Høgskolen i Gjøvik

Avdeling for Teknologi

KONTINUASJONSEKSAMEN

FAGNAVN:	**Grunnleggende datakunnskap og programmering**
FAGNUMMER:	**L 182 A**
EKSAMENSDATO:	**16. august 2000**
KLASSE:	**99HINDA / 99HINDB / 99HINEA 99HDMUA / 99HDMUB / 99HINGA**
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, evt. trykkblyant som gir gjennomslag.
  Pass på at du ikke skriver på mer enn ett innføringsark om gangen (da det blir uleselige gjennomslag om flere ark ligger oppå hverandre når du skriver).
- Ved innlevering skilles hvit og gul besvarelse og legges i hvert sitt omslag.
  Oppgavetekst, kladd og blå kopi beholder kandidaten.
- Ikke skriv noe av din besvarelse på oppgavearkene.
- Husk kandidatnummer på alle ark.

Dette 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 (20 %)

a) Internet:

a1) Nevn fem kategorier for tradisjonelle tjenester på Internet.

a2) Redegjør kort (max. ½ A4-ark) omkring TCP/IP.

b) Datamaskinen (periferiutstyr / innmat):

b1) Redegjør kort (max. ½ A4-ark) om portene på en datamaskin.

b2) Redegjør kort (max. ½ A4-ark) omkring CD-ROM.

b3) Nevn fem ulike typer tilleggskort (utenom hovedkortet) i en PC.

b4) Redegjør kort (max. fire setninger pr. punkt) omkring:

b4-a) Diskorganisering
b4-b) Drivere
b4-c) Lydkort

c) Datasikkerhet:

c1) Hvilke to hovedkategorier for datakriminalitet har vi ?

c2) Nevn fem områder/typer for datakriminalitet.

c3) Nevn fem eksempler på "fysiske sikringstiltak".

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:

```
#include <iostream>

using namespace std;

const int LEN = 15;

int main()  {
  char tekst[] = "SJOKOLADEPUDDING";
  int i = 0;

  while (tekst[i])  {
    if (i % 3 == 0)
       cout << tekst[i] << "  " <<  tekst[LEN - i] << '\n';
    i++;
  }
  return 0;
}
```

b) Hva blir utskriften fra følgende program:

```
#include <iostream>

using namespace std;

struct rec  {
  char  text[10];
  float tall;
};

rec recorder[] = { {"AAA", 1.1}, {"BBB", 2.2}, {"CCC", 3.3},
                   {"DDD", 4.4}, {"EEE", 5.5}, {"FFF", 6.6} };


void funk(rec a)
  { cout << a.text << "  " << a.tall << '\n'; }

void funk(char txt[], float f = 7.7)
  { cout << txt << "  " << f << '\n'; }

void funk(float f, char txt[]="GGG")
  { cout << txt << "  " << f << '\n'; }

int main()  {
  funk(recorder[3]);
  funk(recorder[1].text);
  funk(recorder[2].tall);
  funk(recorder[5].text, 8.8);
  funk(recorder[4].tall, "HHH");
  return 0;
}
```

Oppgave 3 (60 %)

NB: Les hele teksten for denne oppgaven nøye, før du begynner å besvare noe som helst. Studer vedlegg 1 nøye, som inneholder mange viktige opplysninger som du trenger/ skal bruke.

Innledning

I denne oppgaven skal du lage et lite program som holder orden på arbeidstiden for ansatte ved en eller annen arbeidsplass. Ved hoveddøren på arbeidsplassen er det plassert en datamaskin der programmet du skal lage "står og kjører". Det eneste de ansatte gjør er å skrive inn sitt unike ansattnummer. Dette skjer når de kommer til eller går fra arbeidsplassen. Dvs. det eneste som programmet ditt skal håndtere av input er slike stadig innskrevne ansattnumre !
Hver morgen startes programmet på nytt (av den første som ankommer arbeidsplassen). Da leses status for inneværende måned inn fra fil og legges i datastrukturen. Etter hvert som dagen går, og ansatte kommer og går (og skriver sine ansattnumre), så oppdateres disse dataene. Ved arbeidsdagens slutt, vil den siste som går først skrive sitt ansattnummer og deretter skrive "9999". Dette gjør at programmet avsluttes (se hovedprogrammets while-løkke), og at den oppdaterte datastrukturen automatisk skrives til fil igjen.

Datastrukturen

I vedlegg 1 kan du på den første siden se deklarasjoner/definisjoner av const'er, to struct'er og globale variable (en int og en array). Legg merke til den nestede bruken av den ene struct'en inni den andre.
Dette skal være alt det du trenger av slike globale variable for å løse denne eksamensoppgaven !
Datastrukturen vil derfor bestå av 'ansatt'-arrayen. Alle de ANT_ANSATTE indeksene er til enhver tid i bruk. Inni hver 'ansatt' er det bl.a. en to-dimensjonal array. Når programmet kjører en gitt dag, så opereres det på linje nr. 'dagnr' i hver enkelt ansatts 'tider'-array. Kolonne nummer '0' i 'tider' representerer ankomst-tidspunkt, kolonne nr.1 er avgangs-tidspunkt, kolonne nr.2 er arbeidstiden for denne dagen (dvs. differansen mellom kolonne nr.1 og 0).

Under løsningen av denne eksamensoppgaven skal du bruke denne datastrukturen !

Oppgaven

a) **Tegn datastrukturen.** Gjør dette detaljert, pent og klart.

b) **Lag resten av innmaten for while-løkken i "main".**

Når du kommer til dette punktet i koden, vil variablene 'nr', 'time', 'min' og 'dagnr' inneholde sine aktuelle verdier. Det du her skal gjøre er å legge inn 'time' og 'min' i "alle_ansatte[nr].tider [dagnr] [X].Y". X= 0 dersom vedkommende ankommer arbeidsplassen, 1 dersom hun/han går og 2 dersom det er differansen som er beregnet. Y er 'time' eller 'min'. Om vedkommende kommer eller går kan du sjekke ved å se etter om 'time' for vedkommendes ankomst-tidspunkt fortsatt er lik '0'. Er den det, så regner vi med at hun/han ankommer arbeidet.
Ved ankomst skal 'time' og 'min' bare lagres hos vedkommende i datastrukturen.
Ved avgang er det litt mer komplisert: 'time' og 'min' skal da også lagres. Men, vi skal også beregne og lagre unna hvor lang tid hun/han har vært på arbeidet. Dessuten skal det beregnes og lagres vedkommendes totale arbeidstid denne måneden (i 'total_arbtid').

Hint: For å beregne arbeidstiden en gitt dag kan det lønne seg å gjøre om ankomst-tidspunkt og avgangs-tidspunkt til antall minutter siden midnatt (time*60+min). Beregne differansen på disse, og deretter gjøre om igjen til timer og minutter vha '/' (heltallsdeling) og '%' (mod).
'total_arbtid' beregnes på stort sett samme måten, bare at den vil inneholde summen av 'total_arbtid' hittil pluss dagens arbeidstid.

c) Lag funksjonen "void les_fra_fil()".

Vedlegg 2 viser filen det skal leses fra og skrives til. Først på filen kommer et tall som angir hvor mange dager det hittil er lest inn data om. Deretter kommer det repeterende poster som angir data om de ulike ansatte. For hver ansatt ligger først vedkommendes navn på en egen linje. Deretter ligger det data om en og en dag på hver sin linje. De to første tallene på hver linje er ankomst-time og -minutt, deretter kommer avgangs-time og -minutt. Til slutt på linjen kommer arbeidstiden den dagen i timer og minutter. Helt til slutt i posten, på en egen linje, kommer total arbeidstid denne måneden i timer og minutter for vedkommende. Les disse dataene inn i datastrukturen din.

d) Lag funksjonen "void skriv_til_fil()".

Skriv datastrukturen til fil, på samme formatet som angitt ovenfor og i vedlegg 2.
NB: Husk å øke variabelen 'dagnr' !

Klargjøring og forutsetninger

- Legg merke til at funksjonen "int les_ansattnr()" godtar tall fra 1 til ANT_ANSATTE, samt 9999.
- Bruken av #include "timer.h" og funksjonen "void hent_tidspunkt(int& ti, int& mi)" trenger du ikke å bekymre deg med. Det du trenger å vite er at "main"-koden i vedlegg 1 foretar det eneste påkrevde kallet til "hent_tidspunkt(time, min)", og at parametrene blir oppdatert vha. referanseoverføring.
- Ansatt nr.0 finnes/brukes ikke, men vi bruker indeks nr.'0' i begge retningene i 'tider'-arrayen for hver 'ansatt'. Dette vil bl.a. medføre at arbeidsdag nummer 1 er lagret på indeks/linje nr.0.
- I en måned er det maksimalt 23 arbeidsdager (helgene fratrukket), derfor er ANT_DAGER=23.
- Vi gjør følgende forutsetninger/forenklinger:
  - Ingen ansatte ankommer før kl.01.00. Alle forlater arbeidsplassen før midnatt samme dag.
  - Alle ansatte er 100% pålitelige med å bruke programmet korrekt, dvs. alle skriver inn sitt ansattnummer to eksakt ganger de dagene de er på jobb. (Dermed utelukkes det at noen kommer og går flere ganger i løpet av dagen, glemmer å skrive nummeret, o.l.)
  - Ingen arbeider i helgene (på lørdager og søndager).
- Det foretas en del unødvendig dobbellagring av data i programmet: Kolonne nr.2 i "tider" er bare differansen mellom kolonne nr.1 og 0. Samt at 'total_arbtid' er summen av denne kolonne nr.2 for vedkommende ansatt. Dette er selvsagt litt sløsing med hukommelse, men det gjør den manuelle lesingen av filen "ARBTID.DTA" enklere for den som vil ha en enkel oversikt over status.
- Filen "ARBTID.DTA" inneholder data kun om arbeidstiden for inneværende måned. Hva som skjer ved et månedsskifte (angående arkivering og nullstilling av filen) trenger du ikke å bekymre seg med.
- Det er plass til alle dataene fra fil i datamaskinens primærhukommelse.
- 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 1: Halvferdig programkode (.tpl-fil)

```
                        // INCLUDE:
#include <iostream>                 // cin, cout
#include <fstream>                  // ifstream, ofstream
#include "timer.h"                  // For å avlese klokka på datamaskinen.
using namespace std;
                        // CONST:
const int NVN_LEN     = 30;         // Max. lengde for ansattes navn.
const int ANT_DAGER   = 23;         // Max. antall arbeidsdager i en måned.
const int ANT_ANSATTE = 5;          // Eksakt antall ansatte.

                        // STRUCT:
struct tid  {                       // Struct for tidspunkt m/
  int time, min;                            //     time- og minuttangivelse.
};

struct ansatt  {                    // Struct for en ansatt:
  char navn[NVN_LEN];               // Vedkommendes navn
  tid  tider[ANT_DAGER][3];         // 0=kommer,  1=går,  2=dagens arb.tid.
  tid  total_arbtid;                // Total arbeidstid for HELE måneden.
};

                        // DEKLARASJON AV FUNKSJONER:
void les_fra_fil();                 // Oppgave 3c
void skriv_til_fil();               // Oppgave 3d
int  les_ansattnr();
void hent_tidspunkt(int& ti, int& mi);

                        // GLOBALE VARIABLE:
int dagnr;                          // Nåværende arbeidsdag i måneden.
ansatt alle_ansatte[ANT_ANSATTE+1]; // Tidsdata om alle de ansatte.

int main()  {           // MAIN:
  int time, min, nr;                // Avlest tidspunkt (time, min) og ansattnr.
  les_fra_fil();                    // Leser datastrukturen fra fil.
  nr = les_ansattnr();              // Leser et ansattnummer.
  while (nr != 9999)  {                     // Så lenge ikke skal avslutte:
     hent_tidspunkt(time, min);     // Avleser tidspunkt på datamaskinen.

        // Oppgave 3b:  Lag resten av innmaten

     nr = les_ansattnr();           // Leser et nytt ansattnummer.
  }
  skriv_til_fil();                  // Skriver datastrukturen til fil.
  return 0;
}

                        // DEFINISJON AV FUNKSJONER:
void les_fra_fil()  {               // Leser hele datastrukturen fra fil:

        // OPPGAVE 3c:     Lag innmaten
}
```

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

int les_ansattnr()  {                            //  Leser et LOVLIG ansattnummer:
  int num;
  do  {
    cout << "Ansattnummer (1-" << ANT_ANSATTE << "): ";
    cin >> num;                                          //  MÅ være i rett intervall eller "9999":
  } while ( num < 1  ||  (num > ANT_ANSATTE  &&  num != 9999) );
  return num;
}

void hent_tidspunkt(int& ti, int& mi)  {       // Avleser nåværende tidspunkt (time og minutt).
  Timer tid;      int sek;
  tid.hent(ti, mi, sek);
}
```

Vedlegg 2: Eksempel på datafil (”arbtid.dta”)

```
4
Arne Andersen
7 45   15 45  8 0
7 45   15 45  8 0
7 45   15 45  8 0
7 45   15 45  8 0
32 0
Berit Berg
8 0    16 0   8 0
8 0    16 0   8 0
8 0    16 0   8 0
8 0    16 0   8 0
32 0
Charlie Carlsen
8 15   16 0   7 45
8 15   16 0   7 45
8 15   16 0   7 45
8 15   16 0   7 45
31 0
Dag Davidsen
8 0   16 10  8 10
8 0   16 10  8 10
8 0   16 10  8 10
8 0   16 10  8 10
32 40
Ellen Eriksen
7 55   16 15  8 20
7 55   16 15  8 20
7 55   16 15  8 20
7 55   16 15  8 20
33 20
```