Laboration 2 – HI1024, Programmering, grundkurs, 8.0 hp

Dataingenjörsprogrammet, elektroingenjörsprogrammet och medicinsk teknik

KTH – MTH

Redovisning: Se särskilda instruktioner om hur redovisningen ska gå till. Läs och följ dessa instruktioner noga!

Mätdataanalys

Att använda datorprogram för mätdatainsamling och analys är mycket vanligt. Ni ska skriva ett välstrukturerat program där man kan mata in mätvärden och analysera dessa. När programmet startar får anvädaren upp en meny där denne kan välja:

v (View) - visar vilka mätvärden som finns lagrade
e (Enter) - tillåter användaren att lägga till mätvärden
c (Compute) - presenterar max, min, medelvärde och normaliserade mätvärden
r (Reset) - raderar alla inmatade mätvärden
q (Quit) - avslutar programmet

Användaren ska göra sitt val genom att ange en bokstav.

View
Visar de i programmet just nu lagrade mätvärdena. Skall kunna hantera noll mätvärden.

Enter
Användaren får här mata in mätvärden (heltal) till dess att denne slår q eller till dess att totala antalet mätvärden lagrade är tio. Då avslutas inmatningen och användaren kommer tillbaka till huvudmenyn. Observera att om användaren väljer detta alternativ flera gånger ska de nya mätvärdena läggas till. Tidigare inmatade mätvärden ska finnas kvar. Väljer man detta alternativ då programmet redan innehåller tio mätvärden får man alltså inte mata in några nya värden.

Compute
Använder de i programmet just nu lagrade mätdata för att beräkna och presentera:

Max – det största värdet
Min – det minsta värdet
Medel – medelvärdet av alla mätdata. Observera att detta ska beräknas som ett decimaltal och presenteras med två decimaler.
Normaliserade mätvärden – I detta fall betyder normaliserade mätvärden att man subtraherar medelvärdet avrundat till heltal från varje mätvärde (använd round i math.h). Har man mätvärden [2 5 7] får man normaliserade mätvärden [-3 0 2] och det är dessa som ska presenteras.

Den ska också kunna hantera att det saknas mätdata.

Viktigt förtydligande
Arrays ska användas på ett korrekt och effektivt sätt med lämpliga loop-konstruktioner. Programmet ska dela upp sitt arbete med funktioner på ett bra sätt och skicka data mellan funktioner med hjälp

av in-parametrar och returnerade värden. Inga globala variabler är tillåtna. Programmet ska bestå av en huvudfunktion (main) som bör innehålla koden för vår meny. Menyval ett till tre ska hanteras i **varsin** funktion som anropas när användaren gör respektive val. En bra uppdelning av funktionen compute kräver en separat funktion som beräknar max och returnerar detta värde. Denna skall inte skriva ut något utan bara göra en beräkning. Detsamma gäller min, medel och normaliserade värden. Dvs du ska ha separata funktioner som beräknar och returnerar dessa värden och dessa skall inte skriva ut något. För normaliserade värden gäller att eftersom vi inte kan returnera en array måste vi ha en parameter som denna kan fylla. Detta kan inte vara samma array som värdena ligger i eftersom vi inte får förstöra dessa.

Exempel på hur en körning skall kunna se ut (inmatningar i fetstil):

```
Measurement tool 2.0
VECRQ? c
No measurements
VECRQ? e
Enter measurement #1 (or q to quit): 2
Enter measurement #2 (or q to quit): 5
Enter measurement #3 (or q to quit): 7
Enter measurement #4 (or q to quit): q
VECRQ? v
[ 2 5 7 ]
VECRQ? c
Max value: 7
Min value: 2
Avr value: 4.67
[ -3 0 2 ]
VECRQ? e
Enter measurement #4 (or q to quit): -1
Enter measurement #5 (or q to quit): 0
Enter measurement #6 (or q to quit): q
VECRQ? v
[ 2 5 7 -1 0 ]
VECRQ? q
Exit Measurement tool
```

Tips

Börja som vanligt med HelloWorld. Skriv sedan koden för menyn i main. I första versionen skriver programmet bara ut vilket val användaren gjort för att sedan presentera menyn igen förutom att om man väljer q så avslutar programmet. En körning skulle då kunna se ut enligt:

```
Measurement tool 1.0
VECRQ? e
enter
VECRQ? v
view
VECRQ? q
Exit measurement tool
```

Kom ihåg problemet med enterslaget som ligger kvar när vi läser en char. En lösning var att ha ett mellanslag framför %c i scanf.

Lägg nu till att programmet anropar en tom funktion när man gör menyval 1-3. Lägg till kod för menyval quit. Troligen behöver du ingen funktion för detta val. Nu behöver du fundera på hur de olika funktionerna ska kommunicera. De flesta behöver ha tillgång till arrayen med mätdata och de behöver veta hur många mätdata som just nu är inlästa. Lämpligen deklarerar man en array measurements som ska innehålla mätdata i main och en variablel nrOfMeasurement som ska hålla reda på hur många mätvärden som i varje läge finns i programmet. Dessa behöver skickas med till de flesta funktioner.

En bra ordning att arbeta med funktionerna är:

enter (så att vi kan få in data)
view (så vi kan se data)
compute

Kom ihåg att om man skickar en array som parameter så gäller att om man ändrar värden i arrayen så ändras värdena även i main. Detta är ju bra eftersom vi vill fylla på arrayen i enter. Vi vill också ändra nrOfMeasurements i en del funktioner. Dock ändras inte in-parametrar i den anropande funktioner om vi ändrar variabelns värde i funktionen. Så om vi tex skickar in nrOfMeasurements som för tillfället är noll till funktionen enter och denna funktion ändrar värdet på nrOfMeasuremnts till 4 så kommer nrOfMeasuremnts fortfarande att vara noll i main. Vi måste därför returnera det nya värdet för nrOfMeasurements från funktionen enter till huvudprogrammet. Ett anrop skulle då kunna se ut enligt:

```
nrOfMeasurements = enter(measurements, nrOfMeasurements);
```

När du ska presentera normaliserade mätdata i compute bör du kunna anropa samma funktion som anropas vid menyvalet view men nu med den normaliserade arrayen. Det är aldrig bra att skriva in samma kod flera gånger.

Tips för enter. Den ska ju sluta läsa in tal om användaren anger q. Detta löser du genom att använda att scanf returnerar 0 om den misslyckas med att läsa ett heltal men 1 om den lyckas. Dock kommer den att lämna q att läsa för nästa funktion. Du måste därför läsa bort denna. Skiss på förslag:

```
int readInteger = scanf(”%d”,&oneMeasurement);
if(readInteger)
{
          //kod som hanterar att vi last in ett nytt mätdata
}
else
{
          char tmp;
          scanf(" %c",&tmp);
          //se till att vi avslutar funktionen
}
```

Observera att den kommer sluta läsa in tal oberoende bokstav men det är ok.

Sist fixar du så att reset fungerar. Troligen behöver du ingen funktion för detta alternativ.