NTNU
Kunnskap for en bedre verden

Institutt for datateknologi og informatikk

Kontinuasjonseksamensoppgave i PROG1003 – Objekt-orientert programmering

Faglig kontakt under eksamen:	**Frode Haug**
Tlf:	**950 55 636**
Eksamensdato:	**3.august 2020 - HJEMME**
Eksamenstid (fra-til):	**09:00-13:00 (4 timer)**
Hjelpemiddelkode/Tillatte hjelpemidler:	**F - Alle trykte og skrevne.** (kalkulator er *ikke* tillatt)
Annen informasjon:	
Målform/språk:	**Bokmål**
Antall sider (inkl. forside):	**8**

Informasjon om trykking av eksamensoppgaven

Originalen er:

1-sidig X **2-sidig** ☐

sort/hvit X **farger** ☐

Skal ha flervalgskjema ☐

Kontrollert av:

_____

Dato Sign

NB: Oppgave 1a, 1b, 1c og 2 er totalt uavhengige og kan derfor løses separat.

Oppgave 1 (30%)

I hele oppgave 1, være nøye med å :

- skrive *hele* funksjonen, dvs. *både* headingen og *hele* innmaten
- bruke `const` og/eller '&' (når dette er aktuelt) ifm. parametre
- kommentere funksjonen (hva den gjør, evt. parametre og evt. returverdi) etter Doxygen

a)

Lag funksjonen `int antallImellomToBokstaver(.....)`

Funksjonen tar en tekst (`string`) og to tegn (`char`) som parametre.
Vi forutsetter at tegnene er *bokstaver*, og at den første *alltid* kommer før den andre i alfabetet. Den returnerer *totalantallet* av hvor mange bokstaver det til sammen er i teksten som alfabetisk ligger *imellom* de to bokstavene. Altså totalt antall bokstaver som er *større enn* (*ikke* lik) den første bokstaven *og mindre enn* (*ikke* lik) den andre bokstaven. **NB:** store og små bokstaver *skal* telle likt, uansett om de to `char`-parameterne er små eller store bokstaver.

Eks: Parameter-teksten: «Bergen er tidvis flott Ikaria gikk i vasken Konte er i august»
Antall bokstaver imellom 'a' og 'r' er: **30** (altså det funksjonen returnerer)

b)

Lag funksjonen `void dupliserAnnenhvertElement(.....)`

Funksjonen tar *kun* en `vector` med `int`'er som parameter. Den går gjennom *den originale vectoren*, og oppdaterer alle elementene/«skuffene» med *partalls indeks* til å bli lik det forrige (oddetalls) elementet. Vi ignorerer element nr.0 (start altså med nr.2).

Eks: vectoren før: 0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38
vectoren etter: 0 2 **2** 6 **6** 10 **10** 14 **14** 18 **18** 22 **22** 26 **26** 30 **30** 34 **34** 38

c)

Lag funksjonen `void flyttOver(.....)`

Funksjonen tar to parametre: en `<list>` og en `<vector>`, begge med pekere til objektet `A`. Listen inneholder null eller flere elementer/pekere. Vektoren er tom. Funksjonen skal *flytte* (*ikke* kopiere) *alle* elementene fra den *originale* medsendte listen over til den *originale* medsendte vektoren. I det funksjonen avslutter skal altså listen være helt tom, og vektoren inneholde *alle* elementene fra listen, i den *samme rekkefølgen* som de hadde i listen.

Oppgave 2 (70%)

Les *hele* teksten for denne oppgaven (2a-2g) *nøye*, før du begynner å besvare noe som helst. Studér vedlegget, som inneholder mange viktige opplysninger som du trenger/skal bruke. Legg spesielt merke til `constene`, `enumene`, de tre klassene med datamedlemmer og (mange ferdiglagde) funksjoner, global variabel, `main` og `skrivMeny()`. Husk også på de ferdiglagde funksjonene på `LesData2.h`. Bruk *alt* dette *svært* aktivt.

Vi skal lage et program som holder orden på nettarrangementer (dvs. *kun* konserter og foredrag).

Datastrukturen

Datastrukturen består *kun* av `map`en `gArrangementer` (se vedlegget). I denne ligger det *kun* `Musikk-` og `Foredrag`-objekter. Det finnes maksimalt ett arrangement på en gitt dato, derfor er datamedlemmet `dato` unik. Denne er på formen ÅÅMMDD (år-måned-dag). Likenavnede foredrag (men *ikke* musikk-arrangement) kalles å være *i en og samme serie* (mer i oppgave 2C og 2E).

Oppgaven

a) **Skriv innmaten til** `void skrivAlt()` **og de 3x virtuelle** `skrivData()`
Den første funksjonen skriver *helt til slutt* ut på skjermen antall arrangementer i datastrukturen. *Før* dette sørger den for at (sammen med de tre andre funksjonene) at *alle* arrangementene får skrevet ut *alle* sine data på skjermen (på ett eller annet format).

b) **Skriv innmaten til** `void skrivTidsperiode()`
Funksjonen kommer med en melding om det *ikke* finnes noen arrangementer. I motsatt fall spør den om to datoer, der den siste er senere eller lik den første. Deretter skriver den ut alle data om alle arrangementer som er f.o.m den første datoen og t.o.m. den siste. Den teller også opp dette antallet arrangementer, og skriver det ut helt til slutt i funksjonen.

c) **Skriv innmaten til** `void skrivForedragsserie()`
Funksjonen kommer her også med en melding om det *ikke* finnes noen arrangementer. I motsatt fall spør den om et arrangementsnavn. Alle *foredrag* (*ikke* musikk-arrangement) med dette navnet (altså *serien* med foredrag med det samme navn) får skrevet ut alle sine data på skjermen.

d) **Skriv innmaten til** `void slettArrangement()`
Funksjonen leser inn en dato. Om det *ikke* finnes et arrangement på denne datoen, kommer det en egen melding. I motsatt fall skrives alle arrangementets data ut, og deretter slettes det helt fra hukommelsen og datastrukturen. **NB:** Blir et foredrag inni en serie slettet, trenger *ikke* alle de etterfølgende foredragene i serien å få sitt `serieNr` redusert med minus *en*.

e) **Skriv innmaten til** `void nyttArrangement()`**, de 3x virtuelle** `lesData()` **og evt. andre funksjoner som kreves**

Den første funksjonen spør først om brukeren vil legge inn et nytt `Musikk`- (valget 'M') eller `Foredrag`- (valget 'F') arrangement. Velger brukeren *ikke* 'M' eller 'F' ignoreres resten av funksjonens innmat. Ellers spørres det etter en dato. Finnes denne allerede, kommer en melding, og resten av innmaten ignoreres også. Valgte brukeren 'M' opprettes et nytt `Musikk`-objekt, *alle* dets data leses (vha. `lesData`-funksjoner) og den legges inn i datastrukturen. Ble 'F' derimot valgt, får brukeren to valg: Er foredraget 'N'(ytt) eller fortsettelse av en 'S'(erie)? Velger brukeren 'N', så skjer det samme som ved valget 'M', bare med et nytt `Foredrag`-objekt i stedet. Velges derimot 'S', skal følgende skje:
Det spørres først om navnet for et arrangement. Finnes ikke dette, eller at arrangementet *ikke* er et foredrag, kommer en melding, og resten av funksjonens innmat ignoreres. Ellers finner koden det *siste arrangementet* i denne foredragsserien. *Om* datoen (som tidligere ble lest inn) er *før* dette siste foredragets dato, kommer også en melding, og resten ignoreres. Ellers opprettes et nytt `Foredrag`, den blir en *kopi av det siste* foredraget, dets dato settes til det ønskede, og dets `serieNr` økes med 1-en. **Hint:** Til noe av dette siste, trengs det nok ny(e) selvlagd(e) funksjon(er). Til slutt skrives alle dets data, og den legges inn i datastrukturen.

f) **Skriv innmaten til** `void skrivTilFil()` **og de 3x virtuelle** `skrivTilFil(…)`

Disse funksjonene sørger til sammen for at *hele* datastrukturen blir skrevet til filen «NETTARRANGEMENT.DTA». Formatet bestemmer du helt selv, men det **dette *skal* oppgis som en del av besvarelsen**. Antall arrangementer skal *ikke* ligge på filen. La hver post starte med en verdi, som angir om det nå er et `Musikk`- eller `Foredrag`-objekt som nå kommer.

g) **Skriv innmaten til** `void lesFraFil()` **og de 3x constructorene som tar *en* parameter**

Disse funksjonene sørger til sammen for at *hele* datastrukturen blir lest inn fra filen i forrige oppgave (ut fra det formatet du der selv bestemte).

Annet (klargjørende):

- `dato` er altså på formen ÅÅMMDD og skal ved innlesning fra brukeren stort sett ligge mellom `FDATO` (200101) og `SDATO` (301231). Men det er ingen sjekk på dets gyldighet utover dette. Altså kan vi risikere at brukeren angir datoen: 224589 – dvs. 89/45-22. Det får vi bare leve med …
- Du *skal* bruke `LesData2.h` ifm. løsningen av denne oppgaven. Du får nok også bruk for (deler av) pensumets temaer innen STL, men *ikke* bruk templates eller stoff/biblioteker utenfor pensum.
- Gjør dine egne forutsetninger og presiseringer av oppgaven, dersom du skulle finne dette nødvendig. Gjør i så fall klart rede for disse *i starten* av din besvarelse av oppgaven.

Lykke til med også arrangementet «Eksamen»!

FrodeH

Vedlegg til PROG1003, 3.aug. 2020: Halvferdig programkode

```
#include <iostream>                 //  cout, cin
#include <fstream>                  //  ifstream, ofstream
#include <string>                   //  string
#include <map>                      //  map
#include <algorithm>
#include "LesData2.h"               //  Verktøykasse for lesing av diverse data
using namespace std;

const int FDATO = 200101;           ///<  Første mulige dato (1/1-2020).
const int SDATO = 301231;           ///<  Siste mulige dato (31/12-2030).

/**
 *  ArrType (arrangementstype som:  'ikkeSatt', 'musikk' og 'foredrag').
 */
enum  ArrType  {  ikkeSatt, musikk, foredrag  };

/**
 *  Kategori (om er arrangert av:  en 'privat' eller 'offentlig' aktør).
 */
enum  Kategori  {  privat,  offentlig  };

/**
 *  NettArrangement (med dato (unik - max ett arrangement hver dag),
 *                   startklokkeslett, Vippsnummer, ca varighet i timer,
 *                   arrangementsnavn, navn på aktør og en enum 'type').
 */
class NettArrangement {
  private:
      int     dato,               //  = ID - dette er key'en i <map>en,
              klokken,            //         på formen: ÅÅMMDD.
              vippsNr;
      float   varighet;
      string  arrNavn,
              aktor;
  protected:
      ArrType type;

  public:
      NettArrangement(const int dato) {  this->dato = dato;   type = ikkeSatt;
                                  klokken = vippsNr = 0;   varighet = 0.0F;  }
      NettArrangement(ifstream & inn);                      //  Oppgave 2G
      int  hentID() const  {  return dato;  }
      bool erAvType(const ArrType type) const { return (this->type == type); }
      bool liktArrNavn(const string nvn) const  {  return (arrNavn == nvn);  }
      virtual void lesData();                               //  Oppgave 2E
      virtual void skrivData() const;                       //  Oppgave 2A
      virtual void skrivTilFil(ofstream & ut) const;        //  Oppgave 2F
};


/**
 *  Musikk (med kun antall personer (i bandet) på scenen.
 */
class Musikk : public NettArrangement {
  private:
      int  antPersoner;
  public:
      Musikk(const int dato) : NettArrangement(dato)
            {  type = musikk;  antPersoner = 0;  }
      Musikk(ifstream & inn);                               //  Oppgave 2G
      virtual void lesData();                               //  Oppgave 2E
      virtual void skrivData() const;                       //  Oppgave 2A
      virtual void skrivTilFil(ofstream & ut) const;        //  Oppgave 2F
};
```

```
/**
 *  Foredrag (med hvilket nummer i en evt serie, og om 'privat'/'offentlig').
 */
class Foredrag : public NettArrangement {
  private:
      int      serieNr;
      Kategori kategori;
  public:
      Foredrag(const int dato) : NettArrangement(dato)
              {  type = foredrag;   serieNr = 1;   kategori = privat;  }
      Foredrag(ifstream & inn);                          //  Oppgave 2G
      virtual void lesData();                            //  Oppgave 2E
      virtual void skrivData() const;                    //  Oppgave 2A
      virtual void skrivTilFil(ofstream & ut) const;     //  Oppgave 2F
};

void lesFraFil();                   //  Oppgave 2G
void nyttArrangement();             //  Oppgave 2E
void skrivAlt();                    //  Oppgave 2A
void skrivForedragsserie();         //  Oppgave 2C
void skrivMeny();
void skrivTidsperiode();            //  Oppgave 2B
void skrivTilFil();                 //  Oppgave 2F
void slettArrangement();            //  Oppgave 2D


map <int, NettArrangement*>  gArrangementer;    ///<  ALLE arrangementene.


/**
 *  Hovedprogrammet:
 */
int main() {
    char valg;

    lesFraFil();                                          //  Oppgave 2G

    skrivMeny();
    valg = lesChar("\n\nKommando");
    while (valg != 'Q') {
        switch (valg) {
          case 'A': skrivAlt();              break;     //  Oppgave 2A
          case 'T': skrivTidsperiode();      break;     //  Oppgave 2B
          case 'F': skrivForedragsserie();   break;     //  Oppgave 2C
          case 'S': slettArrangement();      break;     //  Oppgave 2D
          case 'N': nyttArrangement();       break;     //  Oppgave 2E
          default:  skrivMeny();             break;
        }
        valg = lesChar("\n\nKommando");
    }

    skrivTilFil();                                        //  Oppgave 2F

    cout << "\n\n";
    return 0;
}

// ---------------------------------------------------------------------------
//                       DEFINISJON AV KLASSE-FUNKSJONER:
// ---------------------------------------------------------------------------
/**
 *  Oppgave 2G - Leser inn ALLE egne data fra fil.
 *
 *  @param  inn  -  Filobjektet det leses inn data fra
 */
NettArrangement::NettArrangement(ifstream & inn) {         /*  LAG INNMATEN  */    }
```

```
/**
 *  Oppgave 2E - Leser inn ALLE egne data fra tastaturet.
 */
void NettArrangement::lesData() {                          /*  LAG INNMATEN  */    }


/**
 *  Oppgave 2A - Skriver ALLE egne data ut på skjermen.
 */
void NettArrangement::skrivData() const {                  /*  LAG INNMATEN  */    }


/**
 *  Oppgave 2F - Skriver ALLE egne data ut på fil.
 *
 *  @param   ut  -  Filobjektet det skrives ut data til
 */
void NettArrangement::skrivTilFil(ofstream & ut) const {   /*  LAG INNMATEN  */    }


// ---------------------------------------------------------------------------
/**
 *  Oppgave 2G - Leser inn egne data fra fil.
 *
 *  @param  inn  -  Filobjektet det leses inn data fra
 *  @see    NettArrangement::NettArrangement(...)
 */
Musikk::Musikk(ifstream & inn) : NettArrangement(inn) {    /*  LAG INNMATEN  */    }


/**
 *  Oppgave 2E - Leser inn ALLE egne data fra tastaturet.
 */
void Musikk::lesData() {                                   /*  LAG INNMATEN  */    }


/**
 *  Oppgave 2A - Skriver ALLE egne data ut på skjermen.
 */
void Musikk::skrivData() const {                           /*  LAG INNMATEN  */    }


/**
 *  Oppgave 2F - Skriver ALLE egne data ut på fil.
 *
 *  @param   ut  -  Filobjektet det skrives ut data til
 */
void Musikk::skrivTilFil(ofstream & ut) const {            /*  LAG INNMATEN  */    }


// ---------------------------------------------------------------------------
/**
 *  Oppgave 2G - Leser inn egne data fra fil.
 *
 *  @param  inn  -  Filobjektet det leses inn data fra
 *  @see    NettArrangement::NettArrangement(...)
 */
Foredrag::Foredrag(ifstream & inn) : NettArrangement(inn) { /*  LAG INNMATEN  */   }


/**
 *  Oppgave 2E - Leser inn ALLE egne data fra tastaturet.
 */
void Foredrag::lesData() {                                 /*  LAG INNMATEN  */    }
```

```
/**
 *  Oppgave 2A - Skriver ALLE egne data ut på skjermen.
 */
void Foredrag::skrivData() const {                          /*  LAG INNMATEN  */    }

/**
 *  Oppgave 2F - Skriver ALLE egne data ut på fil.
 *
 *  @param   ut  -  Filobjektet det skrives ut data til
 */
void Foredrag::skrivTilFil(ofstream & ut) const {           /*  LAG INNMATEN  */    }


// ---------------------------------------------------------------------------
//                        DEFINISJON AV ANDRE FUNKSJONER:
// ---------------------------------------------------------------------------
/**
 *  Oppgave 2G - Leser hele datastrukturen (ulike arrangementer) inn fra fil.
 */
void lesFraFil() {                                          /*  LAG INNMATEN  */    }

/**
 *  Oppgave 2E - Legger inn (om mulig) ett nytt arrangement (Musikk/Foredrag).
 */
void nyttArrangement() {                                    /*  LAG INNMATEN  */    }

/**
 *  Oppgave 2A - Skriver ALT om ALLE arrangementene ut på skjermen.
 */
void skrivAlt() {                                           /*  LAG INNMATEN  */    }

/**
 *  Oppgave 2C - Skriver ALLE foredragene i en NAVNGITT SERIE ut på skjermen.
 */
void skrivForedragsserie() {                                /*  LAG INNMATEN  */    }


/**
 *  Skriver programmets menyvalg/muligheter på skjermen.
 */
void skrivMeny() {
    cout << "\nFølgende kommandoer er tilgjengelige:\n"
         << "\n   A  - skriv Alle arrangementer"
         << "\n   T  - skriv alle arrangement i en viss Tidsperiode"
         << "\n   F  - skriv en hel Foredragsserie"
         << "\n   S  - Slett arrangement"
         << "\n   N  - Nytt arrangement"
         << "\n   Q  - Quit / avslutt";
}

/**
 *  Oppgave 2B - Skriver ALLE arrangementene i en TIDSPERIODE ut på skjermen.
 */
void skrivTidsperiode() {                                   /*  LAG INNMATEN  */    }


/**
 *  Oppgave 2F - Skriver hele datastrukturen (ulike arrangementer) ut til fil.
 */
void skrivTilFil() {                                        /*  LAG INNMATEN  */    }


/**
 *  Oppgave 2D - Sletter (om mulig) ett gitt arrangement på en gitt dato.
 */
void slettArrangement() {                                   /*  LAG INNMATEN  */    }
```