Høgskolen i Gjøvik
Avdeling for teknologi

E K S A M E N

FAGNAVN: **Grunnleggende programmering**

FAGNUMMER: **IMT 1031**

EKSAMENSDATO: **15. desember 2003**

KLASSE(R): **03HBIND*, 03HBINFA, 03HBINE*, 03HBMETEA, 03HBMEMAA, 03HBGEOA**

TID: **09.00-13.00**

FAGLÆRER: **Frode Haug**

ANTALL SIDER UTLEVERT: **7 (inkludert denne forside)**

TILLATTE HJELPEMIDLER: **Alle trykte og skrevne**

- Kontroller at alle oppgavearkene er tilstede.
- 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>
using namespace std;

char txt[] = "STEINALDERSPILLESTIL";

int main()  {
  int i, j;

  for (i = 5;  i <= 19;  i += j)  {
    cout << txt[i + 10 / 3] << '\n';
    j = i * 2;
  }

  j = 18;  i = j / 3;
  while (j > i-3)  {
    cout << txt[--j] << '\n';
    j -= i;
  }
  return 0;
}
```

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

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

class A  {
  private:
    int  tall;
    char tegn;
  public:
    A()               {  tall = 99;  tegn = 'w';   skriv();    }
    A(int t, char c)  {  tall = t;   tegn =  c-2;              }
    void skriv()      {  cout << tall << ' ' << tegn  << '\n'; }
    bool funk1(A a)   {  return (tegn == a.tegn);             }
    void funk2(int & t, int & c)  {  t = tall/3;  c = int(tegn);  }
    int  funk3()      {  return (tall + 6 / 3 * 4);           }
};

int main()  {
  A a1, a2(66, 'y');
  int te, ch;

  a2.skriv();
  cout << a1.funk1(a2) << '\n';

  a1.funk2(te, ch);  cout << te << ' ' << ch << '\n';
  cout << a2.funk3() << '\n';    return 0;
}
```

Oppgave 2 (70 %)

NB: **Les *hele* teksten for denne oppgaven *nøye*, før du begynner å besvare noe som helst. Studer vedlegget, som inneholder mange viktige opplysninger som du trenger/skal bruke. Legg spesielt merke til (og bruk) const'ene, klassen og dets medlemsfunksjoner, globale variable og funksjonen "les_tekst(…)".**

I denne oppgaven skal du lage et lite program som holder orden på ulike lag/foreninger/klubber m.m. og deres medlemmer/deltagere/spillere/…

Datastrukturen

Et lag/forening/klubb er representert vha. klassen 'Enhet'. Det finnes MAXENHET slike objekter, representert vha. arrayen 'enheter'. Av denne er indeksene f.o.m. 1 t.o.m. 'siste_enhet' i bruk. Navnene på de personene som er deltagere i en 'Enhet' er lagt inn i vedkommende objekts to-dimenjonale char-array 'delt_navn'. Her er indeksene/linjene f.o.m. 1 t.o.m. 'ant' i bruk inni et aktuelt objekt.
På den første siden av vedlegget kan du se deklarasjoner/definisjoner av const'er, klassen "Enhet" og de globale variablene. *Dette skal være alt du trenger av klasser, datamedlemmer og globale variable for å løse denne eksamensoppgaven!*

Oppgaven

a)

Lag funksjonene void skriv_alle() og void Enhet :: skriv()

Funksjonene sørger for at det gås gjennom *alle* 'Enhet'-objekter som er i bruk. For hvert av dem skrives *alle* dets datamedlemmer til skjermen, *unntatt* navnene som ligger lagret i 'delt_navn'.

b)

Lag funksjonen int finn_enhet(char t[])

Funksjonen får inn navnet den skal lete etter som parameteren 't'. Den finner så ut hvilket objekt som har dette navnet (vha. 'Enhet :: er_lik(…)'). Funksjonen returnerer dette objektets indeks i arrayen 'enheter'. Om ingen har dette navnet, så returnerer den 0 (null).
NB: Denne funksjonen skal brukes i/kalles av funksjonene du lager i 2c, 2d og 2f.

c)

Lag funksjonene void skriv_enhet() og void Enhet :: skriv_alt()

Funksjonene spør først om navnet for en enhet (bruk 'les_tekst(…)'). Om denne enheten *ikke* finnes (bruk 'finn_enhet(…)'), så kommer det en melding om det. I motsatt fall så skriver vedkommende objekt ut *alle* sine data. Dvs. det skriver ut *alt* det som det skrev ifm. oppgave 2a, pluss *alle* deltagernes navn, ett og ett på hver sin linje.

d) **Lag funksjonene** void ny_deltager() og void Enhet :: ny()

Funksjonene spør først om navnet for en enhet (bruk 'les_tekst(…)'). Om denne enheten *ikke* finnes (bruk 'finn_enhet(…)'), så kommer det en melding om det. I motsatt fall så leser objektet selv inn navnet til den nye deltageren/medlemmet (bruk også her 'les_tekst(…)'). Om det *ikke* er plass til flere deltagere inni objektet, så må det komme en melding om dette. **NB:** Du trenger *ikke* å sjekke for duplikater eller at vedkommende også er deltager i en annen enhet!

e) **Lag funksjonene** void finn_deltager() og void Enhet :: finn(char t[])

Funksjonen 'finn_deltager' leser først inn navnet på en potensiell deltager (bruk 'les_tekst(…)'). Det gås så gjennom alle de brukte objektene. For hvert av dem tilkalles dets 'finn'-funksjon. Denne funksjonen går så gjennom objektets deltagere og sjekker om det aktuelle navnet finnes
(vha. 'strcmp'). Om så er tilfelle, skrives de samme dataene som i oppgave 2a.

f) **Lag funksjonene** void slett_deltager() og void Enhet :: slett()

Funksjonen 'slett_deltager' spør først om navnet for en enhet (bruk 'les_tekst(…)'). Om denne enheten *ikke* finnes (bruk 'finn_enhet(…)'), så kommer det en melding om det. I motsatt fall så kalles funksjonen 'slett' i det aktuelle objektet. Denne funksjonen presenterer først indeksen for og navnene til alle deltagerne. Den spør så om en indeks mellom 1 og 'ant'. Denne deltageren blir så fjernet fra listen over deltagere for vedkommende enhet.

g) **Lag funksjonene** void les_fra_fil() og void Enhet :: les_fra_fil(istream* inn, char buf[])

Funksjonene leser inn hele datastrukturen fra filen "ENHET.DTA". På *hver sine linje* ligger en enhets navn, adresse, email, telefon og antall deltagere. Etter dette siste tallet (antall deltagere) så kommer dette antallet linjer med navnene til/på de ulike deltagerne. (Altså: *en* deltagers navn, som *kan* bestå av flere ord, pr.linje.) **NB:** Husk å oppdatere variabelen 'siste_enhet'!

Klargjøring og forutsetninger

- Legg merke til at du i denne eksamensoppgaven *ikke* skal lage funksjoner for å:
    - legge inn en ny enhet
    - slette en enhet
    - skrive til fil
- Gjør dine egne forutsetninger dersom du finner oppgaveteksten upresis eller ufullstendig. Gjør i så fall rede for disse forutsetningene *først* i besvarelsen din.

Lykke til !
frode@haug.com

Vedlegg: Halvferdig programkode (.tpl-fil)

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

                        //  CONST:
const int MAXENHET  =    30;       //  Max. antall enheter i datastrukturen.
const int MAXPERS   =   100;       //  Max. antall personer i enheten.
const int STRLEN    =    40;       //  Max. lengde på tekststrenger.


                        //  KLASSE:
class Enhet  {                  //  Enhet:
  private:
    char navn[STRLEN];      //  Navn.
    char adr[STRLEN*2];     //  Adresse (gate, postnr, poststed).
    char email[STRLEN];     //  Email-adresse
    int  tlf;               //  Telefonnummer.
    int  ant;               //  Antall medlemmer/deltagere/spillere/.....
    char delt_navn[MAXPERS][STRLEN];  //  2-dimensjonal char-array med navnet
                                      //     til/på enhetens deltagere.
  public:
    Enhet();                //  Deklarasjon av medlemsfunksjoner:
    bool er_lik(char t[]);
    void skriv();
    void skriv_alt();
    void ny();
    void finn(char t[]);
    void slett();
    void les_fra_fil(istream* inn, char buf[]);
};

                            //  DEKLARASJON AV FUNKSJONER:
void skriv_meny();
char les_kommando();
void les_tekst(char t[], char s[], int LEN);
void skriv_alle();
int  finn_enhet();
void skriv_enhet();
void ny_deltager();
void finn_deltager();
void slett_deltager();
void les_fra_fil();

                        //  GLOBALE VARIABLE:
int siste_enhet;                //  Indeksen for siste brukte enhet.
Enhet enheter[MAXENHET+1];      //  Array med enhet-objekter.
```

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

  les_fra_fil();                        //  Oppgave 2g
  skriv_meny();
  kommando = les_kommando();
  while (kommando != 'Q')  {
    switch (kommando)  {
      case 'A':  skriv_alle();     break;  //  Oppgave 2a
      case 'E':  skriv_enhet();    break;  //  Oppgave 2c
      case 'N':  ny_deltager();    break;  //  Oppgave 2d
      case 'F':  finn_deltager();  break;  //  Oppgave 2e
      case 'S':  slett_deltager(); break;  //  Oppgave 2f
      default:   skriv_meny();     break;
    }
    kommando = les_kommando();
  }
  return 0;
}

                        //  DEFINISJON AV KLASSE-FUNKSJONER:
Enhet :: Enhet()                //  Nullstiller/initierer alle medlemmene:
 {  ant = tlf = 0;   strcpy(navn, "");   strcpy(adr, "");   strcpy(email, "");   }

                                //  Returnerer true/false til om objektet har et gitt navn:
bool Enhet :: er_lik(char t[])  {  return (!strcmp(navn, t));  }

void Enhet :: skriv()  {        //  Skriver alle HOVEDDATA om objektet:
        //  Oppgave 2A:  Lag innmaten
}

void Enhet :: skriv_alt()  {    //  Skriver ALLE DATA om objektet:
        //  Oppgave 2C:  Lag innmaten
}

void Enhet :: ny()  {           //  Legger inn (om plass) navnet til/for en ny deltager i enheten:
        //  Oppgave 2D:  Lag innmaten
}

void Enhet :: finn(char t[])  { //  Leter etter en gitt deltager i enheten:
        //  Oppgave 2E:  Lag innmaten
}

void Enhet :: slett()  {        //  Sletter (om mulig) en deltager i enheten:
        //  Oppgave 2F:  Lag innmaten
}

void Enhet :: les_fra_fil(istream* inn, char buf[])  {//  Leser alle data om en enhet fra fil:
        //  Oppgave 2G:  Lag innmaten
}
```

```
                        //  DEFINISJON AV FUNKSJONER:
void skriv_meny()  {            //  Presenterer lovlige menyvalg:
  cout << "\n\nFØLGENDE KOMMANDOER ER LOVLIG:\n";
  cout << "\tA = skriv data om Alle enhetene\n";
  cout << "\tE = skriv ALLE data om EN Enhet\n";
  cout << "\tN = Ny deltager i en enhet\n";
  cout << "\tF = Finn en deltager\n";
  cout << "\tS = Slett/fjern en deltager\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);
}

void les_tekst(char t[], char s[], int LEN) {  //  Leser inn ikke-blank tekst:
  do  {
    cout << '\t' << t << ": ";      //  Skriver ledetekst.
    cin.getline(s, LEN);            //  Leser inn tekst.
  } while (strlen(s) == 0);         //  Sjekker at tekstlengden er ulik 0.
}

void skriv_alle()  {                //  Skriver HOVEDDATAENE om ALLE enhetene:
        //  Oppgave 2A:  Lag innmaten
}

                                    //  Ut fra enhetens navn, så returneres dens
int finn_enhet(char t[])  {         //     indeks i "enheter", evt.'0' om ikke finnes:
        //  Oppgave 2B:  Lag innmaten
}

void skriv_enhet()  {               //  Skriver ALLE data om en enhet:
        //  Oppgave 2C:  Lag innmaten
}

void ny_deltager()  {               //  Legger inn en ny deltager i en enhet:
        //  Oppgave 2D:  Lag innmaten
}

void finn_deltager()  {             //  Skriver ALLE enheter i deltager er med i:
        //  Oppgave 2E:  Lag innmaten
}

void slett_deltager()  {            //  Sletter (om mulig) en deltager:
        //  Oppgave 2F:  Lag innmaten
}

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