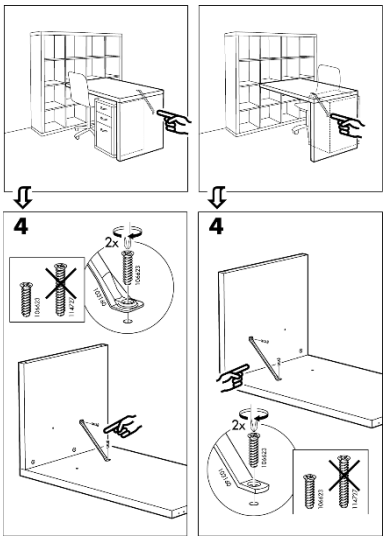
Programmeer concepten

Algoritmen

Algoritmen

Een **algoritme** is een set regels die stap voor stap handelingen aangeeft voor het verrichten van een bepaalde taak. Dit kan in tekst vorm maar wordt vaak ook door middel van een serie afbeeldingen uitgevoerd.



Pina Colada

Een Pina Colada is een heerlijke cocktail die bestaat uit witte rum, kokosmelk en ananassap. Het is een cocktail die bij uitstek past op het strand. Maar je hoeft niet ver weg te gaan om de cocktail te bestellen. Je kunt de Pina Cola namelijk ook zelf maken, in je eigen keuken. Dat is natuurlijk erg leuk en lekker om te doen. Je kunt op vele manieren een Pina Colada maken en het bereiden ervan neemt ongeveer 10 minuten in beslag. De cocktail is dus ook nog eens heel snel klaar!

Ingrediënten

- Witte rum (30 ml)
- Vloeibare room, 1 eetlepel
- Ananassap (30 ml)
- Kokosmelk (30 ml)

In sommige situaties is een afbeelding duidelijker dan een lange tekst zoals getoond wordt met de twee afbeeldingen.

Het doel van een algoritme kan van alles zijn met een duidelijk resultaat. De *instructies* kunnen in het algemeen omgaan met eventualiteiten die bij het uitvoeren kunnen optreden. Algoritmen hebben in het algemeen stappen die zich herhalen, *iteratie*, of die beslissingen, *logica* of vergelijkingen, vereisen om de taak te voltooien.

Eenzelfde taak kan gewoonlijk met verschillende reeksen instructies worden opgelost. Het verschil ligt dan meestal in de hoeveelheid *tijd*, ruimte of inspanning die het algoritme vergt; dit is de *complexiteit* van een algoritme.

Waar een algoritme de *beschrijving* is van een oplossing van een probleem, is een *computerprogramma*, in een of andere *programmeertaal*, de *implementatie* van dat algoritme.

De eigenschappen van algoritmen zijn:

- Er moet een eindig aantal stappen worden doorlopen.
- De volgorde, *sequentie*, waarin de stappen moeten worden doorlopen kan van belang zijn.
- Soms moeten stappen worden herhaald, *iteratie*.

Programmeer concepten

- Soms moeten er keuzes worden gemaakt, *selectie*, hoe het algoritme verder moet gaan.

Programmeer proces

Een Mixologist, iemand die drankjes klaar maakt, moet weten welke drankjes en andere ingrediënten (*invoer*) in de mix beker moeten. Daarna schud het de beker (*verwerking*) en schenkt het gemaakte drankje in een glas (*uitvoer*).

Voor het maken van programma's en bij vele andere computer gerelateerde zaken komt het onderstaande schema veel ter spraken.

Als programmeur, moet je bepalen welke invoer (*data/variabelen*) het programma nodig heeft. En deze zodanig te verwerken dat de gevraagde uitvoer kan worden gegenereerd door het programma.

Programma code

Imperatief programmeren, ook wel procedureel programmeren genoemd, hierbij wordt in de vorm van opdrachten die direct uitgevoerd kunnen worden een programma opgesteld. Het tegenovergestelde van imperatief programmeren is declaratief programmeren, dat niet iets *doet* maar iets *beschrijft*.

De moderne programmeertalen zijn echter niet meer procedureel maar object georiënteerd van aard.

Echter bij alle methoden worden er *vier basis structuren*, bouwstenen, gebruikt voor het schrijven van elk computer programma.

1. Sequentie
2. Selectie
3. Iteratie
4. (sub)Routines

Programmeer concepten

Sequentie

Je trekt eerst je vuile kleren uit, gaat dan douchen en trekt daarna de schone kleren aan. Je gaat niet eerste je schone kleren aantrekken en dan douchen. Er is een bepaalde volgorde van handelen. Sequentie bij programmeren is wanneer opdrachten in volgorde één voor één achter elkaar worden uitgevoerd door de computer.

Selectie

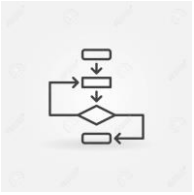
Je gaat winkelen en ziet in een winkel een mooie broek. Je besluit om eerst in andere winkels te kijken voor je de broek koopt. In een andere winkel vindt je exact dezelfde broek voor een lagere prijs en besluit dus om die te kopen. Je hebt de prijzen vergeleken en een keuze gemaakt. Selectie is wanneer er een keuze wordt gemaakt aan de hand van een criteria (*voorwaarde*) welke opdrachten van het programma de computer moet uitvoeren.

Iteratie

Iteratie is het herhalen van dezelfde programma instructies afhankelijk van een criteria. Dit wordt in programeer termen een loop of wel een lus genoemd. Er zijn drie type lussen te onderscheiden:

1. **Met controle vooraf**. Voordat de opdrachten kunnen worden uitgevoerd wordt eerst getest of dat een gegeven voorwaarde voldoet. De opdrachten worden herhaald zolang de voorwaarde blijft gelden.
2. **Met controle achteraf**. Hier worden de opdrachten de eerste keer meteen uitgevoerd. Daarna wordt er pas gekeken naar de voorwaarde of deze mogen worden herhaald. Deze constructie wordt tegenwoordig zelden of nooit meer gebruikt daar het voor fouten in de programma's kan zorgen.
3. **Vaste aantal herhalingen**. Hier wordt van te voren al vast gesteld hoe vaak de opdrachten in de lus herhaalt moet worden. Via een teller wordt bijgehouden hoeveel keer de opdrachten zijn uitgevoerd. Met de controle vooraf lus is deze lus na te bootsen.

De *controle vooraf lus* wordt gebruikt wanneer niet bekend is hoeveel keer iets herhaald moet worden maar waar wel bekend is bij welke conditie het herhalen moet worden beëindigd. De lus met *vaste aantal herhalingen* wordt gebruikt wanneer precies bekend is hoeveel keer de opdrachten dienen te worden uitgevoerd.



Programmeer concepten

Subroutines

Voor het oplossen van een probleem of het maken van iets wordt in de praktijk dit verdeelt over kleinere delen. Neem het bouwen van een huis. Hier wordt er eerst een bouwtekening gemaakt, wat op zichzelf ook een proces is. Daarna wordt begonnen aan de bouw. Eerst de fundering, dan optrekken van cementblokken tot aan de ringbalk, dan het dak er op, etc. Elk van deze onderdelen heeft zijn eigen handelingen en voor de bouw van ieder huis worden deze zelfde handelingen iedere keer weer opnieuw gedaan.

Dit komt ook voor bij het schrijven van een programma. Zie een programma als een geheel dat een complex probleem oplost. Je kunt dit probleem verdelen in kleinere, minder complexe delen die gezamenlijk het probleem oplossen. Dit worden dan de subroutines waaruit het programma is opgebouwd.

Subroutines worden ook wel Procedures, Functies, Module of Methods genoemd afhankelijk van de programmeertaal en de methodiek die gebruikt wordt.

Subroutines zorgen ook voor meer structuur en maken programma's overzichtelijker en makkelijker te lezen. Verder zorgen goed gedocumenteerde programma's met subroutines ervoor dat het onderhoudt van het programma wat eenvoudiger wordt. Ook het opsporen van eventuele fouten wordt makkelijker daar deze in een bepaalde routine geïsoleerd wordt.

Wanneer een Subroutine wordt aangeroepen, kunnen een of meerdere waarden worden meegegeven aan de routine voor verwerking.
Functies zijn speciale subroutines die een waarde terug sturen naar de aanroepende.

Methodologie

Door de jaren heen zijn er steeds andere technieken ontwikkeld voor het schrijven van programma's. De meeste gebruikte twee zijn de Procedurele en Object georiënteerde technieken.

Procedureel

Bij de procedurele manier van programmeren wordt het doel van het programma verdeeld over meerdere kleinere delen die de procedures vormen van een programma.

Programmeer concepten

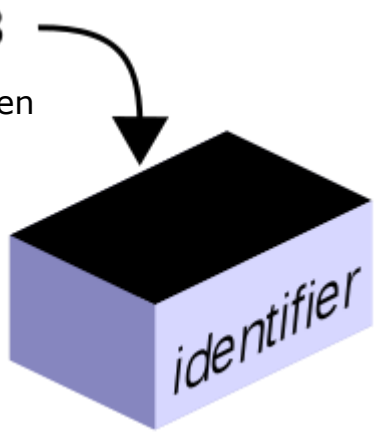
Object georiënteerd

De object georiënteerde techniek maakt voornamelijk gebruik van Functies en Method's. Bij deze programmeertechniek wordt een programma gemaakt door het beschrijven van objecten. **Object Oriented Programming**, of wel **OOP**. Dit is de moderne manier voor het ontwikkelen van programma's. Zo'n beschrijving wordt een Class (*Klasse*) genoemd.

Datastructuren

Programma's hebben gegevens nodig om hun taak te kunnen verrichten. Deze gegevens worden ook wel Data genoemd. Data komt in verschillende vormen voor in programma's
Computers gebruiken geheugen om data vast te kunnen houden. Dit gebeurd in het werkgeheugen (RAM). Er wordt een deel van het RAM gereserveerd om de informatie in op te slaan.

Variabelen

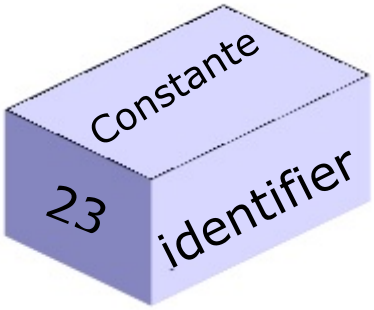


Variabele zijn waarden die meestal worden geïnitialiseerd naar *NULL*, "0", lege string ("") of een standaardwaarde omdat de werkelijke waarden worden geleverd door de gebruiker van een programma.

Zie een variabele als een doos waar je iets in kunt bewaren. De doos geef je een naam, identifier, waar aan je kunt herkennen wat er in die doos is opgeslagen.

Variabelen hebben een naam die bij de meeste programmeertalen moet beginnen met een letter, gevolgd door meerdere letters en of cijfers. Geen speciale tekens. Behalve de underscore "_". Deze wordt meestal wel toegestaan. Voor de programmeertaal PHP moet elke variabele beginnen met het "$" teken.

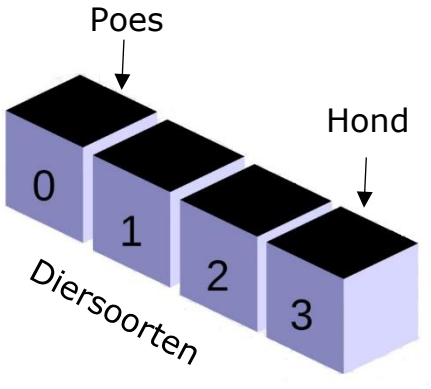
Constanten



Een gegeven (variabele) die een constante is, heeft een *vaste waarde* die nooit veranderd. Deze waarde wordt aan het begin van het programma vastgesteld door de programmeur.

Programmeer concepten

Array



Een array is een list van variabelen van het zelfde type en met dezelfde naam. De individuele waarden van de elementen worden bereikt door hun indexnummer te gebruiken.

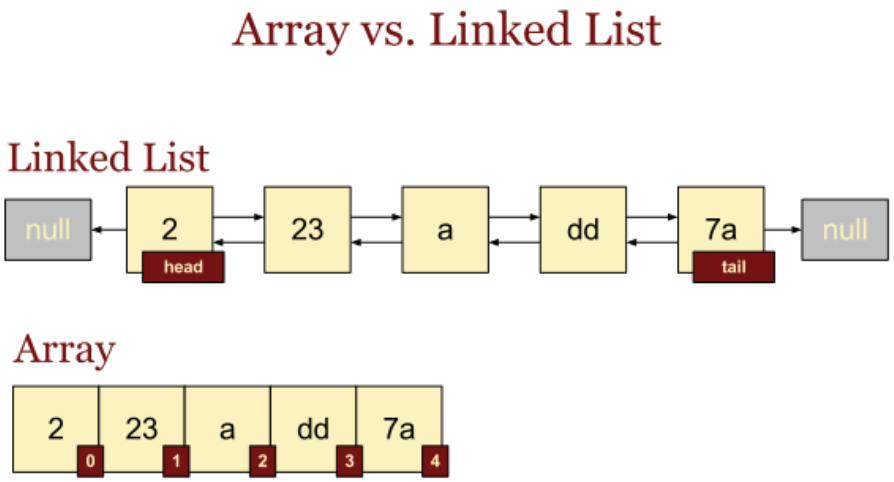
In de afbeelding zie je dat in het array Diersoorten element 0 de waarde “Poes” bevat en element 3 “Hond”.

Er zijn arrays met één index, maar je kunt ook arrays maken met meerdere indexen. Bijvoorbeeld twee of drie dimensionale arrays. Bij de meeste programmeertalen moet je aangeven uit hoeveel elementen een array gaat bestaan. Dit gebeurd aan het begin van het programma. De computer weet dan hoeveel werkgeheugen het moet reserveren voor de array. Arrays hebben dus een vaste lengte/grote.

Lijsten

Lijsten lijken heel veel op arrays. Ze bestaan ook uit meerdere elementen. Het verschil is dat lijsten geen vaste lengte hebben en er wordt gewerkt met verwijzingen naar het vorige of volgende element. De meeste programmeertalen hebben ook instructies om lijsten te bewerken. Voor of achteruit door de elementen van een lijst lopen. Instructies om elementen toe te voegen en verwijderen van een lijst. Ook zijn er instructies om lijsten te sorteren of te doorzoeken. Afhankelijk van de programmeertaal zijn deze niet altijd aanwezig bij arrays.

In de afbeelding hieronder zie je het verschil tussen een gelinkte lijst en een array.



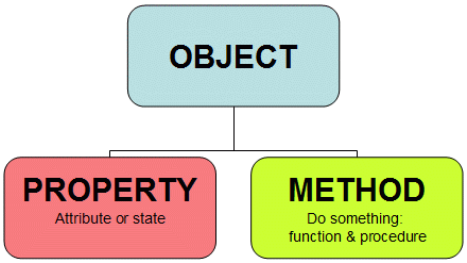
Programmeer concepten

Objecten

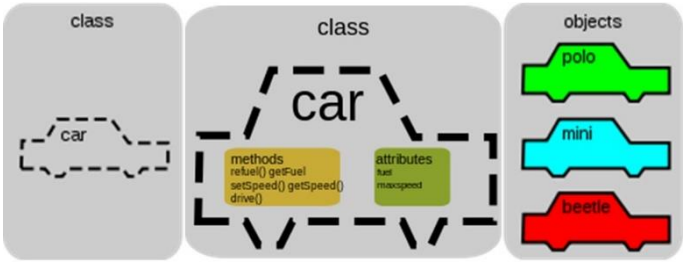
Bij **O**bject-ge**O**riënteerd **P**rogrammeren – OOP, wordt er een beschrijving gemaakt van een object.

In OOP is een object een op zichzelf staand geheel met een eigen taak. Zo'n object kan data opslaan en kan ook een bepaald 'gedrag' hebben. De data van een object wordt opgeslagen in wat je noemt attributen, *properties*. Attributen van een object kunnen zelf ook weer objecten zijn.

Bij OOP kun je functies aan objecten koppelen. De functies vormen het gedrag van het object, zij bepalen wat het object doet. Een functie dat aan een object is gekoppeld wordt een methode, *method*, genoemd.



De beschrijving van het object heet een Klasse, *Class*. Een klasse is een soort blauwdruk (plattegrond/bouwtekening) waarvan meerdere objecten gemaakt zouden kunnen worden.



Neem bijvoorbeeld de bouwtekening van een auto, dat is de Klasse. Wanneer in een autofabriek nu verschillende type auto's gemaakt worden die allemaal gebaseerd zijn op de bouwtekening dan zijn al deze auto's objecten van die ene klasse.

De vier functies die een taal een Object-georiënteerde taal maken zijn:

- **Inheritance**
  - Met overerving kunt u een klasse definiëren in termen van een andere klasse, waardoor het eenvoudiger wordt om een toepassing te maken en te onderhouden. Voordeel van het gebruik van dit concept is code herbruikbaarheid en snelle implementatietijd. De ervende eigenschappen van bestaande klasse worden overgenomen in de nieuwe klasse en hoeven dus niet opnieuw te worden beschreven. Bestaande klasse wordt een basisklasse genoemd en de klasse die de eigenschappen van bestaande klassen erven, wordt afgeleide klasse genoemd.

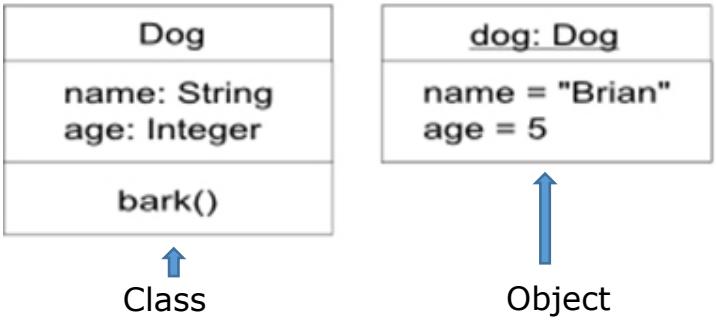
- **Polymorphism**
  - Zoals een naam impliceert, betekent polymorfisme "vele vormen". Het is de vaardigheid van elk object om vele vormen aan te kunnen nemen. Wordt meestal gebruikt als een verwijzing naar een ouderklasse wordt gebruikt om naar een onderliggende klasobject te verwijzen.

Programmeer concepten

- **Abstraction**
  - Het primaire doel van Abstractie is om de complexiteit van de grote programma's te beheren door de irrelevante details van de gebruiker te verbergen.

- **Encapsulation**
  - Encapsulation heeft alles te maken met het verbergen van details van de buitenwereld en een interface voor de entiteiten van de buitenwereld om met objecten te communiceren.

De schematechniek die gebruikt wordt voor het beschrijven van objecten heet **U**nified **M**odeling **L**anguage – UML. Deze bestaat uit een aantal verschillende modeleringstechnieken.



Standaard Algoritmen

Er zijn een aantal standaard algoritmen die heel belangrijk zijn om te leren. Ze komen in vele programma's voor en er zijn veel varianten van elke.

- Sorteren
  - Bubble sort, Merg sort en Quick sort zijn voorbeelden hiervan.
- Zoeken
  - Lineair zoeken en Binair zoeken
- Kortste pad bepalen

Zijn enkele voorbeelden. Sorteren en zoeken van informatie zijn taken die veelvuldig door computers moeten worden uitgevoerd. Neem bijvoorbeeld het lineair zoek algoritme. Deze methode heeft andere eisen en andere voor en nadelen tegenover het binair zoeken.

Opdracht 1:
Ga op het Internet en zoek uit hoe deze verschillende Algoritmen werken. Beschrijf de werking van elk algoritme, geef aan wanneer ze het beste gebruikt kunnen worden, geef aan de eisen voor het gebruiken van elk algoritme en geef de voor- en nadelen van elk aan.

Opdracht 2:
Maak een PSD voor het lineair zoek algoritme.

Opdracht 3:
Maak een PSD voor het Bubble sort algoritme.