SJØKRIGSSKOLEN

Fredag 16. mai 2014

# Eksamen våren 2014

# TD 2050

**Datasystemer – skriftlig**

## **Klasse M-ED2, I-ED2**

TILLATT TID: 5 timer

HJELPEMIDLER: Lærebok Programmering i C++

**Svar på emnet «Operativsystemer med mer» og emnet «Datamodellering» oppgave 2a) og 3 skal leveres på ark.**

**Svar på emnet «Datamodellering» oppgave 1 , 2 b) og 2c) skal leveres på USB-penn.**

Oppgavetekst og testprogram ligger på USB-pinne merka Kand. Nr. Før på ditt kandidatnummer på pinnen. Dette er også innleveringspinne. Start med å kopiere alle filer fra denne pinnen inn i katalogen **C:\Data**. Du må eventuelt lage katalogen først. **Bruk denne katalogen som arbeidskatalog for programmeringsoppgavene under eksamen**.

Start Microsoft Visual Studio 2010 og lag et C++-prosjekt **PROVE** **som skal ligge på C:\Data\**. Alle program skal legges inn i dette prosjektet for kompilering og kjøring.

Lag en innleveringskatalog på C:\Data med kandidatnummeret ditt som katalognavn. **Kopier svara på hver oppgave inn i denne katalogen** . Når eksamen er ferdig, kopierer du denne katalogen over på USB-pinnen som skal leverest inn. Sjekk før innlevering at alle filer er på plass på pennen og at de er større end 0 kb.

**Innleveringspinnen** skal inneholde disse filene (i katalogen med navn lik kandidatnummeret ditt) :

**Posisjonsklasse.cpp, TestFrielager.cpp, Kadettklasse.cpp, TestKadettklasse.cpp**

Skriv kandidatnummeret ditt som kommentar øverst i alle program.

Etter eksamen for alle er ferdig, kan du oppsøke PC-en du satt på og ta kopi av svara dine med som "Elevens kopi". Lat filene ligge på C:\Data som skolens kontroll-kopi.

ANTALL OPPGAVER: 7 Eksamenssettet er på 7 sider.

Dagfinn Andreassen Lars Olav Tveita Jan M Johannessen

faglærer faglærer sensor

**Emne: Datamodellering.**

Denne delen tel 65% av eksamen. Den består av 3 oppgåver. På kvar oppgåve er det angitt kor mange % av heile eksamenssettet den oppgåva tel.

**Oppgåve 1 (25 %)**

Når vi skal angi posisjon i navigasjon, oppgir vi grader og minutt slik:

58°, 1.530' N 10°, 53.900' Ø.

Denne posisjonen kan vi også gi som desimaltal slik:

58.0255° N 10.8983° Ø

Vi skal bruke eit C++ program til å angi posisjonar og finne kompasskursar innafor eit **lite** kartutsnitt der vi kan rekne lengdesirklane(meridianane) som parallelle rette linjer i kartet og breiddesirklane(parallellsirklane) som parallelle rette linjer i kartet. Vi får eit rettvinkla koordinatsystem der x-aksen angir lengdegrader(Ø) og y-aksen angir breiddegrader(N).

For å finne kursen frå eitt punkt (x1,y1) til eit anna punkt (x2,y2) i dette kartet, kan vi bruke denne formelen:



cos(y1) der y1 er i radianar, kompenserer for at avstanden mellom lengdesirklane er mindre enn avstanden mellom breiddesirklane (untatt ved ekvator).

a)

Definer ein klasse **Posisjon** med private data graderN, minuttN, graderO og minuttO og public sett- og finnfunksjonar. Klassen skal også ha funksjonar for omrekning frå (grader,minutt) til (grader som desimaltal) og ein funksjon for å finne kursen frå ein posisjon til ein annan. Klassen skal ha ein konstruktør utan argument. Alle funksjonane skal implementerast. Lagre klassen på fila **Posisjonsklasse.cpp** på USB-pinnen.

Bruk dette programmet **Testposisjon.cpp** som ligg på USB-pinnen til å teste ut klassen:

#include "Posisjonsklasse.cpp"

#include <iostream>

void main(void)

{

Posisjon A,B;

A.SettPosisjon(60,23.670,5,15.950);

//Sjøkrigsskolens posisjon er 60 grader,23.670 minutt Nord,5 grader,15.950 minutt Øst

cout<<"Posisjonen kan også skrivast slik: ";

cout<<A.DesimalverdiBreidde()<<" grader N ,"<<A.DesimalverdiLengde()<<" grader Ø"<<endl;

B.SettPosisjon(60,24.670,5,16.950);

//1 minutt lenger Nord og 1 minutt lenger Øst enn A

cout<<"Kurs frå A til B: "<<A.KursTil(B)<<endl;

}

b)

Skriv om testprogrammet slik at posisjonane A og B blir lagra i det frie lageret. Lagre programmet som **TestFrielager.cpp** på USB-pinnen.

**Oppgåve 2 (30 %)**

1. Vi skal lage eit C++ -program for administrering av kadettar. Lag ein UML-modell med følgjande entitetstypar:

**Avdeling** med private data Avdelingsnummer

**Offiser** med private data Ansattnr

**Kadett** med private data Engelskkarakter (fellesfag)

**Operativkadett** med private data Navkarakter

**EDkadett** med private data Elektrokarakter

Sett namn på relasjonstypane, spesifiser multiplisitet og om relasjonar er total/delvis, ovelappande/disjunkt.

UML-modellen skal teiknast/skrivast på papir som skal leverast inn.

Lag eit C++ program der du definerer klassar som er i samsvar med UML-modellen:

1. Definer klassane Avdeling, Offiser, Kadett og EDkadett med konstruktørar utan argument, og med offentlege sett- og finnfunksjonar for dei relevante private dataene som er spesifisert i oppgåve a). Klassen Offiser skal ha ein ekte virtuell utskriftsfunksjon for alle data og ein protected funksjon for utskrift av fellesdata. Implementer **utskriftsfunksjonane**. Det blir ikkje kravd at sett- og finn-funksjonane og konstruktørane skal implementerast. Lagre klassen på fila **Kadettklasse.cpp** på USB-pinnen.

c) Lag ein C++ -applikasjon (hovedprogram main()) som definerer ein EDkadett med elektrokarakter B, engelskkarakter C, ansattnr 12345 og avdelingsnummer 4752.

Applikasjonen skal også gi utskrift av alle dataene for denne kadetten. Lagre programmet på fila **TestKadettklasse.cpp** på USB-pinnen. Programmet skal kunne kompilerast utan feil.

Sidan alle funksjonane ikkje er implementert, blir det ikkje kravd at programmet er køyrbart.

(Du skal kunne ta Build/Compile utan feilmelding, men det blir ikkje kravd at du kan ta Build/Build utan feilmelding)

**Oppgåve 3 (10 %)**

Svara på denne oppgåva skal leverast på papir.

Under etablering av databasen FUBASE til hjelp for administrering av eksamen for fjernundervisningskurs, er eit første framlegg til datatabell slik (med ein del eksempeldata innlagt):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| studnr | etternavn | postnr | poststed | fagkode | fagnavn | karakter |
| 101 | Hansen | 7563 | MALVIK | LO323D | DATABASER | 2,5 |
| 101 | Hansen | 7563 | MALVIK | LO343D | GRUNNKURS EDB | 2,1 |
| 102 | Jensen | 9900 | LAKSELV | LO248D | DATAKOMMUNIKASJON | 1,8 |
| 102 | Jensen | 9900 | LAKSELV | LO323D | DATABASER | 2,8 |
| 102 | Jensen | 9900 | LAKSELV | LO370D | WEBTEKNIKKER | 1,5 |
| 103 | Tormodsen | 7223 | MELHUS | LO323D | DATABASER | 1,3 |

Analyser denne tabellen ved å skissere determineringar og kandidatnøklar og bestemme normalform og primærnøkkel. Konstruer så normaliserte tabellar for dataene. Tabellane skal ikkje fyllast ut med data.

Lag ei SQL-spørjing frå dei nye tabellane som kan gi tabellen ovanfor som utskrift.