

## Introduktion till C++ för Java- och C-programmerare - 4 dagar

*kurser 337*

- Du får lära dig att**
- Utnyttja C++ för att bygga utökningsbara och anpassningsbara tillämpningar
  - Konvertera, anpassa och ansluta Java- och C-applikationer till C++
  - Skriva stabila och överblickbara applikationer i C++
  - Bygga upp programvara med mallar (templates) och behållare (containers)
  - Använda internationella standardbibliotek för att göra programmen enklare, mer portabla och pålitligare
  - Minneshantering med konstruktörer och destruktörer
- Sammanfattning** Objektorienterade (OO) program är lättare att förstå och underhålla än de traditionella. Objektorienterade metoder är nyckeln till återanvändbar programvara och minskar kostnaderna för utveckling och anpassning av programvara. Kursen omfattar hela C++, inklusive de senaste tilläggen samt standardbiblioteken och konvertering från C till C++ och Java. Kursen fokuserar på hur språket används, fallpropar, samt principerna för OOP. Övningarna ger erfarenhet av att utveckla objektorienterade program med C++.
- Vem bör delta** IT-specialister som arbetar med utveckling och/eller underhåll av avancerade applikations- eller systemprogram i C++. Erfarenhet av programmering i språk som Java eller C förutsätts.
- Praktiska övningar** Omfattande praktiska övningar förstärker de objektorienterade koncept och programmeringskunskaper du får under kursen. Övningarna omfattar:
- Skriva återanvändbara programkomponenter
  - Använda mallar för datastrukturer
  - Använda undantag för robust felhantering
  - Kapsla in data i klasser
  - Utöka basklasser med arv

# Introduktion till C++ för Java- och C-programmerare - 4 dagar

*kurser 337*

## Introduktion och översikt

### Objektorienterade program

- Vad är objektorienterad programmering?
- OO-programmeringens utveckling
- C++-klasser för inkapsling av data
- Objekt, typer och klasser

### Fördelar med objektorienterade metoder

- Undvika global data och funktioner
- Tillförlitlighet och underhållsmöjligheter

## Introduktion till C++

### C++ för objektorienterad programmering

- Begränsningar hos C som objektorienterat språk
- Konstruktionsmål för C++
- C++ = C + stark typning + klasser

### Strukturen hos ett C++-program

- C++-syntax
- Parameteröverföring
- Överlagring av funktioner och operationer
- Funktionsspecifikationer och deklarationer
- Separat kompilering; inkludera filer för moduluppbyggnad

### Klasser i C++

- Deklaration och användning av klasser
- Förenkling av klassgränssnitt
- Konstruktörer

## Utnyttja arv

### Härledda klasser

- Polymorfism och dynamisk bindning
- Publika, privata och skyddade
- Delegering kontra arv
- Hierarkisk initialisering
- Nästlade klasser
- Rent virtuella funktioner
- Flerfaldigt arv
- Abstrakta klasser

### Dynamisk typning

- Downcasting av pekare
- Hitta ett objekts typ
- Jämföra typer

## C++ på djupet

### Avancerade typer i C++

- Använda referenstyper
- Använda **const** medlemsfunktioner
- Förbättra driftsäkerhet och effektivitet med **const**

### Avancerade funktioner i C++

- **friends**
- Överlagring av operatorer

- Överlagring av ( ) och [ ]
- "Inline"-funktioner
- "Default"-argument

## Hantera minne

### Statisk lagring

- Statiska datamedlemmar
- Initiera globala data

### Dynamisk minneshantering

- Minneshantering i C++
- **new** och **delete**
- Kopieringskonstruktörer
- Faran med "alias"
- Använda destruktörer
- Definiera tilldelning för att undvika alias

## Standardbibliotek i C++

### Generiska klasser och funktioner

- Återanvändning via typparametrar
- Deklarera "container"-klasser
- Deklarera och använda mallar

### Återanvändbara och portabla bibliotek

- Använda standardalgoritmer: **find**, **for each**, **sort**
- Formatering med I/O-manipulatorer
- Portabla datastrukturer och behållare
- I/O klasshierarki
- Iteratorer
- Lagra data i standardbehållare: list, set, vector

### Undantagshantering

- Felhantering i bibliotek
- Undantag: **catch**, **throw** och **try**
- Hantera undantag på ett säkert sätt
- Undantag i standardbiblioteket

## Underhåll och design i C++

### Organisera ett C++-projekt

- Underhålla C++-applikationer
- Organisera system med hjälp av namespaces
- Styra dynamisk typkonvertering
- Definiera och använda gränssnitt

### Kombinera C med Java och C++

- Länka objektfiler från C och C++
- Konvertera "struct" och globala funktioner till klasser
- Eliminera **case**-fraser
- Gränssnitt mellan Java och C/C++

## Strukturera program med C++, Java och C

- Programorganisation
- Minnestilldelning

- Programkonvertering
- Hitta vanliga konverteringsfel