Logische variabelen in Basic

H. J. C. Otten

Hogere programmeertalen beschikken meestal over het datatype boolean, een datatype met als mogelijke waarden true en false (0 of 1 etc.). Logische of boolean variabelen komen van pas bij het nemen van beslissingen en als true en false als waarden voldoende zijn. Een boolean variabele maakt een programmastructuur vaak duidelijker. In standaard Basic ontbreekt dit type, in de Microsoft implementaties van Basic blijkt dit type wel in een bruikbare vorm aanwezig. De summiere documentatie vereist echter wat uitleg.

Datatypen in Basic

In Basic hebben we de beschikking over drie datatypen waarmee we informatie kunnen voorstellen in een programma. Dit zijn de real, de integer en de string waarmee we getallen, gehele getallen en rijtjes karakters kunnen voorstellen.
Uitgaande van deze eenvoudige datatypen kunnen we gestructureerde datatypen construeren met het arraytype: arrays van real, integer of string. Het datatype boolean ontbreekt helaas in Basic.

Relaties

Tussen de eenvoudige datatypen zijn relaties te bepalen met de relatie operatoren (= ,<, >, = < en = >).
Het resultaat van zo'n relatie is waar of niet waar, een boolean resultaat dus.
Relaties komen meestal voor in de conditie van een
IF...THEN... constructie:
IF relatie THEN statement
Afhankelijk van de relatie wordt het statement achter THEN wel of niet uitgevoerd. Ook in Basic hebben we met een boolean resultaat te maken.

Logische operatoren

In vele Basic-implementaties, zo ook in de Microsoft versies, zijn er drie logische operatoren beschikbaar: AND, OR en NOT. Met deze operatoren kunnen relaties worden gecombineerd tot ingewikkelder relaties met nog steeds een boolean resultaat. Een voorbeeld is:
IF (A=B) AND (A$=B$) THEN statement
Aan de relaties A=B en A$=B$ moet tegelijk worden voldaan wil het statement worden uitgevoerd. De haakjes zijn trouwens niet alleen voor de duidelijkheid geplaatst maar ook om van de uitvoeringsvolgorde te zijn verzekerd.

Boolean variabele

Uit het bovenstaande blijkt dat Basic wel in staat is om boolean resultaten te bepalen met de relatie-operatoren en deze resultaten met de logische operatoren te bewerken. Of we nog meer met deze boolean bewerkingen kunnen doen is afhankelijk van de Basic-implementatie.
In de Microsoft Basics is een echt boolean datatype aanwezig waarmee we logische of boolean variabelen kunnen gebruiken. Een variabele heeft als eigenschap dat we er een waarde aan kunnen toekennen en later die opgeborgen waarde weer kunnen terug roepen.
Omdat het boolean datatype niet is gedefinieerd in Basic moeten we proberen of we een wel voorkomend datatype kunnen (mis)bruiken. Dat een integer of een real als boolean bruikbaar is blijkt als we het volgende programma uitproberen:

```
10 INPUT A
20 IF A THEN PRINT „TRUE"
30 END
```

Met een Microsoft Basic (PET, Apple, KIM, SYM, Challenger, TRS80 etc.) blijkt dat alleen het invoeren van het getal Ø niet het printen van TRUE ten gevolge heeft. Een real (een integer werkt ook) is dus een boolean met de waarde false als de getalwaarde Ø is, anders true.
Nu moeten we proberen of we het resultaat van een boolean bewerking kunnen toekennen aan een boolean variabele. Met behulp van het volgende programma kunnen we dit uitproberen:

```
10 INPUT A, B
20 C= ((A<B) OR (A=B))
30 PRINT "C="; C
40 IF C THEN PRINT "A<B OF A=B"
```

Als we voor A en B zodanige getallen invoeren dat A kleiner of gelijk is aan B, dan wordt geprint:

```
C= −1
A<B OF A=B
```

Als A groter dan B is wordt geprint:

```
C= Ø
```

Niet elke Microsoft Basic geeft dezelfde output, de Apple wijkt bijvoorbeeld af, waarover later meer. Uitproberen is aan te raden. De haakjes staan er weer voor de zekerheid en duidelijkheid.
Bovenstaande programma's wijzen er op dat het datatype boolean toch in Basic voorkomt en wel als subset van de real of integer.
Een boolean variabele heeft de waarde true als de getalwaarde van de als boolean gebruikte real of integer −1 is (in ieder geval ongelijk Ø) en false als de getalwaarde Ø is. Toekennen aan een boolean is mogelijk door gelijkstellen aan een relatie-expressie of eventueel een samengestelde expressie met logische operatoren. Ook de gebruikelijke real of integer toekenning is natuurlijk te gebruiken. Verder

geldt dat true (−1) kleiner is dan false (Ø).
Omdat de getalwaarde van een boolean waarde onbelangrijk is, is het verstandig in een programma, dat met boolean variabelen werkt, twee constanten in te voeren:
TRUE = −1: FALSE = Ø
In het volgende voorbeeld wordt deze constanten declaratie gebruikt.

Voorbeeld
Als voorbeeld van het gebruik van boolean variabelen kan de volgende subroutine dienen. De subroutine heeft als doel te onderzoeken of in het array RIJ de waarde GETAL voorkomt. Voor de subroutine wordt aangeroepen krijgt GETAL de te zoeken getalwaarde en na terugkeer van de subroutine is AANWEZIGGETAL, een boolean variabele, true als GETAL in RIJ voorkomt.

```
1000 REM SUBROUTINE
AANWEZIGGETAL
1010 REM ONDERZOEKT OF
GETAL IN RIJ VOORKOMT
1020 AANWEZIGGETAL =
FALSE
1030 FOR TELLER = 1 TO 10
1040 IF (RIJ(TELLER) = GETAL)
THEN AANWEZIGGETAL =
TRUE
1050 NEXT TELLER
1060 REM EINDE
AANWEZIGGETAL
```

Aanvankelijk is AANWEZIGGETAL false, als in de FOR... NEXT loop ooit het getal GETAL voorkomt is AANWEZIGGETAL true en blijft dat verder.
De subroutine wordt iets efficiënter als regel 1040 wordt vervangen door:

```
1040 AANWEZIGGETAL =
(AANWEZIGGETAL OR RIJ
(TELLER) = GETAL)
```

waarbij slechts een expressie wordt bepaald en meer de boolean mogelijkheden worden benut.
De subroutine kan als volgt worden gebruikt:

```
100 GETAL=3 : GOSUB 1000
110 IF AANWEZIGGETAL THEN
PRINT "GETAL"; GETAL; "IS
AANWEZIG"
```

Microsoft Basic laat voor variabelen namen met meerdere karakters toe, maar herkent alleen de eerste twee. Langere namen zijn duidelijker te begrijpen, zolang we de eerste twee karakters maar uniek houden.

Real relatie

Als A en B van het type real zijn kan de relatie:
IF A=B THEN....
een onzeker resultaat hebben. De floating point notatie voor een real veroorzaakt een onnauwkeurigheid. Bij het laten printen van een real wordt bovendien afgerond. De relatie A=B kan als resultaat ongelijk opleveren terwijl de getallen A en B toch bij printen hetzelfde getal te zien geven. Gelijkheid testen kan beter met:
IF ABS (A−B) <.1 E−6 THEN....
waarmee we testen of de getallen A en B zo dicht bij elkaar liggen dat we ze gelijk vinden. Kies het verschil natuurlijk niet kleiner dan de rekennauwkeurigheid (6 of 9 cijfers).

String als boolean

Ook het datatype string kan als boolean variabele worden misbruikt. False wordt dan voorgesteld door de lege string (""), elke andere string is true. Toch is de string niet geschikt als boolean variabele omdat relaties een getal en geen string opleveren.
Bruikbaar is wel de test op de lege string:
IF A$ THEN....
is equivalent met
IF A$ = "THEN"....

AND, OR en NOT

De logische operatoren AND, OR en NOT kunnen meer dan boolean bewerkingen uitvoeren. Het zijn operatoren die met 16 bits twee-complement getallen werken.
Als bijvoorbeeld A OR B wordt uitgerekend, dan worden A en B eerst omgevormd tot een integer (en dat is een 16 bits twee-complement getal intern), waarna bit voor bit de logische OR operatie wordt uitgevoerd.

Voorbeeld
6 OR 1 = (ØØØØØ11Ø OR ØØØØØØØ1) = (ØØØØØ111) = 7
(laatste 8 bit getoond).
Het omzetten naar integers vereist dat de getallen tussen −32767 en +32767 liggen, een integer voldoet daar aan, zodat integers geschikte argumenten zijn. Reals voldoen ook met de genoemde beperking.
AND en OR zijn de bekende logische bit bewerkingen, NOT is de twee-complement bewerking:
NOT X = − (X +1)
De waarde −1 voor de boolean waarde true en Ø voor false worden nu ook duidelijk:
true AND false = −1 AND Ø = Ø = false
Het getal −1 is in twee-complement 1111 1111 1111 1111 en Ø is 0000 0000 0000 0000.
Not true = NOT (−1) = −(−1 +1) = Ø = false
De andere bewerkingen zijn even eenvoudig na te gaan.

Nog een voorbeeld
De logische operatoren zijn uitermate geschikt om bitmanipulaties vanuit Basic uit te voeren. Een voorbeeld zal dit duidelijk maken.
Stel dat we van een PIA-poort bit Ø hoog willen maken zonder de andere PIA bit's te beïnvloeden. In machinetaal (6800 en 6502) gaat dit als volgt:

```
LDA PIA lees PIA output register
ORA # $Ø1 maak bit Ø hoog
STA PIA en andere bits intakt
```

Dezelfde operatie is vrijwel identiek vanuit Basic uit te voeren met

```
POKE PIA, ( PEEK (PIA) OR 1 )
```

waaruit blijkt dat Microsoft Basic uitstekend in staat is tot besturen van de computer op bit niveau.

Applesoft

Applesoft, de Apple versie van Microsoft Basic, wijkt behoorlijk af van de andere Basic-implementaties van Microsoft wat betreft de hier beschreven logische bewerkingen. True kan worden voorgesteld door het getal 1, false door Ø, zodat false kleiner dan true is.
AND, OR en NOT leveren altijd een boolean resultaat, zodat de beschreven bitmanipulaties in deze vorm niet mogelijk zijn.
Andere Microsoft Basic implementaties kunnen ook afwijken van de hier beschreven mogelijkheden. Het beste is de gegeven programmavoorbeelden na te lopen en wat te experimenteren.
In het gebruik zullen de boolean mogelijkheden van Basic in programma's een waardevolle aanvulling blijken te zijn.