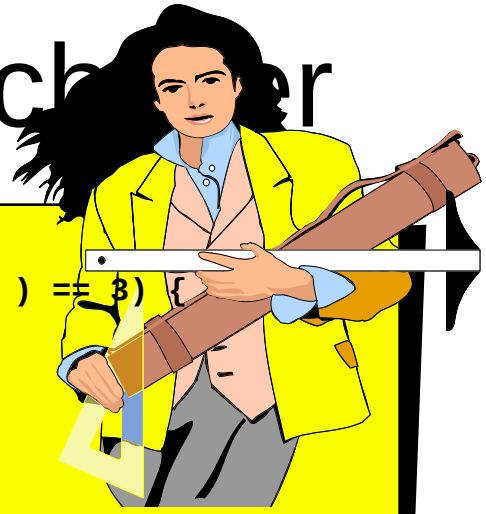


```
/* kalkylat.c */
#include "kalkylat.h"

void Kalkylator( void )   /* Enkel kalkylator */
{
   float x, y;
  char c;

  printf( "\nKalkylator som klarar de fyra räknesätten t ex ");
  printf("3+2\n");
  printf( "A, avslutar\n");
```

forts



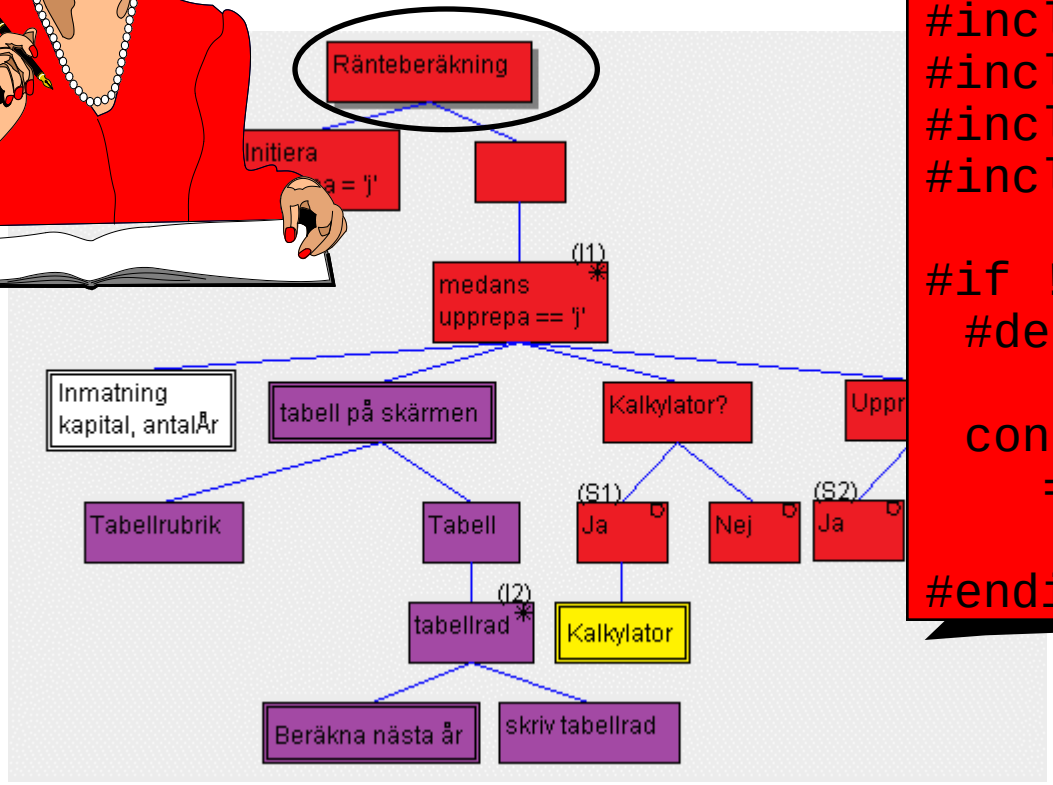
Arbetsfördelning och ...er



forts

```
while (printf("-->"), scanf("%f%c%f", &x, &c, &y ) == 3) {
  switch(c) {
      case '+':
        printf("%f\n", x + y);
      break;
      case '-':
        printf("%f\n", x - y);
        break;
    case '*':
        printf("%f\n", x * y);
     break;
    case '/':
        if (y != 0)
          printf("%f\n", x / y);
        else
           printf("Division med noll\n");
        break;
      default:
        printf("Felaktig operator\n");
        break;
    }
  }
  scanf("%*s");
  return;
}
```

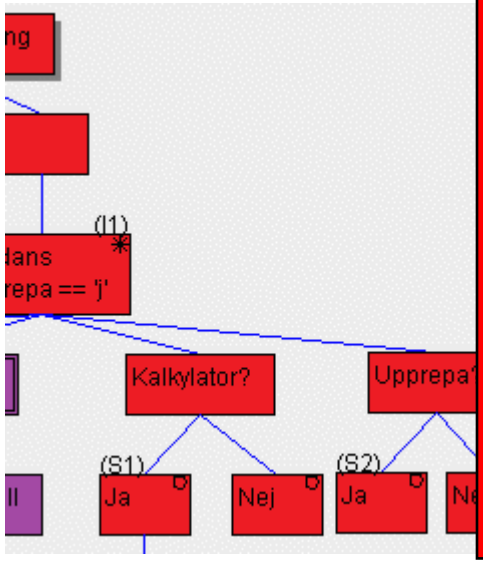
Arbetsfördelning och filer




```
/* main.h */
#include "ranta.h"
#include "inmatnin.h"
#include "tabell.h"
#include "kalkylat.h"
#include"ranta.h"
#include <stdio.h>

#if !defined(main_h)
  #define main_h

  const double ranteFaktor
     = RANTESATS / 100;

#endif
```

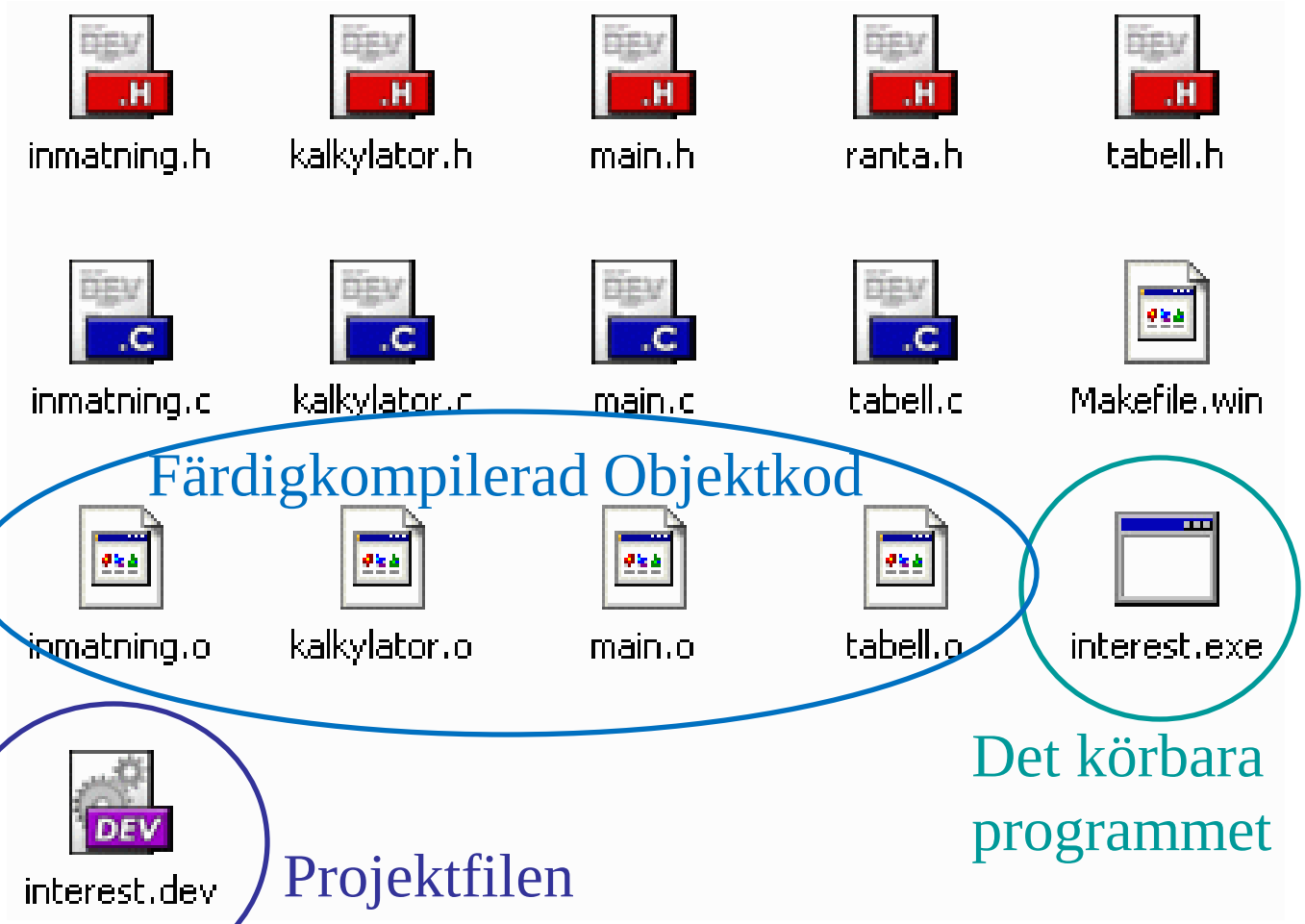
Anders Sjögren as@kth.se

Arbetsfördelning och filer

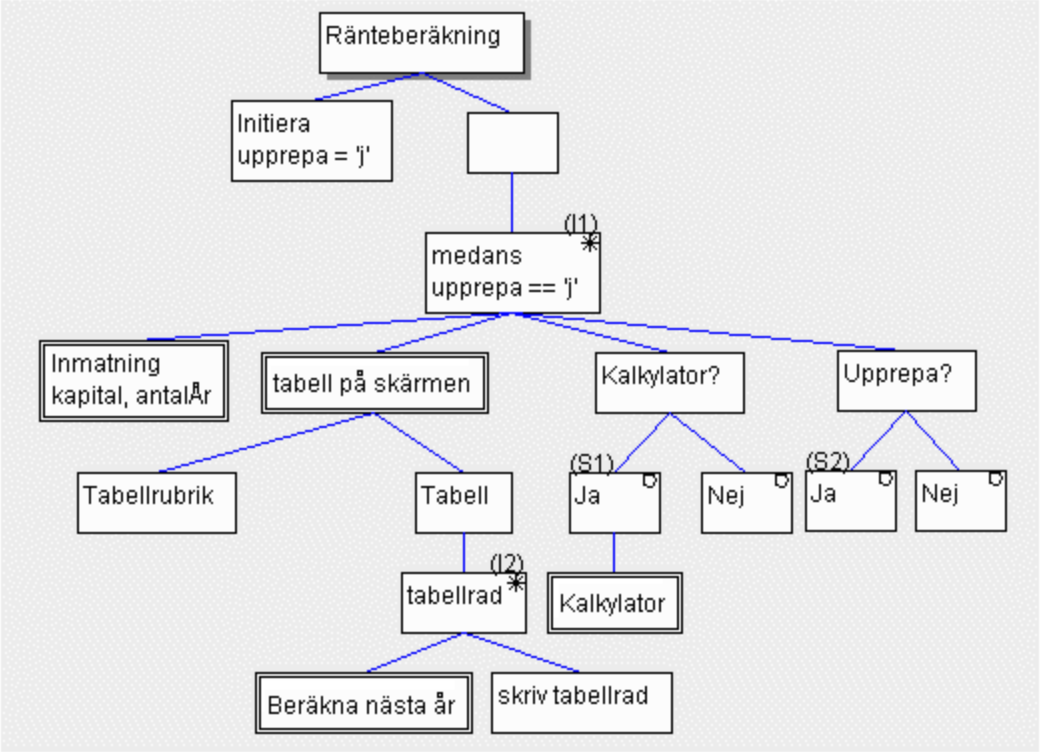



```
/* main.c */
#include "main.h"

int main( void ) {

  char   kalkylera, upprepa = 'j' ;
  int  antalAr;
  double   kapital;

  while ( upprepa == 'j' || upprepa == 'J' ) {
    Inmatning( &kapital, &antalAr );
    TabellPaSkarmen( kapital, antalAr );

    printf("\nÖnskas kalkylator? ( j/n ) --> ");
    scanf(" %c", &kalkylera );
    if ( kalkylera == 'j' || kalkylera == 'J' )
      Kalkylator();

    printf("\nUpprepa programmet? (j/n) --> ");
    scanf(" %c", &upprepa);
  }
  printf("\nSLUT");
  return 0;
}
```

Filer i projektet



Bygg projektet själv



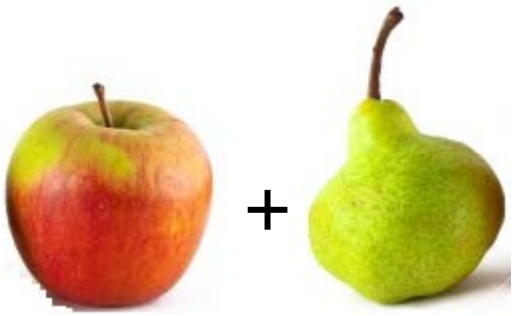
Alla källkodstexter finns på ID120V kurswebben …

`http://www.ict.kth.se/courses/ID120V/files/funktioner/project.htm`

William Sandqvist william@kth.se

Typecast av variabler

Man kan ju ***inte*** lägga ihop äpplen med päron – men C gör i så fall det bästa (?) möjliga av situationen.



C gör automatiska typomvandlingar :

double int float int

```
d = i + f/100
```

Typomvandling implicit (automatisk)

```
d = i + f/100
```

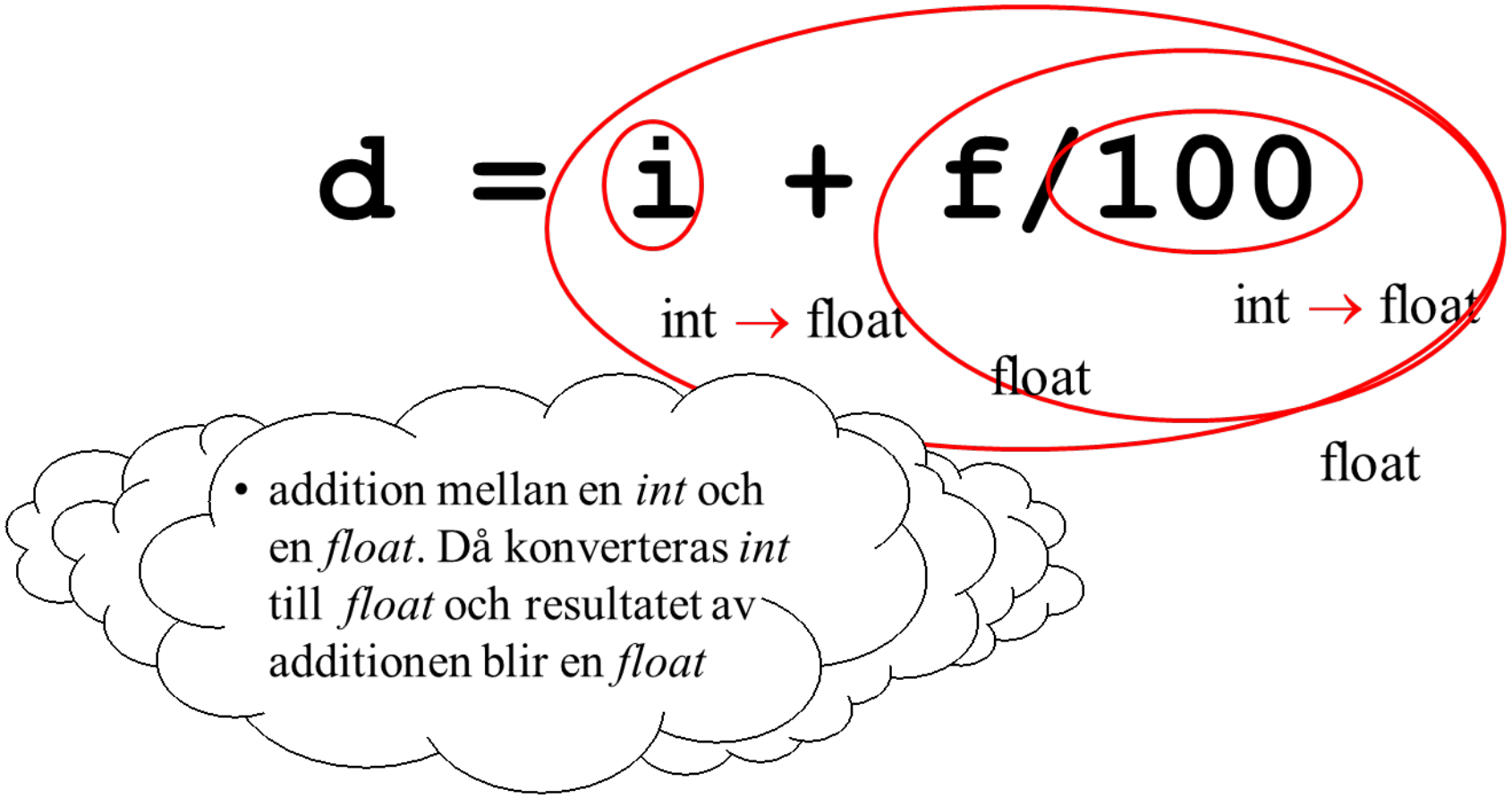
int → float

float

division mellan en *float* och en *int*. Då konverteras *int* till *float* och resultatet av divisionen blir en *float*

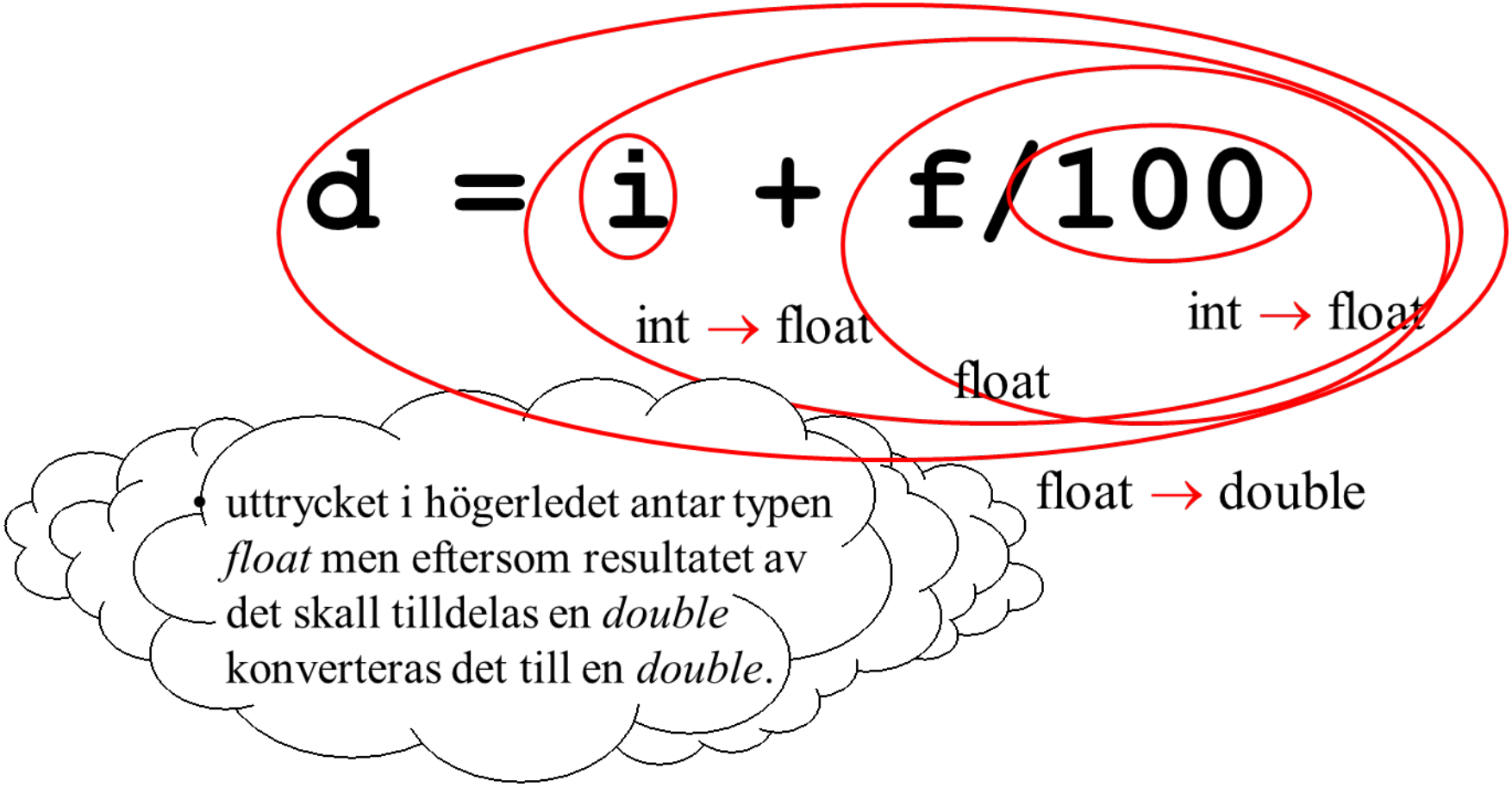
William Sandqvist william@kth.se

Typomvandling implicit (automatisk)



William Sandqvist william@kth.se

Typomvandling implicit (automatisk)



William Sandqvist william@kth.se

Typomvandling implicit (automatisk)

```
long double
double
float
unsigned long int
long int
unsigned int
int
unsigned short int
short int
unsigned char
signed char
```

högsta typ

lägsta typ

typomvandlingar där programmet får ”välja typ självt”, sker typomvandling från ”lägre” typ till ”högre” typ

William Sandqvist william@kth.se

Typomvandling

- alla typer som är av tal-typ (skalär typ) kan konverteras mellan varandra
- man bör fundera och kontrollera vad som händer vid typomvandling, följand gäller dock
  - från flyttal till heltal
    - om talet ryms (till storlek) så trunkeras det dvs **avhuggning av decimaldel** sker, ingen avrundning
  - från heltal till fyttal
    - **precisionsförlust**, det är inte säkert att flyttalsformen kan lagra alla värdesiffror

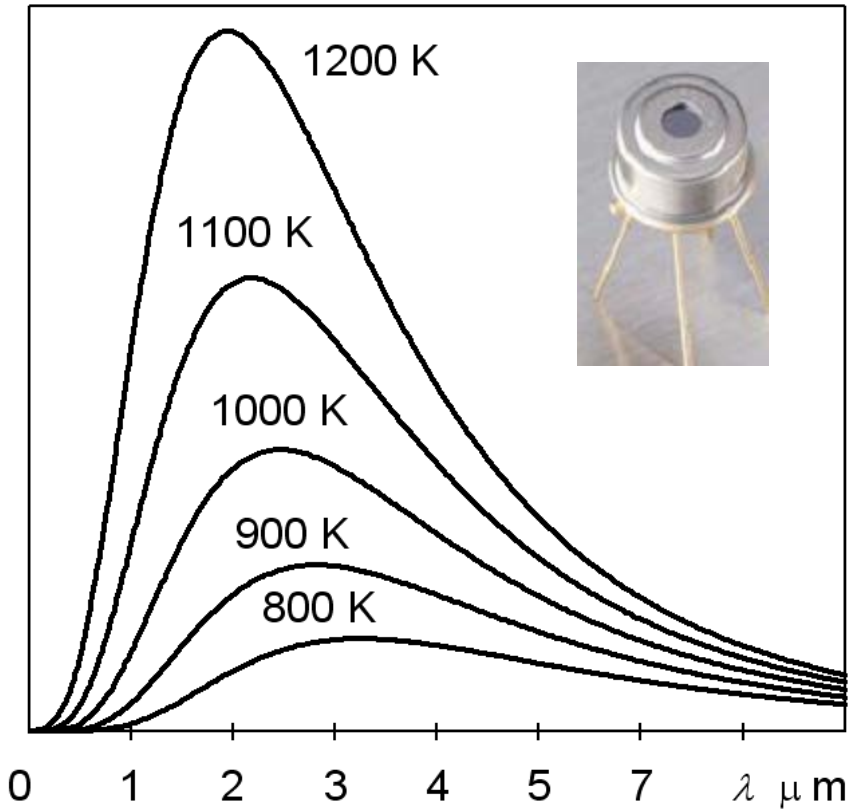
Typomvandling

explicit (av programmeraren påkallad, cast)

(*typnamn*) *uttryck*

Praktiskt exempel. En Strålningstermometer är ansluten till en 10 bitars AD-omvandlare. Mätvärdet måste justeras med hjälp av C:s matte-funktioner.

Strålningstermometer, Plank's lag



Totalstrålningens temperaturberoende.

Ytan under kurvorna är proportionell mot absoluta temperaturen T^4.

Temperaturen får man således genom att beräkna

$\sqrt[4]{\ }$ ur mätvärdet.

William Sandqvist william@kth.se

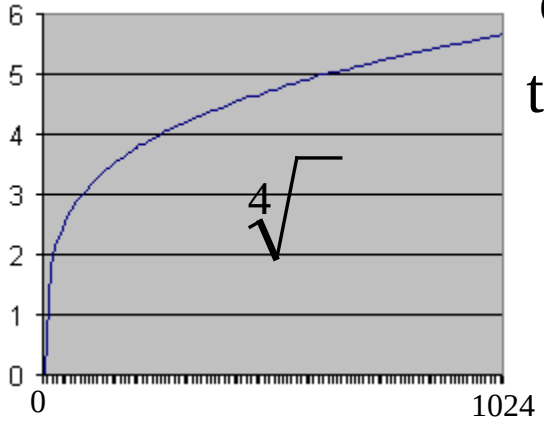
AD-omvandlare 10 bitar

$b_9 b_8 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$

Strålningstermometer

AD-omvandlare
10 bitar
1024 steg
5 mV/steg
Upplösning ≈0,1%

Linjärisering



AD

0 … 1024

`short int`

`double`

temp T

0 … 6.0

Offset och skalning

`-`

`*`

Så att 0.0° och 100.0° stämmer!

William Sandqvist william@kth.se

```
#include <math.h>
short int ad_value;
double temperature;
double offset = 3.14;
double gain = 51.4;

temperature = pow( (double) ad_value, 0.25 );
temperature -= offset;
temperature *= gain;
```

C:s matematikfunktioner

linjärisering
dra fjärde roten ur mätvärdet

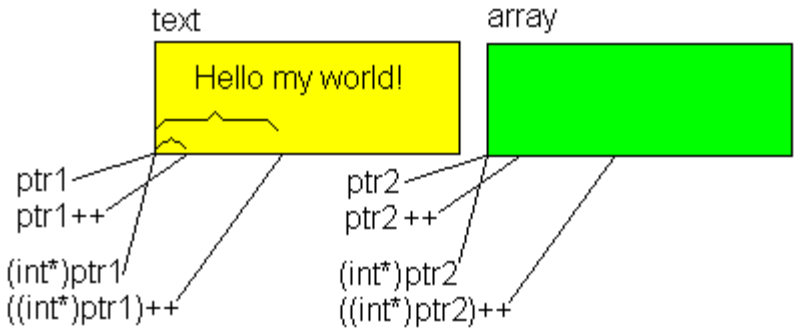
kanske justering av 0-nivå
och omskalning

Alla C:s matematik-funktioner använder flyttal med dubbel precision. Genom att typecasta visar vi vilken omvandling vi önskar.
(Vad hade C:s implicita omvandlingsregler gett för resultat denna gång?)

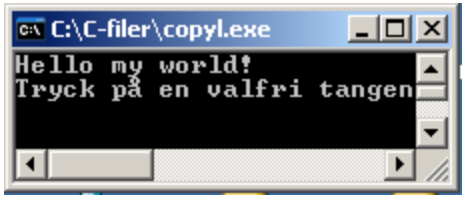
Typecast av pekare

Vi kan kopiera 16 bokstäver som 4 int – detta går snabbare (om datorn har 32 bitars buss).

```
#include <stdio.h>
int main(void)
{
  char text[16]="Hello my world!";
  char array[16];
  char * ptr1 = text;
  char * ptr2 = array;
  int i;
  for( i=0;i<4;i++)
   {
     *( ((int*)ptr2)++ ) = *( ((int*)ptr1)++ );
   }
  printf("%s\n",array);
  system("PAUSE");
  return 0;
}
```



Pekarna *typecastas* till int*, och då ökar steget från en Byte till 4 Byte!



C:s biblioteksfunktioner

C:s biblioteksfunktioner är ***generellt*** skrivna och använder därför oftast **void-pekare** som parametrar och för returvärden.

Som exempel tar vi funktionen **quicksort** som storleks-sorterar element i en array för att visa hur man använder C:s biblioteksfunktioner.

Sir C.A.R Hoare

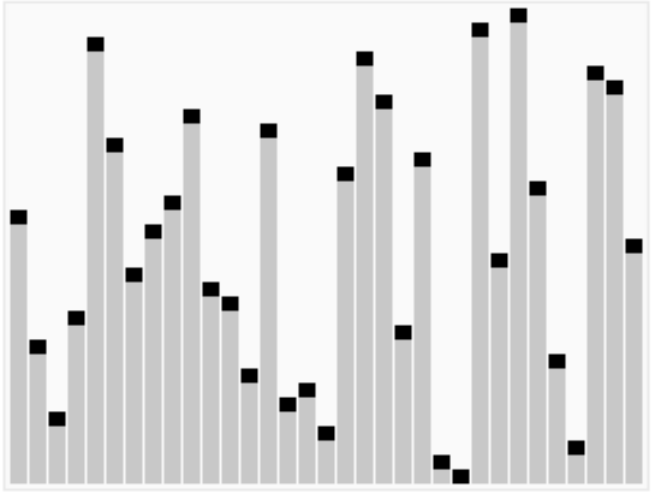
Turing award

William Sandqvist william@kth.se

Sortera element i en array

`qsort()`

`i stdlib.h`



Wikipedia Quicksort

William Sandqvist william@kth.se

Sorteringsfunktion Quick Sort

Alla programmeringsspråk har inbyggda sorteringsfunktioner.
C har funktionen **`qsort()`** som sorterar dataposterna i en array (eller structarray) i enlighet med en **sorteringsordning** som man själv specifierar.

Exempel, en array med fem bokstäver:

```
char alfa[5] = { 'C', 'E', 'A', 'D', 'F', 'B' };

qsort( (void*) alfa, 5, sizeof( char ), SorteringsOrdning );
```

Sorteras till: A, B, C, D, E, F om funktionen **`SorteringsOrdning()`** jämför alfabetiskt!

Pekare till arrayen

Antal element i arrayen

Storlek på elementen i Bytes

Funktionspekare, namnet på den sorteringsfunktion som skall användas

Sorteringsfunktionen

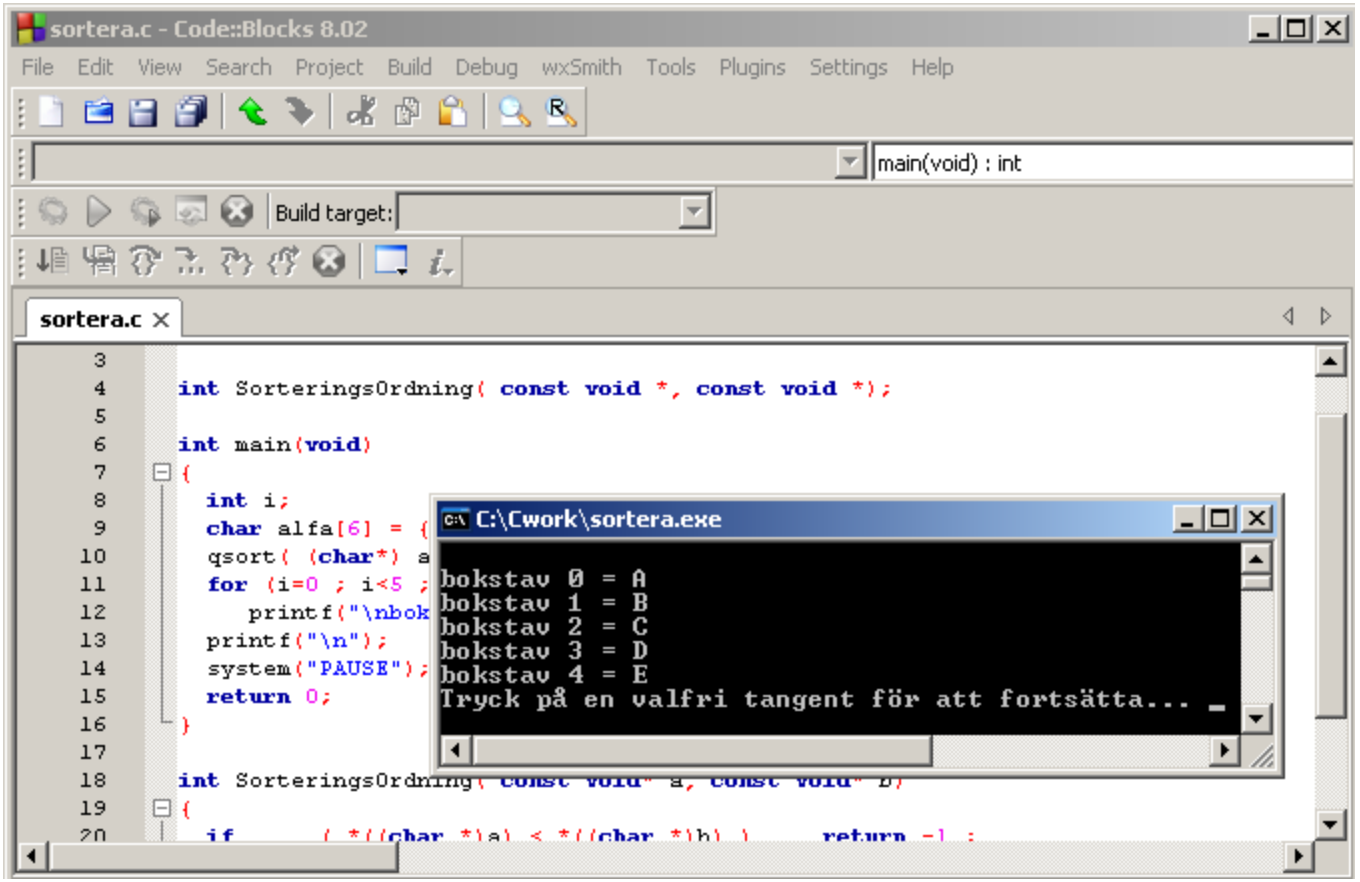
```
int Sorteringsordning( const void * a, const void * b)
{
  if ( *((char *)a) < *((char *)b) )      return -1 ;
  else if ( *((char *)a) > *((char *)b) ) return  1 ;
  else                                    return  0 ;
}
```

QSORT arbetar med void-pekare, och kan därför användas till att sortera vad som helst i arrayer. Sorteringsfunktionen, som man skriver själv och som QSORT sedan använder, skall alltid returnera -1, +1, eller 0 om parameter **b** är före, efter, eller samma som parameter **a**. På så sätt kan man sortera arrayen efter valfri egenskap.

```
#include <stdio.h>
#include <stdlib.h>

int SorteringsOrdning( const void *, const void *);

int main(void)
{
  int i;
  char alfa[6] = { 'C', 'E', 'A', 'D', 'F', 'B' };

  qsort( (char*)alfa, 6, sizeof(char), SorteringsOrdning );

  for (i=0 ; i<5 ; i++)  // now in order?
     printf("\nbokstav %d = %c",i, alfa[i]);
  printf("\n");
  system("PAUSE");
  return 0;
}

int SorteringsOrdning( const void* a, const void* b)
{
  if      ( *((char*)a) < *((char*)b) )     return -1 ;
  else if ( *((char*)a) > *((char*)b) )     return  1 ;
  else                                      return  0 ;
}
```

Inte i alfabetisk ordning!

Nu alfabetiskt sorterat?



William Sandqvist william@kth.se

Ex. Medianfilter

Ett **medianfilter** tar hela tiden fram det till storleken **”mittersta”** värdet av löpande mätvärden. Man behöver då kunna storlekssortera mätvärden – kanske med **quick-sort**! Brus-störningar undertrycks extremt bra med ett sådant filter.

Ex. varje pixel har ersatts med **medianvärdet** av de pixlar som finns inuti en cirkel (eller kvadrat) med pixlet som centrum.

William Sandqvist william@kth.se

Egna datatyper

Eftersom storleken på C:s datatyper kan variera är det ganska vanligt att programmerare definierar om de datatyper som finns till egna namn – med mer exakt betydelse.

```
typedef unsigned long ULONG;
```

Från och med nu så kan en variabel vara av typen:

```
ULONG tal = 1234567890;
```

ALTERA har egna datatyper, Windows har egna …

struct elephant

Från Datortekniken ….

```
typedef struct
{
   float left;
   float right;
} oron ;

typedef struct
{
  char namn[13];
  int vikt;
  oron oronarea;
} elephant ;
```

Från och med nu finns det en datapost av typen

elephant

William Sandqvist william@kth.se

Delminnen

Så här når vi delminnen:

```
elephant dumbo;
elephant * pek = &dumbo;
dumbo.oron.right = 0.25;
pek->oron.left = 0.36;
```

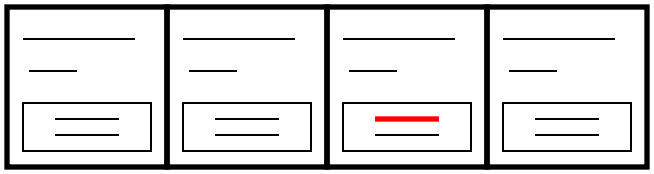
`.` punktoperator

`->` piloperator

Vektor `circus[]`

deklarerar en vektor, **`circus[]`**, med plats för fyra elefanter.

```
elephant circus[4];
```

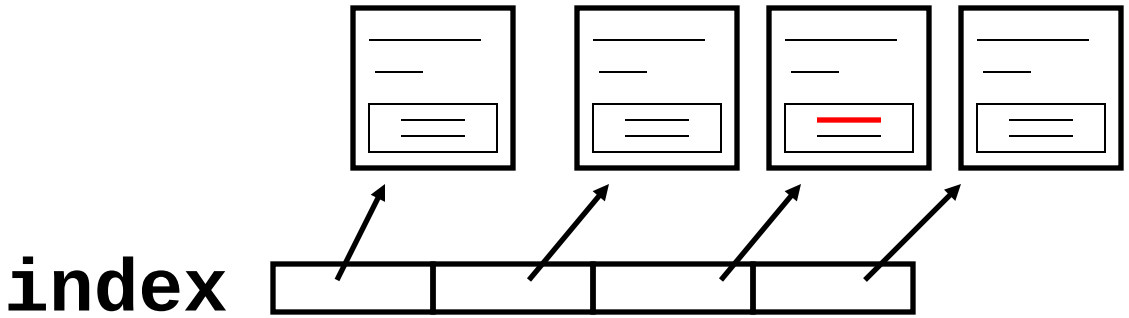
`circus` 

```
circus[2].oron.left = 0.56;
```

Indexarray **index[]**

deklarerar en vektor, **index[]**, med plats för fyra elefant-pekare.

```
elephant * index[4];
```

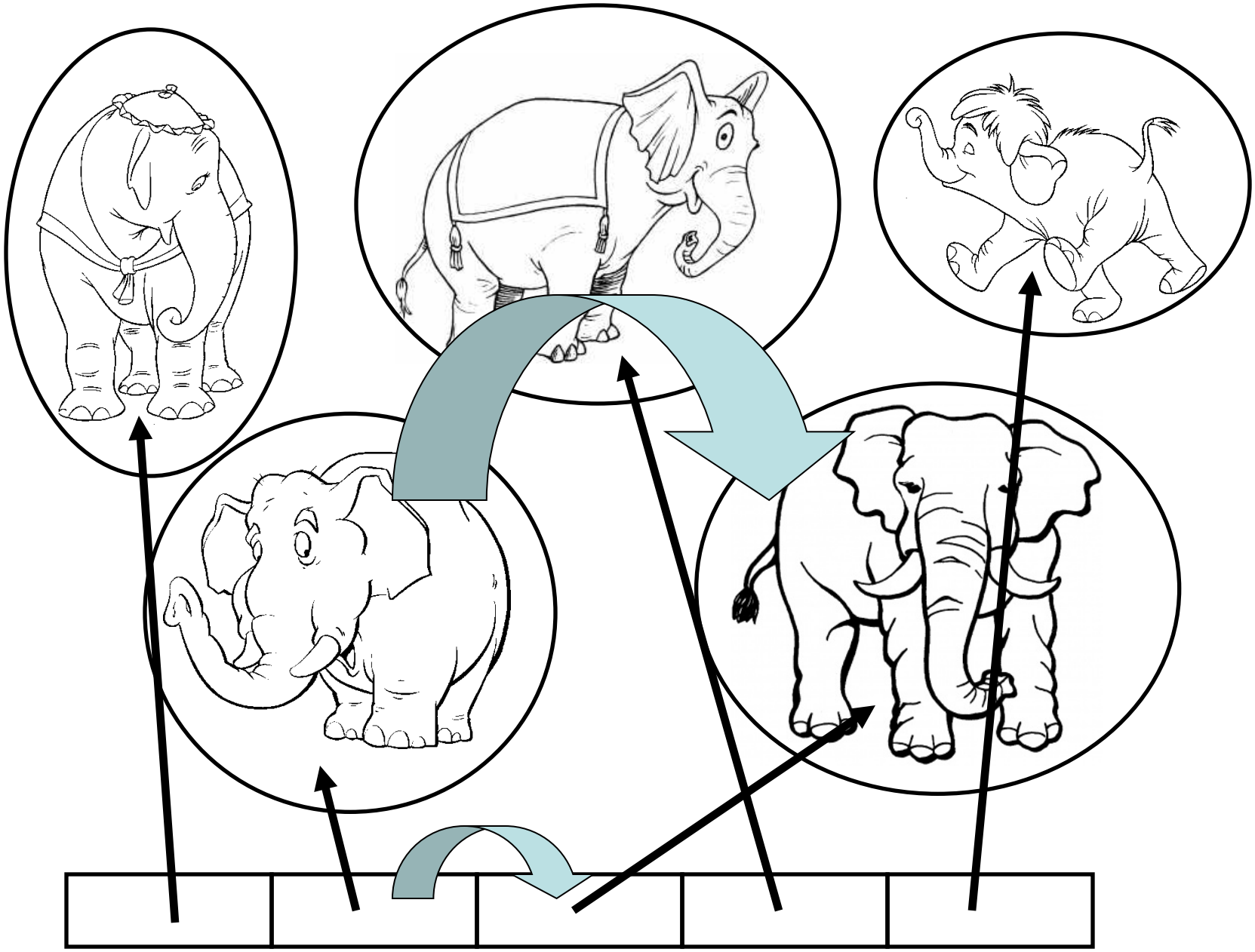



```
index[2]-> oron.left = 0.56;
```

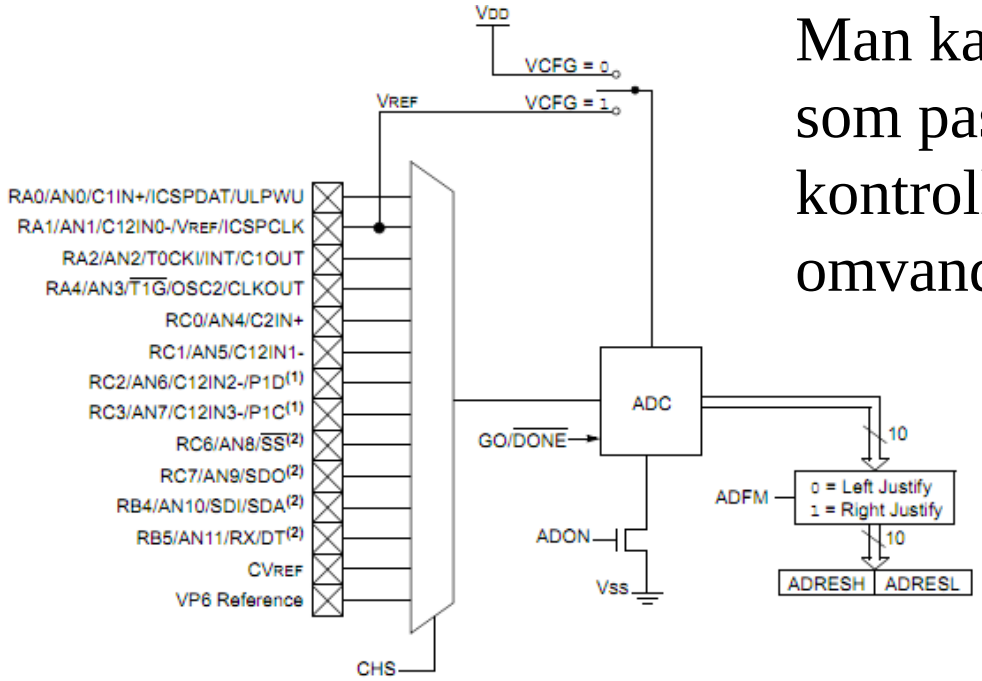
Hur sorterar man elefanter?

Genom att sortera pekarna – inte gärna genom att flytta på elefanterna!

William Sandqvist william@kth.se



Bitfield



Man kan skapa en datatyp som passar tex. ett kontrollregister för en AD-omvandlare.

REGISTER 9-1: ADCON0: A/D CONTROL REGISTER 0

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	VCFG	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

William Sandqvist william@kth.se

Bitfield

REGISTER 9-1: ADCON0: A/D CONTROL REGISTER 0

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	VCFG	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

```
typedef struct
{
  unsigned ADFM :1;
  unsigned VCFG :1;
  unsigned CHS  :4;
  unsigned GO   :1;
  unsigned ADON :1;
 } ADCON0 ;
```

```
ADCON0 tempinit;
tempinit.CHS = 9;
tempinit.GO = 0;
tempinit.ADFM = 1;
tempinit.VCFG = 0;
tempinit.ADON = 1;
```

Tänkbar användning:

```
void AD_init( ADCON0 init );
```

William Sandqvist william@kth.se

`volatile` ?

Periferienheter, I/O, ansluts ofta till en CPU som om dom vore minneskretsar (fast med bara ett fåtal ”minnesceller”). Ex. en realtidsklock-krets – håller reda på tid och datum. Den styrs/avläses från 8 inbyggda register.

Eftersom I/O-enheter *inte* är riktiga minnen – det kan verka som om innehållet kan ändras ”av sig självt” – så kan man vid programmeringen av processorn behöva ”hjälpa” kompilatorn att förstå detta genom att deklarera dem som **`volatile`**

(= flyktiga) i sina C-program.

Har till exempel processorn ett **cache-minne** är det viktigt att man läser klock-kretsen direkt och inte några *äldre* ”cachade” tider!

William Sandqvist william@kth.se

`const` ?

`const` innebär att en variabel inte går att modifiera under körningen (ingen tilldelning kan ske). Variabeln initieras därför när den definieras.
En pekare som är **`const *`** kan peka på olika adresser – och man kan då läsa på dessa adresser men inte skriva där.

`const` deklarationen gör att kompilatorn kan kontrollera att man inte uttryckligen ändrar en variabel någonstans i programmet.
En sådan variabel kan placeras i ett läsminne ROM.

`#pragma` ?

Några exempel på användningen av pragma – här gäller det en liten PIC-processor.

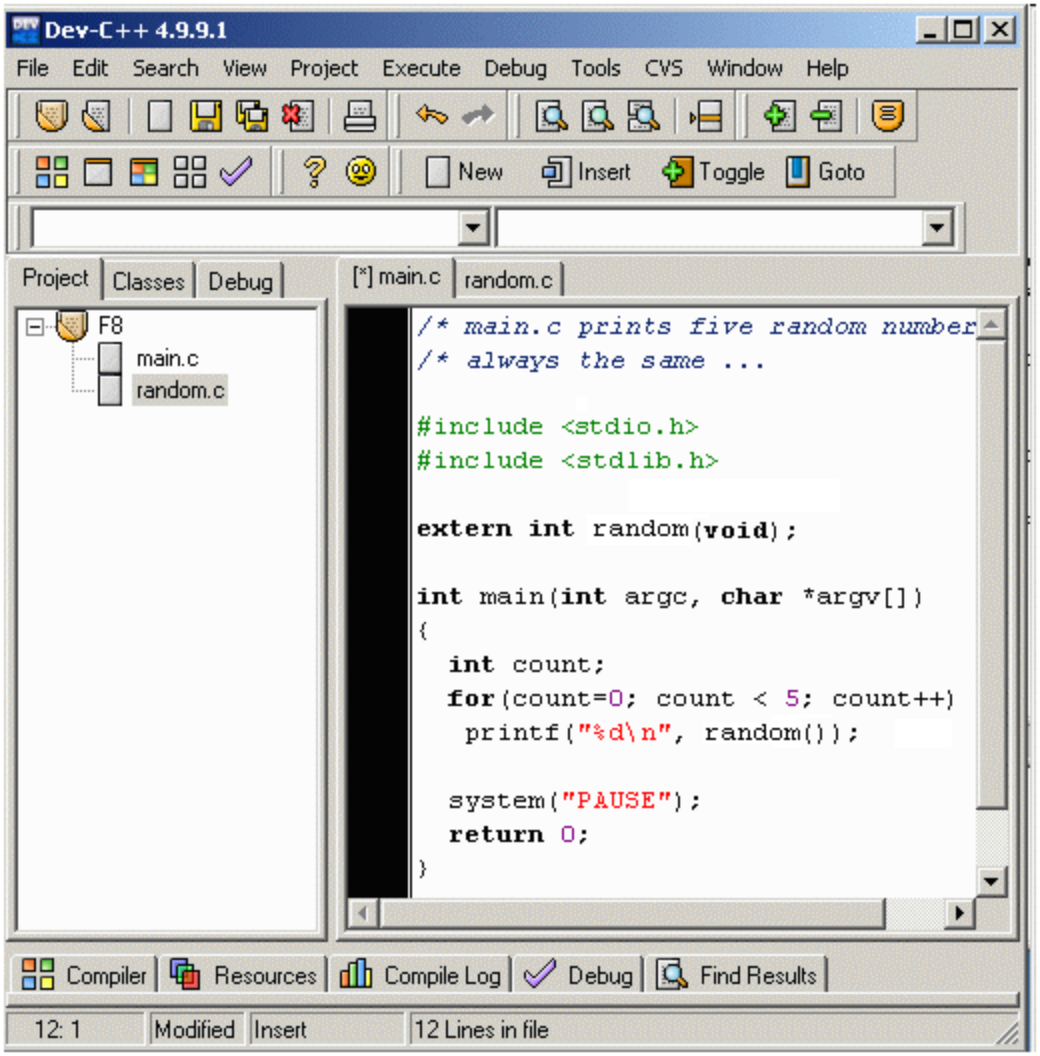
```
#pragma config |= 0x3fb0

#pragma bit lightdiode @ PORTB.0
```

En pragmatisk person är en som låter sitt handlande styras av vad som är praktiskt hellre än vad reglerna föreskriver.

Med Preprocessorkommandot **`#pragma config |= 0x3ff1`** förs information om vilka inställningar som man ska använda vid själva Krets-programmeringen över till krets-programmeraren.
Preprocessorkommandot **`#pragma`** betyder att man gjort en sådan pragmatisk avvikelse från ANSI-C standarden.
`#pragma bit lightdiode @ PORTB.0` betyder att man infört en helt ny datatyp, bit, bitvariabel (som kan rymma 1 eller 0). Vi ger därefter bitvariabeln namnet **`lightdiode`** och tecknet **`@`** betyder att variabeln har en fast adress (kompilatorn får inte placera den någon annanstans).

William Sandqvist william@kth.se

Användningen av **`static`**



William Sandqvist william@kth.se

Lagringsklasser och deklarationsområde

Exempel på användning av lagringsklassen static. Portabla slumptal (Pseudorandom numbers). Exemplet från **C Primer Plus**, *Stephen Prata*, ISBN 1-57169-161-8.
Skapa ett projekt i Dev-C++ med två filer `main.c` och `random.c`.
Kompilera och provkör.

`/* main.c prints five random numbers */`

```
#include <stdio.h>
#include <stdlib.h>
extern int random(void);

int main(int argc, char *argv[])
{
  int count;
  for(count=0; count < 5; count++)
    printf("%d\n", random());
  system("PAUSE"); return 0;
}
```

Slumpfunktionen

```
/* random.c produces random numbers */
/* uses ANSI C portable algorithm   */

static unsigned long int next =1;  /* the seed */

int random(void)
{
  /* magic formula to generate pseudorandom number */
  next = next * 1103515245 + 12345;
  return (unsigned int) (next/65536) % 32768;
}
```

Slumpfunktionen, problem

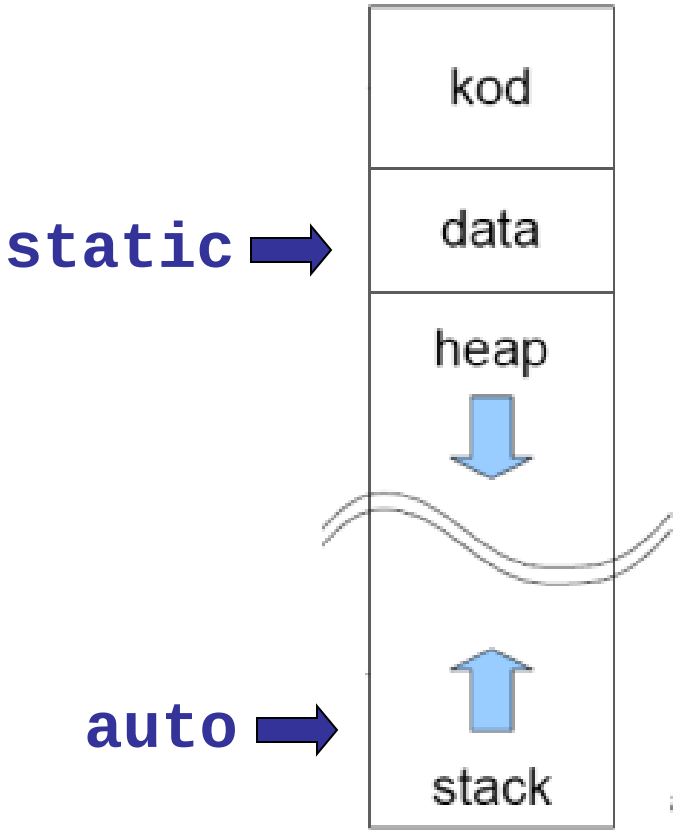
Vad skulle hända om next *inte* deklareras som static?

```
/* random.c produces random numbers */
/* uses ANSI C portable algorithm   */

int random(void)
{
  unsigned long int next =1;    /* the seed */

  /* magic formula to generate pseudorandom number */
  next = next * 1103515245 + 12345;
  return (unsigned int) (next/65536) % 32768;
}
```

Ooops!

William Sandqvist william@kth.se

static



En variabel som är **`static`** ”lever” för evigt (som en global) variabel, men den är bara känd i det block där den är definierad och kan bara nås där.

En lokal variabel i en funktion (**`auto`**) ”lever” bara så länge som funktionen körs.

Slumpfunktionen, mer problem

```
/* random.c produces random numbers */
/* uses ANSI C portable algorithm   */

static unsigned long int next =1;  /* the seed */
int random(void)
{
  /* magic formula to generate pseudorandom number */
  next = next * 1103515245 + 12345;
  return (unsigned int) (next/65536) % 32768;
}
```

Måndag

Tisdag

Onsdag

Samma slumptalsserie vid varje körning … (lösning! byt seed/Startvärde mellan körningarna)

William Sandqvist william@kth.se