Docentenhandleiding

– Algoritmisch Denken en Gestructureerd Programmeren in Greenfoot –

1 Licentie

2 Inleiding

Voor u ligt de docentenhandleiding voor de cursus 'Algoritmisch denken en gestructureerd programmeren (in Greenfoot)'. In dit document geven we uitleg over hoe het materiaal ingezet kan worden in de klas. We reiken daarbij (interactieve) lesideeën en adviezen aan, waardoor de concepten sterker vormgegeven worden. Algemene informatie over de cursus, zoals de doelstellingen, opbouw, inhoud en thema's van de opdrachten staan beschreven in het document "Algemeen opzet".

Het materiaal van de cursus bestaat uit het volgende:

- Algemeen opzet: beschrijft het didactisch model en geeft een overzicht van de concepten, leerdoelen, vereiste voorkennis en studielast.
- Docentenhandleiding (dit document): uitleg over hoe het materiaal in te zetten in de les, inclusief lesideeën en adviezen.
- Opdrachten: werkbladen met ieder een set afgebakende leerdoelen en daarbij behorende theorie, vragen, opgaven, (diagnostische toets) en samenvatting;
- Voorbeelduitwerkingen van de opdrachten;
- Scenarios: voorgegeven programmacode waarmee de leerlingen aan de slag kunnen;
- Template stroomdiagrammen (zoals voorbeeld stroomdiagrammen.pptx): een template voor de stroomdiagrammen gemaakt in Powerpoint die de leerlingen kunnen aanpassen en uitbreiden (dit is gegeven als onderdeel van scenario 2);

3 Opbouw van de opdrachten

Elk opdracht bevat:

- **Inleiding**: Elke opdracht begint met een inleiding waarin kort wordt uitgelegd wat er in de opdracht gedaan wordt en hoe dat zich verhoudt tot de voorgaande opdrachten.
- **Leerdoelen**: Beschrijving van wat de leerling na afloop van het opdracht kan.

[1] De volledige licentie-tekst is te lezen op: https://creativecommons.org/licenses/by/4.0/.

- **Theorie blokken**: Uitleg van benodigde kennis wordt in een rechthoek weergegeven. Dit wordt altijd expliciet aangeboden, ook al heeft de leerling in een voorafgaande opgave een deel van de theorie al zelf ontdekt. Nieuwe begrippen zijn direct te herkennen, doordat ze *schreef* zijn gedrukt. Code en sleutelwoorden zijn direct te herkennen, doordat deze in een ander `font−type` zijn gedrukt. Een belangrijk onderdeel van de theorie zijn de voorbeelden. Hier ziet een leerling onder andere welke strategie voor het oplossen van een probleem gevolgd kan worden. Daarna worden de voorbeelden (in woorden) uitvoerig toegelicht. De uitleg volgend op een voorbeeld dient niet als een voorbeeldуitwerking; deze is enkel bedoeld om een ontwerp of implementatie toe te lichten. De voorbeelden zijn opgenomen binnen een theorie blok.

- **Opgaven**: De opgaven, met als doel de theorie eigen te maken, zijn van verschillende aard.
  - Oriëntatie-opgaven waarbij leerlingen zelf theorie of eigenschappen ontdekken
  - Oefenopgaven: Door middel van een serie oefenopgaven met een opbouw van relatief eenvoudig naar gecompliceerder, kan de leerling zich de technieken behorende bij de aangeboden theorie eigen maken
  - Reflectie-opgaven: De leerling wordt uitgenodigd om een bepaald aspect van de theorie nog eens te overdenken, om zich af te vragen waarom voor een bepaalde aanpak is gekozen, om nog eens op een andere manier tegen een probleem aan te kijken, of om af te wegen welke voor- en nadelen de gekozen strategie heeft (ten opzichte van een alternatieve aanpak). Reflectie is een belangrijk aspect van het zelfverantwoordelijk leren. Het is belangrijk de leerlingen het nut van deze opgaven te laten inzien, bijvoorbeeld door er af en toe klassikaal mee aan de slag te gaan. Bijkomend voordeel is dat de leerlingen ontdekken dat verschillende strategieën tot een juiste oplossing kunnen leiden. Door deze strategieën met elkaar te vergelijken krijgen leerlingen inzicht in waarom een bepaalde oplossing mogelijk beter is dan een andere. Het wenselijke doel is dat de leerlingen zichzelf reflecterende vragen gaan stellen, ook als deze niet in de opgaven staan. Op deze manier wordt het zelfverantwoordelijk leren omarmd.
  - Afsluitende opgaven: Meestal wordt een opdracht afgesloten door een of meer afsluitende opgaven, welke je aan de toevoeging '(A)' kunt herkennen. Deze heeft een iets hoger niveau dan de voorgaande oefenopgaven. Om een oplossing te vinden dienen hiervoor belangrijke vaardigheden uit andere opgaven toegepast te worden. In de meeste gevallen ontbreken hierbij tussenvragen. Een sterke leerling die met minder oefening toe kan, zal in elk geval de afsluitende opgaven moeten maken. Lukken de afsluitende opgaven, dan mag worden aangenomen dat de vaardigheden worden beheerst en dat de leerstof goed is verwerkt.

- **Samenvatting**: Aan het einde van de opdracht is kort samengevat wat de belangrijkste leerdoelen van de opdracht zijn. De leerling kan bij zichzelf (op hoog niveau) nagaan of de doelstelling van de opdracht bereikt is. Daarnaast gaat hierdoor het overzicht niet verloren.

- **Diagnostische toets (en uitwerkingen)**: (in ontwikkeling) Een aantal opdrachten sluit af met een diagnostische toets. De leerling heeft zo de mogelijkheid zichzelf na elke opdracht te toetsen. Lukt de toetsopgave niet, dan is het belangrijk de opdracht nog eens door te nemen. Eventueel kan de docent aanvullende opdrachten aanreiken. De toetsopgaven toetsen alleen de basisvaardigheden en begrippen van elke opdracht. Het voldoende maken van de toetsopgaven van elk opdracht garandeert niet dat de volgende opdrachten met succes kunnen worden gemaakt. In de uitwerkingen wordt één voorbeeldoplossing gegeven. Bij twijfel dient de docent aan te geven of een alternatieve oplossing adequaat is.

- **Afsluitende instructies**: Aan het einde van elke opdracht is aangegeven hoe het werk opgeslagen kan worden. De docent dient verder af te stemmen met de leerlingen hoe dit eventueel ingeleverd dient te worden (bv. per e-mail of ELO).

4 Programmeeromgeving

4.1 Greenfoot

Er wordt geprogrammeerd in de taal Java. In de opdrachten werken we met de *Greenfoot* programmeeromgeving. Deze is gratis beschikbaar. In opdracht 1 wordt uitgelegd hoe deze te downloaden, installeren en te gebruiken.

4.2 Stroomdiagrammen

Voor het tekenen van stroomdiagrammen hebben leerlingen pen en papier nodig. Indien leerlingen de stroomdiagrammen op de computer willen maken kunnen ze hiervoor PowerPoint gebruiken. Gebruik dan de voorbeeld diagrammen (voorbeeld stroomdiagrammen.pptx) die gegeven zijn bij het tweede scenario 'DodoScenario2'.

In elk stroomdiagram zijn begin- en eindsituatie beschreven. Het doel is om de leerlingen te laten nadenken over de benodigde voorwaarden waaraan voldaan moet worden zodat een algoritme correct werkt en de eindsituatie die ontstaat na het uitvoeren van de stappen van het algoritme. De intentie is om leerlingen hierover te laten nadenken en te begrijpen wat de rollen van begin- en eindsituaties zijn in het beschrijven van een algoritme; het is niet de bedoeling dat leerlingen pre- en postcondities uitvoerig of formeel beschrijven.

4.3 Scenario's

Het is de bedoeling dat de leerlingen vanaf opdracht 2 steeds hun eigen code uitbreiden. Omdat het voor het begrip overzichtelijker is om eigen code uit te breiden dan om voorgegeven code uit te bouwen, heeft het doorgaan met het eigen code de voorkeur.

Echter, ervaring leert dat de code na de eerste paar opgaven wellicht rommelig en daardoor onoverzichtelijk en onwerkbaar kan worden. Indien het echt nodig is, kan door de leerling overschakelen op een voorgegeven scenario. Voor iedere opdracht is een voorgegeven startscenario beschikbaar. Vanaf opdracht 7 wordt telkens gestart met een nieuw voorgegeven scenario. Dit is nodig om complexere opdrachten te kunnen maken.
Hieronder een overzicht:

Opdracht nr	Te gebruiken scenario	Indien leerlingcode onbruikbaar
1	DodoScenario1	
2	DodoScenario2	
3	Eigen code nav opdracht 2, (`Opdr2_jouwNaam`)	DodoScenario3
4	Eigen code nav opdracht 3, (`Opdr3_jouwNaam`)	DodoScenario4
5	Eigen code nav opdracht 4, (`Opdr4_jouwNaam`)	DodoScenario5
6	Eigen code nav opdracht 5, (`Opdr5_jouwNaam`)	DodoScenario5
7	DodoScenario7	
8	SokobanScenario8	
9	BallWorldScenario9	

5 Adviezen aan de docent

Naar aanleiding van onze ervaring met het materiaal en het lesgeven van programmeeronderwijs, volgen hier een aantal (vrijblijvende) adviezen.

5.1 Wijze van werken

Het laten werken van de leerlingen in tweetallen (pair-programming) wordt sterk aanbevolen. Het hard-op laten denken, dus elkaar vertellen wat ze aan het doen zijn, bevordert het leerproces.

Elk leerling dient regelmatig van taak te wisselen, bijvoorbeeld per opgave (niet per opdracht). Voorbeelden van een rolverderling zijn:

- samen bespreken van het algoritme;
- de één maakt het stroomdiagram;
- de ander schrijft de code waarbij de eerste meekijkt en adviseert;
- debuggen wordt samen uitgevoerd;
- samen bespreken en analyseren van het ontwikkelproces.

Door regelmatig van rol te verwisselen wordt het denken van beide leerlingen op twee niveau's gestimuleerd: op code-niveau en op strategisch-niveau.

Het reflecteren en nabespreken van de opdrachten (verschillende aanpakken, oplossingen en moeilijkheden) dient bij voorkeur ook klassikaal of in groepjes plaats te vinden.

5.2 Lesideeën

Hier een overzicht met thema's en verwijzingen naar (interactief) lesmateriaal:

1. Wat is informatica? Wat is algoritmisch denken? Het gaat om puzzelen!
   - Error-detection (data representatie)
     Activiteit 3: Card flip magicerror detection and correction
     `http://www.ncwit.org/sites/default/files/resources/computerscience-in-a-box.pdf`
   - Hoe werken computers samen
     Activiteit 6: The orange gamerouting and deadlock in networks
     `http://www.ncwit.org/sites/default/files/resources/computerscience-in-a-box.pdf`
   - Binaire representatie:
     Activiteit: `http://csunplugged.org/binary-numbers/`
     Lesactiviteit (film en werkblad): `http://csunplugged.org/binary-numbers/`
     Bijbehorende uitgewerkte opdracht in Greenfoot voor klassikale demonstratie (implementantie als differentiërende verdiepingsopdracht): `http://greenroom.greenfoot.org/resources/6`
     Alternatief: Activiteit 1 met extension: counting the dots
     `http://www.ncwit.org/sites/default/files/resources/computerscience-in-a-box.pdf`
2. Algoritmes
   - Opstellen en vergelijken van algoritmes
     Activiteit: Locked-In
     lesinformatie: `http://teachinglondoncomputing.org/resources/inspiring-unplugged-class the-locked-in-activity/`
     opdrachtenblad: `http://teachinglondoncomputing.org/resources/inspiring-unplugged-cla the-locked-in-activity/`
   - Opstellen en vergelijken zoek algoritmes
     Activiteit: Wie-ben-ik
     lesinformatie: `http://teachinglondoncomputing.org/resources/inspiring-unplugged-class the-20-questions-activity/`

- Algoritme efficiëntie: snel tellen (5′)
  Lesactiviteit (voorbeeld filmpje fragment 20:28 - 24:53): https://www.youtube.com/watch?v=z-OxzIC6pic
- Efficiëntie en testen van algoritmes:
  Lesactiviteit (werkblad): http://teachinglondoncomputing.org/resources/inspiring-unplugged-classroom-activities/the-swap-puzzle-activity/

3. Protocollen opstellen en analyseren
   - Communicatie protocol (werkblad):
     http://csunplugged.org/network-protocols/
   - Cryptografie protocol
     lesactiviteit (voorbeeld filmpje): http://csunplugged.org/cryptographic-protocols/
   - Public key encryption protocol
     lesactiviteit (voorbeeld filmpje): http://csunplugged.org/public-key-encryption/
4. Data representatie en compressie algoritmes:
   - Compressie uitleg: http://www.cs4fn.org/compression/burrowswheeler.php
   - Tekstcompressie (werkblad): http://csunplugged.org/text-compression/#You_Can_Say_That_Again
   - Werkblad: Japanse-puzzels
   - Plaatjes representatie en compressie uitleg: http://www.cs4fn.org/pixels/pixels.html
   - Lesactiviteit (film en werkblad): http://csunplugged.org/image-representation/
5. Constructies(keuze/herhaling)
   - Lesactiviteit (werkblad): http://teachinglondoncomputing.org/resources/inspiring-unplugged-classroom-activities/the-imp-computer-activity/
6. Variabelen (behandelen vóór opdracht 5)
   - Variabelen
     Activiteit: interactief oefenen met variabelen
     lesinformatie: http://teachinglondoncomputing.org/resources/inspiring-unplugged-class
     the-box-variable-activity/
   - Toekennen waarden aan variabelen
     Activiteit: op papier (Java) oefenen met variabelen en toekenning lesinformatie: https://teachinglondoncomputing.files.wordpress.com/2014/05/activity-assignmentdryrunj
     pdf
     film om te laten zien fragment 0:20 - 1:27) https://www.youtube.com/watch?v=aeoGGabJhAQ
   - Voorbereidend op Greenfoot opdracht 5:
     Activiteit: zoek het maximum aantal eieren uit een rij
     Lesinformatie (voorbeeld filmpje frament v.a. 36:10): http://cs50.tv/2014/fall/lectures/3/m/
7. Lijsten en sorteren:
   - Lesactiviteit (film en werkblad): http://csunplugged.org/sorting-algorithms/
   - Filmpje waarin sorteeralgoritmes worden uitgebeeld en vergeleken: https://www.youtube.com/watch?v=aXXWXz5rF64

- Filmpje met uitleg over beslisbaarheid `https://www.youtube.com/watch?v=92WHN-pAFCs`
- Lesactiviteit (film en werkblad): `http://csunplugged.org/sorting-networks/`

8. NP-complete problemen:
   - Kortste-route algoritmes (en NP-complete problemen): OOP_lecture5.ppt
   - Lesactiviteit kaart met minimaal aantal kleuren (werkblad): `http://csunplugged.org/graph-colouring/`
   - Lesactiviteit kortste netwerk aanleggen (werkblad): `http://csunplugged.org/minimal-spanning-tr`
   - Lesactiviteit hoe het minst aantal brievenbussen te plaatsen (werkblad): `http://csunplugged.org/dominating-sets/`

5.3 Verdiepings/uitbreidingsmogelijkheden

De volgende onderwerpen komen in het cursus niet aan bod, maar kunnen mogelijk door de docent toegevoegd worden als (differentiërende) verdiepingsslag:

- JavaDoc
- Recursief denken / programmeren
- Concurrency / parallellisme
- Projecten en teams: samenwerking, planning (b.v. Gantt charts), scrum, (project, ontwerp, test, javadoc, gebruikershandleidingen) documentatie, fasen (van requirements analyse tot deployment)
- RUP modellen: sequence diagrammen / collaboration diagrams
- Pre-planning/pre-ordering en caching/pre-fetching (Building libraries of pre-formed elements for future use)
- Pseudocode analyseren en opstellen om een algoritme te representeren
- Complexiteit van een algoritme berekenen
- Operatoren: mod/div
- Andere programmeertalen. Nadelen van OO programmeren (b.v. voor kleine problemen)
- Arrays, queues

6 Vragen of suggesties

We werken met enthousiasme nog met regelmaat aan ons materiaal. Tips en suggesties zijn van harte welkom! Wil je het materiaal gebruiken? Neem per email contact met ons op voor de nieuwste scenario's. Heb je nog andere vragen of opmerkingen, neem dan ook contact met ons op via renske.weeda @gmail.com.