Innehållsförteckning

6 Funktioner

Funktioner är grundstenar både i traditionell funktionsorienterad programmering och i OOP. Detta avsnitt är en introduktion i funktionernas värld.

En funktion är en avgränsad del i ett program.
En funktion kan anropas från en annan del av programmet.

Det finns två huvudtyper av funktioner:

- Funktioner med returvärde
- Funktioner utan returvärde

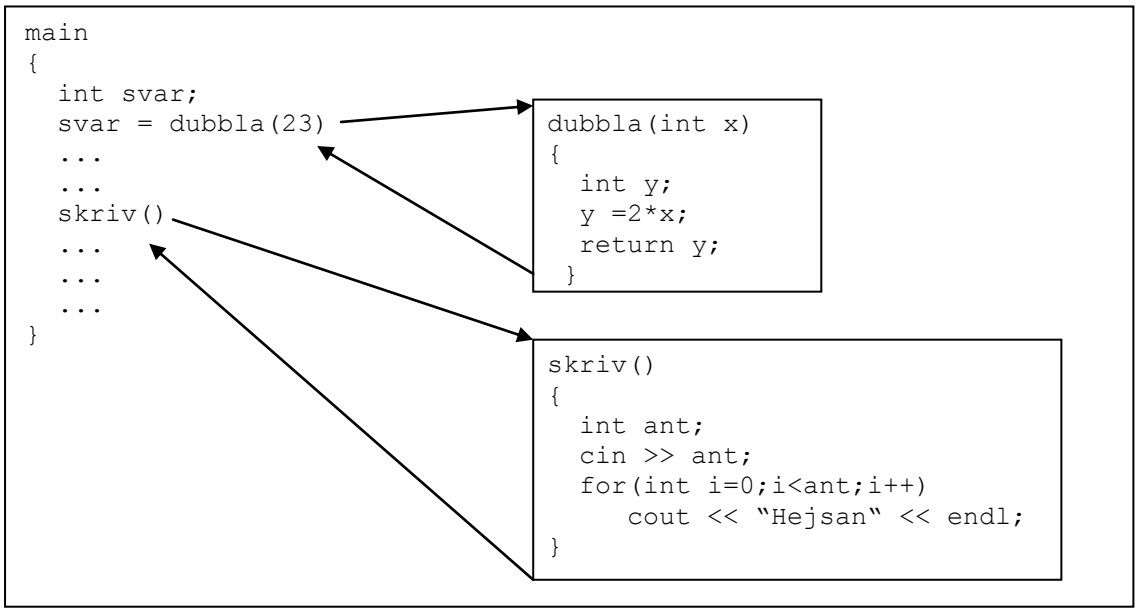
Funktioner kan dessutom ta emot eller inte ta emot indata.

Vi har tidigare sett att det finns färdiga funktionsbibliotek och vi ska nu se hur man kan skriva egna funktioner.

6.1 Anrop av en funktion

I figuren nedan visas hur två funktioner anropas från ett huvudprogram. I det första fallet skickas talet 23 till funktionen *dubbla* som dubblar det, skickar tillbaks värdet 46 som tilldelas variabeln svar.

I det andra fallet anropas funktionen *skriv* utan att något värde skickas med. Här utförs de satser som finns i funktionens kropp (mellan { och }), sedan återgår kontrollen till huvudprogrammet. Man kan se denna funktion som ett delprogram, en subrutin. Varje funktion kan anropas flera gånger. Detta gör att kod inte behöver upprepas.



Vi har tidigare sett hur man kan använda färdiga funktioner, t ex:

```
float x = sqrt(3.4);
```

6.2 Funktion med returvärde och en inparameter

Vi ska skriva en egen funktion som omvandlar längdmåttet tum till mm.
Indata: inch (heltal)
Utdata: mm (heltal)

Funktionsdefinition

```
int inchToMm(int inch)
{
  //Omvandla tum till mm
  int mm = 25 * inch;

  return mm;
}
```

Funktionsdefinitionen består av

- funktionshuvud: `int inchToMm(int inch)`
- funktionskropp: {…}.

I funktionshuvudet finns det: `int inchToMm(int inch)`

- **`int`**: returvärdets datatyp
- **`inchToMm`**: funktionens namn (inled med gemen)
- **`int inch`**: formell parameter

Funktionskroppen består av

- ett antal satser
  - mm är en **lokal** variabel
- `return`: det som står efter return skickas tillbaks till anropande program (funktion)

Anrop av funktionen görs i huvudprogrammet med

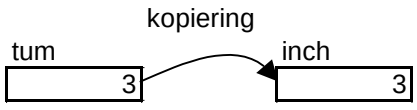
```
float millimeter;
millimeter = inchToMm(4); // ger millimeter = 101,6 mm
```

eller

```
int tum =3;
millimeter = inchToMm(tum); // ger millimeter = 76,2
```

tum kallas för **aktuell** parameter (argument)

Dessa sätt att anropa en funktion kallas att man gör ett **värdeanrop** (by value). Ett värdeanrop innebär att aktuellt värde kopieras till den formella parametern, se figur nedan:



inch (och mm) lever enbart inne i funktion mellan {…}

Ur användarprogrammets synpunkt ska funktionen ses som en svart låda:

6.3 Funktionens placering i ett program

I C++ programmeringsstil skriver man:

- en funktionsprototyp (=funktionshuvudet) **före** huvudprogrammet (main)
- funktionsdefinitionen **efter** huvudprogrammet

Så här ser det ut i ett programexempel:

```
// func_030
// Funktioner: returvärde och en inparameter
// Per Ekeroot 2013-09-01
// Ver 9
//-----------------------------------------------------------------------------
#include <iostream>
using namespace std;
//-----------------------------------------------------------------------------
// Funktionsprototyp
//-----------------------------------------------------------------------------
int inchToMm(int inch );

//-----------------------------------------------------------------------------
// main() börjar
//-----------------------------------------------------------------------------
int main()
{
  // Funktionsanrop
  cout << " Three inches = " << inchToMm(3) << " mm"  << endl << endl;

  return 0;
}// main() slutar

//-----------------------------------------------------------------------------
// Funktionsdefinition
//-----------------------------------------------------------------------------
int inchToMm(int inch)
{
  //Omvandla tum till mm
  int mm = 25 * inch;

  return mm;
}
```

Kommentarer:

- Funktionsprototypen float inchToMm(int inch) anger:
  - funktionen returnerar ett heltal (int)
  - funktionens namn = inchToMm
  - indata = ett heltal. Den formella parametern är namngiven här, inch (behövs inte).
- I funktionsdefinitionen
  - görs omvandlingen från inch till mm. resultatet läggs i en **lokal variabel**, mm
  - mm:s värde skickas till huvudprogrammet med satsen **`return`** `mm`.
- Funktionsanropet InchToMm(3) innebär:
  - inch = 3
  - mm = 25*3 (=75)
  - värdet 75 returneras ”i funktionsnamnet”, d.v.s. inchToMm(3) ersätts med värdet 75
- Alternativt skrivsätt
  - `int svar = inchToMm(3)`
  - här tilldelas svar värdet 75

6.4 Funktion med returvärde och två inparametrar

Funktionens uppgift: Beräkna summan av två flyttal.

Indata: tal1, tal2 (flyttal)
Utdata: summan av tal1 och tal2 (flyttal)

Funktionsdefinition:

```
float add( float x1, float x2)
{
  float sum = x1 + x2;
  return sum;
}
```

Alternativt kan man utesluta den lokala variabel för summan och direkt returnera resultatet av additionen:

```
float add( float x1, float x2)
{
   return x1 + x2;
}
```

Svart låda:

Hela programmet:

```
// func_040
// Funktioner: returvärde och två inparametrar
// Per Ekeroot 2013-09-01
// Ver 9
//-----------------------------------------------------------------------------
#include <iostream>
using namespace std;
//-----------------------------------------------------------------------------
// Funktionsprototyp
//-----------------------------------------------------------------------------
float add( float x1, float x2);

//-----------------------------------------------------------------------------
int main()
{
  // Deklarera variabler
  float num1, num2;

  // Mata in data från tangentbordet
  cout << "Input two numbers" << endl;
  cout << "Number 1: ";
  cin >> num1;
  cout << "Number 2: ";
  cin >> num2;
```

```
// Anropa funktionen add()
  float sum = add(num1,num2);


// Skriv ut resultatet
  cout << endl << num1 << " + " << num2 << " = " << sum << endl << endl;

  return 0;
}

//-----------------------------------------------------------------------------
// Funktionsdefinition
//-----------------------------------------------------------------------------
float add( float x1, float x2)
{
   return x1 + x2;
}
```

Kommentarer:

- Som alternativ till:

```
float sum = add(num1,num2);
cout << endl << " Number 1 + number 2 = " << sum;
```

  kan man skriva

```
cout << endl << " Number 1 + number 2 = " << add(num1,num2);
```

- Funktionsprototypen anger
  - returvärdet är ett flyttal
  - namnet =Add
  - Indata är två flyttal (x1, x2)
- Funktionsdefinitionen:
  - beräknar summan av x1 och x2 och returnerar denna (`return x1+x2`)
- I funktionsanropet `float sum = Add(num1,num2)` sätter man in två **aktuella parametrar** `num1,num2` på de **formella parametrarnas** plats. `num1` och `num2` har fått värden genom inmatning från tangentbordet
- Man kan anropa en funktion på samma ställen som man placerar en variabel, se alternativet.

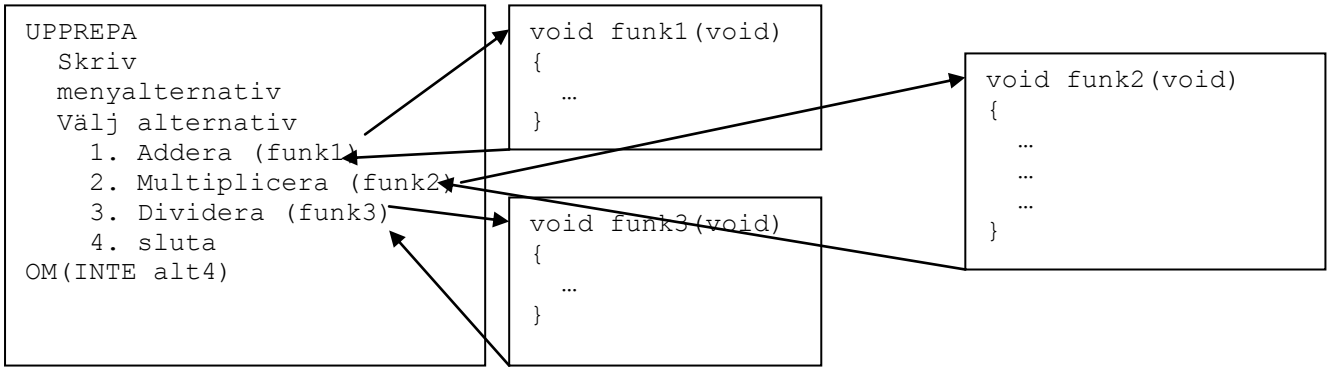
6.5 God programmeringssed beträffande funktioner

Dokumentera alltid funktionens uppgift, indata och utdata. Skriv på följande sätt:

```
//-----------------------------------------------------------------------------
// Namn    : add
// Uppgift : Adderar två flyttal
// Indata  : float x1
//           float x2
// Utdata  : summan av x1 och x2 som ett flyttal
//-----------------------------------------------------------------------------
 float add( float x1, float x2)
{
   return x1 + x2;
}
```

6.6 Funktion utan indata eller returvärde

Det finns funktioner som inte har indata eller returvärde. Man kallar dessa funktioner för void-funktioner eftersom funktionshuvudet får utseendet `void funk(void)`. Sådana funktioner kan man använda för att skapa bättre struktur på programmet. I exempel func_070 ska vi göra ett program med en meny i vilken man väljer vad programmet ska utföra. Strukturen för ett menyprogram kan se ut så här:



Först ett enkelt program som använder en void-funktion. Här visas funktionsdefinitionen till en void-funktion, dess uppgift är att skriva en textrad på skärmen.

```
void print(void)
{
   cout << "My programming is beautiful!";
}
```

Funktionens prototyp:

```
void print(void);
```

Kommentarer:

- Indata: void
- returdatatyp: void
- void = ingenting. Void kan utelämnas i parentesen men **inte** för returdatatypen

Funktionsanrop:

```
print();
```

OBS! Parenteserna måste skrivas!

Hela programmet:

```
// func_050
// Funktioner: inget returvärde och inget indata
// Per Ekeroot 2013-09-01
// Ver 9
//-----------------------------------------------------------------------------
#include <iostream>
using namespace std;

//-----------------------------------------------------------------------------
// Funktionsprototyp
//-----------------------------------------------------------------------------
void print(void);
```

```
int main()
{
  // Funktionsanrop
  print();

  return 0;
}// main slutar

//--------------------------------------------------------------------------
// Funktionsdefinition
//--------------------------------------------------------------------------
void print(void)
{
   cout << "My programming is beautiful!" << endl << endl;
}
```

Kommentar:

- Det här exemplet gör ju koden krångligare med funktion än utan. Tanken är dock att visa principen för en void-funktion, huvudprogrammet blir enklare och "jobbet" görs i funktioner

6.7 Funktion med indata men utan returvärde

Vi ska nu utveckla skrivfunktionen så att vi kan skicka data till den. Man ska kunna ange hur många gånger textraden ska skrivas ut. Här följer funktionsdefinitionen

```
void print(int nr)
{
  for( int i=1; i<=nr; i++)
    cout << My program is beautiful!!" << endl;
}
```

Funktionen har nu en (formell) parameter, *nr*. Ett anrop av funktionen kan se ut så här:

```
print(3);
```

Vilket resulterar i att ”Jag programmerar vackert” skrivs tre gånger.

Ett helt program som använder skriv()-funktionen:

```
// func_060
// Funktioner: inget returvärde och en inparameter
// Per Ekeroot 2013-09-01
// Ver 9
//--------------------------------------------------------------------------
#include <iostream>
using namespace std;
//--------------------------------------------------------------------------
// Funktionsprototyp
//--------------------------------------------------------------------------
void print(int nr);

// ----- main() börjar
int main()
{
  // Funktionsanrop
  print(3);
  cout << endl;

  return 0;
}// -----main() slutar
```

```
//-------------------------------------------------------------------------
// Funktionsdefinition
//-------------------------------------------------------------------------
void print(int nr)
{
  for( int i=1; i<=nr; i++)
    cout << "My program is beautiful!!" << endl;
}
```

Kommentarer:

- Funktionsprototypen, `void skriv(int)`, anger att funktionen `skriv` returnerar ”inget”, och att man ska ge den indata i form av ett heltal (heltalsargument).
- I funktionsdefinitionen har vi gett denna parameter ett namn, **nr.**
- Parametern nr är en **formell** parameter.
- Den formella parametern nr används inne i funktionen på samma sätt som en variabel.
- När funktionen anropas i huvudprogrammet, skriv(3), sätts den formella parametern med argumentet 3, och överallt i funktionen där det står ett **nr** ersätts detta med värdet 3.
- Detta sätt att anropa heter **värdeanrop**
- int i är en **lokal** variabel

6.8 Ett menysystem

Nu har vi verktygen för att bygga ett menysystem, nämligen funktioner och switchsatsen.
I följande program visas en meny i vilken vi kan välja mellan tre alternativ:

1. Addera
2. Multiplicera
3. Sluta

Beroende på användarens val körs olika delprogram. Se skissen under punkt 6.6.

Pseudokod:

```
again = true;
UPPREPA
  Skriv menyalternativen
  Användaren väljer
    Alt1: kör add()
    Alt2: kör mult()
    Alt3: again = false;
OM(again)
```

Funktioner som används i programmet:

```
void showMenu();   // Skriver ut menyalternativen
void add();        // Delprogram för addition
void mult();       // Delprogram för multiplikation
void waitForKey(); // Programmet fortsätter vid knapptryckning
```

Dessa funktioner anropas från huvudprogrammet som ser ut så här:

```
int main()
{
  bool again = true;
  do
  {
    showMenu();                    // Visa menyalternativ
    char ch;
    cin >>ch;                      // Användaren gör sitt val

    switch(ch)
    {                              // Kör valt delprogram
      case '1': add();
                break;
      case '2': mult();
                break;
      case '3':                     // Sluta med 3, q eller Q
      case 'q':
      case 'Q': again = false;
    }
  }while(again);

  return 0;
}
```

Kommentarer:

- do-while-satsen upprepas tills användaren matar in 3, q, Q eller ESC.
- Vi använder en boolsk variabel för att hålla reda på om användaren vill sluta eller fortsätta.
- Vi använder **inte** använda break bara för att kunna avsluta med 3, q, Q eller ESC
- Delprogrammet Addera, funktionen add(), och Multiplicera, funktionen mult() anropas från switch()-satsen. Funktionerna är definierade sist i programmet
- Om användare matar in något annat tecken än 1, 2, 3, q, Q eller ESC så händer inget utan menyn visas och programmet väntar på en ny inmatning

Hur ser då add() och mult(), d.v.s. delprogrammen, ut? Jo, här får du funktionsdefinitionerna.
Först showMenu:

```
void showMenu()
{
  cout << endl;
  cout << " ***** MENU *****" << endl << endl;
  cout << "   1. Addition" << endl;
  cout << "   2. Multiplication" << endl;
  cout << "   3. Quit" << endl;
}
```

Kommentar:

- Menyalternativen skrivs ut på skärmen

Sedan add:

```
void add()
{
  // Inmatning av tal
  float num1, num2;
  cout << " Input two numbers and get the sum." << endl<<endl;
  cout << " Number 1: ";
  cin >> num1;
  cout << " Number 2: ";
  cin >> num2;
  cin.get();  // Läser bort ENTER från inströmmen

  // Beräkna och skriv resultat
  cout << num1 << " + " << num2 << " = " << num1 + num2 << endl << endl;

  waitForKey();
}
```

Kommentarer:

- `num1, num2` är lokala variabler som existerar enbart inom funktionen add()
- `waitForKey();` stoppar programkörningen. Fortsätter vid knapptryckning.

och till sist mult:

```
void mult()
{
   // Inmatning av tal
  float num1, num2;
  cout << " Input two numbers and get the product." << endl<<endl;
  cout << " Number 1: ";
  cin >> num1;
  cout << " Number 2: ";
  cin >> num2;
  cin.get();                // Läser bort ENTER från inströmmen

  // Beräkna och skriv resultat
  cout << num1 << " * " << num2 << " = " << num1 * num2 ;
  waitForKey();
}
```

Kommentarer:

- `num1, num2` är lokala variabler som existerar enbart inom funktionen mult()

Till slut hela programmet så att du kan se var funktionsprototyperna, funktionsdefinitionerna och huvudprogrammet placeras:

```
// func_070
// Visar hur man kan bygga upp en meny
// Per Ekeroot 2013-09-01
// Ver 9
//--------------------------------------------------------------------------
#include <iostream>
using namespace std;

//--------------------------------------------------------------------------
// Funktionsprototyper
//--------------------------------------------------------------------------
void showMenu();
void add();
void mult();
void waitForKey();
```

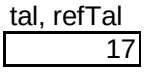
```
// ----- Huvudprogram
int main()
{

  bool again = true;
  do
  {
    showMenu();                    // Visa menyalternativ
    char ch;
    cin >>ch;                      // Användaren gör sitt val

    switch(ch)
    {                              // Kör valt delprogram
      case '1': add();
                break;
      case '2': mult();
                break;
      case '3':                     // Sluta med 3, q eller Q
      case 'q':
      case 'Q': again = false;
    }
  }while(again);

  return 0;
}
//---------------------------------------------------------------------------
// Funktionsdefinitioner
//---------------------------------------------------------------------------
//  showMenu
//---------------------------------------------------------------------------
// Uppgift: Skriver ut menyalternativ på skärmen
// Indata : -
// Utdata : -
//---------------------------------------------------------------------------
void showMenu()
{
  cout << endl;
  cout << " ***** MENU *****" << endl << endl;
  cout << "   1. Addition" << endl;
  cout << "   2. Multiplication" << endl;
  cout << "   3. Quit" << endl;
}
// Fortsättning på nästa sida!
```

```
//-------------------------------------------------------------------------------
//  add
//-------------------------------------------------------------------------------
// Uppgift: Ett delprogram som utför addition
// Indata : -
// Utdata : -
//-------------------------------------------------------------------------------
void add()
{
  // Inmatning av tal
  float num1, num2;
  cout << " Input two numbers and get the sum." << endl<<endl;
  cout << " Number 1: ";
  cin >> num1;
  cout << " Number 2: ";
  cin >> num2;
  cin.get();  // Läser bort ENTER från inströmmen

  // Beräkna och skriv resultat
  cout << num1 << " + " << num2 << " = " << num1 + num2 << endl << endl;

  waitForKey();
}

//-------------------------------------------------------------------------------
// mult
//-------------------------------------------------------------------------------
// Uppgift: Ett delprogram som utför multiplikation
// Indata : -
// Utdata : -
//-------------------------------------------------------------------------------
void mult()
{
   // Inmatning av tal
  float num1, num2;
  cout << " Input two numbers and get the product." << endl<<endl;
  cout << " Number 1: ";
  cin >> num1;
  cout << " Number 2: ";
  cin >> num2;
  cin.get();                // Läser bort ENTER från inströmmen

  // Beräkna och skriv resultat
  cout << num1 << " * " << num2 << " = " << num1 * num2 ;

  waitForKey();
}

//-------------------------------------------------------------------------------
// waitForKey
//-------------------------------------------------------------------------------
// Uppgift: Stoppar programexekveringen. Exekveringen fortsätter när användaren
//          trycker på någon tangent
// Indata : -
// Utdata : -
//-------------------------------------------------------------------------------
void waitForKey()
{
  cout << endl << endl << "Press a key to continue!";
  cin.get();
}
```

6.9 Referensvariabler

En referensvariabel är ett "extranamn" (synonym) för en annan variabel.
Exempel:

```
int tal = 17;
int &refTal = tal;
```



Tal och refTal använder **samma** minnesutrymme

```
// func_080
// Referensvariabler
// Per Ekeroot 2013-09-01
// Ver 9
//--------------------------------------------------------------------------
#include <iostream>   // cout
using namespace std;
//--------------------------------------------------------------------------
int main()
{
  // Deklarera ett heltal och en referens till detta heltal
  int num = 17;
  int &refNum = num;

  // Skriv referensvariabeln på skärmen
  cout << "Referens number: " << refNum << endl;

  // Addera ett till referensvariabeln
  refNum++;

  // Skriv ut heltalsvariabeln
  cout << "Number         : " << num<< endl<< endl;

  // Skriv ut variablernas adresser
  cout << "Address to 'num'   : " << &num << endl;
  cout << "Address to 'refNum': " << &refNum << endl << endl;

  return 0;
}
```

Kommentarer:

- Deklarationen int &refNum = num innebär att refNum får samma adress som num
- referensvariabeln måste initieras vid deklarationen
- &num och &refNum ger adressen till resp. variabel.

6.10 Referensanrop i en funktion

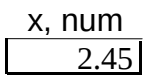
Vi ska nu utnyttja att en referensdeklarerad parameter använder samma minneutrymme som den anropande variabel. Funktionen `multiplyWith_2()` ska fördubbla ett flyttal genom att direkt fördubbla den anropande variabelns värde. Funktionsdefinitionen för en sådan funktion kan se ut så här:

```
void multiplyWith_2(float &num) // Referensdeklarerad parameter
{
  num*=2;
}
```

Om denna funktion anropas så här:

```
  float x;
  cout << "Input a number: ";
  cin >> x;

  multiplyWith_2(x);

  cout << endl << "The number multiplied with 2: " << x  << endl << endl;
```

så har x fördubblats efter anropet av `multiplyWith_2(x)`! Hur går detta till?
Jo, då `multiplyWith_2()` anropas kommer den referensdeklarerade parametern `num` att använda samma minnesutrymme som x, så när `num` fördubblas så är det egentligen x som fördubblas.



Hela programmet:

```
// func_090
// En funktion med referensdeklarerad parameter
// Per Ekeroot 2013-09-01
// Ver 9
//-----------------------------------------------------------------------------
#include <iostream>   // cout, cin
using namespace std;
//-----------------------------------------------------------------------------
// Funktionsprototyp
void multiplyWith_2(float &num);

int main()
{
  float x;
  cout << "Input a number: ";
  cin >> x;

  multiplyWith_2(x);

  cout << endl << "The number multiplied with 2: " << x  << endl << endl;

  return 0;
}
//-----------------------------------------------------------------------------
// multiplyWith_2
// Uppgift: Fördubblar ett flyttal.
// Indata : tal, talet som ska fördubblas
// Utdata : tal, det fördubblade talet
//-----------------------------------------------------------------------------
void multiplyWith_2(float &num)
{
  num*=2;
}
```

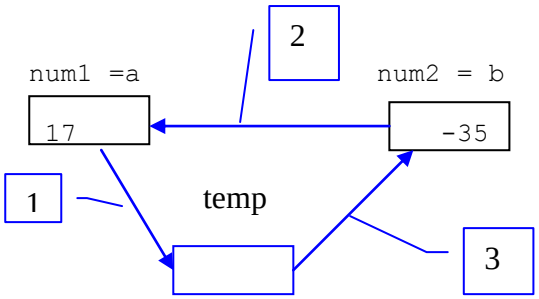
6.11 Två referensparametrar i en funktion

Referensvariabler är mest användbara som formella parametrar i funktioner.
Funktionen `swap(int &a, int &b)` i byter plats på värdena i parametrarna a och b.

```
void swap(int &a, int &b)
{
  int temp = a;
  a = b;
  b = temp;
}
```

Exempel på anrop:

```
int num1=17, num2=-35;
swap(num1,num2);
```



Blått = swaps operationer
Siffrorna anger den ordningen som operationerna utförs.

Programexempel:

```
// func_100 Ver 9
// Funktion med två referensparametrar
// Per Ekeroot 2013-09-01
#include <iostream>   // cout
using namespace std;
//-----------------------------------------------------------------------------
// Funktionsprototyp
//-----------------------------------------------------------------------------
void swap(int &a, int &b);

//-----------------------------------------------------------------------------
int main()
{
  // Deklarera och initiera två tal
  int num1 = 17, num2 = -35;

  // Skriv ut talen
  cout << "Number 1 = "<< num1 << "  Number 2 = " << num2 << endl << endl;

  // Byt plats på talens värden
  swap(num1,num2);

  // Skriv ut talen
  cout << "Number 1 = "<< num1 << " Number 2 = " << num2 << endl << endl;

  return 0;
}
//-----------------------------------------------------------------------------
//  swap
//-----------------------------------------------------------------------------
// Uppgift: Byter plats på data i inparametrarna a och b
// Indata : a, b (int) referensdeklarerade heltalsvariabler
// Utdata : -
//-----------------------------------------------------------------------------
void swap(int &a, int &b)
{
  int temp = a;
  a = b;
  b = temp;
}
```

6.12 Funktion för lösning av andragradsekvation

Som ytterligare ett exempel på hur man kan använda referensanrop visas här ett sätt att skriva en funktion som löser en andragradsekvation:

$x^2 + px + q = 0$ har lösningen $x_{1,2} = -\frac{p}{2} \pm \sqrt{\frac{p^2}{4} - q}$

Exempel:

$x^2 + 2x - 3 = 0$ har lösningen: $x_1 = -\frac{2}{2} + \sqrt{\frac{2^2}{4} - (-3)} = -1 + \sqrt{1+3} = -1 + 2$

och $x_2 = -\frac{2}{2} - \sqrt{\frac{2^2}{4} - (-3)} = -1 - \sqrt{1+3} = -1 - 2$

Vi ska skriva en funktion som har 4 parametrar varav 2 är indata och 2 är utdata. Funktionen har dessutom ett returvärde. Funktionsdefinitionen:

```
bool solveEquation(float p, float q, float &x1, float &x2)
{
  x1 = x2 =0;

  float pp = p*p/4-q;         // Beräkna värdet under rottecknet
  bool solution = (pp >= 0); // Lösning finns om värdet under rottecknet >= 0
  if(solution)
  {
    x1 = -p/2 + sqrt(pp);
    x2 = -p/2 - sqrt(pp);
  }
  return solution;
}
```

Kommentarer:

- `pp = p*p/4-q`, tilldela det som står under rottecknet till pp.
- `solution` blir *true* om talet under rottecknet är större eller lika med noll.
- Tre värden returneras från funktionen:
  - x1 (via referensparameter)
  - x2 (via referensparameter)
  - lösbart (true/false) (via return)
- Indata ges i parametrarna: p och q (float)

Anrop:

```
if(solveEquation(p_in,q_in,root1,root2))
  cout << endl << "x1= " << root1 << "  x2= " << root2;
else
  cout << " No solution!";
```

Kommentarer:

- solveEquation returnerar true eller false
- om ekvationen inte är lösbar skrivs texten ”Lösning saknas”, x1=0 och x2=0 i detta fall eftersom funktionen inleds med x1=x2=0;.
- För den matematiskt intresserade: Med lösbar ekvation menas i detta fall att lösningarna ska vara reella.

Hela programmet:

```
// func_110
// Funktioner och referensanrop
// Lösning till en andragradsekvation
// Per Ekeroot 2013-09-01
// Ver 9
//-----------------------------------------------------------------------------
#include <iostream>   // cout
#include <string>
#include <cmath>      // sqrt()
using namespace std;

//-----------------------------------------------------------------------------
// Funktionsprototyper
//-----------------------------------------------------------------------------
bool solveEquation(float p, float q, float &x1, float &x2);
bool yesOrNo(string str);

//-----------------------------------------------------------------------------
// Huvudprogram
//-----------------------------------------------------------------------------
int main()
{
  do
  {
    // Inmatning av värden
    cout << endl << "Solution to the equation x*x + p*x + q = 0 " << endl;

    float p_in,q_in;
    cout << "Input p: ";
    cin >> p_in;
    cout << "Input q: ";
    cin >> q_in;

    // Lös ekvationen och skriv resultatet
    float root1, root2;

    if(solveEquation(p_in,q_in,root1,root2))
      if(root1 != root2)
        cout << endl << "x1= " << root1 << "  x2= " << root2;
      else
        cout << endl << " Double root x1 = x2 = " << root1;
    else
      cout << "No solution!";
    cout << endl;
  }while(yesOrNo("Again (y/n)? "));

  return 0;
}

//-----------------------------------------------------------------------------
// Funktionsdefinitioner
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// solveEquation
//-----------------------------------------------------------------------------
// Uppgift: Löser en andragradsekvation som ges på formen x²+px+q=0
// Indata : p, q (float) - parametrarna i andragradsekvationen
// Utdata : Funktionen returner true om ekvationen är lösbar annars false
//          x1, x2 (float) - lösningarna till andragradsekvationen
//-----------------------------------------------------------------------------
```

```
bool solveEquation(float p, float q, float &x1, float &x2)
{
  x1 = x2 =0;

  float pp = p*p/4-q;         // Beräkna värdet under rottecknet
  bool solution = (pp >= 0); // Lösning finns om värdet under rottecknet >= 0
  if(solution)
  {
    x1 = -p/2 + sqrt(pp);
    x2 = -p/2 - sqrt(pp);
  }
  return solution;
}
//-------------------------------------------------------------------------
// YesOrNo
//-------------------------------------------------------------------------
// Uppgift: Skriver ut en fråga och väntar sedan att användaren ska trycka på
//          y(Y) eller n(N) (yes eller no)och sedan ENTER, som svar på frågan.
// Indata : str (string) - en sträng med frågan
// Utdata : true (bool) - om användaren svarar y(Y)
//          false (bool) - om användaren svarar n(N)
//-------------------------------------------------------------------------
bool yesOrNo(string str)
{
  cout << str;
  char ch;
  do
  {
    cin >> ch;
    ch=toupper(ch);
  }while(!(ch=='Y'||ch=='N'));
  return (ch=='Y');
};
```

Kommentarer:
Jag har också lagt till en funktion som är praktisk att ha då man vill upprepa program. Den heter: yesOrNo(). Dess uppgift är att skriva ut en fråga och sedan vänta på att användaren ska trycka på y(Y) eller n(N) (yes eller no), som svar på frågan. Den returnerar *true* om användaren svarat ja (Y) och *false* om användaren svarat no (N). Funktionen känner enbart av y, Y, n och N.

Indata till funktionen är

- frågan som man vill ställa (string str).

Utdata från funktionen:

- true om användaren trycker y eller Y
- false om användaren trycker n eller N

6.13 Placering av funktioner i egna filer

Vi ska göra ett funktionsbibliotek som innehåller funktionerna solveEquation och yesOrNo. Detta bibliotek kan vi kompilera separat och anropa från ett huvudprogram.

I C++ delar man upp funktionerna så att man lägger:

- funktionsprototyperna i en headerfil, *.h
- funktionsdefinitionerna i en definitionsfil, *.cpp

I headerfilen har man i funktionsprototypen ett gränssnitt mot användaren (programmeraren) medan man gömmer undan detaljerna för hur funktionen är implementerad i definitionsfilen. Dessa filer hör ihop parvis.

6.13.1 Skapa headerfil och definitionsfil

1. Headerfil
   a. Högerklicka på "Header Files" i "Solution Explorer" och välj "Add"| "New Item…".
   b. Välj filtyp "Header File (.h)"
   c. Skriv funktionsprototyperna i h-filen
   d. Lägg in ev inkluderingsfiler i h-filen (#include <iostream> etc.)
2. Definitionsfil
   a. Högerklicka på "Source Files" i "Solution Explorer" och välj "Add"| "New Item…".
   b. Välj filtyp "C++ file (.cpp)".
   c. Skriv funktionsdefinitionerna i cpp-filen. OBS! #include ”FuncLib.h” läggs i denna fil.

Headerfilen:

```
// FuncLib.h
// Funktionsprototyper till ett funktionsbibliotek
// Per Ekeroot 2013-09-01
// Ver 9
//---------------------------------------------------------------------------
#ifndef FuncLibH
#define FuncLibH
  #include <string>
  using namespace std;

  bool solveEquation(float p, float q, float &x1, float &x2);
  bool yesOrNo(string str);
#endif
```

Kommentarer:

- Headerfilen innehåller funktionsprototyper och kompileringsdirektiv
- `ifndef, define och endif` förklaras i nästa stycke.

Till denna headerfil hör en definitionsfil:

```
// FuncLib.cpp
// Funktionsdefinitioner till ett funktionsbibliotek
// Per Ekeroot 2012-09-01
// Ver 8
//--------------------------------------------------------------------------
#include <iostream>
#include <string>
using namespace std;
#include "FuncLib.h"

//--------------------------------------------------------------------------
// solveEquation
//--------------------------------------------------------------------------
// Uppgift: Löser en andragradsekvation som ges på formen x²+px+q=0
// Indata : p, q (float) - parametrarna i andragradsekvationen
// Utdata : Funktionen returner true om ekvationen är lösbar annars false
//          x1, x2 (float) - lösningarna till andragradsekvationen
//--------------------------------------------------------------------------
bool solveEquation(float p, float q, float &x1, float &x2)
{
  x1 = x2 =0;

  float pp = p*p/4-q;         // Beräkna värdet under rottecknet
  bool solution = (pp >= 0); // Lösning finns om värdet under rottecknet >= 0
  if(solution)
  {
    x1 = -p/2 + sqrt(pp);
    x2 = -p/2 - sqrt(pp);
  }
  return solution;
}

//--------------------------------------------------------------------------
// YesOrNo
//--------------------------------------------------------------------------
// Uppgift: Skriver ut en fråga och väntar sedan att användaren ska trycka på
//          y(Y) eller n(N) (yes eller no) och sedan ENTER, som svar på frågan.
// Indata : str (string) - en sträng med frågan
// Utdata : true (bool) - om användaren svarar y(Y)
//          false (bool) - om användaren svarar n(N)
//--------------------------------------------------------------------------
bool yesOrNo(string str)
{
  cout << str;
  char ch;
  do
  {
    cin >> ch;
    ch=toupper(ch);
  }while(!(ch=='Y'||ch=='N'));
  return (ch=='Y');
};
```

Kommentarer:

- I denna fil finns funktionsdefinitionerna
- Kopplingen till funktionsprototyperna i headerfilen görs med #include "FuncLib.h"
- Observera också att definitionsfilen FunkLib.cpp måste var med i projektet!

6.13.2 Anropa filbiblioteket

För att använda det egenhändigt hopsnickrade filbiblioteket inkluderar man headerfilen i sitt program genom att lägga till #include ”funcLib.h” bland de andra inkluderingsfilerna. Sedan anropar man funktionerna på vanligt sätt.

”…” istället för <…> i #include ”funcLib.h” betyder att kompilatorn ska leta i **aktuell** katalog.

Här följer huvudprogrammet:

```
// func_120
// Lägg funktionen solveEquation och yesOrNo i en separat fil
// Per Ekeroot 2013-09-01
// Ver 9
//-----------------------------------------------------------------------------
#include <iostream>
#include <string>
using namespace std;

#include "FuncLib.h"
//-----------------------------------------------------------------------------
int main()
{
  do
  {
    // Inmatning av värden
    cout << "Solution to the equation x*x + p*x + q = 0 "<<endl<<endl;
    float p_in,q_in;
    cout << "Input p: ";
    cin >> p_in;
    cout << "Input q: ";
    cin >> q_in;

    // Lös ekvationen och skriv resultatet
    float root1, root2;

    if(solveEquation(p_in,q_in,root1,root2))
      if(root1 != root2)
        cout << endl << "x1= " << root1 << "  x2= " << root2;
      else
        cout << endl << " Double root x1 = x2 = " << root1;
    else
      cout << "No solution!";

    cout << endl;
  }while(yesOrNo("Again (y/n)? "));

  return 0;
}
```

6.13.3 Programstruktur

```
// mittProgram.cpp
#include ”funcLib.h”
main()
{
  funk1();
  ...
  funk2();
  ...
}
```

```
// funcLib.h
#ifndef funcLib H
#define funcLibH

//Funktionsprototyper
  void funk1();
  void funk2();


#endif
```

```
// funcLib.cpp
#include ” funcLib.h”
// Funktionsdefinitioner
void funk1()
{
  …
}

void funk2()
{
  . . .
}
```

Definitionsfilen *funcLib.cpp* ska vara med i projektet.
I utvecklingsmiljön för Microsoft Visual C++ är filen med i projektet om den finns med i listan under *Source Files.* *Source Files* hittar du i fönstret *Solution Explorer*.

6.13.4 Kompileringsdirektiv

Om kompilatorn försöker kompilera en unit mer än en gång blir det tvärstopp. För att detta inte ska ske skrivs följande kompileringsdirektiv i h-filen:

```
#ifndef namn
#define namn
.
.
.
#endif
```

Om *namn* är ”definierat” kompileras **inte** koden t o m endif.
Om *namn* inte är definierat, definieras *namn* och koden kompileras.

6.14 Funktionsparametrar med förvalt värde

Man kan sätta standardvärden (= förvalda värden) på parametrar. Detta innebär att om man inte anger något argument när man anropar funktionen så sätts standardvärdet. Om man anger argument så använder funktionen detta värde. Som exempel på detta kommer en funktion som gör ett antal tomma rader.

Funktionsprototyp med parameter som har förvalt värde (=3)

```
void spaceRow(int num=3);
```

Funktionsdefinition

```
void spaceRow(int num)
{
  for(int i=0; i<num; i++)
    cout << endl;
};
```

Kommentarer:
* Observera att det förvalda värdet sätts **enbart** i prototypen!

Två exempel på anrop:

```
spaceRow();  // Ger tre tomma rader
spaceRow(5); // Ger fem tomma rader
```

Hela programmet:

```
// func_130
// Funktionsparameter med förvalt värde
// Per Ekeroot 2013-09-01
// Ver 9
//-----------------------------------------------------------------------------
#include <iostream>
using namespace std;

//-----------------------------------------------------------------------------
// Funktionsprototyp
//-----------------------------------------------------------------------------
void spaceRow(int num=3);

//-----------------------------------------------------------------------------
int main()
{
  cout <<"Row 1";
  spaceRow();
  cout << "Next row";
  spaceRow(5);
  cout << "Last row" << endl << endl;

  return 0;
}
//-----------------------------------------------------------------------------
// Funktionsdefinition
//-----------------------------------------------------------------------------
void spaceRow(int num)
{
  for(int i=0; i<num; i++)
    cout << endl;
}
```

6.15 Överlagring av funktioner

Flera funktioner kan ha samma namn om dom har olika parameterlistor. Kompilatorn anser att det är skilda funktioner om de har samma namn men olika parameteruppsättningar (signaturer).

Funktionsprototyper:

```
void print(const string str);
void print(double d);
void print(double d, int width);
void print(int num, int width);
```

Funktionsdefinitioner:

```
// Skriv en string-sträng och gör en radframmatning --------------------------
void print(const string str)
{
  cout << str << endl;
}
// Skriv en double -----------------------------------------------------------
void print(double d)
{
  cout << d << endl;
}
// Skriv en double högerjusterad i ett fält med width tecken -----------------
void print(double d, int width)
{
  cout << setw(width) << d << endl;
}
// Skriv ett heltal (int) högerjusterat i ett fält med width tecken -----------
void print(int i, int width)
{
  cout << setw(width) << i << endl;
}
```

Anrop:

```
  string s1;
  s1 =" This is a string ";
  int num = 45;

  print(s1);
  print(12.34);
  print(12.34,8);
  print(num,8);
```

Hela programmet:

```
// func_140
// Överlagring av funktioner
// Per Ekeroot 2013-09-01
// Ver 9
//-----------------------------------------------------------------------------
#include <iostream>
#include <string>
#include <iomanip>
using namespace std;

//-----------------------------------------------------------------------------
// Funktionsprototyper
//-----------------------------------------------------------------------------
void print(const string str);
void print(double d);
```

```
void print(double d, int width);
void print(int tal, int width);
//-----------------------------------------------------------------------------
// Huvudprogram
//-----------------------------------------------------------------------------
int main()
{
  string s1;
  s1 ="This is a string";
  int num = 45;

  // Utskrift med olika printfunktioner
  print(s1);
  print(12.34);
  print(12.34,8);
  print(num,8);

  cout << endl << endl;
  return 0;
}
//-----------------------------------------------------------------------------
// Funktionsdefinitioner
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
// Skriv en sträng
//-----------------------------------------------------------------------------
void print(const string str)
{
  cout << str << endl;
}
//-----------------------------------------------------------------------------
// Skriv en flyttal (double)
//-----------------------------------------------------------------------------
void print(double d)
{
  cout << d << endl;
}
//-----------------------------------------------------------------------------
// Skriv ett flyttal (double) högerjusterat i ett fält med width tecken
//-----------------------------------------------------------------------------
void print(double d, int width)
{
  cout << setw(width) << d << endl;
}
//-----------------------------------------------------------------------------
// Skriv ett heltal (int) högerjusterat i ett fält med width tecken
//-----------------------------------------------------------------------------
void print(int i, int width)
{
  cout << setw(width) << i << endl;
}
```

Kommentarer:

- Det är signaturen som skiljer funktionerna åt.
- Det är inte tillåtet att enbart ändra returdatatypen. Se nedan:
    - `void print(const char *str);`
    - `bool print(const char *str);`
    - `[C++ Error] Ex0102.cpp(15): E2356 Type mismatch in redeclaration of 'print(const char *)'`