UNIVERSITETET I AGDER

TEKNOLOGI OG REALFAG

Emnekode:
Emnenavn: Innebygde datasystemer for mekatronikk

Dato: 19. desember 2017
Varighet: 4 timer

Antall sider inkl. forside: 7

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

cfMerknader: 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 i en eller annen form, skal det programmeres C++ på papir.
- Det kan antas at header <iostream> er inkludert fra før, alle andre nødvendige includes må spesifiseres i løsningen.
- Kodesnuttene i oppgaveteksten er i en del tilfeller listet "frittstående". Det skal antas at disse kjøres i kontekst av en funksjon (eksempelvis inni en main-funksjon).

Velg en byggeblokk.

Oppgave 1 (7%)

Programforståelse.

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) Kodesnutten er plassert inni en main-funksjon. Hva er spesielt med main-funksjonen i et C++-program?

Den globale funksjonen main er startpunktet i programmet. Dette gjelder i de tilfellene hvor programmet kjører på et operativsystem. For programmer som eksempelvis skal kjøres direkte på en mikrokontroller, er startpunktet definert av implementasjonen (f.eks. bestemt av kompilatorprodusenten). http://en.cppreference.com/w/cpp/language/main_function)

b) Hva skrives ut?

a = 6, b = 43

c) Dersom kodesnutten utvides med følgende linje på slutten. Hva skrives ut i tillegg?

std::cout << "z = " << z << std::endl;

z = 43

d) Hva betyr & etter int på linje 6?

Dette betyr at vi deklarerer en referanse. Variabelen b har følgelig type ”referanse til int” og refererer (er et alias) til variabelen z.

Oppgave 2 (7%)

Funksjoner.

a) Skriv deklarasjonen til en funksjon som tar inn en int og returnerer en bool. Funksjonen skal hete checkSubjectId.

```
bool checkSubjectId(int subjectId);
```

b) Lag implementasjonen til funksjonen fra deloppgave a). Funksjonen skal returnere true hvis tallet som sendes inn er 234, false ellers.

```
bool checkSubjectId(int subjectId)
{
    return subjectId == 234;
}
```

Oppgave 3 (15%)

Løkker.

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

```
const uint8_t n = 25;

for (uint8_t i = 0; i < n; ++i)
{
    std::cout << +i << std::endl;
}
```

a) Hva er første og siste tall som skrives ut?

Henholdsvis 0 og 24.

b) Hvor mange tall skrives ut, og skrives de ut etter hverandre horisontalt eller vertikalt?

25 tall skrives ut. Tallene skrives ut etter hverandre vertikalt pga. Linjeskift (std::endl).

c) Kan variabelen ”i” være const i denne kodesnutten?

Nei, den kan ikke være const fordi variabelen i må endre verdi etter at den er opprettet. I dette tilfellet benyttes i som indeks i for-løkken, og ++i inkrementerer (øker) verdien på i med 1 for hver iterasjon (”runde”).

d) Skriv en tilsvarende implementasjon som skriver ut alle oddetall fra og med 5 til og med 131. Bruk den typen løkke som er best egnet for formålet.

Det bør benyttes en for-løkke her, men bruk av andre løkker kan aksepteres. En mulig løsning:

```
for (uint8_t i = 5; i <= 131; i+=2)
{
    std::cout << +i << std::endl;
}
```

En annen variant:

```
for (uint8_t i = 5; i <= 131; )
{
    std::cout << +i << std::endl;
    i = i + 2;
}
```

Oppgave 4 (8%)

Løkker.

Følgende kodesnutt er skrevet for å printe heltallene fra og med n til og med 0 i synkende rekkefølge til terminalen.

```
const uint8_t n = 10;

for (uint8_t i = n; i >= 0; --i)
{
    std::cout << +i << std::endl;
}
```

a) Er tallet 0 ett av tallene som printes til terminalen når programmet kjøres?
Ja.

b) Kodesnutten i denne oppgaven inneholder en rimelig ugrei feil. Hva er feilen, og hvordan kan denne rettes opp? (Flere ulike løsninger vil bli godtatt her).

Merk: Feilen gjelder ikke programsnutten i oppgave 3, selv om disse ligner noe strukturmessig.

Dette blir en evig løkke. Sjekken i >= 0 vil aldri bli false, da i er unsigned og følgelig ALLTID vil være >= 0. En mulig løsning er å benytte heltall med fortegn (bytt ut uint8_t med int8_t begge steder).

Oppgave 5 – Objektorientering (15%)

Gitt følgende C++-klasse:

```
class PidController : public IGenericController
{

public:
    PidController();
    virtual ~PidController();

    void setPid(double p, double i, double d);

private:

    void setP(double p);
    void setI(double i);
    void setD(double d);
};
```

a) Skriv implementasjonen til en funksjon som oppretter en PidController-instans og setter regulatorparametrene til: P=4.2, I=0.5, D=0.0.

```
int main()
{
    PidController pidController;
    pidController.setPid(4.2, 0.5, 0.0);

    return 0;
}
```

b) Kan PidController-instansen være const? Hvorfor / hvorfor ikke?

Nei, vi kan ikke deklarere variabelen pidController const. Medlemsfunksjonen setPid, som kalles senere, er ikke deklarert const, og denne kan følgelig modifisere eventuelle medlemsvariable (selv om det ikke er noen her nå).

c) PidController arver fra IGenericController. IGenericController har en destructor som er deklarert virtual. Hva betyr dette?

Bruk av virtual fører til at det opprettes en vtable for den gitte medlemsfunksjonen. Den får dynamisk binding. Hvis det opprettes en IGenericController-peker som peker til et PidController-objekt, så vil destructoren til både IGenericController OG PidController bli kalt når delete-operatoren benyttes på pekeren.

Uten bruk av virtual, blir kun foreldreklassens (IGenericController)-destructor kalt. Dette kan resultere i udefinert oppførsel dersom klassen har én eller flere andre virtual-deklarerte funksjoner.

http://en.cppreference.com/w/cpp/language/virtual

d) PidController arver fra IGenericController. IGenericController har en funksjon setTimeStepLength(double dt) som er deklarert pure virtual. Hva betyr dette for oss når vi lager klassen PidController?

Merk: Det er ikke listet kildekode for IGenericController, da denne ikke er nødvendig for å løse oppgaven.

Når medlemsfunksjonen setTimeStepLength er deklarert pure virtual, så er klassen den tilhører abstrakt. Ved arv fra en abstrakt parent-klasse har vi to valg: La child-klassen også være abstrakt, eller implementere alle pure virtual funksjoner.

I praksis benyttes pure virtual funksjoner ofte i grensesnitt for å ”tvinge” den som arver til å implementere bestemte funksjoner/metoder.

En abstrakt klasse er en klasse som ikke kan instansieres ref.
http://en.cppreference.com/w/cpp/language/abstract_class

Oppgave 6 (10%)

Programforståelse.

En kryptert beskjed er mottatt og må dekrypteres. Følgende program utfører dekrypteringen:

```
void fun(char* message, unsigned int startIndex, unsigned
int stopIndex)
{
    for (unsigned int i = startIndex; i > stopIndex;)
    {
        std::cout << message[--i];
    }
    std::cout << std::endl;
}


void decryptMessage(char* message)
{
    const unsigned int christmasEve = 24;

    std::cout << "Navnet er: " << std::endl;

    fun(message, christmasEve/2 - 2, 0);
}


int main()
{
    char secretMessage[] = "naihsadraknu-gnoj";

    decryptMessage(secretMessage);

    return 0;
}
```

a) Beskriv hva tredje parameter til fun-funksjonen gjør.
Tredje parameter til fun-funksjonen er stopIndex. Den bestemmer når (på hvilken index) for-løkken i funksjonen skal stoppe.

b) Basert på innholdet i funksjonen "fun", foreslå et bedre og mer beskrivende navn på funksjonen. Navnet skal være skrevet C++-teknisk sett korrekt, og gjøre det unødvendig med kommentarer i koden for å forklare hva funksjonen gjør.
printReverse

c) Hva er den hemmelige beskjeden? Hint: En mye omtalt Kim.

kardashian

Oppgave 7 (4%)

Lag en enumerator som inneholder de mulige verdiene red, green og blue. Gi enumeratoren et beskrivende navn. Det skal *ikke* opprettes en instans av enumeratoren, den skal kun defineres.

```
enum Color
{
    red,
    green,
    blue
};
```

Oppgave 8 (10%)

Begrepsforståelse.

a) Beskriv kort hva som skjer når vi kompilerer et C++-program.

Den tekstlige programkoden blir transformert / oversatt til et annet språk. Språket det oversettes til er typisk maskinkode (binær kode basert på instrukssjonssettet til datamaskinen vi planlegger å kjøre programmet på). Men det må ikke være maskinkode.

Bygging av et C++-program omfatter typisk tre steg; preprosessering, kompilering og linking. Dette omtales i en del sammenhenger som ”kompilering”, selv om det er noe upresist.

b) Hva skiller ”ordinær” kompilering fra krysskompilering?

Ved krysskompilering genereres det maskinkode (eller annen form for kode) ment for en annen plattform enn den kompileringen utføres på.

Ved ”ordinær” kompilering, genereres det maskinkode for plattformen kompileringen utføres på.

c) Hva mener vi med ”host” og ”target” når vi utvikler programvare for et innebygget datasystem? Tegn gjerne en enkel figur for å illustrere.

Host er maskinen utvikleren sitter på (f.eks. med et integrert utviklingsmiljø (IDE)).

Target er maskinen det utvikles programvare for.

Oppgave 9 (10%)

Mikrokontrollerkretser og CAN-bus

a) Forklar kort hva en avkoblingskondensator er, og hvordan disse normalt bør plasseres relativt til f.eks. en mikrokontroller.

Avkoblingskondensatoren er en kondensator som benyttes til å dekoble/avkoble en del av kretsen fra en annen. Først og fremst med tanke på å forhindre spredning av høgfrekvente, uønskede signaler (støy). Det benyttes ordinære kondensatorer som avkoblingskondensatorer, men typisk rimelig raske kondensatorer med forholdsvis lav kapasitans.

Avkoblingskondensatoren representerer en liten lokal energireserve der den er plassert. Når strøm eksempelvis trekkes i kortvarige pulser av en mikrokontroller, vil avkoblingskondensatoren fungere som en shunt mot jord (tett inntil mikrokontrolleren) og de skarpeste "kantene" fjernes fra signalet som brer seg videre utover i kretsen.

Avkoblingskondensatoren bør typisk plasseres fysisk nær kretselementet som skal avkobles.

b) Hva bør være foretrukket jordingsstrategi når vi designer et kretskort med flere relativt effektkrevende integrerte kretser (eller andre komponenter som har høyt strømtrekk)?

(Det samme gjelder også når vi designer en krets bestående av flere "break-out-boards" etc..).

Stjernejording. Dedikert jordplan på kretskortet.

c) Det er målt 100 kOhm motstand mellom Can High og Can Low på en CAN-bus. Hva kan dette indikere? Vil CAN-bus-en fungere i et slikt tilfelle?

Bussen er høgimpedant. Dette indikerer manglende termineringsmostand. Den fysiske CAN-bus-en skal ved bruk av kobberpar termineres som beskrevet her: http://www.ni.com/white-paper/9759/en/ (basert på ISO 11898).

Det er lite sannsynlig at CAN-bus-en vil fungere i dette tilfellet.

Oppgave 10 (7%)

Nettverkstopologier.

Tegn opp tre valgfrie vanlige nettverkstopologier og navngi disse.

F.eks. tre av disse: stjerne, bus, ring, mesh.

Oppgave 11 (7%)

Sanntidssystemer.

Velg de to punktene du mener beskriver et sanntidssystem best. (Skriv setningene fullt ut i besvarelsen).

- Et sanntidssystem må regne korrekt og ha svært god ytelse (kunne utføre mange beregninger raskt).
- Et sanntidssystem må forholde seg til tidsfrister.
- Et sanntidssystem må ha resultatet av beregningene klart til rett tid.
- Et sanntidssystem må har mange innganger og utganger.