Skilyrða setningar í C#

Skilyrðasetningar eru stundum einnig kallaðar „control statements“ eða stýri setningar vegna þess að þær stjórna flæði forritsins. Sjálfur kýs ég að kalla þetta skilyrða setningar vegna þess að þær stjórnast allar af skilyrðum, þó markmið þeirra sé að sjálsögu að stýra flæði í keyrslu forrits.

Skilyrða setningar eins og þessar sem hér fara á eftir eru undirstaða ákvörðunartöku í forritum og því er mikilvægt að þekkja vel þá kosti sem eru mögulegir og velja rétt fyrir hvert tilfelli fyrir sig.
Fyrsta setningin er IF. Hún virkar þannig að ef skilyrðið sem sett er fram í haus IF setningarinnar stenst, er sönn, er sá kóði sem er innan IF setningarinnar keyrður.
Framsetning IF setningar getur verið mismunandi og fer það bæði eftir persónulegum stíl manna og hversu mikill kóði er innan IF setningarinnar sem ræður því helst hvernig hún er útfærð.
Dæmi um framsetningu IF setningar:

```
IF(skilyrði)
{
        //kóði sem keyra á ef skilyrði stenst
}
```

Annað dæmi:

```
IF(skilyrði)
{//kóði sem keyra á, ef skilyrði stenst}
```

Þar sem við erum oft að vinna með skilyrði sem fela það í sér að vera tvíkosta þá er bætt við ELSE hluta við IF setninguna og þannig getum við látið eitthvað gerast ef frumskilyrði stenst, annars förum við í ELSE hlutann og keyrum þann kóða.

Framsetning á IF með ELSE hluta:

```
IF (skilyrði)
{
        //kóði innan IF setningar.
}
ELSE
{
        Kóði innan ELSE hluta IF setningar.
}
```

Núna er virkni setningarinnar orðin þannig að ef IF skilyrðið stenst ekki, hoppar þýðandi forritsins yfir IF hlutann og keyrir ELSE hluta setningarinnar. Þessa útgáfu setningarinnar er því eðlilegast að nota þegar möguleikanir eru 2 og annar hvor þeirra þarf alltaf að keyra. Við getum ímyndað okkur að við séum að búa til forrit sem tekur við tölu frá notanda og ef hún er 2 eða hærri eigi að margfalda töluna með 2 annars skal hún margfölduð með 3.

Þetta mætti setja fram í IF setningu:

```
IF (tala >= 2)
{
	Tala = tala * 2
}
ELSE
{
	Tala = tala *3
}
```

IF setningar geta líka haft IF setningar innra með sér (nested IF) dæmi um þetta er IF setningin hér að neðan

```
IF(skilyrði)
{
	IF(skilyrði)
	{
		//Kóði innan innri if setningar.
	}
}
```

Eins og sést á þessu er hægt að búa til mjög flókin tré af IF setningum og þau geta innihaldið eins margar IF setningar og forritaranum langar að búa til. En eins og gefur að skilja er það ekki mjög læsilegur forrits kóði ef þetta yrði gert í miklum mæli þess vegna er ágætis viðmið að nota IF setningar ef valmöguleikarnir eru 1 eða 2. Þegar valið er orðið meira er gott að ath hvort valið megi ekki útfæra með öðrum aðferðum. Þá kemur case setningin sér vel. Skoðum dæmi um hana

```
Switch(skilyrði)
{
	case skilyrði útkoma 1:
	//kóði fyrir hvað á að gerast þegar skilyrði er útkoma 1.
	case skilyrði útkoma 2:
	//kóði fyrir hvað á að gerast þegar skilyrði er útkoma 2
	Default:
	//kóði sem keyrður er þegar útkoma sem ekki er tekið á hér að ofan kemur upp
}
```

Skoðum fyrst hvað case setningin hérna fyrir ofan er að gera switch(skilyrði) segir til um hvaða breytu er verið að fylgjast með í case setningunni. Forritarinn þarf svo að ákveða hvað útkomu þess skilyrðis sem hann vill fylgjast með og bregðast við. það fer síðan eftir því hvaða útkomu breytan hefur hvaða kóði er keyrður. Breytan, skilyrði, getur t.d. verið tala 1,2,3 og svo framvegis eða bókstafur a,b,c og svo framvegis. Athugið default þarf ekki að vera í case setningum en gott er að

hafa það til þess að taka á óvæntri útkomu úr breytunni okkar. Þannig getum við t.d. bent notandanum á að hann hafi valið einhvern möguleika sem við gerðum ekki ráð fyrir og getum þá beðið hann um að velja aftur.

Skoðum dæmi

```
Switch (tala)
{
	case 1:
	//kóði sem keyrir þegar tala = 1
	case 2:
	//kóði sem keyrir þegar tala = 2
	Default:
	// val á ekki við veldu aftur
}
```

Ef breytan er svo bókstafur gæti case setningin litið svona út:

```
Switch (stafur)
{
	case a:
	//kóði sem keyrir þegar stafur = a
	case b:
	//kóði sem keyrir þegar stafur = b
}
```

Eins og sést á þessum case setningum hér að ofan er mjög hentugt að nota þær þegar valmöguleikarnir eru orðnir mjög margir vegna þess að uppsetning case setninga er þétt og taka þær þess vegna ekki mjög mikið pláss í kóða. Lang oftast koma case setningar í stað margra IF setninga því IF setninga tré verða mjög pláss frek í forritum.

Síðasta skilyrðasetningin er do while setningin hún virkar þannig að kóðin á milli do while er keyrður þar til skilyrði setningarinnar hefur verið uppfyllt þá er keyrslu forritsins haldið áfram í næstu línu. Skoðum dæmi um do while

```
do
{
	//kóði sem keyrir innan do while.
} while key != “x“
```

Í dæminu hérna að ofan mun do while lykkjan keyra þar til ýtt er á x á lyklaboðinu þá heldur keyrsla forritsins áfram. Gott dæmi um notkun do while er t.d. til að endurtaka eitthvað aftur og aftur þar til notandi vill hætta keyrslu. Dæmi um þetta gæti t.d. verið spurning til notanda sem endurtekin er þar til ýtt er á x.

Það gæti þá litið svona út í kóða.

```
Do
{
        //skrifa spurningu
} while(key != “x”);
```