# Mer om funktioner

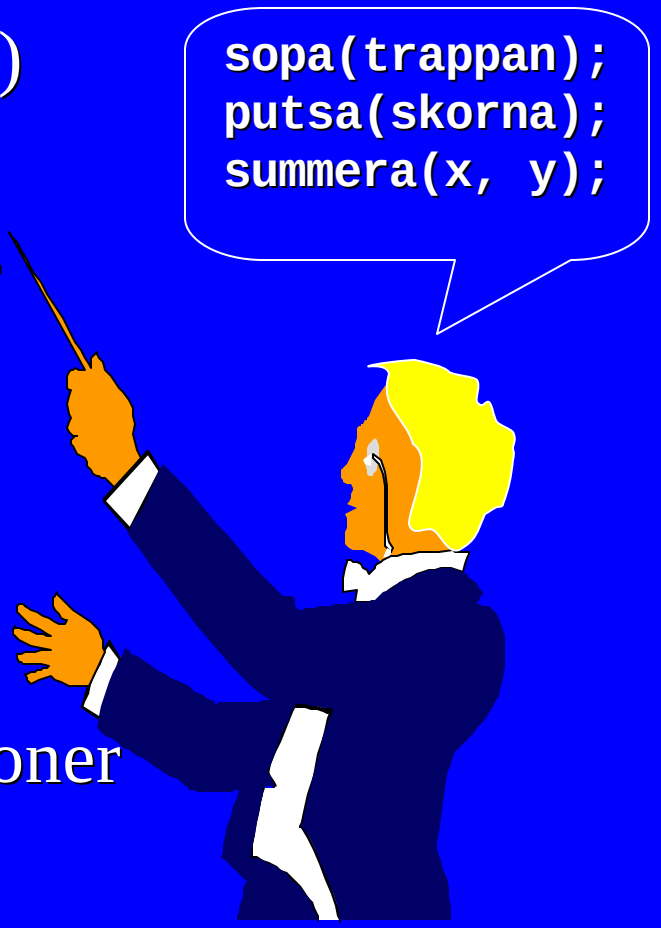
Arrayer som argument (parametrar)

Arrayer som returvärden?

Att skicka data i egna struct - typer
- som argument
- som returvärde

Uppbyggnad av ett
- strukturerat program med funktioner

Men först...

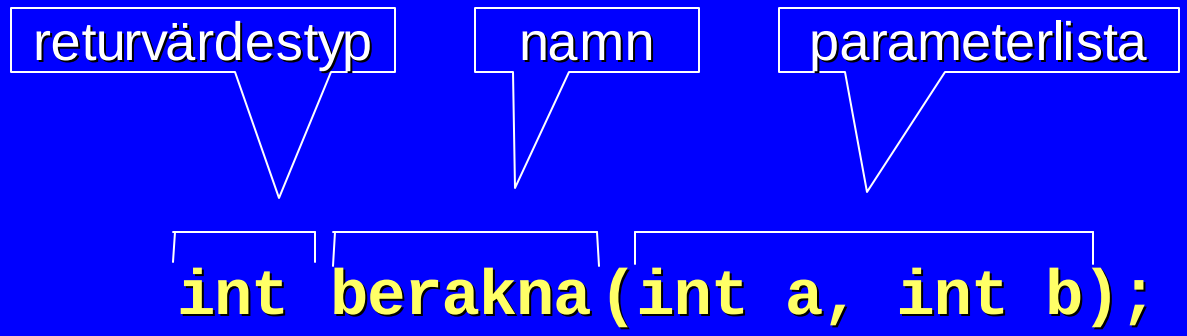


# Funktionsdeklaration

Vi minns:

Funktioner måste *deklareras*

Funktionshuvudet är funktionens *signatur*



# Funktionsdefinition

Man talar om vad funktionen skall göra

```
// Definitiona va funktionen
int berakna(int a, int b)
{
  int res = a * b;
  return res;
}
```

**kropp / body**

# Skicka parametrar

Funktionsanrop:

```
z = berakna ( x, y );
```

```
int berakna( int a, int b )
{
  int res = a * b;
  return res;
}
```

Värdena *kopieras* till funktionen

i den ordning de räknas upp iparameterlistan

# Tur & retur

Endast enkla variabeltyper kan överföras!

- En eller flera parametrar som argument

```
void jobbaMed( int a, float f );
```

- Endast ett returvärde kan överföras

```
float mataIn();
```

Datatyper som kan överföras

```
char    short
int     long        + unsigned-varianter
float   double      long double
```

# Arrayer som argument

Arraynamnet är en pekare ( adress till element 0 )

Pekaren skickas som argument

Detta kallas ibland *referensanrop*

Man skickar förstås pekarens *värde*.

```
float vikter[20], medelvikt;
for (i = 0; i < 20; i++)
   cin >> vikter[i];

//--- funktionsanrop ----
medelvikt = medelvarde(  vikter, 20 );
```

medel.cpp

Pekare till array

# Data kopieras inte

Vid *värdeanrop* får funktionen *en kopia av originaldata* att arbeta med.

Vid *referensanrop* får funktionen en *pekare till origialdata*

ACHTUNG! Funktionen kan ändra i originalet!

Både bra… :

- Man behöver inte kopiera stora arrayer på stacken
- Man kan ha en funktion som fyller på data i en array

..och dåligt:

Håll koll på funktionerna så de inte förstör originalet!

# Att fylla en array

Ett försök att fylla en array med värden

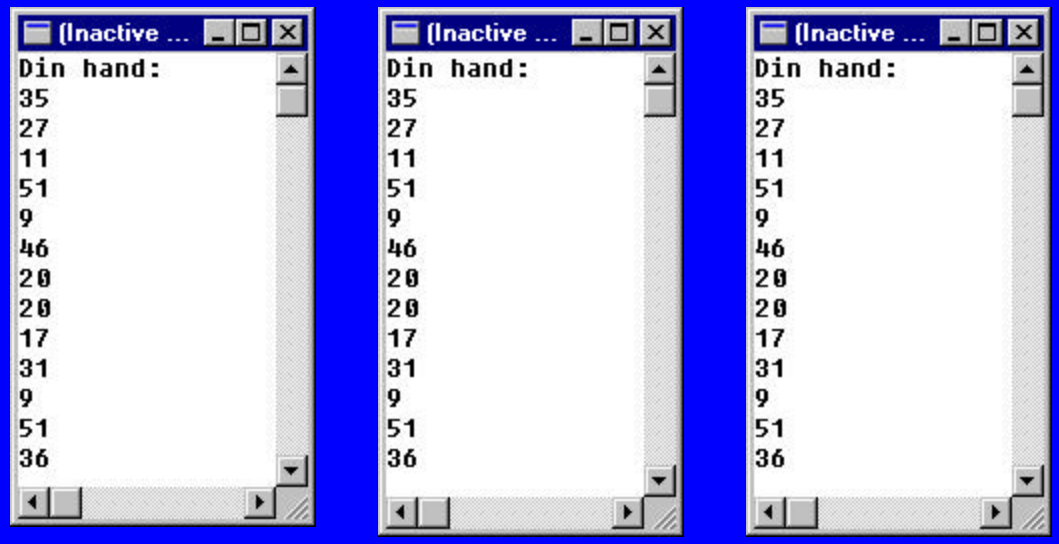
```
void deal(int [], int);      // deklaration
void main(void){
   int hand[13];
   deal( hand, 13 );
}

// funktionen slumpar ett av 52 kort
void deal( int array[], int antal ){
   for (int i = 0; i < antal; i++)
      array[i] = (rand()%52 + 1);
}
```

bigdeal.cpp

Vi måste skicka med antal

# Slumpa med `rand()`

## Resultat av körning





Pseudoslump!

Använd `randomize();`

# Slumpa starten

En pseudoslumpand funktion kan förmås att starta på olika ställen varje gång.

```
// funktionen slumpar ett av 52 kort
int* deal(int array[], antal){
   randomize();
   for (int i = 0; i < antal; i++)
      array[i] = (rand()%52 + 1);
    return array;
}
```

# Att returnera en array - fel

baddeal.cpp

```
const int antal = 13;
void main(void){
   int hand[antal]; int*  pek = hand;
   pek = deal();
    cout << "Din hand: " <<  endl;
    for (int i = 0; i < 13; i++)
       cout << hand[i]<<  endl;
}

// funktionen slumpar ett av 52 kort
int* deal(){
      int array[antal];
      for (int i = 0; i < antal; i++)
            array[i] = (rand()%52 + 1);
   return array;
}
```

Lokal array

# Att returnera en array på rätt sätt

Nytt och bättre försök - skicka med arrayen

```
const int antal = 13;
void main(void){
   int hand[antal];
   int* pek = hand;
   pek = deal(pek, antal);
    cout << "Din hand: " <<  endl;
    for (int i = 0; i < 13; i++)
      cout << hand[i]<<  endl;
}
```

Skicka med pekaren

# Pekaren returneras

…och här är funktionen

**gooddeal.cpp**

```
// funktionen slumpar ett av 52 kort
int* deal(int array[], antal){
   for (int i = 0; i < antal; i++)
      array[i] = (rand()%52 + 1);
    return array;
}
```

**Returnera pekaren**

Funktionen fyller på värden och returnerar samma pekare!

Vanligt vid t.ex. strängfunktioner

# Mät en sträng igen

Här är funktionen:

```
int stringlen( char* pek){
   int count  = 0;
    while(*pek != '\0')
      pek++, count++;
    return count;
}
```

Komma-operatorn ’,’ ( används sällan )

# Anrop av **stringlen**

stringlen.cpp

```
//Deklaration av funktionen
int stringlen( char* );
int main( void ){
   char text[81];
    int len;
    cout << "Mata in en textsträng :" <<  endl;
    cin.getline(text, 81);

      // Anropa stringlen
    len = stringlen(text);
    cout << "Din text var " << len
         << " tecken lång!" <<  endl;
}
```

# Returnera en sträng

Ett program som ser till att första tecknet
i ett inmatat ord blir Versal - resten gemena.

Data?

En vektor av typen char för inmatning

En vektor av typen char för resultatet

Funktionen? Vad gör den? Hur ser den ut?

1. Tar emot en sträng
2. Sätter första tecknet till versal
3. Går igenom resten av strängen och gör gemener
4. Returnerar strängen

`toupper`

`tolower`

# Funktionen toName

```
// Definition av  toName
char* toName( char* rad){ //  teckenpekare
  int len = strlen(rad);
  if (len == 0)
     return rad;

   rad[0] = toupper(rad[0]); //  Första tecknet

   for (int i = 1; i < len; i++){
     rad[i] = tolower(rad[i]);
   }
   return rad;
}
```

Testa - returnera direkt

# Inmatning och anrop

```
// Deklaration av funktioner
char* toName( char* ); // returnerar char*

void main(void){
   char inrad[81];
   cout << "Mata in en rad --> " <<  endl;
    cin.getline(inrad, 81);
    // Behandla strängen
   toName(inrad);
   cout << "det blev så här: " <<  inrad;
}
```

toname.cpp

# Bevara indata

Vid användningen av toName förändrades indata.

Hur ändra funktionen så att den bevarar **`inrad`** ?

```
// Deklarera två teckenvektorer
char inrad[81], utrad[81];
```

- Skicka med båda pekarna till **`toName()`**
- Låt **`toName()`** lägga ändrade tecken i utrad
- Glöm inte att lägga till **`'\0'`** i slutet av **`utrad`**
- returnera **`utrad`** (om du vill :)

# Räkna med vektorer

Beräkna ett antal ytor där höjder och bredder matas in.

Data: två vektorer för indata

en vektor för utdata - båda av flyttalstyp (double)

ett heltal som anger maxantal ytor

Funktion: beräknar ytorna och lägger resultat i utdatavektorn.

# Ytberäkning

```
float* beraknaYtor(float* b, float* h,
                   float* y, int no){
  for (int i = 0; i <  no; i++)
     y[i] = b[i] * h[i];
   return y;
}
```

# Mata in och beräkna

**Med:**

```
float bredder[MAX_ANTAL],
   hojder[MAX_ANTAL], ytor[MAX_ANTAL];
```

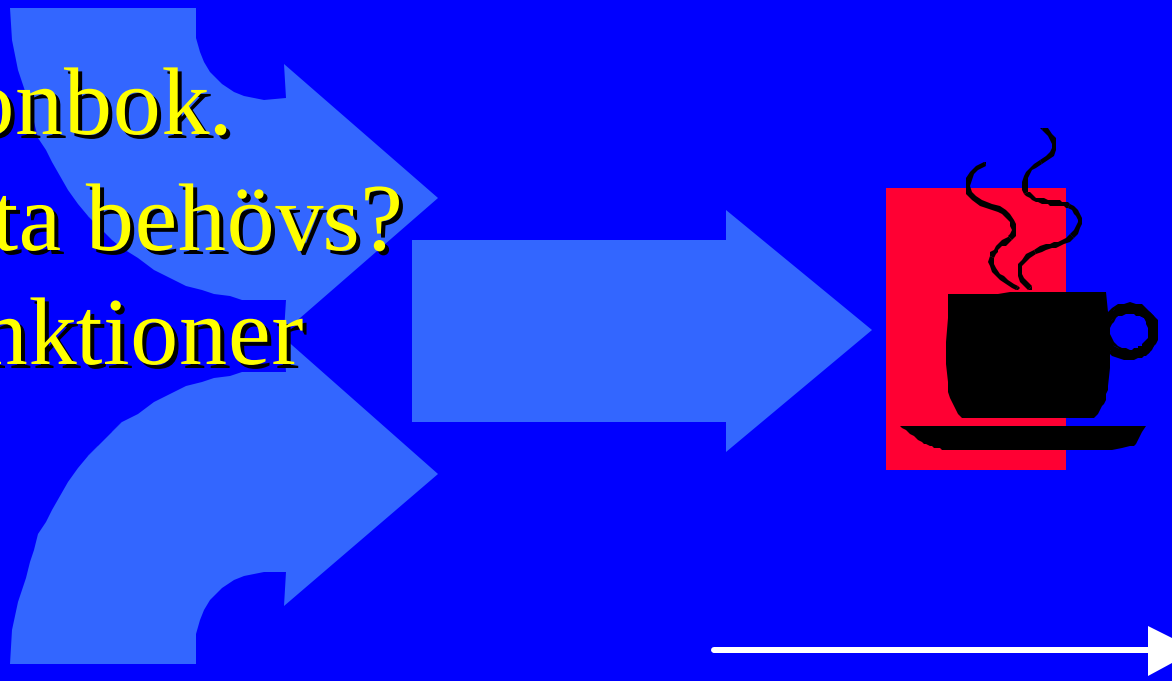
**görs anropet:**

```
y = beraknaYtor(bredder, hojder, ytor, antal);
```

**ytberakning.cpp**

# Skizz till en enkel databas

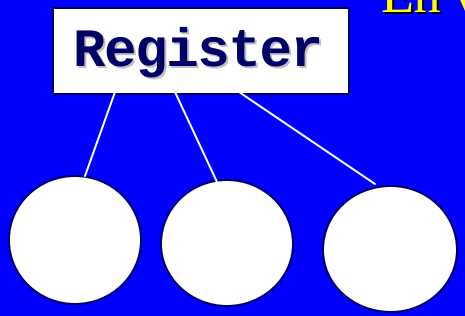
En telefonbok.
Vilka data behövs?
Vilka funktioner



# Skizz till en enkel databas

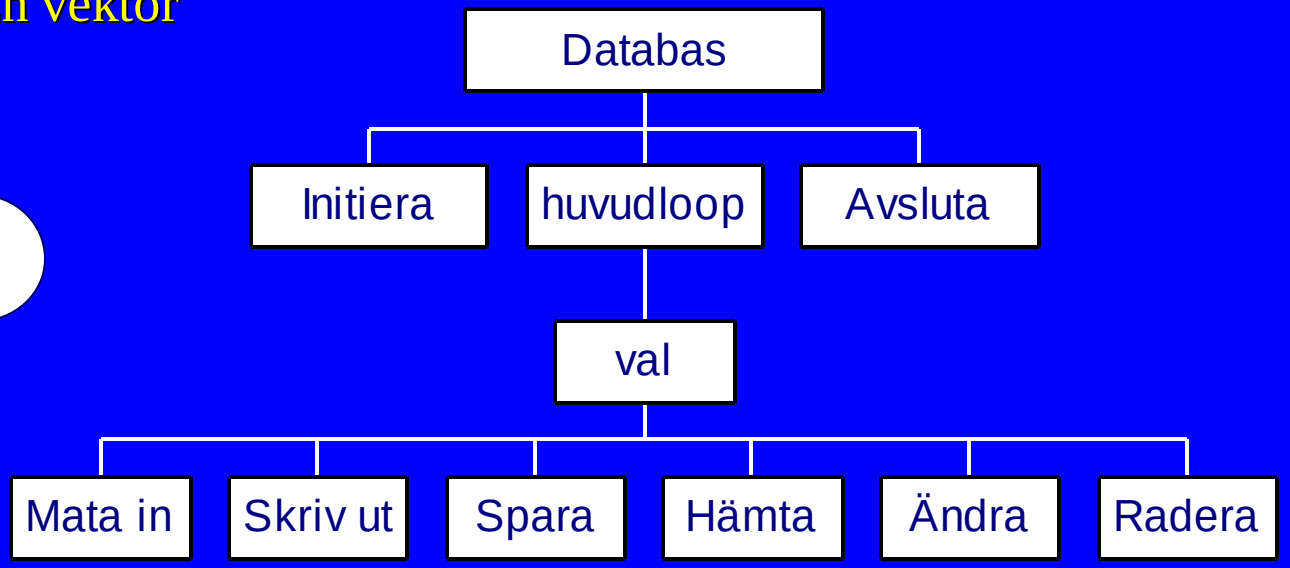
Data:

En vektor




Poster (struct)

Funktioner:




**Enligt diskussion**