FAKULTET FOR TEKNOLOGI OG REALFAG

E K S A M E N

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

Dato: 12. desember 2019
Varighet: 4 timer

Antall sider inkl. forside: 9

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 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).

Oppgave 1 (10%)

Programforståelse.

Gitt følgende kodesnutt:

```
unsigned int x = 50;
unsigned int y = 100;

int a = x + y;

int* z = &a;
*z += 1;

std::cout << "a = " << a << ", z = " << *z << std::endl;
```

a) Kodesnutten er plassert inni en main-funksjon i en cpp-fil. Hva skrives ut når programmet bygges og kjøres?

a = 151, z = 151

b) Hvilken funksjon har & slik operatoren benyttes i kodesnutten?

Address-of (hente adressen til variabelen)

c) Kan variabelen a i kodesnutten deklareres const? Hvorfor/hvorfor ikke?

Nei, ikke når vi endrer den via peker. Vi kan heller ikke ha den const dersom vi IKKE endrer verdien via pekeren, det er nok at vi KAN gjøre det via pekeren. Dersom vi skal ha variabelen a const, så må pekeren z også være en peker til en const int.

d) Hva er forskjellen på pre-inkrementering (eksempelvis ++i) og post-inkrementering (eksempelvis i++)? Gi et kort eksempel på bruk hvor oppførselen er forskjellig.

Pre-inkrementering endrer verdien før den eventuelt leses/brukes. Post-inkrementering endrer den etter den eventuelt leses/brukes.

Eksempel; her vil a tilordnes verdien 0, mens b vil tilordnes verdien 1. Samtidig vil både i og j ha verdien 1 etter siste statement:

```
int i = 0;
int j = 0;
int a = i++;
int b = ++j;
```

Oppgave 2 (9%)

Funksjoner og løkker.

Det skal lages en funksjon som tar inn en array av double-elementer. Funksjonen skal returnere det største elementet.

a) Skriv funksjonsdeklarasjonen.

```
double largestElement(double* numbers, int length);
```

b) Lag implementasjonen til funksjonen fra deloppgave a).

```
double largestElement(double* numbers, int length)
{
    double largest = numbers[0]; // Assume first element is largest.

    for (int i = 1; i < length; ++i)
    {
        if (numbers[i] > largest)
        {
            largest = numbers[i];
        }
    }

    return largest;
}
```

c) Hva vil være viktig her dersom funksjonen skal være effektiv på store arrays? Eksempelvis arrays med 100.000 elementer eller mer.

Her er det viktig at vi faktisk sender inn en peker til array-en med elementene, og ikke en kopi. Når det gjelder POD-arrays av denne typen, så er det uansett pekere som overføres, vi kan ikke sende en kopi av array-en uten std::copy-kall etc..

Hadde vi derimot benyttet en std::vector eller lignende, i stedet for en array, så ville en referanse vært å foretrekke.

Oppgave 3 (8%)

Standardbibliotek.

Det skal lages en funksjon som tar inn en `std::vector` av `double`-elementer. Funksjonen skal fjerne det siste elementet (merk at endringen også skal være synlig på utsiden av funksjonen!).

a) Skriv funksjonsdeklarasjonen.

```
void removeLast(std::vector<double>& v);
```

b) Lag implementasjonen til funksjonen fra deloppgave a).

```
void removeLast(std::vector<double>& v)
{
    if (!v.empty())
    {
        v.pop_back();
    }
}
```

Oppgave 4 (10%)

Programforståelse.

Gitt følgende:

```
class Something
{
    public:
        int x;
    private:
        int y;
};

int main()
{
    Something s;
    s.x = 314;
    s.y = 157;
    return 0;
}
```

a) Hvorfor kompilerer ikke programmet?

Det forsøkes å lese en variabel som er deklarert private i klassen Something fra utsiden av klassen.

b) Nå skal klassen `Something` modifiseres slik at begge medlemsvariablene er deklarert private.

I tillegg skal det lages en funksjon som gir medlemsvariabelen `x` en verdi, og en funksjon som kan hente ut verdien til `x`. Kall funksjonene henholdsvis `setX` og `getX`.

Lag en ny deklarasjon til klassen som inkluderer endringene.

```
class Something
{
public:
    void setX(int x);
    void setY(int y);
private:
    int x_;
    int y_;
};
```

c) Lag implementasjonen til `getX` og `setX`. Dette skal lages på samme måten som hvis det hadde vært plassert i en egen implementasjonsfil.

```
void Something::setX(int x)
{
    x_ = x;
}

void Something::setY(int y)
{
    y_ = y;
}
```

d) Gi et eksempel på en vanlig konvensjon for filendelser på henholdsvis implementasjonsfiler og header-filer i C++.

.cpp og .h (flere andre varianter godtas også).

Oppgave 5 (15%)

Objektorientering.

Gitt følgende C++-klasser og main-funksjon.

En typisk bruk av filtrene vil være at vi har en array eller en std::vector hvor vi ønsker å kalle filter-funksjonen på hvert enkelt element i rekkefølge.

En annen bruk vil kunne være at vi leser en sensor med fast rate, f.eks. ti ganger i sekunder, og filtrerer verdiene vi leser inn før de brukes videre i programmet.

```
class Filter
{
public:
    virtual ~Filter() = default;
    virtual double filter(double value) = 0;
};



class NoPassFilter : public Filter
{
public:

    virtual ~NoPassFilter() = default;

    virtual double filter(double value) override
    {
        return 0.0;
    }

};


class PassThroughFilter : public Filter
{
public:

    virtual ~PassThroughFilter() = default;

    virtual double filter(double value) override
    {
        return value;
    }

};
```

```
// Oppgave 5 fortsettelse:


int main()
{

    Filter* filter = new NoPassFilter();

    const double unfilteredValue = 0.9;
    const double filteredValue = filter->filter(unfilteredValue);

    std::cout << "Unfiltered value is: " << unfilteredValue
              << ", filtered value is: " << filteredValue
              << std::endl;

    return 0;
}
```

a) Hva skrives ut dersom programmet bygges og kjøres?

```
Unfiltered value is: 0.9, filtered value is: 0
```

b) Det skal byttes filter. Vi ønsker å benytte et PassThroughFilter i stedet for filteret som er benyttet nå. Vis endrede linjer.

```
Filter* filter = new PassThroughFilter();
```

c) Lag en ny filterklasse som også arver fra Filter. Din nye filterklasse skal sende signalet rett igjennom («pass through») frem til første verdi lik 0.0 dukker opp, og fra og med dette tidspunktet skal filteret kun returnere 0.0.

Gi filterklassen et passende navn. Deklarasjon og implementasjon kan lages på samme sted, som i eksemplene over.

Flere løsninger godtas. Deklarasjon og implementasjon vises her sammen (ref. oppgaveteksten) -- i praksis vil det gjerne være ønskelig å skille dette i separate filer.

```
class ZeroPassTriggeredFilter : public Filter
{
public:

    ZeroPassTriggeredFilter()
        : active_(false)
    {}

    virtual ~ZeroPassTriggeredFilter() = default;

    virtual double filter(double value) override
    {
        // Eller gjerne en betingelse som sjekker et lite område
        // rundt 0.0, i og med at eksakt 0.0 kan være vanskelig å
        // oppnå dersom verdien er samplet fra en reell sensor.
        if (value == 0.0)
        {
            // Husk at vi har funnet verdien 0.0.
            active_ = true;
        }

        // Returnerer 0.0 hvis vi nå eller tidligere har funnet
        // verdien 0.0 i inn-signalet.
        return active_ ? 0.0 : value;
    }

private:

    bool active_;

};
```

Oppgave 6 (10%)

Systemmodellering.

a) I SysML benyttes blant annet aktivitetsdiagram for å modellere systemers oppførsel. Hvilke diagramtyper er egnet for å modellere systemers struktur?

Minst to av de følgende gir full poengsum her:
Block definition diagram
Internal block diagram
Package diagram

b) Hva kjennetegner en god systemmodell i MBSE-sammenheng?

Flere svar aksepteres; se forelesningsfoiler MBSE.

Oppgave 7 (10%)

Begrepsforståelse.

a) Hva er et namespace i C++ ?

Et namespace (navnerom) gir muligheten til å organisere entiteter (eksempelvis variabler, funksjoner og datatyper inkl. klasser) i et hierarki, slik at samme navn kan benyttes flere steder i ulike namespace. Ved korrekt bruk av namespace reduseres risikoen for feil som følge av navnekollisjoner, og det blir også tydelig hvor vi henter en gitt funksjon eller datatype fra (eks. `std::vector`).

Standardbiblioteket i C++ benytter eksempelvis namespace std, slik at vi kan benytte funksjonen max i navnerommet std ved å kalle `std::max(a, b)`.

b) Lag et namespace med navn controller, og opprett en const-variabel av typen int inni namespace-et. Gi variabelen et valgfritt navn og verdi 2.

```
namespace controller
{
    const int numberOfWheels = 2;
}
```

c) Hva gjør preprosessoren når vi bygger et C++-program?

Preprosessoren kjøres før kompilering. Den håndterer blant annet ut include-filer og preprosessor-direktiver slik som #defines, #ifndef etc. Resultatet etter preprosessor-steget er en enkelt fil klar til kompilering. Preprosessering er et av stegene når C++-programmet bygges.

d) Hva gjør kompilatoren når vi bygger et C++-program?

Kompilerer filen fra preprosseserings-steget til object-fil (maskinkode + informasjon til linkesteget). Dette kan inkludere et mellomsteg hvor assembly-kode for plattformen først genereres. Men det er ikke påkrevd å gjøre dette basert på C++-standarden.

Oppgave 8 (9%)

Kommunikasjon

a) Hva menes med begrepet integritet i sammenhengen overføring av digitale data?

Integritet i sammenhengen dataintegritet, betyr at vi kan være sikre (nok) på at data ikke er endret på vei fra avsender til mottaker.

b) Nevn to metoder som kan benyttes for å verifisere integritet på en overført digital melding. Hvilke typer feil kan detekteres?

CRC (Cyclic Redundancy Check). CRC er en feildetekterende kode med stor utbredelse. CRC-er er relativt gode til å ta burst-feil, altså feil hvor flere påfølgende bit er endret.

Paritetsbit. Dette er den enkleste formen for feildetekterende kode, og kan kun detektere et odde antall feil. Hvis det oppstår endringer i et partall antall bit, så vil dette ikke gi utslag i paritetsbit-et, og feilen vil dermed ikke detekteres. Paritetsbit er egnet til å detektere at noe er galt med overføringen for relativt korte meldinger hvor konsekvensene ikke er store ved feil som ikke oppdages. Metoden er svært utbredt, pga. enkel implementasjon og enkle beregninger (som ikke tar mye kjøretid).

c) Nevn en metode som kan benyttes for å korrigere for feil i en overført melding. Hvilke typer feil kan korrigeres med denne metoden?

Flere mulige svar her. Eksempelvis:

Raptor-code (eller mer generelt fountain-code). Benyttes blant annet i DVB-T. Kan korrigere for ønsket mengde feil ved å sende en bestemt mengde ekstra informasjon. Relativt robust mot både burst-feil og enkeltbit-feil. Relativt komplisert implementasjonsmessig, og også krevende kjøretidsmessig.

Oppgave 9 (6%)

Nettverk.

Det skal overføres data mellom 3-5 enheter. To av enhetene er henholdsvis en motorstyringsnode og en node med hastighetsregulator.

Fysisk avstand mellom nodene er 200-3000 meter.

a) Velg en passende type nettverk/kommunikasjonsstandard og begrunn valget.

En mulig løsning er å holde nodene for motorstyring og hastighetsregulator så nær hverandre som mulig, slik at eksempelvis RS422 kan benyttes (3000 meter er her for stor avstand, da RS422 på «normal» lav rate går 1500 meter). Det kan settes på en repeater på midten for å oppnå 3 km rekkevidde med RS422 uten å gå uvanlig lavt på datarate.

Eventuelt kan det velges en trådløs kommunikasjonsstandard (som LoRa), noe som i de fleste tilfeller vil være rimeligere enn kablet forbindelse. Men for motorstyringsdelen i lukket sløyfe vil en kablet løsning antageligvis være å foretrekke. Hvis mulig, så er det antageligvis best å anvende en fiberoptisk forbindelse f.eks. med en passende tranceiver for større distanser.

CAN-bus på vanlig twisted pair kan også være et alternativ, distanser på opp mot 5-6 km bør gå greit med høgkvalitetskabel så lenge hastigheten er satt ned til ca. 10 kbit/s. For avstander større enn 50 meter må følgende være oppfylt:

$$\text{Signaling Rate (Mbps)} \times \text{Bus Length (m)} \leq 50$$

Noe som ved en bus-lengde på 3000 meter gir at vi må under ca. 16 kbit/s hastighet for å overholde bit-timing på bus-en. (0,016 Mbps). Denne grensen er gitt av lyshastigheten. I praksis velger vi da 10 kbit/s for å få litt sikkerhetsmargin.

b) Hvilken topologi vil du benytte for oppkoblingen? Lag en enkel figur som illustrerer valget.

Flere mulige svar basert på valgene i a).

Oppgave 10 (7%)

Operativsystemer.

Forklar kort:

1) Hva er den viktigste forskjellen på en tråd og en prosess?

   En prosess har eget minne, en tråd deler minne med andre tråder i prosessen den tilhører.

2) Hva er prosessisolasjon (eng. «process isolation»)?

   Prosessisolasjon er en maskinvare- og operativsystemfunksjonalitet som forhindrer prosesser i å, ved en feil, lese og/eller endre minne som ikke tilhører prosessen. Prosessisolasjon er sterkt ønskelig, da det er en god mekanisme for å isolere feil til prosessen de eventuelt oppstår i. En lang rekke moderne operativsystem har prosessisolasjon.

   https://en.wikipedia.org/wiki/Process_isolation

Oppgave 11 (6%)

Single board computer vs. mikrokontroller. Operativsystem/ikke-operativsystem.

Velg typen embedded-maskin du mener er egnet for følgende to applikasjoner. Gi en kort begrunnelse for valget.

a) Styring av signalgenerator/funksjonsgenerator (faktisk generere signalene). Det er behov for sinus-signaler med frekvens opp til 20 kHz.

Her kan det være en grei løsning å anvende en mikrokontroller, da tidskravene er svært strenge, og det kan være enklere å oppnå nødvendig tidsoppløsning. Alternativt kan et spesialisert sanntidsoperativsystem eller en SBC uten operativsystem anvendes. Spesialiserte prosessorer av typen DSP kan også være en god løsning her for å syntetisere signaler (men det er neppe strengt nødvendig med slikt på 20kHz frekvens).

b) Logging av tidsserie-data fra solcellepanel-system (spenning, strøm, effekt, temperatur etc.), og tilgjengeliggjøring av loggede data på HMI (brukergrensesnitt) og skytjeneste på internett.

Her er det en fordel med innebygget støtte for skjerm, ethernet, nødvendige bus-systemer for å koble seg mot solcellepanel-kontrollerene etc.. Et standard operativsystem vil være et godt valg, det ser ikke ut til å være nødvendig med strenge tidskrav og sanntidsegenskaper.