Høgskolen i Gjøvik

KONTINUASJONSEKSAMEN

FAGNAVN:	**Grunnleggende datakunnskap og programmering**
FAGNUMMER:	**L 182 A**
EKSAMENSDATO:	**17. august 1999**
KLASSE:	**98HINDA / 98HINDB / 98HINEA 98HDMUA / 98HDMUB / 98HINGA**
TID:	**09.00-13.00**
FAGLÆRER:	**Frode Haug**
ANTALL SIDER UTLEVERT:	**9 (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) Forklar kort (max. 0.5 A4-ark) de mest typiske funksjonene/mulighetene i et mail-program.

a2) Hva står forkortelsen "FTP" for ? Og forklar kort (max. 0.5 A4-ark) hva "FTP" er.

b) Datamaskinen (periferiutstyr / innmat):

b1) Hvilke fire hoveddeler består en datamaskin av ?

b2) Hva står forkortelsen "TSR" for ? Og forklar kort (max. 0.5 A4-ark) hva "TSR" er.

b3) Hva står forkortelsen "FAT" for ? Og forklar kort (max. 0.5 A4-ark) hva "FAT" er.

b4) Nevn noen typiske felles egenskaper (min.5 stk) for disketter og harddisker.

c) Datasikkerhet:

c1) Nevn fem områder for sårbarhet.

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

c3) Nevn fem eksempler på smitteveier for datavirus.

d) Personvern / etikk:

d1) Hvilke typer registre har konsesjonsplikt ?

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

Oppgave 2 (15 %)

a) Hva blir utskriften fra følgende program:

```
#include <iostream>

using namespace std;

const int N   = 10;
const int M   = 5;
const int INC = 2;

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

b) Hva blir utskriften fra følgende program:

```
#include <iostream>

using namespace std;

const int X = 4;
const int Y = 3;

int arr[X][Y] =  { {0, 1, 2}, {1, 2, 3}, {2, 3, 4}, {3, 4, 5} };

int regn(int a[], int n)  {
  int k = 0;
  for (int i = 0;  i < n;  i++)  k += a[i];
  return k;
}

int main()   {
  int i, j;
  cout << regn(arr[0], Y) << '\n';
  cout << regn(arr[2], Y) << '\n';
  cout << regn(arr[3], Y) << '\n';
  for (i = X-1, j = 0;  i > 0;  i--, j++)
      cout << arr[i][j] << "  ";
  cout << '\n';
  return 0;
}
```

Oppgave 3 (65 %)

NB: Les hele teksten for denne oppgaven nøye, før du begynner å besvare noe som helst.

Gjør dine egne forutsetninger dersom du finner oppgaveteksten upresis eller ufullstendig.
Gjør i så fall rede for disse forutsetningene først i besvarelsen din.

Innledning

I denne oppgaven skal du lage et lite program som holder orden på fag (deres numre og navn) og studenter (deres navn og karakter i ulike fag). Under kjøring av programmet vil alle dataene ligge i datamaskinens primærhukommelse. **Studer nøye vedlegg 1, som inneholder masse opplysninger om const'er, struct'er, globale variable, et hovedprogram og noen hjelpefunksjoner ('strip', 'les', 'finn_fag' og 'finn_student').**

Datastrukturen

I vedlegg 1 kan du på den første siden se deklarasjoner/definisjoner av globale const'er, struct'er, arrayer og int'er. Dette skal være alt det du trenger av slike variable for å kunne løse denne eksamensoppgaven.
Datastrukturen vil derfor bestå av to arrayer av struct'er: 'fagene' og 'studentene' og to globale variable som forteller hvor mye av disse arrayene som er tatt i bruk. Ingen av array'ene er sortert på noe kriterie. Legg spesielt merke til at inne i hver student-struct ligger vedkommendes navn, 'ant_resultat' som forteller hvor mange "skuffer" som er tatt i bruk av studentens 'resultat'-array, og at denne arrayen består av struct'er som henviser til et fagnummer og karakteren i dette faget.

Du skal bruke denne datastrukturen, const'er og globale variable gitt i vedlegg 1.
Studer og bruk også de ferdiglagde hjelpefunksjonene: 'strip', 'les', 'finn_fag' og 'finn_student'.

Oppgaven

a) Lag funksjonen "void ny_student()" (Kommandoen 'S')

Dersom det ikke er plass til flere studenter så gis en melding om dette. I motsatt fall så leses aktuell students navn inn. Om en student med dette navnet allerede finnes (duplikate studentnavn får ikke lov til å forekomme), så kommer en melding om dette. Ellers så tas neste "skuff" i bruk av student-arrayen og aktuelle datamedlemmer initialiseres/nullstilles.

Hint: Studer og lær av funksjonen "void nytt_fag()".

b) Lag funksjonen "void ny_karakter()" (Kommandoen 'K')

Les først inn et studentnavn. Om denne ikke finnes, så kommer en melding om dette. I motsatt fall så blir det sjekket om studenten har plass til flere resultater. Om vedkommende ikke har plass, så kommer en melding om dette. Ellers så leses et fagnummer. Finnes ikke dette fagnummeret, så kommer en melding. I motsatt fall få tar man i bruk neste "skuff" av studentens resultat-array, der fagnummeret legges inn sammen med en karakter som skal være i intervallet 1.0-6.0.

NB: Du trenger ikke å sjekke om studenten allerede har en karakter i vedkommende fag !

c) Lag funksjonen "void oversikt()" (Kommandoen 'O')

Les først inn et studentnavn. Om denne ikke finnes, så kommer en melding om dette. I motsatt fall skrives følgende ut om vedkommende student: navn og antall resultater/karakterer på en linje, og en linje pr. fag der vedkommende har fått karakter. Hver slik linje inneholder: fagets nummer, fagets navn og studentens karakter i faget.

d) Lag funksjonen "void les_fra_filer()"

I vedlegg 2 ser du helt konkrete eksempler på hvordan filene "KAR_FAG.DTA" og "KAR_STUD.DTA" kan se ut. I vedlegget er deres format også beskrevet. Les inn alt fra disse to ASCII-filene slik at den (i starten av programmet) tomme datastrukturen fylles med verdiene fra filene.

Klargjørende (?)

- Et fagnummer trenger ikke bare å bestå av tall, men kan være en lett blanding av opptil 10 bokstaver, tall og blanke tegn, f.eks: L 171A, S 142B og F 110A.

Lykke til !

frode@haug.com

Vedlegg 1: Halvferdig programkode (.tpl-fil)

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

using namespace std;

                    //  CONST:
const int FAGNR_LEN     =   11;     //  Max.lengde for et fagnummer (inkl. '\0').
const int NAVN_LEN      =   51;     //  Max.lengde for fag-/studentnavn (inkl.'\0').
const int ANT_FAG       =   31;     //  Max. antall ulike fag EN student kan ta.
const int MAX_FAG       =   61;     //  Max. antall ulike fag på skolen.
const int MAX_STUD      = 501;      //  Max. antall studenter i kartoteket.
                                    //  NB: Bruker ikke element nr.0 for de tre sistnevnte !

                    //  STRUCT:
struct fag {
  char nr[FAGNR_LEN];           //  Fagets unike nummer (bokstaver og tall).
  char navn[NAVN_LEN];          //  Fagets navn.
};

struct fagdata  {
  char  nr[FAGNR_LEN];          //  Fagnummer.
  float karakter;               //  Karakteren i dette faget.
};

struct student  {
  char    navn[NAVN_LEN];       //  Studentens unike navn.
  int     ant_resultat;         //  Antall resultater registrert på studenten.
  fagdata resultat[ANT_FAG];    //  Studentens resultater (fagnummer og karakter).
};

                    //  DEKLARASJON AV FUNKSJONER:
void strip(char txt[]);
void les(char txt1[], char txt2[], const int N);
int  finn_fag(char nr[]);
int  finn_student(char nvn[]);
void skriv_meny();
void nytt_fag();
void ny_student();              //  Oppgave 3a
void ny_karakter();             //  Oppgave 3b
void oversikt();                //  Oppgave 3c
void les_fra_filer();           //  Oppgave 3d

                    //  GLOBALE VARIABLE:
fag     fagene[MAX_FAG];        //  Array av fag-struct'er
student studentene[MAX_STUD];   //  Array av student-struct'er
int     siste_fag,  siste_student;  //  Siste brukte indeks i de to arrayene ovenfor.
```

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

  les_fra_filer();                          //  Oppgave 3d
  skriv_meny();
  cout << "\n\nKommando:  ";   cin >> kommando;   cin.ignore();
  kommando = toupper(kommando);

  while (kommando != 'Q')  {
    switch (kommando)  {
      case 'F': nytt_fag();         break;
      case 'S': ny_student();       break;      //  Oppgave 3a
      case 'K': ny_karakter();      break;      //  Oppgave 3b
      case 'O': oversikt();         break;      //  Oppgave 3c
      default:  skriv_meny();       break;
    }
    cout << "\n\nKommando:  ";   cin >> kommando;  cin.ignore();
    kommando = toupper(kommando);
}
//   skriv_til_filer();                   //  IKKE en del av eksamensoppgaven å lage denne.
  return 0;
}

                    //  DEFINISJON AV FUNKSJONER:
void strip(char txt[])  {               //  Tar vekk alle overflødige ' ' (blanke) på slutten av en tekst.
  while (txt[strlen(txt)-1] == ' ')     txt[strlen(txt)-1] = '\0';
}


void les(char txt1[], char txt2[], const int N)  {      // Ber om og leser en tekst.
  do {
    cout << "\n\t" << txt1 << ": ";                     //  Skriver ut ledetekst/prompt.
    cin.getline(txt2, N);                              //  Leser tekst fra brukeren.
  } while (strlen(txt2) == 0);                         //  Forkaster så lenge kun ENTER/CR.
}


int finn_fag(char nr[])  {                             //  Returnerer indeksen for et gitt fagnummer:
  for (int i = 1;  i <= siste_fag;  i++)
      if (!strcmp(fagene[i].nr, nr))  return i;        // Funnet !
  return 0;                                            //  Ingen treff.
}

int finn_student(char nvn[])  {                        //  Returnerer indeksen for en gitt student:
  for (int i = 1;  i <= siste_student;  i++)
      if (!strcmp(studentene[i].navn, nvn))  return i; // Funnet !
  return 0;                                            //  Ingen treff.
}
```

```
void skriv_meny()  {                        //  Skriver brukerens lovlige valg:
  cout << "\n\nFølgende kommandoer er tilgjengelige:";
  cout << "\n\tF - nytt Fag";
  cout << "\n\tS - ny Student";
  cout << "\n\tK - ny Karakter hos en student";
  cout << "\n\tO - Oversikt over EN students karakterer";
  cout << "\n\tQ - Quit / avslutt";
}


void nytt_fag()  {                          //  Registerer et nytt fag/kurs:
  char num[FAGNR_LEN];                      //  Nytt unikt fagnummer.

  if (siste_fag < MAX_FAG-1)  {             //  Plass til flere fag:
     les("Nytt fagnummer", num, FAGNR_LEN);     //  Leser inn nytt fagnummer.
     if (!finn_fag(num))  {                 //  Faget eksisterer IKKE allerede:
        siste_fag++;                        //  Tar i bruk neste "skuff" i 'fagene'-arrayen.
        strcpy(fagene[siste_fag].nr, num);      //  Kopierer inn nytt fagnummer.
        les("Nytt fagnavn  ", fagene[siste_fag].navn, NAVN_LEN);     //  Leser inn navnet til det nye
faget:
     } else                                 //  Faget finnes allerede:
       cout << "\n\tDette fagnummeret eksisterer allerede !\n";
  } else                                    //  IKKE plass til flere fag:
    cout << "\n\tFULLT med " << MAX_FAG-1 << " fag allerede !";
}


void ny_student()  {            //  OPPGAVE 3a:
        //  Lag innmaten
}


void ny_karakter()  {           //  OPPGAVE 3b:
        //  Lag innmaten
}


void oversikt()  {              //  OPPGAVE 3c:
        //  Lag innmaten
}


void les_fra_filer()  {         //  OPPGAVE 3d:
        //  Lag innmaten
}
```

Vedlegg 2: Eksempler på filers innhold

KAR_FAG.DTA:

3
L 182A Grunnleggende datakunnskap og programmering
L 183A Objekt-orientert programmering
L 171A Algoritmiske metoder I

Format:

< Antall fag/linjer på filen >
< Fagnummer 1 > < Fagnavn 1 >
< Fagnummer 2 > < Fagnavn 2 >
……..
……..
< Fagnummer N > < Fagnavn N >

NB: Fagnummer starter alltid i posisjon nr.1 på linja, mens fagnavn starter i posisjon nr.11.

KAR_STUD.DTA:

3
Arne Andersen
2
L 182A 2.2
L 183A 2.7
Bjarne Berg
1
L 182A 3.7
Charlotte Carlsen
3
L 182A 1.6
L 183A 1.9
L 171A 2.2

Format:

< Antall studenter på filen >
< Studentnavn 1 >
< Antall karakterer for student 1 >
< Fagnummer 1 for student 1 > <Karakter 1 for student 1 >
< Fagnummer 2 for student 1 > <Karakter 2 for student 1 >
…….
…….
< Fagnummer M for student 1 > <Karakter M for student 1 >
……..……
……..……