Laboration 3

ÖVERLAGRA OPERATORER

OBJEKTBASERAD PROGRAMMERING I C++
Ver 11

Mål: Du ska i denna laboration lära dig att överlagra operatorer

Redovisning: Koden skall vara lättläst där indragningar, kommentarer och variabelnamn är logiska och konsekventa.

Skriv en laborationsbeskrivning enligt anvisning i Moodle.

Skicka in källkod och laborationsbeskrivning via Moodle.
Packa alla filer i en zip-fil.

Regler för inlämning:

Genom att du lämnar in detta arbete försäkrar du att alla svar är skapade av dig själv. Du är även ansvarig att se till att det inte finns någon plagierad text i dokumentet. När du refererar och citerar andra verk måste korrekta källhänvisningar finnas och i fallet citering ska den citerade texten vara tydligt markerad. http://www.bib.miun.se/student/skriva/referenser

Om plagierad text finns i dokumentet riskerar du att stängas av från studier.

Om samarbete sker utan att detta har stöd i instruktionen för examinationen utgör det normalt en disciplinförseelse och du som student riskerar att stängas av från dina studier.

LABORATION 3 Överlagra operatorer

I den här laborationen ska du bygga vidare på klasserna Name, Address och Person som du skapade i laboration 2. Dessa klasser ska utvidgas med överlagring av jämförelseoperatorer, operatorer för att spara/läsa in Person-objekt på/från textfil. Du ska sedan bygga ett program som använder personklassen. I programmet hanteras ett antal Person-objekt i en lista (vector). Programmet byggs objektorienterat.

Överlagringar

I denna laboration ska du för klasserna Name, Address och Person överlagra operatorerna

- < mindre än.
- == likhet.
- << spara på fil.
- >> läs från fil.

När du överlagrar mindre än – operatorn ska du för

- Name jämföra efternamnen – om de är lika jämföra förnamnen.
- Address jämföra stad – om de är lika jämföra gatuadresserna.
- Person jämföra namnen - om de är lika jämföra adresserna.

När du skriver överlagringsfunktionerna får du **inte** använda friend-funktioner.
Du kommer att använda dessa överlagrade operatorer också i laboration 4.

Fortsättning följer på nästa sida!

Klassen PersonList

Skriv en klass med för att hantera en lista (vector) av Person-objekt. Denna klass ska sedan användas i klassen UserInterface, se nästa sida.

Klasspecifikation:

Namn: PersonList

Datamedlemmar: En vector som innehåller personer av klassen (datatypen) Person.
Filnamn (string)

Medlems-funktioner:
Get-funktion för filnamn
Set-funktion för filnamnet
Lägg till en person i listan (vectorn)
Läs en person från listan (vectorn)
Aktuellt antal personer i listan (vectorn)
Sortera listan efter namn
Sortera listan efter personnummer
Sortera listan efter skonummer
Läs listan från fil
Skriv listan till fil

Specifikationer för några av medlemsfunktionerna:

- Lägg till en person
  - Namn: addPerson
  - Skicka in ett Person-objekt som läggs in på första lediga plats i listan
- Läs en person från vectorn
  - Namn: getPerson
  - Skicka in index och returnera Person-objektet som har detta index i vectorn.
- Sorterafunktionerna
  - Namn: sortName, sortPersnr, sortShoenr
  - Void-funktioner utan parametrar
  - Använd gärna algoritmen sort()
- Läs vectorn från fil
  - Namn: readFromfile
  - Läs från filen med namnet *filnamn* (datamedlem) till listan
  - Denna funktion ska inte ha någon parameter
- Skriv vectorn på fil
  - Namn: writeToFile
  - Skriv listan på filen med namnet *filnamn*
  - Denna funktion ska inte ha någon parameter

De funktioner som inte är beskrivna ovan får du själv bestämma vad de ska innehålla.

Inga utskrifter till skärmen får göras från dessa funktioner.

Inga inmatningar från tangentbordet får göras till dessa funktioner.

Allt informationsutbyte med funktionerna ska göras via parametrar.

Placera klassen i ett eget filpar, headerfil och definitionsfil.

Klassen UserInterface

Skriv en klass som utgör gränsskiktet mellan användaren och personlistan. I denna klass görs allt informationsutbyte mellan användaren och personlistan. På detta sätt skiljer man på data och gränssnitt mot användaren.

Klasspecifikation

Namn:	UserInterface
Datamedlem:	En personlista. Datatyp: PersonList
Medlems-	En funktion som programmet körs i. Kalla den *run()*.
funktioner	Meny
	Lägg till en person
	Skriv ut listan med personer
	Spara på fil
	Läs från fil
	Sortera efter namn
	Sortera efter personnummer
	Sortera efter skonummer

Låt medlemsfunktionen run() vara **public** och **alla** de övriga medlemsfunktionerna **private**.

Specifikationer för några av medlemsfunktionerna:

- run()
  - I denna funktion körs programmet.
  - Låt den innehålla en meny från vilken de övriga medlemsfunktionerna anropas
- meny
  - skriv ut menyalternativen
  - låt användaren välja alternativ
  - anropa lämplig medlemsfunktion från klassen PersonList
- Lägg till en person
  - Låt användaren mata in data för en person till ett Person-objekt
  - Anropa medlemsfunktionen *Lägg till en person* från PersonList med ovanstående Person-objekt som argument
- Skriv ut listan med personer
  - Skriv ut inmatade personer på skärmen
- Spara på fil
  - Låt användaren ange filnamn
- Läs från fil
  - Låt användaren ange filnamn

Övriga medlemsfunktioner utformar du efter eget tycke.

Placera klassen i ett eget filpar, headerfil och definitionsfil.

Klientprogram

Skriv ett klientprogram som ser ut så här

```
#include "UserInterface.h"
int main()
{
  UserInterface userinterface;
  userinterface.run();
  return 0;
}
```