Programmering i C++
EDA623
Funktioner

Funktioner

Innehåll

- Funktionsdefinition
- Funktionsdeklaration
- Överlagring
- Uppdelning av program i filer
- Värdeanrop och referensanrop
- Parametrar med defaultvärden
- Rekursiva funktioner

Funktionsdefinition

Exempel: Medelvärde – variant 1

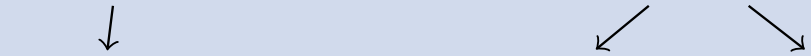
```
double medelv(double x1, double x2) {
   return (x1+x2)/2;
} // Funktionsdefinition

int main() {
   double a=2.3, b=3.9;
   cout << medelv(a, b) << endl;
}
```

Funktionsdefinition

Exempel: Medelvärde – variant 1

Returtyp Parametertyp



```
double medelv(double x1, double x2) {
    return (x1+x2)/2;
} // Funktionsdefinition

int main() {
    double a=2.3, b=3.9;
    cout << medelv(a, b) << endl;
}
```

Funktionsdeklaration

Exempel: Medelvärde – variant 2

```
double medelv(double, double);
// Funktionsdeklaration (prototyp)

int main() {
   double a=2.3, b=3.9;
   cout << medelv(a, b) << endl;
}
double medelv(double x1, double x2) {
   return (x1+x2)/2;
} // Funktionsdefinition efter main
```

Överlagring

Olika varianter av samma funktion:

- Samma namn men olika antal parametrar och/eller parametertyper
- Kan inte skilja på enbart returtyp

Exempel: Medelvärdesfunktioner

```
double medelv(int x1, int x2) {
   return ((double)(x1+x2))/2;
}

int medelv(int x1, int x2) {
   return (x1+x2)/2;
} /* Ger kompileringsfel ty trots olika
returtyper tolkas de som samma funktion */
```

Mer överlagring

Medelvärde av heltal i ett fält

```
double medelv(int x[], int n) {
   double s=0;
   for (int i=0; i<n; i++)
      s += x[i];
   return s / n;
}
```

Uppdelning av program i filer

- Placering av funktionsdeklarationer i s.k. headerfiler (med .h i slutet av namnet)
- Uppdelning av koden på flera källfiler (.cpp eller .c i filefternamn)
- Separatkompilering av varje källfil (.cpp eller .c)
- Minimalt exempel: Uppdelning av medelv:
  - medelv.h
  - medelv.cpp
  - main.cpp

Uppdelning av program i filer

Headerfil

medelv.h

```
double medelv(double, double);
double medelv(int, int);
double medelv(int*, int);
```

Uppdelning av program i filer

Källfil

medelv.cpp

```
double medelv(double x1, double x2) {
   return (x1+x2)/2;
}

double medelv(int x1, int x2) {
   return ((double) x1+x2)/2;
}

double medelv(int x[], int n) {
   double s=0;
   for (int i=0; i<n; i++) {
      s += x[i];
   }
   return s/n;
}
```

Uppdelning av program i filer

Huvudprogram

main.cpp

```
#include <iostream>
#include "medelv.h"

using namespace std;
int c[] = {3, 4, 3, 3, 4, 5, 4, 3, 5};

int main() {
   double a=2.3, b=3.9;
   int m=3, n=4;
   cout << medelv(a, b) << endl;
   cout << medelv(m, n) << endl;
   cout << medelv(c, 9) << endl;
}
```

Vid 'vanliga' anrop kopieras alltid värdena från de aktuella parametrarna till de formella (vilka fungerar som lokala variabler)

Exempel: Byta plats på två heltalsvärden

```
void swap(int a, int b) {
    int tmp=a;
    a = b;
    b = tmp;
}
// Fungerar inte!!
```

Istället för *värdeanrop* används *referensanrop*:

Exempel: Byta plats på två heltalsvärden

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

I princip används nu *adresserna* till de aktuella parametrarna istället för själva *värdena*. Därför byter verkligen talen plats denna gång.

Referensanrop kan användas för att 'returnera' flera värden via s.k. *utparametrar*. I följande exempel fungerar utparametrarna även som inparametrar:

Exempel: Kvadrera två tal

```
void kvadrera(int& x, int& y) {
   x = x*x;
   y = y*y;
}
```

Eftersom fält redan är ett slags referenser kan 'in-ut-parameter'-tekniken användas direkt:

Exempel: Heltalsvektorns element kvadreras

```
void kvadrera(int vec[], int n) {
  for (int i=0; i<n; i++) {
     vec[i] = vec[i]*vec[i];
  }
}
```

En fältvariabel innehåller adressen till första elementet i fältet.

Parametrar med defaultvärden

Funktioner kan ha variabelt antal parametrar utan att överlagring används genom att använda *defaultvärden*.

Exempel: Kast med tärning

```
#include <cstdlib>
#include <ctime>

void kasta(int antal=1, int sidor=6) {
   srand(time(0));
   for (int i=0; i<antal; i++) {
      cout << rand()%sidor + 1 << " ";
   }
   cout << endl;
}

kasta();     //ger 1 slag med 6-sidig tärning
kasta(5);    //ger 5 slag med 6-sidig tärning
kasta(1,5); //ger 1 slag med 5-sidig tärning
```

Rekursiva funktioner

En funktion som anropar sig själv (direkt eller indirekt) sägs vara *rekursiv*.

Exempel: Binomialkoefficientberäkning

```
int binom(int n, int k) {
   if (k==0) return 1;
   if (n==k) return 1;
   return binom(n-1, k-1) + binom(n-1, k);
}
```

http://sv.wikipedia.org/wiki/Binomialkoefficient
http://sv.wikipedia.org/wiki/Pascals_triangel

Rekursiva funktioner

Exempel: Stirlingtal av 2:a slaget (antalet sätt att dela in en mängd av n element i m icke-tomma delmängder)

Exempel: Stirlingtal av 2:a slaget

```
int stirling2(int n, int m) {
   return (m==0) ? 1 : (n==m) ? 1 :
      stirling2(n-1, m-1) + m*stirling2(n-1, m);
}
```

http://sv.wikipedia.org/wiki/Stirlingtal