Kapitel 3 Betinget logik i C#

Betinget logik er udelukkende et spørgsmål om ordet IF. Det er faktisk umuligt at programmere effektivt uden at gøre brug af IF. Du kan skrive små simple programmer. Men når det bliver mere kompliceret er du nød til at lære betinget logik.

Som et eksempel, kan vi tage et program som f.eks. en lommeregneren. Det har en Plusknap og en minusknap. Vi kan dog ikke på forhånd sige, hvilken af de to knapper dine brugere vil klikke på. Ønsker de at subtrahere eller addere? Du skal derfor være i stand til at skrive kode, der kan gør følgende:

IF der klikkes på plus knappen så skal der lægges samme
IF der klikkes på minus knappen så skal der trækkes fra

Du kan om arrangere dine sætninger ovenfor.

Blev der klikket på plusknappen? Yes eller No?
Blev der klikket på Minuskappen? Yes eller No?

Svaret på hvert af dem kan enten være Ja eller Nej – der er enten klikket på knappen eller ej.

IF udtryk

Når du skal teste for YES eller NO værdier, kan du bruge et IF udtryk. Du skriver dem på følgende måde:

```
if ()
{

}
```

Du starter med ordet if (skrevet med småt), og taster et par bløde parenteser. I mellem de bløde parenteser skriver du det der skal undersøges for (Hvilken knap blev der klikket på?). Efter de bløde parenteser er det bekvemt (men strengt taget ikke nødvendigt) at tilføje et par krøllede parentes. I mellem de krøllede parenteser kan du skrive din kode. Din kode er det du vil have der skal udføres hvis der svares YES til dit spørgsmål, eller hvis svaret var NO. Her er et kode eksempel:

```
bool knapKlikket = true;

if (knapKlikket == true)

{

        MessageBox.Show("Der blev klikket på knappen");

}
```

Bemærk den første linje:

bool knapKlikket = true;

Her er en variabeltype du ikke har set før – bool. Bool er en forkortelse **Boolean**[2]. Du bruger Booleske variabler når du ønsker at undersøge for true (sandt) eller false (falsk) værdier (YES eller NO). Denne variabeltype kan kun være true eller false. Navnet på bool variablen er i dette tilfælde knapKlikket. Vi sætter værdien til true.

De næste linjer i vores IF sætning ser sådan ud:

```
if (knapKlikket == true)
{
        MessageBox.Show("Der blev klikket på knappen");
}
```

De dobbelte lighedstegn (==) er en anden ting du skal bruge når du arbejder med IF sætninger. Det betyder "Har værdien". De dobbelte lighedstegn kaldes også en **Betinget operator**. (Der er lidt flere af dem som du ser lidt senere.) Hele linjen lyder:

"IF knapKlikket har værdien true"

Hvis du mangler et af lighedstegnene, vil du have følgende:

```
if (knapKlikket = true)
```

Det der sker her er at du tildeler en værdi til variablen knapKlikket. Der undersøges ikke om knapKlikket "Har værdi" true. Forskellen er vigtig og vil volde dig mange problemer hvis du ikke gør det rigtigt!

Imellem de krøllede parenteser i IF sætningen har vi en simpel linje med MessageBox. Linjen vil dog kun blive udført hvis IF knapKlikket har værdien true.

Lad os prøve det. Start et nyt projekt (File og New Project). Tilføj en ny knap til din formular og sæt Text egenskaben til "IF sætning". Dobbeltklik på knappen og tilføj kode vist ovenfor. Dit kodevindue ser nu sådan ud:

```
private void btnLogik_Click(object sender, EventArgs e)
{
    bool knapKlikket = true;

    if (knapKlikket == true)
    {
        MessageBox.Show("Der blev klikket på knappen");
    }
}
```

Kør dit program og klik på knappen. Du ser nu en meddelelsesboks. Stop programmet og ændre linjen:

bool knapKlikket = true;

[2] George Boole (udtales / bu ː l /) (2. november 1815 - 8. december 1864) var en engelsk matematiker og filosof.

Til dette:

```
bool knapKlikket = false;
```

Det eneste vi ændre er true til false. Kør dit program igen og klik på knappen. Hvad sker der? Ingen ting!

Årsagen til at der ikke sker noget er at vores IF sætning undersøger for værdien true:

if (knapKlikket == true)

C# udfører kun koden mellem parenteserne hvis (IF) og kun hvis (IF) knapKlikket har værdien true. Da du har ændret værdien til false, bekymre den sig ikke om den MessageBox der er mellem parenteserne. Den haster videre i koden.

Else

Man kan dog også angive hvad der skal ske hvis svaret er false. Det du skal gøre er at bruge ordet **else**. Du gør det sådan:

```
if (knapKlikket = = true)
{
}
Else
{
}
```

Man skriver bare ordet **else** efter de krøllede parenteser i IF sætningen. Bagefter kommer endnu et par parenteser. Du skriver så hvad der skal ske hvis IF sætningen var false. Ændre din kode til følgende:

```
if (knapKlikket == true)
{
        MessageBox.Show("knapKlikket har værdien true");
}
Else
{
        MessageBox.Show("knapKlikket har værdien false");
}
```

Vi læser det på følgende måde:

"Hvis det er sandt at **knapKlikket** har værdien true, så gør noget. Hvis værdien ikke er true, så gør noget andet."

Kør dit program og klik på knappen. Du ser nu en Meddelelsesboks. Stop programmet og ændre den første linje tilbage til true. Sådan her:

```
bool knapKlikket = true;
```

i stedet for:

```
bool knapKlikket = false;
```

Kør dit program igen og klik på knappen. Denne gang vil den første meddelelsesboks vises.

Hele ideen med at brugen af IF...Else udtrykket, er at vi er interesseret i at få udført en del af koden i stedet for en anden del af koden.

Du kan også udvide din IF sætning og tilføje en **else...if** del. Det gør den mere brugbar i forbindelse med vores lommeregner program.

Else ... IF i C#

I stedet for kun at benytte **else** kan du også bruge **else if**. Hvis vi bruge en lommeregner som eksempel, vil vi gerne have mulighed for følgende:

```
bool plusButtonClicked = true;
bool minusButtonClicked = false;

if (plusButtonClicked == true)
{
//Skriv den kode der skal udføre hvis der skal lægges sammen
}
else if (minusButtonClicked == true)
{
//Skriv den kode der skal udføre hvis der skal trækkes fra
}
```

Koden undersøger altså hvilken knap der blev klikket på. Hvis det var Plus knappen bliver den første If sætning udført. Hvis det var minus knappen der blev klikket, så er det den anden IF sætning der blev udført.

Så vi kan sige at **else if** er det sammen med **if**, med ordet **else** i starten.

I den næste lektion skal vi se nærmere på Switch sætninger.

Switch sætning i C#

En nemmere måde at skrive kode til situationer med mange valgmuligheder er at bruge **switch** sætningen i stedet for **else if** sætningen. En switch sætning lader dig undersøge hvorvidt der blandt en række muligheder forefindes en mulighed der er true. Det er som en liste af sætninger. Strukturen i en switch sætning ser ud på følgende måde:

```
int caseSwitch = 1;
switch (caseSwitch)
{
    case 1:
        Console.WriteLine("Case 1");
        break;
    case 2:
        Console.WriteLine("Case 2");
        break;
    default:
        Console.WriteLine("Default case");
        break;
}
```

Efter ordet switch har du et par parenteser. Imellem parenteser skriver du det du vil have undersøgt. Man tester normalt indholdet i en variabel. Derefter laves et par krøllede parenteser. Imellem de krøllede parenteser har du et senarie for hver ting din variabel kan indeholde. Du skriver derefter den kode du vil have udført, hvis det pågældende senarie er true. Efter din kode skriver du ordet break. Dette gør det muligt for C# at bryde ud af switch sætningen.

Operatorer i C#

Du har allerede set en betingelses operator, det dobbelte lighedstegn (==). Du så det i IF sætningen da du skulle undersøge om en variabel antog en værdi:

```
If (minVariabel == 10)
{
        // Udfør noget kode her
}
```

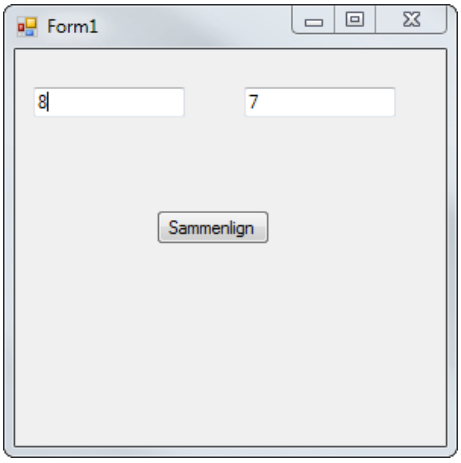
Disse linjer læses som, "If (hvis) variablen **minVariabel** har værdien 10, udfør da koden."

Her er de andre betingelses operatorer du vil se når du koder:

>	Større end
<	Mindre end
>=	Større end eller lig med
<=	Mindre end eller lig med
!=	Ikke lig med (Not equal to)
!	Ikke (Not)
&&	Og (And)
||	Eller (Or)

Da du skal lære at arbejde med disse operatorer kan vi ligeså godt afprøve dem.

Start et nyt projekt. Tilføj to tekstbokse og en knap til din formular. Ændre størrelsen på dine tekstbokse og skriv 8 i Text egenskaben i den første tekstboks, og 7 i den anden tekstboks Text betingelse. Sæt Text betingelsen for knappen til "Sammenlign". Din formular vil nu se sådan ud:



Dobbelt klik på knappen for at se koden i kodevinduet. Det vi skal gøre nu er at vi skal tage tallene fra tekstboksene og sammenligne dem. Vi starter derfor med at angive et par variabler:

```
int forsteTal;

int andetTal;
```

Derefter skal tage teksten fra tekstboksene og gemme dem i variablerne (efter vi har konverterede den til heltal først.)

```
forsteTal = int.Parse(textBox1.Text);

andetTal = int.Parse(textBox2.Text);
```

Nu skal vi sammenligne de to tal. Er det første tal er større end det andet tal? For at besvare dette kan vi gøre brug af en IF sætning, sammen med en af vores betingelses operatorer. Tilføj dette til din kode:

```
if (forsteTal > andetTal)

{

        MessageBox.Show("Det første tal er større end det andet tal.");

}
```

Dit kodevindue ser nu sådan ud:

```
private void button1_Click(object sender, EventArgs e)
{
    int forsteTal;
    int andetTal;

    forsteTal = int.Parse(textBox1.Text);
    andetTal = int.Parse(textBox2.Text);

    if (forsteTal > andetTal)
    {
        MessageBox.Show("Det første tal er større end det andet tal.");
    }
}
```

Imellem de to parenteser efter **if**, har vi vores to variabler. Vi sammenligner dem for at undersøge om en af dem er **Større End** (>) den anden. Hvis **forsteTal** er Større end **andetTal** vil meddelelsesboksen blive vist.

Kør programmet og klik på din knap. Du skulle nu se at dialogboksen bliver vist. Prøv at indtaste 6 i den første tekstboks og klik på knappen igen. Meddelelsesboksen vise ikke. Det er fordi 6 ikke er større end 7. Koden til meddelelsesboksens er i mellem de krøllede parenteser i IF sætningen. Og IF sætningen bliver kun udført hvis forsteTal er Større End andetTal. Hvis det ikke er opfyldt vil C# bare køre videre til næste linje. Du har dog ikke flere linjer, så C# stopper.

Stop dit program og vend tilbage til din kode. Tilføj en ny if sætningen nedenunder den første:

```
if (forsteTal < andetTal)
{
        MessageBox.Show("Det første tal er mindre en det andet tal.");
}
```

Det vi har skiftet er brugen **Mindre End** (<) i stedet for symbolet Større End (>). Vi har desuden skiftet teksten der skal vises i meddelelsesboksen.

Kør dit program igen og skriv 6 i den første tekstboks. Du er nu den nye meddelelsesboks. Tast nu 8 i den første tekstboks, og klik på knappen. Den første meddelelsesboks vises. Kan du se hvorfor? Hvis dit program ikke virker, så sikre dig at der ser ud som det vist nedenfor:

```
private void button1_Click(object sender, EventArgs e)
{
    int forsteTal;
    int andetTal;

    forsteTal = int.Parse(textBox1.Text);
    andetTal = int.Parse(textBox2.Text);

    if (forsteTal > andetTal)
    {
        MessageBox.Show("Det første tal er større end det andet tal.");
    }

    if (forsteTal < andetTal)
    {
        MessageBox.Show("Det første tal er mindre end det andet tal.");
    }
}
```

Kør dit program igen og tast 7 i den første boks. Du har da 7 i begge tekstbokse. Før du klikke, kan du da gætte hvad der sker?

Grunden til at der ikke sker noget er fordi du ikke har skrevet noget kode, der fortæller hvad der skal ske hvis begge tal er ens. Derfor tilføjer vi følgende symbol:

>= (Større end eller lig med)

Og dette

<= (Mindre end eller lig med)

Prøv disse nye betinget operatorer i stedet for dem du har. Skift teksten i meddelelsesboksen så de passer. Kør din kode igen. Når du klikker på knappe, vil begge meddelelsesbokse blive vist, en efter en. Kan du se hvorfor dette sker?

En anden betinget operator du kan prøver er Ikke Lig Med (!=). Det er et udråbstegn efterfulgt af et lighedstegn. Det bruges på følgende måde:

```
if (forsteTal != andetTal )
{
        // Her kommer noget kode
}
```

Det læses som, "IF forsteTal ikke er lig med andetTal udfør da noget kode."

Du kan også bruge udråbstegnet alene. Du gør typisk dette hvis du vil undersøge for værdien false mellem de runde parenteser efter if. Det bruges oftest i forbindelse med booleske værdier. Her er et eksempel:

```
bool testValue = false;
if (!testValue)
{
        MessageBox.Show("Value was false");
}
```

Udråbstegnet står foran den booleske værdi du vil undersøge. Det er en måde at sige "Hvis den booleske værdi er falsk". Du kan også skrive denne linje i stedet for:

```
if (testValue == false)
```

Erfarne programmører bruger dog udråbstegnet i stedet for. Det kaldes NOT operatoren. Eller "IF NOT TRUE" operatoren.

Du skal dog ikke bekymre dig hvis du ikke helt har styr på alle de betingede operatorer endnu – du lærer dem efterhånden som du kommer i gang. Prøv følgende øvelse.

Lav et lille program med en tekstboks og en knap. Tilføj en label hvor du beder brugeren om at indtaste deres alder. Brug betinget logik til at undersøge hvor gamle de er. Vis følgende meddelelser, afhængig af hvor gamle de er:

Mindre end 16: "Du er stadig den yngste."
Over 16 men under 25: "Du er ved at være der!"
Over 25 men under 40: "Der er stadig tid."
Over 40: "Åh nej, du er sikkert for sent ude!

Der skal kun vises en meddelelsesboks, når du klikker på knappen. Her er noget kode du kan starte med:

```
int alder;

alder= int.Parse(textBox1.Text);

if (alder < 17)
{
                MessageBox.Show("Du er stadig den yngste.");
}
```

Til de andre skal du bare bruge flere IF sætninger og flere betinget operatorer.

AND og OR
De to sidste operatorer vi skal se på er disse:

&& (And)
|| (Or)

De kendes også som logiske operatorer, snare end betinget operatorer (som NOT operatoren),

De to og-tegn (&&) betyder AND. Du kan bruge dem på følgende måde:

```
bool isTrue = false;
bool isFalse = false;

if ( isTrue == false && isFalse == false )
{

}
```

Du bruger AND operatoren når du vil undersøge mere end en værdi. I linjen ovenfor undersøger du om begge værdier ef false. Hvis og KUN hvis begge dine egenskaber er opfyldt vil koden mellem de krøllede parenteser blive udført. I koden ovenfor siger vi:

"Hvis isTrue har værdien false AND hvis isFalse har værdien false da og kun da vil koden blive udført.

Hvis isTrue rent faktisk er true, vil koden mellem de to krøllede parenteser ikke blive udført – de skal begge være false, i vores kode.

Du kan også undersøge om bare en af egenskaberne er opfyldt. Det gør du med OR (||) operatoren. OR operatoren er to lodrette streger. (Tegnet | kaldes også pipe tegnet.) Du kan bruge dem på følgende måde:

```
bool isTrue = false;

bool isFalse = false;

if ( isTrue == false || isFalse == false )

{

}
```

Vi siger nu:

”Hvis isTrue har værdien false OR hvis isFalse har værdien false, da og kun da vil koden blive udført.”

Hvis bare en af variablerne er false, vil koden mellem de krøllede parenteser blive udført.

Hvis det lyder lidt kompliceret, skal du ikke være bekymret – du vil få noget øvelse lige om lidt.

I det næste kapitel skal vi se på løkker, som en anden vigtig forhindring du skal igennem i programmering.