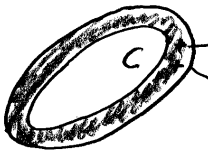
Kap 2, Grundläggande C++



Objektorienterade egenskaper

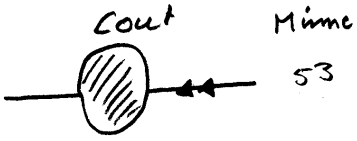
Säkrare och bättre C

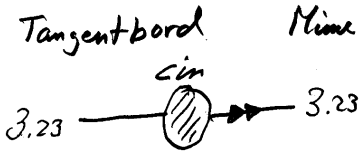
In- och utmatning

```
#include <iostream>      // Definierar de globala
using namespace std;     // objekten cin och cout
```

Ex

```
#include <iostream>
using namespace std;
int tal = 53;
cout << tal;
!
float x;
cout << "Ge ett reellt tal: ";
cin >> x;
!
```





Objekt har tillstånd som man kan kontrollera eller ändra.

Ex

```
!
cin >> x;
while (!cin)   // inte cin ok
{
    cin.clear(); //ok nu
    cin >> x;
!
```

(1)

Dynamisk allokering

(Ex) Allokera dynamiskt en vektor med ett inläst antal reella tal.

Statiskt! float rvek[100]

Dynamiskt!

a) ANSI-C

```
:
float *rvek;
int antal;
printf("Ge antal element: ");
scanf("%d", &antal);
rvek = calloc(antal, sizeof(float));
:
free(rvek);
:
```

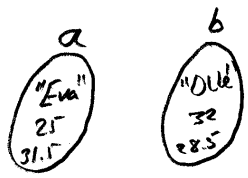
b) C++

```
:
float *rvek;
int antal;
cout << "Ge antal element: ";
cin >> antal;
rvek = new float[antal];
:
delete [] rvek;
:
```

(Ex) Allokera dynamiskt två objekt av klassen STUDENT ovan och läs in studenterna och skriv ut studenterna med den som har mest poäng först.

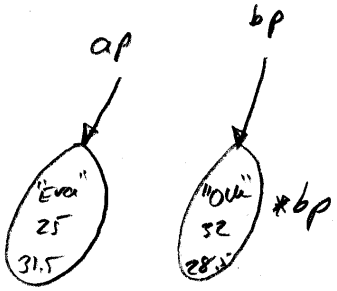
a) Statiskt

```
STUDENT a, b;
a.las();
b.las();
if(a.mindre(b))
{
   b.skriv();
   a.skriv();
}
else
   ;
```



b) Dynamiskt

```
STUDENT *ap, *bp;
ap = new STUDENT;
bp = new STUDENT;
ap->las();
bp->las();
if(ap->mindre(*bp))
{
   bp->skriv();
   ap->skriv();
}
else
   ;
```



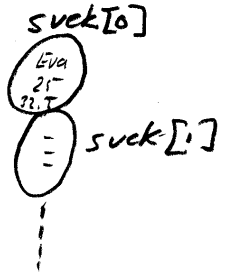
OBS! Glöm ej
delete ap;
delete bp;

(Ex) Allokera dynamiskt en vektor av ett inläst antal studenter samt läs in studenter och sortera dessa efter ålder och skriv ut.

```
#include "Student.h"
void main()
{
    STUDENT *svek;
    int antal;
    cout << "Ge antal studenter : ";
    cin >> antal;
    svek = new STUDENT[antal];

    for (int i=0; i<antal; i++) //OBS! Kan def variabler var som helst!
    {
        svek[i].las();
    }

    for(int i=0; i<antal-1; i++)
    {
      for(int k=i+1; k<antal; k++)
      {
        if ( svek[k].yngre(svek[i]))
        {
            STUDENT temp = svek[i];
            svek[i] = svek[k];
            svek[k] = temp;
        }
      }
    }
    delete [] svek;
}
```



Funktioner

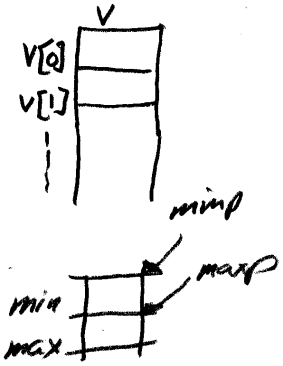
Referensparametrar, defaultvärden för formella parametrar, överlagrade funktioner och parametrar till main-funktionen.

(EX) Funktion för beräkning av min och max i en reell vektor.

a) ANSI-C

```
void vekminmax(float v[], int nr, float *minp, float *maxp)
{   int i;
    *minp = *maxp = v[0];

    for(i=1; i<nr; i++)
    {  if (v[i] < *minp)
           *minp = v[i];
       else if (v[i] > *maxp)
           *maxp = v[i];
    }
}
!
float vek[3] = {1.5, 2.4, 1.8}, min, max;
vekminmax(vek, 3, &min, &max);
printf("Min = %.2f\n", min);
printf("Max = %.2f\n", max);
!
```



b) C++

OBS! Referenser

```
void vekminmax( float v[], int nr, float &minr, float &maxr)
{
    minr = maxr = v[0];
    for (int i = 1; i < nr; i++)
    {
        if (v[i] < minr)
            minr = v[i];
        else if (v[i] > maxr)
            maxr = v[i];
    }
}
;
float vek[3] = {1.8, 2.3, 2.1}, min, max;
vekminmax(vek, 3, min, max);
cout << "Min = " << min << endl;
cout << "Max = " << max << endl;
!
```

minr [1.8] min

maxr [2.3] max

Formella referensparametrar blir nya namn på anropande aktuella parametrar (alias).

Med referensparametrar kan en funktion både ta emot information och skicka tillbaks information (in- ut parametrar)

(Ex) Skriv en funktion som byter värden mellan två studenter

```
#include "Student.h"
void byt_student (STUDENT &ar, STUDENT &br)
{
    STUDENT temp = ar;
    ar = br;
    br = temp;
}
void main()
{
    STUDENT a, b;
    a.las();
    b.las();
    byt_student(a,b);
    a.skriv();
    b.skriv();
}
```

Referensen används för att få tillbaka information via parametern för vanliga objekt. För vektorer och strängar ska man ej använda referensen eftersom man då automatiskt skickar en adress och den måste tas emot som en pekare

(Ex) Funktion som byter värden mellan två namn med max 80 tecken.

```
void byt_namn(char *anamn, char *bnamn)
{
    char temp[80];

    strcpy(temp, anamn);
    strcpy(anamn, bnamn);
    strcpy(bnamn, temp);
}
:
char forsta[30], andra[30];
cout << "Ge förstanamnet: ";
cin.getline(forsta, sizeof(forsta));
cout << "Ge andra namnet: ";
cin.getline(andra, sizeof(andra));
byt_namn(forsta, andra);
cout << "Första namnet är nu " << forsta << endl;
cout << "Andra namnet är nu " << andra << endl;
:
```

I C++ kan funktioner överlagras dvs. ha samma namn bara parameterlistorna ser olika ut till typ och/eller antal parameter.

(Ex) Istället för att ha olika namn på funktionerna ovan, byt_namn och byt_student, kan man använda samma namn eftersom parametarna har olika typer. Kompilatorn går efter parametatyperna när den väljer funktion.

```
void byt(char *anamn, char *bnamn) ①
{
   ⋮
}

void byt(STUDENT &ar, STUDENT &br) ②
{
   ⋮
}

char c[20], d[20];
STUDENT a, b;
//Läs in namnen c och d
byt(c,d); // Funktionen ① anropas
//Läs in studenterna a och b
byt(a,b); // Funktionen ② anropas
```

Formella parametrar kan i C++ tilldelas defaultvärden som antas om motsvarande parameter saknar aktuellt värde. Defaultvärden får bara sättas från slutet av parameterlistan

(Ex)
```
float vekmed(float rv[], int nr=100)
{
  float sum=0;
  for(int i=0;i<nr;i++)
  {
    sum += rv[i];
  }
  return sum/nr;
}
```

Anrop! ① `cout << vekmed(avek,50);`

② `cout << vekmed(bvek);`

① får nr 50

② får nr 100

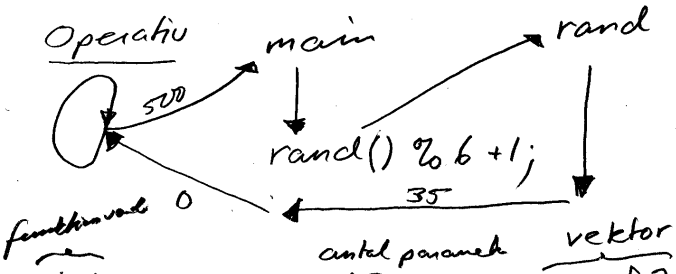
Parametrar till main-funktionen

Ett program skrivet i C++ startar alltid sin körning i funktionen main. Det är operativsystemet som anropar main-funktionen. Man kan i samband med detta anrop skicka information till main via parametrar. Man kan också använda funktionsvärdet för main som ett returvärde till operativet.

(Ex) Skriv ett program som slumpar ett på kommandoraden angivet antal tärningskast.

```
M:\PCSA\OOP\slumpa 500
```

exefilnamn, antal kast




```
int main(int argc, char *argv[])
{
  int nr;
  if(argc==1)
    nr=100;
  else if(argc==2)
    nr=atoi(argv[1]);
  else
    return -1;
  srand((unsigned)time(NULL));
  for(int k=1;k<=nr;k++)
    cout<<rand()%6+1;
  return 0;
}
```

Datatyper i C++

(Ex)
```
struct bil
{
  char reg[7];
  char agare[20];
};
```

Typ variabel

```
bil minbil;
```

Typ Variabel Värde

```
bool ok = true;
```

(Ex) `enum bool {false, true};`

(Ex) Program som kontrollerar om tecknet är en siffra eller ej. Är det en siffra ska den returnera.

```
bool siffra(char &c)
{
  if (c>='0' && c<='9')
  {
    c='0';
    return true;
  }
  return false;
}
```

Anrop!

```
char ch = 'A';
if(siffra(ch))
```

Hemuppgift: Gör om funktionen byt till en medlemsfunktion i klassen PERSON så att den byter värden mellan 2 personer