Høgskolen i Gjøvik
Avdeling for teknologi

E K S A M E N

FAGNAVN: **Grunnleggende programmering**

FAGNUMMER: **IMT 1031**

EKSAMENSDATO: **21. desember 2004**

KLASSE(R): **04HBIND*, 04HBINE*, 04HBINFA, 04HBMETEA, 04HBGEOA**

TID: **09.00-13.00**

FAGLÆRER: **Frode Haug**

ANTALL SIDER UTLEVERT: **8 (inkludert denne forside)**

TILLATTE HJELPEMIDLER: **Alle trykte og skrevne**

- Kontroller at alle oppgavearkene er til stede.
- Innføring med penn, eventuelt trykkblyant som gir gjennomslag. Pass på så du ikke skriver på mer enn ett innføringsark om gangen (da det blir uleselige gjennomslag når flere ark ligger oppå hverandre).
- Ved innlevering skilles hvit og gul besvarelse, som legges i hvert sitt omslag.
- Oppgavetekst, kladd og blå kopi beholder kandidaten til klagefristen er over.
- Ikke skriv noe av din besvarelse på oppgavearkene. Men, i oppgavetekst der du skal fylle ut svar i tegning/tabell/kurve, skal selvsagt dette innleveres sammen med hvit besvarelse.
- Husk kandidatnummer på alle ark. Ta vare på dette nummeret til sensuren faller.

Eksamenssettet består av to ulike oppgavetyper:

Oppgave 1 omhandler hva som blir utskriften fra/av to ulike programmer.

Oppgave 2 omhandler et litt større programmerings-case.

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

Oppgave 1 (30 %)

a) Hva blir utskriften fra følgende program (litt hjelp: det blir 5 linjer):

```
#include <iostream>
#include <cstring>
using namespace std;

const int X = 32;
char txt[] = "GRUNNLEGGENDE PROGRAMMERING MED FRODEH ER SPESIELT FESTLIG!";

int main()  {
     int i, j = 3;

     for (i = X;  i < strlen(txt)-1;  i++)  cout << txt[i];
     cout << '\n';

     for (i = 6;  i < 25;  i += j)
       if (txt[i] != 'E'  &&  txt[i] != 'R')  cout << txt[i] << ' ';
     cout << '\n';

     while (j <= X-i)
       cout << txt[5*j++] << '\n';
     return 0;
}
```

b) Hva blir utskriften fra følgende program (litt hjelp: det blir 5 linjer):

```
#include <iostream>
#include <cstring>
using namespace std;

class A  {
  private:
    char txt[20];
    int  tall;
  public:
    A()            {  strcpy(txt, "GOSSINGUTT");  tall = 44;  skriv(); }
    A(char t[], int ta) {  strcpy(txt, t); tall = ta;            }
    void skriv()       {  cout << txt << '\t' << tall  << '\n';  }
    bool funk1(char c)  {  return (txt[0] == c);          }
    char funk2()       {  return (txt[2]);               }
    char funk3(int n)   {  return (char(int(txt[2])+n));  }
};

int main()  {
     A a1, a2("SLAPPFISK", 19);
     a2.skriv();
     cout << a1.funk1('S') << '\n';
     cout << a2.funk1(a1.funk2()) << '\n';
     cout << a1.funk3(5) << '\n';
     return 0;
}
```

Oppgave 2 (70 %)

NB: Les *hele* teksten for denne oppgaven *nøye*, før du begynner å besvare noe som helst. Studer vedlegg 1, som inneholder mange viktige opplysninger som du trenger/skal bruke. Legg spesielt merke til (og bruk) const'ene, enum'en, klassene og deres medlems-funksjoner, globale variable og funksjonene: `int les_og_finn_oppsetning()` og `int les(char t[], int min, int max)`**.**

I denne oppgaven skal du lage et lite program som holder orden på en masse aktører (skuespillere, sangere og dansere) og hvilke oppsetninger disse til enhver tid deltar i.

Datastrukturen

En aktør er representert vha. klassen 'Aktor'. Det finnes MAXAKT slike objekter, representert vha. arrayen 'aktorer'. En oppsetning er representert vha. klassen 'Oppsetning'. Det finnes MAXOPPS slike objekter, representert vha. arrayen 'oppsetninger'. I begge disse er indeksene f.o.m. 1 t.o.m. henholdsvis 'siste_aktor' og 'siste_oppsetning' i bruk. Arrayen 'ensemble' inni hvert 'Oppsetning'-objekt henviser til indeksene (i 'aktorer') for de aktørene som deltar i den aktuelle oppsetningen.
På den første siden og øverst på den andre siden av vedlegg 1 kan du se deklarasjoner/definisjoner av const'er, enum'en, klassene 'Aktor' og 'Oppsetning' og globale variable. *Dette skal være alt du trenger av klasser, datamedlemmer og globale variable for å løse denne eksamensoppgaven!*

Oppgaven

a)

Lag funksjonene `void skriv_aktorer()` og `void Aktor::skriv(int n)`

Funksjonene sørger for at det gås gjennom *alle* 'Aktor'-objektene som er i bruk. For hvert av dem skrives dets nummer (parameteren 'n' som er dets indeks i arrayen) og de to data-medlemmene til skjermen – *en* linje pr.aktør. Ved utskriften må 'profesjon' gjøres om til en passende tekst. Finnes det ingen aktive aktører skal det komme en melding om det.

b)

Lag funksjonene `void ny_aktor()` og `void Aktor::ny()`

Er det *ikke* plass til flere aktører kommer det en melding om det. I motsatt fall tas neste indeks i 'aktorer'-arrayen i bruk, og vedkommende objekt leser resten av *alle* sine data selv.Den må sørge for at brukeren angir en *lovlig* profesjon, og at dette gjøres om til aktuell enum-verdi.

c)

Lag funksjonene `void skriv_oppsetning()` og `void Oppsetning::skriv()`

Finnes det ingen aktive oppsetninger skal det komme en melding om det. I motsatt fall kalles funksjonen `les_og_finn_oppsetning()`. Om denne *ikke* finner en oppsetning med dette navnet, kommer det en melding om det. I motsatt fall bes objektet om å skrive *alt* om seg selv, dvs. *alle* sine datamedlemmer inkludert *alle* data om *alle* aktørene i ensemblet (bruk/kall `Aktor::skriv(int n)`).

d)

Lag funksjonene `void ny_oppsetning()` og `void Oppsetning::ny()`

Er det *ikke* plass til flere oppsetninger eller er det *ikke* registrert noen aktører kommer det to ulike meldinger om dette. I motsatt fall tas neste indeks i 'oppsetninger'-arrayen i bruk, og vedkommende objekt leser resten av *alle* sine data selv.
Bruk/kall `int les(char t[], int min, int max)` *alle* de stedene det er rimelig.

e)

Modifiser funksjonen(e) du laget i 2c, slik at når aktørene skrives ut, så skrives de med

samme profesjon samlet (rett etter hverandre). Dvs. først skrives alle skuespillerne, deretter alle sangerne og til slutt alle danserne. Det holder at du viser hvilke nye koderlinjer du må lage evt. endre i forhold til det du har kodet/laget i oppgave 2c.

f) **Lag funksjonene** `void skriv_oppsetninger_paa_tidspunkt()` og `bool Oppsetning::spilles(int d)`

Den siste funksjonen skal returnere 'true'/'false' til om 'd' ligger mellom objektets 'start_dato' og 'slutt_dato' eller ei. Den første funksjonen skal spørre brukeren om en dato. Deretter går den gjennom alle 'Oppsetning'-objektene og ber *alle* de som spilles den aktuelle dagen om å skrive ut *alle* sine data (bruk/kall den ene av funksjonene du laget i oppgave 2c). Om *ingen* oppsetninger forekommer denne dagen (ingen utskrift har foregått) skal det komme en melding om også dette.

g) **Lag funksjonene** `void les_fra_fil()`, `void Aktor::sett(……)` og `void Oppsetning::les_fra_fil(……)`

Den første funksjonen sørger for at *alle* data fra *begge* filene angitt i vedlegg 2 blir lest inn i datastrukturen. Til dette bruker den de to andre funksjonene angitt ovenfor. Legg merke til at `Aktor::sett(……)` tar imot to parametre (navn og profesjon) som den oppdaterer sine datamedlemmer med. Dvs. den leser selv *ikke* dataene inn fra fil, den får de bare inn som parameter. Den inni 'Oppsetning' derimot leser resten av *alle* sine data selv, etter at den får inn navnet sitt og filen det skal leses fra som parameter. Husk å oppdatere 'siste_.....'-variablene!

Klargjøring og forutsetninger

- Noen begrensninger ved programmet er:
  - Det er *ikke* mulig å legge til/ta vekk noen i et ensemble etter at funksjonene i 2d er utført.
  - Det er *ikke* mulig å slette/endre noen aktører/oppsetninger etter at de er registrert.
  - En aktør har bare *en* profesjon. De er enten det ene eller det andre, kombinasjon er uaktuelt.
  - En aktør er kun en person som står på scenen. Dvs. vi tar ikke med roller/profesjoner som f.eks: regisør, musikere, scenearbeidere, sminkører, billettører o.l.
- Du trenger ***ikke*** å sjekke slikt som at:
  - datoene er på lovlig format (ååååmmdd) (jfr. oppgave 2d og 2f).
  - det er flere oppsetninger i en og samme sal/scene samtidig.
  - en aktør deltar i flere oppsetninger samtidig.
  - det er duplikate aktør-numre i ett og samme ensamble.
- Gjør dine egne forutsetninger dersom du finner oppgaveteksten upresis eller ufullstendig. Gjør i så fall rede for disse forutsetningene *først* i besvarelsen din.

Uansett hvordan det går i dag: Hold på masken og rollen din, og gjør god mine til slett spill ! ☺

Lykke til !
frode@haug.com

Vedlegg 1: Halvferdig programkode (.tpl-fil)

```
                        //  INCLUDE:
#include <iostream>             //  cin, cout
#include <fstream>              //  ifstream
#include <cstring>              //  strcmp, strcpy
#include <cctype>               //  toupper
using namespace std;

                        //  CONST:
const int MAXAKT   = 500;  //  Max. antall aktører i datastrukturen.
const int MAXOPPS  = 200;  //  Max. antall oppsetninger i datastrukturen.
const int MAXENS   =  30;  //  Max. ant. aktører i en oppsetning (ensemble).
const int STRLEN   =  40;  //  Max. lengde på tekststrenger.

                        //  ENUM:
enum Profesjon  { skuespiller, sanger, danser };

                        //  KLASSER:
class Aktor  {                  //  Aktør:
  private:
        char navn[STRLEN]; //  Navn.
        Profesjon profesjon;   //  Aktørens profesjon (skuesp, sanger, danser).
  public:
        void skriv(int n);      //  Deklarasjon av medlemsfunksjoner:
        void ny();
        void sett(char nvn[], char prof);
        bool har_profesjon(Profesjon prof);
};

class Oppsetning  {             //  Oppsetning:
  private:
        char navn[STRLEN]; //  Navn.
        int  start_dato,        //  Start- og sluttdato for oppsetningen,
        slutt_dato,             //     begge på formen:  åååammdd .
        sal,                    //  Sal/scenen det spilles på/i: Tall fra 1-10.
        antall;                 //  Antall aktører i ensemblet (oppsetningen).
        int ensemble[MAXENS+1];  //  Indeksene i "aktorer" for de i ensemblet.
  public:
        void skriv();           //  Deklarasjon av medlemsfunksjoner:
        void ny();
        bool spilles(int d);
        bool lik(char t[]);
        void les_fra_fil(istream* inn, char nvn[]);
};

                        //  DEKLARASJON AV FUNKSJONER:
void skriv_meny();                      void skriv_aktorer();
char les_kommando();                    void ny_aktor();
int  les(char t[], int min, int max);   void skriv_oppsetning();
int  les_og_finn_oppsetning();          void ny_oppsetning();
                                        void skriv_oppsetninger_paa_tidspunkt();
                                        void les_fra_fil();
```

```
                        //  GLOBALE VARIABLE:
int siste_aktor;                                    //  Indeksen for siste brukte aktør.
int siste_oppsetning;                       //  Indeksen for siste brukte oppsetning.
Aktor aktorer[MAXAKT+1];                    //  Array med Aktor-objekter.
Oppsetning oppsetninger[MAXOPPS+1];   //  Array med Oppsetning-objekter.


int main()  {               //  HOVEDPROGRAM:
  char kommando;

  les_fra_fil();                                                //  Oppgave 2G
  skriv_meny();
  kommando = les_kommando();
  while (kommando != 'Q')  {
     switch (kommando)  {
        case 'S':  skriv_aktorer();                     break;   //  Oppgave 2A
        case 'A':  ny_aktor();                          break;   //  Oppgave 2B
        case 'T':  skriv_oppsetning();                  break;   //  Oppgave 2C  og  2E
        case 'O':  ny_oppsetning();                     break;   //  Oppgave 2D
        case 'P':  skriv_oppsetninger_paa_tidspunkt(); break; //  Oppgave 2F
        default:   skriv_meny();                        break;
     }
     kommando = les_kommando();
  }
 cout << "\n\n";
 return 0;
}

                        //  DEFINISJON AV KLASSE-FUNKSJONER:
void Aktor::skriv(int n)  {                 //  Skriver ALLE aktørens data:
        //  Oppgave 2A:  Lag innmaten
}

void Aktor::ny()  {                         //  Leser inn data om ny aktør:
         //  Oppgave 2B:  Lag innmaten
}

void Aktor::sett(char nvn[], char prof)  {     //  Setter verdier:
        //  Oppgave 2G:  Lag innmaten
}

bool Aktor::har_profesjon(Profesjon prof)  { //  Returnerer om har/er av en gitt profesjon
         return (profesjon == prof);
}

void Oppsetning::skriv()  {                 //  Skriver ALT om oppsetningen:
        //  Oppgave 2C og 2E:  Lag innmaten
}

void Oppsetning::ny()  {                    //  Leser inn data om ny oppsetning:
        //  Oppgave 2D:  Lag innmaten
}

bool Oppsetning::spilles(int d)  {          //  Returnerer ”true” om 'd' er inni tidsintervallet:
```

```
        // Oppgave 2F: Lag innmaten
}

bool Oppsetning::lik(char t[])  {               // Returnerer ”true” om har navnet 't':
        return (!strcmp(navn, t));
}

void Oppsetning::les_fra_fil(istream* inn, char nvn[])  {    // Leser alt om oppsetning fra fil:
        // Oppgave 2G: Lag innmaten
}

                        // DEFINISJON AV FUNKSJONER:
void skriv_meny()  {                            // Presenterer lovlige menyvalg:
  cout << "\n\nFØLGENDE KOMMANDOER ER LOVLIG:\n";
  cout << "\tS = Skriv alle data om en aktør\n";
  cout << "\tA = ny Aktør\n";
  cout << "\tT = skriv alle data om en oppseTning\n";
  cout << "\tO = ny Oppsetning\n";
  cout << "\tP = skriv alle oppsetninger på et gitt tidsPunkt\n";
  cout << "\tQ = quit/avslutt\n";
}

char les_kommando()  {                          // Henter ett ikke-blankt upcaset tegn:
 char ch;
 cout << "\n\nOppgi ønske:  ";   cin >> ch;    cin.ignore();
 return toupper(ch);
}

int les(char t[], int min, int max)  {          // Leser et tall i et visst intervall:
 int n;
 do  {                                          // Skriver ledetekst og leser inn ett tall:
    cout << '\t' << t << " (" << min << '-' << max << "): ";     cin >> n;
 } while(n < min  ||  n > max);                 // Sjekker at i lovlig intervall.
 return n;                                      // Returnerer innlest tall.
}

int les_og_finn_oppsetning()  {                 // Leser og evt. finner oppsetnings indeks:
 char nvn[STRLEN];
 int i;                                         // Leser aktuell oppsetnings navn:
 cout << "\tOppsetnings-navn:  ";     cin.getline(nvn, STRLEN);
 for (i = 1;  i <= siste_oppsetning;  i++)      // Leter etter denne:
    if (oppsetninger[i].lik(nvn)) return i;     // Match: returnerer indeks.
 return 0;                                      // Ingen treff/match.
}

void skriv_aktorer()  {                         // Skriver ALLE aktørene:
        // Oppgave 2A: Lag innmaten
}

void ny_aktor()  {                              // Ny aktør registreres:
        // Oppgave 2B: Lag innmaten
}

void skriv_oppsetning()  {                      // Skriver ALT om en oppsetning:
```

```
        // Oppgave 2C og 2E:  Lag innmaten
}

void ny_oppsetning()  {                    // Ny oppsetning registreres:
        // Oppgave 2D:  Lag innmaten
}

void skriv_oppsetninger_paa_tidspunkt()  { //   Skriv ALLE oppsetninger som går en viss dato:
        // Oppgave 2F:  Lag innmaten
}

void les_fra_fil()  {                      // Leser HELE datastrukturen fra fil:
        // Oppgave 2G:  Lag innmaten
}
```

Vedlegg 2: Eksempler på datafilene

AKTOR.DTA: Format: <Profesjon> <Navn>

```
S Andrea Arnesen
A Bengt Bentzen
D Camilla Collett
S Dorthe Dappel
A Erika Emilsen
D Frode Fredriksen
S Gulla Gabrielsen
A Harriet Hermansen
D Iselin Ingvaldsen
S Janne Jormo
A Kenneth Karlsen
D Laila Liaker
S Magne Moe
A Nora Nielsen
D Oscar Osmundsen
S Petra Pollesen
A Qa Qvale
D Reidun Rotherman
```

OPPSETN.DTA: Format: <Navn>
<Start-dato> <Slutt-dato> <Sal>
<Ant.aktører> <Aktør1> …… <AktørN>

```
Et dukkehjem
20040810 20041218 1
6 1 4 7 10 13 14
Svanesjøen
20040929 20041224 2
8 1 3 4 6 9 12 15 18
```