Kapitel 4 Loopar

Tillämpning: Varningsmeddelanden

Denna tillämpning (VARNINGSMEDDELANDEN) använder loopar för att mata in så många värden som behövs, och som en utvidgning, kontrollerar om inmatade värden är giltiga och använder **If**-satser för att visa ett passande meddelande.

Lärarkommentar: Detta projekt illustrerar tendensen mot större komplexitet när man utvecklar en bit programvara. Nästling relaterar till idén om att lägga en kontrollstruktur innanför en annan. Som vi förklarar längre fram i detta dokument så vet räknarens operativsystem vilket End som hör till vilken programsats.

Syfte:

- Lära sig om räkneverk och ackumulatorsatser.
- Använda en **loop** i ett program för att få en obestämd mängd av data.
- Använda ett ”flaggvärde” för att avsluta en loop.

Nästlade Strukturer

- *Nästling* är en programmeringsteknik för att placera en kontrollsats inne i en annan.
- För att undvika fel är det viktigt att placera en hel struktur fullständigt inne i ett block hos en annan struktur.
- Programlistningen till höger visar en **If**-struktur innanför **Else**-blocket hos en annan **If**-struktur. Observera att det finns flera **End**-satser. Datorn håller reda på vilka **End** och **If** som hör ihop.
- **Indragen** i programlistningen finns bara för att tydliggöra strukturen. Du kan inte ha indrag i TI-Basic.
- Programmet kontrollerar först om **A<0**. Om A är negativt så visar programmet meddelandet “A är negativt!” och ingenting annat. Men om **A** inte är negativt så exekveras den **understrukna** kvadratrotsberäkningen, en annan **If**-sats och **Disp** satsen.

```
prgmSQUARE
Prompt A
If A<0
Then
  Disp "A är negativt!"
Else
  √(A)→S
  If S=heltal (S)
  Then
    Disp "A är en perfekt
kvadrat."
  Else
    Disp "A är inte en”
    Disp ”perfekt kvadrat”
  End
  Disp "Dess kvadratrot är",S
End
```

Sammanfattning av de tre typerna av loopar:

For(*variabel*, *start*, *slut***)**	**While** *villkor är sant*	**Repeat** *tills villkor är sant*
End	**End**	**End**

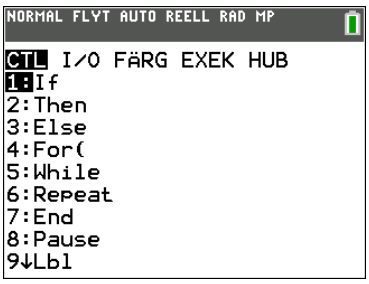
- **For(** används när man ”räknar upp” eller processar en aritmetisk sekvens av värden (iteration).
- **While** används när du har möjlighet att helt hoppa över loopkroppen.
- **Repeat** används när du är säker på att du vill att loopkroppen ska ”köras” åtminstone en gång.

Om ”End”

End används för alla flerradiga kontrollstrukturer:

If … **Then** **End**	**If …** **Then** **Else** **End**	**For(…)** **End**	**While …** **End**	**Repeat …** **End**

Se tabellen på förra sidan. Nu förstår du varför kontrollmenyn i programeditorn är uppställd enligt If, Then, Else … End



7: End kommer efter de första 6 posterna eftersom End avslutar var och en av de sex kontrollstrukturerna.

End-satser kan förekomma många gånger i ett program och datorn ”vet” hur alla **End** hör ihop med sina respektive kontrollstrukturer.

Kapitel 4 Tillämpning: Programmet “Varningsmeddelanden”

Många skolor skickar ut regelbundet resultatrapporter som talar om hur eleverna ligger till i olika ämnen. Man anger då ett slags medelvärde, baserat på ett valfritt antal delresultat, i en skala mellan 0 och 100. Ett medelvärde på 70 eller högre räknas som *godkänt*. Ligger det mellan 60 och 70 är det ”*på gränsen*” och ligger det under 60 är det ”*underkänt*”.

Skriv ett program som låter användaren mata in ett antal delresultat och sedan matar ut antal inmatade delresultat, medelvärde och omdöme (underkänt, på gränsen eller godkänt.

Vi kan använda två metoder för att mata in ett okänt antal delresultat:

- Metod 1: man frågar först efter antal delresultat och använder en **For**-loop för att mata in poängen för delresultaten.
- Metod 2: man frågar efter delresultat men använder en ”flaggvärde”, t.ex. -999, för att tala om att det inte finns fler delresultat. Denna metod använder en **While**-loop eller en **Repeat**-loop.

I båda metoderna måste vi löpande se till att ha en totalpoäng. I Metod 2 måste vi också räkna *antalet* delresultat så att vi kan dividera totalresultatet med detta antal.

Ditt program ska alltså visa antalet delresultat, medelvärdet och ett omdöme.

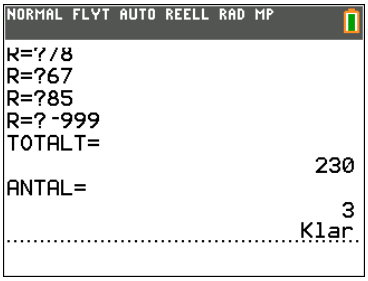
Lärarkommentar: Avsnittet nedan diskuterar två relaterade programmerings-idéer: ett räkneverk och en ackumulatorvariabel. Poängtera att variabeln på vänster sida av lagringsoperatorn är samma variabel som den till höger men att de representerar olika värden på grund av den involverade bearbetningen.

Räkneverk och ackumulatorer

En sats som **C+1→C** kallas ett *räkneverk* eftersom den adderar 1 till variabeln **C** varje gång den exekveras. **T+N→T** kallas en *ackumulator* eftersom den löpande håller reda på totalen för variabeln **N**. Värdet hos **N** adderas till variabeln **T** och sedan lagras den summan tillbaka till variabeln **T**. I slutet av en loop kommer **T** att innehålla totalen av **N**-värdena.

Här är ett exempel som använder ett räkneverk, en ackumulator och en ”flaggvärde” för att hålla reda på **R** (Resultat)i programmet.

	Kommentarer:
	Initialvärden för variablerna;
0→A	**A** står för Antal
0→R	**R** står för Resultat
0→T	**T** står för Totalt
Prompt R	Frågar efter första **R**esultatet
While R≠-999	så länge det inte är -999
A+1→A	Lägger till 1 till **A**ntal
T+G→T	Lägger till **R**esultat till **T**otalt
Prompt R	Frågar efter ett nytt Resultat
End	
Disp "TOTALT =",T	
Disp "ANTAL =",A	



While-loopen ovan fortsätter och ”ackumulerar” resultaten **(R)** så länge inte -999 är inmatat. När -999 är inmatat stannar loopen och resultaten visas.

Lärarkommentar: Titta närmare på de två **Prompt**-satserna i koden ovan. Den första ger det första resultatet och den sista resten av resultaten. Den sista är i slutet av loopen för att förbereda för att återgå till **While**-satsen och testa värdet på R igen.

Utvidgning

Som en del av inmatningsrutinen ska du se till att värden som inmatas är giltiga (ska ligga mellan 0 och 100). Om inmatat värde ligger utanför detta intervall ska lämplig åtgärd vidtas (vilken?).

Lärarkommentar: Utvidgningen kräver en annan loop runt inmatnings-satserna för att säkerställa att inmatat värde är giltigt. Koden på nästa sida stoppar helt enkelt programmet när ett ogiltigt värde matas in. Kanske kan några elever hitta en bättre lösning!

Indragen i programmet finns här bara för att öka tydligheten. I TI-Basic finns inte indrag.

```
ClrHome
0→R
0→A
0→T
Input "MATA IN ETT RESULTAT:",R

While R≠-999
 If R<0 eller R>100
 Then
  Disp "UTANFÖR OMRÅDET!"
  Stop
 Else
  A+1→A
  T+R→T
 End
 Input "ETT TILL ELLER -999):",R
End
T/A→M

ClrHome
Output(1,1,"ANTAL DELRESULTAT:")
Output(2,1," TOTALPOÄNG:")
Output(4,1,"MEDELVÄRDE:")
Output(1,19,A)
Output(2,19,T)
Output(4,13,M)
If M<60
Then
 Output(5,9,"UNDERKÄNT")
Else
 If M<70
 Then
  Output(5,9,"PÅ GRÄNSEN")
 Else
  Output(5,9,"GODKÄNT")
 End
End
Pause
ClrHome
```

När man testkör ett program bör man prova extrema situationer. Mata t.ex. in -999 som det första resultatet eller pröva att använda negativa värden. Om programmet visar ett felmeddelande så har man inte tagit hänsyn till alla ”scenarion”. Programmet på denna sida kommer att avslutas om man matar in -999 som det första värdet eftersom det då försöker dividera 0 med 0, vilket inte är definierat. Beräkningarna av medelvärdet, M, borde omgärdas av en test för att säkerställa att åtminstone ett resultat har matats in. Nedan har vi listat ett sådant test.

```
If A>0
Then
 T/A→M
Else
 Disp “INGA RESULTAT INMATADE!”
 Stop
End
```

Stop satsen används i detta program för att avsluta ett program *omedelbart*. Man kan lägga till en sådan sats var som helst i ett program så att exekveringen avbryts och meddelandet ”Klart” visas på startskärmen.