PROJEKT

OBJEKTBASERAD PROGRAMMERING I C++
2019-02-22
V3.2

Mål: Du ska i projektet visa att du kan:

- skapa klasser och använda klasser
- använda containrar i STL
- använda algoritmer i STL
- skapa och använda dynamiska listor
- spara på och läsa från textfiler
- skapa god programstruktur

Redovisning: Projektet redovisas med programmets **källkod** och en projekt-beskrivning. Se bilaga för vad projektbeskrivning ska innehålla.

Packa källkod tillsammans med projektbeskrivningen och skicka in den packade filen via Moodle.

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 kod eller 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

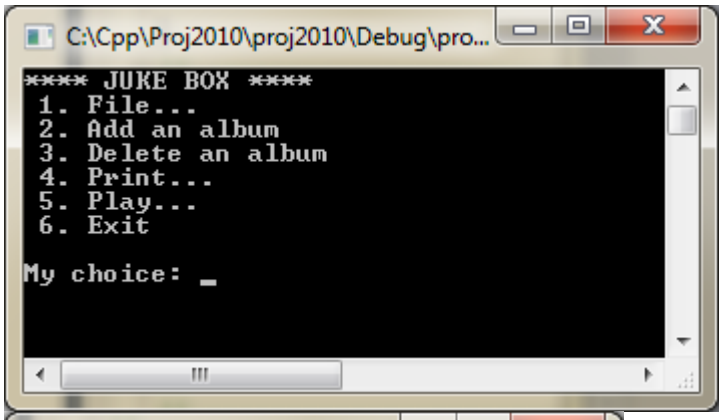
Om plagierad kod eller text finns i dokumentet riskerar du att stängas av från studier.

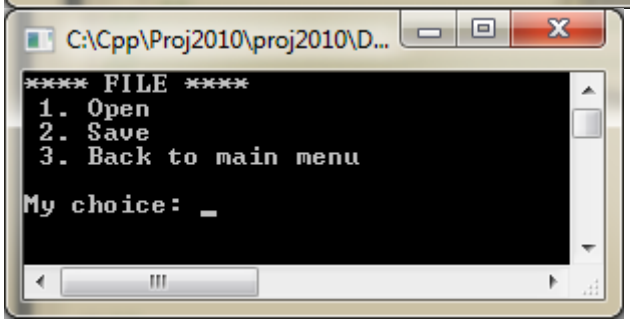
Om samarbete sker utan att detta har stöd i instruktionen för examinationen utgör det normalt en disciplinförseelse och du som student riskerar att stängas av från dina studier.

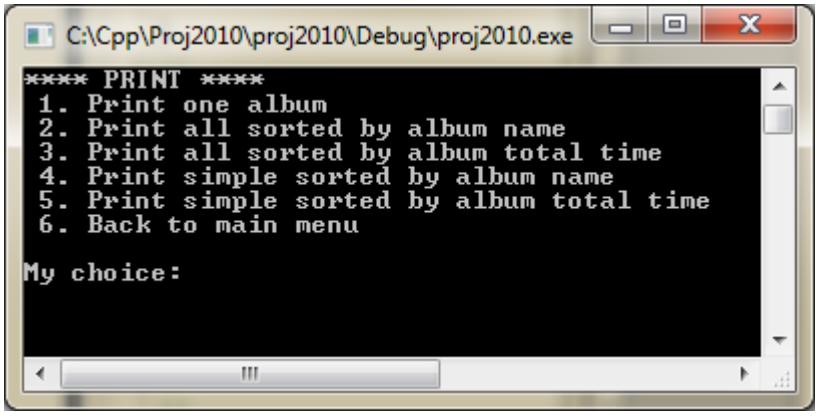
Inledning

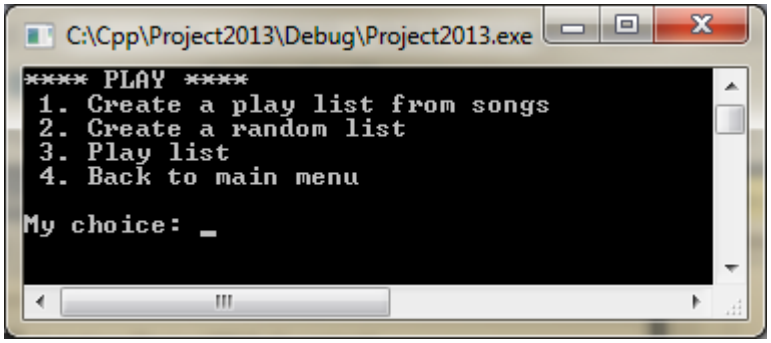
Du ska skapa ett program vars uppgift är att hantera uppgifter om sånger (låtar, tracks) som ligger på ett album (CD/LP) . Ett antal album ligger i en Jukebox och man ska i denna Jukebox kunna lägga in låtar i ett album, spara album på fil, läsa album från fil, göra spellistor, ”spela upp” spellistorna mm.

Programmet ska styras från menyerna:









Läs igenom **hela** dokumentet innan du börjar fundera på hur du ska lägga upp koden!

Klasser

Här kommer först en översikt över de klasser som programmet ska byggas av:

Song:	Innehåller information om en sång (en låt, ett track), sångens namn, artist och sångens längd
Time	En klass för att hantera tid i form av timmar, minuter och sekunder.
Album:	Albumets namn och ett antal sånger
Jukebox:	Innehåller ett antal album och är gränssnittet mot användaren . Här finns menyerna, en spellista och funktionerna som anropas i menyerna.
Menu:	En menyklass som används för huvudmeny och undermenyer
MenuItem	En hjälpklass till klassen Menu. Innehåller texten för ett menyalternativ och om alternativet är valbart (enabled) eller inte.
Queue	En kö i form av en dynamisk lista. Du ska skriva klassen själv. Kön ska användas till att skapa en spellista. Det som läggs in först ska spelas först!

Kravspecifikation

Klassen Song

Namn: Song
Klassen ska hantera data om en sång (låt, track)

Datamedlemmar

- Sångens titel (string)
- Sångens artist (string)
- Sångens längd (Time). Klassen Time beskrivs nedan

Medlemsfunktioner

- lämpliga konstruktorer och destruktor om så krävs
- set- och getfunktioner för klassens datamedlemmar

Överlagringar

- Överlagra inström (>>) och utström (<<) så att dessa kan användas för att läsa från fil och spara på fil. Formatet i filen ges av bifogade fil, `jukebox.txt`. För denna klass är formatet: titel|artist|längd (string|string|int). Sångens längd sparas som dess längd omräknat till sekunder! Se klassen Time!

Utmatning till skärm och inmatning från tangentbord är inte tillåtet från den här klassen

Klassen Time

Namn: Time
Klassen ska hantera data för tid!

Datamedlemmar

- timmar (int)
- minuter (int)
- sekunder (int)

Medlemsfunktioner

- lämpliga konstruktorer och destruktor om så krävs
- set- och getfunktioner för klassens datamedlemmar

Överlagringar

- operatorn + : addera två tider (Time-objekt)
- operatorn < : jämför om en tid (Time-objekt) är mindre än den andra
- operatorn == : jämför om två tider (Time-objekt) är lika
- Överlagra inström (>>) och utström (<<) så att dessa kan användas för att läsa från fil och spara på fil. Sångens längd sparas som timmar + minuter + sekunder omvandlade till sekunder (int)!

Utmatning till skärm och inmatning från tangentbord är inte tillåtet från den här klassen

---

Klassen Album

Namn: Album
Klassen ska kunna hantera ett antal sånger. Man kan tänka sig dessa sånger (låtar, tracks) ligger på en CD eller LP.

Datamedlemmar

- Albumets namn (string)
- en lista med låtar (Song) som använder behållarklassen std::vector

Medlemsfunktioner

- lämpliga set- och getfunktioner för
- Lägg till en sång. Skicka ett song-objekt som parameter.

Överlagringar

- Överlagra <-operatorn som jämför total speltid för de i albumen ingående sångerna.
- Överlagra inström (>>) och utström (<<) så att dessa kan användas för att läsa från fil och spara på fil. Formatet i filen ges nedan och i bifogad fil, `jukebox.txt`.
  Filformat:

```
Albumets namn
Antal sånger i albumet
sångtitel|artist|sånglängd
sångtitel|artist|sånglängd
```

  Osv.

Skriv gärna fler medlemsfunktioner om du tycker att några saknas. De ovan uppräknade funktionerna måste dock vara med.

Utmatning till skärm och inmatning från tangentbord är inte tillåtet från den här klassen

Klassen Jukebox

Namn: Jukebox
Klassen ska hantera ett antal Album och användas som gränssnitt mot användaren.

Datamedlemmar

- en lista med Album. Använd behållarklassen std::vector .
- ett menyobjekt för varje meny (Menu)
- en spellista (Queue). *För högre betyg än C.*

Medlemsfunktioner

- default constructor
  - här initieras menyerna
- run()
  - I denna funktion körs programmet.
  - Låt den innehålla en meny från vilken de övriga medlemsfunktionerna och undermenyerna anropas
  - När programmet startas ska man bara kunna välja *File* och sedan *Open* eller *Back to main menu* i File-menyn samt *Exit* i huvudmenyn.
- Lägg till ett album

  För högre betyg än C. Testa om ett album redan finns i listan så att dubbletter inte läggs in. Använd albumets namn för att identifiera ett album.
- Ta bort ett album. Använd albumets namn som söknyckel
- Anropa undermenyn File
- Anropa undermenyn Print
- Anropa undermenyn Play *För högre betyg än C.*

Undermenyn File

- Open. Läs in en fil till listan med album. Använd en konstant för filnamnet
  - Om det vid inläsning redan finns ett antal album i listan ska de raderas!
  - Alla menyval ska vara valbara (enabled) sedan detta menyval har körts!
- Save. Spara listan på fil. Använd en konstant för filnamnet.
- Tillbaka till huvudmenyn

Undermenyn Print

- Skriv ut ett album på skärmen. Använd albumets namn som söknyckel
- Skriv ut alla album sorterade alfabetiskt efter albumens namn. Skriv all information (namn, artist, låtar och låttid) om varje album.
- Skriv ut alla album sorterade efter total låttid i respektive album. Skriv all information (namn, artist, låtar och låttid) om varje album. Längst total låttid först.
- Skriv ut alla albumnamn sorterade alfabetiskt efter albumens namn. Skriv bara albumens namn.
  - Använd algoritmen for_each() för att stega genom listan av album och skriva ut respektive albums namn.
- Skriv ut alla albumnamn sorterade efter total låttid i respektive album med längst tid först. Skriv albumens namn och totaltid.
- Tillbaka till huvudmenyn

Format för utskrift av tid. Om h > 0 : (h)h:mm:ss
Om h<=0 : (m)m:ss
(h) och (m) innebär att inledande nolla (0) **inte** ska skrivas ut!

Sökning av ett albums namn görs i ett antal av medlemsfunktionerna i klassen Jukebox. Sökningen ska i alla funktioner göras oberoende av om albumets namn eller söknyckeln skrivs med versaler och/eller gemener. Även sortering av albumen med avseende på albumens namn ska göras oberoende av om albumets namn skrivs med versaler och/eller gemener

Undermenyn Play *För högre betyg än C.*

- Skapa en spellista från de sånger som finns i albumen
    1. Skriv en numrerad lista med alla befintliga sånger på skärmen
    2. Låt användaren välja ett antal sånger genom att skriva t.ex. 2, 10, 23 (en sträng). Det ska inte vara nödvändigt att ange siffrorna i ordning! T.ex. ska 10, 2, 23 vara en möjlig sifferkombination.
    3. Lägg in valda sånger i spellistan. Spellistan ska vara en kö (Queue)
- ”Spela” sångerna i spellistan
    - Simulera att sångerna spelas genom att ta dem från spellistan (kön) och skriv ut dem en och en med några sekunders mellanrum på skärmen.
    - När man tar sånger från spellistan(kön) så töms den eftersom den spelas. För att kunna spela samma spellista en gång till ska spellistan kopieras till en temporär spellista som är den spellista som ”spelas” (töms). Gör man på detta sätt finns originalet kvar och kan kopieras på nytt om användaren vill spela igen!
- Slumpa en spellista
    - Skapa en spellista som är en blandlista med sånger från alla sånger som finns i jukeboxen.
    - Sångerna ska väljas slumpvis från befintliga sånger.
    - Dubbletter får inte förekomma.
    - Låt användaren ange antal sånger som ska ingå blandlistan.

Allmänt i klassen Jukebox:

- Låt medlemsfunktionen run() vara **public** och **alla** de övriga medlemsfunktionerna **private**.
- Använd gärna den förenklade for-loopen, iteratorer och/eller algoritmer för att stega genom listan med Album (std::vector).
- Använd algoritmer (find(), find_if())för sökning i listan.

Klassen Queue *För högre betyg än C*

Klassen ska hantera en dynamisk array (C-array) med element av klassen Song som en äkta kö! I en äkta kö kan man enbart lägga till element i ena änden och ta bort element i den andra.

Namn: Queue

Datamedlemmar

- pekare till en array (C-array)
- arrayens storlek
- index som hanterar köns första element och sista elementet.

Medlemsfunktioner

- lämpliga konstruktorer och destruktor om så krävs
- lägg till data sist i kön
- ta bort data från början av kön
- överlagring av operator(er) som bör göras i en dynamisk klass

När ett objekt av klassen Queue skapas ska kön kunna innehålla upp till 5 sånger. Om det krävs plats för fler sånger ska arrayens storlek ökas med 5 element åt gången. Det ska vara möjligt att tilldela en kö till en annan kö.

Klassen Menu

Namn: Menu
Klassen hanterar en meny, se nedan!

Datamedlemmar

- en std::vector av datatypen (klassen) MenuItem (se nedan)
- en rubrik för menyn t.ex JUKEBOX, FILE etc. Se ovan. Datatyp string.

Medlemsfunktioner

- lämpliga konstruktorer och destruktor om så krävs
- addItem(string menuText, bool enabled). Lägger till ett menyalternativ i vectorn
- lämpliga setfunktioner
- printMenuItems(): skriver menyalternativen på skärmen
- getMenuChoice() låter användaren välja menyalternativ
- lämpliga getfunktioner

Menyn är tänkt att användas så här:
1) Deklarera ett menu-objekt

```
Menu menu;
```

2) Lägg in menyalternativ:

```
menu.addItem("File...", true);            // 1
menu.addItem("Add an album", false);      // 2
menu.addItem("Delete an album", false);   // 3
menu.addItem("Print...", false);          // 4
menu.addItem("Play...", false);           // 5
menu.addItem("Exit", true);               // 6
```

3) Kör menyn:

```
bool again = true;
do
{
  menu.printMenuItems();
  switch(menu.getMenuchoice())  // Låt användaren välja menyalternativ
  {                             // Kör valt delprogram
    case  1: file();
     // osv ….
    case  6: again = false;
   }
  }while(again);
```

Du får själv fundera ut var det är lämpligt att lägga de olika koddelarna!

Klassen MenuItem

Namn: MenuItem
Klassen används av klassen Menu för att enkelt kunna hantera menytext och valbarhet för ett menyalternativ i samma datatyp.

Datamedlemmar

- menytext (string)
- valbar (enabled) (bool)

Medlemsfunktioner

- lämpliga konstruktorer och destruktor om så krävs
- lämpliga set- och getfunktioner

Allmänt för hela projektet

- Globala variabler är inte tillåtna
- Friendfunktioner är inte tillåtna
- Globala konstanter är tillåtna.
- **const** – deklarera parametrar mm där så är möjligt
- Använd C++ 11 – kod om så är möjligt. C-kod är inte ok.
- Använd den förenklade for-loopen[1)] där så är möjligt!
- Inkludera alla bibliotek, t.ex. #include <string>, som behövs för att koden ska kunna kompileras, även om din kompilator gör det automatiskt för något/några bibliotek.
- Programkod i main() ska enbart bestå av

```
Jukebox jukebox;
jukebox.run();
```

1) for(auto idx: songs) {…;}

Bedömning

Projektet ger något av betygen A, B, C, D, E, F eller Fx.
Detta betyg blir också kursbetyg.

Följande punkter kommer att bedömas

- Kravspecifikationen
  - Är alla krav i specifikationen uppfyllda
  - Observera att kravspecifikationen är skriven för hela betygsskalan, E-A. De avsnitt som krävs för betyg A eller B är märkta med *För högre betyg än C*
- Källkod
  - konsekvent typografi
  - indentering
  - beskrivande variabelnamn
  - kommentering av kod
  - användning av STL (bl.a. iteratorer, algoritmer)
- Struktur
  - uppdelning i klasser
  - uppdelning i filer
  - placering av konstanter
- Användarvänlighet
  - Informativa utskrifter
  - Enkel inmatning
- Projektbeskrivning
  - Innehållet enligt specifikationen i bilaga 1.
- Provkörning av programmet

I varje avseende som din lösning tydligt avviker från beskrivningen och kravspecifikationen görs ett *påpekande*. Antalet påpekanden påverkar slutbedömningen av projektet.

Betyg

E max 8 påpekanden
D max 5 påpekanden
C max 2 påpekande
B Godkänd för betyg C och godkänt på tre av uppgifterna för högre betyg än C
A Godkänd för betyg C och godkänt på alla uppgifterna för högre betyg än C

Uppgifter för högre betyg än C

- test av att dubbletter av album inte läggs in
- klassen Queue korrekt skriven
- funktionen för att skapa spellistan korrekt skriven
- funktionen för att spela spellistan korrekt skriven
- funktionen för att skapa blandlistan korrekt skriven

Projektbeskrivning

Skriv ett dokument som innehåller rubrikerna:

Projektdata
Filer
Slutsatser och kommentarer

Vad ska det finnas med under respektive rubrik

Projektdata
Ditt namn och loginid i portalen
Projektets namn och vilket datum du lämnar in det.
Vilket betyg du siktar på!
Ange vilken utvecklingsmiljö du använder

Filer
Skriv en lista som innehåller de filer du skickar in.

Slutsatser och kommentarer
Beskriv vilka erfarenheter du gjort under projektet. Har du fått användning av vad du lärt under kursen? Vad har varit svårt, vad har varit lätt?

Du får även gärna ge kommentarer till kursen som helhet. Jag tar gärna emot konstruktiv kritik och förslag till förbättringar.

Lycka till!
Awais Ahmad