OVERZICHT VAN HET PROGRAMMEREN

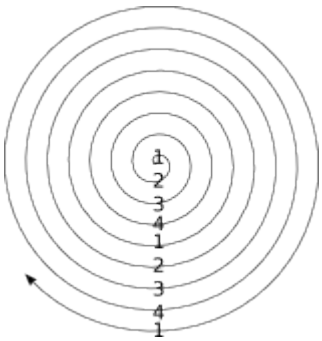
Inleiding

Als je leert programmeren lijkt het nogal overweldigend om die eerste stappen te doorworstelen. Er zijn dan ook heel wat programmeertalen (Java, Ruby, Python, Perl, PHP, Pascal, ASP, Javascript, C, C++, C#, ASP.NET, VB, VBA, StarBasic, COBOL,...), waardoor je door de bomen het bos misschien niet meer ziet.

Dit document wil een overzicht geven van de meest algemene stappen die je in bijna elke programmeertaal tegenkomt. Immers, de programmeertaal die je aanleert is misschien niet de programmeertaal die je later in de praktijk zal gebruiken. Het is niet de bedoeling om hier alles te behandelen: er wordt van het VVKSO-leerplan D/2005/0279/049 uitgegaan, d.i. een leerplan informatica voor de tweede graad TSO.

Ter informatie: dit document valt onder het publiek domein, m.a.w. (pd) i.p.v. ©[1].

Het spiraalmodel



1. Probleemstelling lezen/begrijpen
2. Ontwerpen
3. Implementeren
4. Controle / Evaluatie

Wanneer je programmeert is een oplossing niet zomaar van de ene op de andere dag klaar, zo merk je dat de kantoorapplicaties (zoals een tekstverwerker) van vroeger nog steeds dezelfde basis hebben, maar dat er toch ook heel wat zaken veranderd/bijgekomen zijn.

De programmeertaal

COMMENTAAR

Een van de belangrijkste dingen bij het programmeren is commentaar. Commentaar wordt gebruikt om aan te geven wat een stuk code doet en maakt je code stukken duidelijker. Het lijkt alsof dat alleen maar tijd en energie kost. Als je echter na een tijdje jouw code opnieuw bekijkt of als iemand anders jouw code moet bekijken, dan kan die commentaar een heel welgekomen hulp zijn (zeker bij miljoenen lijnen code).

Vb. in VBA: `'vervolgens de berekening van het totale bedrag, incl. BTW:`

Vb. in PHP: `// vervolgens de berekening van het totale bedrag, incl. BTW:`

VARIABELEN

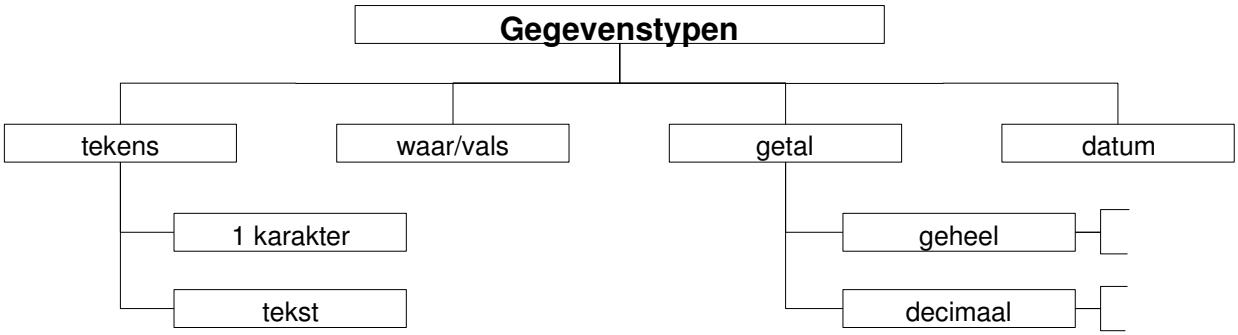
Soms heb je een bepaalde (berekende) waarde doorheen de programmacode meer dan één keer nodig. In dat geval kun je – i.p.v. die waarde steeds opnieuw te berekenen – de waarde opslaan in een *variabele* en dan gewoon de naam van de variabele gebruiken. Bij een complexe berekening kan je d.m.v. variabelen deze berekening opsplitsen in tussenstappen, waarvan je elk tussenresultaat kan

1 zie http://creativecommons.org/licenses/publicdomain voor meer informatie

opslaan in een aparte variabele. Een variabele is dus een benoemde geheugenlocatie waar gegevens opgeslagen kunnen worden.

Gegevenstypen

Wanneer de computer werkt met gegevens, dan is het belangrijk dat hij weet met welk soort gegevens hij werkt, *tekst* is immers niet hetzelfde als een *getal*, met tekst kan je bijvoorbeeld niet gaan rekenen. Elk gegevenstype neemt een bepaalde plaats in in het geheugen. Op de volgende pagina wordt een overzicht gegeven van vaak voorkomende gegevenstypen.



Programmeertaal		VBA			Java	
Type		**Type**	**Grootte**		**Type**	**Grootte**
Geheel getal		Integer	2 bytes		int	4 bytes
Decimaal getal		Single	4 bytes		float	4 bytes
Waar/Vals		Boolean	2 bytes		boolean	1 bit
Tekst		String			String	

Declaratie

Belangrijk bij het *declareren* van een variabele is het opgeven van een *naam* en het *gegevenstype*[2].

Vb. in VBA: `Dim sngSom As Single`
Vb. in Java: `int nummer;`

Initialisatie

Een variabele krijgt een waarde door deze te *initialiseren*. Na het uitvoeren van onderstaande opdracht heeft de variabele een inhoud gekregen bepaald door de uitdrukking rechts.

Vb. in VBA: `sngSom = 3`
Vb. in Java: `nummer = 3;`

Merk op dat als je tekst wilt voorstellen, dit in de meeste programmeertalen moet omgeven zijn door aanhalingstekens (“”).

CONSTANTEN

Bepaalde waarden blijven gedurende gans het programma – onafhankelijk van wanneer of hoe het uitgevoerd wordt – dezelfde. Zo is het getal π een constante, alsook de BTW op voedselproducten (nl. 6%). Als je in programmeren gebruikt maakt van een constante, dan kan je de waarde niet meer veranderen (op die manier worden fouten vermeden). Dezelfde gegevenstypes zijn hier beschikbaar als bij variabelen.

Vb. in VBA: `Const conPi As Single = 3.14159265`
Vb. in PHP: `define ("PI", "3.14159265");`

2 Bij sommige programmeertalen is het niet nodig om een gegevenstype op te geven.

BEWERKINGEN, UITDRUKKINGEN, OPERATOREN

Toekenningsoperator

Met de toepassingsoperator (vaak voorgesteld door het gelijkheidsteken =) kan je

- (nieuwe) waarden toekennen aan variabelen;
- (nieuwe) waarden toekennen aan objecteigenschappen;

Rekenkundige operatoren

De eerste belangrijke uitdrukkingen zijn de rekenkundige operatoren: hoe kan je getallen (of de variabelen waar ze in zitten opgeslagen) **optellen, aftrekken, delen** of **vermenigvuldigen**? In heel wat programmeertalen wordt dit gedaan met respectievelijk **`+`, `-`, `/` en `*`**.

Relationele operatoren

Met de relationele operatoren kan je relaties tussen waarden of variabelen testen en als uitkomst geven ze steeds 'waar' of 'onwaar'. Zo heb je groter dan (`>`), groter dan of gelijk aan (`>=`), kleiner dan (`<`) en kleiner dan of gelijk aan (`<=`).

Andere belangrijke relationele operators zijn gelijk aan en niet gelijk aan, maar de syntaxis kan per taal anders zijn. Zo wordt in VBA gebruik gemaakt van respectievelijk `=` en `<>` en in PHP en Java van `==` en `!=`.

Logische operatoren

Soms moeten meerdere voorwaarden voldaan zijn: dat kan door het werken met '*en*' of met '*of*': in heel wat talen wordt dat voorgesteld door '`and`' en '`or`'. Ook belangrijk is het "tegenovergestelde" van een voorwaarde verkrijgen, wat je in Java met `!` doet en in VBA met `Not`.

Voorwaarde A	*Voorwaarde B*	*A and B*	*A or B*	*Not A*
onwaar	onwaar	onwaar	onwaar	waar
onwaar	waar	onwaar	waar	waar
waar	onwaar	onwaar	waar	onwaar
waar	waar	waar	waar	onwaar

Samenvoegen van tekst

Vaak gebeurt het dat je twee of meerdere stukken tekst moet samenvoegen. Het lijkt logisch om daarvoor een plusteken (+) te gebruiken, maar in de meeste programmeertalen wordt hiervoor een `&`-teken gebruikt.

FUNCTIES

Om programmeerwerk te besparen zijn vaak gebruikte functies ingebouwd in de programmeertaal. Zo'n functie wordt opgeroepen door het opgeven van zijn naam en nul, één of meerdere argumenten. Vaak worden deze argumenten meegegeven tussen haakjes.

In de meeste programmeertalen heb je volgende soorten van functies:

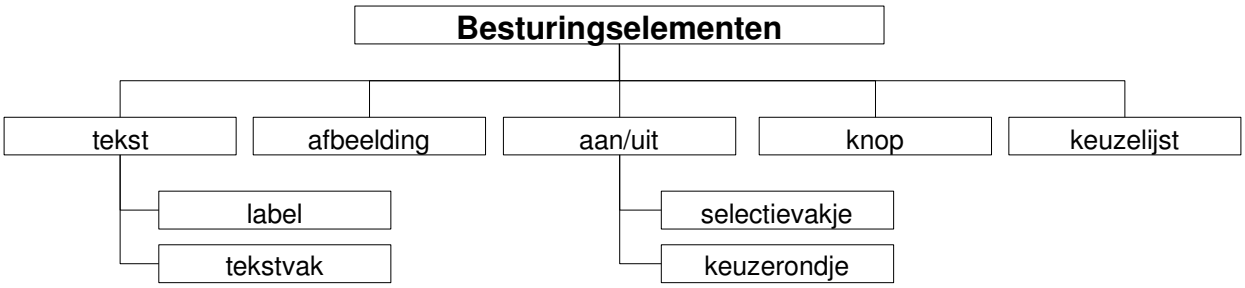
- **tekst**functies (deel van een tekst, alles naar hoofdletters,...)
- **datum**functies (datum van vandaag, de maand uit een datum halen,...)
- **wiskundige** functies (vierkantswortel, sinus,...).

Vb. tekstfunctie in VBA: `strInitialen = Mid("President J.F. Kennedy", 11, 4)`
=> `strInitialen` bevat "J.F."

Vb. tekstfunctie in PHP: `initalen = substr("President J.F. Kennedy", 10, 4)`
=> `initalen` bevat "J.F."

BESTURINGSELEMENTEN

Als je werkt met grafische gebruikersinterfaces (GUI's), dan komen daar vlug knoppen en tekstvakken aan te pas. Deze worden besturingselementen (*widgets*) genoemd. Hieronder zie je een overzicht van mogelijke en vaak voorkomende besturingselementen.



Belangrijk om te weten is dat zo'n besturingselement vaak gebruikt wordt om een bepaalde *gebeurtenis* te veroorzaken: dit wordt later besproken.

EIGENSCHAPPEN, METHODES EN GEBEURTENISSEN

Eigenschappen en Methodes

- Eigenschappen (*properties*) definiëren de toestand van objecten.
- Een methode (*method*) zorgt ervoor dat het object een actie ondergaat of onderneemt. Zo kan via een methode een eigenschap worden ingevuld (het object ondergaat een actie) of kan een een eigenschap worden doorgegeven aan bv. een variabele (het object onderneemt een actie).

Vb. eigenschap in VBA: `txtBoodschap.Visible = False`
Vb. methode in Java: `lengte = tekst.length();`
Vb. methode in VB.NET: `txtLogin.Focus`

Gebeurtenissen

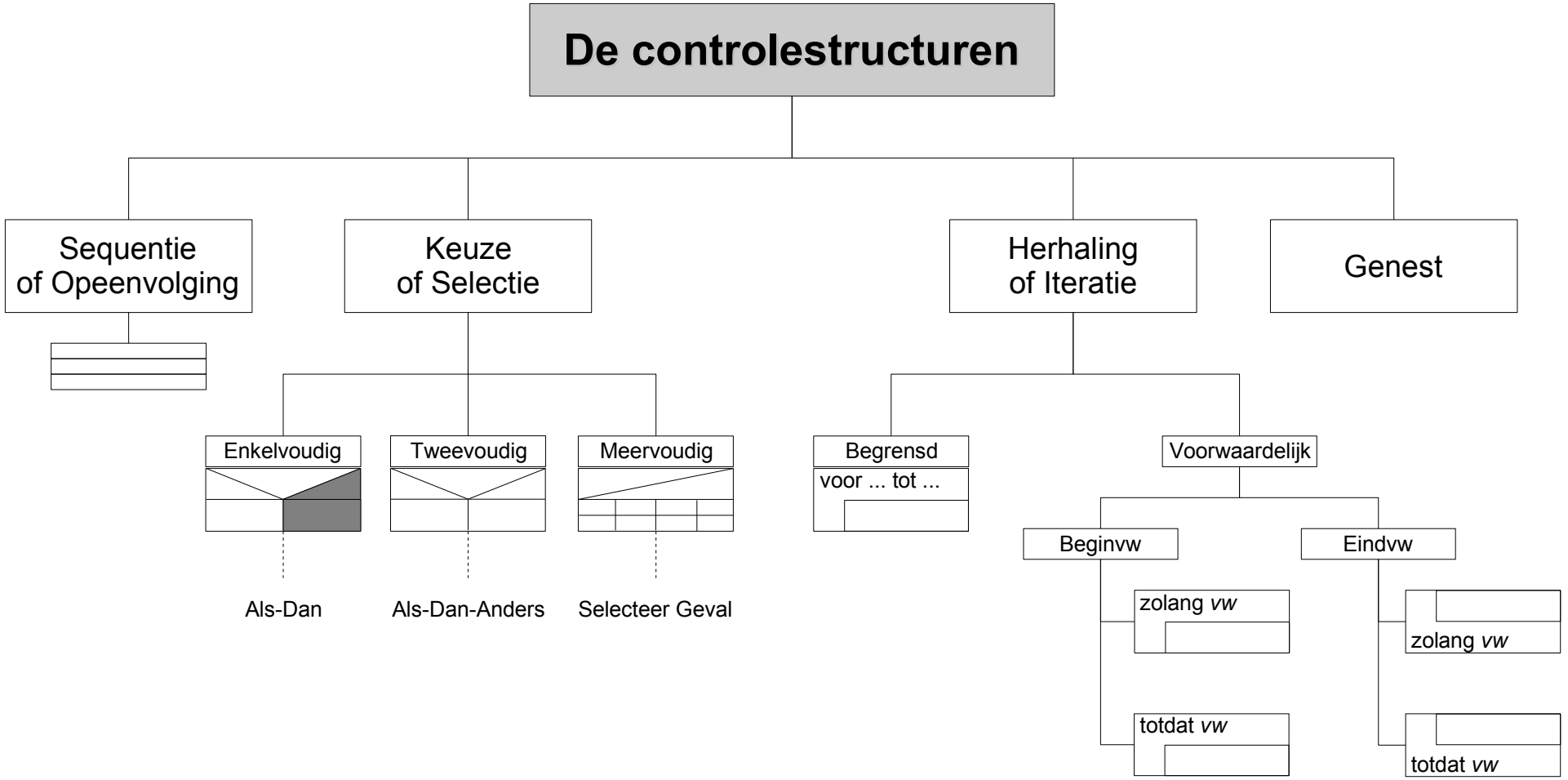
Een gebeurtenis (*event*) is een actie die door een object herkend wordt, bv.:

- Klikken op een knop.
- Het verlaten van een tekstvak.

Elk type object (bv. besturingselementen) heeft dus zijn eigen gebeurtenissen.

Vb. in JavaScript: `document.klant.druk.onclick = function (){alert('Premie berekend!')}`
Vb. in VBA: `Private Sub cmdDraai_Click()`

DE CONTROLESTRUCTUREN



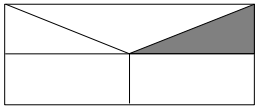
DE SEQUENTIE OF OPEENVOLGING

- alle opdrachten gewoon na elkaar

KEUZE OF SELECTIE

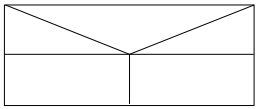
De enkelvoudige keuze

NS-diagram	Pseudocode	VBA	PHP

	**Als** <voorwaarde> **Dan**
	  <instructie 1>
	  <instructie 2>
	  …
	  <instructie n>
		If <voorwaarde> **Then**
		  <instructie 1>
		  <instructie 2>
		  …
		  <instructie n>
		End If
			if (<voorwaarde>**) {**
			  <instructie 1>;
			  <instructie 2>;
			  …
			  <instructie n>;
			}

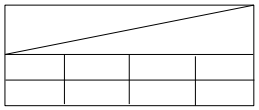
De tweevoudige keuze

NS-diagram	Pseudocode	VBA	PHP

	**Als** <voorwaarde> **Dan**
	  <instructie 1>
	  <instructie 2>
	  …
	  <instructie n>
	Anders
	  <instructie a>
	  <instructie b>
	  …
	  <>
		If <voorwaarde> **Then**
		  <instructie 1>
		  <instructie 2>
		  …
		  <instructie n>
		Else
		  <instructie a>
		  <instructie b>
		  …
		  <>
		End If
			if (<voorwaarde>**) {**
			  <instructie 1>
			  <instructie 2>
			  …
			  <instructie n>
			} else {
			  <instructie a>;
			  <instructie b>;
			  …
			  <>;
			}

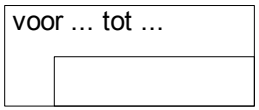
De meervoudige keuze

NS-diagram	Pseudocode	VBA	PHP

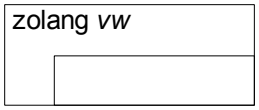
	**Selecteer Geval** <variabele>
	  Geval <expressielijst 1>
	  <instructieblok 1>
	  Geval <expressielijst 2>
	  <instructieblok 2>
		Select Case <variabele>
		  Case <expressielijst 1>
		  <instructieblok 1>
		  Case <expressielijst 2>
		  <instructieblok 2>
		End Select
			switch (<variabele>**) {**
			  case <expressielijst 1>**:**
			  <instructieblok 1>
			  case <expressielijst 2>**:**
			  <instructieblok 2>
			}

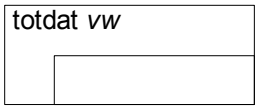
ITERATIE OF HERHALING

De begrensde herhaling

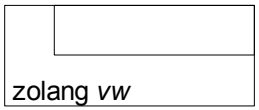
NS-diagram	Pseudocode	VBA	PHP
	**Voor** <variabele>**=** … tot … <instructieblok>	**For** <teller>= begin **To** einde <instructieblok> **Next**	**for (…** ; **…** ; **…) {** <instructieblok 1> **}**

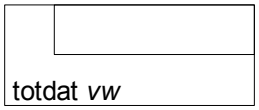
De voorwaardelijke herhaling met beginvoorwaarde

NS-diagram	Pseudocode	VBA	PHP
	**Zolang** <voorwaarde> <instructieblok>	**Do While** <voorwaarde> <instructieblok> **Loop**	**while (**<voorwaarde>**) {** <instructieblok> **}**

NS-diagram	Pseudocode	VBA	PHP
	**Totdat** <voorwaarde> <instructieblok>	**Do Until** <voorwaarde> <instructieblok> **Loop**	

De voorwaardelijke herhaling met eindvoorwaarde

NS-diagram	Pseudocode	VBA	PHP
	**Doe** <instructieblok> **Zolang** <voorwaarde>	**Do** <instructieblok> **Loop While** <voorwaarde>	**do {** <instructieblok> **}** **while (<voorwaarde>)**

NS-diagram	Pseudocode	VBA	PHP
	**Doe** <instructieblok> **Totdat** <voorwaarde>	**Do** <instructieblok> **Loop Until** <voorwaarde>	

GENESTE CONTROLESTRUCTUREN

Dit is een combinatie van sequenties, keuzes en/of herhalingen.

Hieronder zie je in pseudocode een voorbeeld, waarbij een begrensde herhaling genest is binnen een enkelvoudige keuze.

```
Als <voorwaarde> Dan
  <instructie 1>
  <instructie 2>
  Voor <variabele>= … tot …
    <instructieblok>
  <instructie 3>
```