UNIVERSITETET I AGDER

FAKULTET FOR TEKNOLOGI OG REALFAG

P R Ø V E E K S A M E N

Emnekode: **MAS234**
Emnenavn: **Innebygde datasystemer for mekatronikk**

Dato: MAS234 PRØVEEKSAMEN 2017
Varighet: Eksamen har 4 timers varighet, dette prøvesettet inneholder færre oppgaver enn en faktisk eksamen.

Antall sider inkl. forside: -

Tillatte hjelpemidler: Alle trykte og håndskrevne hjelpemidler tillatt. Kalkulator tillatt.

Merknader: Kandidaten må selv kontrollere at oppgavesettet er fullstendig. Les nøye igjennom oppgavene slik at du forstår hva det spørres etter. Beskriv eventuelle antagelser du må gjøre dersom oppgaven er formulert uklart. Lesing av spesifikasjoner, herunder oppgavetekster, er en del av det dere blir testet i. All kildekode i oppgaveteksten er gitt i C++11. Der det spørres om implementasjon, skal det programmeres på papir. Det kan antas at header <iostream> er inkludert fra før, alle andre nødvendige includes må spesifiseres i løsningen.

Oppgave 1

Gitt følgende kodesnutt:

```
const int x = 3;
const int y = 2;
int z = 42;

const int a = x*y;
int& b = z;
b++;

std::cout << "a = " << a << ", b = " << b << std::endl;
```

a) Hva skrives ut?

a = 6, b = 43

b) Kan z være const i denne programsnutten? Hvorfor / hvorfor ikke?

Nei, z kan ikke deklareres const. Hvis z er const, så må også referansen til z (altså variabelen b) være const, noe som gjør at vi ikke kan inkrementere b slik det gjøres i kodesnutten. (Inkrementering innebærer å skrive til variabelen.)

Oppgave 2

Begrepsforståelse.

a) Beskriv kort hva en ”toolchain” er i forbindelse med krysskompilering. Nevn to av de viktigste komponentene i en toolchain.

En toolchain for krysskompilering er en samling verktøy som gjør det mulig å lage kjørbare kode (”programmer”) for en maskin som har en annen arkitektur enn den platformen hvor krysskompilatoren kjører. Toolchain-en kan også inneholde f.eks. debuggingsverktøy, ferdigkompilerte bibliotek for target-plattform etc..

Et eksempel kan være krysskompilering og linking av programvare for en Rapsberry Pi 3 med ARM-prosessor på en host-maskin som kjører f.eks. Ubuntu Linux på en 64 bit intel x86-plattform.

Et annet eksempel kan være krysskompilering for en AVR mikrokontroller på f.eks. en host-maskin som kjører Windows eller MacOS på en intel x86-plattform.

De viktigste verktøyene i en generell toolchain er kompilator og linker, mens det ved krysskompilering blir mer korrekt å si krysskompilator og linker.

b) Vi har krysskompilert programmet vårt, og har fått ut en kjørbar fil. Vil denne kjørbare filen generelt sett kunne kjøres på host-maskinen? Hvorfor / hvorfor ikke?

Nei, i dette vil ikke være mulig i det generelle tilfellet. Når det er snakk om krysskompilering, så kompileres programmet for en target-plattform med et annet instruksjonssett enn det vi finner på host-plattformen. Binærkoden/maskinkoden vil dermed ikke vil kjøre korrekt (eller kjøre i det hele tatt) på host-plattformen (hvor vi utførte krysskompileringen).

Dette er riktignok en sannhet med modifikasjoner, da vi i mange tilfeller kan lage en virtuell maskin på host-maskinen (f.eks. ved hjelp av VirtualBox eller qemu), slik at vi faktisk kan kjøre programmer for f.eks. ARM-prosessorer på en maskin med ”tradisjonell” intel x86 eller AMD64 CPU. I dette tilfellet kjøres instruksjonene i det krysskompilerte programmet vårt på den virtuelle CPU-en, som oversetter dem til instruksjoner for den fysiske CPU-en på host-maskinen.

Oppgave 3

Arrays. Gitt følgende funksjon ”arrayFun”:

```
void arrayFun()
{
    int magNumbers[] = {1, 3, 7, 11, 24, 42};

    // a)
    std::cout << "a = " << magNumbers[2] << std::endl;

    // b)
    std::cout << "b = " << *magNumbers << std::endl;
}
```

a) Hva skrives ut av linjen etter kommentar a) ?
7 (med linjeskift etter hvis vi skal være pirkete..)

b) Hva skrives ut av linjen etter kommentar b) ?
1 (dette blir det samme som å skrive ut magNumbers[0]

Oppgave 4

Funksjoner.

a) Skriv deklarasjonen til en funksjon som tar inn to heltall av samme datatype og returnerer en bool. Funksjonen skal hete firstIsLargest. Velg datatype selv.

```
bool firstIsLargest(int first, int second);
```

b) Lag implementasjonen til funksjonen fra deloppgave a). Funksjonen skal returnere true hvis den første parameteren er størst, false hvis ikke.

```
bool firstIsLargest(int first, int second)
{
    return first > second;
}
```

Oppgave 5

Bruk av SSH for fjerntilgang til innebygget datasystem.

Ved hjelp av en SSH-klient kan vi logge oss på en datamaskin som kjører eksempelvis Dropbear eller OpenSSH-server over et IP-nettverk. Dropbear ble benyttet under obligatorisk lab i MAS234.

a) Hva bør sjekkes dersom følgende feilmelding dukker opp ved forsøk på tilkobling med SSH? Nevn de to-tre første sjekkpunktene dine i et slikt tilfelle.

   ssh: connect to host 10.0.0.3 port 22: Network is unreachable

   Her er det viktige å vise frem "feilsøkingsferdigheter". Det er ikke et låst sett med punkter som er korrekt, men noen gode punkter er:

   Sjekke at power på target er på og at operativsysemet har startet uten feil. Sjekke at nettverkskabel er koblet til. Sjekke at korrekte IP-adresser er satt (og at andre nettverksinnstillinger er OK på begge maskiner). Forsøke å pinge ip-adressen 10.0.0.3 fra host-maskinen. Sjekke at dropbear eller annen SSH-server er installert på target-maskinen, og at denne kjører og lytter på port 22.

b) Hvilke krav vil du sette til et passord for bruk til SSH-innlogging på et innebygget datasystem over internett? La oss si at maskinen styrer lastesystemet på et skip.

   Se gode råd her for tips: https://www.nsm.stat.no/blogg/passordrad/

   Og mer konkret og kortere "liste" her: https://www.stopthinkconnect.org/resources/preview/tip-sheet-passwords-and-securing-your-accounts

c) Hvis du ofte må logge inn på maskinen med SSH; hva kan gjøres for å få ned "arbeidsmengden" ved å taste et langt og sikkert passord – samtidig som graden av sikkerhet bevares eller økes? Svaret skal begrunnes.

   Bruk av public key authentication er brukervennlig samtidig som det kan gi høg grad av sikkerhet. Se generelle tips her: https://help.ubuntu.com/community/SSH/OpenSSH/Keys

   I praksis må man generere et public/private nøkkelpar (slik vi gjorde for bruk av

Github tidlig i øvingsopplegget), og plassere public-nøkkelen i authorized keys på maskinen som kjører ssh-server.

Oppgave 6

Løkker.

Følgende kodesnutt printer tall til standard out (vanligvis terminalvinduet).

```
const uint32_t n = 8;

for (uint8_t i = 5; i < n; ++i)
{
    std::cout << +i << std::endl;
}
```

a) Hva er første og siste tall som skrives ut?
   5 først 7 sist.
b) Hvor mange tall skrives ut, og skrives de ut etter hverandre eller under hverandre?
   3, under hverandre.
c) Hva gjør + foran i?
   + promoterer unsigned-variabelen til et tall som kan printes. Alternativet, som på mange måter er ”ryddigere”, er å benytte static_cast<int>(i)

Oppgave 7

Bruk av const.

a) Hva betyr det dersom vi deklarerer en variabel const?
Kompilatoren passer på at vi ikke skriver til variabelen etter at den er opprettet.

b) Gi et kort eksempel (maksimalt åtte linjer) med bruk av const.

Her benyttes const for å sikre at det ikke utilsiktet skrives til innparameteren r eller variabelen pi (pi må naturlig nok kun beregnes én gang, da det er en konstant).

```
#include <cmath>

const double pi = std::acos(-1);

void computeAreaOfCircle(const double r, double& area)
{
   area = pi * r * r;
}
```

c) Kompilerer det følgende programmet, hvis ikke – hvorfor?

```
#include <iostream>

int main()
{
    const double dt = 1.0e-6; // One microsecond sampling time
    const double acceleration_z = 9.81; // m/(s*s)

    int numberOfTimesteps = 100000;

    velocity = acceleration_z * dt * static_cast<double>(numberOfTimesteps);

    std::cout << "Velocity after " << numberOfTimesteps << " time steps";
    std::cout << " w/constant accl: " << velocity << " [m/s]" << std::endl;
}
```

Nei programmet kompilerer ikke, da variabelen velocity ikke er deklarert. C++ er et strengt typet språk, så alle variabler må deklareres eksplisitt med type.

Litt tilleggsinfo: I C++11 og nyere kunne vi benyttet ”auto” for å få typen automatisk dedusert/avledet fra initialiseringsverdien (altså det på høyre side av assignment i dette tilfellet). Auto har ikke vært dekket i forelesningene, og er følgelig ikke pensum.
http://en.cppreference.com/w/cpp/language/auto

d) Gi to eksempler hvor det med fordel kan benyttes const bør benytte const i et C++-program, og hva const gjør i disse tilfellene.

Tre eksempler:

- Når vi deklarerer variabler som det ikke senere skal skrives til -- som i oppgave 7 b).
- Når vi skal ta inn parametre til en funksjon – og ikke har til hensikt å skrive til disse. Dette er spesielt viktig når vi sender inn referanser, da utilsiktet skriving til en referanse vi får inn som en parameter til en funksjon vil ha effekt utenfor funksjonen. Kompilatoren vil nekte oss å sende inn en referanse til en const-variabel hvis parameteren ikke er deklarert const.
- Når vi deklarerer funksjoner tilhørende klasser kan disse deklareres const. Dette forteller kompilatoren at vi ikke har tilhensikt å skrive til medlemsvariablene i klassen, og all skriving til medlemsvariable vil dermed gi kompileringsfeil.

```
class ProportionalController
{

public:

    // const her betyr at vi "lover" å ikke skrive til
    // medlemsvariablene (f.eks. p_) når vi implementerer
    // getPValue()-funksjonen. Kompilatoren passer da på
    // at vi ikke gjør dette ved en feil.
    void getPValue() const;

private:

    double p_;
};
```

Oppgave 8

Structs.

a) Skriv deklarasjonen til en struct med navn "Fluid" som inneholder tre felt: id, pricePerLitre og litre. Velg selv datatype på feltene.

```
struct Fluid
{
    int id;
    float pricePerLitre; // Bør ha flyttall-type.
    float litre;         // Bør ha flyttall-type.
};
```

b) Lag en funksjon som tar inn en array med struct-typen du definerte i a) og summerer prisen for alle elementene. Funksjonen skal returnere prisen.

Hint: Prisen for ett element i arrayen er pricePerLitre ganget med antall liter.

Merk: Deloppgave b) kan løses selv om deloppgave a) ikke er besvart korrekt.

Det finnes (etter forelesers mening) ingen robust og "brukervennlig" fremgangsmåte for å finne antall elementer i en innebygget C eller C++-array. En grei rett-frem-løsning er å sende inn antall elementer som en ekstra parameter.

En bedre løsning vil i mange tilfeller være å benytte en "container-type". Container-typen er gjerne en klasse. Vi kan lage denne selv ved behov, men vi foretrekker typisk å benytte noe vi finner i standardbiblioteket. Eksempelvis std::vector.

Det følgende er en OK og kompakt løsning uten bruk av standardbibliotek eller klasser:

```
float computeTotalPrice(Fluid fluidElements[], int numberOfElementsInArray)
{
    float totalPrice = 0.0;

    for (int ii = 0; ii < numberOfElementsInArray; ++ii)
    {
        totalPrice += fluidElements[ii].litre * fluidElements[ii].pricePerLitre;
    }

    return totalPrice;
}
```

Oppgave 9

Mikrokontrollerkretser – Krystall (med utdrag fra datablad). Tegn kretsskjema for en AtMega128 mikrokontroller med strømforsyning og ekstern krystall.

Det er ikke nødvendig å tegne programmeringsgrensesnitt.

(Merk: jeg har ikke lagt ved datablad til prøveeksamen).

Lager ikke LF på denne. Krav til løsning er på nivå med krav til kretsskjema for strømforsyningskrets på lab. Husk å tegne på alle nødvendige signaler.

Oppgave 10

To programvareutviklere er leid inn for å lage styrings- og datainnsamlingsprogramvare for en solcellefarm. Prosjektet er beregnet å ta to måneder (kalenderuker) pluss testing.

a) Se for deg at du jobber i en hovedsakelig mekanisk orientert bedrift, og at det er din oppgave å argumentere overfor ledelsen for bruk av GIT versjonskontroll i dette prosjektet.

   Besvarelsen skal beskrive de viktigste fordelene ved bruk av versjonkontroll-programvare generelt. Se video fra forelesningen.

b) Hva har GIT til felles med Linux? (Ikke teknisk, men opphavsmessig).

   Linus Torvalds er skaperen av både operativsystemet Linux og det distribuerte versjonskontrollsystemet Git. Begge deler er lisensiert som fri programvare.

   (Linus Torvalds er ikke den eneste utvikleren, Linux videreutvikles nå vha. en ”community”-basert utviklingsmodell.)