Les C-01: Algoritmen

1.0 Inleiding

Moeilijke problemen pakken we vaak stapsgewijs aan: “Een olifant eet je met kleine hapjes”. Het is van belang om de stappen waarmee we een probleem oplossen goed te omschrijven. Dat maakt het gemakkelijk bij een volgende keer dat een dergelijk probleem opgelost dient te worden.

De stappen of instructies waarmee een probleem opgelost / doel bereikt kan worden noemen we ook wel een algoritme. In de loop der tijd zijn ontzettend veel algoritmen bedacht om problemen waar de mensheid tegenaan liep op te lossen. In deze lesbrief wordt een aantal van deze algoritmen besproken.

Doel daarvan is dat je gevoel ontwikkelt voor het formuleren van instructievoorschriften waarmee een bepaald doel kan worden bereikt. Deze vaardigheid kan je goed gebruiken wanneer je zelf computerprogramma’s gaat schrijven.

1.1 Al-Chwarizmi en algoritmen

Ten grondslag aan het beschrijven van programmastructuren (les C-02) en het uiteindelijk programmeren (les C-04) ligt het menselijk vermogen om instructievoorschriften te ontwerpen die tot een bepaald resultaat leiden. Zo'n voorgeschreven recept van instructies noemen we ook wel een **algoritme**.

Dit woord is afgeleid van de Perzische geleerde Abu Adbdallah ben Mosa **al-Chwarizmi** (790-840). Hij is geboren in het dorpje Chwarizm (nu: Chiva) in de provincie Chorasan van Perzië (nu: Oezbekistan). De naam al-Chwarizmi betekent "afkomstig uit Chwarizm". Kort na zijn geboorte verhuisde zijn familie naar een dorpje vlak bij Bagdad, waar hij tot grote wetenschappelijke prestaties kwam.
Al-Chwarizmi was één van de beroemdste oosterse wetenschappers op het gebied van de wiskunde, aardrijkskunde en astronomie. Zijn systematische en logische aanpak van het oplossen van lineaire en kwadratische vergelijkingen maakten hem één van de grondleggers van de algebra, een vakgebied dat zijn naam ontleent aan al-Chwarizmi's beroemde boek: "*Hisab **al-jabr** wa al-muqabala*".
De systematische wijze waarop je zelf in de basisvorming hebt geleerd om lineaire vergelijkingen op te lossen (*x*-en links, getallen rechts) en kwadratische vergelijkingen op te lossen (*abc*-formule) heb je te danken aan het werk van al-Chwarizmi.

Een algoritme is dus een aantal instructies die moeten worden uitgevoerd om een bepaald doel te bereiken. Eigenschappen van algoritmen zijn:

- er moet een eindig aantal stappen worden doorlopen
- de volgorde (**sequentie**) waarin de stappen moeten doorlopen kan van belang zijn
- soms moeten stappen worden herhaald (**iteratie**)
- soms moeten er keuzes worden gemaakt (**selectie**) hoe het algoritme verder moet gaan

Algoritmen hoeven niet persé wiskundig van aard te zijn. Ook als je met behulp van een recept een maaltijd bereidt of met behulp van een instructie een bouwpakket in elkaar zet ga je algoritmisch te werk.

Het beschrijven van algoritmen maakt mogelijk dat iedereen, als hij/zij maar de aangegeven stappen uitvoert, tot het gewenste resultaat kan komen. Het proces en het resultaat zijn **reproduceerbaar**. Met behulp van de *abc*-formule is iedereen in staat om de oplossing(en) van een kwadratische vergelijking te vinden en met behulp van het recept voor het koken van aardappelen is iedereen in staat om aardappelen te koken.
Sommige algoritmen zijn **automatiseerbaar**, door een apparaat uit te voeren. De *abc*-formule is gemakkelijk in een (grafische) rekenmachine te zetten en dat maakt het oplossen van kwadratische vergelijkingen een stuk makkelijker.

Om algoritmen te kunnen automatiseren dienen apparaten geprogrammeerd te worden. Om een grafische rekenmachine de *abc*-formule te kunnen laten doorrekenen, moet de rekenmachine eerst worden geprogrammeerd.
Aan het programmeren gaat dus een belangrijke stap vooraf: het bedenken van de stappen waaruit het programma bestaat. In het vervolg van deze lesbrief wordt een aantal aansprekende algoritmen behandeld.

1.2 Het algoritme van Euclides

In de klassieke wiskunde speelde het begrip deelbaarheid een belangrijke rol. De Griekse wiskundige Euclides (325 – 265 voor Chr.) schreef een algoritme voor het bepalen van de grootste gemeenschappelijke deler (ggd) van twee getallen, dat is het grootste gehele getal waardoor je beide getallen kunt delen.

Voorbeeld:
De delers van 24 zijn 1, 2, 3, 4, 6, 8, 12 en 24
De delers van 15 zijn 1, 2, 3, 5 en 15
De grootste gemene deler van 24 en 15 is 3.

Het ggd-algoritme van Euclides

Noem het grootste van de beide getallen A, het andere B.
Trek B net zo vaak van A af totdat er 0 over blijft of een getal kleiner dan B.
Wanneer er 0 over blijft zijn we klaar, en is B de ggd.
Zo niet, herhaal dan het algoritme met B en wat er van A over is.

Voorbeeld:
We kiezen de getallen A = 1140 en B = 900.
We kunnen B één keer van A aftrekken: A = 240.
Nu is het grootste getal A = 900 en B = 240.
We kunnen B drie keer van A aftrekken: A = 180.
Nu is het grootste getal A = 240 en B = 180.
We kunnen B één keer van A aftrekken: A = 60.
Nu is het grootste getal A = 180 en B = 60.
We kunnen B drie keer van A aftrekken: A = 0.
De grootste gemene deler is dus 60.

OPDRACHT

Opdracht 1.1
Bepaal met behulp van het algoritme van Euclides de ggd van de getallen 810 en 198.

Opdracht 1.2
Bepaal met behulp van het algortime van Eucliders de ggd van de getallen 84 en 65.

1.3 De Zeef van Eratosthenes

In de klassieke wiskunde speelde het begrip deelbaarheid een belangrijke rol. Het is dan ook niet vreemd dat priemgetallen (slechts deelbaar door zichzelf en 1) in de klassieke wiskunde een bijzondere positie innamen.

De Zeef van Eratosthenes (circa 240 v.Chr.) is een al zeer lang bekend algoritme om priemgetallen te vinden kleiner dan een vooraf vastgestelde waarde. De methode is vooral efficiënt wanneer hij wordt gebruikt voor de kleinere priemgetallen.

De Zeef van Eratosthenes

We willen alle priemgetallen kleiner dan n weten.
Schrijf alle getallen 2 tot en met n op.
Start met het kleinste getal (2 dus).
Streep alle veelvouden van dat getal door.
Ga naar het volgende niet doorgestreepte getal (3 dus).
Streep alle veelvouden van dat getal door.
Herhaal dit zolang het getal kleiner is dan $\sqrt{n}$.

We zoeken bijvoorbeeld de priemgetallen tot en met 30.
We beginnen met de lijst van 2 tot en met 30:

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

In de eerste ronde strepen we de veelvouden van 2 weg:

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~ 21 ~~22~~ 23 ~~24~~ 25 ~~26~~ 27 ~~28~~ 29 ~~30~~

Daarna gaan we verder met de veelvouden van 3:

2 3 5 7 ~~9~~ 11 13 ~~15~~ 17 19 ~~21~~ 23 25 ~~27~~ 29

Het getal 4 is al weggestreept, dus gaan we verder met veelvouden van 5.

2 3 5 7 11 13 17 19 23 ~~25~~ 29

Het volgende getal, 7, is groter dan de wortel uit 30, dus we kunnen nu stoppen.
Wat we overhouden zijn de priemgetallen tot en met 30:

2 3 5 7 11 13 17 19 23 29

OPDRACHT

Opdracht 1.3
Waarom is het in de Zeef van Eratosthenes voldoende om door te gaan tot het getal $\sqrt{n}$?

1.4 Schrikkeljaren

In de loop der eeuwen is een algoritme ontstaan dat bepaalt of een jaar een schrikkeljaar is of niet. Hieronder is beschreven hoe de huidige omgang met schrikkeljaren is ontstaan.

Schrikkeljaaralgoritme van Gregorius XIII

Onze jaarkalender stamt van de Romeinse jaarkalender. Onder Julius Caesar werd in 45 voor Christus de jaarkalender hervormd. In de door Ceasar aangepaste kalender werd gewerkt met een jaar van 365 dagen en 6 uur. Dat is de periode waarin de aarde om de zon draait. Eens in de vier jaar moest deze overschrijding van zes uur worden goedgemaakt met een extra dag, de schrikkeldag.

Met de meetapparatuur van onze tijd is vastgesteld dat de periode waarin de aarde om de zon draait 365 dagen, 5 uur, 48 minuten en 45,1875 seconden bedraagt. Per jaar had de methode van Caesar dus een afwijking van ongeveer 11 minuten van wat de natuur aangaf. In 128 jaar is dat een afwijking van een dag.

In het jaar 325 na Christus werd op de Concilie van Nicaea het begin van de lente vastgesteld op 21 maart. Deze vaste datum was noodzakelijk om het christelijke paasfeest te kunnen bepalen. Door de afwijking van één dag in de 128 jaar kwam de lente echter steeds vroeger. Dat bracht paus Gregorius XIII er toe om in 1582 (vanaf het jaar 325 gerekend kwam de lente in dat jaar 10 dagen eerder dan 21 maart) de jaarrekening aan te passen. Hij maakte in één keer het tijdverschil van 10 dagen goed door donderdag 4 oktober 1582 te laten volgen door vrijdag 15 oktober 1582 (deze periode van tien dagen werd gekozen omdat er geen christelijke feestdagen in vielen). Daarnaast voegde hij aan de bestaande jaarrekening met schrikkeljaren het volgende algoritme toe:

een jaar is een schrikkeljaar als het jaartal deelbaar is door 4
een jaar is geen schrikkeljaar als het jaartal deelbaar is door 100
tenzij het jaartal deelbaar is door 400

Voorbeelden van schrikkeljaren zijn de jaren: 1988, 2000, 2144. Geen schrikkeljaren zijn de jaren: 1900, 1987, 2100. Door deze aanpassing werd de lengte van een jaar in de jaarrekening $365\frac{97}{400}$ *oftewel precies 365 jaar, 5 uur, 49 minuten en 12 seconden. De afwijking van de exacte omlooptijd van de aarde rond de zon is in deze jaarrekening dus nog geen halve minuut. In 3000 jaar tijd is slechts een aanpassing van een dag noodzakelijk.*

OPDRACHTEN

Opdracht 1.4
Waarom is in de Gregoriaanse jaartelling een jaar $365\frac{97}{400}$ dagen lang?

Opdracht 1.5
Welke van de onderstaande jaren waren / zijn schrikkeljaren?

300, 364, 567, 1400, 1500, 1600, 1920, 1969, 1996, 2030, 2200, 2400

1.5 Algoritme van Gauss

Een algoritme dat tot de verbeelding spreekt is verzonnen door de wiskundige Carl Friedrich Gauss (1777-1855). Hij bedacht een algoritme dat bij een datum de bijbehorende dag bepaalt:

Algoritme van Gauss

Lees jaartal, maand en dag

Als maand > 2	*dan*	*trek 2 van de maand af*
	anders	*tel 10 bij de maand op en trek 1 van het jaartal af*

Splits het jaartal in tweeën: *de eerste twee cijfers heten eeuw*
de laatste twee cijfers heten jaar

Bereken nu met de gevonden waarden van dag, maand, jaar eeuw
A := (13 x maand –1) / 5 *rest verwaarlozen*
B := jaar / 4 *rest verwaarlozen*
C := eeuw / 4 *rest verwaarlozen*
D := A + B + C + dag + jaar – 2 x eeuw + 700

Bepaal de rest van de deling D / 7 en noem die rest weekdag

De weekdag geeft aan welke dag bij de ingevoerde datum hoort:
0 = zo, 1 = ma, 2 = di, 3 = wo, 4 = do, 5 = vr, 6 = za

OPDRACHTEN

Opdracht 1.6
Probeer het algoritme uit met de dag van vandaag.

1.6 Zelf algoritmen maken

Het bedenken van een algoritme of recept waarmee je een bepaald resultaat kunt bereiken, is van groot belang bij het uiteindelijk programmeren. Daarbij is het niet alleen belangrijk dat het algoritme het gewenste doel ook bereikt (**effectiviteit**). Ook is het van belang dat het algoritme niet al te omslachtig is, te veel stappen omvat (**efficiëntie**).

OPDRACHTEN

Opdracht 1.7
Schrijf een algoritme voor het omzetten van een decimaal getal kleiner dan 256 in een binair getal.

Opdracht 1.8
Schrijf een algoritme dat bepaalt of een ingevoerd getal een priemgetal is.

1.7 Samenvatting

Een **algoritme** (naar de Perzische wetenschapper Al-Chwarizmi) is een reeks instructievoorschriften waarmee een bepaald resultaat kan worden bereikt.

Eigenschappen van algoritmen zijn:

- er moet een eindig aantal stappen worden doorlopen
- de volgorde (sequentie) waarin de stappen moeten doorlopen kan van belang zijn
- soms moeten stappen worden herhaald (iteratie)
- soms moeten er keuzes worden gemaakt hoe het algoritme verder dient te verlopen

Algoritmen maken het bereiken van het resultaat **reproduceerbaar** en in sommige gevallen ook **automatiseerbaar**.

De volgende algoritmen zijn behandeld:

- het ggd algoritme van Euclides
- de Zeef van Eratosthenes
- het schrikkeljaaralgoritme van Gregorius XIII
- het algoritme van Gauss

Algoritmen hoeven niet persé wiskundig van aard te zijn. Wel is altijd systematisch en logisch handelen vereist. Algoritmen worden beoordeeld op **effectiviteit** (wordt het doel bereikt ?) en **efficiëntie** (wordt het doel op een doelmatige manier bereikt ?).

1.8 ANTWOORDEN

Opdracht 1.1
We kiezen de getallen A = 810 en B = 198.
We kunnen B vier keer van A aftrekken: A = 18.
Nu is het grootste getal A = 198 en B = 18.
We kunnen B elf keer van A aftrekken: A = 0.
De grootste gemene deler is dus 18.

Opdracht 1.2
We kiezen de getallen A = 84 en B = 65.
We kunnen B één keer van A aftrekken: $A = 84 - 1 \cdot 65 = 19$.
Nu is het grootste getal A = 65 en B = 19.
We kunnen B drie keer van A aftrekken: $A = 65 - 3 \cdot 19 = 7$.
Nu is het grootste getal A = 19 en B = 7.
We kunnen B twee keer van A aftrekken: $A = 19 - 2 \cdot 7 = 5$.
Nu is het grootste getal A = 7 en B = 5.
We kunnen B één keer van A aftrekken: $A = 7 - 1 \cdot 5 = 2$.
Nu is het grootste getal A = 5 en B = 2.
We kunnen B twee keer van A aftrekken: $A = 5 - 2 \cdot 2 = 1$.
Nu is het grootste getal A = 2 en B = 1.
We kunnen B twee keer van A aftrekken: $A = 2 - 2 \cdot 1 = 0$.
De grootste gemene deler is dus 1. Er is dus geen (grotere) gemene deler.

Opdracht 1.3
Getallen groter dan $\sqrt{n}$ zijn of priemgetallen of veelvouden van kleinere getallen, maar die zijn al uitgezeefd.

Opdracht 1.4
In de jaartelling van Caesar is een jaar $365\frac{1}{4}$ oftewel $365\frac{100}{400}$ dagen lang.
In de Gregoriaanse jaartelling vallen van elke 400 jaar 3 schrikkeljaren af.
Van elk jaar valt dus $\frac{3}{400}$ jaar af.
Een jaar in de Gregoriaanse jaartelling is dus $365\frac{97}{400}$ dagen lang.

Opdracht 1.5
De jaren 300, 364, 567, 1400, 1500 vallen onder de jaartelling van Caesar. Van deze jaren zijn 300, 364, 1400 en 1500 schrikkeljaren en 567 niet.

De jaren 1600, 1900, 1920, 1969, 1996, 2000, 2030, 2100, 2400 vallen onder de jaartelling van Gregorius XIII. Van deze jaren zijn 1600, 1920, 1996, 2000, en 2400 schrikkeljaren en 1900, 1969, 2030 en 2100 niet.

Opdracht 1.6
Stel het is vandaag 27 april 2006, dus jaartal = 2006, maand = 4 en dag = 27.

maand > 2, dus maand = maand - 2 = 2.

eeuw = 20, jaar = 06
A = (13 x 2 – 1)/ 5 = 25 / 5 = 5
B = 1
C = 6
D = 5 + 1 + 6 + 27 + 5 – 2 x 20 + 700 = 704

D / 7 =704 / 7 = 100 met rest 4

27 april 2006 valt dus op dag 4 = donderdag

Opdracht 1.7
Om een decimaal getal in een binair getal om te zetten moet worden nagegaan uit welke machten van 2 het decimale getal is opgebouwd.

Voorbeeld:
187 = **1** x 128 + **0** x 64 + **1** x 32 + **1** x 16 + **1** x 8 + **0** x 4 + **1** x 2 + **1** x 1 = 10111011

Om te bepalen of je een bepaalde macht van 2 nodig hebt voor het opbouwen van het decimale getal gebruik je het volgende algoritme:

- begin met de grootste macht van 2, 2^7 (N.B. $2^7 = 128$).
- deel het decimale getal door deze macht, 2^7.
- is de uitkomst groter of gelijk aan 1, dan is deze macht van 2 nodig: “1”.
- is de uitkomst kleiner dan 1, dan is deze macht van 2 niet nodig: “0”.
- als de macht nodig is, haal dan deze macht van het decimale getal af.
- herhaal de bovenstaande stappen met 2^6 , 2^5 , 2^4 …. , 2^1 .

Opdracht 1.8
Een priemgetal is een getal dat slechts deelbaar is door 1 en zichzelf. Om te bepalen of een getal n een priemgetal is moet dus worden nagegaan of er geen andere delers zijn dan 1 en het getal zelf. Een mogelijk algoritme is:
Ga voor alle getallen 2 t/m $n – 1$ na of het een deler is van n.
Zijn geen van deze getallen een deler van n, dan is n een priemgetal.