**Høgskolen i Gjøvik**
**Avdeling for teknologi**

# KONTINUASJONSEKSAMEN

**FAGNAVN:** **Grunnleggende programmering**

**FAGNUMMER:** **IMT 1031**

**EKSAMENSDATO:** **9. august 2005**

**KLASSE(R):** **04HBIND*, 04HBINE*, 04HBINFA, 04HBMETEA, 04HBGEOA**

**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[] = "INNI KORNSILOEN HANG EN BJØRK TIL TØRK";

int main()  {
    int i = 17;

    while (i <= 26)  {
       i += 9;
       cout << txt[i] << txt[i+1] << txt[i+2] << '\n';
    }
    for (i = 6;  i <= 312;  i*=i)
        cout << ((txt[i] > 'Q') ? "<" : ">") << '\n';

    i = 9;
    cout << txt[i * 7 / 3 % 6] << '\n';
    return 0;
}
```

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

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

class A  {
  private:
    char txt[20];
    int  tall;
  public:
    A()                 {  strcpy(txt, "BANAN-MATHIESEN");  tall = 44;   }
    A(char t[], int ta) {  strcpy(txt, t);  tall = ta;  skriv();  }
    void skriv()        {  cout << txt << '\t' << tall  << '\n';  }
    bool funk1(char c)  {  return (txt[4] == char(int(c)+2));     }
    char funk2()        {  return (txt[6]);              }
    char funk3(int n)   {  return (char(int(txt[2])+n));  }
};

int main()  {
    A a1, a2("MAKRELLSTIMEN", 45);
    a1.skriv();
    cout << a1.funk1('L') << '\n';
    cout << a1.funk1(a2.funk2()) << '\n';
    cout << a2.funk3(3) << '\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, klassene og deres medlemsfunksjoner, globale variable og funksjonen** `int les(char t[], int min, int max)`**. Husk å bruke denne funksjonen *aktivt,* dvs. *alle* steder der det er aktuelt med innlesning av et tall.**

I denne oppgaven skal du lage et lite program som holder orden på en masse varer og leverandørene (grossistene/distributørene/importørene/forhandlerne) av disse varene. Dvs. til hvilken leverandør man må henvende seg for å etterbestille/påfylle noe av en gitt vare.

## Datastrukturen

En vare er representert vha. klassen 'Vare'. Det finnes MAXVARE slike objekter, representert vha. arrayen 'varer'. En leverandør er representert vha. klassen 'Leverandor'. Det finnes MAXLEV slike objekter, representert vha. arrayen 'leverandorer'. I begge disse er indeksene f.o.m. 1 t.o.m. henholdsvis 'siste_vare' og 'siste_leverandor' i bruk. Klassen 'Vare' inneholder bl.a. datamedlemmet 'levnr'. Dette angir indeksen i 'leverandorer' for hvem som er leverandør av den aktuelle varen.
På den første siden vedlegget kan du se deklarasjoner/definisjoner av const'er, de to klassene og globale variable. *Dette skal være alt du trenger av klasser, datamedlemmer og globale variable for å løse denne eksamensoppgaven!*

## Oppgaven

### a)

**Lag funksjonene** `void ny_leverandor()` og `void Leverandor::les_inn()`

Er det *ikke* plass til flere leverandører kommer det en melding om det. I motsatt fall tas neste indeks i 'leverandorer'-arrayen i bruk, og vedkommende objekt leser selv inn *alle* sine data.

### b)

**Lag funksjonene** `void skriv_vare()` og `void Vare::skriv(int n)`

Først spørres det etter et aktuelt nummer for en vare. Deretter skrives *alle* data om denne varen ut til skjermen, inkludert *alle* data om den aktuelle leverandøren.

### c)

**Lag funksjonene** `void ny_vare()` og `void Vare::les_inn()`

Er det *ikke* plass til flere varer kommer det en melding om det. I motsatt fall tas neste indeks i 'varer'-arrayen i bruk, og vedkommende objekt leser selv inn *alle* sine data. Pass på at alle de tallene som leses inn er lovlige/fornuftige/rimelige. Dette gjøres ved aktiv bruk av `les(……)` med de rette parametrene. 'bestilt' settes til 'false'.

**d)** **Lag funksjonene** `void selg_vare() , void Vare::selg()`
`void Vare::bestilling() og void Leverandor::skriv_til_fil(…)`

Den første funksjonen spør om aktuelt varenummer og kaller deretter varens `selg()`. Denne funksjonen igjen spør om antallet som skal selges, og reduserer 'ant_igjen' med dette tallet. *Om* 'ant_igjen' <= 'grense' og det ennå ikke er etterbestilt ('bestilt' er 'false'), så settes 'bestilt' til 'true', og `bestilling()`kalles. Denne funksjonen igjen skal skrive en etterbestilling av den aktuelle varen *bakerst* (appende) på filen "BESTILL.DTA". Dette gjør den ved å fortelle brukeren om hvor mange som er igjen og om bestillingsgrensen. Den spør så om hvor mange brukeren ønsker å bestille, og skriver dette antallet og varens navn til filen. Alle data om den aktuelle leverandøren blir også skrevet til filen. Dette utføres av `skriv_til_fil()`.
**NB:** Oppgave 2d teller *det dobbelte* av de andre deloppgavene i oppgave 2.

**e)** **Lag funksjonene** `void bestilling_ankommet()` og `void Vare::paafyll()`

Den første funksjonen spør om aktuelt varenummer og kaller deretter varens `paafyll()`. Denne funksjonen igjen spør om antallet som har kommet inn, og øker 'ant_igjen' med dette tallet. 'bestilt' oppdateres. *Om* 'ant_igjen' fortsatt er <= 'grense', så oppdateres 'bestilt' igjen og `bestilling()` (som du laget(?) ifm. oppgave 2d) kalles.

**f)** **Lag funksjonene** `void les_fra_fil(), void Vare::les_fra_fil (……)` og
`void Leverandor::les_fra_fil (……)`

Den første funksjonen sørger for at *alle* data fra filene "VARER.DTA" og "LEVERAND.DTA" blir lest inn i datastrukturen. Til dette bruker den de to andre funksjonene angitt ovenfor.
Filene har følgende format:

VARER.DTA: <Leverandørnr> <Navn>
<Antall igjen> <Bestillingsgrense> <0 eller 1>
der <0 eller 1> angir om varen allerede er under etterbestilling eller ei.

LEVERAND.DTA: <Telefonnr> < Navn>
<Adresse>

Husk å oppdatere 'siste_.....'-variablene!

# Klargjøring og forutsetninger

- Noen begrensninger ved programmet er:
  - Det er *ikke* mulig å slette/endre noen varer/leverandører etter at de er registrert.
  - Datastrukturen blir *ikke* skrevet til fil (`skriv_til_fil()` skal *ikke* lages).
  - Brukeren må stort sett huske alle slags vare- og leverandørnumre i hodet.
- Du trenger ***ikke*** å sjekke på duplikate vare-/leverandørnavn.
- Gjør dine egne forutsetninger dersom du finner oppgaveteksten upresis eller ufullstendig. Gjør i så fall rede for disse forutsetningene *først* i besvarelsen din.

**Eksamensvaren er levert! Og lykke til !**
**frode@haugianerne.no**

# Vedlegg: Halvferdig programkode (.tpl-fil)

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

                        // CONST:
const int MAXVARE   = 500;      // Max. antall varer i datastrukturen.
const int MAXLEV    = 200;      // Max. antall leverandører i datastrukturen.
const int MAXANT  = 10000;      // Max. antall av en vare.
const int STRLEN     =  40;     // Max. lengde på tekststrenger.

                        // KLASSER:
class Vare  {                   // Vare:
  private:
        char navn[STRLEN];      // Navn.
        int  levnr,             // Leverandørens nummer/indeks.
             ant_igjen,         // Antall igjen på lager.
             grense;            // Etterbestillingsgrense.
        bool bestilt;           // Har bestilt eller ei.
  public:
        void skriv(int n);      // Deklarasjon av medlemsfunksjoner:
        void les_inn();
        void selg();
        void bestilling();
        void paafyll();
        void les_fra_fil(istream* inn, int nr);
};


class Leverandor  {             // Leverandør:
  private:
        char navn[STRLEN];      // Navn.
        char adr[STRLEN];       // Adresse.
        int tlf;                // Telefon.
  public:
        void skriv(int n);      // Deklarasjon av medlemsfunksjoner:
        void les_inn();
        void skriv_til_fil(fstream* ut);
        void les_fra_fil(istream* inn, int nr);
};


                        // DEKLARASJON AV FUNKSJONER:
void skriv_meny();                      void skriv_leverandorer();
char les_kommando();                    void ny_vare();
int  les(char t[], int min, int max);   void selg_vare();
void ny_leverandor();                   void bestilling_ankommet();
void skriv_vare();                      void les_fra_fil();
```

```
                        //  GLOBALE VARIABLE:
int siste_vare;                             //  Indeksen for siste brukte vare.
int siste_leverandor;                       //  Indeksen for siste brukte leverandør.
Vare varer[MAXVARE+1];                      //  Array med Vare-objekter.
Leverandor leverandorer[MAXLEV+1];          //  Array med Leverandor-objekter.


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

  les_fra_fil();                                    //  Oppgave 2f
  skriv_meny();
  kommando = les_kommando();
  while (kommando != 'Q')  {
     switch (kommando)  {
         case 'L':  ny_leverandor();            break;  //  Oppgave 2a
         case 'V':  ny_vare();                  break;  //  Oppgave 2c
         case 'X':  skriv_vare();               break;  //  Oppgave 2b
         case 'Y':  skriv_leverandorer();       break;
         case 'S':  selg_vare();                break;  //  Oppgave 2d
         case 'B':  bestilling_ankommet();      break;  //  Oppgave 2e
         default:   skriv_meny();               break;
    }
    kommando = les_kommando();
  }
 cout << "\n\n";
 return 0;
}

                        //  DEFINISJON AV KLASSE-FUNKSJONER:
void Vare::skriv(int n)  {      //  Skriver ALLE data:
        //  Oppgave 2B  Lag innmaten       }

void Vare::les_inn()  {         //  Leser inn ALLE data:
        //  Oppgave 2C:  Lag innmaten      }

void Vare::selg()  {            //  Et antall av varen selges:
        //  Oppgave 2D:  Lag innmaten      }

void Vare::bestilling()  {      //  Etterbestilling skrives til fil:
        //  Oppgave 2D:  Lag innmaten      }

void Vare:: paafyll()  {        //  Påfylling/bestilling ankommet:
        //  Oppgave 2E:  Lag innmaten      }

void Vare::les_fra_fil(istream* inn, int nr)  {             //  Leser inn ALLE data fra fil:
        //  Oppgave 2F:  Lag innmaten      }

void Leverandor::skriv(int n)  {        //  EKSTRA - ferdiglaget
 cout << '\t' << n << ": " << navn << '\t' << adr << '\t' << tlf << '\n';        }

void Leverandor::les_inn()  {           //  Leser inn ALLE data:
        //  Oppgave 2A:  Lag innmaten      }
```

```
void Leverandor::skriv_til_fil(fstream* ut) {          //  Data ifm. etterbestilling til fil:
        //  Oppgave 2D:  Lag innmaten      }

void Leverandor::les_fra_fil(istream* inn, int nr)  { //  Leser inn ALLE data fra fil:
        //  Oppgave 2F:  Lag innmaten      }


                        //  DEFINISJON AV FUNKSJONER:
void skriv_meny()  {            //  Presenterer lovlige menyvalg:
  cout << "\n\nFØLGENDE KOMMANDOER ER LOVLIG:\n";  cout << "\tL = ny Leverandør\n";
  cout << "\tV = ny Vare\n";                       cout << "\tX = skriv vare\n";
  cout << "\tY = skriv leverandører\n";            cout << "\tS = Selg vare\n";
  cout << "\tB = Bestilling/påfylling ankommet fra leverandør\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(char t[], int min, int max)  {    //  Leser et tall i et visst intervall.
  int n;
  do  {                                   //  Skriver ledetekst:
    cout << '\t' << t << " (" << min << '-' << max << "): ";    cin >> n;        //  Leser inn ett tall.
  } while(n < min  ||  n > max);          //  Sjekker at i lovlig intervall.
  cin.ignore();                           //  Forkaster ett tegn ('\n').
  return n;                               //  Returnerer innlest tall.
}

void ny_leverandor()  {                   //  Registrerer en ny leverandør:
        //  Oppgave 2A:  Lag innmaten      }

void skriv_vare()  {                      //  Skriver ALLE data om EN vare:
        //  Oppgave 2B:  Lag innmaten      }

void skriv_leverandorer()  {              //  EKSTRA – ferdiglaget
   cout << "\nOVERSIKT OVER ALLE LEVERANDØRENE:\n";
  for (int i = 1;  i <= siste_leverandor;  i++)
        leverandorer[i].skriv(i);                 }

void ny_vare()  {                         //  Registrerer en ny vare:
        //  Oppgave 2C:  Lag innmaten      }

void selg_vare()  {                       //  Selg et antall av en vare:
        //  Oppgave 2D:  Lag innmaten      }

void bestilling_ankommet()  {             //  Påfyll av en vare:
        //  Oppgave 2E:  Lag innmaten      }

void les_fra_fil()  {                     //  Leser HELE datastrukturen fra filer:
        //  Oppgave 2F:  Lag innmaten      }
```