Høgskolen i Gjøvik
Avdeling for teknologi

KONTINUASJONSEKSAMEN

FAGNAVN: **Grunnleggende programmering**

FAGNUMMER: **IMT 1031**

EKSAMENSDATO: **12. august 2004**

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 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>
using namespace std;

char txt[] = "KROMKAKER-OG-MANDELTERTEBUNN-MED-JORDBÆRSAUS-OG-KREM";

int main()  {
  int i, j;

  for (i = 3;  i <= 46;  i += j)  {
     cout << txt[(2*i) +  (i/2)] << '\n';
      j = i * 3;
  }

  j = 36;  i = j % 10;
  while (j != i)  {
    cout << txt[j+1] << '\n';
    j -= 10;
  }
  return 0;
}
```

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

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

class B  {
  private:
     int  tall;
     char tegn;
  public:
     B()               { tall = 123;  tegn = 'S';   skriv();    }
     B(int t, char c)  { tall = t-19;    tegn =  c+2;          }
     void skriv()      { cout << tall << ' ' << tegn  << '\n';  }
     bool funk1(B b)   { return (tegn <=  b.tegn);             }
     void funk2(int & t, char & c)  { t = tall * 3 ;  c = char (tall/3); }
     int  funk3()      { return (tall % 4 * 12);               }
};

int main()  {
  B b1(321, 'q'), b2;
  int te;  char ch;

  b1.skriv();
  cout << b1.funk1(b2) << '\n';
  b1.funk2(te, ch);  cout << te << ' ' << ch << '\n';
  cout << b2.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_nummer()`

I denne oppgaven skal du lage et lite program som holder orden på ulike deltagere og deres årlige deltagelse på ett eller annet arrangement (korpsstevne, speiderleir, turnering, o.l.) i tenårene.

Datastrukturen

En deltager (tenåring) er representert vha. klassen 'Deltager'. Det finnes MAXDELT slike objekter, representert vha. arrayen 'deltagere'. Av denne er indeksene f.o.m. 1 t.o.m. 'siste_brukt' i bruk.
Om vedkommende deltok på det årlige arrangementet i sine tenår (det *kalenderåret* vedkommende *fylte* 13-19 år - et syv års intervall) er respresentert vha. den boolske arrayen 'deltatt'. Der indeks nr.0 representerer alderen 13 år, og så videre oppover til 19 år (indeks nr.6).
På den første siden av vedlegget kan du se deklarasjoner/definisjoner av const'er, klassen 'Deltager' 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 Deltager::skriv()`

Funksjonene sørger for at det gås gjennom *alle* 'Deltager'-objekter som er i bruk. For hvert av dem skrives *alle* dets datamedlemmer (på *en* linje) til skjermen, *unntatt* det som er i 'deltatt'.

b)

Lag funksjonen `int finn_deltager(int n)`

Funksjonen får inn nummeret den skal lete etter som parameteren 'n'. Den finner så ut hvilket objekt som har dette nummeret (vha. `Deltager::er_lik(…)`). Funksjonen returnerer dette objektets indeks i arrayen 'deltagere'. Om ingen har dette nummeret, så returnerer den 0 (null).
NB: Denne funksjonen skal brukes i/kalles av funksjonene du lager i 2c, 2d, 2e og 2g.

c)

Lag funksjonene `void skriv_deltager()` og `void Deltager::skriv_alt()`

Funksjonene spør først om nummeret til en deltager (bruk `les_nummer()`). Om denne deltageren *ikke* finnes (bruk `finn_deltager(…)`), 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 *alt* i 'deltatt' (på *en* linje). Hver utskrift av en indeks i 'deltatt' skrives som: "N år:" (der N er et tall mellom 13 og 19) etterfulgt av ordet "Ja" eller "Nei".

d)

Lag funksjonene `void ny_deltager()` og `void Deltager::ny(int n)`

Om det *ikke* er plass til flere deltagere, så må det først og fremst komme en melding om det. Funksjonene spør så om nummeret til en deltager. Om denne deltageren finnes allerede, så kommer det en melding om det. I motsatt fall så leser objektet selv inn resten av alle sine datamedlemmer, unntatt 'deltatt' som får alle sine verdier satt til 'false'.

e) **Lag funksjonene** `void slett_deltager()`

Funksjonen spør først om nummeret til deltager. Om denne deltageren *ikke* finnes, så kommer det en melding om det. I motsatt fall så flyttes den siste deltageren til den plassen der den slettede har vært, og 'siste_brukt' blir telt ned med 1.

f) **Lag funksjonene**
```
void les_fra_fil()  og
void Deltager::les_fra_fil(istream* inn, int n)
```

Funksjonene leser inn *hele* datastrukturen fra filen "TENARING.DTA". Opplysningene om hver deltager ligger som en post fordelt over fire linjer: På *hver sine linje* ligger nummer, navn og adresse. På den fjerde linjen ligger først fødeåret, etterfulgt av syv nuller eller enere (adskilt med blanke imellom). Disse syv skal gjøres om til 'false/true' i vedkommendes 'deltatt'-array.
NB: Husk å oppdatere variabelen 'siste_brukt'!

g) **Lag funksjonene** `void les_deltagelse()` og `void Deltager::vaert_med(int aar)`

Funksjonen `les_deltagelse()` leser inn opplysninger fra fil om hvem (numre) som har deltatt på arrangementet et gitt år. Filen "DELTAGER.DTA" inneholder *kun* tall. Først ligger årstallet filen gjelder for. Så kommer fortløpende (til filens slutt) numrene til de deltagerne som har vært med det aktuelle året. Om et deltagernummer ikke finnes, så kommer det en melding om det, og funksjonen leser deretter bare det neste nummeret fra filen.
Funksjonen `Deltager::vaert_med(int aar)` må finne ut hvor mange år vedkommende fyller det aktuelle året ('aar' – 'fodeaar'). Om vedkommende *ikke* er tenåring dette året, så kommer det en melding om det. I motsatt fall så sjekkes det om vedkommende faktisk har deltatt dette året allerede, i så fall kommer det nok en melding. I motsatt fall oppdateres vedkommende 'deltatt' verdi med 'true'.

Klargjøring og forutsetninger

- Legg spesielt merke til at:
  - deltagerne godt kan bli liggende i datastrukturen *lenge* etter at de har passert tenårene. (Dette går meget bra, også om de senere skulle dukke opp på filen "DELTAGER.DTA", koden i oppgave 2g sørger for det!).
  - programmet *kun* håndterer en masse personers deltagelse på *ett* fast årlig arrangement.
  - den alderen vedkommende *fyller* det aktuelle *kaldenderåret* er den indeksen det skal slås opp på i 'deltatt'-arrayen, selv om vedkommende egentlig har bursdag *etter* selve arrangementet.
- Du trenger *ikke* å sjekke om fødeåret er en fornuftig verdi.
- 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>          // strcpy
#include <cctype>           // toupper

using namespace std;

                        // CONST:
const int MAXDELT = 200;        // Max. antall tenårings-deltagere.
const int STRLEN  =  40;        // Max. lengde på tekststrenger.
const int TENLEN  =   7;        // Antall år man er tenåring.
const int TENSTART=  13;        // Første tenåringsår.
const int TENSLUTT=  19;        // Siste tenåringsår.

                        // KLASSE:
class Deltager  {               // Tenåringsdeltager:
  private:
        int  nr;                // Nummer/ID.
        char navn[STRLEN];      // Navn.
        char adr[STRLEN*2];     // Adresse (gate, postnr, poststed).
        int  fodeaar;           // Fødeselsår.
        bool deltatt[TENLEN];   // Hvilke tenår (13-19) man har deltatt.
  public:
        Deltager();             // Deklarasjon av medlemsfunksjoner:
        bool er_lik(int n);
        void skriv();
        void skriv_alt();
        void ny(int n);
        void les_fra_fil(istream* inn, int n);
        void vaert_med(int aar);
};

                        // DEKLARASJON AV FUNKSJONER:
void skriv_meny();
char les_kommando();
int  les_nummer();

void skriv_alle();
int  finn_deltager();
void skriv_deltager();
void ny_deltager();
void slett_deltager();
void les_fra_fil();
void les_deltagelse();

                        // GLOBALE VARIABLE:
Deltager deltagere[MAXDELT+1];      // Tenåringsdeltager-objekter.
int siste_brukt;                    // Indeksen for siste brukte deltager.
```

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

  les_fra_fil();                              //  Oppgave 2F

  skriv_meny();
  kommando = les_kommando();
  while (kommando != 'Q')  {
     switch (kommando)  {
         case 'A':  skriv_alle();          break;  //  Oppgave 2A
         case 'E':  skriv_deltager();      break;  //  Oppgave 2C
         case 'N':  ny_deltager();         break;  //  Oppgave 2D
         case 'S':  slett_deltager();      break;  //  Oppgave 2E
         case 'D':  les_deltagelse();      break;  //  Oppgave 2G
         default:   skriv_meny();          break;
     }
    kommando = les_kommando();
  }
 cout << "\n\n";
 return 0;
}

                        //  DEFINISJON AV KLASSE-FUNKSJONER:
Deltager::Deltager()  {          //  Nullstiller/initierer alle medlemmene:
 nr = fodeaar = 0;
 strcpy(navn, "");  strcpy(adr, "");
 for (int i = 0;  i < TENLEN;  i++)  deltatt[i] = false;
}
                                 //  Returnerer true/false til om objektet har et gitt nummer:
bool Deltager::er_lik(int n)  {
 return (nr == n);
}

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

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

void Deltager::ny(int n)  {     //  Leser inn ALLE data til objektet:
        //  Oppgave 2D:  Lag innmaten
}

void Deltager::les_fra_fil(istream* inn, int n)  {      //  Leser alle ALLE om en deltager fra fil:
        //  Oppgave 2F:  Lag innmaten
}

void Deltager::vaert_med(int aar)  {                    //  Setter (om mulig) at har deltatt et gitt år:
        //  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 deltagerne\n";
  cout << "\tE = skriv ALLE data om EN deltager\n";
  cout << "\tN = Ny deltager\n";
  cout << "\tS = Slett/fjern en deltager\n";
  cout << "\tD = les Deltagelse fra fil\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_nummer()  {                     //  Henter et potensielt deltagernummer:
  int nr;
  cout << "\n\tDeltagernummer:  ";   cin >> nr;   cin.ignore();
  return nr;
}

void skriv_alle()  {                    //  Skriver HOVEDDATAENE om ALLE deltagerne:
        //  Oppgave 2A:  Lag innmaten
}
                                        //   Ut fra deltagerens nummer, så returneres dens
int finn_deltager(int n)  {             //       indeks i 'deltagere', evt.'0' om ikke finnes:
        //  Oppgave 2B:  Lag innmaten
}

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

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

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

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

                                //  Leser hvem (nummer) som har deltatt på arrangementet et gitt år:
void les_deltagelse()  {
        //  Oppgave 2G:  Lag innmaten
}
```