slo nationaal expertisecentrum leerplanontwikkeling

Handreiking schoolexamen informatica

Deel 1 Kerndomeinen

Januari 2018

Erik Barendsen
Victor Schmidt
Erik Woldhuis

Inhoudsopgave

Ten Geleide 4

1 Domein A: Vaardigheden 5

1.1 Eindtermen 5

1.2 Toelichting op het domein 5

1.3 Ontwerpen en ontwikkelen 6

1.4 Informatica hanteren als perspectief 6

1.5 Samenwerken en interdisciplinariteit 6

1.6 Ethisch handelen 6

1.7 Informatica-instrumentarium hanteren 7

1.8 Werken in contexten 7

1.9 Voorbeeldspecificaties 7

2 Domein B: Grondslagen 9

2.1 Eindtermen 9

2.2 Toelichting op het domein 9

2.3 Algoritmen 9

2.4 Datastructuren 11

2.5 Eindige Automaten 13

2.6 Grammatica's 14

2.7 Voorbeeldspecificaties 16

3 Domein C: Informatie 18

3.1 Eindtermen 18

3.2 Toelichting op het domein 18

3.3 Doelstellingen, identificeren en representeren 18

3.4 Standaardrepresentaties 20

3.5 Gestructureerde data 21

3.6 Voorbeeldspecificaties 21

4 Domein D: Programmeren 23

4.1 Eindtermen 23

4.2 Toelichting op het domein 23

4.3 Ontwikkelen, inspecteren en aanpassen 23

4.4 Voorbeeldspecificaties 24

5 Domein E: Architectuur 26

5.1 Eindtermen 26

5.2 Toelichting op het domein 26

5.3 Decompositie 26

5.4 Security 27

5.5 Voorbeeldspecificaties 28

6 Domein F: Interactie 30

6.1 Eindtermen 30

6.2 Toelichting op het domein 30

6.3 Usability 30
6.4 Maatschappelijke aspecten 30
6.5 Privacy 31
6.6 Security 31
6.7 Voorbeeldspecificaties 31
Bronnen 33

Ten Geleide

Deze versie van de handreiking voor het schoolexamen in de kerndomeinen van het vak informatica heeft tot doel feedback te ontvangen van belangstellenden en betrokkenen. Ze wordt in de vorm van een tekstdocument gepubliceerd; na een feedbackronde wordt de definitieve versie gepubliceerd op http://handreikingschoolexamen.slo.nl in de vorm van een aantal gelinkte webpagina's.

De handreiking geeft suggesties en ideeën voor de inrichting en vormgeving van het schoolexamen van de zes kerndomeinen van het vak informatica, te weten:

A. Vaardigheden
B. Grondslagen
C. Informatie
D. Programmeren
E. Architectuur
F. Interactie

In deze versie worden bij elk van de domeinen de eindtermen uit het examenprogramma vermeld, gevolgd door een korte toelichting op het gehele domein – meestal afkomstig uit het adviesrapport (Barendsen & Tolboom, 2016). Vervolgens worden begrippen en concepten uit het domein nader toegelicht. Een domeinbeschrijving wordt afgesloten met enkele voorbeeldspecificaties. Daarin staat hoe de eindtermen nader gespecificeerd kunnen worden en staan bij sommige specificaties suggesties voor het niveau of nadere uitwerkingen. In de definitieve versie op de website worden deze onderdelen met elkaar gelinkt en worden daar gaandeweg voorbeelden van lessen en toetsen toegevoegd. Bovendien bevat de definitieve versie te zijner tijd beschrijvingen van de twaalf keuzedomeinen.

Wellicht ten overvloede zij gemeld dat de gehele inhoud van deze handreiking met uitzondering van de eindtermen van het examenprogramma géén voorschrijvend karakter heeft. Het staat scholen vrij eigen keuzen te maken, mits die niet in tegenspraak zijn met eindtermen en mits eindtermen niet genegeerd worden.

1 Domein A: Vaardigheden

1.1 Eindtermen

Subdomein A1: Ontwerpen en ontwikkelen
De kandidaat kan:
- in een context mogelijkheden zien voor het inzetten van digitale artefacten;
- deze mogelijkheden vertalen tot een doelstelling voor ontwerp en ontwikkeling, daarbij technische factoren, omgevingsfactoren en menselijke factoren betrekken;
- wensen en eisen specificeren en deze op haalbaarheid toetsen;
- een digitaal artefact ontwerpen;
- bij het ontwerp van een digitaal artefact keuzes afwegen via onderzoeken en experimenteren;
- een digitaal artefact implementeren;
- de kwaliteit van digitale artefacten evalueren,

en deze vaardigheden in samenhang inzetten voor het ontwikkelen van digitale artefacten.

Subdomein A2: Informatica hanteren als perspectief
De kandidaat kan in contexten:
- verschijnselen duiden, uitleggen en verklaren in termen van informatica;
- informatica-concepten herkennen en met elkaar in verband brengen;
- mogelijkheden en beperkingen van digitale artefacten inschatten en beredeneren in vaktermen.

Subdomein A3: Samenwerken en interdisciplinariteit
De kandidaat kan:
- bij het ontwerpen en ontwikkelen van digitale artefacten op een gestructureerde wijze samenwerken in een team;
- samenwerken met mensen afkomstig uit een toepassingsgebied.

Subdomein A4: Ethisch handelen
De kandidaat kan:
- beschrijven welke ethische normen en waarden een rol spelen bij het gebruik en de ontwikkeling van digitale artefacten;
- het eigen handelen expliciet vergelijken met ethische richtlijnen;
- *(vwo:) het eigen handelen kritisch analyseren en relateren aan ethische dilemma's.*

Subdomein A5: Informatica-instrumentarium hanteren
De kandidaat kan voor de informatica relevante gereedschappen hanteren, waar nodig met aandacht voor risico's en veiligheid. Daarbij gaat het om (computer)apparatuur, besturingssystemen, applicaties, vaktaal, vakconventies en formalismen.

Subdomein A6: Werken in contexten
De kandidaat kan de in domein A genoemde vaardigheden en de in domeinen B tot en met F, en in de gekozen domeinen uit G tot en met R, genoemde concepten tenminste gebruiken in beroepscontexten, in maatschappelijke contexten en *(vwo:) in wetenschappelijke contexten*.

1.2 Toelichting op het domein
In dit domein worden generieke (dat wil zeggen: conceptoverstijgende) vaardigheden beschreven. De gedachte is dat kandidaten beheersing van deze vaardigheden aantonen in combinatie met leerinhouden uit de kern- en/of de keuzedomeinen. Drie vaardigheden zijn cruciaal en karakteristiek voor informatica als wetenschaps- en vakgebied en voor het schoolvak informatica in het bijzonder: (1) ontwerpen en ontwikkelen, (2) informatica hanteren als perspectief, en (3) samenwerken en interdisciplinariteit. Daarnaast worden in het examenprogramma drie andere vaardigheden onderscheiden die van belang zijn voor het schoolvak.

1.3 Ontwerpen en ontwikkelen

Informatica wordt door velen gezien als een construerende discipline: een vakgebied waarin het maken van dingen centraal staat. Die dingen zijn in het algemeen digitale artefacten. In deze visie levert het wetenschapsgebied informatica kennis over zulke artefacten en het maakproces. Deze kennis bestaat uit een conceptuele basis en uit typische denk- en werkwijzen.

Het maken van digitale artefacten is veel meer dan een technische aangelegenheid. Informatica dient immers niet alleen voor automatisering van bestaande en bekende processen. De wereld om ons heen is doordrongen van informaticatoepassingen die ook dienen om het dagelijks leven makkelijker of leuker te maken (bijvoorbeeld: navigatiesystemen, games, gezichtsherkenning in een fotoverzameling), veiliger te laten zijn (bijvoorbeeld: robots, botsingsdetectie) en om nieuwe mogelijkheden te scheppen (bijvoorbeeld: spraakherkenning, kunstmatige intelligentie). Bij ontwerp en ontwikkeling staan contexten en gebruikers dan ook centraal. Die contexten bieden mogelijkheden om aan te sluiten bij de belangstelling en het profiel van leerlingen, en zo ontwerpen en ontwikkelen tot een creatieve uitdaging te maken.

1.4 Informatica hanteren als perspectief

Naast het maakperspectief heeft informatica ook een analytische kant: met kennis van zaken kunnen we verschijnselen in het dagelijks leven en de maatschappij duiden en redeneringen en oplossingen op waarde schatten. In een wereld waarin informatica zo in elk facet is doorgedrongen, is een dergelijke vaardigheid onontbeerlijk. Een leerling die het schoolvak heeft afgesloten, zal niet alleen trefzeker kunnen redeneren over mogelijke oorzaken van het haperen van het wifi-netwerk in huis, maar ook kunnen uitleggen of we ons zorgen moeten maken over ons banktegoed als een site voor internetbankieren niet beschikbaar is vanwege een DDoS-aanval.

Deze eindterm bevat elementen van **computational thinking**. Deze term werd geïntroduceerd door Jeannette Wing (2006) om een verzameling mentale gereedschappen aan te duiden die nodig zijn om computers effectief in te kunnen zetten. Hiertoe behoren analytische vaardigheden om problemen zodanig kunnen formuleren dat we computers en andere gereedschappen kunnen gebruiken om ze te helpen oplossen, en ook probleemoplossend vermogen, zoals het zoeken van oplossingen in termen van algoritmen en gegevens. Wing (2006) en vele anderen zien computational thinking als een basisvaardigheid naast lezen, schrijven en rekenen. Onder deze eindterm worden informaticaconcepten gebruikt als bril om naar de wereld te kijken. Deze vaardigheid past daarom bij het analytische aspect van computational thinking, terwijl Ontwerpen en ontwikkelen aansluit bij het aspect van probleemoplossen.

1.5 Samenwerken en interdisciplinariteit

Een informaticus werkt zelden alleen. Het maken van digitale producten gebeurt meestal in een team: enerzijds omdat die producten vaak groot zijn en vele handen maken licht werk maken, anderzijds omdat het ontwikkelen de bijdrage van verschillende specialismen uit de informatica vergt. Kunnen werken in teamverband is daarom een essentiële vaardigheid. De informatica (met name het deelgebied **software engineering**) heeft allerlei methoden opgeleverd om gestructureerd in een ontwikkelteam te werken. Daaronder worden traditionele watervalmethoden gerekend, maar ook meer 'agile' aanpakken, waaronder Scrum. Voor het onderwijs is de specifieke variant onder de naam eduScrum ontwikkeld.

1.6 Ethisch handelen

Bij toepassing van informatica en het ontwikkelen van digitale artefacten spelen ethische normen (zoals beroepscodes) en ethische dilemma's een rol. De Vereniging van Register Informatici VRI kent een gedragscode met vijf uitgangspunten en tien gedragsregels (VRI, 2015). Deze gedragsregels gaan over bewustzijn van de gevolgen van handelen voor de samenleving, handelen in overeenstemming met het belang van de opdrachtgever, vertrouwelijk omgaan met gegevens die onder ogen komen en niet meewerken aan ontwikkeling van digitale artefacten die tot doel hebben personen of instellingen te schaden dan wel illegaal zijn. De Association for Computer Machinery (ACM, 1992) kent een Code of Ethics and Professional Conduct met soortgelijke gedragsregels.

1.7 Informatica-instrumentarium hanteren

Informatici zijn naast makers ook vaardige gebruikers van digitale hulpmiddelen. In het subdomein Informatica-instrumentarium hanteren wordt deze vaardigheid beschreven, evenals het hanteren van informaticaspecifieke termen en formalismen. Onder informatica-instrumentarium kunnen IDE's, compilers, database-engines, ontwikkeltools en computerapparatuur gerekend worden. Ook correct gebruik van vaktermen en conventies voor bijvoorbeeld het noteren van flowcharts, eindige automaten of grammatica's wordt tot deze eindterm gerekend. Voorbeelden van formalismen in de informatica ten slotte zijn eindige automaten, grammatica's en lambdacalculus. Dit subdomein is een informaticaspecifieke variant van soortgelijke subdomeinen in de bètavakken, waarin bijvoorbeeld laboratoriumvaardigheden en formulegebruik worden beschreven.

1.8 Werken in contexten

Ten slotte dienen kandidaten informaticaconcepten in diverse maatschappelijke contexten kunnen gebruiken. Ook beroepscontexten zijn belangrijk, waarbij voor vwo en havo goed kan worden aangesloten bij beroepen waarvoor een wetenschappelijke respectievelijk een beroepsopleiding is vereist. Vwo'ers komen tevens in aanraking met wetenschappelijke contexten.

1.9 Voorbeeldspecificaties

A1 Ontwerpen en ontwikkelen

De kandidaat kan:

- in een context mogelijkheden zien voor het inzetten van digitale artefacten;
- deze mogelijkheden vertalen tot een doelstelling voor ontwerp en ontwikkeling, daarbij technische factoren, omgevingsfactoren en menselijke factoren betrekken;
    - In havo kan de doelstelling van het digitale artefact gegeven zijn.
- wensen en eisen specificeren en deze op haalbaarheid toetsen;
- een digitaal artefact ontwerpen;
- bij het ontwerp van een digitaal artefact keuzes afwegen via onderzoeken en experimenteren;
- een digitaal artefact implementeren;
    - Implementatie hoeft niet van scratch plaats te vinden; samenstelling van bestaande componenten tot een digitaal artefact wordt eveneens tot deze eindterm gerekend.
- de kwaliteit van digitale artefacten evalueren;
    - Het gestructureerd testen van een digitaal artefact aan de hand van een testplan wordt hier niet bedoeld.

en deze vaardigheden in samenhang inzetten voor het ontwikkelen van digitale artefacten.

A2 Informatica hanteren als perspectief

De kandidaat kan in contexten:

- verschijnselen duiden, uitleggen en verklaren in termen van informatica;
- informaticaconcepten herkennen en met elkaar in verband brengen;
- mogelijkheden en beperkingen van digitale artefacten inschatten en beredeneren in vaktermen.

A3 Samenwerken en interdisciplinariteit

De kandidaat kan:

- bij het ontwerpen en ontwikkelen van digitale artefacten op een gestructureerde wijze samenwerken in een team;
    - Hij kan één zo'n gestructureerde werkwijze hanteren. Meer werkwijzen zijn niet noodzakelijk.
    - Hij kan samenwerken met teamleden die zich een specialisme eigen gemaakt hebben. In de schoolsituatie kunnen dat teamleden zijn die een verdiepend keuzethema gevolgd hebben.
- samenwerken met mensen afkomstig uit een toepassingsgebied.

A4 Ethisch handelen

De kandidaat kan:

- beschrijven welke ethische normen en waarden een rol spelen bij het gebruik en de ontwikkeling van digitale artefacten;
- het eigen handelen expliciet vergelijken met ethische richtlijnen;
- *(vwo:) het eigen handelen kritisch analyseren en relateren aan ethische dilemma's.*

A5 Informatica-instrumentarium hanteren

De kandidaat kan voor de informatica relevante gereedschappen hanteren, waar nodig met aandacht voor risico's en veiligheid. Daarbij gaat het om (computer)apparatuur, besturingssystemen, applicaties, vaktaal, vakconventies en formalismen.

- Gedetailleerde kennis van (gebruik en beheer van) computerapparatuur en besturingssystemen wordt niet voorgesteld. Het gaat er om dit informatica-instrumentarium te kunnen gebruiken bij uitvoering van opdrachten die deel uit maken van een schoolleerplan.
- Van welk niveau formalismen gehanteerd moeten worden, hangt af van welke keuzedomeinen een kandidaat kiest. Een generiek niveau voor het gebruik van formalismen wordt niet voorgesteld.

A6 Werken in contexten

De kandidaat kan vaardigheden en concepten gebruiken in contexten.

- Contexten kunnen afkomstig zijn uit beroepen en uit de samenleving.
- *Voor vwo-kandidaten zijn contexten uit wetenschap eveneens geschikt.*

2 Domein B: Grondslagen

2.1 Eindtermen

Subdomein B1: Algoritmen
De kandidaat kan een oplossingsrichting voor een probleem uitwerken tot een algoritme, daarbij standaardalgoritmen herkennen en gebruiken, en de correctheid en efficiëntie van digitale artefacten onderzoeken via de achterliggende algoritmen.

Subdomein B2: Datastructuren
De kandidaat kan verschillende abstracte datastructuren met elkaar vergelijken op elegantie en efficiëntie.

Subdomein B3: Automaten
De kandidaat kan eindige automaten gebruiken voor de karakterisering van bepaalde algoritmen.

Subdomein B4: Grammatica's
De kandidaat kan grammatica's hanteren als hulpmiddel bij de beschrijving van talen.

2.2 Toelichting op het domein

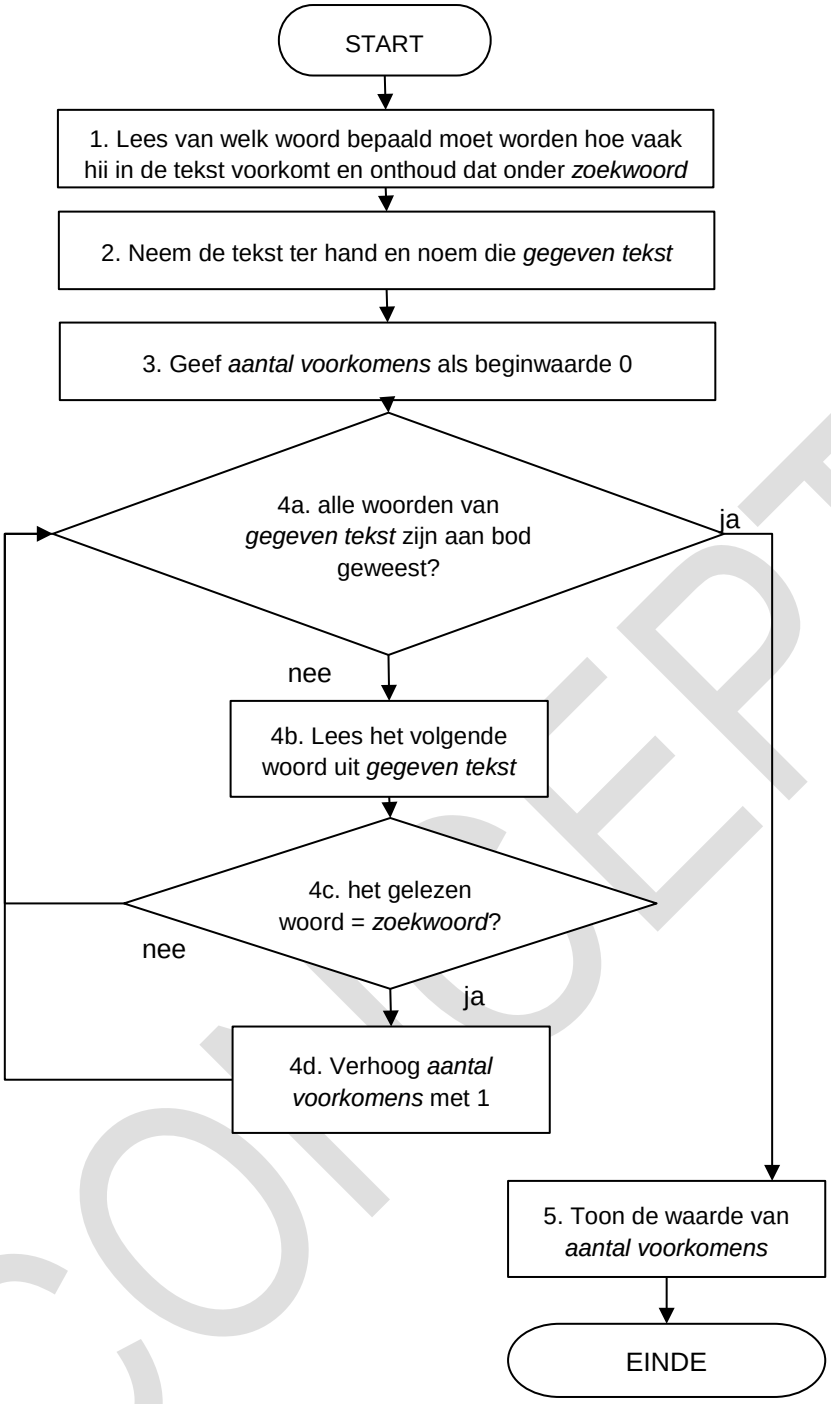
Doel
In dit domein worden digitale artefacten op een abstracte manier bekeken, dat wil zeggen onafhankelijk van een concrete implementatie. Op dit abstracte niveau kunnen we redeneren over de efficiëntie van een oplossing nog vóór er een programma in een concrete programmeertaal is geschreven. Ook wordt het redeneren over een bestaand programma gemakkelijker als we abstraheren van de implementatiedetails en kijken naar het achterliggende algoritme.

Korte beschrijving
In dit domein is het dan ook vooral de bedoeling dat algoritmen, datastructuren, eindige automaten en grammatica's worden gehanteerd als tussenstap tussen een oplossingsrichting en een digitaal artefact. De kandidaat kan deze stap beiden kanten op zetten. Een kandidaat kan een algoritme opstellen, een datastructuur kiezen et cetera, uitgaande van een oplossingsrichting maar ook een bestaand digitaal artefact modelleren met eindige automaten, algoritmen et cetera. Om hiertoe in staat te zijn, moet een kandidaat een zelf opgesteld of gegeven algoritme, datastructuur, eindige automaat of grammatica kunnen analyseren op bijvoorbeeld correctheid en efficiëntie. Dit met als doel het beoordelen van de haalbaarheid van een oplossingsrichting zonder dat deze eerst geïmplementeerd hoeft te worden.

2.3 Algoritmen

Een **algoritme** is een serie acties (of instructies) en beslissingen met een bepaald doel. De uitvoerder van deze instructies hoeft geen computer te zijn, maar kan ook een mens zijn. Zo zou je een partituur kunnen zien als algoritme voor een muziekstuk en een recept als een algoritme voor een gerecht. Beide worden uitgevoerd door een mens. Basisbouwstenen voor algoritmen zijn **opeenvolging**, **keuze** en **herhaling**. Als een algoritme zichzelf aanroept, dan heet dat **recursie**.

Hieronder staat een voorbeeldalgoritme om te bepalen hoe vaak een bepaald woord in een gegeven tekst voorkomt.



Figuur 1: Voorbeeld van een algoritme dat telt hoe vaak een woord in een tekst voorkomt

Over het voorbeeld:

- Het algoritme begint met een **opeenvolging** van instructies, stap 1, 2 en 3. Het algoritme doet hier altijd hetzelfde; onafhankelijk van de omstandigheden.
- Stap 4a tot en met 4d zijn onderdeel van een **herhaling**: zij worden meerdere malen uitgevoerd. Het aantal herhalingen hangt in dit voorbeeld af van het aantal woorden in de tekst, maar kan in het algemeen ook een van tevoren bepaald aantal zijn.
- Zowel stap 4a als 4c zijn **keuzen**: afhankelijk van de omstandigheden wordt gekozen welke volgende stap wordt doorlopen.
- Het algoritme is weergegeven als een **flowchart**, met instructies in blokken en pijlen tussen de blokken die de opeenvolging aangeven. In dit flowchart komen drie symbolen voor uit de norm ISO 5807:1985. Een algoritme kan ook in **pseudocode** worden weergegeven:

herhalings- en keuzestructuren worden dan in woorden beschreven die doen denken aan een programmeertaal, bijvoorbeeld “ZOLANG *voorwaarde* geldt, voer *instructies* uit.
Uit onderzoek (….) blijkt dat flowcharts didactische voordelen hebben. Bovendien is het in het geval van pseudocode nodig om afspraken te maken over de formulering van instructies, terwijl de blokken in de flowcharts teksten kunnen bevatten op een willekeurig niveau van abstractie (een concrete handeling of een complexere taak). Dit maakt flowcharts ruimer inzetbaar: zowel voor globaal ontwerp als voor weergave van gedetailleerde instructies.

- In de flowchart worden enkele objecten aangeduid met een gekozen naam (gegeven tekst, aantal voorkomens). Zulke **benoemde objecten** in een algoritme zien we in een programma meestal terug als variabelen.

Correctheid
Een algoritme is **correct** als bij iedere invoer de ‘juiste’ uitvoer oplevert. De bedoeling van een algoritme (de beoogde relatie tussen in- en uitvoer) kan worden uitgedrukt door een combinatie van aannamen over de invoer (ook wel precondities genoemd) en eigenschappen van de uitvoer (postconditie). Bijvoorbeeld: “als de invoer een niet-lege rij van gehele getallen is, dan is de uitvoer het gemiddelde van de getallen in die rij.”
De correctheid van een algoritme kan op verschillende manieren aannemelijk worden gemaakt of zelfs bewezen. Dat laatste valt buiten het bereik van het examenprogramma. Testen is een veelgebruikte manier om de correctheid te onderzoeken (door fouten op te sporen), maar met testen is ongeschikt om de correctheid van een programma waterdicht aan te tonen.

Efficiëntie
Net als programma’s kunnen algoritmen onderling vergeleken worden aan de hand van de (verwachte) rekentijd bij gegeven invoer. Meestal heeft een sneller (“efficiënter”) algoritme de voorkeur. Een alternatief voor het hierboven beschreven algoritme is om de tekst eerst woord voor woord op te slaan om deze daarna pas te doorzoeken. Alle woorden worden dan twee keer gelezen en het algoritme is dus langer bezig. Dit nieuwe algoritme is nog steeds correct, maar minder efficiënt dan het voorbeeldalgoritme
De rekentijd kan in absolute zin worden beschouwd, maar interessanter is het om te kijken naar de rekentijd in relatie tot de grootte van de invoer, ook wel de **complexiteit** van een algoritme of programma genoemd. Welk effect heeft een verdubbeling van de invoer op de rekentijd, bijvoorbeeld? Hierop gaat het keuzethema G dieper in.

2.4 Datastructuren

Een **datastructuur** is een manier om gegevens te ordenen. Deze manier van ordenen heeft consequenties voor de efficiëntie van algoritmen die met deze gegevens werken.

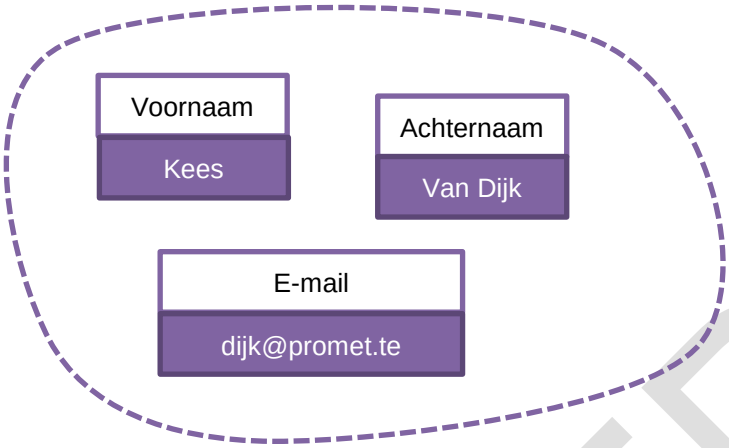
Datastructuren zijn dus *ordeningsprincipes*. We bespreken hieronder een aantal belangrijke datastructuren: tupels, records, rijen, lijsten en bomen. Als we zulke datastructuren toepassen op concrete verzamelingen gegevens (zogenaamde elementaire datatypen) zoals getallen of letters, dan ontstaan nieuwe (samengestelde) **datatypen**, bijvoorbeeld: ‘rij van getallen’, ‘lijst van bomen van letters’, enzovoort. Zo kan het datatype ‘woord’ worden uitgedrukt als ‘rij van letters’.
Programmeertalen bevatten meestal een aantal elementaire datatypen en taalconstructies voor datastructuren.

Tupels
Een **tupel** of n-tupel is een combinatie van *n* gekoppelde gegevens die tezamen een betekenis hebben. Coördinaten op het aardoppervlak - lengte- en breedtegraad - vormen bijvoorbeeld een 2-tupel. Het tijdstip 12:09:10 uur, kan als volgt als een 3-tupel voorgesteld worden.

12	**9**	**10**

Van een tupel kun je de afzonderlijke gegevens opvragen en uit *n* gegevens van het juiste type een *n*-tupel construeren. Afhankelijk van het tupel zijn er ook andere bewerkingen mogelijk, zoals het optellen van tupels van gelijke lengte gevuld met getallen.

Records
Een record is een combinatie van een eindig aantal gegevens, net als een tupel. De velden hebben echter geen vaste volgorde, maar worden onderscheiden via hun *veldnamen*, bijvoorbeeld een record van drie velden met namen ‘voornaam’, ‘achternaam’ en ‘email’. Een record met deze velden kan als volgt voorgesteld worden.

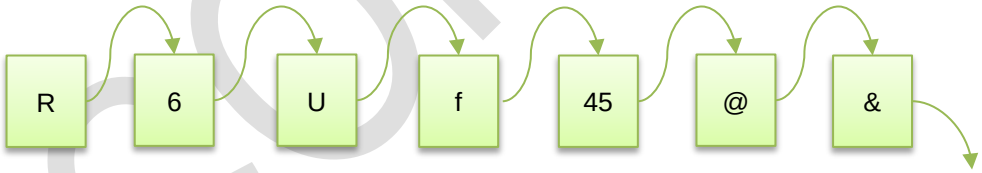


Figuur 2: Voostelling van een record met voornaam, achternaam en e-mailadres

Rijen
Een **rij** is een verzameling van gegevens die een volgnummer of index hebben. Een rij heeft een vaste lengte. Elk gegeven kan door middel van zijn index aangeduid worden. Het voorbeeld hieronder is een rij van lengte 8 met volgnummers 0 t/m 7.

index	0	1	2	3	4	5	6	7
gegeven	12	23	4	0	15	9	2	11

Lijsten
Een **lijst** is een serie gegevens die (afgezien van het laatste element in de lijst) steeds naar een volgend lijstelement verwijzen, zoals in onderstaande figuur te zien is.



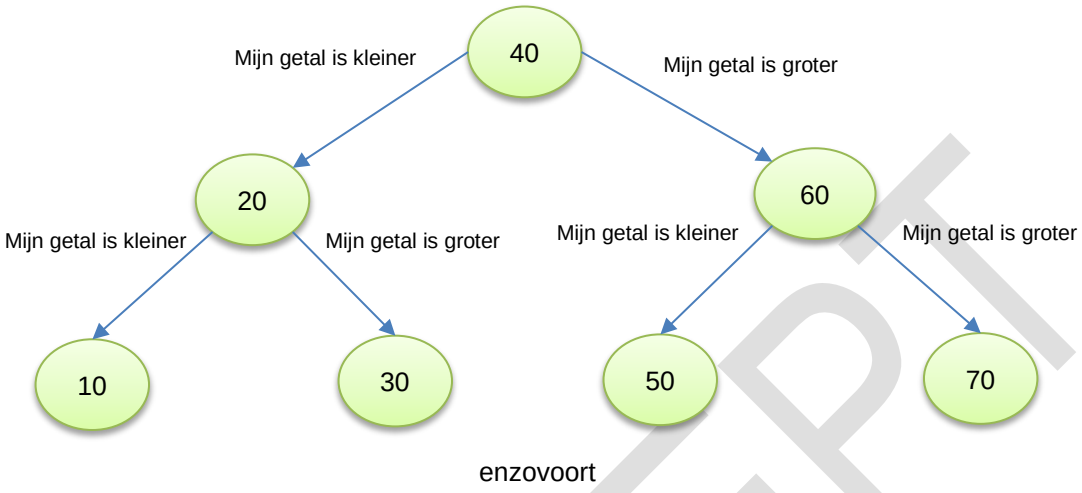
Figuur 3: Voorstelling van een lijst met zeven elementen

Het aantal elementen in een lijst is willekeurig. De elementen hebben geen index, zoals dat in een rij wel het geval is.

Bomen
In een rij en een lijst is er één vaste volgorde van elementen: je kunt spreken van bijvoorbeeld een eerste, een volgende of een laatste element. Bij **bomen** kan één element juist verwijzen naar meer dan één andere elementen. De elementen in een boom worden knopen genoemd, en de verwijzingsrelaties takken. Elke knoop kan dus meer dan één tak hebben. Een knoop zonder inkomende takken wordt een wortel genoemd; knopen zonder uitgaande takken worden bladeren genoemd.

Bomen worden vaak gebruikt voor zoek-algoritmen. Onderstaande boom representeert bijvoorveeld het zoeken naar een antwoord in een raadspelletje, waarbij iemand (A) een positief geheel getal onder

een gegeven bovengrens in gedachten heeft en de tegenstander (B) dat getal moet raden. Bij elke raadpoging van *B* geeft *A* informatie: 'mijn getal is groter dan wat je geraden hebt', 'mijn getal is kleiner dan wat je geraden hebt' of 'je hebt het getal geraden'. Stel dat de bovengrens gelijk is aan 80, dan kan een algoritme voor het raadproces goed vormgegeven worden aan de hand van onderstaande boom: je begint bovenaan met raden en afhankelijk van het antwoord (‘groter’ of ‘kleiner’) kies je de linker of de rechter tak.



Figuur 4: Voorstelling van een boom ten behoeve van een raadspelletje

Efficiëntie
De **efficiëntie** van een datastructuur is de mate waarin het een efficiënt, dus snel, algoritme mogelijk maakt. Bij de keuze tussen rij, lijst en boom kunnen bijvoorbeeld de volgende overwegingen een rol spelen:

- Bomen lenen zich goed voor het vlot zoeken en vinden van individuele gegevens, met name als er veel gegevens zijn. Toevoegen en verwijderen van gegevens gaat moeizaam, omdat dan de hele boom geherstructureerd moet worden.

- Lijsten zijn handig als er in een algoritme vaak gegevens tussengevoegd of verwijderd moeten worden. Zoek- en vindopdrachten kosten relatief veel inspanning als het aantal gegevens groot is: de lijst moet van voor naar achteren langs gelopen worden.

- Het voordeel van het gebruik van een lijst is juist het nadeel van gebruik van een rij en omgekeerd. Als er gegevens in een rij tussengevoegd moeten worden, moet er eerst een vrije plek gemaakt worden door alle gegevens die verderop staan, een plaats op te schuiven. Bij een verwijdering moeten juist alle verder gelegen gegevens een plaats teruggeschoven worden. Daar staat tegenover dat het met behulp van de index relatief eenvoudig is een gegeven dat op een bepaalde plek staat te benaderen.

Elegantie
Evenzo kunnen we niet over **elegantie** van een datastructuur op zichzelf spreken – zoiets is meer van toepassing op de combinatie van oplossingsmethode (algoritme) en een bijbehorende, listig gekozen datastructuur.

2.5 Eindige Automaten

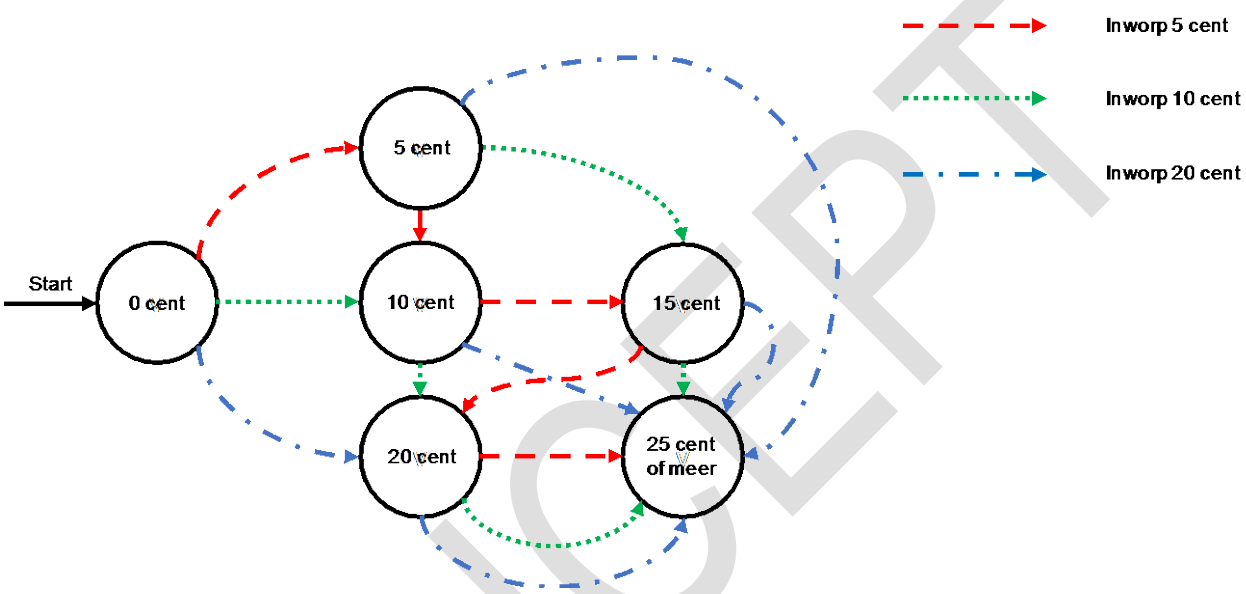
Eindige automaten worden vaak gebruikt om het gedrag van een systeem te beschrijven.

Een **eindige automaat** is een systeem met
- een eindig aantal **toestanden,** waaronder een begintoestand
- mogelijk een of meer eindtoestanden

- een aantal regels die beschrijven tussen welke toestanden er een **overgang** mogelijk is en eventueel onder welke voorwaarden

Een verkeerslicht in Nederland kent bijvoorbeeld drie toestanden: 'rood' (de begintoestand), 'oranje' en 'groen' en drie mogelijke overgangen: 'rood' → 'groen', 'groen'→ 'oranje' en 'oranje' → 'rood'. In deze eindige automaat is er geen eindtoestand.

Vaak zijn bij een eindige automaat vanuit een toestand meerdere overgangen mogelijk en bepaalt de input welke overgang plaatsvindt, zoals in onderstaande voorbeeld van de muntinvoer van een snoepautomaat voor snoepjes van 25 cent zonder wisselgeld. De gekozen overgang hangt steeds af van de grootte van de inworp. Deze eindige automaat heeft de toestand '25 cent of meer' als eindtoestand.



Figuur 5: Snoepuitgifte in de vorm van een eindige automaat

In dit voorbeeld zijn alle overgangen **deterministisch**: bij een gegeven input kan maar één overgang plaatsvinden (met 100% kans). In een nondeterministische eindige automaat spelen kansprocessen een rol: er zou bijvoorbeeld bij iedere muntinworp een kans kunnen zijn dat de munt niet geregistreerd wordt en de toestand niet verandert.

Als een (gewenst) systeem als eindige automaat is beschreven, kan vervolgens aan de hand van die beschrijving een algoritme opgesteld worden.

2.6 Grammatica's

Een **grammatica** is een set regels waarmee woorden gevormd kunnen worden uit basisbouwstenen. Die basisbouwstenen worden meestal 'alfabet' genoemd.

- De elementen van een **alfabet** hoeven niet per se de 26 letters uit het westerse alfabet te zijn; het kunnen ook getallen of woorden zijn, zoals de gereserveerde woorden van een programmeertaal (while, if, else, enzovoorts); of iets heel anders zoals kleuren.
- Een **woord** is een combinatie van letters die volgens de regels van de grammatica achterelkaar zijn gezet.

Als we het over de grammatica van programmeertalen hebben dan bestaat het alfabet uit zowel woorden - onder andere woorden als *if*, *then*, *else*, *for* en *while* – als leestekens zoals *{*, *}*, +, *:* en =. De woorden van deze grammatica zijn dan statements zoals '*c = c + 1*' en '*if c == 0 then c = -3*'. Een uitdrukking als '*then c+= if 3 for*' betstaat wel uit letters uit het alfabet maar is geen woord omdat de volgorde niet geldig is.

Grammatica's worden meestal gedefinieerd aan de hand van **productieregels**: regels waarmee alle geldige woorden kunnen worden geproduceerd. Er zijn verschillende manieren om de productieregels van een grammatica weer te geven. Een veelgebruikt formalisme is de zogenaamd BN-vorm, bijvoorbeeld: Straatnaam ::= Symbool | Symbool; Straatnaam.

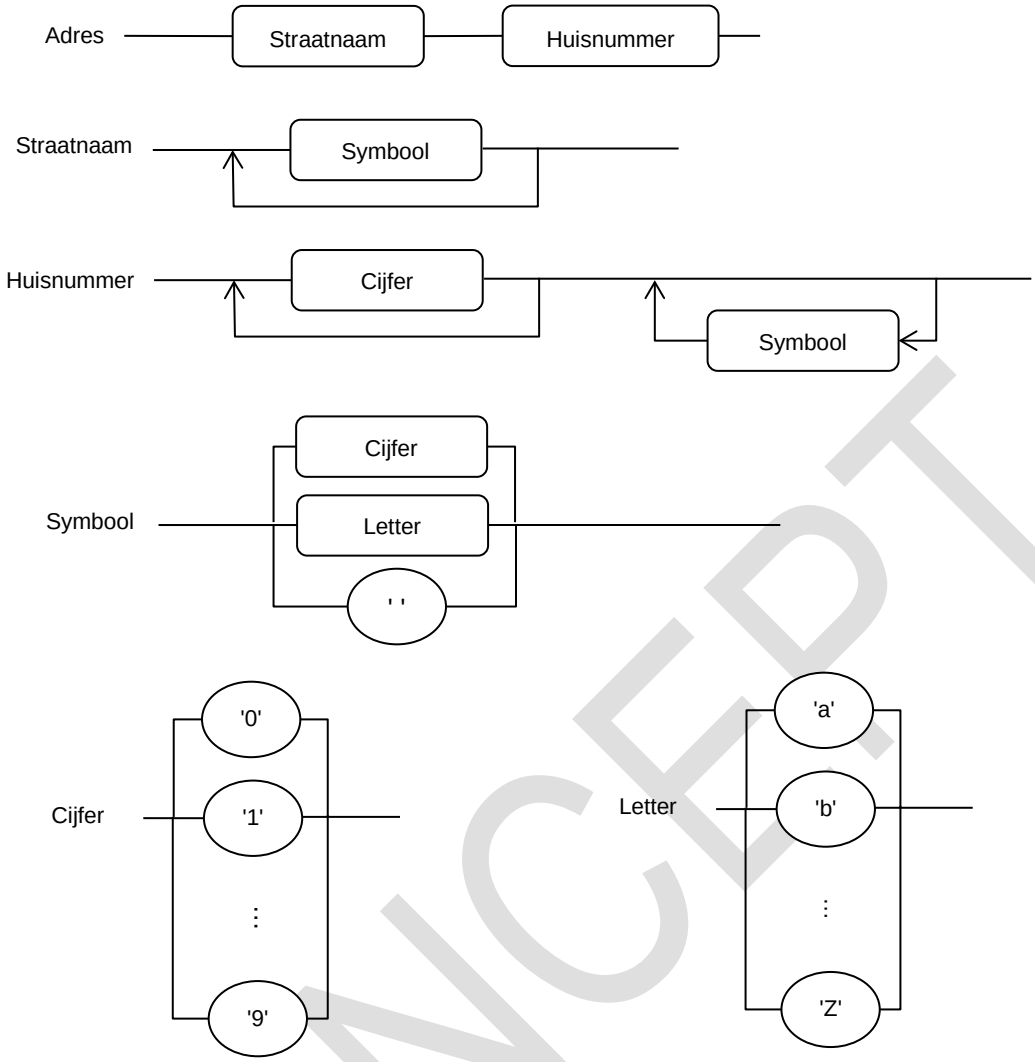
Grammatica's kunnen ook met behulp van diagrammen worden weergegeven: zie het volgende voorbeeld van een grammatica voor woonadressen. Woorden die aan deze grammatica voldoen zijn onder andere:

- Beursstraat 10 6
- Van Speijkstraat 12a
- 2e Willemstraat 02
- Postbus 24000

De volgende woorden kunnen niet door de grammatica worden gegenereerd en zijn dus (volgens de grammatica) ongeldige adressen:

- Beursstraat 10-6
- Van Speijkstraat a

In dit voorbeeld wordt het alfabet gevormd door de letters A tot en met Z, a tot en met z, de cijfers 0 tot en met 9 en de spatie. Straatnaam, Huisnummer, Symbool, Cijfer en Letter zijn hulpsymbolen om de grammatica mee te beschrijven. Ze behoren niet tot het alfabet en (dus) niet tot de woorden van de taal. Daarom worden ze soms 'niet-terminale symbolen' genoemd, en de symbolen van het alfabet, in analogie hiermee, 'terminale symbolen'.



Figuur 6: Een voorbeeld van een grammatica voor woonadressen

2.7 Voorbeeldspecificaties

B1 Algoritmen

De kandidaat kan:

- een gegeven oplossingsrichting voor een probleem weergeven als een algoritme. Hierbij wordt verwacht dat kandidaten een algoritme op een gestructureerde wijze kunnen weergeven met bijvoorbeeld een flowchart of in pseudocode.
- Het algoritme is opgebouwd uit de basisbouwstenen opeenvolging, keuze en herhaling.
    - *vwo: Het algoritme kan recursie bevatten*
- een gegeven digitaal artefact modelleren met behulp van een algoritme.
- het gedrag van een programma onderzoeken via het onderliggende algoritme, en zo problemen (bijvoorbeeld fouten of inefficiëntie) herleiden tot aspecten van dat algoritme.
- een aantal standaardalgoritmen herkennen en gebruiken.
    - Kandidaten kennen standaardalgoritmen voor elementaire numerieke operaties zoals minimum en maximum bepalen, sommeren, en ten minste twee andere doelen zoals zoeken, sorteren, datacompressie en graafalgoritmen (bijvoorbeeld routebepaling).
    - Kandidaten kennen bij tenminste één doel ten minste twee standaardalgoritmen.
- de correctheid van een gegeven algoritme onderzoeken, en algoritmen (waaronder standaardalgoritmen), vergelijken met betrekking tot efficiëntie.
    - Formele bewijzen van de correctheid van een algoritme, bijvoorbeeld met behulp van post- en precondities en invarianten van herhalingslussen, zijn niet noodzakelijk.

- Absolute uitspraken over efficiëntie van een algoritme in termen van complexiteitsklassen komen aan bod in **keuzedomein G**.

B2 Datastructuren

De kandidaat kan:

- meerdere (abstracte) datastructuren herkennen, onderling vergelijken en beoordelen op toepasbaarheid. Het gaat in elk geval om tupels, records, rijen en lijsten.
  - *vwo: ook bomen*.
  - Implementatie van een samengesteld datatype met behulp van datastructuren en elementaire datatypen in een programmeertaal maakt deel uit van **domein D**.
  - Efficiëntie en elegantie komen hierbij aan de orde in samenhang met algoritmen.

B3 Eindige automaten

De kandidaat kan:

- automaten gebruiken voor het karakteriseren van een digitaal artefact.
  - Hierbij kan de kandidaat ten minste de termen toestanden en overgang gebruiken. De overgangen zijn alleen deterministisch van aard.
  - *vwo: De kandidaat kan inschatten of een eindige automaat een geschikt instrument is om een bepaald digitaal artefact te karakteriseren.*
- bij een gegeven (deterministische) eindige automaat een algoritme construeren.

B4 Grammatica's

De kandidaat kan:

- grammatica's hanteren als hulpmiddel bij de beschrijving van talen.
  - De kandidaat kan het concept grammatica's uitleggen aan de hand van ten minste de termen alfabet, woord en productieregels.
  - De kandidaat kan bij een gegeven grammatica en gegeven woorden vaststellen of de woorden aan de grammatica voldoen.
  - *vwo: De kandidaat kan een gegeven grammatica aanpassen, bijvoorbeeld naar aanleiding van een uitbreiding van de taal.*
  - Formele logica is geen onderdeel van dit domein, maar maakt deel uit van **keuzedomein G**.

3 Domein C: Informatie

3.1 Eindtermen

Subdomein C1: Doelstellingen
De kandidaat kan doelstellingen voor informatie- en gegevensverwerking onderscheiden, waaronder *zoeken* en *bewerken.*

Subdomein C2: Identificeren
De kandidaat kan informatie en gegevens identificeren in contexten, daarbij rekening houdend met de doelstelling.

Subdomein C3: Representeren
De kandidaat kan gegevens representeren in een geschikte datastructuur, daarbij rekening houdend met de doelstelling en kan daarbij verschillende representaties met elkaar vergelijken op elegantie, efficiëntie en implementeerbaarheid.

SubdomeinC4: Standaardrepresentaties
De kandidaat kan standaardrepresentaties van numerieke gegevens en media gebruiken en aan elkaar relateren.

Subdomein C5: Gestructureerde data
De kandidaat kan een informatiebehoefte vertalen in een zoekopdracht op een verzameling gestructureerde data.

3.2 Toelichting op het domein

In dit domein komen de begrippen ‘gegevens’ en ‘informatie’ voor. Hoewel het onderscheid niet altijd scherp is, wordt met ‘gegevens’ doorgaans verwezen naar vastgelegde, in beginsel betekenisloze data. Informatie ontstaat door deze gegevens *betekenis* te geven, bijvoorbeeld wanneer een gebruiker of ontvanger deze interpreteert.

Het woord ‘informatie’ wordt dan ook vaak gehanteerd in de context van toepassingen en gebruikers, terwijl we het begrip ‘gegevens’ meestal gebruiken in de context van vastleggen en opslaan van data.

3.3 Doelstellingen, identificeren en representeren

Ter verdere verheldering gebruiken we twee voorbeelden.

Voorbeeld 1: het meerdaags popfestival
Elke zomer wordt op een centrale plek in Nederland een meerdaags popfestival georganiseerd. Je kunt dit festival bijwonen voor één dag, maar ook voor een aantal dagen achtereen. Wie het festival wil bezoeken, koopt een toegangskaart via Internet. Je opent een webpagina waarop je een aantal gegevens invult, zoals je naam, adres, woonplaats, je mailadres, met hoeveel personen je komt, wanneer je aankomt, wanneer je weer vertrekt en (als je langer dan een dag blijft) hoeveel tenten je meeneemt. Je betaalt het entreegeld met creditcard, iDeal of anderszins en als de betaling gelukt is, krijg je per mail een entreebewijs toegestuurd. Daarop staat een barcode met een uniek nummer.

Op de dag van aankomst kunnen bezoekers zich bij de toegangspoort melden. Daar wordt het entreebewijs gecontroleerd en word je toegang verschaft tot het festivalterrein. Voor elke dag dat je het festival bezoekt, krijg je een polsbandje. Elke dag heeft een andere kleur. Als je langer dan een dag blijft, kun je met je entreebewijs naar de festivalcamping gaan. Daar wordt je een plek met een nummer toegewezen, waar je je tent(en) kunt neerzetten. In verband met de veiligheid moet de festivalorganisatie op ieder moment aan de plaatselijke brandweer kunnen melden welke tentplaatsen er bezet zijn. Het feest kan beginnen!

Barcodenummer, naam, adres, woonplaats, mailadres, aantal personen, aankomstdatum, vertrekdatum, aantal tenten en toegewezen tentplaatsnummer(s) vormen in dit voorbeeld **gegevens**.

Deze worden opgeslagen en naar behoefte in een bepaalde weergave en/of na een bepaalde bewerking beschikbaar gesteld. **Informatie** wordt via het computersysteem onder andere verstrekt aan:

- de medewerkers van de toegangspoort. Zij moeten controleren dat er niemand zich onterecht toegang verschaft tot het festivalterrein en de bezoekers polsbandjes verstrekken. Daartoe zoekt het computersysteem de barcode op een entreebewijs en retourneert de bijbehorende aankomstdatum, vertrekdatum en het aantal personen. Op basis daarvan wordt de bezoekers voldoende polsbandjes van de juiste kleur uitgereikt.

- de medewerkers van de festivalcamping. Zij moeten controleren of iemand niet te veel tenten wil plaatsen en moeten campingbezoekers (een) plaats(en) van voldoende omvang toewijzen. Ten behoeve van de controle kan het computersysteem de barcode op het entreebewijs opzoeken en teruggeven hoeveel tenten erbij horen. De medewerkers voeren het (de) nummer(s) van de toegewezen tentplaatsen in het computersysteem in.

- de plaatselijke brandweer. Zij moet (bij een calamiteit) kunnen vaststellen welke plaatsen van de festivalcamping in gebruik zijn. Daartoe krijgen ze via Internet toegang tot een digitale plattegrond van de camping, waarop staat aangegeven welke plaatsen in gebruik zijn. Als de brandweer daar aanleiding toe ziet, kan ze de festivalorganisatie opdragen de tentplaatsen anders toe te wijzen.

Elk van de hierboven genoemde partijen heeft specifieke taken. Om die taken te vervullen hebben ze informatie nodig. En deze informatie is het resultaat van **verwerking** van de geregistreerde **gegevens**. In dit voorbeeld zijn de **doelstellingen** van de betrokken partijen: kunnen controleren dat niemand zich onterecht toegang verschaft tot het festivalterrein, de juiste polsbandjes in de juiste hoeveelheid kunnen uitreiken, kunnen controleren of iemand niet te veel tenten wil plaatsen, een tentplaats kunnen toewijzen en kunnen vaststellen wie zich in een bepaalde nacht waar op de festivalcamping bevindt.

De medewerkers van de toegangspoort moeten daarnaast een vorm van **informatieverwerking** uitvoeren. Op basis van de aankomst- en vertrekdatum van deelnemers moet ze de juiste polsbandjes in de juiste hoeveelheid aan de bezoekers uitreiken.

In het voorbeeld staat met behulp van welke informatie de doelstellingen van de betrokken partijen gerealiseerd kunnen worden. Hier heeft een vorm van **identificatie van informatie** plaats gevonden. Vervolgens is het mogelijk te bepalen welke gegevens geregistreerd moeten worden om deze informatie te kunnen produceren. Dit is **identificatie van** de benodigde **gegevens**. Het resultaat hiervan is de lijst met gegevens die in het voorbeeld vermeld staan. Nadien kunnen de gegevens zodanig gegroepeerd worden dat ze in een relationele database opgeslagen kunnen worden. Dit gegevensmodelleren maakt geen deel uit van het examenprogramma.

Op deze manieren kunnen we in de informatica een doelstelling die met informatieverwerking te maken heeft, beschrijven in termen van intenties of handelingen van gebruikers. Deze doelstelling kunnen we dan *vertalen* naar een concretere doelstelling in termen van opslaan, zoeken, manipuleren, transporteren, et cetera van gegevens.

Om na te gaan *welke* gegevens moeten worden vastgelegd, wordt eerst geïdentificeerd welke informatie benodigd is. Daaruit kan worden afgeleid welke gegevens nodig zijn om deze informatie op te leveren (‘te produceren’). De afweging welke *representatie* van de gegevens op de computer handig is, is afhankelijk van de doelstelling van informatie- en gegevensverwerking. Dit laatste komt aan bod in het volgende voorbeeld.

Voorbeeld 2: fietsroutes

Op verschillende websites is het mogelijk je eigen fietstrainingsroute uit te zetten en die te bewaren. Dat werkt zo ongeveer als Google Maps. Eerst zet je een startpunt op een kaart. Vervolgens zet telkens een punt op een kaart en de software achter de website bepaalt telkens hoe de route loopt van het voorlaatste punt tot het nieuwe punt. De route wordt opgeslagen in de vorm van een (groot)

aantal GPS-coördinaten van de punten die samen de route vormen. Deze coördinaten vormen de gegevens van de route.

Aan de hand van deze gegevens kan met behulp van een GPS-viewer een routekaart gemaakt worden. Als het bijvoorbeeld je doelstelling is om je route ergens te publiceren, kun je zo'n routekaart gebruiken. Deze weergave van de fietsroute vormt in dat geval informatie waarmee je je doelstelling kunt realiseren. De GPS-viewer verwerkt in dit voorbeeld gegevens tot informatie.

Een andere doelstelling is om de fietsroute te rijden met behulp van een navigatieapparaat die op een fiets geplaatst kan worden. In dat geval worden de GPS-coördinaten ingelezen in het apparaat en de fietsroute in combinatie met de digitale kaarten die in het apparaat opgeslagen zijn, weergegeven in de vorm van een reeks routeaanwijzingen ('links', 'rechts', enzovoorts). Deze weergave van de route vormt informatie voor de gebruiker van het navigatieapparaat, waarmee hij zijn doelstelling – het rijden van de route – kan realiseren. Deze routeaanwijzingen vormen geen goede weergave van de route voor wie wil weten hoe de fietsroute globaal verloopt.

Het navigatieapparaat bevat software die bepaalt welke aanwijzingen er op welke plek getoond worden. Onderdeel van deze software is een datatype ten behoeve van de opslag van de ingelezen coördinaten. Hiertoe moet een datastructuur gekozen worden. Deze datastructuur vormt een **representatie** van de routegegevens. Welke datastructuur het meest geëigend is, hangt af van de doelstelling een fietsroute te kunnen rijden met het GPS-apparaat. Vermoedelijk worden de coördinaten in volgorde langs gelopen en vergeleken met de digitale kaart, op basis waarvan er routeaanwijzingen opgesteld en gevisualiseerd worden. Een rij van tupels met telkens een tweetal GPS-coördinaten past hier goed bij, mits het geheugen van het navigatieapparaat voldoende ruimte biedt. Laatstgenoemde voorwaarde kan een reden zijn dat de **implementeerbaarheid** van een rij of lijst beperkt is. Dat kan aanleiding zijn een andere datastructuur te kiezen, bijvoorbeeld twee rijen met losse coördinaten die door middel van de index aan elkaar gerelateerd zijn en zodoende een tweetal vormen.

3.4 Standaardrepresentaties

Gegevens als getallen, letters, kleuren, geluid en beeld kunnen op verschillende manieren gerepresenteerd worden. In het subdomein Standaardrepresentaties worden kandidaten geacht een aantal standaardrepresentaties te kennen en te gebruiken.

Woorden, letters, cijfers en leestekens kunnen worden gepresenteerd door middel van symbolen in verschillend schrift. In computergeheugens worden ze gepresenteerd door middel van een getalscode uit een bepaald coderingsschema. Dit maakt geen deel uit van het examenprogramma, tenzij men woorden, letters, cijfers en leestekens als mediacomponenten beschouwt.

Getallen kunnen worden gepresenteerd met behulp van cijfers en letters in een bepaald talstelsel. Een talstelsel wordt gekenmerkt door zijn grondtal. Bekende talstelsels zijn **binair** (grondtal = 2), **decimaal** (grondtal = 10) en **hexadecimaal** (grondtal = 16). Daarnaast bestaan er minder gangbare representaties van getallen, zoals Romeinse getallen. In een computergeheugen worden getallen in hun binaire representatie opgeslagen.

Kleuren kunnen in een computergeheugen worden gerepresenteerd door middel van een kleurcode die uit drie getallen bestaat. Het eerste getal staat voor het aandeel rood, het tweede voor het aandeel groen en het derde voor het aandeel blauw in de kleur. Als de getallen hexadecimaal worden genoteerd, bestaan ze elk uit twee symbolen. Door deze getallen in hexadecimale representatie aaneen te schakelen ontstaat een code van zes symbolen die een kleur representeert.

Geluidsfragmenten kunnen gerepresenteerd worden door middel van een grafiek waarin de druk van lucht in de omgeving van de luisteraar of de toonhoogten uitgezet zijn tegen te tijd. Maar ook het notenschrift is een representatie van geluid, in het bijzonder van muziek. Als een geluidsfragment in een computergeheugen opgeslagen moet worden, wordt ze eerst gesampled. Dat betekent dat de luchtdruk of de toonhoogte met korte tussenpozen gemeten wordt en dat de meetresultaten

opgeslagen worden. Hierbij treedt noodgedwongen informatieverlies op. Door middel van **compressie** kan een dergelijk geluidsbestand verkleind worden ten koste van nog meer kwaliteitsverlies.

(Bewegende) beelden kunnen in een computergeheugen gerepresenteerd worden door een serie kleurcodes die elk de kleur van een pixel weergeeft (= **pixel- of bitmaprepresentatie**). Een alternatieve representatie bestaat uit de vormen waaruit het beeld is samengesteld. Deze vormen op hun beurt worden door middel van wiskundige vergelijkingen gerepresenteerd. Dit wordt een vectorrepresentatie genoemd. Wordt voor een **vectorrepresentatie** gekozen, dan behoudt het beeld in tegenstelling tot de bitmaprepresentatie zijn kwaliteit als hij vergroot wordt, omdat het beeld telkens opnieuw uit zijn vormen wordt opgebouwd. Bitmaprepresentaties kunnen desgewenst gecomprimeerd worden hetgeen al dan niet met informatieverlies (lossy respectievelijk lossless) gepaard gaat. Voor vectorrepresentaties bestaat daartoe meestal geen noodzaak.

Welke standaardrepresentatie in welk geval gekozen wordt, wordt bepaald door een aantal aspecten:

- in hoeverre ze past bij wat beoogd wordt met de representatie van de gegevens;
- hoeveel opslagcapaciteit gegevens in bepaalde representaties vergen;
- hoeveel verwerkingscapaciteit het kost om gegevens van een representatie naar een andere om te zetten.

3.5 Gestructureerde data

Als een gegevensverzameling een bepaalde structuur heeft, spreken we van gestructureerde data. Een **databasemodel** beschrijft de uitgangspunten op basis waarvan deze structuur vormgegeven is. Het relationele model, waarin de gegevens gestructureerd zijn in groepen die aan elkaar gerelateerd zijn, is het meest bekende databasemodel, maar zeker niet het enige. Zo kan in een gegevensverzameling bijvoorbeeld ook met behulp van XML structuur aangebracht worden.

Zoekvragen of **queries** op een verzameling gestructureerde data kunnen worden geformuleerd in een bepaalde zoektaal. Voor het relationele databasemodel is SQL de bekendste zoektaal. Sommige databaseproducten kennen hun eigen zoektaal, zoals Query by Example in MS Access. Voor gegevens die met behulp van XML gestructureerd zijn, bestaat de zoektaal XML Query Language.

3.6 Voorbeeldspecificaties

C1 Doelstellingen

De kandidaat kan:

- het onderscheid uitleggen tussen gegevens en informatie.
- in een context voorbeelden van gegevens- en van informatieverwerking geven.
- in een context doelstellingen van gegevensverwerking noemen.

C2 Identificeren

De kandidaat kan:

- bij een gegeven doelstelling en in een gegeven context beargumenteren welke informatie noodzakelijk is om de doelstelling te realiseren.
    - Informatiemodellering is in dit domein niet noodzakelijk.
- bij informatie bepalen welke gegevens noodzakelijk zijn om deze informatie te produceren.
    - Gegevensmodellering is in dit domein niet noodzakelijk.
- beoordelen of uit een verzameling gegevens bepaalde informatie betrokken kan worden.
- identificeren welke gegevens in een bepaalde context geregistreerd kúnnen worden en in welke informatie daarmee geproduceerd kan worden.
    - Bijvoorbeeld in de context van mediagedrag, persoonlijke gezondheid, economische activiteit en apparaten in huis (‘the internet of things’).

C3 Representeren

De kandidaat kan:

- voor een verzameling (geïdentificeerde) gegevens die in een programma verwerkt moeten worden, een geschikte datastructuur kiezen op grond van overwegingen van elegantie, efficiëntie en implementeerbaarheid.

C4 Standaardrepresentaties

De kandidaat kan:

- de functie beschrijven van standaardrepresentaties van verschillende typen gegevens, zoals bijvoorbeeld getallen, kleuren, geluidsfragementen en (bewegende) beelden
    - Kennis van representaties van letters, cijfers en andere symbolen als zodanig is niet noodzakelijk.
- in voorkomende gevallen een onderbouwde keuze doen voor een standaardrepresentatie voor numerieke gegevens en media en in de onderbouwing het *doel* van gegevensrepresentatie en –verwerking meenemen.
- getallen en getalbewerkingen in ten minste twee verschillende representaties, zoals bijvoorbeeld decimaal, binair en hexadecimaal, uitwerken en aan elkaar verbinden.
- van een type gegevens ten minste twee verschillende representaties, zoals bijvoorbeeld de pixelrepresentatie of de vectorrepresentatie, tegen elkaar afwegen.
- de rol van compressie bij het digitaal opslaan van grote hoeveelheden gegevens uitleggen.
    - *Mogelijke uitbreiding: de werking van één of meer standaard compressietechnieken beschrijven.*

C5 Gestructureerde data

De kandidaat kan:

- zoekvragen formuleren om uit een verzameling gestructureerde data de relevante gegevens te halen om daarmee informatie te produceren.
    - Het gehanteerde databasemodel hoeft niet relationeel te zijn, maar als dat wel het geval is dan:
        - hoeft de gehanteerde zoektaal niet SQL te zijn.
        - is het aantal tabellen in de gegevensverzameling is bij voorkeur groter dan 1.
    - Als een ander databasemodel gebruikt wordt, dan:
        - hoeft evenmin gebruik gemaakt te worden van een geassocieerde zoektaal.
        - werkt de kandidaat bij voorkeur met gegevensverzamelingen die, zouden ze worden omgezet naar een relationele database, uit meer dan één tabel bestaan.

4 Domein D: Programmeren

4.1 Eindtermen

Subdomein D1: Ontwikkelen
De kandidaat kan, voor een gegeven doelstelling, programmacomponenten ontwikkelen in een imperatieve programmeertaal, daarbij programmeertaalconstructies gebruiken die abstractie ondersteunen, en programmacomponenten zodanig structureren dat ze door anderen gemakkelijk te begrijpen en te evalueren zijn.

Subdomein D2: Inspecteren en aanpassen
De kandidaat kan structuur en werking van gegeven programmacomponenten uitleggen, en zulke programmacomponenten aanpassen op basis van evaluatie of veranderde eisen.

4.2 Toelichting op het domein

Doel
In dit domein worden digitale artefacten daadwerkelijk gemaakt, aangepast en geanalyseerd. We gebruiken de term ‘programmacomponent’ om aan te geven dat het beoogde resultaat een compleet programma kan zijn, maar ook een deel van zo’n programma dat samenwerkt met bestaande delen. Het is belangrijk dat programmacomponenten niet alleen doen wat ze moeten doen, maar ook door andere programmeurs kunnen worden gehanteerd. Dat houdt in dat er veel aandacht is voor het leesbaar (en daarmee inzichtelijk, controleerbaar en onderhoudbaar) maken van programma’s.

Korte beschrijving
In dit domein komen daarom zowel het maken als het aanpassen van programmacomponenten aan de orde. Een leerling moet zowel vanuit het niets een werkende programmacomponent kunnen maken als een bestaande programmacomponent kunnen aanpassen of uitbreiden binnen de bestaande structuur en logica van dat programma. In beide gevallen zijn systematisch testen en debuggen van groot belang. Een aantal fundamentele informaticaconcepten, zoals algoritmen (B1) en datastructuren (B2), wordt in dit domein ook daadwerkelijk geïmplementeerd.

4.3 Ontwikkelen, inspecteren en aanpassen

Programmeren (in beperkte zin) is het uitdrukken van een algoritme in een voor een computer uitvoerbare taal, om zo een werkende, op de computer uitvoerbare, versie van dat algoritme te verkrijgen.

Programmeertalen
De taal waarin een programma wordt geschreven wordt programmeertaal genoemd. Binnen de vele bestaande programmeertalen zijn groepen te onderscheiden. In dit domein richten we ons op de zogenaamde imperatieve talen, waarin stapsgewijze instructies (die in een voorgeschreven volgorde worden uitgevoerd) het belangrijkste ingrediënt vormen. Imperatieve talen sluiten goed aan bij de karakterisering van algoritmen in Domein B: zo heeft elke imperatieve taal specifieke taalconstructies die overeenkomen met de basisbouwstenen opeenvolging, keuze en herhaling. Voorbeelden van imperatieve talen zijn Python, Java en Scratch.

Met de term ‘paradigma’ worden grondbeginselen aangeduid voor programma-ontwikkeling en voor programmeertalen die deze manier van ontwikkelen ondersteunen. Voorbeelden van paradigma’s en bijbehorende talen zijn: object-georiënteerd (Java, C++), functioneel (Haskell) en logisch (Prolog). Programmeertalen worden ook wel globaal ingedeeld in twee klassen: procedureel (talen waarin vooral het ‘hoe’ van een berekening in detail wordt weergegeven) en declaratief (talen waarin veeleer de bedoelde uitkomst, het ‘wat’, wordt geformuleerd).

Programmeertaalconstructies
In een programma in een imperatieve programmeertaal kunnen, onder andere, de volgende elementen voorkomen:

- aanduidingen voor data-objecten (**variabelen** en **constanten**) en **toewijzing** om waarden aan variabelen toe te kennen
- zogenaamde **controlestructuren**: taalconstructies voor
    - **opeenvolging**
    - **keuze**
    - **herhaling**

De meeste programmeertalen bevatten abstractiemechanismen die de programmeur in staat stellen om stukken programma apart te beschrijven en met een simpele aanduiding te activeren (‘aan te roepen’) in een groter programma. **Taalconstructies die abstractie ondersteunen** zijn bijvoorbeeld

- **procedures** (methoden, subroutines, …) voor stukken programma die iets doen;
- **functies**, al dan niet met parameters, voor stukken programma die een waarde opleveren.

Toepassing van abstractie maakt het programma overzichtelijker (door stukken programma te vervangen door een betekenisvolle naam) en vaak ook korter (door een stuk code dat meermalen wordt gebruikt te vervangen door meerdere aanroepen van eenzelfde procedure of functie).

Programma’s schrijven
Het **debuggen** van een programmacomponent is het opsporen en verhelpen van fouten (*bugs*) in de programmacomponent. Dit wordt bijvoorbeeld gedaan door de programmacomponent regel voor regel uit te voeren in plaats van in het geheel en regelmatig controlewaardes te printen. Debugging wordt in de regel ondersteund door een programmeeromgeving.

Eigenschappen van programma’s
Een programmacomponent is **correct** als deze bij iedere invoer de juiste uitvoer oplevert. De correctheid van een programmacomponent kan bijvoorbeeld onderzocht worden door het programma te testen. Correctheid kan ook onderzocht of beredeneerd worden aan de hand van het achterliggende algoritme.

Als een programma ‘het niet doet’ kan dat verschillende oorzaken hebben: bij het samenstellen van het programma kan een fout zijn gemaakt (‘plan composition problem’), maar het achterliggende algoritme zou ook incorrect kunnen zijn. Systematisch zoeken naar de oorzaak van een fout zal dan ook beide aspecten moeten beslaan.

Naast correctheid kan ook **efficiëntie** van een programmacomponent worden bekeken. De uitvoertijd van een programma kan *gemeten* worden aan de hand van verschillende invoer, maar ook via het achterliggende algoritme worden *beredeneerd*. Absolute rekentijd is meestal niet zo interessant; het is zinvoller om te bekijken welk effect het vergroten van de invoer heeft op de uitvoertijd van het programma.

De **leesbaarheid** van een programmacomponent kan verbeterd worden via een aantal mechanismen, zoals passend gebruik van abstractiemechanismen (procedures, functies), kiezen van betekenisvolle namen voor dataobjecten, procedures en functies, toevoegen van commentaar, en suggestieve lay-out. Het idee is dat vergroten van de leesbaarheid ook de inzichtelijkheid, overdraagbaarheid en onderhoudbaarheid van een programma ten goede komt. Zo zal een programma gemakkelijker op lange termijn zijn aan te passen, ook door anderen dan de oorspronkelijke maker.

4.4 Voorbeeldspecificaties

D1 Ontwikkelen
De kandidaat kan:

- werkende programmacomponenten ontwikkelen in een imperatieve programmeertaal naar keuze en daarbij,
    - gebruik maken van aanduidingen voor dataobjecten zoals variabelen en constanten; toewijzing van waarden aan variabelen;
    - gebruik maken van de algoritmische bouwstenen opeenvolging, keuze en herhaling implementeren met behulp van controlestructuren in de gekozen programmeertaal;
    - datatypen implementeren aan de hand van elementaire datatypen en taalconstructies voor datastructuren in de gekozen programmeertaal;
    - gebruik maken van taalconstructies die abstractie ondersteunen;

    - doelgericht gebruik maken van mechanismen om de leesbaarheid van een programmacomponent te vergroten, zoals kiezen van betekenisvolle namen voor dataobjecten, gebruik van procedures en functies, commentaar, en suggestieve lay-out;
    - debugging en testen inzetten.
- vanuit een algoritme een werkend programma ontwikkelen in de gekozen programmeertaal.

D2 Inspecteren en aanpassen

De kandidaat kan:

- de structuur en werking van een gegeven programmacomponent uitleggen.
- een gegeven programmacomponent evalueren aan de hand van de eigenschappen correctheid, efficiëntie, en leesbaarheid.
- een bestaande programmacomponent aanpassen
    - als gevolg van een evaluatie van bijvoorbeeld de correctheid, efficiëntie of leesbaarheid van de programmacomponent.
    - als gevolg van een veranderde of uitgebreide doelstelling.

5 Domein E: Architectuur

5.1 Eindtermen

Subdomein E1: Decompositie
De kandidaat kan de structuur en werking van digitale artefacten uitleggen aan de hand van architectuurelementen, dat wil zeggen in termen van de niveaulagen *fysiek*, *logisch* en *toepassingen*, en in termen van de componenten in deze lagen en hun onderlinge interactie.

Subdomein E2: Security
De kandidaat kan enkele security-bedreigingen en veelgebruikte technische maatregelen benoemen en relateren aan architectuurelementen.

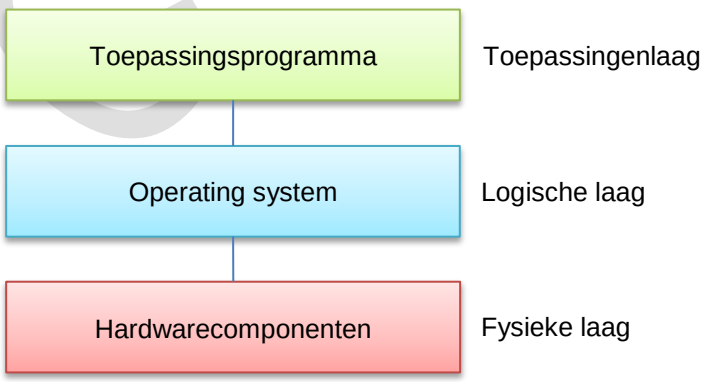
5.2 Toelichting op het domein

De **architectuur** van een digitaal artefact beschrijft, net zoals de architectuur van een gebouw of een brug, globaal uit welke componenten het artefact bestaat en hoe die met elkaar samenhangen en samenwerken. De Von Neumannarchitectuur voor een computer is een bekend voorbeeld. In het kernprogramma ligt de nadruk op het begrijpen van bouw en werking van digitale artefacten. Enkele keuzethema's bieden verdiepingsmogelijkheden waarin ook ontwerpen op architectuurgebied aan de orde komt.

Als beschrijvingsmiddel voor architectuuraspecten hanteren we de het klassieke lagenmodel, van de fysieke laag (hardware: werkgeheugen, processor, achtergrondgeheugen, rand- en netwerkapparaten, verbindingen) via de logische laag (operating system, talen/compilers, DBMS, netwerk) tot de toepassingenlaag (toepassingsprogramma's, databases, webdiensten). Architectuurelementen vormen een goede kapstok voor securityaspecten, voor zover ze technische maatregelen betreffen. Menselijke factoren rond security zijn onderdeel van kerndomein F: Interactie.

5.3 Decompositie

De componenten van een digitaal artefact hebben elk een specifieke rol in de architectuur van het artefact. Als voorbeeld nemen we een 'toepassing' of 'applicatie'. Dit artefact levert een of meer functionaliteiten aan een gebruiker. Deze functionaliteiten worden gerealiseerd door middel van een toepassingsprogramma. Dit programma maakt op zijn beurt gebruik van de diensten van een operating system en het operating system maakt gebruik van de diensten van hardwarecomponenten. De architectuur van een toepassing kent daarmee een hiërarchische opbouw met een drietal **lagen**. Soft- en hardwarecomponenten en de architectuurlagen vormen samen de **elementen** van de architectuur. In de onderstaande figuur wordt deze opbouw in beeld gebracht en worden de drie lagen benoemd.



Figuur 7: Decompositie van een toepassing in architectuurlagen

Ook de architectuur van databasetoepassingen kan in deze drie lagen beschreven worden, namelijk:

- toepassingenlaag: user interfaces met zoek- en mutatieopdrachten in een zoektaal;
  levert aan de gebruiker als dienst gegevensverwerking;
- logische laag: het database managementsysteem of de database-engine;
  levert aan de toepassingenlaag als dienst verwerking van zoekinstructies;
- fysieke laag: de gegevensbestanden
  levert aan de logische laag als dienst opslag en beschikbaarstelling van gegevens in een extern geheugen.

Niet alle architectuurmodellen kennen drie lagen. De TCP/IP-architectuur voor netwerken kent vijf lagen en de OSI-architectuur voor netwerken bestaat zelfs uit zeven lagen.

5.4 Security

Security gaat over bedreigingen van de drie aspecten vertrouwelijkheid, integriteit en beschikbaarheid van een digitaal artefact, evenals maatregelen die de risico's beperken die met die bedreigingen te maken hebben.

- De **vertrouwelijkheid** van een digitaal artefact is de mate waarin gegevens die door het artefact behandeld worden, afgeschermd zijn tegen ongeoorloofde inzage.

- De **integriteit** van een digitaal artefact is de mate waarin gegevens die door het artefact behandeld worden, beschermd zijn tegen verlies of (on)bedoelde wijziging.

- De **beschikbaarheid** van een digitaal artefact is de mate waarin er storingsvrij gebruik van gemaakt kan worden.

Soms wordt er een vierde aspect bekeken, namelijk 'onloochenbaarheid': de zekerheid dat (digitale) informatie afkomstig is van een bepaalde afzender of bron. We zullen hier niet verder op in gaan.

Het falen van digitale artefacten door technische storingen (schijfcrashes, uitval van diensten) kan gezien worden als bedreiging, maar het vakgebied security gaat specifiek over bedreigingen die te maken hebben met moedwillige (en kwaadwillende) aanvallen op digitale artefacten.

We kunnen zulke bedreigingen ruwweg indelen in drie categorieën, door te kijken van welke soort zwakheden aanvallers gebruik maken:

1. **Zwakheden in de computerarchitectuur**. Voorbeelden zijn: bypass van een architectuurelement, bijvoorbeeld een SQL-injectie (= bypass van de toepassingenlaag) of rechtstreekse benadering van gegevensbestanden (= bypass van de logische laag) en een DDOS-aanval waarin de capaciteit van een netwerk wordt overschreden. De bijbehorende maatregelen zijn meestal technisch van aard, zoals firewalls.
2. **Zwakheden in de communicatie** tussen partijen over (inherent) onbetrouwbaar medium. Voorbeeld van een bedreiging is het onderscheppen van een vertrouwelijk bericht. Ook hier zijn de maatregelen technisch, met name het inzetten van versleuteling (cryptografie).
3. **Zwakheden van gebruikers**. Voorbeelden zijn: laten rondslingeren van wachtwoorden, en onbedoeld installeren van kwaadaardige software zoals een virus. Bij deze categorie horen socio-technische risico's en maatregelen, bijvoorbeeld phishing en wachtwoorddiscipline.

Deze drie categorieën vormen een handige classificatie, maar de categorieën sluiten elkaar niet helemaal uit: zo kan het risico vanwege rondslingerende USB-sticks zowel bestreden worden met technische maatregelen (encryptie van de inhoud van zo'n stick) als met socio-technische (voorlichting over de risico's, gedragscodes voor het gebruik van USB-sticks).

Security behelst bedreigingen onder ogen te zien en een risicoanalyse te maken: in te schatten hoe ernstig de bedreigingen zijn. Maatregelen kunnen nooit waterdicht zijn; daarom wordt een afweging gemaakt aan de hand van ernst van de risico's en de kosten en verwachte effecten van maatregelen.

Maatregelen kunnen zich richten op:

- preventie: voorkomen dat een bedreiging optreedt;
- detectie: vaststellen dat een bedreiging opgetreden is;
- repressie: de schade die door het optreden van een bedreiging ontstaat, beperken;
- correctie: de schade doe door het optreden van een bedreiging ontstaat, herstellen.

Technische aspecten van security (rond de categorieën 1 en 2) komen aan de orde in het domein Architectuur, socio-technische aspecten (rond categorie 3) in het domein Interactie.

Encryptie is versleutelen van berichten en gegevens bij transport en opslag. Een encryptiemethode bestaat uit een algoritme en een of meer sleutels. Als bij vercijfering en ontcijfering van een bericht gebruik gemaakt wordt van dezelfde sleutel, dan spreken we van symmetrische encryptie. Bij asymmetrische encryptie wordt gebruik gemaakt van twee verschillende sleutels (de vercijfer- en de ontcijfersleutel), waartussen een vaak rekenkundig verband bestaat. De ontcijfersleutel kan eenvoudig uit de vercijfersleutel worden afgeleid. De omgekeerde bewerking is theoretisch ook mogelijk, maar in de praktijk zeer moeilijk. De ontcijfersleutel is daarom vaak openbaar, maar de vercijfersleutel niet. Deze vorm van encryptie heet daarom ook wel **public key encryption**.

5.5 Voorbeeldspecificaties

E1 Decompositie

De kandidaat kan:

- de architectuur van digitale artefacten uitleggen in termen van drie lagen: de fysieke laag, de logische laag en de toepassingenlaag.
- beschrijven welke diensten de fysieke laag levert aan de logische laag, de functie van enkele componenten van de fysieke laag beschrijven en beschrijven op welke wijze deze componenten met elkaar samenwerken om de diensten van de fysieke laag vorm te geven.
    - Voorbeelden van componenten in de fysieke laag zijn het werkgeheugen, de processor, het achtergrondgeheugen, lokale randapparaten, lokale verbindingen als USB en netwerkwerkverbindingen en –apparaten.
- beschrijven welke diensten de logische laag levert aan de toepassingenlaag, de functie van enkele componenten van de logische laag beschrijven en beschrijven op welke wijze deze componenten met elkaar samenwerken om de diensten van de logische laag vorm te geven.
    - Voorbeelden van componenten in de logische laag zijn het operating system, talen, compilers, DBMS, het netwerkbesturingssysteem en routersoftware.
- beschrijven welke diensten de toepassingenlaag levert aan gebruiker, de functie van enkele componenten van de toepassingenlaag beschrijven en beschrijven op welke wijze deze componenten met elkaar samenwerken om de diensten van de toepassingenlaag vorm te geven.
    - Voorbeelden van componenten in de toepassingenlaag zijn toepassingssoftware en zoektalen.
- identificeren welke diensten die door een architectuurelement geleverd worden, noodzakelijk zijn voor bepaalde toepassingen.
- identificeren welke architectuurelementen en componenten bepalend zijn voor de snelheid van bepaalde toepassingen.
- uitleggen dat het mogelijk is de snelheid van toepassingen te vergroten door een laag te bypassen, uitleggen welke consequenties dat heeft voor het ontwikkelen van een digitaal artefact en uitleggen dat hierdoor serieuze bedreigingen kunnen ontstaan voor de security van een digitaal artefact.

E2 Security

De kandidaat kan:

- enkele securitybedreigingen benoemen en in gegeven bedreigingen in verband brengen met de drie beveiligingsaspecten vertrouwelijkheid, integriteit en beschikbaarheid en enkele technische maatregelen benoemen.
    - Daarbij behoren de distributie van gegevens en de verwerking daarvan over de drie lagen uitleggen en de gevolgen daarvan voor security analyseren.

    - Uitvoering van risicoanalyses maakt deel uit van keuzedomein N.
    - Socio-technische maatregelen komen aan bod in kerndomein F.
- principes van cryptografische maatregelen uitleggen en voorbeelden van zulke maatregelen geven.
    - Kennis van specifieke encryptiemethoden is niet noodzakelijk.

6 Domein F: Interactie

6.1 Eindtermen

Subdomein F1: Usability
De kandidaat kan gebruikersinterfaces van digitale artefacten evalueren aan de hand van heuristieken, en vuistregels van *goed ontwerp* met betrekking tot interfaces toepassen bij ontwerp en ontwikkeling van digitale artefacten.

Subdomein F2: Maatschappelijke aspecten
De kandidaat kan de invloed van digitale artefacten op sociale interactie en persoonlijke levenssfeer herkennen en in historisch perspectief plaatsen.

Subdomein F3: Privacy
De kandidaat kan redeneren over de gevolgen van de veranderende mogelijkheden van digitale artefacten op de persoonlijke vrijheid.

Subdomein F4: Security
De kandidaat kan enkele security-bedreigingen en veelgebruikte socio-technische maatregelen benoemen en deze relateren aan sociale en menselijke factoren.

6.2 Toelichting op het domein

Het doel van dit domein is het redeneren over de verbinding tussen informatica en de omgeving. Er zijn drie perspectieven opgenomen: de interactie tussen digitaal artefact en gebruiker (*usability*), de impact van digitale artefacten op de maatschappij en de impact van digitale artefacten op het individu (in het bijzonder *privacy*). Deze drie perspectieven bieden tevens goede aanknopingspunten voor de socio-technische kant van security, als aanvulling op de meer technische aspecten onder het domein E Architectuur.

6.3 Usability

Heuristieken voor gebruikersinterfaces
Een **heuristiek** is een oplossingsmethode die niet altijd de goede of de beste oplossing oplevert, maar wel heel snel in de meeste gevallen een redelijk goede oplossing oplevert; een vuistregel. Er zijn heuristieken die vragen beantwoorden en heuristieken om ontwerpen te beoordelen. Een heuristiek om het weer van morgen te bepalen kan bijvoorbeeld zijn “het weer is hetzelfde als vandaag”. Dit is heel makkelijk om te bepalen en klopt meestal, maar gaat natuurlijk weleens fout.

Een **gebruikersinterface** is iedere mogelijke manier waarop een digitaal artefact (een programma of apparaat) informatie deelt met de gebruiker en een gebruiker invloed kan uitoefenen op een digitaal artefact. Denk hierbij aan een touchscreen, een helpmenu, een *statusbar*, een *webform* etc. De **heuristieken van Nielsen** zijn tien vuistregels voor het ontwerpen van gebruikersinterfaces. Enkele voorbeelden zijn:

- Zorg dat de huidige status van het systeem altijd zichtbaar is (denk bijvoorbeeld aan laad-balken)
- Zorg dat veelgebruikte handelingen versneld kunnen worden op manieren die voor een beginnende gebruiker niet opvallen (zoals *hot keys*) zodat zowel beginnende als gevorderde gebruikers met de interface uit de voeten kunnen.

De volledige lijst is (in het Engels) te vinden op de website van de auteur.

6.4 Maatschappelijke aspecten

Market pull en **technology push** zijn twee verschillende manieren waarop toepassingen van nieuwe technologie uiteindelijk bij consumenten terecht kunnen komen. In het geval van market pull leidt een bestaande vraag tot ontwikkeling van producten die aan die vraag voldoen. Een bekend voorbeeld is de ontwikkeling van digitale fotografie: er was een behoefte om meer foto’s te kunnen nemen en deze direct te kunnen zien; digitale fotografie voorzag hierin. Bij technology push leiden nieuwe technische

mogelijkheden tot producten waar daarvoor nog geen bewuste vraag naar is. Een voorbeeld is een bekende quote van Steve Jobs: “It's really hard to design products by focus groups. A lot of times, people don't know what they want until you show it to them.”

Mogelijke invloeden van digitale artefacten zijn

- In het economische domein:
  - Internet winkelen
  - Deeleconomie: je bezit zelf geen auto, klopboor of vakantiehuis, maar huurt die van andere burgers
  - *the internet of things*: het internet verbindt niet alleen menselijke gebruikers, maar ook apparaten
- Politiek en meningsvorming
  - *filterbubble*: je krijgt alleen berichten te zien van en nieuws binnen via mensen die het met je eens zijn
- Menselijke relaties
  - Het is veel makkelijker relaties te onderhouden met mensen die zich op lange afstand bevinden
  - cyberpesten: pesten vindt plaats op sociale media, voor een groot deel buiten het zicht van ouders en leraren

6.5 Privacy

Privacy is de mogelijkheid van een individu om zichzelf, of informatie over zichzelf, af te schermen van personen of groepen waarmee hij of zij die informatie niet wil delen. In Nederland is het recht op privacy in de grondwet opgenomen.

De komst van krachtige computers en slimme analysetechnieken maakt het mogelijk om grote hoeveelheden gestructureerde, maar ook ongestructureerde data te combineren en analyseren; dit heet **big data**. Een veelbesproken toepassing van big data zijn nieuwe systemen die zijn en worden ontwikkeld om het (koop)gedrag van mensen te voorspellen.

6.6 Security

Toelichting op dit subdomein staat beschreven in paragraaf 5.4. Merk op dat dit domein alleen de socio-technische aspecten van security omvat.

6.7 Voorbeeldspecificaties

F1 Usability

De kandidaat kan:

- de heuristieken in ten minste één stelsel van heuristieken voor interfaceontwerp, zoals bijvoorbeeld de heuristieken van Nielsen, gebruiken.
  - deze heuristieken gebruiken om de gebruikersinterface van een gegeven digitaal artefact te evalueren.
  - deze heuristieken, of andere principes van goed ontwerp, toepassen bij ontwerp en ontwikkeling van de gebruikersinterface van digitale artefacten.

F2 Maatschappelijke aspecten

De kandidaat kan:

- de invloed van digitale artefacten op sociale interactie herkennen en bespreken.
  - Bijvoorbeeld aan de hand van de begrippen market pull en technology push.
  - Invloeden op een aantal domeinen bekijken, bijvoorbeeld economie, politiek en meningsvorming, menselijke relaties.
- de invloed van digitale artefacten op de persoonlijke levenssfeer herkennen
  - Invloeden op een aantal domeinen bekijken, bijvoorbeeld persoonlijke gezondheid en menselijke relaties.
- deze invloeden in historisch perspectief plaatsen.
  - Een gegeven verschijnsel relateren aan ontwikkelingen uit het verleden.

- *Mogelijke uitbreiding: een beredeneerde inschatting maken van de mogelijke gevolgen van een recente of toekomstige ontwikkeling voor sociale interactie en/of de persoonlijke levenssfeer*.

F3 Privacy

De kandidaat kan:

- redeneren over de gevolgen van de veranderende mogelijkheden van digitale artefacten op de persoonlijke vrijheid.
    - de wederzijdse beïnvloeding van technische, juridische en sociale factoren in het privacyvraagstuk benoemen
        - Bijvoorbeeld de effecten van big data voor de privacy bespreken
    - Mogelijke uitbreiding: privacyaspecten van het eigen handelen evalueren en gemaakte keuzes onderbouwen.

F4 Security

De kandidaat kan:

- enkele securitybedreigingen benoemen en gegeven bedreigingen in verband brengen met de drie beveiligingsaspecten vertrouwelijkheid, integriteit en beschikbaarheid
- enkele socio-technische maatregelen benoemen.
    - voor- en nadelen van gegeven technische en socio-technische maatregelen ten opzichte van elkaar benoemen.
    - Uitvoering van risicoanalyses maakt deel uit van keuzedomein N.
    - Technische maatregelen komen aan bod in kerndomein E.

Bronnen

Association for Computer Machinery. (1992). ACM Code of Ethics and Professional Conduct. Geraadpleegd op 11 januari 2018 van https://www.acm.org/about-acm/acm-code-of-ethics-and-professional-conduct.

Barendsen, E., & Tolboom, J. (2016). *Advies examenprogramma informatica havo/vwo*. Enschede: SLO.

Nielsen Norman Group. (z.j.). *10 Usability Heuristics for User Interface Design*. Geraadpleegd op 11 januari 2018 van https://www.nngroup.com/articles/ten-usability-heuristics/.

Vereniging van Register Informatici. (2015). *Gedragscode*. Nijkerk: VRI.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35.