Övningar
Dag 5 – Defaultkonstruktorer och referenser

Vi kommer i dag att fortsätta bygga vidare på vårt kalender-exempel. Observera att övningarna och tidigare lösningsförslag finns i wikin som finns länkad via Google+-sidan eller direkt via

`www8.cs.umu.se/kursmaterial/C++/Tieto/wiki`

Övning 2, går ut på att fortsätta där du slutade senast, så känn dig inte stressad om du inte han med allt sist.

Övning 1
Öppna projektet Calendar där du skapade klasserna `Timestamp` och `Date`. `Implementera` defaultkonstruktorer för `Date` och `Timestamp`. Fundera på vilka värden attributen i objekten ska ha.

Övning 2
Om du inte är klar förra tillfällets övningar 1-5 så kan du fortsätta jobba på dessa. Observera att övning 3 förra gången hade lite problem relaterat till frågan om default-konstruktorer.

Om du är färdigt men är osäker på om du gjort rätt, kolla på lösningsförslaget i wikin. Du kan även kopiera från lösningsförslaget.

Fördjupningsuppgifter
Läs avsnitten om friend-konstruktioner och överlagring av operatorer i wikin. Länkar till beskrivning av funktionerna finns även länkade på övningsfunktionerna i wikin.

Övning A
Skapa en funktion (ej metod) `void truncateMinutes(Timestamp&)` som sätter klassvariabeln `minutes` till `0` i det inskickade `Timestamp` objektet. För att skriva till den inskickade variabeln så måste din parameter vara en **referens** till en `Timestamp`, och för att din funktion ska kunna läsa/skriva till `Timestamp`:s privata variabler så måste funktionen göras om till en sk. friend-funktion.

Övning B
`toString()` i `Calendar` skriver ut sparade händelser (av klassen `Event`) i samma ordning som de skapades, men man vill ju gärna att en kalender ska vara sorterad efter tid.

Använd `sort` i `std::list` för att sortera listan av `Event`:s innan utskrift.

För att listan ska gå att sortera måste du överlagra operatorn mindre-än < för `Event`. Det är en god ide att även överlagra operatorn < för `Date` och `Timestamp` så att `Event` inte själv behöver kunna läsa privata variabler i de två klasserna.

Övning B - Extra

`sort` i `std::list` kan även sortera listan med en komparatorfunktion istället för att överlagra <-operatorn. Testa skapa en sådan komparatorfunktion för `Event` (du kan fortfarande använda <-operatorn för `Date` och `Timestamp` som du skapade tidigare) och använd funktionen vid sorteringen av listan i `toString()`.

I wikin finns en länk till hur funktionen bör se ut.

Övning C

Överlagra <<-operatorn som en friend-funktion till `Date` och `Timestamp` så att objekt kan skrivas ut direkt med `std::cout` utan att behöva anropa `toString()`–metoden. Du kan t.ex. anropa `toString` i den överlagrade operatorn. `toString` kan vara en bra funktion att ha, samtidigt som kod då inte dubbleras.