HOOFDSTUK 3

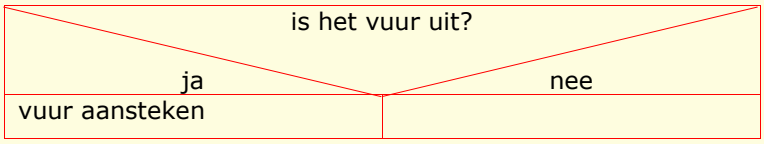
Imperatief programmeren

3.1 Stapsgewijs programmeren

In de vorige hoofdstukken zijn programmeertalen beschreven die imperatief zijn. Imperatief programmeren is het stapsgewijs in code omschrijven wat een programma moet doen, net als een algoritme of een PSD. Je kunt imperatief programmeren ook procedureel programmeren noemen.

3.2 If Then Else

Als een PSD selecties bevat, deelt de lijn van het programma zich op met de verschillende antwoorden op het vraagstuk.



Als je imperatief programmeert, heb je veel met zulke selecties en voorwaarden te maken. Hier gebruik je de 'if then'-statements en de 'else'-statements voor.

Voorbeeld: toegangsprijs

Voor het bepalen van de toegangsprijs van het pretpark kun je de volgende procedure opstellen.

De code wordt sequentieel, van boven naar beneden, afgewerkt.
Je ziet dat er allerlei keuzes gemaakt moeten worden.

Het resultaat is dat mensen jonger dan 12 jaar en ouder dan, of net zo oud als, 60 voor de toegang 5 euro betalen.
De overige leeftijden betalen 10 euro.

Voorbeeld

```
ALS leeftijd < 12
DAN
  toegang := 5 euro
ANDERS
  ALS leeftijd >= 60
  DAN
    toegang := 5 euro
  ANDERS
    toegang := 10 euro
  EINDE-ALS
EINDE-ALS
```

Niet elke If heeft een Else, maar wel elke Else heeft een If

Bij een if-statement hoort niet altijd een else-statement. Andersom is dit wel zo. Elke else-statement heeft een if-statement.

Informatie

Alle operaties binnen een statement staan iets naar rechts. Dit noem je inspringen. Een programma werkt meestal ook als je dit inspringen niet doet, maar het is wel belangrijk om te doen. Op die manier is de code overzichtelijk.

3.3 Lus

Als een programma een iteratie bevat, kun je zeggen dat het een lus bevat (in het Engels: loop). De operaties die in een lus zitten, worden uitgevoerd zolang er aan een bepaalde voorwaarde wordt voldaan. De bekendste is de while-lus. Die kun je aangeven met 'ZOLANG'.

In het voorbeeld wordt er een getal net zolang opgebouwd, totdat deze niet meer kleiner is dan 3.
De lus stopt zodra het getal 3 is geworden.

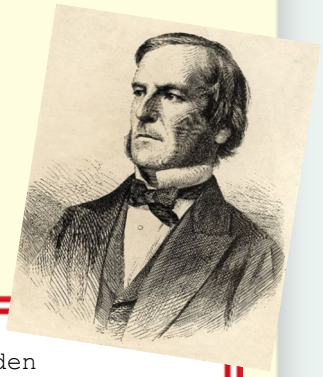
Voorbeeld

```
getal := 0
ZOLANG getal < 3
  getal := getal + 1
EINDE-ZOLANG
```

Oneindige lus

Als de voorwaarde voor een while-lus nooit onwaar wordt, zal de lus niet vanzelf stoppen. Het programma blijft dan hangen. Dit noem je een 'oneindige lus'. Dit is niet altijd de bedoeling. De programmeur kan een fout hebben gemaakt in de voorwaarde of ergens anders.

3.4 Booleaanse expressies



In de meeste imperatieve programma's vind je Booleaanse expressies.

Informatie

Een Booleaanse expressie is een expressie die de waarden true of false aan kan nemen.

De naam Booleaans komt van de wiskundige en filosoof George Boole. Hij introduceerde in de 19e eeuw het redeneren in waarden als true en false.

De voorwaarde van een if-statement is een Booleaanse expressie. Is de waarde true, dan wordt de onderliggende code uitgevoerd en niet de code onder het else-statement. Heeft de Booleaanse expressie de waarde false, dan geldt het andersom.

Symbolen Booleaanse logica

Booleaanse expressies van programmeertalen kun je weergeven in symbolen.

Symbool	Naam	Gebruik
>	groter-dan	x > y is true als x groter is dan y
>=	groter-dan-of-gelijk-aan	x >= y is true als x groter is dan y of net zo groot
<	kleiner-dan	x < y is true als x kleiner is dan y
<=	kleiner-dan-of-gelijk-aan	x <= y is true als x kleiner is dan y of net zo groot
not	not	not x is true als x en y beide true zijn
and	and	x and y is true als x en y beide true zijn
or	or	x or y is true als x of y true is (of als ze allebei true zijn)
xor	exclusive or	x xor y is true als x of y true is (en niet als ze allebei true zijn)

Door het gebruik van symbolen, kun je de code van de toegangsprijs van het pretpark korter omschrijven.

Voorbeeld

```
ALS leeftijd < 12 OF leeftijd >= 60
DAN
  toegang := 5 euro
ANDERS
  toegang := 10 euro
EINDE-ALS
```

Wijze van weergeven

Je moet bij het opstellen van een Booleaanse expressie zorgvuldig werken. Een verschil in de volgorde van de expressie kan andere resultaten opleveren. Voorzie daarom delen van de expressie van haakjes, zoals (en). De delen tussen haakjes moet je als geheel interpreteren.

Waarheidstabel

De expressie:

(x EN y) OF z

Is niet hetzelfde als:

x EN (y OF z)

In een waarheidstabel geef je aan wanneer de expressie true of false is bij welke waarden. Beide expressies staan in verschillende waarheidstabellen.

(x EN y) OF z			
x	y	z	Waarde van expressie
true	true	true	**true**
true	true	false	**true**
true	false	true	**true**
true	false	false	**false**
false	true	true	**true**
false	true	false	**false**
false	false	true	**true**
false	false	false	**false**

x EN (y OF z)			
x	y	z	Waarde van expressie
true	true	true	**true**
true	true	false	**true**
true	false	true	**true**
true	false	false	**false**
false	true	true	**false**
false	true	false	**false**
false	false	true	**false**
false	false	false	**false**

Informatie

Meestal worden in waarheidstabellen de waarden true en false vervangen door 0 en 1. Dit maakt de tabel beknopter.

3.5 Variabelen

Een variabele is een element dat een bepaalde waarde aan kan nemen. Zoals in het voorbeeld van de toegangsprijs, is de toegangsprijs de variabele. De waarde die de variabele kan hebben, hangt af van het type van de variabele. Er komen verschillende typen voor in programmeertalen. De volgende typen zijn in ieder geval aanwezig:

- **Geheel getal**
  Dit noem je een **integer**. Je kort het vaak af met int. '42' is een heel getal.

- **Kommagetal**
  Een kommagetal noem je een **double** of **float**. '109,75' is een kommagetal. In veel programmeertalen moet je deze getallen op z'n Engels noteren met een decimale punt, '109.75'.

- **Karakter**
  Een karakter noem je een **character** of **char** (spreek je uit als kar). Een karakter kan elk ANSI-, EBCDIC- of Unicode-teken zijn, bijvoorbeeld $.

- **Boolean**
  Dit noem je een **bool** (in het Nederlands een Booleaanse). Een boolean kan alleen maar true of false zijn. Sommige programmeertalen (C en C++) laten ook gehele getallen toe. Hier staat 0 voor false en alle andere getallen voor true.

- **String**
  De string is een apart soort variabele. Dit is een reeks karakters, bijvoorbeeld 'tijger' of '$2,50'.

3.6 Arrays

In de meeste programmeertalen kun je arrays (Nederlands: rijen) maken. Een array is een geordende rij van variabelen van hetzelfde type. Je kunt bijvoorbeeld een rij van gehele getallen maken.

De variabelen van een array zijn genummerd. Het volgnummer noem je de index. In veel programmeertalen begint de indexering van een array met een 0 en niet met een 1.

Voorbeeld

De code:

```
rij_van_getallen := [32, 59, 825, 9042]
```

De indexering:

rij_van_getallen				
Index:	0	1	2	3
Waarde:	32	59	825	9042

3.7 Functies

Je gebruikt een functie (procedure) als:

- je een bepaald stuk code vaker moet gebruiken
- je een programma overzichtelijker wilt maken.

Met een functie maak je een subprogramma. Als je in het programma de naam van een functie zet, wordt het subprogramma dat erbij hoort, uitgevoerd. Je noemt dit het aanroepen van de functie.

De functie bevat de volgende begrippen:

- **De naam**
  Elke functie moet een unieke naam hebben. In sommige programmeertalen kunnen functies dezelfde naam hebben, maar dan moeten de parameters verschillen. Je noemt dat overloading.

- **De parameters**
  Parameters zijn speciale variabelen die een waarde krijgen bij het aanroepen van de functie. De functie kan dan werken met de meegegeven waarden. Een functie van nul of meer parameters hebben.

- **Het returntype**
  Na het uitvoeren van de functie, krijg je het returntype. Een functie kan maar één returntype hebben. De waarde wordt op de plek van de aanroep verder gebruikt.

- **De body**
  De body is de code die wordt uitgevoerd als de functie wordt aangeroepen.

Voorbeeld

In het volgende voorbeeld zie je een variabele die is gespecificeerd.

Voorbeeld

```
som(int getal1, int getal2) : int
  int optelling := getal1 +getal2
  return optelling
```

De naam van de functie is 'som'. Deze telt twee getallen van het type 'int' bij elkaar op. De functie geeft het resultaat ('optelling') terug als 'int'.

De termen 'getal1' en 'getal2' zijn de parameters. Ze worden gescheiden door een komma. Je kunt deze functie aanroepen en het resultaat in een andere variabele stoppen.

Voorbeeld

```
int getalA := som(10,20)
int getalB := som(getalA, 70)
int getalC := som(som50, 50), getalB)
```

Als je bovenstaande code zou uitvoeren, krijg je de volgende resultaten:

getalA = 30
getalB = 100
getalC = 200

Het resultaat kan gebruikt worden als het returntype van de functie. Het returntype van de functie is een integer en kun je dus direct gebruiken.

3.8 Parameters en argumenten

Getal1 en getal2 zijn de **parameters** van de functie 'som'. Als je deze parameters aanroept, worden ze ingevuld door de argumenten. Een argument is een waarde die wordt meegegeven bij een functie-aanroep.

Voorbeeld

```
int getalA := som(10,20)
int getalB := som(getalA, 70)
int getalC := som(som50, 50), getalB)
```

Bij het aanroepen van een functie wordt de waarde van de variabele meegeven. De variabele wordt niet meegestuurd. De variabele verandert dus niet na het aanroepen van een methode.

Voorbeeld

Hieronder zie je de definitie van de functie:

Voorbeeld

```
zomaarEenFunctie(int eenGetal) : int
  eenGetal :=999
  return 0
```

Je roept deze functie aan:

Voorbeeld

```
mijnEersteGetal := 5
mijnTweedeGetal :=
zomaarEenFunctie(mijnEersteGetal)
```

Als je de code hebt uitgevoerd, zal mijnEersteGetal nog steeds de waarde 5 hebben en mijnTweedeGetal de waarde 0.

3.9 Void

Een functie heeft niet altijd een returnwaarde. Als dit niet zo is, geef je dit aan met het returntype void. Void is geen variabele-type, maar betekent de afwezigheid van een waarde. Void is een Engels woord en betekent 'leegte'.

Voorbeeld

```
zegHallo() : void
  print("hallo")
```

De functie in het voorbeeld hierboven heeft geen parameters en geen returnwaarde. De functie met de naam 'print' wordt aangeroepen. Hier wordt een string als argument meegegeven.

3.10 Syntax

De syntax (Nederlands: syntaxis) is de grammatica van een programmeertaal. De syntax beschrijft hoe de code er in een bepaalde programmeertaal uit moet komen te zien. Hij beschrijft wat de grammaticale regels zijn. Elke programmeertaal heeft een eigen syntax, maar sommigen hebben wel overeenkomsten, zoals C++, Java en C#. De overeenkomsten zorgen ervoor dat het voor programmeurs makkelijker is een andere taal te leren.

Voorbeeld

```
int som(int getal1, int getal2)
{
  return getal1 + getal2;
}
```

Voorbeeld van een functie die zou werken in C, C++, Java en C#

3.11 Recursie

Wanneer een patroon zich binnen zichzelf herhaalt is er sprake van recursie. Een voorbeeld van recursie in een programma is:

Voorbeeld

```
mijnFunctie() : void
  mijnFunctie();
```

Binnen het programmeervak spreekt men van recursie als de body van een functie een aanroep van deze zelfde functie bevat, met andere woorden: als een functie zichzelf aanroept.

Dit lijkt vreemd, maar sommige programmeerproblemen zijn alleen met recursie op te lossen.

Recursieve functies moeten met zorg geschreven worden. Als de programmeur niet goed oplet bij het schrijven van een recursieve functie, blijft de functie zichzelf oneindig aanroepen en loopt het programma vast of crasht het doordat het geheugen van de computer volgelopen is (overflow). Om te voorkomen dat recursie eindeloos (tot aan een crash) door blijft gaan, moet de programmeur de functie voorzien van een stopconditie. Dit is een punt waarmee het recursieve proces tot stoppen gebracht wordt.

3.12 Functioneel programmeren

De techniek van recursie is zo universeel en krachtig dat er een hele familie van programmeertalen rond dit concept bestaat. Dit zijn de zogenaamde functionele programmeertalen.

In een functionele programmeertaal heeft een variabele, net als in de wiskunde, altijd dezelfde waarde. Je hebt dus geen assignments (toekenningen) zoals x := x + 1 .

Variabelen, veelal parameters, krijgen een waarde bij aanroep van de functie waar ze in voor komen. Daarom bestaan deze programma's uit veel aanroepen, vaak recursief, van relatief kleine functies. Deze functies lijken erg op wiskundige functies en zijn daardoor ook goed wiskundig te beschrijven en te analyseren.

Programmeren in een functionele programmeertaal is niet zo makkelijk te leren. Maar een ervaren programmeren kan er snel en effectief mee programmeren. Voorbeelden van functionele programmeertalen zijn Lisp, Haskell, Clojure.

Voorbeeld van een stukje Haskell:

Voorbeeld

```
som :: [Integer] -> Integer       -- som is een functie van een
lijst integers naar een integer

som [] = 0                        -- de som van de lege lijst is 0

som (x:xs) = x + som xs           -- de som van een niet-lege lijst
is het eerste element plus
```

Deze functie telt de elementen van een lijst op.

3.13 Vragen en opdrachten

3.13.1 Open vragen

1. Wanneer zal van een else-statement de inhoud worden uitgevoerd?
   Geef de waarheidstabel van de volgende expressie: x OR (NOT y).
   a. Noem de Engelse namen van de verschillende typen variabelen die je kent.
   b. Wat valt je op als je de string met de lijst vergelijkt?

2. a. Welke eigenschappen moet je van een functie of procedure definiëren?
   b. Wat is het verschil tussen een parameter en een argument?

3. Leg in je eigen woorden uit wat syntax betekent.

3.13.2 Meerkeuzevragen

1. Welke van onderstaande beweringen over lijsten is niet waar?
   a. De eerste waarde van een lijst staat altijd op index 0.
   b. Een lijst heeft een volgorde waarin de waardes gerangschikt zijn.
   c. In één lijst kunnen verschillende waardes worden opgeslagen van verschillende typen.
   d. Een lijst wordt ook wel array genoemd.

2. Bekijk dit voorbeeld:

```
int a = 2;
int b = 4;
int c = 6;
vermenigvuldig(int x, int y) : int
return x * y
c := vermenigvuldig(a, b)
```

   Stel dat deze code uitgevoerd zou worden, welke waardes zouden dan correct zijn ná het uitvoeren:
   a. x is 2, y is 4 en c is 6.
   b. a is 2, b is 4 en c is 8.
   c. b is 4, y is 4 en c is 4.
   d. a is 2, b is 4 en c is 6.

3. Welke van onderstaande beweringen over lussen is niet waar?
   a. Van een lus wordt de booleaanse expressie herhaaldelijk gecontroleerd.
   b. Een lus is vergelijkbaar met de iteratie van een PSD.
   c. Een ander woord voor lus is loop.
   d. Bij een oneindige lus wordt niet aan de booleaanse expressie voldaan.

4. Van de volgende booleaanse expressie zijn de operatoren vervangen door cijfers. De waarden voor x en y zijn: x = true, y = false. Welke operatoren kunnen hier staan om de expressie de waarde true te laten zijn?

   x 1 y 2 3 (x 4 y)

   a. 1 = XOR, 2 = AND, 3 = OR, 4 = XOR
   b. 1 = OR, 2 = AND, 3 = NOT 4 = AND
   c. 1 = AND, 2 = AND, 3 = NOT, 4 = AND
   d. 1 = OR, 2 = XOR, 3 = NOT, 4 = AND

3.13.3 Korte opdrachten

1. Zoek ten minste vijf syntactische regels voor de programmeertaal Java en beschrijf deze kort in je eigen woorden.

2. a. Schrijf zelf code voor het zetten van koffie. Maak hierbij gebruik van variabelen, ten minste één lus en ten minste één functie.
   b. Beschrijf vervolgens stapsgewijs de werking van jouw code. Geef hierbij onder andere aan welke variabelen op welk moment welke waarde hebben.

3.14 Samenvatting

- Imperatief programmeren is het stapsgewijs schrijven van instructies die van boven naar beneden uitgevoerd worden. Kenmerken hierbij zijn:
  - if-, then-, else-statemens: gebruikt om voorwaarden aan variabelen te stellen
  - lussen: herhalingen, denk hierbij aan de while-lus
  - functies of procedures: kleine deelprogramma's om het programma overzichtelijker te maken en herhalende code slechts één keer te hoeven schrijven.

- Een parameter is een variabele die gedefinieerd is bij een functie. Een argument is de waarde die je meegeeft bij het aanroepen van een functie.

- Er zijn verschillende typen variabelen:
  - int: heel getal
  - double of float: kommagetal
  - char: een karakterteken
  - bool of boolean: heeft de waarde true of false.

- Een string is een reeks van karakters.

- Een lijst (of array) is een reeks van waardes van één type variabele.

- De syntax van een programmeertaal beschrijft de grammaticaregels waaraan een programmeur zich voor die taal moet houden.