leren. durven. doen.

SYNTRA WEST

Module
Leren Programmeren

LOGISCH REDENEREN
BASIS PROGRAMMEREN

west-vlaanderen POM

Inhoud Leren Programmeren

- Inleiding
- Algoritmisch & «Programmeer» denken
- Introductie HTML & Javascript
- Leren Programmeren
  - Basisbegrippen: variabelen, primitieve datatypes.
  - Eerste programma schrijven
  - Instructies en operatoren.
  - Functies –Hoisting.
  - Arrays.

leren. durven. doen.

SYNTRA WEST

Leren Programmeren

ALGORITMISCH DENKEN

west-vlaanderen POM

Inhoud

Inleidende begrippen

- informatie
- gegevensverzamelingen
- methodes van gegevensverwerking
- probleemidentificatie
- probleemoplossing
- programmeren
- programmeertalen

Basis van programmeren: algoritmisch denken

Inleidende begrippen - Informatie

Informatie

- Een verzameling van gegevens die begrijpelijk zijn, een betekenis hebben

Belangrijkste eigenschappen

- volledig
- Ondubbelzinnig
- Actueel
- Functioneel (bruikbaar)
- Beknopt (zonder overtolligheid)

Informatie

Soorten of vormen van informatie

- Numerieke (bv getallen, prijzen, …)
- Alfanumeriek of beschrijvende (beschrijvingen met woorden, letters en leestekens)
- Grafische (schetsen, tekeningen, grafieken)
- Symbolen (een reeks afgesproken tekens bv muziekschrift, morse, scheikundige symbolen,…)
- …

Het informatieverwerkend proces

De wijze waarop men informatie verwerkt

(mens, computer), staat model voor elk informatieverwerkend proces

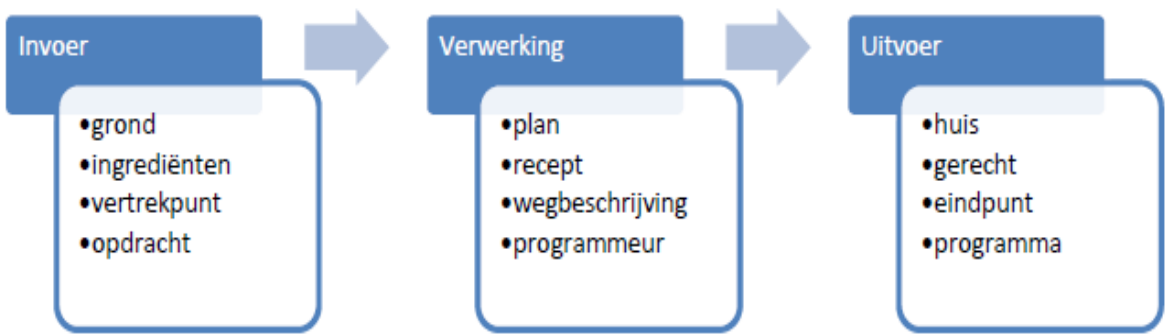
3 fasen van elk informatieverwerkend proces

(bv Bestelling bij kelner op restaurant of koken volgens recept)

1.Invoeren van de informatie (input)

2.Verwerking van de informatie (processing)

3.Uitvoer van de informatie (output)



Computerprogramma

Een reeks van instructies die de computer in strikte volgorde moet uitvoeren om de gewenste informatieverwerking te bekomen.

- = software zoals tekstverwerking, rekenblad, gegevensbeheer, browser
- programma's zijn geschreven in één of andere programmeertaal, bv Javascript

Programmeertalen

Laag niveau

(bijna) rechtstreeks toegang tot de hardware

Cryptisch, onleesbaar, niet portable en zelfs gevaarlijk (computer doen ‘crashen’)

Bv Machinecode, Assembler

Gemiddeld niveau (bv C)

Door compiler vertaald naar machinecode

Portable, leesbaarder, soms ook risico geven in verband met schrijven naar geheugen

Hoog niveau

Worden door interpreter of compiler vertaald naar machinecode

Eenvoudiger te leren,veilig, gemakkelijker in onderhoud, portable

Trager (snellere processor nodig), meer « ruimte » nodig =meer geheugen nodig

FORTRAN (FORmula TRANslator), COBOL (Common Business Oriented Language), Pascal, BASIC, C++, VB, Prolog, Java, Perl, Python, C#, VB.Net,…

Soorten programmeertalen

Databasetalen

SQL (Structured Query Language) MS Access, db2

T-sql (MS SQL Server Databases)

Scripttalen

Bv VBA (Visual Basic for Applications) in MS Word en Excel en MS Access (macro's)

Webprogrammeertalen

HTML(HyperText Markup Language), CSS, …

Javascript, …

Algoritme en Algoritmisch denken

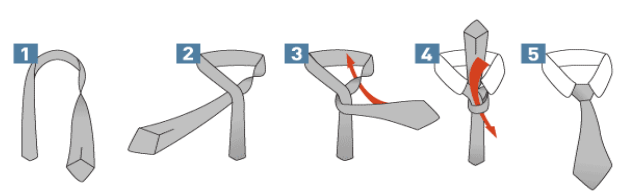
Algoritme

Een algoritme is een reeks opdrachten die in een bepaalde volgorde uitgevoerd moeten worden om een probleem op te lossen.
Zowel een bouwplan, een keukenrecept, een wegbeschrijving, instructies voor knopen van een das… geven in verschillende stappen een oplossingsmethode weer.
Ze beschrijven hoe je van een beginsituatie tot een resultaat kunt komen.

3 fasen in beschrijving van algoritme

- **Beschrijving van de beginsituatie**, de middelen nodig voor het algoritme (**invoerobjecten**) (bv ingrediënten voor cake)
- **Procesbeschrijving**: opeenvolgende stappen die nodig zijn om het gewenste resultaat te bereiken, zorgt voor de verwerking (bv beschrijving van recept cake)
- Beschrijving van de **eindsituatie (het gewenste resultaat)**. Meestal een opsomming van de uitvoerobjecten (bv cake met rozijnen)

Algoritmisch denken is leren hoe je een algoritme moet opbouwen.



Algoritme en Programma

1. Probleemdefinitie

het probleem en de gewenste resultaten worden beschreven

2. Probleemanalyse en -oplossing

wegen onderzoeken die naar een oplossing leiden beschrijven

- Wat is er gegeven?
- Wat moet er ingevoerd worden?
- Hoe zullen we dit verwerken?
- Wat moet de uiteindelijke uitvoer zijn?

3. Probleemanalyse en -oplossing

de stappen nodig om een probleem correct en efficiënt op te lossen

4. Programma

Om het algoritme te kunnen uitvoeren is er een programma nodig

5. Testen

Test of aan je probleemdefinitie voldaan wordt.

Voorbeeld Algoritme

1. Probleemdefinitie

Ik wil *de aflevering van The Simpsons* (zender: *VIER*) van morgen om 14u opnemen op de digicorder.

2. Probleemanalyse en -oplossing

de stappen nodig om een probleem correct en efficiënt op te lossen:

• Wat is er gegeven?	De zender, de dag en het uur van de opname.
• Wat moet er ingevoerd worden?	De digicorder moet opnemen op de gegeven dag, de juiste zender en het gegeven uur.
• Hoe zullen we dit verwerken?	De dag ingeven, de zender ingeven en het uur ingeven.
• Wat moet de uiteindelijke uitvoer zijn?	Het programma is opgenomen.

4. Algoritme

- INVOER: dag
- INVOER: zender
- INVOER: uur

5. Programma

Programmeer op de digicorder

5. Testen

Test of de digicorder de opname bevat van het programma.

Oefening Algoritme

Maak een stappenplan voor volgend probleem:

- ***Ik wil de oppervlakte berekenen van een rechthoekig stuk grond***

1. Probleemdefinitie

2. Probleemanalyse en –oplossing

3. Algoritme

4. Programma

5. Testen

Oefening Algoritme - Oplossing

Maak een stappenplan voor volgend probleem:

- *Ik wil de oppervlakte berekenen van een rechthoekig stuk grond*

1. Probleemdefinitie

Oppervlakte van een rechthoek bepalen

2. Probleemanalyse en -oplossing

de stappen nodig om een probleem correct en efficiënt op te lossen:

- **Gegeven**: rechthoek met afmetingen
- **Invoer**: afmetingen
- **Verwerking**: lengte x hoogte
- **Uitvoer**: oppervlakte

Oefening Algoritme

Maak een stappenplan voor volgend probleem:

- *Ik wil de oppervlakte berekenen van een rechthoekig stuk grond*

3. Algoritme

- VRAAG: lengte rechthoek?
- INVOER lengte
- VRAAG: breedte rechthoek?
- INVOER breedte
- BEREKEN:oppervlakte = lengte x breedte
- UITVOER:oppervlakte rechthoek

4. Programma

Schrijf de code voor het algoritme in een programmeertaal (bv Javascript)

5. Testen

Test het programma door een aantal testwaarden in te geven

en het resultaat te controleren.

.

Hulpmiddelen om algoritme te beschrijven

Gestructureerde schema's

- Grafische voorstelling (tekening) of beschrijving in woorden van de opeenvolgende stappen van een algoritme
- Om een duidelijk overzichtelijk geheel te verkrijgen
- Tussenstap bij het omzetten van het probleem in de taal van de computer

Voorbeelden

- Nassi-Schneidermann-diagram(structogram)

= Algoritme in een gesloten blok, dwingt tot gestructureerde opbouw, kan zowel van boven naar onder als van onder naar boven

Pseudocode

- Gestructureerde beschrijving met keywords

Stroomdiagram (Flow chart)

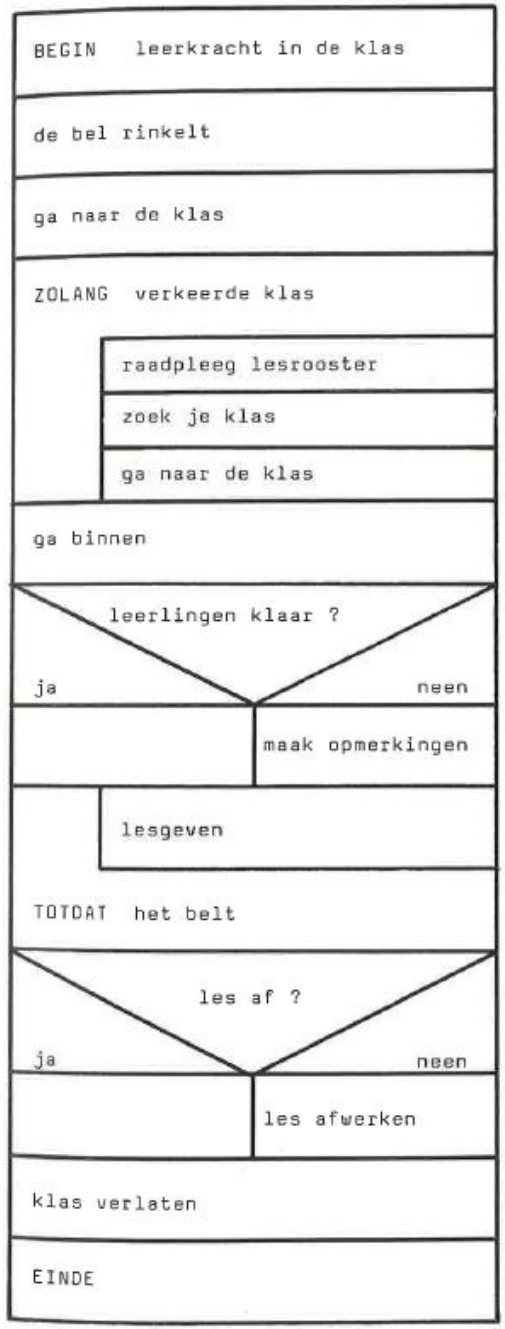
- Geeft visueel beeld van het verloop van een algoritme (populair bij analisten: UML en BPMN)

Pseudocode (verbale beschrijving)

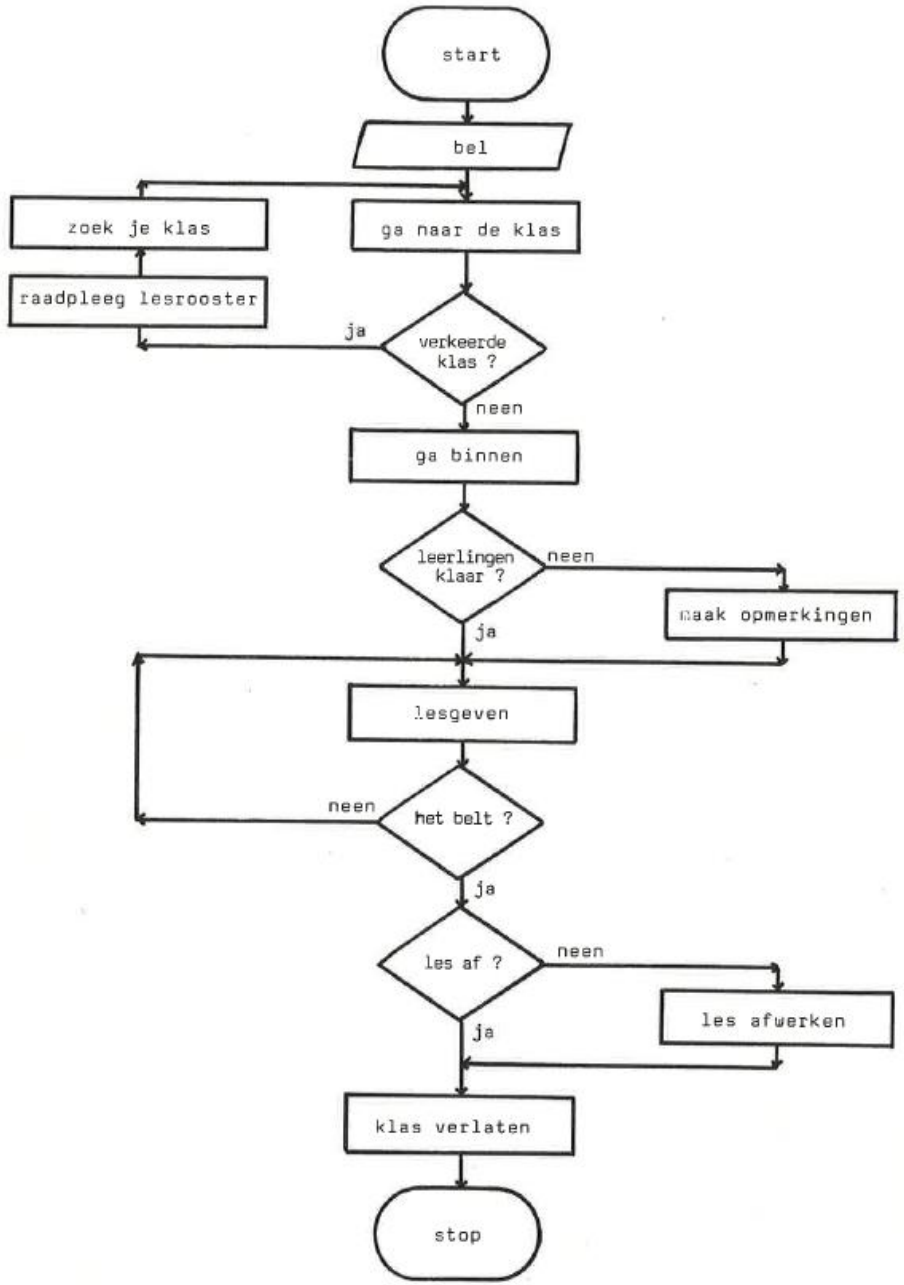
Leerkracht in de klas:

De bel rinkelt;
Ga naar de klas;
ZOLANG verkeerde klas
HERHAAL raadpleeg lesrooster;
    zoek je klas;
    ga naar de klas;
EINDE HERHAAL;
Ga binnen;
ALS studenten niet klaar
    DAN maak opmerkingen
EINDE ALS;
HERHAAL lesgeven
TOTDAT het belt;
EINDE HERHAAL;
ALS les niet af
    DAN les afwerken;
EINDE ALS;
Klas verlaten.

Nassi-Schneidermann-diagram



Stroomdiagram (Flow chart)



Referenties

https://www.learncs.org/

https://www.w3schools.com/cs/default.asp

Algoritmisch Denken, Ann-Sophie Fevery