

6. Variabelen & Constanten

1. Wat is een variabele

Tijdens het programmeren wordt er vaak gebruik gemaakt van variabelen. Dit zijn gereserveerde plaatsen in het geheugen waarop gegevens (getallen of cijfers) kunnen worden bewaard. Een variabele moet worden aangemaakt (declareren). Het krijgt een naam en een datatype toegewezen

2. Declareren van variabelen en constanten

Het declareren (aanmaken) van een variabele gebeurt via een declaratie statement.

Een declaratiestatement bestaat uit drie stukken:

een bereik; hiermee wordt de range van de variabele aangegeven

een naam; dit is een naam die je zelf kiest en die gebruikt zal worden om de

variabele aan te roepen.

Een type; hiermee wordt aangegeven welk type van gegevens er zullen

opgeslagen worden.

Syntax van een declaratiestatement:

[dim | public] "naam" as type

bereik, naam, type

vb dim teller as integer

Ook voor het definiëren van een constante is er een declaratie statement nodig.

public const "naam" = waarde

vb: public const pi = 3.14159

3. Bereik

Met dim of public wordt het bereik van een variabele uitgedrukt.

Met de prefix "dim" geef je aan dat je de variabele enkel lokaal wil gebruiken. Wanneer je bijvoorbeeld een declaratiestatement met een dim-prefix in een eventroutine zet, dan kan je deze enkel gebruiken binnen deze event-routine. De variabele wordt aangemaakt wanneer de event-routine start en wordt opnieuw vernietigd op het moment dat de routine uitgevoerd is.

```

Project1 - frmEerst (Code)
cmdBereken Click
Private Sub cmdBereken_Click()
' declaratie van de variabelen x en y
Dim x As Integer
Dim y As Integer
' het toekennen van een waarde aan x
x = txtBedrag.Text
' we vermenigvuldigen x met 40.3399 en stoppen het
'resultaat in de variabele y
y = x * 40.3399
'we geven de waarde van y door aan het tekstvak
txtBedrag.Text = y
End Sub

```

De prefix "public" geeft aan dat het gebruik van de variabele het lokale niveau overschrijdt. Gebruik je de "public" prefix om een variabele te definiëren in het declaratiegedeelte van de form, dan zal je deze variabele kunnen aanroepen vanuit elke event-routine van de form of van een object op die form. Je kan hem niet aanroepen vanuit een andere form. Wil je een variabele aanmaken die je doorheen het volledige programma kan gebruiken, dan moet je die aanmaken met de "public"-prefix in het declaratiegedeelte van een module.

```

Project1 - Module1 (Code)
(General) (Declarations)
Option Explicit
' aanmaak van een variabele
Public varTeller As Integer

```

Tenslotte kan je een variabele ook definiëren binnen een procedure of een functie. Dan gebruik je de "dim"-prefix. De variabele is dan uitsluitend binnen die procedure of routine bruikbaar.

| Aanmaak | Statement | bereik |
|--|-----------|---------------------------|
| Binnen de event-routine | Dim | event-routine |
| General gedeelte form, declaratiegedeelte | Public | form |
| General gedeelte form in een algemene procedure of functie | dim | alg. procedure of functie |
| Modulenniveau, declaratiegedeelte | public | project |
| Modulenniveau in een algemene procedure of functie | dim | alg. procedure of functie |

Een constante wordt gedeclareerd in het declaratiegedeelte van de form of module. Een constante, gedeclareerd op formniveau kan enkel worden aangeroepen vanuit event-routines binnen het formbestand. Constanten gedeclareerd op modulenniveau kunnen in heel het project gebruikt worden.

| Aanmaak | Statement | bereik |
|---|--------------|---------|
| General gedeelte form, declaratiegedeelte | Public const | form |
| Modulenniveau, declaratiegedeelte | Public const | project |

4. Naam van een variabele

De naam van een variabele moet voldoen aan de volgende voorwaarden:

- begint met een letter
- bevat enkel letters, cijfers en
- is niet langer dan 40 posities
- mag geen gereserveerd woord zijn (vb. run, end,...)

5. Datatypes

Een pc werkt met verschillende soorten geheugenregisters (chips waar hij gegevens op bewaart). Een stuk tekst wordt anders opgeslagen als een geheel getal, een geheel getal wordt anders opgeslagen als een getal met een komma, etc. Daarom is het nodig om op voorhand aan te geven welke soort gegevens je in de variabele wil stoppen. Dit doe je via het datatype.

Een voorbeeld:

Dim teller as integer

De term integer wijst erop dat in deze variabele gehele getallen kunnen bewaard worden die liggen tussen -32.768 en +32.768.

Mijdt het gebruik van het type variant. In de eerst plaats omdat dit type zeer veel geheugenruimte inneemt, waardoor geheugen niet optimaal wordt gebruikt. Ten tweede omdat dit problemen kan opleveren bij het programmeren omdat de programmeur niet op voorhand weet welk soort gegevens de variabele bevat.

6. Tabellen

Declareren van tabellen

Tot nu toe hebben we enkel variabelen bekeken die maar één waarde op hetzelfde moment kunnen bevatten (1 getal, 1 zin etc). Nu is het ook mogelijk om een variabele te definiëren die meerdere waarden tegelijk kan bevatten. In dat geval spreken we van een tabel.

Het aanmaken van een tabel doe je door aan een variabele een dimensie toe te kennen. Deze dimensie geeft aan hoeveel verschillende waarden je kan opslaan met deze variabele. Merk op dat men start met het cijfer 0.

Vb. Public Cijfers(24) As integer Tabel "Cijfers" kan 25 integers bevatten

Een tabel wordt beschouwd als een gewone variabele. Alle regels van een gewone variabele gelden dus ook voor een tabel.

Refereren naar een element in de tabel

Wil je naar een bepaald element uit een tabel verwijzen of wil je een waarde opslaan in een veld van de tabel, dan typ je de naam van de tabel met daarna, tussen ronde haakjes, de index van het bedoelde element.

Vb. Naam(6) = "Bart"

In dit voorbeeld kennen we aan het zesde element van tabel "naam" de waarde "Bart" toe. Let wel, de tabel "naam" moet natuurlijk gedefinieerd worden voor hij kan gebruikt worden.

Multi-dimensionele tabellen

In Visual Basic kunnen multi-dimensionele tabellen worden aangemaakt (max. 60 dimensies). Een multi-dimensionele tabel aanmaken gebeurt op gelijkaardige wijze als het aanmaken van een 1-dim tabel, met dit verschil dat, tussen de ronde haken, verschillende cijfers terug te vinden zijn. Elk cijfer stelt een dimensie voor en geeft de omvang aan van het aantal elementen dat bewaard wordt. Vb. Public names(3,2) As string

Dit is een tabel met vier rijen (0 - 3) en drie (0 - 2) kolommen, waarin strings kunnen bewaard worden.

7. Controle Structuren

Tot nu toe werden alle commando's in de programmacode eenmalig en sequentieel uitgevoerd. Soms is het nodig dat commando's meerdere malen na elkaar uitgevoerd worden of dat commando's slechts uitgevoerd worden indien aan bepaalde voorwaarden is voldaan. Hiervoor moet er ingegrepen worden in de afwikkeling van de programmacode. Dit gebeurt door middel van controle structuren. Hierin onderscheiden we twee soorten, namelijk beslissingsstructuren en lussen. Het eerste wordt gebruikt om voorwaarden te testen, het tweede wordt gebruikt om commando's te herhalen.

1. Beslissingsstructuren

If - Then

Syntax

```
        If conditie then
    programmacode
        End If
```

Voorbeeld

```
        If winstmarge > 10 then
    print "verkopen!"
        End If
```

Gebruik

De programmacode tussen If en End if wordt enkel uitgevoerd indien aan de conditie voldaan is. In dit voorbeeld verschijnt de tekst "verkopen!" op het scherm wanneer de waarde van de variabele "winstmarge" groter is dan 10.

If - Then - Else

Syntax

```
        If conditie 1 then
    programmacode
        ElseIf conditie2 then
    programmacode
        ElseIf
    ...
        ElseIf conditie x -1 then
            programmacode
        Else
            programmacode
```

End if

Voorbeeld

```
        If Winst > 10 then
```

```

print "verkopen!"
  Elseif 10 >= Winst and 5 < Winst then
print "wachten"
  Elseif 5 > = Winst and 0 < Winst then
print "wachten"
  Else
    print "in prullenmand"
  End If

```

Gebruik

If - Then - Else wordt gebruikt om meerdere condities te testen. De condities worden door het programma sequentieel doorlopen tot er een conditie wordt gevonden waaraan voldaan is. De bijhorende code wordt uitgevoerd.

Tracht de condities zo op te stellen dat er nooit aan meer dan één conditie tegelijk voldaan is. Indien er aan meerdere condities voldaan is, zal het programma enkel de eerste daarvan beschouwen en de andere overslaan.

Select case

Syntax

```

Select case testexpressie
case expressionlist 1
  programmacode
case expressionlist2
  programmacode
case else
  programmacode
End Select

```

Voorbeeld

```

Select case naam
case "Milis"
  print "Koen"
case "jacobs"
  print "Bart"
case else
  print "Lieve"
End Select

```

Gebruik

Een case-structuur wordt, net zoals een if-then-else structuur, gebruikt om een expressie te testen. Er wordt een testexpressie opgesteld. Dan wordt gezocht naar een casewaarde die overeenkomt met het resultaat van de testexpressie. De overeenkomstige programmacode wordt uitgevoerd.

In het voorbeeld fungeert de variabele "naam" als testexpressie. Nu wordt er op zoek gegaan naar een casewaarde (een achternaam) die overeenkomt met de naam die opgeslagen is in de variabele. Is de variabele en de casewaarde gelijk, dan wordt de bijbehorende code uitgevoerd.

In principe kan je if-then-else en case structuren door elkaar vervangen.

2. Lusstructuren

Do loop

Syntax

```

1. Do while <conditie>
  programmacode

```

```
loop
2. Do until <conditie>
   programmacode
```

```
Loop
3. Do
   programmacode
Loop until <conditie>
4. Do
   programmacode
Loop while <conditie>
```

Voorbeeld

```
Do while x < 100 print x
  x = x + 1
```

Loop

Gebruik

De do.. loop structuur wordt gebruikt om commando's een aantal maal na elkaar te laten uitvoeren. Dit is een conditionele lus. Met andere woorden: de lus bevat een testexpressie die aangeeft of de lus al dan niet moet beëindigd worden. In het voorbeeld wordt de lus herhaald zolang x kleiner is dan 100. Onder syntax worden een aantal varianten van de do loop beschreven. Het is de bedoeling dat de student hiermee zelf experimenteert en de verschillen tussen de varianten achterhaalt. Let op: wanneer je een fout maakt in de testexpressie kan het gebeuren dat je in een oneindige lus terecht komt (er kan bijvoorbeeld nooit aan de expressie voldaan worden). Een oneindige lus kan je enkel doorbreken door Visual Basic af te sluiten met ctrl-alt-del. Je verliest alle informatie die nog niet was opgeslagen.

For next

Syntax

```
For Teller = Getal to Getal
  programmacode
next Teller
```

Voorbeeld

```
Dim Getal as integer
For Getal = 1 to 100
  print "hallo"
next Getal
```

Gebruik

For. . . next wordt eveneens gebruikt om instructies een aantal maal te herhalen. Het verschil met een do loop is dat het aantal lussen al op voorhand bepaald is. Om een for... next lus te gebruiken moet je eerst een variabele definiëren. In het voorbeeld is dat de variabele "Getal". Deze variabele fungeert als teller. De eerste maal dat de lus doorlopen wordt krijgt deze variabele de waarde van het eerste getal (1). Elke keer dat de loop wordt uitgevoerd, wordt de teller standaard verhoogd met 1 eenheid. Dit gaat door tot de teller een waarde heeft bereikt die gelijk of groter is dan het tweede getal (100).

De teller kan ook aftellen of sprongen nemen die groter of kleiner zijn dan 1.

Vb. For Getal = 2.4 to 1.6 step -0.2

```
print "hallo"
```

```
next getal
```

Test uit.

3. Oefeningen

Schrijf een programma voor toegangscontrole. De Naam het Nummer en het Paswoord van de aanvrager moeten bij het indrukken van OK vergeleken worden met een interne lijst in de vorm van een Array 'Toegangscontrole (x,y,z)'. Bij het drukken op Nieuw wordt eerst het paswoord van de beheerder gevraagd om na goedkeuring de gegevens aan de lijst toe te voegen.

