Innebygde datasystemer eksamen 2019

Oppgave 1

a)

Gangen i koden går som følger:
a = x + y = 50 + 100
a = 150
peker z = &a = adressen til a
derefererer z, setter z += 1
printer a og z
Det som skrives ut er:
a = 151, z = 151
https://www.javatpoint.com/cpp-pointers
https://www.w3schools.com/cpp/cpp_pointers_dereference.asp

b)

& operatoren gjør at man får tak i adressen til variabelen (hente adressen til variabelen

c)

Nei, variabelen kan ikke deklareres const når man endrer den via peker.

d)

Pre-inkrementering: inkrementeringen skjer før uttrykket evalueres.
Post-inkrementering: inkrementeringen skjer etter at uttrykket er evaluert.

```
1  // Eksempel 1
2  int a = 10;
3  int x = ++a;
4  std::cout << x << std::endl;
5  // Skriver ut 11
6
7  // Eksempel 2
8  int b = 12;
9  int y = b++;
10 std::cout << y << std::endl;
11 // Skriver ut 12
```

Oppgave 2

a)

```
double returnLargestDouble(double inputArray[], int lenght);
```

b)

```
double returnLargestDouble(double inputArray[], int length)
{
    double largest = inputArray[0];

    for (int i = 0; i<length;i++) {
        if (largest < inputArray[i]) {
            largest = inputArray[i];
        }
    }
    return largest;
}

//Tester at det virker
int main()
{
    double input[4] = {55, 1200, 79, 455};
    double output = returnLargestDouble(input, 4);
    std::cout << output << std::endl;
    // Forventet output er 1200, og det er det man fÃ¥r.
    return 0;
}
```

c)

Det som vil være viktig dersom denne funksjonen skal være effektiv på veldig store arrays, er at det blir brukt peker til å peke på hvor dataen ligger, i stedenfor å lage en kopi av hele arrayen som skal legges inn.

Oppgave 3

a)

```
std::vector<double> removeLastElement(std::vector<double> input);
```

b)

```cpp
std::vector<double> removeLastElement(std::vector<double> input)
{
    input.pop_back();
    return input;
}

// Tester at det virker
int main()
{
    std::vector<double> inputVector = {1,2,3,4};

    inputVector = removeLastElement(inputVector);
    for (unsigned i = 0;i < inputVector.size();i++) {
        std::cout << inputVector[i] << " ";
    }
    std::cout << std::endl;
    return 0;
}
// Printer ut: 1 2 3
```

Oppgave 4

a)

Programmet kompilerer ikke fordi int y er definert som private i klassen Something, og dermed kan man ikke kalle på s.y i main.

b)

```cpp
class Something
{
    public:
    void setX(int x);
    int getX();
    private:
        int x;
        int y;
};
```

c)

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

int Something::getX()
{
    return Something::x;
```

```
}
```

d)

- header filer har .h som filendelse
- implementasjonsfier fra .cpp som filendelse

Oppgave 5

a)

Programmet skriver ut "Unfiltered value is: 0.9, filtered value is: 0.0"

b)

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

c)

```
class ZeroTriggerFilter : public Filter
{
public:

    virtual ¬ZeroTriggerFilter() = default;

    virtual double filter(double value) override
    {
        if (value > 0.0) {
            return value;
        }else if (value < 0.0) {
            return 0.0;
        }else {
            return 0.0;
        }
    }
};
```

Oppgave 6

a)

- Block definition diagram
- Internal block diagram
- Package diagram

b)

En god MBSE modell kan ha disse spesifikasjonene:

- Svarer på behovet. Modellens ambisjon må avstemmes med tilgjengelige ressurser
- Modellen må være forståelig og oversiktlig
- Kan ha forskjellige visninger som viser kun relevante deler for forskjellige fagfelt f. eks.
- Modellens omfang/dekningsområde må være tilstrekkelig for formålet
- Modellen bør være komplett innenfor sitt omfang/dekningsområde
- Modellen må være konsistent i navngivning og utseende
- Et eksempel på overnevnte kan være at grensesnitt som kobles mot hverandre må passe sammen

Oppgave 7

a)

Et namespace i C++ er et "område" som vi kan bruke til å definere variabler som ellers ville vært globale, til å være i et mindre og definert område. Et eksempel kan være at variablene x og y er noe man ønsker å bruke flere ganger, da dette er vanlige variabler i forbindelse med koordinatsystem. Hvis man definerer x og y i to forskjellige namespace, kan man få tilgang til disse tilsynelatende like variablene i forskjellige namespace, og dermed kan de ha forskjellige verdier.

b)

```
1 namespace controller {
2 const int valgFri = 2;
3 }
```

c)

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

d)

Kompilatoren kompilerer filen fra preprosseseringssteget til en object-fil (maskinkode + informasjon til linkesteget). Dette kan da inkludere et mellomsteg hvor assembly kode for plattformen først genereres, men det er ikke påkrevd å gjøre dette basert på C++ standarden.

Oppgave 8

a)

Integritet i sammenhengen overføring av digitale data er at man ønsker å verifisere at den dataen som ble overført, er den dataen man ønsket at skulle bli overført. Man vet med andre ord hva man ønsker å sende på forhånd, og lager en type sjekk som man bruker til å verifisere at dataen ble overført. Dersom man får samme verdi i sjekken tilbake etter overføring, er alt riktig overført.

b)

CRC

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).
Paritetbit og sjekksummer i ulike varianter kan benyttes for å detektere at ett eller flere overførte bit har endret verdi på veien fra sender til mottager

c)

- Gi opp med feilmelding / feilindikator. Logge feilen, slik at den kan finnes i ettertid!
- Ikke gjøre noe aktivt (vente på neste gyldige verdi/melding).
- Kopiere forrige verdi/melding og benytte denne om igjen.
- Be om retransmisjon og vente til melding mottas på nytt (forhåpentligvis uten feil...).
- Benytte feilkorrigerende kode og beregne den korrekte meldingen/dataverdien ved hjelp av tilleggsinformasjon som er lagt til i meldingen av avsender.

Fra løsning:
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

a)

- LoRa (trådløst)
- CAN

CAN-bus på vanlig twisted pair kan 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.

b)

Ved bruk av CAN, så må det brukes bus topologi. Se google for bildeeksempel

Oppgave 10

10.1)

Prosesser har et eget minneområde, mens tråder ikke har det. Prosesser er ofte tyngre å starte enn tråder. Tråder deler minne med hverandre og foreldreprosessen.

10.2)

Prosessisolasjon er en maskinvare og operativsystemfunksjonalitet som forhindrer å lese feil minne/endre på 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.

Oppgave 11

a)

Mikrocontroller, f. eks Teensy eller Arduino. Det stilles ikke krav til HMI, WiFi, eller slike funksjoner, og da er det mer enn godt nok med en MCU.

b)

Single board computer, f. eks Raspberry Pi. Her skal det logges mye data, som også kunne vært gjort med en MCU. Det stilles derimot krav til HMI og å tilgjengeliggjøre denne dataen i sky. Dermed er en single board computer mer egnet, men disse kravene kunne også blitt tilfredsstilt med en MCU.