Høgskolen i Gjøvik
Avdeling for teknologi

KONTINUASJONSEKSAMEN

FAGNAVN:	**Grunnleggende datakunnskap og programmering**
FAGKODE:	**L 182 A**
EKSAMENSDATO:	**9. august 2003**
KLASSE(R):	**02HIND*, 02HINE*, 02HDMU*, 02HING*, 02HGEOMAA, 02HSIV5**
TID:	**09.00-13.00**
FAGLÆRER:	**Frode Haug**
ANTALL SIDER UTLEVERT:	**7 (inkludert denne forside)**
TILLATTE HJELPEMIDLER:	**Kun læreboka ”OOP in C++”**

- 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 tre ulike oppgavetyper:

Oppgave 1 omhandler teori fra datateknikk-delen.
Oppgave 2 omhandler hva som blir utskriften fra noen ulike program.
Oppgave 3 omhandler et litt større programmerings-case.

NB: De tre oppgavene er totalt uavhengige og kan derfor løses separat.

Oppgave 1 (30 %)

a) Internet:

a1) Nevn fem faser som samarbeid går gjennom.

a2) Nevn åtte av de mest vanlige funksjonene/mulighetene i et mail-program.

a3) Forklar kort (max. ½ A4-ark) om filoverføring (FTP).

b) Datamaskinen:

b1) Forklar kort (max. ½ A4-ark) om systembussen.

b2) Forklar kort (max. ½ A4-ark) om mikroprosessoren/CPU'en.

c) Datasikkerhet:

c1) Nevn fem områder/typer for datakriminalitet.

c2) Nevn fem eksempler på "EDB-tekniske sikkerhetstiltak".

d) Personvern / etikk:

d1) Hvilke fem hovedoppgaver har Datatilsynet ?

d2) Hva er forskjellen på "etikk", "moral" og "etiske retningslinjer" ?

Oppgave 2 (20 %)

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

```
#include <iostream>

using namespace std;

const int N   = 23;
const int M   = 4;

int main()   {
  int i=3, j, k;
  while (i < N)  {
    cout << i << ": ";
    for (j = 1;  j <=  i/2;  j += 3)
        cout << j << ' ';
    cout << '\n';
     for (k = 0; k < M; k++)  i++;
  }
  return 0;
}
```

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

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

char tekst[] = "KONTINUASJONSEKSAMEN";

void skriv(char t[], int n, int m)  {
  for (int i = n;  i <= m;  i+=3) cout << t[i] << ' ';
  cout << '\n';
}

char hent(char t[], int nr)
  {  return t[--nr];  }


void rokk(char & a, char & b)
  {  char c = a;  a = b;  b = c;  }


int main() {
  skriv(tekst, 0, strlen(tekst)-2);
  cout << hent(tekst, 4) << ' ' << hent(tekst, 8) << '\n';
  cout << char(hent(tekst, 12)+4) << ' ' << int(hent(tekst, 8)) << '\n';
  rokk(tekst[4], tekst[7]);
  skriv(tekst, 1, 9);
  rokk(tekst[5], tekst[hent(tekst, 14)-'D']);
  skriv(tekst, 1, 3);
  return 0;
}
```

Oppgave 3 (50 %)

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, struct'en og den globale arrayen.

Innledning

I denne oppgaven skal du lage et lite program som holder orden på en arbeiders (snekker, rørlegger, murer, datakonsulent, interiørkonsulent el.l.) oppdrag hos ulike kunder på årets forskjellige datoer. Det utføres kun *ett* oppdrag pr.*ukedag* (dvs. kun *en* kunde blir besøkt pr.*ukedag*).

Datastrukturen

Oppdragene lagres i en to-dimensjonal array ("oppdragene") for hele året. Hver linje representetrer *en* måned og hver kolonne representerer de ulike dagene i hver måned (se vedleggets globale variable). Hva som lagres for hvert oppdrag ser du også i vedlegget (struct'en "Oppdrag"). Vi bruker ikke linje og kolonne nr.0, derfor vil det som f.eks. er lagret i "oppdragene[4][6]" være data om det som skjer 6.april. *Alle deklarasjoner/definisjoner som du strengt tatt skulle trenge for å besvare denne eksamens-oppgaven er ferdig definert i vedlegget.*

Oppgaven

a)

Lag funksjonen void les_inn(int & m, int & d)

Funksjonen ber om et måneds- og dagnummer. Disse *skal* den sørge for at blir liggende i henholdsvis intervallene 1-ANTMND og 1-ANTDAG. De referanseoverførte parametrene oppdateres med de to innleste verdiene.

b)

Lag funksjonen void nytt_oppdrag()

Funksjonen leser først inn et måneds- og dagnummer (vha. funksjonen "les_inn"). Om denne datoen er opptatt med med et oppdrag allerede så gis en melding om det. I motsatt fall leses data inn fra tastaturet til *alle* oppdragets verdier ("navn", "adr", "tlf" og "merknad").

c)

Lag funksjonen void slett_oppdrag()

Funksjonen leser først inn et måneds- og dagnummer (vha. funksjonen "les_inn").Om denne datoen *ikke* er opptatt med med et oppdrag allerede så gis en melding om det. I motsatt fall så nullstilles *alle* oppdragets verdier ("tlf" blir lik 0, de tre tekstene fylles med "" (blank tekst)).

d) **Lag funksjonen** void oversikt()

Funksjonen leser først inn et måneds- og dagnummer (vha. funksjonen ”les_inn”). *Vi forutsetter at dette er en mandag.* Funksjonen går så gjennom denne datoen *og de fire neste* og skriver ut alle deres data til skjermen. Om datoen *ikke* er opptatt med oppdrag, så skrives kun teksten ”LEDIG” ut. For å få full uttelling på denne oppgaven må du *ikke* skrive ut dager som er ulovlige. F.eks. om brukeren skriver inn måned nr.1 og dag nr.29, så skrives kun dataene om dagene 29-31 ut. Du trenger *ikke* å ’konvertere’ over til måned nr.2 og skrive dag nr.1 og 2 for denne også. At en dato er lovlig/gyldig sjekkes opp mot arrayen ”DAGANTALL”.

e) **Lag funksjonen** void les_fra_fil()

Funksjonen leser inn data til datastrukturen fra filen ”OPPDRAG.DTA”. Hver post består av tre linjer. På den første ligger de fire feltene, adskilt med *en* blank: <mnd> <dag> <tlf> <navn>.
På den andre linjen ligger *kun* ”adr”, mens den tredje inneholder *kun* ”merknad”.
Legg merke til at filen *kun* inneholder datoer (mnd og dag) som har et oppdrag tilknyttet seg. Alle andre datoer er altså ledige/tomme og inneholder dermed automatisk ’blanke’ data (de er nullstilt da ”oppdragene” er global).

f) **Lag funksjonen** void skriv_til_fil()

Funksjonen skriver ut *alle* datoer *med oppdrag* til filen ”OPPDRAG.DTA”. Samme format som beskrevet i oppgave e) ovenfor skal brukes. Husk altså at datoer som *ikke* inneholder oppdrag *ikke* skal skrives ut!

Klargjøring og forutsetninger

- Om en gitt dato er ledig for oppdrag så vil dens ”tlf” være *lik* 0. (Og motsatt: om datoen er opptatt med et oppdrag allerede så vil dens ”tlf” være *ulik* 0.)
- Vi tar *ikke* hensyn til:
    - at et oppdrag kan gå over flere dager. Om dette er tilfelle, må brukeren manuelt (vha. kommandoen ”N”) legge inn de samme dataene på de aktuelle datoene.
    - at oppdrag ikke legges inn på lørdager, søndager eller andre helligdager. Dette er det brukerens oppgave å besørge at ikke skjer.
    - skuddår. Dvs. vi lar måned nr.2 (februar) *alltid* inneholde 28 dager.
    - årsskifte. Alle oppdrag gjelder dette *ene* året dataene registreres for.
- 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, ofstream
#include <cstring>              //  strcpy
#include <cctype>               //  toupper

using namespace std;

                        //  CONST:
const int ANTMND = 12;      //  Antall måneder.
const int ANTDAG = 31;      //  Max. antall dager i en måned.
                            //  Antall dager i hver måned  (NB: ignorerer skuddår!):
const int DAGANTALL[] = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
const int STRLEN = 40;      //  Max.lengde for en tekst.

                        //  STRUCT:
struct Oppdrag  {               //  Oppdrag (EN kunde, EN dag):
  char navn[STRLEN];        //  Kundens/stedets navn.
  char adr[STRLEN];         //  Kundens/stedets adresse.
  char merknad[STRLEN];     //  Evt.merknad/kommentar til oppdraget.
  int  tlf;                 //  Kundens tlf.
};

                        //  DEKLARASJON AV FUNKSJONER:
void  skriv_meny();
char  les_kommando();       void  les_inn(int & m, int & d);
void  nytt_oppdrag();       void  slett_oppdrag();
void  oversikt();
void  les_fra_fil();        void  skriv_til_fil();

                        //  GLOBALE VARIABLE:
Oppdrag oppdragene[ANTMND+1][ANTDAG+1];         // Bruker indeksene 1-12(x) og 1-31(y).

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

   les_fra_fil();                           //  Oppgave 3e
   skriv_meny();
   kommando = les_kommando();
   while (kommando != 'Q')  {
     switch (kommando)  {
       case 'N':   nytt_oppdrag();      break;  //  Oppgave 3b
       case 'S':   slett_oppdrag();     break;  //  Oppgave 3c
       case 'O':  oversikt();           break;  //  Oppgave 3d
       case 'F':  skriv_til_fil();      break;  //  Oppgave 3f
       default:   skriv_meny();         break;
     }
     kommando = les_kommando();
   }
  return 0;
}
```

```
                        //  DEFINISJON AV FUNKSJONER:

void skriv_meny()  {            //  Presenterer lovlige menyvalg:
  cout << "\n\nFØLGENDE KOMMANDOER ER LOVLIG:\n";
  cout << "\tN = Nytt oppdrag\n";
  cout << "\tS = Slett oppdrag\n";
  cout << "\tO = Oversikt over en ukes oppdrag\n";
  cout << "\tF = skriv til Fil\n";
  cout << "\tQ = quit/avslutt\n";
}


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

                                //  Leser inn måned og dag. Referanseoverførte
void  les_inn(int & m, int & d)  {              // parametre, så de oppdateres direkte:
        //  Oppgave 3a:  Lag innmaten
}


void  nytt_oppdrag()  {         //  Legger inn/registrerer nytt oppdrag:
        //  Oppgave 3b:  Lag innmaten
}


void  slett_oppdrag()  {        //  Sletter/fjerner eksisterende oppdrag:
        //  Oppgave 3c:  Lag innmaten
}


void  oversikt()  {             //  Skriver oversikt over oppdrag i fem dager:
        //  Oppgave 3d:  Lag innmaten
}


void  les_fra_fil()  {          //  Leser hele datastrukturen fra fil:
        //  Oppgave 3e:  Lag innmaten
}


void  skriv_til_fil()  {        //  Skriver hele datastrukturen til fil:
        //  Oppgave 3f:  Lag innmaten
}
```