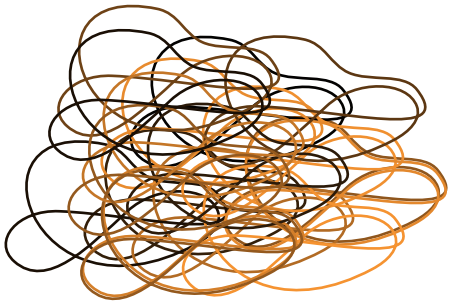
Programmeren met Asymptote

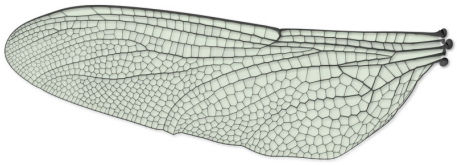
Wilfried Van Hirtum

Versie 2.15
09 augustus 2017



First, solve the problem.
Then, write the code.

— John Johnson



When I am working on a problem,
I never think about beauty.
I Think only how to solve the problem.
But when I've finished,
If the solution isn't beautiful,
I know it's wrong.

— Buckminster Fuller

Bronvermelding

De tekening op de titelpagina is ontworpen samen met mijn jongste zoon.

De foto in het voorwoord is met dank ontleend aan Cristóbal Vila:
http://www.etereaestudios.com

Voorwoord

The most exciting phrase to hear in science,
the one that heralds new discoveries,
is not "Eureka!" (I found it!)
but "That's funny ..."

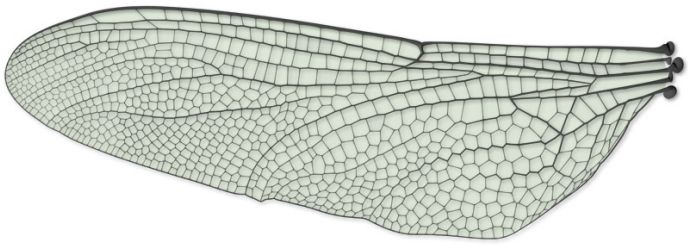
— Isaac Asimov (1920 - 1992)

Veel mensen vinden programmeren een leuke bezigheid, omdat het gereedschap betreft dat de nieuwsgierigheid, de creativiteit en het doorzettingsvermogen voortdurend prikkelt. Bovendien beloont de computer op de een of andere manier altijd jouw inspanningen. Je krijgt precies wat je voor ogen had, of er verschijnt een onverwacht resultaat of een foutmelding, de start voor een nieuwe uitdaging.

Asymptote is in de eerste plaats gericht op het produceren van vectortekeningen voorzien van LaTeX-typografie, dus van professionele kwaliteit. Een vectortekening bestaat uit lijnen en kleuren, maar tekenen is veel meer dan zomaar lijnen trekken. Een complexe tekening ontleden in patronen, en deze vervolgens zo elegant mogelijk beschrijven, dat is programmeren.

Asymptote is een beschrijvende programmeertaal, zodat je complexe taken kunt verwoorden met eenvoudig leesbare code. Het ontwerpen van figuren vraagt dus een minimum aan inspanning, en geeft een maximum aan resultaat. Daarom belooft deze programmeertaal dus heel wat 'fun'.

Wilfried Van Hirtum



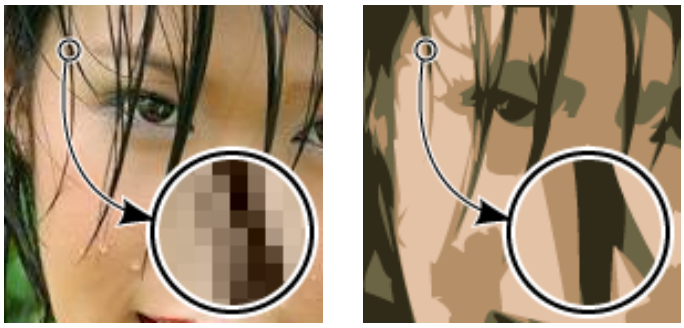
Inhoudsopgave

1 Beginnen met Asymptote

1.1 Raster versus vector

Je kent zeker wel enkele programma's om afbeeldingen te bewerken. Ze zijn grofweg in te delen in twee categorieën. Raster-gebaseerde image editors, zoals MS Paint, Photoshop en GIMP draaien rond het bewerken van afzonderlijke pixels, in tegenstelling tot vector-gebaseerde image editors zoals Adobe Illustrator en Inkscape, die lijnen en vormen bewerken.

Rasterafbeeldingen bestaan uit pixels: mininscule kleine vierkantjes in een bepaalde kleur. Vectorafbeeldingen bestaan uit wiskundig berekende lijnen die punten met elkaar verbinden zodat bepaalde vormen ontstaan. Als je een rasterafbeelding vergroot, dan worden de pixels zichtbaar als vierkanten en de tekening vervaagt. Een vectorafbeelding gebruikt wiskundige formules om de afbeelding op te bouwen en kan dus worden vergroot zo groot als je maar wilt zonder verlies aan kwaliteit. Zie figuur 1 op pagina 7.



(a) Rasterafbeelding **(b)** Vectorafbeelding

Figuur 1 – Een rasterafbeelding vervaagt bij vergroting, omdat de afzonderlijke vierkante pixels zichtbaar worden. Een vectorafbeelding daarentegen blijft even scherp bij een vergroting, omdat de computer alle lijnen opnieuw berekent.

Het sterke punt van een rasterafbeelding is dat je elke pixel afzonderlijk kunt bewerken, meestal door middel van een filter. Dit laat subtiele overgangseffecten toe. Het rasterformaat werkt dus best met realistische foto's. Typische bestandsformaten voor rasterafbeeldingen zijn jpg, png, gif, tiff en bmp.

Het sterke punt van een vectorafbeelding is dat je ze eindeloos kunt transformeren zonder verlies aan informatie. Vectorafbeeldingen komen vooral voor als logo's, technische tekeningen, landkaarten, grafieken, enzovoort. Typische bestandsformaten voor vectorafbeeldingen zijn pdf en eps.

Merk op dat vector-afbeeldingen, als ze op papier of op het beeldscherm verschijnen, tijdelijk rasterafbeeldingen worden. De kwaliteit van deze afbeelding hangt dan af van de resolutie van de printer of van het beeldscherm. De resolutie van een rasterafbeelding op het beeldscherm is gewoonlijk 72 dpi. In een vierkante inch van het beeldscherm bevinden zich dus 72×72 dots of pixels. De resoluties van printers zijn veel hoger, bijvoorbeeld 1200 dpi. Maar de essentie van een vectorafbeelding is, dat wanneer hij vergroot wordt, opnieuw berekend wordt, en dus scherper wordt als de resolutie van het weergevende apparaat groter wordt.

1.2 De programmeertaal Asymptote

Asymptote is een beschrijvende programmeertaal voor vectorafbeeldingen, wat er op neerkomt dat je complexe vormen kunt beschrijven in functie van veelhoeken, cirkels, rechte en gebogen lijnen. Om herhalende onderdelen te beschrijven, kun je gebruik maken van typische programmastructuren zoals de iteratie.

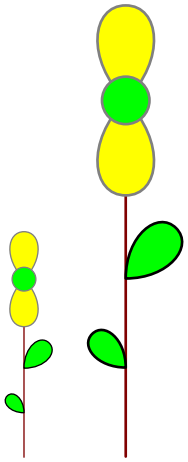
Een voorbeeld

Het volgende Asymptote-programma produceert een boterbloem in de vorm van een vectorafbeelding. Zonder de details te kennen van de programmeertaal Asymptote, is het programma al vrij vlot leesbaar. Probeer maar eens. Het is gemakkelijk om je een voorstelling te maken van het eindproduct. Het programma beschrijft in enkele regels dat de boterbloem bestaat uit vijf gele bloemblaadjes, een groen hart, een bruine steel en twee groene stengelblaadjes. De afzonderlijke componenten worden samengevoegd tot een geheel, de boterbloem. Het programma is gemakkelijk te wijzigen tot een (gemuteerde) boterbloem met bijvoorbeeld zeven bloemblaadjes. Dat komt omdat er handig is gebruik gemaakt van de iteratie, een programmastructuur om bepaalde handelingen te herhalen. De resulterende vectorafbeelding staat in figuur 2 op pagina 9.

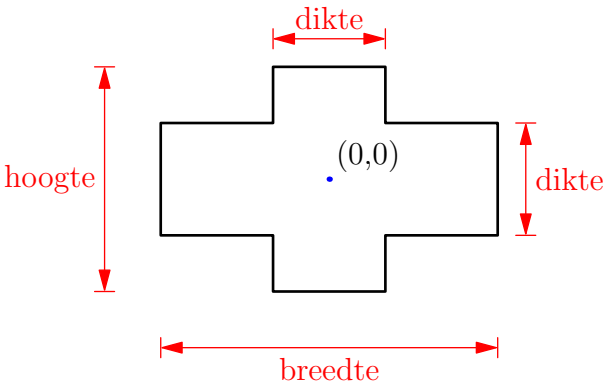
```
1  //boterbloem.asy
2  size (8cm);
3  picture boterbloem;
4  picture bloemblaadje, hart, steel, stengelblaadje;
5  real straal=1;
6  pair M=(0, 0), A=(0, 4*straal);
7  pair B=(0, -15*straal);
8  path cirkel=circle(M, straal);
9  path blad=M{NE}..A..{SE}cycle;
10 draw (steel, M--B, brown+2);
11 filldraw(hart, cirkel, green, gray+2);
12 filldraw(bloemblaadje, blad, yellow, gray+2);
13 filldraw(stengelblaadje, blad, green);
14 int n=5;
15 add(boterbloem, steel);
16 transform draai=rotate(360/n);
17 pair C=(M+B)/2, D=(C+B)/2;
18 add(boterbloem, shift(C)*rotate(-45)*scale(0.75)*stengelblaadje);
19 add(boterbloem, shift(D)*rotate(45)*scale(0.5)*stengelblaadje);
20 for (int i = 0; i <n ; ++i){
21         add(boterbloem, draai^i*bloemblaadje);
22 }
23 add(boterbloem, hart);
24 add(boterbloem);
```

1.3 Asymptote en LaTeX

Er is een hechte samenwerking tussen de programmeertaal Asymptote en de opmaaktaal pdfLaTeX. LaTeX verzorgt het zetten van alle tekstonderdelen op een afbeelding, zodat de afbeelding er nog professioneler uitziet. Voorbeeld:



Figuur 2 – Het resultaat van het Asymptote-programma `boterbloem.asy` is een vectorafbeelding. De cirkels en lijnen waaruit de afbeelding is opgebouwd, zijn het resultaat van wiskundige berekeningen. Bij vergroting blijven de lijnen even scherp.



Figuur 3 – LaTeX staat in voor de professionele kwaliteit van alle tekstonderdelen in een Asymptote-afbeelding. Tekstkarakters zijn zelf mini-vectorafbeeldingen.

Verder is het de bedoeling dat je de afbeeldingen die je met behulp van Asymptote maakt, ook daadwerkelijk gebruikt, bijvoorbeeld in tex-bronteksten, door middel van het commando `\includegraphics`.

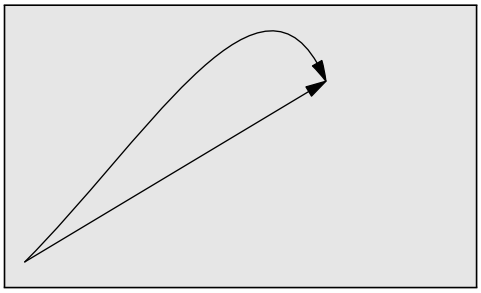
PdfLaTeX kan overweg met pdf-afbeeldingen (al dan niet vectorafbeeldingen).

Je kunt de pdf-afbeeldingen die je met Asymptote gemaakt hebt, desgewenst converteren naar een png-afbeelding, zodat je ze kunt gebruiken in een webpagina. Zie *LaTeX voor beginners – deel 1* (Van Hirtum, *LaTeX voor beginners - deel 1*) om met behulp van het programma ImageMagick afbeeldingen te converteren naar andere formaten.

PdfLaTeX kan ook overweg met raster-afbeeldingen in het formaat jpg of png. Gif, tiff en bmp moeten eerst worden omgezet naar bijvoorbeeld png door middel van het converteerprogramma ImageMagick. Maar we gaan het in boek uiteraard hebben over het maken van vectorafbeeldingen, die we bewaren in pdf-formaat.

1.4 Vectoren

```
//voorbeeldpijlen.asy
size(4cm);
picture rechtepijl;
pair O=(0,0);
pair P=(5,3);
path lijntje=O--P;
draw(rechtepijl, lijntje, Arrow);
picture gebogenpijl;
path boog=O{(1,1)}..{SSE}P;
draw(gebogenpijl, boog, Arrow);
add(rechtepijl);
add(gebogenpijl);
```



Vectoren spelen dus een hoofdrol bij het beschrijven van vectorafbeeldingen. Een tweedimensionale vector heeft een lengte en een richting. Je beschrijft ze in Asymptote door middel van een koppel van twee coördinaatgetallen. De vector $P = (5, 3)$ is een vector in de richting 'vijf stappen naar rechts, drie stappen naar boven' ten opzichte van de oorsprong $O = (0, 0)$. Asymptote kent ook driedimensionale vectoren.

Je ziet in het programma `voorbeeldpijlen.asy` dat vectoren ook worden gebruikt om richtingen aan te geven. In plaats van langs een rechte lijn van het punt `O` naar het punt `P` te gaan, is het ook mogelijk om het traject langs een gebogen pad af te leggen. Je geeft dan in de programmacode de beginrichting en de eindrichting aan van het gebogen pad. De vector $(1, 3)$ betekent richting noordoost: 'een stap naar rechts, een naar boven'. De vector `SSE` is de ingebouwde richtingsvector zuidzuidoost, oftewel de richtingsvector $(1, -2)$.

1.5 Programmeren, hoe gaat dat?

Programmeren lijkt een beetje op *wiskunde*: je gebruikt ook een formele taal om oplossingen te beschrijven. Hoe meer structuur je legt in het programma, hoe mooier het wordt, net zoals in de wiskunde.

Programmeren heeft ook iets van het werk van een *ingenieur*: je voegt afzonderlijke componenten samen tot een werkend geheel, en je zoekt daarbij naar de meest efficiënte technieken.

Programmeren heeft ook iets van *wetenschappelijk onderzoek*: je onderzoekt een complex probleem, doet veronderstellingen over de werking van een model, en test het model uit.

Soms gebeurt het dat je een *onverwacht*, maar toch bruikbaar resultaat krijgt dat op zijn beurt nieuwe perspectieven opent voor verder onderzoek. Zeer veel ontdekkingen zijn een of andere vorm van een serendipiteit.

Problem solving

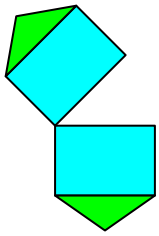
De kernactiviteit van programmeren is *problem solving*:

1. Formuleer het probleem eenduidig. Een schets is vaak nodig.
2. Beschrijf de oplossing helder en exact. Denk daarbij aan de uitspraak van John Johnson:

   First, solve the problem.
   Then, write the code.

3. Test de oplossing uitvoerig in alle mogelijke omstandigheden.

1 Voer de drie stappen van *problem-solving* uit om de volgende figuur te tekenen. Schrijf nog geen Asymptote-programma.



Figuur 4 – Problem-solving toepassen op deze figuur

2 Ontwar het kluwen in figuur 5 en formuleer in eigen, maar duidelijke bewoordingen hoe je deze figuur zelf zou tekenen.

Figuur 5 – Ontrafel dit kluwen

1.6 Programmatekst typen in een editor

Je hebt in de eerste plaats een teksteditor nodig. Een editor is een computerprogramma dat het mogelijk maakt platte tekst in de computer in te voeren en te wijzigen. Platte tekst is tekst zonder opmaak.

Je hebt al een editor gebruikt voor het typen van LaTeX-broncode, bijvoorbeeld WinShell (Windows) of Geany (Windows en Linux). Zie *LaTeX voor beginners – deel 1* (Van Hirtum, *LaTeX voor beginners - deel 1*). Je kunt WinShell in principe ook gebruiken om een progrommatekst te typen, maar er zijn betere alternatieven, bijvoorbeeld Geany.

Figuur 6 – Programmacode typen in een editor

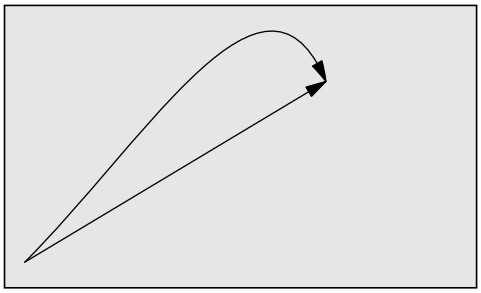
Goede teksteditors 'herkennen' en markeren de typische sleutelwoorden van een programmeertaal. We noemen dit *syntaxismarkering*. Elke regel krijgt ook een nummer, handig bij het lokaliseren van syntaxisfouten. Dat is de enige opmaak die een teksteditor toelaat. Voor de rest is de programmatekst platte tekst.

http://www.geany.org/

Geany is een zeer goede teksteditor die geschikt is voor verschillende programmeertalen, ook voor LaTeX, en zeker voor Asymptote. Ik werk persoonlijk met Geany zowel voor LaTeX (in Linux) als voor het programmeren in Asymptote. Geany is een lichtgewicht teksteditor ontwikkeld onder de GNU General Public Licence, vrij verkrijgbaar, start en sluit snel. Geany heeft bovendien een echte kolommodus, waardoor je heel gemakkelijk verticale blokken tekst kunt bewerken. Je zult snel doorhebben waarvoor dat handig kan zijn.

1.7 Een voorbeeld

```
1  //voorbeeldpijlen.asy
2  size(4cm);
3  picture rechtepijl, gebogenpijl;
4  pair O=(0,0);
5  pair P=(5,3);
6  path lijntje=O--P;
7  draw(rechtepijl, lijntje, Arrow);
8  path boog=O{(1,1)}..{SSE}P;
9  draw(gebogenpijl, boog, Arrow);
10 add(rechtepijl);
11 add(gebogenpijl);
```



Dit programma bestaat uit een aantal commando's om een pijl te tekenen. Het resultaat is de afbeelding `voorbeeldpijlen.pdf`.

Om dit resultaat te krijgen, moet je de programmtekst typen in een editor, bijvoorbeeld in Geany, en het programma compileren en uitvoeren met een sneltoets, bijvoorbeeld F9.

De betekenis van de verschillende commando's:

- `//voorbeeldpijlen.asy`
  Dit is een commentaarregel. Commentaarregels beginnen met een dubbele schuine streep `//` in tegenstelling tot commentaarregels in LaTeX, die met een procentteken % beginnen. Commentaarregels dienen voor de programmeur om het programma te verduidelijken en zijn dus geen echte commando's.
- `size(4cm);`
  De geproduceerde afbeelding wordt maximaal 4 cm breed of hoog. In dit geval past de afbeelding in een rechthoek van 5 op 3 en is de grootste afmeting dus de breedte van de rechthoek.
- `picture rechtepijl, gebogenpijl;`
  Je definieert een figuur met als zelfgekozen naam 'rechtepijl'. Je kunt meerdere figuren definiëren en die achteraf allemaal samenvoegen tot een enkele afbeelding. Door met aparte figuren te werken, kun je een ingewikkelde afbeelding opsplitsen in afzonderlijke eenvoudigere figuren.
- `pair O=(0,0);`
  De variabele `O` stelt een punt voor met coördinaat $(0,0)$. Let op het verschil tussen de letter 'O' en het cijfer nul '0'.
- `pair P=(5,3);`
  De variabele `P` is van het type `pair` (koppel van twee coördinaatgetallen) en krijgt de waarde $(5,3)$.
- `path lijntje=O--P;`
  De variable `lijntje` is van het type `path` en is gedefineerd als het rechtlijnig (`-`) verbindingslijntje tussen de punten `O` en `P`.
- `draw (rechtepijl, lijntje, Arrow);`
  `draw (gebogenpijl, boog, Arrow);`
  Het pad `lijntje` wordt aan de figuur `rechtepijl` toegevoegd, voorzien van een pijlpunt aan het eindpunt. Het pad `boog` wordt op analoge manier aan de figuur `gebogenpijl` toegevoegd.
- `path boog=O..P;`
  `path boog=O{(1,1)}..{SSE}P;`

De variable `boog` is van het type `path` en is gedefineerd als het gebogen `(..)` verbindingslijntje tussen de punten `0` en `P`. De begin- en eindrichting van de boog moeten tussen accolades staan. Deze richtingen zijn de richtingsvectoren $(1, 1)$ (noordoost) en $(1, -2)$ oftewel zuidzuidoost.

- `add (rechtepijl); add (gebogenpijl);`
  De figuren `rechtepijl` en `gebogenpijl` worden toegevoegd aan de afbeelding.

Puntkomma

Merk op dat elk commando in het Asymptote-programma eindigt op een puntkomma (;). Dit is verplicht.

1.8 Het programma uitvoeren

Je hebt de programmatekst in een editor getypt, een naam gegeven, bijvoorbeeld `voorbeeldpijlen.asy`. Je kunt het programma vervolgens uitvoeren zodat het programma een vectorafbeelding produceert, en je kunt de afbeelding bekijken met een pdf-viewer. Dit kan in de editor Geany met een enkele sneltoets F9. Zie verder sectie 4 op pagina 20 om de editor Geany op de juiste manier in te stellen.

Uitvoeren

Programma uitvoeren en de pdf bekijken: F9

Sluit de pdf om terug te keren naar de editor Geany.

1.9 Bestanden in dezelfde map

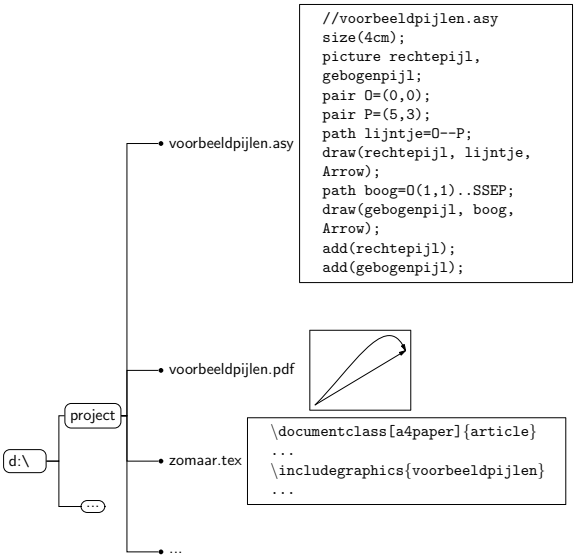
Het resultaat is de afbeelding `voorbeeldpijlen.pdf` die in dezelfde map wordt opgeslagen als de programmetekst `voorbeeldpijlen.asy`.

Je kunt deze afbeelding ook apart bekijken met een pdf-viewer, bijvoorbeeld Foxit Reader of Adobe Reader, en verder bewerken, bijvoorbeeld bijsnijden in een gepast programma zoals bijvoorbeeld F-Spot in Ubuntu, en je kunt de afbeelding ook gebruiken in een pdfLaTeX-bronbestand.

In figuur 7 op pagina 15 staat een overzichtelijke bestandenboom.

Naamgeving bestanden

Let op: zorg er voor dat de naam van de pdf-afbeeldingen *verschillend* is van de naam van de brontekst. Bijvoorbeeld, een brontekst `zomaar.tex` met een ingesloten afbeelding met dezelfde naam `zomaar.pdf` geeft problemen omdat LaTeX ook een pdf produceert, namelijk `zomaar.pdf`, zodat de oorspronkelijke afbeelding `zomaar.pdf` verloren gaat.



Figuur 7 – De programmacode `voorbeeldpijlen.asy` en de pdf-afbeelding `voorbeeldpijlen.pdf` staan in dezelfde map. Eventueel staat in deze map ook een LaTeX-brontekst, bijvoorbeeld `zomaar.tex`, met daarin het commando `\includegraphics` om de afbeelding `voorbeeldpijlen.pdf` in te sluiten.

2 Software installeren in Ubuntu

Alle benodigde software is vrije software en gratis te downloaden.

Als je met Windows werkt, sla dan deze sectie over en ga direct naar sectie 3 op pagina 17.

2.1 LaTeX installeren in Ubuntu

Je beschikt over een computer waarop Ubuntu 10.04 (of een latere versie) is geïnstalleerd.

Je hebt LaTeX reeds geïnstalleerd volgens de richtlijnen beschreven in het boek *pdfLaTeX voor beginners – deel 1* (Van Hirtum, *LaTeX voor beginners - deel 1*).

2.2 Geany installeren in Ubuntu

Je hebt de teksteditor Geany 0.19 (of een latere versie) reeds geïnstalleerd volgens de richtlijnen beschreven in het boek *pdfLaTeX voor beginners – deel 1* (Van Hirtum, *LaTeX voor beginners - deel 1*).

2.3 Asymptote installeren in Ubuntu

Typ het volgende commando in een terminal:

```
Terminal
sudo apt-get install asymptote
```

2.4 Een pdf-viewer kiezen voor Asymptote

Kies een standaard pdf-viewer voor Asymptote, bijvoorbeeld 'Evince'. Maak zelf het volgende bestand aan:

`/home/username/.asy/config.asy`

Dit is een nieuw bestand. Open dit bestand met een teksteditor, bijvoorbeeld Geany, en typ de volgende twee regels in dit bestand:

```
import settings;
pdfviewer="evince";
```

Sluit het bestand `config.asy`

Ga nu door naar sectie 4 op pagina 20 om de teksteditor Geany te configureren om te programmeren in Asymptote.

3 Software installeren in Windows

Alle benodigde software is vrije software en gratis te downloaden.

3.1 LaTeX installeren in Windows

Je beschikt over een computer waarop Windows (XP, Vista of 7) is geïnstalleerd.

Je hebt LaTeX reeds geïnstalleerd volgens de richtlijnen beschreven in het boek *pdfLaTeX voor beginners – deel 1* (Van Hirtum, *LaTeX voor beginners - deel 1*).

3.2 Geany installeren in Windows

Voer het volgende bestand uit:

`geany-0.19_setup.exe`

Dit bestand staat op de `latexcdrom`, of je kunt het downloaden op de volgende website (rubriek Download/Windows Binaries):

`http://www.geany.org/`

Accepteer alle voorstellen tijdens de installatie.

Voer het volgende bestand uit:

`geany-plugins-0.19_setup.exe`

Dit bestand staat eveneens op de `latexcdrom`, of je kunt het downloaden op de volgende website (rubriek Download/Windows Binaries):

`http://plugins.geany.org/geany-plugins/`

Start de editor Geany, via de Windowsknop op de taakbalk en typ het commando: `geany` gevolgd door Enter.

Sluit daarna Geany terug. Dit is nodig opdat Geany bij de eerste opstart enkele mappen aanmaakt, waarin we zo dadelijk enkele nodige wijzigingen gaan aanbrengen.

3.3 Asymptote installeren in Windows

Voer het volgende bestand uit:

`asymptote-x.xx-setup.exe`

Op de plaats van '`x.xx`' staat een versienummer, bijvoorbeeld '2.02' of een latere versie. Dit bestand staat op de `latexcdrom`, of je kunt het downloaden op de volgende website (rubriek Download):

`http://asymptote.sourceforge.net/`

Accepteer alle voorstellen tijdens de installatie.

3.4 Foxit Reader installeren in Windows

Je hebt Foxit Reader (versie 4.0 of eeen latere versie) reeds geïnstalleerd volgens de richtlijnen beschreven in het boek *pdfLaTeX voor beginners – deel 1* (Van Hirtum, *LaTeX voor beginners - deel 1*).

Je kunt best de volgende instelling aanpassen:

Foxit Reader → `Tools` → `Preferences`
→ `Page Display` → `Magnification` → `Actual size`

Vanaf nu toont Foxit Reader de pdf-afbeeldingen op ware grootte als je ze op het scherm bekijkt.

3.5 Windows instellen

Zorg er voor dat de snelle pdf-viewer Foxit Reader de standaard pdf-viewer wordt voor Asymptote, en dat Asymptote kan worden gestart aan de opdrachtprompt van Windows:

- Ga naar de geavanceerde systeeminstellingen van Windows:
  - In Windows XP:
    Start → rechts klikken op `Deze computer`
  - In Windows Vista en Windows 7:
    `Verkenner` → rechts klikken op `Computer`
- Vanaf hier zelfde werkwijze voor Windows XP, Windows Vista en Windows 7:
  → `Eigenschappen` → tabblad `Geavanceerd`
  → Omgevingsvariabelen
  → onderste kader `Systeemvariabelen` → Nieuw
  → `Naam:` `asymptote_pdfviewer` (zelf typen, let op het onderlijningsteken)
  → `Waarde:` c:\Program Files\foxit software\foxit reader\Foxit Reader.exe
  (zelf typen)
  → OK
  → nog steeds in onderste kader `Systeemvariabelen`
  → Kies in de lijst `Path` → Bewerken
  → Vul het bestaande pad aan met: ;c:\program files\asymptote
  (zelf typen)
  → OK
  → OK → OK

Zorg er voor dat bestanden met extensie `.asy` automatisch geopend worden met Geany:

- In Windows XP:
  Open Verkenner `Extra` → `Mapopties`→ tabblad `Bestandstypen` → Nieuw
  → `Bestandsextensie:` asy (zelf typen) → OK
  → `De ASY-extensie openen met:` Wijzigen
  → ⦿ `Het programma in een lijst selecteren` → OK
  → Bladeren → `C:\Program Files\Geany\geany.exe`
  → Openen → OK → Sluiten
- In Windows Vista en Windows 7:

Ga naar een map waarin een `asy`-programma staat, bijvoorbeeld in de map `c:/project` het bestand `zomaar.asy`. Als je zo'n bestand nog niet hebt, maak er dan zelf een aan als volgt: open Geany, open een nieuw bestand (Ctrl - N) en sla dit leeg bestand op (Ctrl - S) onder bijvoorbeeld de naam `zomaar.asy` (extensie `.asy` ook typen) in bijvoorbeeld de map `c:\project`.

→ open Verkenner
→ ga naar de map `c:/project` (of een ander map met `asy`-bestanden)
→ rechts klikken op een `zomaar.asy` (of een ander `asy`-bestand)
→ Openen met
→ Standaardprogramma selecteren...
→ Bladeren
→ blader naar `c:\program files\Geany\geany.exe`
→ ✓ Dit type bestand altijd met dit programma openen
→ Openen
→ OK

De extensie `.asy` is nu verbonden met Geany. Als je in de Windows verkenner dubbelklikt op een `asy`-bestand, wordt het bestand geopend in Geany.

Ga nu door naar sectie 4 op pagina 20 om de teksteditor Geany te configureren om te programmeren in Asymptote.

4 Geany configureren voor Asymptote

4.1 Configuratiebestanden downloaden en uitpakken in de juiste map

Je hebt reeds de nodige software geïnstalleerd, maar de editor Geany heeft nog informatie nodig om goed te kunnen werken met Asymptote. Deze informatie staat in een aantal zogenaamde configuratiebestanden. Om het jou gemakkelijk te maken, heb ik deze configuratiebestanden reeds aangemaakt en verzameld in een archiefbestand. Je vindt dit bestand op mijn website:

`http://users.skynet.be/denkendehanden/asymptote.html`

Je kiest voor het juiste bestand, afhankelijk van jouw besturingssysteem:

Ubuntu:	Windows:
`geanyconfiglinux.zip`	`geanyconfigwindows.zip`

Download dit archiefbestand, en bewaar het voorlopig in de volgende map:

Ubuntu:	Windows:
`/home/username/latexcdrom`	`c:/latexcdrom`

Open vervolgens het archiefbestand, en pak alle bestanden en mappen uit naar de juiste map, afhankelijk van jouw besturingssysteem:

Ubuntu:	Windows:

`/home/username/.config/geany/`	(Vista/7)
	`c:/Gebruikers/xxx/AppData/Roaming/Geany/`
	(Windows XP)
	`c:/Documents And Settings/xxx/Application Data/Geany/`

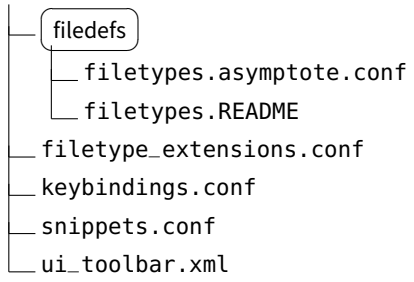
Ubuntu: De naam `username` is jouw gebruikersnaam. Let op, `.config` is een verborgen map. Je kunt verborgen mappen zichtbaar maken in Ubuntu met behulp van Ctrl - H.	Windows: De naam 'xxx' is uw eigen gebruikersaccountnaam van Windows. Let op: als deze map niet zichtbaar is in de verkenner, zet dan de mapoptie om verborgen mappen en bestanden zichtbaar te maken aan.

Controleer even of je nu over de juiste configuratiebestanden beschikt: In de zojuist genoemde map —afhankelijk van jouw besturingssysteem— moeten nu de volgende bestanden staan: zie figuur 8 op pagina 21.

/home/username/.config/geany/ (Ubuntu).

c:/Gebruikers/xxx/AppData/Roaming/Geany/ (Windows Vista/7).

c:/Documents and Settings/xxx/Application Data/Geany/ (Windows XP).



Figuur 8 – De configuratiebestanden van Geany. Controleer of de juiste bestanden in de juiste map staan.

Een woordje uitleg bij de verschillende configuratiebestanden

Het bestand `filetypes.asymptote.conf` is het belangrijkste configuratiebestand. Hierin staat beschreven:

- welk het commentaarteken is voor de programmeertaal Asymptote;
- welke sleutelwoorden van Asymptote hij moet kleuren (syntaxis-markering);
- welke commando's hij moet geven om een asy-programma te compileren en uit te voeren.

In het bestand `filetype_extensions.conf` staat beschreven dat Geany het bestand `filetypes.asymptote.conf` moet gebruiken, zodra je een tekstbestand in Geany hebt bewaard met de extensie `.asy`.

Het bestand `filetypes.asymptote.conf` zit standaard niet bij een nieuwe installatie van Geany. Je moet het in principe dus zelf maken. Ik heb ten behoeve van jullie, lezers, een dergelijk bestand gemaakt.

In het bestand `keybindings.conf` staan de sneltoetsinstellingen van Geany. Geany bevat naast vele andere, ook drie interessante bewerkingen waar nog geen sneltoets voor bestaat: regels omhoog of omlaag verplaatsen, regels commentariëren of ontcommentariëren, en een bestand 'opslaan als'. Ik heb aan deze acties sneltoetsen verbonden: Ctrl-Up, Ctrl-Down, Ctrl-M Ctrl-Shift-M en Ctrl-Shift-S. Deze informatie staat dus in het bestand `keybindings.conf`.

In het bestand `ui_toolbar.xml` staan de instellingen van de werkbalk van Geany. Ik heb deze werkbalk bewust erg sober gemaakt, zo sober dat er alleen maar de kleurkiezer (een handige plugin) en de zoekbalk in staat.

4.2 De syntaxismarkering in Geany testen

Vanaf nu moet Geany programma's met de extensie `asy` herkennen als Asymptote-programma's.

We gaan dit even testen. Typ nu een minimaal voorbeeld van een asymptote-programma in de editor Geany, bijvoorbeeld:

```
size(5cm);
```

```
write (7*8);
picture fig;
filldraw(fig, unitcircle, green);
label(fig, "$\pi$", (0,0));
add(fig);
```

en bewaar deze tekst onder de volgende naam:

`zomaar.asy`

bijvoorbeeld in de volgende map:

Ubuntu:	Windows:
/home/username/project	`c:/project`

Vanaf het ogenblik dat je het bestand hebt bewaard met de extensie `asy`, herkent de editor Geany deze tekst als een Asymptote-programma. Je ziet dat aan de syntaxiskleuring. De sleutelwoorden `size` en `write` onder andere zijn bruin gekleurd.

Controleer in het menu van Geany, terwijl het bestand `zomaar.asy` nog open staat, de volgende instellingen:

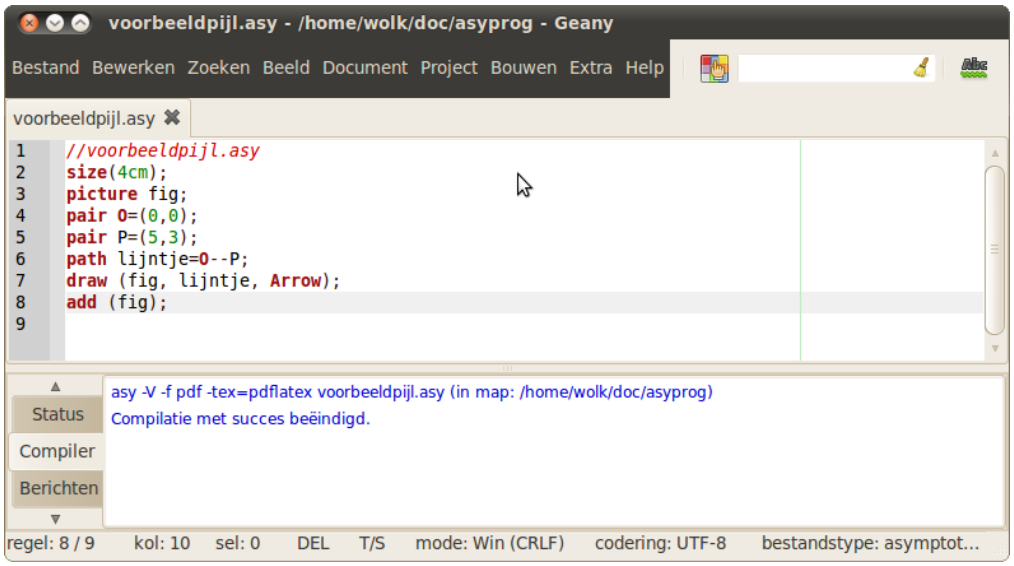
`Bouwen` → `Bouwcommando's instellen` →
`asymptote bronbestand commando's:`
→ asymptote: `asy -V -f pdf -tex=pdflatex %f`

Je kunt de brontekst nu compileren met behulp van de sneltoets F9. De compiler stopt als er een fout wordt ontdekt, en de foutmelding wordt in het rood weergegeven. Als je op deze rood weergegeven foutmelding klikt, gaat de cursor naar de regel waar de fout is ontdekt.

Als je kunt, mag je de fouten proberen te herstellen. Anders wacht je tot na sectie 6.1 op pagina 29. We bespreken in die sectie enkele veel voorkomende soorten van fouten.

Maar je ziet nu alleszins dat Geany asy-programma's herkent.

5 De editor Geany gebruiken



Figuur 9 – Geany is een kleine maar fijne teksteditor.

Geany is een zeer kleine, maar fijne en vooral supersnelle teksteditor, ideaal om programmateksten te typen, te bewerken en uit te voeren. Zie figuur 9.

Geany is zo vriendelijk om automatisch alle bestanden te openen die nog open stonden tijdens de laatste sessie. De positie van de cursor wordt aangegeven in de statusbalk: zowel regelnummer als kolomnummer. Er is een uitvoervenster voor berichten, resultaten en foutmeldingen. Er is onder andere syntaxismarkering en regelnummering, en er is de zeer handige kolommodus.

5.1 Nieuw

Om een nieuw Asymptote-programma te beginnen:

Ctrl - N

5.2 Bewaren als

Om een Asymptote-programma te bewaren onder een bepaalde naam:

Ctrl-Shift - S

Extensie ook typen

Je moet de bestandsextensie `.asy` ook typen!

Gebruik geen spaties in bestandsnamen.

`huis-met-tuin.asy`	~~`huisje met tuintje.asy`~~
Goed	Fout

5.3 Syntaxismarkering

Van zodra de bestandsextensie getypt is, herkent Geany de typische syntaxiselementen van de programmeertaal. De syntaxis van Asymptote komt overeen met die van C++.

De sleutelwoorden van Asymptote (`size`, `pair`, `path`, `draw`, `Arrow`, `write`, `pi`, enzovoort) worden in het bruin weergegeven. En de typische sleutelwoorden die ook in de programmeertaal C++ voorkomen, worden in het blauw weergegeven.

5.4 Hoofdlettergevoeligheid

Asymptote is hoofdlettergevoelig voor de notatie van de sleutelwoorden. De meeste sleutelwoorden worden in kleine letters geschreven.

`size (4cm);` `Arrow`	~~`Size(4cm);`~~ ~~`arrow;`~~
Goed	Fout

5.5 Execute

Om een Asymptote-programma te compileren en uit te voeren:

F9

Voor je terugkeert naar de editor Geany, moet je de pdf-viewer sluiten.

5.6 Berichtenvenster van Geany

In het berichtenvenster (onderaan) verschijnen berichten van Asymptote. Dat kunnen foutmeldingen zijn, of uitkomsten van berekeningen, of de melding dat de compilatie foutloos is verlopen.

```
real functie(real x){
        return x^2+8;
}
write (functie(3));
write (functie(0));
write (functie(1));
```

Berichtenvenster

```
17
8
9
```

Je kunt de inhoud van het berichtenvenster kopiëren:

→ Klik rechts in het berichtenvenster

→ `Kopieer` of `Alles kopiëren`

→ Plak deze berichten naar de plaats waar je ze wilt gebruiken.

5.7 Scheidingstekens markeren

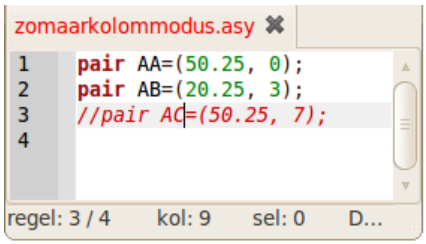
Geany markeert automatisch paren van ronde haakjes, rechte haken en accolades. Dit wil zeggen: als je met de cursor bijvoorbeeld op een rechteraccolade staat, wordt de bijbehorende linkeraccolade automatisch gemarkeerd met een underscore.

Ga naar het overeenkomende haakje/accolade:

Ctrl - B

5.8 Commentaar en ontcommentaren

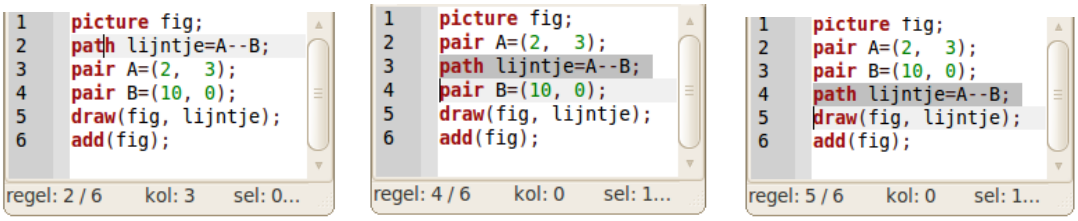
Ga ergens in een regel staan die je wilt omdopen tot commentaar met behulp van de sneltoets Ctrl - M. Als je dit met een regel doet die al commentaar was, wordt er één commentaarteken bijgevoegd. De sneltoets Ctrl - Shift - M verwijdert per indruk één commentaarteken. Voorbeeld:



Dit is geen standaardsneltoets van Geany, maar als je Geany hebt geconfigureerd zoals beschreven in sectie 4 op pagina 20, dan beschik je ook over deze sneltoets.

5.9 Verplaats regel omhoog/omlaag

Ga gewoon ergens in een regel staan die je wilt verplaatsen. Het maakt niet uit of je in het begin van deze regel staat, of ergens in het midden of op het einde. De sneltoets Ctrl-Up en Ctrl-Down verplaatst de huidige regel(s) een regel omhoog of omlaag. Voorbeeld:



Dit is een zeer handige sneltoets. Deze sneltoets vervangt in een keer vier stappen: selecteren-kopiëren-naar omhoog/omlaag-plakken.

Als je Geany hebt geconfigureerd zoals beschreven in sectie 4 op pagina 20, dan beschik je ook over deze sneltoets.

5.10 Kolommodus

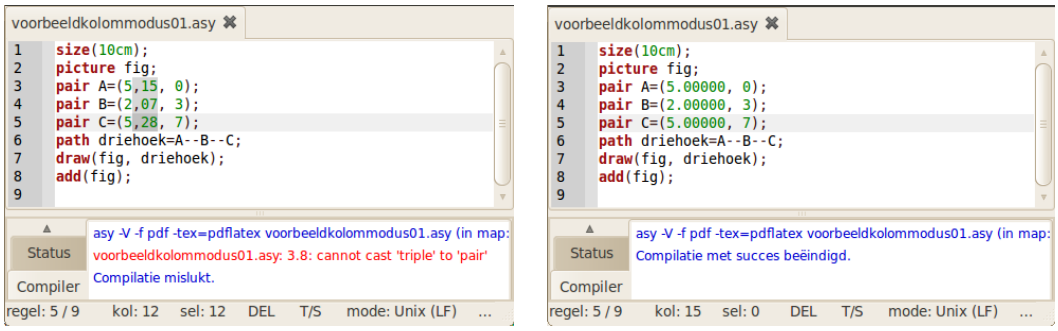
Geany bevat ook een echte kolommodus, handig bij het bewerken van verticale blokken programmatekst.

Ga ergens in de tekst staan, en houd de toetsen Alt en Shift gelijktijdig in, terwijl je rechthoekige blokken tekst selecteert met behulp van de pijltjestoetsen (niet met behulp van

de muis). Je ziet een dun verticaal lijntje dat het begin van een blok aangeeft. Je kunt dit geselecteerd blok groter of kleiner maken behulp van de pijltjestoetsen, terwijl je nog steeds de toetscombinatie Alt-Shift ingedrukt houdt.

Als je deze toetsen loslaat, zie je dat er een rechthoekig blok is geselecteerd. Nadien kan de selectie nog steeds worden aangepast. Houd dan terug de toetscombinatie Alt-Shift ingedrukt.

Voorbeeld:



Je kunt nu tekst bewerken in meerdere regels tegelijkertijd. Ik heb in dit voorbeeld in drie regels gelijktijdig de komma plus enkele cijfers na de komma geselecteerd en tevens gelijktijdig overschreven door de substring `.00000`.

Van zodra je naar een andere plaats in de tekst gaat, met behulp van de pijltjestoetsen of met behulp van de muis, houdt de rechthoekige selectie op. Je kunt nu op de gewone manier tekst typen of bewerken.

5.11 Sneltoetsen

In tabel 1 op pagina 28 zijn veelgebruikte sneltoetsen weergegeven.

Sommige van deze sneltoetsen zijn ook standaard in vele andere toepassingen, bijvoorbeeld Ctrl-N voor 'Nieuw bestand' of Ctrl-O voor 'Open bestand'. Omdat ze zo gewoon zijn, is het aan te raden om deze sneltoetsen zo te laten.

Voor meer geavanceerde sneltoetsen, zie het menu `Bewerken/Voorkeuren/Sneltoetsen` van Geany.

Je kunt, als je dat wenst, nieuwe sneltoetsen definiëren in Geany. Dit gaat als volgt:

`Geany` → `Bewerken` → `Voorkeuren` → `Sneltoetsen` → `Editor`
→ Kies een bewerking, bijvoorbeeld `Verplaats regel omhoog`
→ `Wijzigen` → Druk toetscombinatie, bijvoorbeeld Ctrl-Up → OK

> **Tip**
>
> Laat de muis zoveel mogelijk aan de kant staan. Leer vanaf het begin de sneltoetsen te gebruiken.

5.12 Snippets

Geany heeft nog veel meer mogelijkheden, en is bovendien nog uitbreidbaar met allerlei plugins. Een handige optie is het gebruik van code snippets. Dit zijn stukjes programmacode die je veel nodig hebt, en waarvoor je een afkorting kunt definiëren. Je typt die afkorting,

gevolgd door de Tab, en het stukje programmacode wordt automatisch aangevuld. Je kunt deze snippets zelf definiëren, binnen Geany. Je vindt ze in het menu:

`Extra` → `Configuratiebestanden` → `snippets.conf`

5.13 De handleiding van Geany

Je kunt de handleiding van Geany bekijken als een webpagina:

`Help` → `Help`

of gewoonweg: F1.

Tabel 1 – Sneltoetsen in Geany

Actie	Sneltoets
Nieuw bestand beginnen	Ctrl - N
Openen	Ctrl - O
Opslaan	Ctrl - S
Opslaan als	Ctrl-Shift - S
Sluiten	Ctrl - W
Herladen	Ctrl - R
Afdrukken	Ctrl - P
Ongedaan maken	Ctrl - Z
Herstellen	Ctrl - Y
Huidige regel(s) wissen	Ctrl - K
Huidige regel(s) dupliceren	Ctrl - D
Snippet vervolledigen	Tab
Regel(s) omhoog verplaatsen	Ctrl - Up
Regel(s) omlaag verplaatsen	Ctrl - Down
Selecteren	Shift - pijltjes
Alles selecteren	Ctrl - A
Knippen	eerst selecteren, daarna Ctrl - X
Kopiëren	eerst selecteren, daarna Ctrl - C
Plakken	Ctrl - V
Help	F1
Inspringing vergroten	Ctrl - I
Inspringing verkleinen	Ctrl - U
Zoeken	Ctrl - F
Volgende zoeken	Ctrl - G
Vervangen	Ctrl - H
Kolommodus aan/uit	Alt - Shift ingedrukt houden en pijltjes
Naar regel nr.. gaan	Ctrl - L
Naar bijbehorende haakje gaan	Ctrl - B
Naar begin van de regel gaan	Home
Naar einde van de regel gaan	End
Volledig scherm	F11
Inzoomen	Ctrl - +
Uitzoomen	Ctrl - -
Zoom resetten	Ctrl - 0
Commentariëren	Ctrl - M
Ontcommentariëren	Ctrl-Shift - M
Compileren en uitvoeren	F9

6 De eerste stappen

6.1 Het eerste programma

Heel eigenaardig, maar als mensen een nieuwe programmeertaal leren, willen ze als eerste programma de computer 'iets' laten afdrukken op het scherm, bijvoorbeeld "Hallo". In Asymptote bestaat dit programma uit slechts één regel:

```
//zomaarhallo.asy
write("Hallo");
```

Berichtenvenster

```
Hallo
```

Veel uitleg behoeft dit programmaatje niet. Het commando `write(...)` – let op de ronde haakjes – 'schrijft' een stukje tekst `"Hallo"` – let op de aanhalingstekens – letterlijk in het Output Window. In dit geval wordt er een *string* afgedrukt. Een string is een aaneenschakeling van letters, cijfers en/of leestekens en moet altijd omsloten worden door aanhalingstekens (`"..."`).

3 Maak je klaar voor jouw eerste programma.

1. Open Geany.
2. Open een nieuw bestand: Ctrl - N
3. Sla het programma op onder de volgende naam (de extensie `.asy` ook typen): Ctrl - S

   `zomaarhallo.asy`

4. Typ de programmatekst in van het bovenstaande voorbeeld.
5. Vanaf nu herkent Geany de syntaxis van Asymptote: Het sleutelwoord `write` wordt in het bruin weergegeven, en het linkerhaakje en rechterhaakje horen bij elkaar. Ga maar eens met de cursor op het linker- of rechterhaakje staan: de bij elkaar horende haakjes worden gemarkeerd met een onderlijningstekentje:

   ```
   write ("Hallo");
   ```

   Als je op een haakje staat, kun je direct naar het bijbehorende haakje gaan met behulp van de sneltoets Ctrl - B.
6. Voer het programma uit en bekijk het berichtenvenster, onderaan in Geany: F9

   Ofwel verschijnt daar een foutmelding (of meerdere foutmeldingen):

   Berichtenvenster

   ```
   asy -V -f pdf -tex=pdflatex zomaarhallo.asy (in map: ...)
   zomaarhallo.asy: 2.15: syntax error
   error: could not load module 'zomaarhallo.asy'
   Compilatie mislukt.
   ```

   Ofwel verschijnt de tekst 'Hallo' en het volgende bericht:

   Berichtenvenster

   ```
   asy -V -f pdf -tex=pdflatex zomaarhallo.asy (in map: ...)
   Hallo
   Compilatie met succes beëindigd.
   ```

Als je geen enkele fout getypt hebt, behoor je tot de zelfzame soort mensen die bij hun eerste programma geen enkele fout maken. Proficiat!

Alhoewel ons eerste programmaatje slechts één regel telt, kunnen er al vele dingen fout gaan. Een typische foutmelding is de volgende:

Berichtenvenster

```
asy -V -f pdf -tex=pdflatex zomaarhallo.asy (in map: ...)
zomaarhallo.asy: 2.1: no matching variable 'Write'
Compilatie mislukt.
```

Je krijgt deze fout als je het woord `write` met een hoofdletter typte. De foutmelding moet je als volgt lezen:

- `zomaarhallo.asy` is de naam van het programma.
- De aanduiding `2.1` betekent dat Asymptote een fout vond in de tweede regel (**2**.1), meer bepaald in kolom 1 (2.**1**).
- De foutboodschap is de tekst
  `no matching variable ‘Write’`
  en betekent dat `Write` geen geldige naam is. Inderdaad, `write` moet met een *kleine* letter geschreven worden:

Asymptote is dus hoofdlettergevoelig. Let bijvoorbeeld op de schrijfwijze van `‘write’`, met een kleine letter dus.

`"Hallo"` moet tussen *haakjes* `(...)` staaan. Als je deze ronde haakjes vergeet, dan krijg je:

Berichtenvenster

```
zomaarhallo.asy: 2.14: syntax error
```

Hier geeft Asymptote aan dat er een fout is tegen de syntaxis. Asymptote ‘ontdekt’ de fout in regel 1 pas in kolom 14, alhoewel je de fout moet herstellen vlak voor `"Hallo"`. Als je meer ervaring krijgt, zul je beter leren omgaan met de foutmeldingen van de programmeertaal, en zul je vooral beter leren analyseren wat precies de oorzaak van een fout is.

`"Hallo"` is een string en moet tussen aanhalingstekens staan. Als je de aanhalingstekens vergeet, dan krijg je:

Berichtenvenster

```
zomaarhallo.asy: 2.7: no matching variable 'Hallo'
```

Als je geen aanhalingstekens rond een woord gebruikt, veronderstelt Asymptote dat het woord een variabele aanduidt. De fout is hier natuurlijk niet dat de variabele `Hallo` nog niet gedefinieerd is, maar dat je een string *bedoelde*, en daarbij de aanhalingstekens vergat. De foutboodschap vermeldt kolom 7 als plaats van de fout, precies op de letter `H` van `Hallo`. Dit geeft een aanwijzing in de richting waarin je de fout moet zoeken. Het is deze speurzin die je moet ontwikkelen om goed te kunnen ‘debuggen’.

Als je wél het openingsaanhalingsteken zet, maar het sluitaanhalingsteken vergeet, krijg je:

Berichtenvenster

```
zomaarhallo.asy: 2.1: string not terminated
```

Deze foutboodschap spreekt voor zich: je bent een string begonnen (door de aanhalingstekens te openen), maar je hebt de string niet afgemaakt (aanhalingsteken niet gesloten).

Elk commando moet eindigen op een puntkomme (**;**), anders krijg je deze foutmelding:

Berichtenvenster

```
zomaarhallo.asy: 2.12: syntax error
```

De fout treedt op in kolom 12. Dit is op het einde van de regel, precies daar waar de puntkomma verwacht wordt.

Typ je bijvoorbeeld `wriite` in plaats van `write`, dan krijg je de foutmelding

Berichtenvenster

```
zomaarhallo.asy: 2.1: no matching variable `wriite'
```

In een enkele programmaregel kun je meerdere fouten maken. In het volgende voorbeeld zijn minstens vier fouten gemaakt (hooflettergevoeligheid, linkerhaakje vergeten, aanhalingsteken (rechts) vergeten, puntkomma vergeten):

`write ("Hallo");`	~~`Write "Hallo)`~~
Goed	Fout

6.2 Debugging

Programmeren is een complex proces, door mensen uitgevoerd. Dus af en toe treden er fouten op. Een fout in een computerprogramma heet een *bug* en het methodisch opsporen van fouten heet *debugging*.[1]

Er zijn verschillende soorten van fouten die kunnen optreden. Als je goed het onderscheid kent, kun je gemakkelijker debuggen.

6.2.1 Syntaxis-fouten

Syntaxis omvat de structuur van het programma en de spellingsregels. In het Nederlands bijvoorbeeld, moet elke zin beginnen met een hoofdletter en eindigen op een punt. deze zin bevat dus een syntaxis-fout (begint niet met een hoofdletter). En deze ook (eindigt niet op een punt) Voor de meeste menselijke lezers zijn een paar syntaxisfouten niet erg, zeker niet als het poëzie betreft, omdat de meeste Ndrlndstlg zinnen nog leesbaar zijn, zelfs al laten we enkele klinkers weg.

Programmeertalen zijn niet zo vergevingsgezind. Als er ook maar een enkel leesteken of lettertje verkeerd geschreven is, geeft de computer een foutmelding, zodat de programma-uitvoering stopt, of nog erger: zelfs niet eens start.

Jammer genoeg zijn de meeste foutmeldingen niet erg duidelijk voor beginnende programmeurs. Tijdens de eerste stappen in je programmeercarriére steek je heel wat tijd in het opsporen van syntaxisfouten. Als je meer ervaring krijgt, zul je nog altijd fouten maken, maar ze beter begrijpen en ze vlugger opsporen.

[1] Een *bug* is Engels voor 'ongedierte'.

6.2.2 Runtime errors

Deze fouten treden pas op als het programma aan het lopen is, en als er bewerkingen voorkomen die op dat ogenblik niet gedefineerd zijn, bijvoorbeeld een deling door 0. Dergelijke fouten worden ook *exceptions* genoemd, omdat ze meestal aangeven dat er iets uitzonderlijk (en fout) gebeurd is.

6.2.3 Semantische fouten

Stel je hebt alle syntaxisfouten uit het programma gehaald, en er treden ook geen runtime errors op. Het programma draait probleemloos zonder fouten en zonder onderbreking. De computer geeft dus geen foutmeldingen. Asymptote doet exact wat je *gezegd* hebt wat het moet doen. Toch kan het gebeuren dat Asymptote toch niet doet wat je *verwacht* dat het zou moeten doen. Het programma bevat dan een *semantische fout*.

Het probleem is, dat het programma dat je geschreven hebt niet het programma is dat je *wilde* schrijven. De betekenis (semantiek) van het programma is dus fout. Semantische fouten opsporen kan lastig zijn. Je moet dan achterwaarts redeneren: kijk naar de output van het programma en probeer te ontcijferen wat het programma precies doet.

6.2.4 Debugging, debugging, debugging

Een van de belangrijkste vaardigheden die je moet verwerven als programmeur is debugging.

Je kunt debugging tot op zekere hoogte vergelijken met het werk van een detective. Uit de (soms povere) aanwijzingen die je krijgt, moet je afleiden wat de mogelijke oorzaak van de fout is.

Debuggen heeft ook iets weg van een wetenschappelijk experiment uitvoeren. Als je een idee hebt van wat er fout gaat, breng je een kleine wijziging in het programma aan, en je probeert opnieuw. Als je hypothese correct is, kun je het resultaat van de wijziging voorspellen. Je bent dan een stap dichter bij een werkend programma. Als je hypothese fout blijkt, moet je met een nieuw idee afkomen.

Zoals Sherlock Holmes zei: “when you have eliminated the impossible, whatever remains, however improbable, must be the truth” (uit A. Conan Doyle’s *The Sign of Four*).

Debugging

Alhoewel het behoorlijk frustrerend kan zijn, is debugging een van de intellectueel rijkste, uitdagendste en interessantste delen van het programmeren.

Voor sommige mensen is programmeren en debugging hetzelfde. Dat wil zeggen: programmeren is het proces van geleidelijke debugging totdat het programma doet wat je wilt. De idee is om altijd te vertrekken van een werkend programma dat *iets* doet, en daarna stapsgewijs kleine wijzigingen aan te brengen, ondertussen debuggend, zodat je altijd een werkend en foutloos programma hebt.

Linux bijvoorbeeld, is een besturingssysteem dat bestaat uit duizenden regels programmacode, maar startte als een eenvoudig programma dat Linus Torval gebruikte om de Intel 80386 chip te onderzoeken. Volgens Larry Greenfield: “Linux began with a project to explore the 386 chip. One of Linus’s earlier projects was a program that would switch between printing AAAA and BBBB. This later evolved into Linux. ”
(uit: *The Linux Users’ Guide Beta Version*).

6.2.5 Syntaxis

Mensen spreken natuurlijke talen, zoals Nederlands en Frans. Natuurlijke talen zijn niet ontworpen door mensen, ze ontstonden spontaan. *Formele talen* daarentegen zijn door mensen ontworpen voor specifieke doeleinden. Wiskundigen bijvoorbeeld gebruiken een formele taal die erg geschikt is voor het noteren van relaties tussen getallen en het vastleggen van patronen. Programmeertalen zijn ook formele talen die ontworpen zijn om berekeningen uit te drukken.

Zoals reeds eerder aangestipt, gebruiken formele talen strikte syntaxisregels. In de wiskunde is 'x=5+-3' syntactisch fout, omdat er geen twee bewerkingstekens vlak na elkaar kunnen staan.

Om het even of je een zin leest in de Nederlandse taal, of in een formele taal, je moet uitvissen wat de structuur van de zin is (bij een natuurlijke taal doe je dit onbewust). Dit proces heet *parsing* (zinsontleding). Formele talen zijn kernachtiger dan een natuurlijke taal, zodat er meer tijd nodig is om ze te lezen. De structuur is bovendien zeer belangrijk. Het is meestal geen goed idee om een programma gewoon van voor naar achter te lezen. Leer een programma in je hoofd te ontleden. En onthoud vooral: de details zijn superbelangrijk. Kleine spelfoutjes en verkeerde leestekens kunnen een groot verschil maken bij programmeertalen.

6.2.6 Commentaar

Je kunt in de source code commentaar schrijven. Commentaar kan bijvoorbeeld nuttige informatie zijn zoals de naam van het programma, of de datum en de naam van auteur. Asymptote negeert dit commentaar. Het belangrijkste nut van commentaar in een programma is echter om bijkomende uitleg te geven over stukjes programmacode, zodat je zelf(!) en andere lezers je het programma beter kunt volgen. Soms vult een bijkomende figuur het commentaar nog beter aan. Zie figuur 10 op pagina 34.

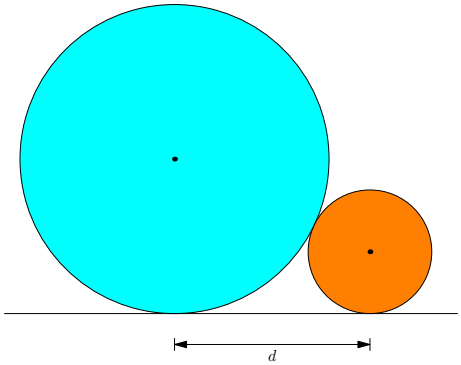
```
//sangakurakendecirkels.asy
size(10cm);

//De stralen van de cirkels zijn r1 en r2;
real r1=5;  //r1 hoeft niet de grootste te zijn.
real r2=2;
//Horizontale afstand tussen de middelpunten van beide cirkels
real d=2*sqrt(r1*r2);
write(d);
```

Commentaar begint met **//**. Asymptote negeert alle tekst vanaf // tot het einde van de regel. Je kunt commentaar op het einde van een regel schrijven, of je kunt commentaar op een aparte regel schrijven.

Je kunt in Geany blokken teskt omvormen tot commentaar met behulp van de sneltoets Ctrl - M, en terug ontcommentaren met behulp van de sneltoets Ctrl-Shift - M.

6.2.7 Debuggen met behulp van commentaar

Als je een hardnekkige fout in een programma hebt, die je maar niet kunt vinden, moet je steeds het volgende proberen: Maak commentaar van dat deel van het programma waarin je denkt dat de fout zit. Dit kan best een groot deel van het programma zijn.



Figuur 10 – Een figuur ter ondersteuning van commentaar in een programma

Ontcommentaar vervolgens beejte bij beetje programmaregels totdat de fout optreedt. Op dat ogenblik heb je precies de regel te pakken die de fout veroorzaakt.

Je hebt deze methode zeker al vaker toegepast bij het zoeken naar fouten in een LaTeX-brontekst.

4 Zie je de overeenkomsten en verschillen tussen commentaar in Asymptote en in LaTeX?

6.3 Opdrachten

If debugging is the process
of removing software bugs,
then programming must be the process
of putting them in.

— Edsger Wybe Dijkstra
(1930 - 2002 Nederland)

Foutboodschappen

Lees altijd de foutboodschappen in het Output Window!

Oefen in het ontcijferen van foutmeldingen en je wordt een echte programmeur.

5 Pas het programma `zomaarhallo.asy` een beetje aan:

- Voeg enkele regels toe aan het programma. Bijvoorbeeld, een commando dat de boodschap “Alles goed?” afdrukt.
- Voeg commentaar toe aan het programma.
- Test het programma opnieuw (F9). Het commentaar mag geen effect hebben.
- Voeg nog een `write`-statement toe.
- Test het programma opnieuw (F9).

Dit moet je steeds consequent doen: kleine wijzigingen aanbrengen en *onmiddellijk* testen (F9). Door telkens onmiddellijk eventuele fouten te herstellen, houd je op elk ogenblik het programma foutvrij.

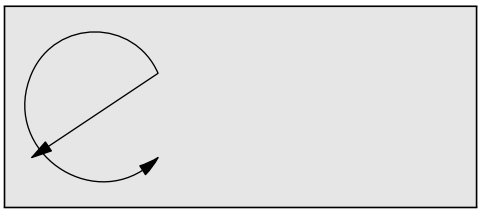
6 Het is een goed idee om eens, bij wijze van experiment, zoveel mogelijk fouten te maken als je maar kunt bedenken. Experimenteer er maar op los. Kijk telkens goed naar de foutboodschappen van Asymptote.

Soms vertelt Asymptote precies waar en wat je fout gedaan hebt. Het is in dat geval gemakkelijk om de fout te herstellen. Maar soms geeft Asymptote misleidende foutmeldingen. Je gaat een soort intuitie ontwikkelen wanneer je Asymptote kunt vertrouwen en wanneer je het zelf moet uitpluizen.

- Schrijf 'read' in plaats van 'write'.
- Doe een van beide haakjes weg. Of voeg een extra haakje toe.
- Laat een aanhalingsteken weg.
- Laat de puntkomma op het einde van een regel weg.
- Typ /, of % zoals in LaTeX, in plaats van // voor een commentaar.
- Laat het woordje 'write' weg.

6.4 Het tweede en derde programma

```
//zomaarlogo.asy
size(2cm);
picture fig;
pair LO=(0,0);
pair RB=(3,2);
pair LB=(0,2);
pair RO=(3,0);
path logo=LO--RB..LB..RO;
draw (fig, logo, Arrows);
add(fig);
```



Berichtenvenster

```
asy -V -f pdf -tex=pdflatex zomaarlogo
Compilatie met succes beëindigd.
```

Dit programma produceert een afbeelding `zomaarlogo.pdf`.

7 Maak je klaar voor jouw tweede programma.

1. Open een nieuw bestand in Geany: Ctrl - N
2. Bewaar het programma onder de volgende naam: Ctrl - S

   `zomaarlogo.asy` (de extensie `.asy` ook typen)

3. Typ de programmatekst in van het bovenstaande voorbeeld.
4. Typ nooit een programma in één keer in om het daarna pas voor de eerste keer te testen. De kans op een opeenstapeling van fouten is dan veel groter.

   Typ slechts enkele regels in, test direct met behulp van de sneltoets F9, ook als er niets gebeurt. Maar let wel op het berichtenvenster of er geen foutmeldingen verschijnen. Corrigeer onmiddellijk de fouten, nog voor je meer programmatekst typt.

   Typ daarna nog enkele regels programmacode, test opnieuw met F9, enzovoort. Zo leer je goed controle te krijgen over de foutmeldingen.
5. Van zodra de laatste letter van een sleutelwoord van Asymptote, zoals bijvoorbeeld `pair`, getypt is, wordt dit sleutelwoord in kleur weergegeven. Let daar op terwijl je typt: sleutelwoorden die niet in kleur verschijnen, bevatten een spelfout. Het veranderen in de bruine of blauwe kleur geeft aan dat het sleutelwoord correct getypt is.
6. Voer het programma uit en bekijk het berichtenvenster, onderaan in Geany: F9

7. Sluit de pdf-viewer Foxit Reader (Windows) of Evince (Ubuntu) om terug te keren naar Geany.
8. Verbeter eventuele fouten. Let daarbij aandachtig op de precieze plaats (regelnummer en kolomnummer) die Asymptote aangeeft in de foutmelding.

```
%test.tex
\documentclass[a4paper]{article}
\usepackage[dutch]{babel}
\usepackage{graphicx}
\begin{document}
Deze afbeelding
is gemaakt in Asymptote:

\includegraphics
  [width=1.5cm]
{zomaarlogo}
\end{document}
```

Deze afbeelding is gemaakt in Asymptote:

Je kunt de afbeelding gebruiken in andere programma's, bijvoorbeeld in LaTeX.

Naamgeving bestanden

Let op: zorg er voor dat de naam van de pdf-afbeeldingen *verschillend* is van de naam van de brontekst. Bijvoorbeeld, een brontekst `zomaar.tex` met een ingesloten afbeelding met dezelfde naam `zomaar.pdf` geeft problemen omdat LaTeX ook een pdf produceert, namelijk `zomaar.pdf`, zodat de oorspronkelijke afbeelding `zomaar.pdf` verloren gaat.

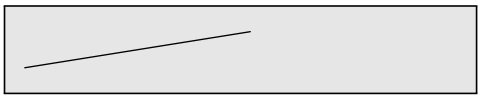
Als je de tekening bijvoorbeeld wilt gebruiken in een webpagina, zet dan de afbeelding `voorbeeldpijlen.pdf` om in een `png`-afbeelding met behulp van ImageMagick:

Ga naar de map waarin de afbeelding staat en geef het volgende commando aan de opdrachtprompt (Windows) of in de terminal (Ubuntu):

Terminal

```
convert zomaarlogo.pdf zomaarlogo.png
```

```
//zomaarhelling.asy
size(3cm);
picture fig;
real h1=40;
real h2=56;
real lengte=100;
pair A=(0,h1);
pair B=(lengte,h2);
path wegdek=A--B;
draw (fig, wegdek);
add(fig);

real rico=(h2-h1)/lengte;
real hoek=atan(rico)*180/pi;
real afgerond=round(hoek);
string s=(string)afgerond+"°";
write (s);
```



Berichtenvenster

```
9°
```

Dit programma produceert zowel een afbeelding als een resultaat in het Output Window.

8 Probeer dit programma ook eens uit.

Let er op om telkens Foxit Reader (Windows) of Evince (Ubuntu) te sluiten om terug te keren naar Geany.

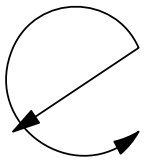
Typ slechts enkele regels per keer, test het voorlopige programma met de sneltoets F9, en verbeter *direct* elke fout.

Herhaal deze stapsgewijze werkwijze totdat het hele programm getypt is. Zorg er voor dat je te allen tijde een foutloos werkend programma hebt.

6.5 Leren programmeren

6.5.1 First, solve the problem

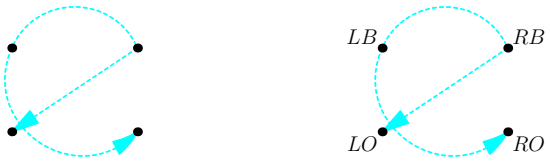
Asymptote produceert in de eerste plaats vectorafbeeldingen. Zie bijvoorbeeld de vectorafbeelding `zomaarlogo.pdf` in figuur 11.



Figuur 11 – De vectorafbelding `zomaarlogo.pdf`

Een afbeelding in Asymptote bestaat uit afzonderlijke basisvormen zoals punten, lijnen, rechthoeken, cirkels, ellipsen, en Bézierkrommen (gebaseerd op derdegraadsfuncties).

Het komt er dus op aan om een tekening die je voor ogen hebt, zo efficiënt mogelijk te ontleden in deze afzonderlijke componenten. Bijvoorbeeld, de afbeelding `zomaarlogo.pdf` is ‘opgehangen’ aan een viertal steunpunten. Zie figuur 12a. Geef deze steunpunten een

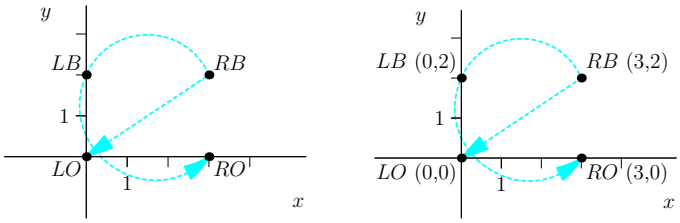


(a) Steunpunten zoeken in de tekening

(b) Een naam geven aan de steunpunten

Figuur 12 – Een vectorafbeelding ontleden in basisvormen.

naam. bijvoorbeeld `LO` (Links Onder), `RB` (Rechts Boven), `LB` (Links Boven) en `RO` (Rechts Onder). Deze namen mag je zelf kiezen, maar kies altijd namen die 'zelfuitleggend' zijn. Zie figuur 12b.



(a) Een zo eenvoudig mogelijk assenkruis kiezen

(b) De coördinaten van de steunpunten invullen

Figuur 13 – De positie van de steunpunten zo eenvoudig mogelijk beschrijven

Teken vervolgens een assenkruis naar keuze. Kies de oorsprong en de schaal op de assen zodanig dat de coördinaten van de steunpunten zo eenvoudig mogelijk zijn. Gebruik, indien mogelijk, gehele getallen voor de x-coördinaat en de y-coördinaat. Zie figuur 13a. Vul daarna de coördinaten van de vier steunpunten in. Zie figuur 13b.

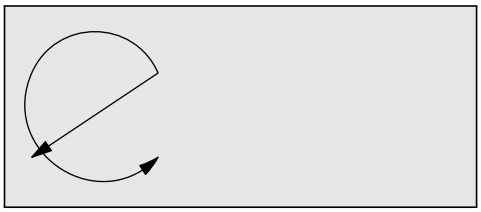
Vanaf nu is de opbouw van de figuur eenvoudig te beschrijven in eenvoudige stappen:

1. rechte lijn van `LO` naar `RB`;
2. gebogen lijn van `RB` via `LB` naar `RO`;
3. deze twee lijnen samenvoegen tot een doorlopend pad;
4. pijlpunten zetten aan de uiteinden van dit pad.

6.5.2 Then, write the code

Dankzij de zorgvuldige ontleding uit vorige sectie, is de omzetting naar programmatekst bijna rechttoe rechtaan:

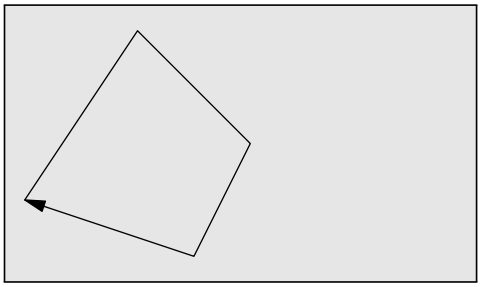
```
//zomaarlogo.asy
size(2cm);
picture fig;
pair LO=(0,0);
pair RB=(3,2);
pair LB=(0,2);
pair RO=(3,0);
path logo=LO--RB..LB..RO;
draw (fig, logo, Arrows);
add(fig);
```



Het `draw`-commando tekent lijnen. Het principe is: een reeks knooppunten opgeven, en deze verbinden met lijnstukjes. De operator `--` betekent: verbinden met een lijnstuk. De operator `..` betekent: verbinden met gebogen lijnen. Voor gebogen lijnen moet je uiteraard minstens drie opeenvolgende punten opgeven.

Een `path` is een lijst van knooppunten verbonden met `--` (rechte lijnstukjes) of met `..` (gebogen lijntjes).

```
//voorbeeldveelhoek;
size(3cm);
picture fig;
pair A=(0,0);
pair B=(2,3);
pair C=(4,1);
pair D=(3,-1);
path vierhoek=A--B--C--D--cycle;
draw (fig, vierhoek, Arrow);
add(fig);
```



Als je een *cyclisch* pad gebruikt, krijg je een gesloten figuur. Het laatste knooppunt in het pad moet dan `cycle` zijn.

```
...
filldraw (fig, vierhoek, green);
```

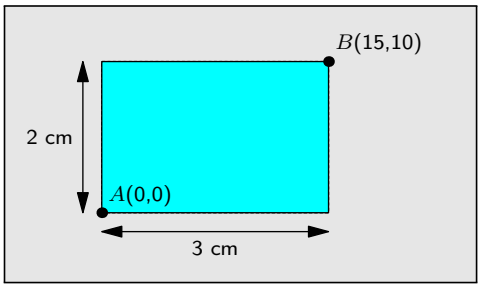
Een gesloten pad (cyclisch pad) kan opgevuld worden met een kleur. Gebruik dan het commando `filldraw` in plaats van `draw`.

6.6 Afmetingen

6.6.1 De grootte van de afbeelding

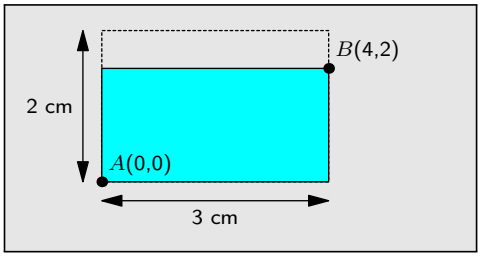
Het `size`-commando bepaalt de grootte van het 'canvas' waarop getekend wordt. Je kunt het canvas vergelijken met het doek waarop een kunstschilder zijn tekening neerzet. Asymptote vergroot of verkleint eventueel de afbeelding zodat de afbeelding het canvas zoveel mogelijk opvult. Als de verhoudingen van de afbeelding en van het canvas niet gelijk zijn, knipt Asymptote de onbenutte rechtermarge of bovenmarge weg.

```
size(3cm, 2cm);
picture fig;
pair A=(0,0);
pair B=(15,10);
path doos=box(A,B);
filldraw (fig, doos, cyan);
add(fig);
```



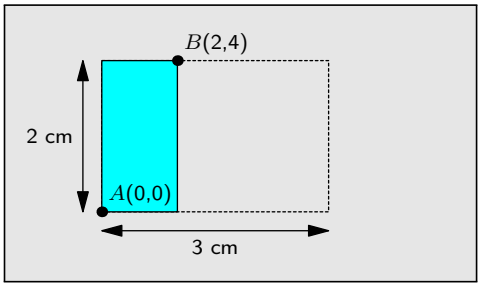
De verhouding van de rechthoek (15 : 10) en van het canvas (3 : 2) zijn gelijk. De afbeelding (blauwe rechthoek) wordt dus juist 3 cm breed en 2 cm hoog. Er worden geen marges weggeknipt.

```
size(3cm, 2cm);
picture fig;
pair A=(0,0);
pair B=(4,2);
path doos=box(A,B);
filldraw (fig, doos, cyan);
add(fig);
```



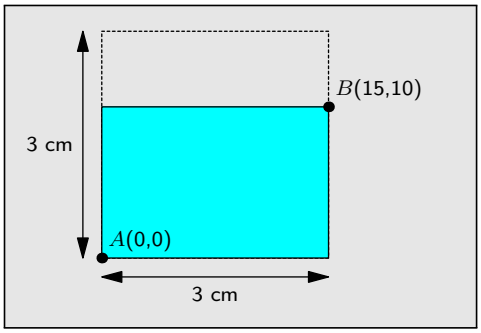
Het canvas heeft een verhouding 3 : 2, maar de afbeelding heeft een verhouding 4 : 2. De afbeelding wordt vergroot tot een rechthoek van 3 cm op 1,5 cm zodat het canvas maximaal gebruikt wordt. Uiteindelijk knipt Asymptote een smalle bovenmarge van 0,5 cm van het canvas weg.

```
size(3cm, 2cm);
picture fig;
pair A=(0,0);
pair B=(2,4);
path doos=box(A,B);
filldraw (fig, doos, cyan);
add(fig);
```



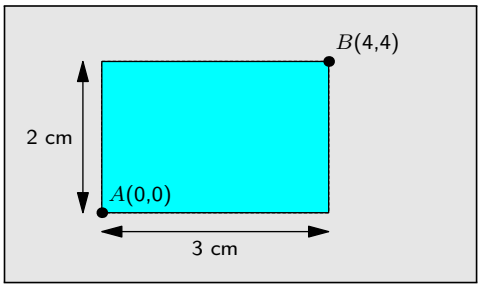
De afmeting wordt uiteindelijk herleid tot 1,5 cm op 2 cm. Asymptote knipt de overbodige rechtermarge van het canvas weg.

```
size(3cm);
//betekent hetzelfde als:
//size(3cm, 3cm);
picture fig;
pair A=(0,0);
pair B=(15,10);
path doos=box(A,B);
filldraw (fig, doos, cyan);
add(fig);
```



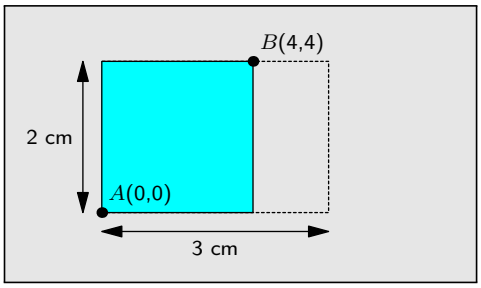
De afmeting wordt uiteindelijk herleid tot 3 cm op 2,5 cm. Asymptote knipt de overbodige bovenmarge van het canvas weg.

```
size(3cm, 2cm, IgnoreAspect);
picture fig;
pair A=(0,0);
pair B=(4,4);
path doos=box(A,B);
filldraw (fig, doos, cyan);
add(fig);
```



Asymptote negeert de aspectverhouding 4 : 4 van de afbeelding, en vervormt het vierkant, zodat de afbeelding precies past in een rechthoek van 3 cm op 2 cm,

```
size(3cm, 2cm, IgnoreAspect=false);
picture fig;
pair A=(0,0);
pair B=(4,4);
path doos=box(A,B);
filldraw (fig, doos, cyan);
add(fig);
```



Als je het argument `IgnoreAspect` weglaat of gelijkstelt aan `false`, wordt de aspectverhouding van het vierkant wel bewaard.

Dus, samengevat:

```
size(3cm, 2cm);
```

Het canvas is 3 cm op 2 cm.
De afbeelding wordt vergroot of verkleind totdat het canvas maximaal opgevuld is.
De afbeelding wordt dus 3 cm breed en/of 2 cm hoog, afhankelijk van de vorm van de afbeelding.
Asymptote knipt de eventuele overbodige marge weg.
De tekening behoudt zijn aspectverhouding.

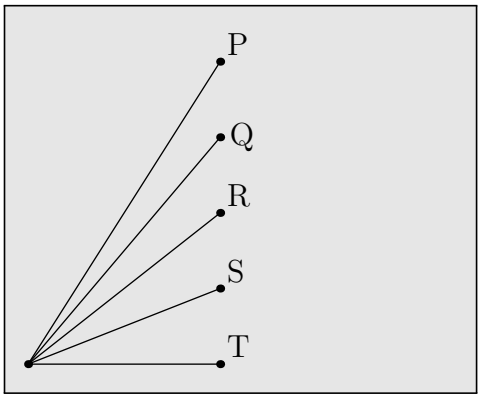
`size(3cm)` is een afkorting van `size(3cm, 3cm)`.

```
size(3cm, 2cm, IgnoreAspect);
```

De afbeelding wordt eventueel vervormd, zodanig dat de afbeelding precies past in een rechthoek van juist 3 cm breed *en* juist 2 cm hoog.

6.6.2 Lengte-eenheden

```
//voorbeeldmaateenheid.asy
//size(5cm, IgnoreAspect);
picture fig;
pair O=(0,0);
pair P=(1inch, 4cm);
pair Q=(2.54cm, 3cm);
pair R=(25.4mm, 2cm);
pair S=(72bp, 1cm);
pair T=(72, 0cm);
path p=O--P;
path q=O--Q;
path r=O--R;
path s=O--S;
path t=O--T;
draw (fig, p);
draw (fig, q);
draw (fig, r);
draw (fig, s);
draw (fig, t);
add(fig);
```



Je kunt lengte-eenheden opgeven in inch, cm, mm, bp (1 PostScript big point = 1 bp= 1/72 inch $\approx \frac{1}{3}$ mm). Als je geen eenheid gebruikt (bijvoorbeeld `pair T=(72, 0);`), gebruikt Asymptote automatisch bp als lengte-eenheid.

Je kunt desgewenst nog altijd het `size`-commando gebruiken om de tekening in een rechthoek met opgegeven afmetingen te dwingen.

Tip

Werk zoveel mogelijk met blote getallen (zonder maateenheid).

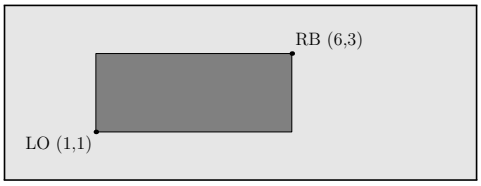
Stel de afmeting van de afbeelding in met behulp van het `size`-commando.

Gebruik lengte-eenheden alleen als dit echt nodig is.

6.7 Rechthoeken en ellipsen

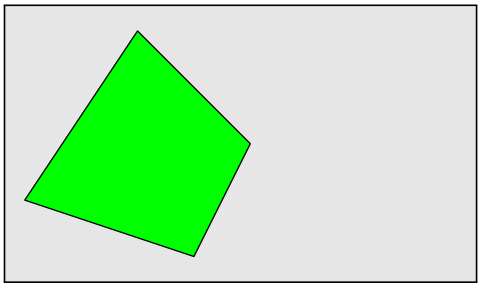
6.7.1 Rechthoek

```
//voorbeeldrechthoek;
size(2.5cm);
picture fig;
pair LO=(1,1);
pair RB=(6,3);
path doos=box(LO,RB);
filldraw (fig, doos, gray, black);
add(fig);
```



Defineer de punten van een diagonaal, bijvoorbeeld 'linksonder' en 'rechtsboven', en gebruik het box-commando om de rechthoek te definieëren.

```
//voorbeeldveelhoek;
size(3cm);
picture fig;
pair A=(0,0);
pair B=(2,3);
pair C=(4,1);
pair D=(3,-1);
path vierhoek=A--B--C--D--cycle;
filldraw (fig, vierhoek, green);
add(fig);
```



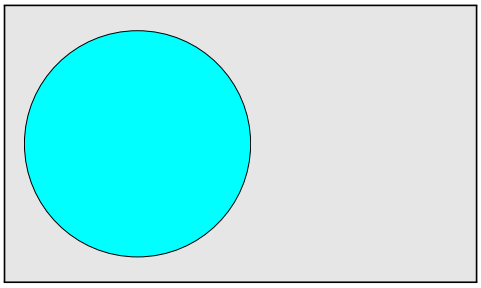
Je kunt gesloten paden *opvullen* met een bepaalde kleur. Gebruik dan `filldraw` in plaats van `draw`. Bij `filldraw` kun je twee kleuren opgeven: eerst de opvulkleur, daarna de randkleur. Als je maar één kleur opgeeeft, is dit automatisch de opvulkleur. De randkleur is dan standaard zwart.

Bij `draw` kun je uiteraard maar één kleur opgeven.

Enkele voorgedefinieerde kleuren zijn: `black`, `blue`, `red`, `gray`, `cyan` en `white`.[2]

6.7.2 Cirkel

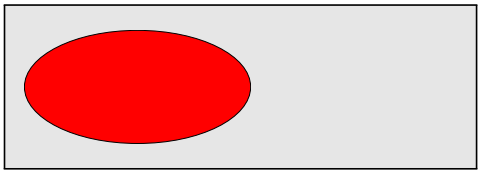
```
//voorbeeldcirkel.asy
size(3cm);
picture fig;
pair M=(5,3);
real straal=10;
path cirkel=circle(M, straal);
filldraw (fig, cirkel, cyan);
add(fig);
```



Je kunt een cirkel tekenen met behulp van het `circle`-path. Geef het middelpunt en de straal op (in die volgorde!).

6.7.3 Ellips

```
//voorbeeldellips.asy
size(3cm);
picture fig;
pair M=(5,3);
real a=10;
real b=5;
path ellips=ellipse(M, a, b);
filldraw (fig, ellips, red);
add(fig);
```



[2] Zie de handleiding van Asymptote voor meer kleuren. Zie sectie 6.9 op pagina 45.

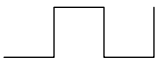
Je kunt een ellips tekenen met behulp van het `ellipse`-path. Geef het middelpunt, horizontale straal en verticale straal op (in die volgorde!).

6.8 Opdrachten

De bedoeling van deze opdrachten is om een geschikte maateenheid, een geschikt assenkruis en geschikte knooppunten te kiezen, zodat het ontwerp eenvoudig te beschrijven is.

9 Maak een nieuw programma `draakje.asy`.

Teken de volgende figuur. Elk van de zes lijnstukjes is 0,5 cm lang.

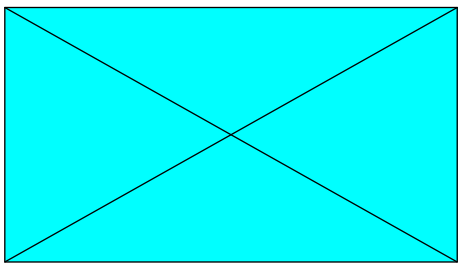


Zorg er voor dat je ook de volgende draak kunt tekenen (op ware grootte), door slechts één commando te wijzigen in het programma. (De dikte van de lijn is voorlopig nog niet belangrijk.)

Maak eerst een schets in potlood, voorzien van een gepast assenkruis. Duid hierop de schaal aan en voorzie enkele steunpunten van eenvoudig gekozen coördinaten.

10 Maak een nieuw programma `vlag.asy`.

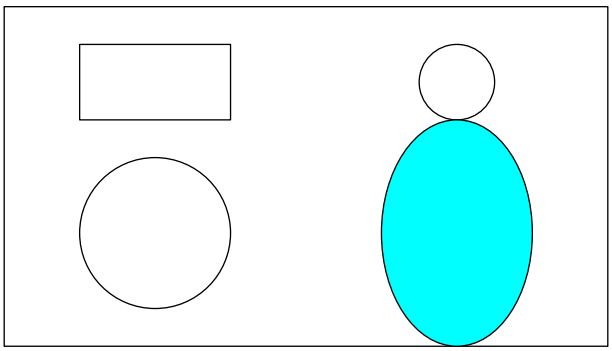
Teken de volgende figuur. De verhouding van lengte tot breedte is 16/10.



Maak eerst een schets in potlood, voorzien van een gepast assenkruis. Duid hierop de schaal aan en voorzie enkele steunpunten van eenvoudig gekozen coördinaten.

11 Maak een nieuw programma `fantasiecirkelsrechthoeken.asy`.

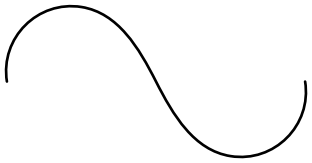
Teken de volgende figuur op ware grootte.



Maak eerst een schets in potlood, voorzien van een gepast assenkruis. Duid hierop de schaal aan en voorzie enkele steunpunten van eenvoudig gekozen coördinaten.

12 Maak een nieuw programma `krul.asy`.

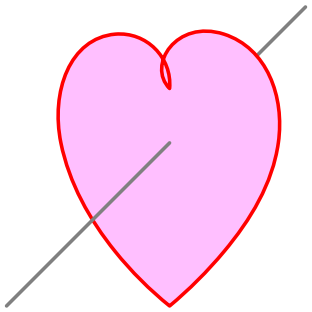
Teken de volgende figuur op ware grootte.



Maak eerst een schets in potlood, voorzien van een gepast assenkruis. Duid hierop de schaal aan en voorzie enkele steunpunten van eenvoudig gekozen coördinaten.

13 Maak een nieuw programma `hartmetpijl.asy`.

Teken de volgende figuur op ware grootte.



Maak eerst een schets in potlood, voorzien van een gepast assenkruis. Duid hierop de schaal aan en voorzie enkele steunpunten van eenvoudig gekozen coördinaten.

6.9 Handleiding Asymptote

De handleiding van Asymptote staat in de volgende map:

Windows:

`c:/Program Files/Asymptote/asymptote.pdf`

Ubuntu:

`/usr/share/doc/asymptote/asymptote.pdf`

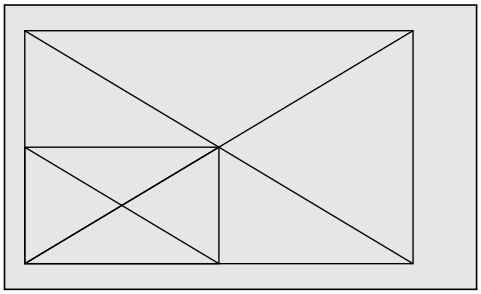
Deze handleiding (Hammerlindl, Bowman en Prince, *Asymptote: the Vector Graphics Language*) telt ongeveer 160 pagina's. Dit is zeker geen beginnerscursus om te leren programmeren, maar wel handig als naslagwerk.

7 Variabelen en types

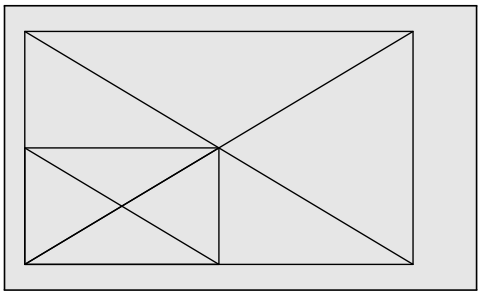
7.1 Variabelen maken programma's leesbaar

Vergelijk de volgende twee programma's met elkaar. Ze produceren beide dezelfde tekening, maar wat zijn dan de verschillen? Welk programma vind jij het meest 'leesbaar'?

```
//voorbeeldzondervariabelen.asy
size(5cm);
draw(box((0,0),(5,3)));
draw((0,0)--(5,3));
draw((0,3)--(5,0));
draw(box((0,0),(2.5,1.5)));
draw((0,0)--(2.5,1.5));
draw((0,1.5)--(2.5,0));
```



```
//voorbeeldmetmetvariabelen.asy
size(5cm);
real breedte=5;
real hoogte=3;
transform verkleining=scale(0.5);
pair LO, LB, RB, RO;
LO=(0,0);
LB=(0,hoogte);
RB=(breedte,hoogte);
RO=(breedte,0);
path rechthoek=box(LO,RB);
path diagonaal1=LO--RB;
path diagonaal2=LB--RO;
picture briefomslag;
draw(briefomslag,rechthoek);
draw(briefomslag,diagonaal1);
draw(briefomslag,diagonaal2);
add (briefomslag);
add (verkleining*briefomslag);
```



In het laatste programma zijn een aantal variabelen gebruikt, bijvoorbeeld `breedte`, `hoogte`, ..., `briefomslag`. Ik heb ze vetjes weergegeven in de broncode.

De namen van de variabelen zijn zodanig gekozen dat ze zichzelf verklaren: `briefomslag` voor de naam van de figuur die op een briefomslag lijkt, `breedte` voor de breedte van de briefomslag, `LO` om het punt LinksOnder aan te duiden, en `RB` staat dus voor RechtsBoven.

Het gebruik van variabelen maakt een programma iets langer, maar er zit een enorme winst in leesbaarheid en efficiëntie. Door het goed gebruik van variabelen is het programma ook veel gemakkelijker aanpasbaar. Stel je voor dat je een andere rechthoek wilt, bijvoorbeeld een rechthoek van 16 op 9.

In het eerste programma (`voorbeeldzondervariabelen.asy`) moet je welgeteld twaalf getallen wijzigen om dat voor elkaar te krijgen. De kans op fouten stijgt met het aantal wijzigingen. In het tweede programma (`voorbeeldmetvariabelen.asy`) moet je slechts twee getallen veranderen.

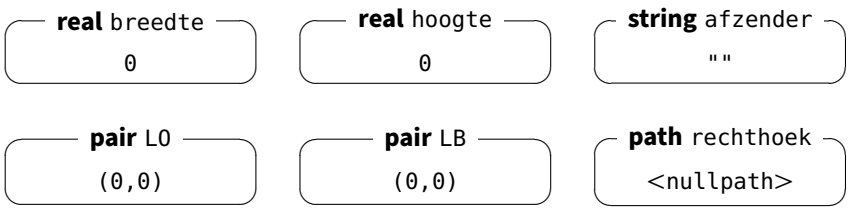
Leer vanaf het begin de gewoonte om met variabelen te werken. Het criterium om over te stappen van een constante naar een variabele is zeer eenvoudig: als hetzelfde getal twee keer of meer voorkomt in een zelfde programma, moet je overschakelen op een variabele. Zo eenvoudig is het. Leer van in het begin dit principe consequent toe te passen, en bespaar jezelf veel ergenis achteraf.

7.2 Variabelen declareren en initialiseren

Een variabele is een geheugenplaats, waarin een waarde kan worden opgeslagen. Een variabele heeft een naam, een type en een waarde. Elke variabele moet eerst *gedeclareerd* worden. Voorbeelden:

```
real breedte, hoogte;
string afzender;
pair LO, LB, RB, RO;
path rechthoek, diagonaal1, diagonaal2;
transform verkleining;
picture briefomslag;
pair positie;
```

Van zodra een variabele gedeclareerd is, is hij klaar om een waarde te ontvangen. De variabelen `breedte`, `hoogte`, `afzender`, `LO`, `LB`, enzovoort, staan te wachten om een waarde te ontvangen. Eigenlijk heeft Asymptote reeds stiekem een waarde toegekend aan elke gedeclareerde variabele. Variabelen van het type `int` en van het type `real` krijgen als beginwaarde automatisch het getal `0`. Strings worden geïnitialiseerd met de lege string `""`. Een pair vertrekt met de beginwaarde `(0,0)`. Een path is initieel gelijk aan het lege path `<nullpath>`.



Als een variabele gedeclareerd is, kun je er een waarde aan toekennen. Voorbeelden:

```
breedte=5;
hoogte=3;
afzender="Annelies Rutten";
LO=(0,0);
LB=(0,hoogte);
RB=(breedte,hoogte);
RO=(breedte,0);
rechthoek=box(LO,RB);
diagonaal1=LO--RB;
diagonaal2=LB--RO;
draw(briefomslag,rechthoek);
draw(briefomslag,diagonaal1);
draw(briefomslag,diagonaal2);
positie=(0.50*breedte, 0.85*hoogte);
label (pic=briefomslag, L=afzender, position=positie);
add(briefomslag);
```

Een waarde toekennen aan een variabele gebeurt met behulp van de operator **=**. Soms gebeurt de toekenning van een waarde op een speciale manier. De picture `briefomslag` krijgt op een impliciete manier een waarde toegekend door middel van het `draw`-commando. Door de rechthoek en de twee diagonalen te tekenen op de picture `briefomslag`, krijgt deze picture een waarde. Het `label`-commando doet iets gelijkaardigs: het voegt de string `afzender` toe aan de picture `briefomslag`.

De toestand van de variabelen `breedte`, `hoogte`, `afzender`, `LO`, `LB`, `rechthoek` en `briefomslag` is nu als volgt:

real `breedte`
`5`

real `hoogte`
`3`

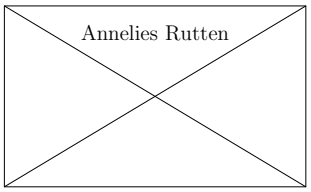
string `afzender`
`"Annelies Rutten"`

pair `LO`
`(5,0)`

pair `LB`
`(0,3)`

path `rechthoek`
`(0,0)-(0,3)-(5,3)-(5,0)-cycle`

picture `briefomslag`



Zoals uit de voorbeelden blijkt, kun je meerdere variabelen tegelijk declareren. De volgende programma's zijn gelijkwaardig:

```
real x;
real y;
real z;
```

```
real x, y, z;
```

Het is ook toegestaan om *tegelijk* een variabele te declareren en aan deze variabele een waarde toe te kennen. Dit heet *initialisatie*. De volgende programma's zijn gelijkwaardig:

```
string voornaam;
voornaam="Annelies";
```

```
string voornaam="Annelies";
```

Een variabele moet maar één keer geïnitialiseerd worden, maar de waarde kan vele keren veranderen binnen de loop van een programma. Voorbeeld:

```
int i=0;
write (i);
i=1;
write (i);
i=2;
write (i);
i=3;
write (i);
```

Output Window

```
0
1
2
3
```

Dergelijke programma's worden natuurlijk veel eleganter geschreven door middel van een herhalingsstructuur. Voorbeeld:

```
int n=3;
for (int i=0; i<=n; ++i){
  write(i);
}
```

Output Window

```
0
1
2
3
```

Let op het commando `++i`. Dit is een afkorting voor 'vermeerder de waarde van de variabele `i` met 1'. Het is aan deze notatie ++ dat de programmeertaal C++ (spreek uit: 'C plus plus') zijn naam te danken heeft. Vele andere programmeertalen, dus ook Asymptote, hebben deze syntaxis overgenomen.

De bewerking `--` bestaat ook. Voorbeeld:

```
int n=3;
for (int i=n; i>=0; --i){
  write(i);
}
```

Output Window

```
3
2
1
0
```

Let ook op de volgende veelgebruikte constructie:

```
int n=3;
n=n+1;
write (n);
```

Output Window

```
4
```

Het commando `n=n+1` betekent dat de variabele `n` een nieuwe waarde krijgt, namelijk de oude waarde van `n` vermeerderd met 1. Schematisch verloopt de waardeverandering van `n` als volgt:


```
int n=3;
++n;
write (n);
```

Output Window

```
4
```

Dit is een iets kortere schrijfwijze van het vorige programma.

14 Wat is de waarde van de variabele `n` en `k` op het *einde* van de programmaloop?

```
int n, k=3;
++k;
++k;
++n;
++k;
++k;
++k;
--k;
```

7.3 De naam van een variabele

```
string voornaam, telefoonnummer;
real uurloon, straal;
real x1, x2, y1, y2;
int aantal, nummer, n1, n2, aantal_zijvlakken;
```

Kies altijd een betekenisvolle naam voor de variabelen, bijvoorbeeld `voormaam` en `straal`. De underscore _ wordt soms gebruikt bij samengestelde variabelennamen, zoals bijvoorbeeld `aantal_zijvlakken`.

`Voornaam` en `voornaam` zijn dus twee verschillende variabelen. Gebruik zoveel mogelijk kleine letters voor namen van variabelen, en kies bij voorkeur ook geen speciale *sleutelwoorden* van de programmeertaal als namen voor variabeleen. Hier hebben we als Nederlandstaligen een voordeel: als je een Nederlandstalig woord kiest als naam van een variabele is het zeker geen sleutelwoord van Asymptote. Deze zijn immers allemaal Engelstalig, zoals `int`, `real`, `string`, `draw`, `picture`, enzovoort. Asymptote is hoofdlettergevoelig, ook voor variabelennamen.

Variabelenaam

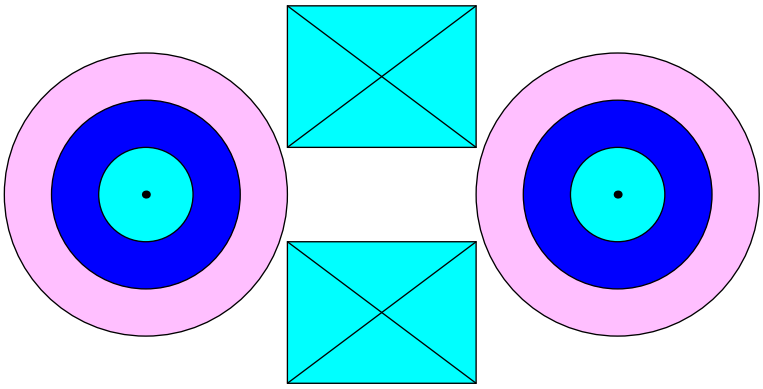
De naam van een variabele moet *beginnen* met een letter, en mag verder bestaan uit letters en/of cijfers en/of de underscore (_), maar *geen andere* leestekens.

Kies zichzelfverklarende namen voor variabelen.

7.4 Opdrachten

15 Maak een nieuw programma `schietrozen.asy`. Maak gebruik van de variabelen `pair M` (middelpunt), `real deltar` (verschil tussen twee stralen van de concentrische cirkels), `picture schietroos` (één schietroos), `picture vlag`, `real breedte` (breedte van de vlag), `real hoogte` (hoogte van de vlag), `real breedte`, `real l` (zie tekening)

Maak ook gebruik van de transformatie `shift(...)` met op de plaats van de ... een verschuivingsvector (een pair).



Verklarende tekening:

7.5 De types int, real, bool en string

7.5.1 Het type int

```
int aantal=5;
int k, l, m;
k=-4;
l=144;
m=10000;
write (aantal, k, l, m);
```

Output Window

```
5     -4    144    10000
```

Variabelen van het type `int` bevatten gehele getallen. Grote getallen zoals 10000 worden geschreven zonder scheidingsteken voor de duizendtallen. Dus niet: ~~10 000~~, maar wel: 10000.

7.5.2 Het type real

```
real x, y, z;
x=6.28;
y=3.5e4;
z=7e-3;
write(x,y,z);
real t=144e120;
write(t);
```

Output Window

```
6.28    35000   0.007
1.44e122
```

Variabelen van het type `real` bevatten kommagetallen of getallen in wetenschappelijke notatie.

7.5.3 Het type bool

```
write (5>=3);
write (5==3+2);
write (5!=3+2);
write (5>10);
```

Output Window

```
true
true
false
false
```

Let op het vergelijkingsteken voor 'groter of gelijk aan' **>=**. Het boolse gelijkheidsteken is speciaal: een *dubbel* gelijkheidsteken **==**.

```
int n=2, i=0;
bool voortdoen;

voortdoen=i<n;
write(voortdoen);
i=1;
voortdoen=i<n;
write(voortdoen);
i=2;
voortdoen=i<n;
write(voortdoen);
```

Output Window

```
true
true
false
```

```
real a, b, c, d;
a=2;
b=5;
c=1;
d=b^2-4*a*c;
if (d>=0)
 {
  write ("d is positief of nul");
  write(d);
 }
else
  write ("d is negatief");
```

Output Window

```
d is positief of nul
17
```

Variabelen van het type `bool` worden vooral gebruikt in voorwaarden.

7.5.4 Het type string

```
string voornaam, familienaam;
string naam;
voornaam="Annelies";
familienaam="Rutten";
naam=voornaam+" "+familienaam;
write(naam);
string code="007";
write (code);
```

Output Window

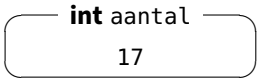
```
Annelies Rutten
007
```

Een `string` is een tekenreeks begrensd door aanhalingstekens. Met tekenreeksen kan niet gerekend worden, zelfs niet als de tekenreeks een getal voorstelt. Tekenreeksen kunnen worden samengevoegd met de operator **+** (concatenatie).

16 Wat is het verschil tussen `"15"` en `15`?
Zelfde vraag voor `"007"` en `7`.
Zelfde vraag voor `"15"+"007"` en `15+007`.

7.6 Casting

7.6.1 Enkele problemen



Je kunt een variabele het best vergelijken met een 'doos'. Deze doos heeft een etiket (de *naam* van de variabele) en een inhoud (de *waarde* van de variabele). De doos is van een bepaald *type*: in een doos van het type `int` passen alleen maar gehele getallen, en bijvoorbeeld niet een picture of een string.

```
1 //zomaar.asy
2 int aantal=17;
3 aantal=13.5;
```

Output Window

```
3.7: cannot convert 'real'
to 'int' in assignment
```

De variabele `aantal` is gedeclareerd als een variabele van het type `int`, en kan dus geen kommagetallen bevatten.

```
1 string jaar="2008";
2 real q=jaar/4;
```

Output Window

```
2.12: no matching function
'operator /(string,int)'
```

Een string blijft een string, ook als hij 'lijkt' op een getal. De string `2008` is slechts een aaneenschakeling van de karakters '2', '0', '0' en '8', en stelt dus een *woord* voor, maar *geen* getal. Je kunt een string niet delen door een getal.

```
1 int deeltal=24;
2 int deler=6;
3 int quotient=deeltal/deler;
```

Output Window

```
2.7: cannot convert 'real'
to 'int'
```

De uitkomst van een deling is altijd een real, ook bij een opgaande deling.

```
1 int a=2;
2 int b=-3;
3 real c=a^b;
4
5 real r=2^(-3);
```

Output Window

```
3.1: Only 1 and -1 can be
raise to negative exponents
as integers.
```

Negatieve exponenten van een machtsverheffing moeten reals zijn, tenzij bij machten van ± 1.

```
1 int a=3.0;
```

Output Window

```
1.7: cannot convert 'real'
to 'int'
```

Het getal `3.0` is een real omdat er een decimaal punt gebruikt is, en past dus niet in een int.

7.6.2 Automatische casting

```
1  int n=10;
2  real x=n;    //dit mag
3  write(x);
```

Output Window

```
10
```

Asymptote doet hier een automatische *casting*. De variabele `n` van het type int wordt in de variabele `x` van het type real 'gegoten'.

Het resultaat van de expressie `a*b`, die van het type `int` is, wordt ook automatisch in een real gegoten.

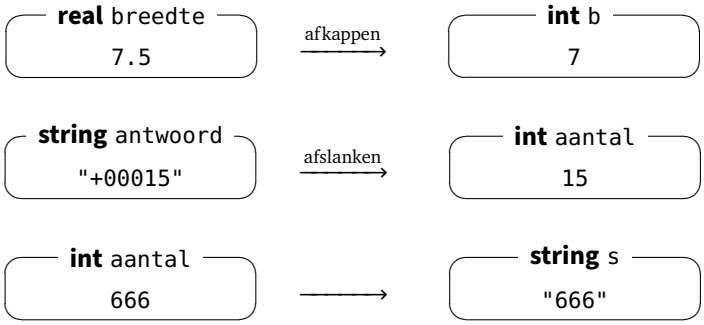
7.6.3 Geforceerde casting

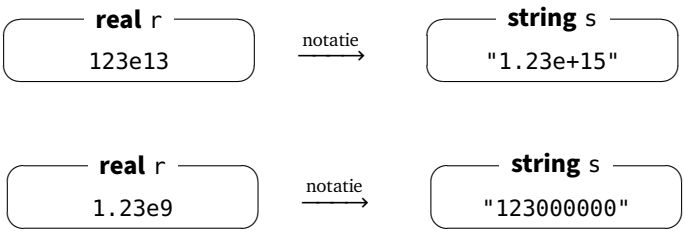
```
1  //zomaar.asy
2  real breedte=7.5;
3  int b=(int)breedte;
4  write(b);
5
6  string antwoord="015";
7  int aantal=(int)antwoord;
8  write(aantal);
9
10 int aantal=666;
11 string s=(string)aantal;
12 string tekst=s+" meter";
13 write (tekst);
14
15 real r=123e13;
16 string s=(string)r;
17 write(s);
18
19 real r=1.23e9;
20 string s=(string)r;
21 write(s);
```

Output Window

```
7
15
666 meter
1.23e+15
123000000
```

Door middel van de operatoren `(int)`, `(real)`, `(string)` kunnen ints, reals en strings onderling in elkaar gegoten worden. Daarbij ontstaat eventueel informatieverlies: cijfers na de komma gaan verloren, voorloopnullen worden weggelaten.





7.6.4 Format

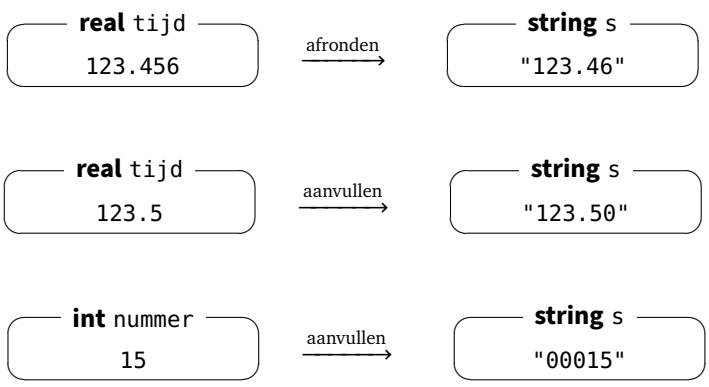
```
//zomaar.asy
real tijd=123.456;
string s=format("%.2f",tijd);
write (s);

real tijd=123.5;
string s=format("%#.2f",tijd);
write (s);

int nummer=15;
string s=format("%.5i",nummer);
write (s);
```

Output Window

```
123.46
123.50
00015
```

De functie `format(..., ...)` zet een real of een int om in een gewenste notatie. Het formaat van deze notatie wordt beschreven door een string `"%..."`, bijvoorbeeld `"%.2f"`: 'floating point' met maximaal twee cijfers na de komma. Het hekje # in `"%#.2f"` duidt aan dat er eventueel aangevuld wordt met nullen om twee decimalen te bereiken. De notatie `"%.5i"` werkt op een getal van het type int en zorgt er voor dat er in totaal minstens vijf cijfers staan, eventueel aangevuld met voorloopnullen.

Zie de documentatie (*C++ Reference – fprint*) van de overeenkomstige functie `fprint` van de programmeertaal C++ voor meer uitleg.



7.6.5 Afronden

```
real a=123.456;
int boven=ceil(a);
int beneden=floor(a);
int rond=round(a);
write (boven);
write (beneden);
write (rond);
```

Output Window

```
124
123
124
```

De functies `ceil(...)`, `floor(...)` en `round(...)` zetten een real om naar een int en ronden daarbij af naar boven, naar beneden, of naar het dichtst bijzijnde geheel. De uitkomst is steeds een int.

7.6.6 Absolute waarde

```
real a=-123.456;
real aa=abs(a);
int k=-14;
int kk=abs(k);
write (aa, kk);
```

Output Window

```
123.456    14
```

De functie `abs(...)`, zet een real of een int om naar zijn absolute waarde. De uitkomst is respectievelijk een real of een int.

7.6.7 Euclidische deling

```
int deeltal=23;
int deler=5;
int q, r;
q=quotient(deeltal,deler);
r=deeltal%deler;
write(q,r);

bool stelling=deeltal==deler*q+r;
write(stelling);
```

Output Window

```
4       3
true
```

De functie `quotient(...,...)` geeft als resultaat het quotiënt van de Euclidische deling, met andere woorden: het gehele deel van het quotiënt *naar beneden afgerond*. De operator % geeft de Euclidische rest van de deling.

Dit werkt ook voor negatieve getallen, zodat de volgende uitspraak steeds waar is voor alle ints `deeltal` en `deler`:

```
deeltal == deler*quotient(deeltal,deler) + deeltal%deler
```

17 Schrijf een programma `behangen.asy` om uit te rekenen hoeveel rollen behang van 10 m op 0.5 m je nodig hebt om een muur te behangen van 3 m op 6 m.

Gebruik in het programma de variabelen `lengtemuur`, `hoogtemuur`, `lengterol`, `breedterol`, `oppervlaktemuur`, `oppervlakterol` en `aantalrollen`.

7.7 Volgorde van de bewerkingen

```
//zomaar.asy
real a=2^3+1;
real b=2*2^3;
real c=3**4;
write(a,b,c);
```

```
Output Window
9       16      81
```

De machtsverheffing heeft de hoogste prioriteit. Om machten te berekenen, kun je kiezen uit de symbolen ^ en **.

```
real a=10-2*3;
real b=16/4*2;
write (a,b);
```

```
Output Window
4       8
```

Figuur 14 – De volgorde van de bewerkingen **/**, *****, **-** en **+**

De volgorde van de bewerkingen /, *, - en + is gemakkelijk te onthouden: op het numeriek toetsenbord staan ze in de juiste volgorde van prioriteit. Sie figuur 14.

```
real a=16/2/2;
real b=3^1^2;
real c=(1+2)*(3-4);
write (a,b,c);
```

```
Output Window
4       9       -3
```

Als twee bewerkingen gelijke prioriteit hebben, wordt de meest linkse bewerking eerst uitgevoerd.

Expressies binnen haakjes worden eerst geëvalueerd.

7.8 Het getal π

```
//pi.asy
write(pi);
```

```
Output Window
3.14159265358979
```

De variabele pi van het type real is standaard geïnitialiseerd in Asymptote op 15 beduidende cijfers.

7.9 Opdrachten

18 Maak een nieuw programma `tijdmiddernacht.asy` met als resultaat (als het op dit ogenblik 15:20:30 zou zijn):

Output Window
```
Er zijn sinds middernacht 55230 seconden verstreken.
Er resten nog 31170 seconden tot middernacht.
```

Gebruik de variabelen `int uur`, `int minuten`, `int seconden`, `int s1` (aantal seconden sinds middernacht), `int s2` (aantal seconden tot middernacht), `string sinds` en `string tot`.

De bedoeling van deze opdracht is om enkele rekenkundige bewerkingen te gebruiken, en om een tijdstip op te vatten als samengesteld uit afzonderlijke waarden.

19 Schrijf een nieuw programma `naargraden.asy` om 1 radiaal om te rekenen naar zestigdelige graden. Gebruik de voorgestelde variabelen.

```
//naargraden.asy
real rhoek=1;
real ghoek;
int graden, minuten, seconden;
string tekst1;
string tekst2;
//...
```

Output Window
```
1 rad = 57.2958°.
1 rad = 57°17'45".
```

Test het programma ook uit met andere hoeken, bijvoorbeeld:

`real rhoek=3.14;` → iets minder dan 180°,

`real rhoek=pi/2;` → 90°,

...

20 Schrijf een nieuw programma `afronden.asy`

```
//afronden.asy
real x=123.456789;
int n=3;
//...
```

Output Window
```
Het getal x afgerond
op n cijfers is: 123.457.
```

Test het programma ook uit met andere waarden voor `x` en `n`.

De bedoeling van deze oefening is om duidelijk te maken dat het uittesten van een ontwerp een belangrijke fase is bij problem-solving.

21 Schrijf een nieuw programma `heron.asy`

```
//heron.asy
real a,b,c;
a=3;
b=4;
c=5;
//...
```

Output Window

```
De oppervlakte van een
driehoek met zijden 3, 4 en 5
is: 6
```

Gebruik de formule van Heron:

$$s = \frac{a+b+c}{2}$$
$$Opp = \sqrt{s(s-a)(s-b)(s-c)} \qquad \text{(Heron)}$$

Gebruik de functie `sqrt(...)` van Asymptote om de vierkantswortel te berekenen uit een getal.

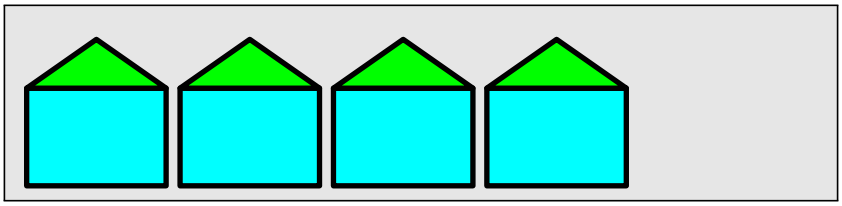
Test de functie uit met andere driehoeken.

De bedoeling van deze oefening is, dat je voldoende gevallen neemt als testgevallen, en dat je daarbij *eerst* manueel (met een rekenmachine, of uit het hoofd) de oppervlakte berekent als controle. Maak ook een schets van de driehoek die je als testgeval neemt. Wat gebeurt er als je de waarden 2, 4, 6 (of 7) invult voor `a`, `b` en `c`?

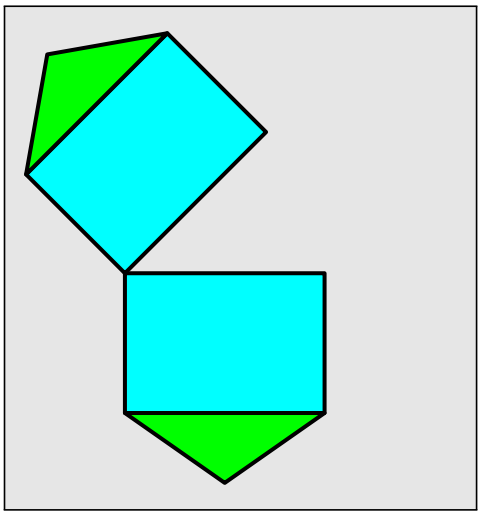
Moraal van het verhaal: testen, testen en nog eens testen...

7.10 De types pair, path, picture, pen en transform

```
//huizen.asy
size(8cm);
picture huis;
real b=5;
real h=3.5;
pair LO=(0,0);
pair LB=(0,h);
pair RB=(b, h);
pair RO=(b,0);
pair T=(b/2, 1.5*h);
path rechthoek=box(LO,RB);
path driehoek=LB--T--RB--cycle;
pen prand=black+2;
pen p1=cyan;
pen p2=green;
filldraw (huis, rechthoek, p1, prand);
filldraw (huis, driehoek, p2, prand);
transform v=shift(1.1*b,0);
add (huis);
add (v*huis);
add (v^2*huis);
add (v^3*huis);
```



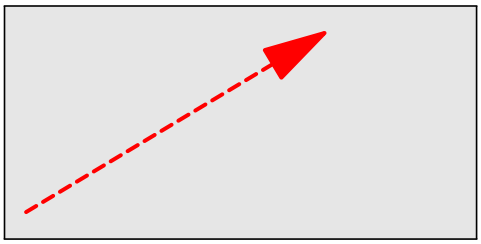
```
//huizen.asy
picture huis;
...

transform
  draai=rotate(45);
add(draai*huis);

transform
  spiegeling=reflect(LO,RO);
add(spiegeling*huis);
```



De transformatie `rotate(...)` roteert een object over een opgegeven hoek in graden. De draaizin is linksom (van de positieve x-as naar de positieve y-as).

De transformatie `reflect(..., ...)` spiegelt een object over de as bepaald door twee opgegeven punten.

```
//stippelpijl.asy
size(4cm);
pair begin=(0,0);
pair einde=(5,3);
path pijl=begin--einde;
pen pstippel
  =red
  +linetype("4 2 ")
  +linewidth(1.5);
draw (pijl, pstippel, Arrow);
```



Een pen bevat informatie over de *kleur*, *lijntype* en *lijndikte*.

Het lijntype wordt bepaald door een string, bijvoorbeeld `"4 2 "`. Dit geeft een stippellijn van 4 lijndiktes lang met onderbrekingen van 2 lijndiktes.

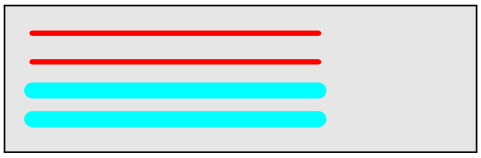
Asymptote voorziet in een afkorting om de lijndikte op te geven. In plaats van:

```
pen pstippel=red+linewidth(1.5);
```

mag je ook schrijven:

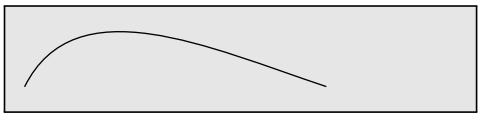
```
pen pstippel=red+1.5;
```

```
//voorbeeldcurrentpen.asy
size(4cm);
path lijn=(0,0)--(10,0);
transform v=shift(0,-1);
currentpen=red+2;
draw(lijn);      //lijn in red+2
draw(v*lijn);    //nog red+2
currentpen=cyan+6;
draw(v^2*lijn);  //nu cyan+6
draw(v^3*lijn);  //nog cyan+6
```



Als je lange tijd met dezelfde tekent, kun je de `currentpen` instellen.

```
//krommelijn.asy
size(4cm);
pair begin=(0,0);
pair einde=(5,0);
pair r1=(1,2);
pair r2=(3,-1);
path kromme=begin{r1}..{r2}einde;
draw (kromme);
```



Voor een path met kromme lijnen kun je bij elk knooppunt opgeven in welke richting het path moet vertrekken en aankomen. De richting `r1` is schuin naar rechtsboven (1 naar rechts, 2 naar boven). De richting `r2` is schuin naar rechtsbeneden (3 naar rechts, 1 naar onder).

```
//golflijn.asy
size(6cm);
```

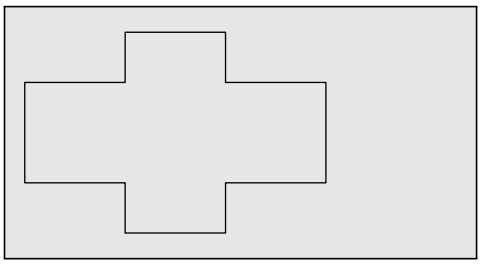
```
real x=2;
pair A=(0,0);
pair B=(x,0);
pair C=(2*x,0);
pair D=(3*x,0);
pair op=(1,1);
pair neer=(1,-1);
pen pgolf=blue+3;
path golf=A{op}..{neer}B..{op}C..{neer}D;
draw (golf, pgolf);
```

```
//kartellijn.asy
size(6cm);
real x=2;
pair A=(0,0);
pair B=(x,0);
pair C=(2*x,0);
pair D=(3*x,0);
real rico=3;
pair op=(1,rico);
pair neer=(1,-rico);
pen pgolf=blue+3;
path golf=A{op}..{neer}B{op}..{neer}C{op}..{neer}D;
draw (golf, pgolf);
```

7.11 Opdrachten

22 Maak een nieuw programma `kruis.asy`.

```
//kruis.asy
real breedte, hoogte, dikte;
pair M=(0,0);
real b=breedte/2;
real h=hoogte/2;
real d=dikte/2;
//...
```



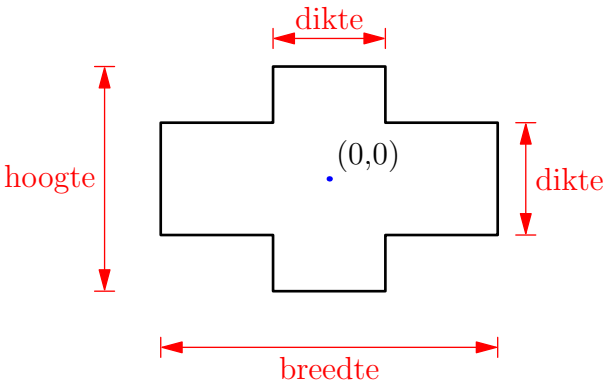
Het resultaat moet een kruis zijn. Initialiseer de variabelen: `breedte`, `hoogte` en `dikte` en laat alle andere knooppunten afhangen van deze afmetingen. Het middelpunt van het kruis moet de oorsprong $O(0,0)$ van het assenkruis zijn.

Hint: gebruik de hulpvariabelen `b`, `h`, `d` die respectievelijk de *halve* breedte, *halve* hoogte en *halve* dikte voorstellen.

Door *alleen* maar de waarde van `breedte`, `hoogte` en `dikte` te wijzigen, moet er een ander kruis ontstaan.

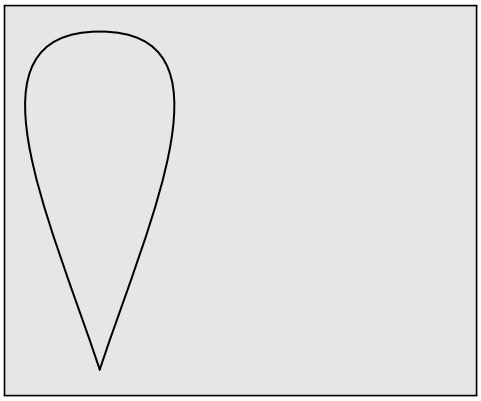
Test het programma dus ook uit met andere waarden voor deze drie afmetingen.

De betekenis van de drie afmetingen van het kruis is als volgt:



23 Maak een nieuw programma `amandel.asy`.

```
//amandel.asy
size(3cm);
pair A=(0,0);
pair B=(0,4);
real rico=3;
//pair r1=...;
//pair r2=...;
//path amandel=...
```

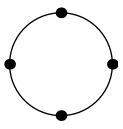


Gebruik de voorgestelde variabelen (en geen andere!).

24 Maak een nieuw programma `cirkel.asy`.

Teken een cirkel *zonder* gebruik te maken van het `circle`-pad.

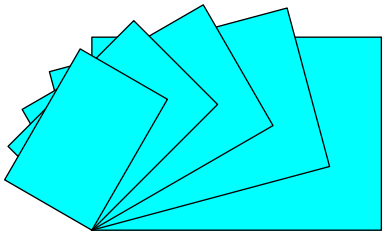
Hint: gebruik een kromme met vier knooppunten.



25 Maak een nieuw programma `waaier.asy`.

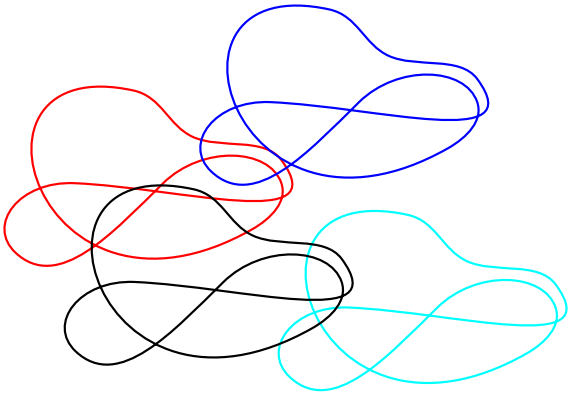
Teken de volgende figuur. De volgende rechthoeken zijn telkens een factor 0.85 kleiner.

Hint: gebruik twee transformaties en gebruik samenstellingen van deze twee transformaties.



26 Maak een nieuw programma `inspiratie.asy`.

Teken de volgende figuur in vier kleuren.



8 Iteratie en selectie

8.1 If

```
real x;
x=5;
if (x>0) write ("positief");
```

Output Window

```
positief
```

```
real x;
x=-17;
if (x>0) write ("positief");
```

Output Window

```
```

De if-structuur verplicht Asymptote om een keuze te maken. Naargelang aan de voorwaarde tussen haakjes voldaan wordt (het getal x al dan niet groter dan nul), wordt het bijbehorende write-commando uitgevoerd, of niet.

```
real x;
x=5;
if (x>=0)
  write ("positief of nul");
else
  write ("negatief");
```

Output Window

```
positief of nul
```

```
real x;
x=-17;
if (x>=0)
  write ("positief of nul");
else
  write ("negatief");
```

Output Window

```
negatief
```

Deze keuzestructuur bevat twee 'takken': de true-tak en de false-tak. Als de voorwaarde tussen haakjes `true` is, wordt de eerste tak uitgevoerd. In het andere geval wordt de tweede tak uitgevoerd.

```
real x;
x=-17;
if (x>=0){
  write ("positief");
  real w= sqrt(x);
  write (w);
  }
else
  write ("negatief");
```

Output Window

```
negatief
```

De takken van de if-structuur kunnen ook groepen van meerdere commando's bevatten. Een groep van commando's wordt bij elkaar gehouden door accolades.

Test dit programma uit met telkens een andere waarde voor de variabele x.

```
bool nogbezig=true;
if (nogbezig)
   write ("voortdoen");
else
   write ("stoppen");
```

Output Window

```
voortdoen
```

De logische uitdrukking tussen haakjes heet een *logische voorwaarde*. Een voorwaarde is van het type `bool`.[3] Een `bool` kan slechts een van de volgende twee waarden hebben: `true` of `false`.

Een logische voorwaarde kan een van de volgende vergelijkingsoperatoren bevatten:

```
x == y        //x gelijk aan y
x != y        //x niet gelijk aan y
x > y         //x groter dan y
x >= y        //x groter dan of gelijk aan y
x < y         //x kleiner dan y
x <= y        //x kleiner dan of gelijk aan y
```

Alhoewel deze symbolen je waarschijnlijk vertrouwd voorkomen, is de syntaxis een beetje verschillend van de wiskundige symbolen $=$, $\neq$, $\geq$. Let vooral op de logische gelijkheid: dubbel gelijkheidsteken $==$.

```
//voorbeeldnamiddag.asy
string tijd=time("%H:%M:%S");
write (tijd);
if (tijd > "12")
  write ("namiddag");
else
  write ("voormiddag");
```

Output Window

```
20:48:41
namiddag
```

De twee leden links en rechts van een vergelijkingsoperator moeten van *hetzelfde type* zijn. Je kunt alleen ints met ints, reals met reals, strings met strings,... vergelijken. Pairs kunnen worden vergeleken op (on)gelijkheid, maar niet op volgorde. Tussen pairs bestaat immers geen orde.

[3] Genoemd naar de Britse wiskundige Boole (1815-1864).

```
//voorbeeldabcformule.asy
real a=1;
real b=5;
real c=4;
real d=b^2-4*a*c;
if (d>0){
  write ("d is positief");
  real w=sqrt(d);
  real x1=(-b+w)/(2*a);
  real x2=(-b-w)/(2*a);
  write (x1, x2);
}
else
  if (d==0)
    write ("d is nul");
  else
    write ("d is negatief");
```

Output Window

```
d is positief
-1      -4
```

Deze selectiestructuur bevat meerdere geneste if-else-structuren. Op die manier kun je ook complexe selectiestructuren behandelen met meer dan twee keuzemogelijkheden.

```
//voorbeeldabcformule.asy
real a=1;
real b=5;
real c=13;
real d=b^2-4*a*c;
if (d>0){
  write ("d is positief");
  real w=sqrt(d);
  real x1=(-b+w)/(2*a);
  real x2=(-b-w)/(2*a);
  write (x1, x2);
}
else if (d==0)
  write ("d is nul");
else
  write ("d is negatief");
```

Output Window

```
d is negatief
```

Dit is hetzelfde programma met een net iets andere lay-out. Deze schikking benadrukt dat er eigenlijk drie afzonderlijke gevallen worden onderscheiden.

```
//voorbeeldweekdag.asy
int n=3;
string w;
if (n==1) w="maandag";
else if (n==2) w="dinsdag";
else if (n==3) w="woensdag";
else if (n==4) w="donderdag";
else if (n==5) w="vrijdag";
else if (n==6) w="zaterdag";
else if (n==7) w="zondag";
else w="";
write (w);
```

Output Window

```
woensdag
```

Deze lay-out benadrukt het feit dat er eigenlijk gekozen wordt uit acht mogelijkheden.

```
//voorbeeldweekdag.asy
int n=3;
string w;
if (n==1)
  w="maandag";
else
  if (n==2)
    w="dinsdag";
  else
    if (n==3)
      w="woensdag";
    else
      if (n==4)
        w="donderdag";
      else
        if (n==5)
          w="vrijdag";
        else
          if (n==6)
            w="zaterdag";
          else
            if (n==7)
              w="zondag";
write (w);
```

Output Window

```
woensdag
```

Dit is hetzelfde programma. De lay-out die hier gebruikt is, benadrukt het feit dat alle if's *genest* zijn. De ene if is een tak van de vorige else.

In voorwaardelijke uitvoering met meerdere takken, wordt er precies éénn tak uitgevoerd. Elke voorwaarde wordt in volgorde getest. Als de eerste voorwaarde `false` is, wordt de tweede getest, enzovoort. De eerstvolgende voorwaarde die `true` is, bepaalt welke tak uitgevoerd wordt, zelfs al komen er nog voorwaarden achter die ook `true` zijn.

8.1.1 Samengestelde voorwaarden

```
//voorbeeldinvolgorde.asy
real a=1, b=7, c=13;
bool involgorde;
if (a<=b && b<=c)
  involgorde=true;
else
  involgorde=false;
write (involgorde);
```

Output Window

```
true
```

```
real a=1, b=13, c=7;
bool involgorde;
if (a<=b && b<=c)
  involgorde=true;
else
  involgorde=false;
write (involgorde);
```

Output Window

```
false
```

De operator && is de logische en-operator. De samengestelde voorwaarde `(a<=b && b<=c)` is `true` als beide afzonderlijke voorwaarden `(a<=b)` en `(b<=c)` `true` zijn.

```
real a=1, b=13, c=7;
bool beetjeinvolgorde;
if (a<=b || b<=c)
  beetjeinvolgorde=true;
else
  beetjeinvolgorde=false;
write (involgorde);
```

Output Window

```
true
```

Er bestaat ook een logische of-operator: `||`. De samengestelde voorwaarde `(a<=b || b<=c)` is `true` als minstens een van beide afzonderlijke voorwaarden `(a<=b)` of `(b<=c)` `true` is.

```
//voorbeeldaltijdwaar.asy
real a=7;
real b=13;
write ( !(a==b) || a==b);
```

Output Window

```
true
```

De operator **!** is de logische niet-operator. De `!`-operator wordt vóór een `bool` geschreven, terwijl de operatoren && en || *tussen* twee logische uitdrukkingen worden gezet.

8.1.2 Deelbaarheid

```
//voorbeelddeelbaarheid.asy
int a, d;
a=75;
d=5;
if (a%d == 0){
  string s=(string)a
    + " is deelbaar door "
    + (string)d;
  write (s);
}
```

Output Window

```
75 is deelbaar door 5
```

Als een getal a deelbaar is door bijvoorbeeld 5, moet de Euclidische rest van de deling van a door 5 gelijk zijn aan 0.

8.2 While

8.2.1 Iteratie

Een van de redenen waarom we programma's schrijven, is om taken te automatiseren die moeten worden *herhaald*. Het herhaald foutloos uitvoeren van gelijkaardige taken, is iets wat computers heel goed (en snel) kunnen, in tegenstelling tot mensen, die traag zijn en veel fouten maken. Mensen bedenken liever patronen en laten de uitvoering over aan machines. Het herhaald uitvoeren van een reeks commando's, heet *iteratie*.

```
//voorbeeldaftellen.asy
int n=3;
while (n>0){
  write(n);
  n=n-1;
}
write("Start");
```

Output Window

```
3
2
1
Start
```

Het `while`-commando is een van de manieren om in Asymptote een iteratie te schrijven. Je kunt een `while`-structuur bijna 'lezen' alsof het een stukje in gewone taal betreft: "Zolang `n` groter is dan nul, ga door met het schrijven van de waarde van `n`, en verminder de waarde van `n` telkens met 1.

Meer formeel uitgedrukt, is de loop van een `while`-structuur als volgt:

1. Evalueer de voorwaarde die tussen haakjes staat, welke `true` of `false` teruggeeft.
2. Als de voorwaarde `false` is, verlaat dan de `while`-structuur en ga verder met de rest van het programma.
3. Als de voorwaarde `true` is, voer dan de commando's uit tussen de accolades **{**...**}**, en herbegin dan terug bij stap 1.

Deze soort van programmaloop heet een *lus*, omdat de loop van het programma terugkeert naar de voorwaarde. De commando's binnen de lus worden soms de *body* van de lus genoemd.

```
//voorbeeldaftellen.asy
int n=0;
while (n>0){
  write(n);
  n=n-1;
}
write("Start");
```

Output Window

```
Start
```

Merk op: als de voorwaarde reeds `false` is bij de eerst keer dat de lus betreden wordt, zullen de commando's binnen de lus *nooit* worden uitgevoerd.

8.2.2 Oneindige lus

```
//voorbeeldaftellen.asy
int n=3;
while (n>0){
  write(n);
  n=n+1;  //bedoel je: n=n-1??
}
write("Start");
```

Output Window

```
4
5
6
7
8
...  (programma stopt niet)
```

Anderzijds, als de while-voorwaarde steeds waar blijft, zullen de commando's van de body steeds opnieuw worden uitgevoerd, tot in het oneindige. We spreken dan van een *oneindige lus*.

Oneindige lussen komen nogal eens voor als gevolg van een bug in een programma. In bovenstaand voorbeeld bedoelden we waarschijnlijk `n=n-1`. Als een programma in een oneindige lus terecht komt, moet je hardwarematig ingrijpen om de computer tot stilstand te brengen. Hiervoor dient de 'noodrem' op het toetsenbord:

Oneindige lus onderbreken

- In Crimson Editor:
  → rechts klikken in het Output Window
  (even geduld: reactie komt met vertraging)
  → `Kill Process`
  (even geduld: reactie komt met vertraging)
- Aan de opdrachtprompt:
  → Ctrl-C of Ctrl-Break

Om oneindige lussen te voorkomen, is het dus noodzakelijk dat de body van de lus de waarde van een of meerdere variabelen wijzigt, zodat de voorwaarde ooit eens `false` wordt en de lus eindigt.

```
//voorbeeldaftellen.asy
int n=3;
while (n>0){
  write(n);
  --n;
}
write("Start");
```

Output Window

```
3
2
1
Start
```

In het voorbeeld van `aftellen`, zijn we zeker dat de lus altijd stopt, wat ook de waarde is van `n`. Want de waarde van `n` wordt steeds kleiner (met stapjes van 1) en zal dus ooit nul of negatief worden. In andere gevallen is het niet zo gemakkelijk om te zeggen dat de lus zal stoppen. n Merk op dat `--n` een afkorting is voor `n=n-1`, en dat `++` een afkorting is voor `n=n+1`.

8.2.3 Het Collatz-probleem

```
//voorbeelddriekaapluseen.asy
int k=13;
write(k);
while (k!=1){
  if (k%2 == 0)
    k=(int)(k/2);
  else
    k=3*k+1;
  write(k);
}
```

Output Window

```
13
40
20
10
5
16
8
4
2
1
```

In de body van de lus wordt de waarde van `k` voortdurend gewijzigd. De waarde van `k` wordt soms kleiner (als `k` even is), soms groter (als `k` oneven is), en het is bijgevolg niet vanzelfsprekend dat `k` ooit de waarde 1 bereikt, wat nodig is om de lus te beëindigen.

Als de beginwaarde van `k` gelijk is aan 13, krijgen we een eindige lus, maar voor `k=-1` zal het programma niet uit zichzelf stoppen: we krijgen een oneindige lus. Voor sommige waarden van `k` kunnen we aantonen dat de rij stopt. Als we bijvoorbeeld starten met een macht van twee, komen we uiteindelijk — door voortdurende halvering — terecht bij 1. De interessante vraag is, of de lus eindigt voor *elke positieve* startwaarde van `k`. Tot op de dag van vandaag is er nog niemand in geslaagd om dit te bewijzen *of* een tegenvoorbeeld te geven. Dit is het beroemde *Collatz-probleem*.

8.2.4 Fibonacci

```
//voorbeeldfibonacci.asy
int a, b, n, nieuw;
a=1;
b=1;
n=10;
int i=2;
while (i<n){
  i=i+1;
  nieuw=a+b;
  write(i, nieuw);
  a=b;
  b=nieuw;
}
```

Output Window

```
3    2
4    3
5    5
6    8
7    13
8    21
9    35
10   55
```

De variabele `i` werkt als een soort teller. Telkens wordt de som berekend van laatste twee getallen `a` en `b` uit de rij. Dit pas berekende getal wordt eventjes opgeslagen in de variabele `nieuw`, en de waarde van de variabelen `b` en `nieuw` worden gekopieerd naar `a` en `b`. Op die manier zijn `a` en `b` opnieuw de laatste twee getallen van de rij.

8.3 For

```
//voorbeeldaftellen.asy
for (int n=3; n>0; --n){
  write(n);
}
write("Start");
```

```
//voorbeeldaftellen.asy
int n=3;
while (n>0){
  write(n);
  --n;
}
write("Start");
```

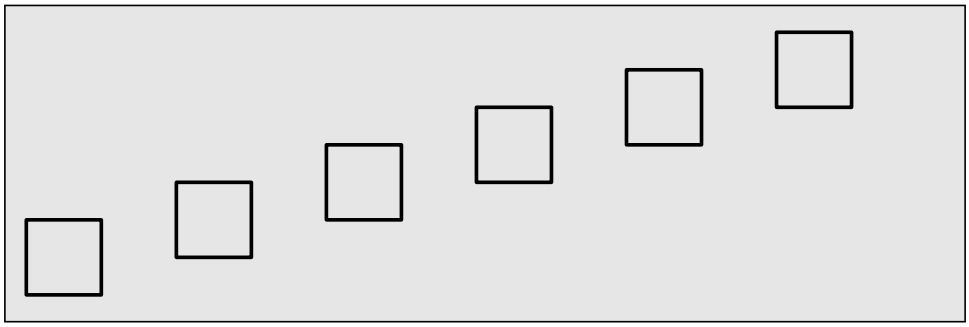
Output Window

```
3
2
1
Start
```

Sommige while-structuren kunnen worden vervangen door een for-structuur. De initialisatie, de lusvoorwaarde en de ophoging van de lusvariabele `n` worden bij elkaar gezet, wat het programma beter leesbaar maakt voor meer ‘ervaren’ programmeurs. Wellicht lukt het jou ook al om deze code met gemak te lezen. Let op de puntkomma’s als scheidingstekens.

```
//voorbeeldvierkantentrap.asy
size(4cm);
picture fig;
path vierkant=unitsquare;
transform v=shift(2, 0.5);
for (int i=0; i<6; ++i){
  draw (fig, v^i*vierkant);
}
add(fig);
```

```
//
size(4cm);
picture fig;
path vierkant=unitsquare;
transform v=shift(2, 0.5);
draw (fig, vierkant);
draw (fig, v*vierkant);
draw (fig, v^2*vierkant);
draw (fig, v^3*vierkant);
draw (fig, v^4*vierkant);
draw (fig, v^5*vierkant);
```



Vergelijk deze twee programma's met elkaar. Ze produceren beide dezelfde tekening, maar wat zijn dan de verschillen? Welk programma vind jij het meest 'elegant' geschreven?

8.3.1 Het Collatz-probleem met for

```
//voorbeelddriekaapluseen.asy
for(int k=13; k!=1;  ){
  if (k%2 == 0)
    k=(int)(k/2);
  else
    k=3*k+1;
  write(k);
}
```

Output Window

```
13
40
20
10
5
16
8
4
2
1
```

Het Collatz-probleem kan ook met een for-structuur worden beschreven. Let op de lege puntkomma (;) om aan te geven dat de ophoging van de lusvariabele `k` niet is ingevuld, wegens te complex. Op die manier kan elke while-structuur in een for-structuur worden omgevormd.

8.3.2 Geneste for's

```
//voorbeeldmachten.asy
int aantal=6;
string rij;
for (int n=0; n<aantal; ++n){
  rij="n="+(string)n+": ";
  for (int p=0; p<=n; ++p){
    rij=rij+" "+(string)(p^n);
  }
  write(rij);
}
```

Output Window

```
n=0:  1
n=1:  0 1
n=2:  0 1 4
n=3:  0 1 8 27
n=4:  0 1 16 81 256
n=5:  0 1 32 243 1024 3125
```

Geneste for-structuren zijn uitermate geschikt om tabellen te ontwerpen.

8.3.3 De inhoud van het berichtenvenster kopiëren

Merk op dat je de inhoud van het uitvoervenster kunt kopiëren naar uw favoriete editor. In Geany gaat dit als volgt:

Berichtenvenster kopiëren

De inhoud van het berichtenvenster kopiëren:

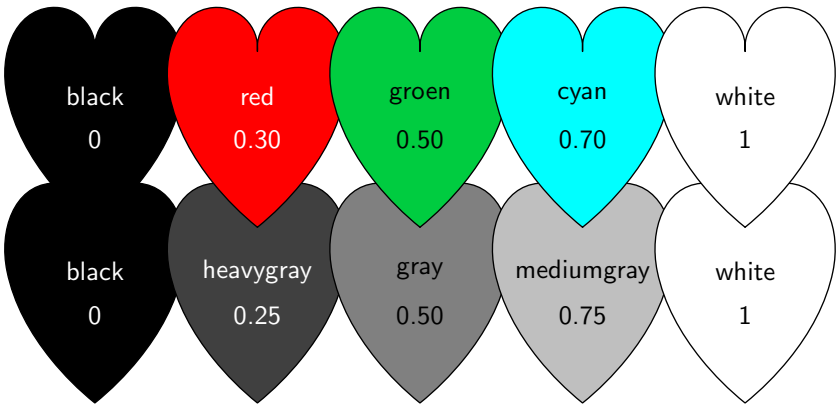
In Geany:

→ rechts klikken in het berichtenvenster (onderaan)

→ `Kopieer` (de regel waar je in staat) of `Alles kopiëren`

8.4 Kleuren in zwart-wit

Let op als je afbeeldingen gebruikt die ook moeten worden afgedrukt op een zwart-wit-printer. Gebruik dan kleuren met duidelijk verschillende grijswaarden.

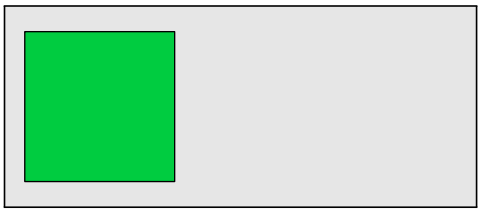


De kleuren `red`, `groen` en `cyan` zijn goed te onderscheiden op een afdruk in grijswaarden. Bij elke kleur staat de grijswaarde vermeld. De grijswaarde van een RGB-kleur wordt berekend met behulp van de functie

$$\text{Grijs} = 0,30 \cdot R + 0,59 \cdot G + 0,11 \cdot B \qquad \text{(YIQ- standaard)}$$

De kleur `groen` is een zelfgemaakte kleur.

```
//voorbeeldgroen.asy
pen groen=rgb(0.00, 0.80, 0.25);

size(2cm);
filldraw(unitsquare, groen);
```



Je kunt zelf kleuren maken door middel van de functie `rgb`. De getallen (tussen 0 en 1) bepalen het percentage rood, groen en blauw van de kleur.

8.5 Functies met strings

```
//voorbeeldstring.asy
string woord="Zon";
int l=length(woord);
write (l);

int k;
k=find("maandagavond", "dag");
write(k);
k=find("braambes", "a");
write(k);

int k=find("maandag", "nacht");
write(k);
```

Output Window

```
3



4

2


-1   (niet gevonden)
```

De functie `length` geeft de lengte (= aantal tekens) van een string. De lege string (`""`) heeft lengte 0.

De functie `find` geeft de positie in de string waar het te zoeken woord begint. Let op: het eerste karakter staat op positie 0, het tweede karakter op positie 1, enzovoort. Asymptote begint dus te tellen vanaf 0.

```
string s=substr("kampioen",3,2);
write (s);
string s=substr("kampioen",3,5);
write (s);
```

Output Window

```
pi

pioen
```

De functie `substr` geeft een deel van een string. Let op: Asymptote telt vanaf 0 voor het eerste karakter. Het tweede getalargument bepaalt de lengte van de substring.

```
//voorbeeldstring.asy
string s;
s=insert("zonkind", 3, "ne");
write (s);
s=erase("pompoen", 1, 4);
write (s);
s=reverse("ROOD");
write (s);
s=replace(
  "De zon gaat naar de zon.",
  "zon",
  "maan"
);
write(s);
```

Output Window

```
zonnekind

pen

DOOR

"De maan gaat naar de maan."
```

De functies `insert`, `erase`, `reverse` en `replace` spreken voor zich.

Een toepassing met for

```
//voorbeeldspatiesopruimen.asy
string s=("Rij jij of rij ik?");
int l=length(s);
int k;
for (int i=0; i<l; ++i){
  k = find(s, " "); //zoek spatie
  s=erase(s, k, 1); //wis spatie
}
write (s);
```

Output Window

```
Rijjijofrijik?
```

Dit programma gaat de hele string af, karakter voor karakter, op zoek naar een spatie, en wist telkens deze spatie.

27 Maak dit programm opnieuw, maar met heel wat minder regels code.

8.6 Time

```
//voorbeeldtime.asy
string s;
s=time("%a"); write (s);
s=time("%b"); write (s);
s=time("%B"); write (s);
s=time("%c"); write (s);
s=time("%d"); write (s);
s=time("%j"); write (s);
s=time("%u"); write (s);
s=time("%W"); write (s);
s=time("%X"); write (s);
s=time("%z"); write (s);
s=time("%Y-%m-%d %H:%M:%S");
write (s);
s=time("%Y%m%d%H%M%%S");
write (s);
s=time("Datum: %Y-%m-%d");
write (s);
```

Output Window

```
Sun
Nov
November
Sun Nov 25 19:21:01 2007
25
329    (dag 001-366)
1      (weekdag 1-7)
47     (weeknummer 00-53)
19:21:01
+0100  (tijdzone)

2007-11-25 19:21:01

20071125192101

Datum: 2007-11-25
```

De functie `time(...)` produceert een string met datum- en tijdgegevens van de computerklok. De opmaak kun je zelf kiezen door middel van lettercodes, voorafgegaan door het procentteken %.

```
//voorbeeldnamiddag.asy
string tijd=time("%H:%M:%S");
write (tijd);
if (tijd > "12:")
  write ("namiddag");
else
  write ("voormiddag");
```

Output Window

```
20:48:41
namiddag
```

Alle strings die op `"12"` volgen, duiden tijdstippen in de namiddag aan. Voorbeeld: `"08:..."` komt voor `"12"`, dat voor `"20:..."` komt. Door middel van het `%H`-formaat worden de eventuele voorloopnullen voorzien.

8.7 Rand

```
//voorbeeldtoeval.asy
int t;
t=rand(); write (t);
t=rand(); write (t);
t=rand(); write (t);
t=rand(); write (t);
t=rand(); write (t);
```

Output Window

```
0
1481765933
1085377743
1270216262
1191391529
```

De functie `rand()` produceert een toevalsgetal. Dit is een 'lukraak' gekozen int. Elke keer als de functie wordt opgeroepen, wordt een nieuw toevalsgetal gegenereerd.

```
//voorbeeldtoeval.asy
int t;
for (int i=0; i<5; ++i){
  t=rand();
  write (t);
}
```

Output Window

```
0
1481765933
1085377743
1270216262
1191391529
```

Computers kunnen geen echte toevalsgetallen genereren. Zij gebruiken altijd een of andere formule om deze getallen te genereren. Dergelijke getallen 'lijken' wel lukraak gekozen, maar schijn bedriegt. Start het programma maar eens opnieuw, dan zie je dat dezelfde reeks toevalsgetallen worden gegenereerd. De reeks van toevalsgetallen is dus zeer voorspelbaar, niet echt geschikt om bijvoorbeeld een loterijtrekking na te bootsen.

We spreken van *pseudo-ramdomgetallen*. Asymptote gebruikt dus een formule om pseudo-randomgetallen te genereren. Hij vertrekt daarbij van een eerste getal, *zaadje* genoemd, en berekent op basis van dit zaadje het eerste toevalsgetal. Op basis van dit eerste toevalsgetal wordt het volgende toevalsgetal berekend, door het vorige toevalsgetal als zaadje in de formule te gebruiken, enzovoort. Standaard wordt voor het zaadje het getal `0` genomen. Op die manier krijg je steeds dezelfde (voorspelbare) reeks toevalsgetallen.

```
//voorbeeldtoeval.asy
srand(seconds());

int toevalsgetal;
for (int i=0; i<5; ++i){
  toevalsgetal=rand();
  write (toevalsgetal);
}
```

Output Window

```
1782289037
2061934816
1870761302
656833343
834270458

(uitgevoerd op
 2008-01-10 15:09:23)
```

Je kunt als programmeur zelf het zaadje bepalen met de functie `srand(...)`. Daardoor krijg je een totaal andere rij van pseudo-toevalsgetallen. Door het zaadje te laten afhangen van een min of meer echte toevalsfactor, kun je toch min of meer echte toevalsgetallen genereren. Je kunt bijvoorbeeld het tijdstip waarop het programma uitgevoerd wordt (de kloktijd van de computer) beschouwen als 'toevallig'. We hebben hier de functie `seconds()` gebruikt, die het aantal seconden (zonder de schrikkelseconden) weergeeft dat verstreken is sinds 00:00:00 GMT 1 januari 1970, de Unix Epoch.

```
//voorbeelddobbel.asy
srand(seconds());

int dobbel;
for (int i=0; i<5; ++i){
  dobbel=rand()%6+1;
  write (dobbel);
}
```

Output Window

```
5
2
5
6
4
```

De functie `rand()` produceert een int, dus een natuurlijk getal tussen 0 en 2 147 483 647. Door de rest te nemen van de deling door 6, krijgen we een natuurlijk getal tussen 0 en 5. Tel er 1 bij op, en je hebt een toevalsgetal tussen 1 en 6, precies de mogelijke uitkomsten van een worp met een dobbelsteen.

```
//voorbeelddobbelspel.asy
srand(seconds());

int dobbel;
bool gewonnen=false;
while (!gewonnen){
  dobbel=rand()%6+1;
  write (dobbel);
  if (dobbel==6) gewonnen=true;
}
write ("Gewonnen");
```

Output Window

```
2
2
5
4
3
5
6
Gewonnen
```

Dit spel eindigt als er een 6 ‘geworpen’ wordt.

8.8 Opdrachten

28

```
//voorbeeldfibonacci.asy
int a, b, n, nieuw;
a=1;
b=1;
n=10;
int i=2;
while (i<n){
  i=i+1;
  nieuw=a+b;
  write(i, nieuw);
  a=b;
  b=nieuw;
}
```

Output Window

```
3    2
4    3
5    5
6    8
7    13
8    21
9    35
10   55
```

De rij van Fibonacci groeit zeer snel. Het 47ste getal is al groter dan de grootste int.

Wat moet je aan het programma veranderen om toch grotere Fibonacci-getallen ($n = 47$, $n = 48$,...) te kunnen berekenen in Asymptote?

29 Schrijf een nieuw programma `dobbelspel.asy`. Er wordt telkens opnieuw een dobbelsteen gegooid. Als je een zes gooit, stopt het spel en verschijnt de volgende boodschap:

Output Window

```
2
2
5
4
3
5
6
Gewonnen na 7 beurten
```

30 Schrijf een nieuw programma `hoeveela.asy`. Het programma telt hoeveel keer de letter 'a' voorkomt in een string.

```
//hoeveela.asy
string s="Afvalkalender";
...
write(aantala);
```

Output Window

```
2
```

31 Schrijf een nieuw programma `derangement.asy`.

Het programm berekent enkele termen van de rij met het volgende voorschrift:

$$d(1) = 0$$
$$d(2) = 1$$
$$d(n) = (n-1) \cdot (d(n-2) + d(n-1)) \text{ voor } n \geq 3$$

```
//derangement.asy
...
```

Output Window

```
1  0
2  1
3  2
4  9
5  44
6  265
7  1854
8  14833
9  133496
10 1334961
```

Het aantal derangementen $D(n)$ is het aantal manieren dat n objecten niet op hun plaats zitten. Voorbeeld: je stuurt vier verschillende brieven naar vier verschillende adressen, zodanig dat geen enkele brief in de juiste omslag zit. Dit kan op $D(4) = 44$ verschillende manieren.

32 Leg uit wat het volgende programma doet.

```
//zomaar.asy
srand(seconds());
int k=rand()%1000;
write(k);
string s=(string)k;
write ((s == reverse(s)));
```

Output Window

```
680
false
```

33 Schrijf een nieuw programma `priem.asy` om te onderzoeken of een gegeven getal een priemgetal is. Bovendien moet de ontbinding in priemfactoren worden afgedrukt.

```
//priem.asy
int k=2008;
bool ispriem;
string ontbinding;
...
write(ispriem);
write(ontbinding);
```

Output Window

```
false
2  2  2  251
```

34 Schrijf een nieuw programma `abcwoord.asy`. Het programma onderzoekt of de letters van een gegeven woord (geschreven in kleine letters) in alfabetische volgorde staan.

```
//abcwoord.asy
string s="bijoux";
...
```

Output Window

```
true
```

```
//abcwoord.asy
string s="zondag";
...
```

Output Window

```
false
```

Test het programma uit met de volgende woorden:

```
jan (false), ann (true), benny (true), erik (false),
abcdef (true), zorro (false), stuw (true)
```

35 Schrijf een nieuw programma `voornaameerst.asy`.

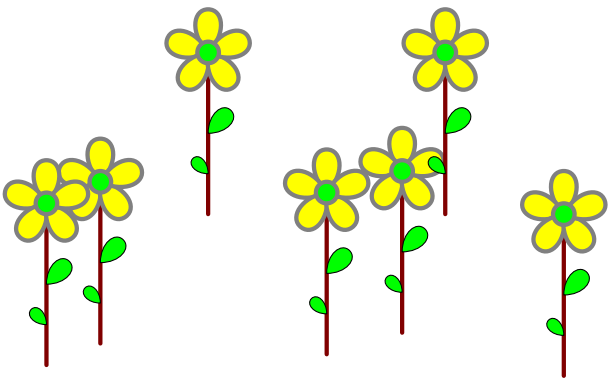
Het programma zet een gegeven naam, bijvoorbeeld `"Gevaert, Kim"` om naar een vorm waarin de voornaam vooraan staat, dus: `"Kim Gevaert"`.

```
//voornaameerst.asy
string s="Gevaert, Kim";
...
```

Output Window

```
Kim Gevaert
```

36 Schrijf een nieuw programma `bloemen.asy`. Het programma tekent een picture `boterbloem`, en zet zeven kopies op willekeurige plaatsen zodat een 'weide' met boeterbloemen ontstaat. Zie figuur 15.



Figuur 15 – Een weide met boterbloemen

37 Schrijf een nieuw programma `brownbeweging.asy`. Het programma tekent een zigzaglijn waarbij telkens één stap naar rechts gezet wordt. Tegelijkertijd wordt er volgens het toeval ofwel een stap naar boven ofwel een stap naar onder gezet. Zie figuur 16.

Figuur 16 – Brownse zigzaglijn, een beweging bepaald door het toeval.

Dergelijke bewegingen staan bekend als *Brownbewegingen*. In 2006 werd een *Fields medaille* uitgereikt aan Wendelin Werner, onder andere voor zijn bijdrage aan de ontwikkeling van de geometrie van de tweedimensionale Brownbeweging.

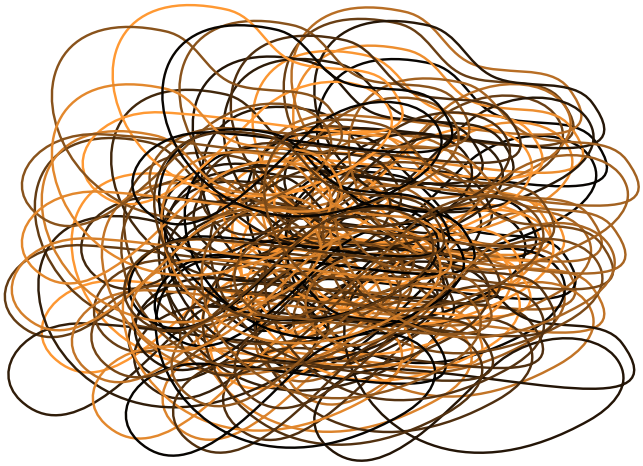
De Fields-medaille is een onderscheiding die elke vier jaar aan twee tot vier wiskundigen wordt toegekend. Ze is genoemd naar de Canadese wiskundige John Charles Fields die de prijs initieerde en financierde. De prijs wordt uitgereikt door de Internationale Wiskundige Unie op de openingsceremonie van het Internationaal Wiskundecongres.

De prijs wordt vaak gezien als de belangrijkste onderscheiding binnen de wiskunde en wordt om die reden regelmatig 'de Nobelprijs voor de wiskunde' genoemd. Deze vergelijking gaat echter niet helemaal op, want behalve dat de prijs slechts eens in de vier jaar wordt uitgereikt, is ze strikt voorbehouden aan wiskundigen jonger dan veertig. Met de invoering van de Abelprijs in 2003 lijkt de wiskunde overigens een beter equivalent voor de Nobelprijs te hebben gekregen.

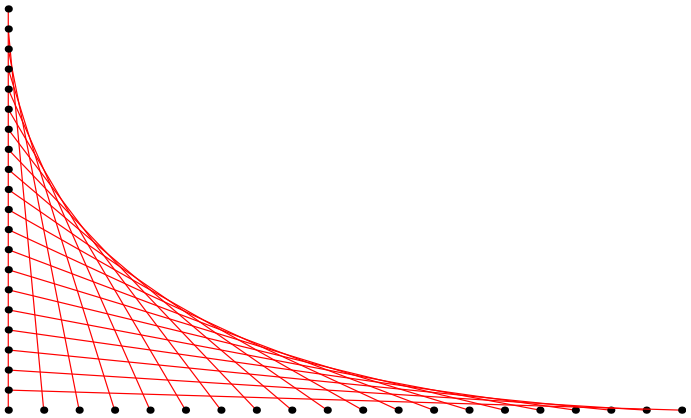
Eén keer werd een kleine uitzondering op de leeftijdsregel gemaakt, toen Andrew Wiles in 1998 een speciale vermelding kreeg voor zijn bewijs van de Laatste Stelling van Fermat. Wiles was ten tijde van de vermelding al 45 jaar oud.

38 Maak een nieuw programma `inspiratie.asy`.

Teken de volgende figuur, die bestaat uit 52 kopies van een gesloten kromme. De plaats van elke kopie is bepaald door het toeval.



39 Maak een nieuw programma `curvestitching.asy`. Het programma produceert de tekening in figuur 17.



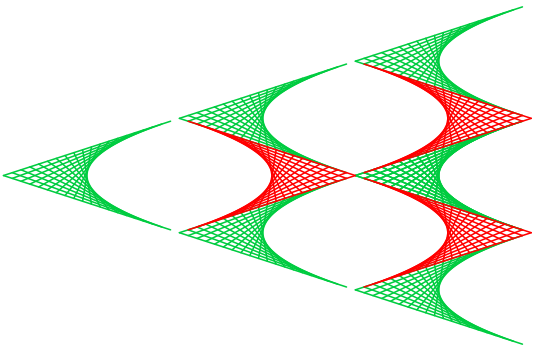
Figuur 17 – Curve stitching, or string art, is a beautiful way of combining geometry with art. These wonderful designs are created on different shaped grids, and straight lines are sewn or drawn to look like curves. Mary Boole designed string geometry to help children learn about the geometry of angles and spaces.

Mary Everest Boole (1832-1916) leerde zichzelf wiskunde en is het meest bekend als auteur van didactische werken over wiskunde, zoals *Philosophy and Fun of Algebra* (Everest Boole, *The Preparation of the Child for Science*), en als de vrouw van de wiskundige George Boole.

Haar leven is ook interessant voor feministes als een voorbeeld van hoe vrouwen carrière maakten in academische milieus, waar ze niet welkom waren. Haar oom, George Everest, gaf zijn naam aan Mount Everest.

40 Maak een nieuw programma `curvestitchingvliegers.asy`.

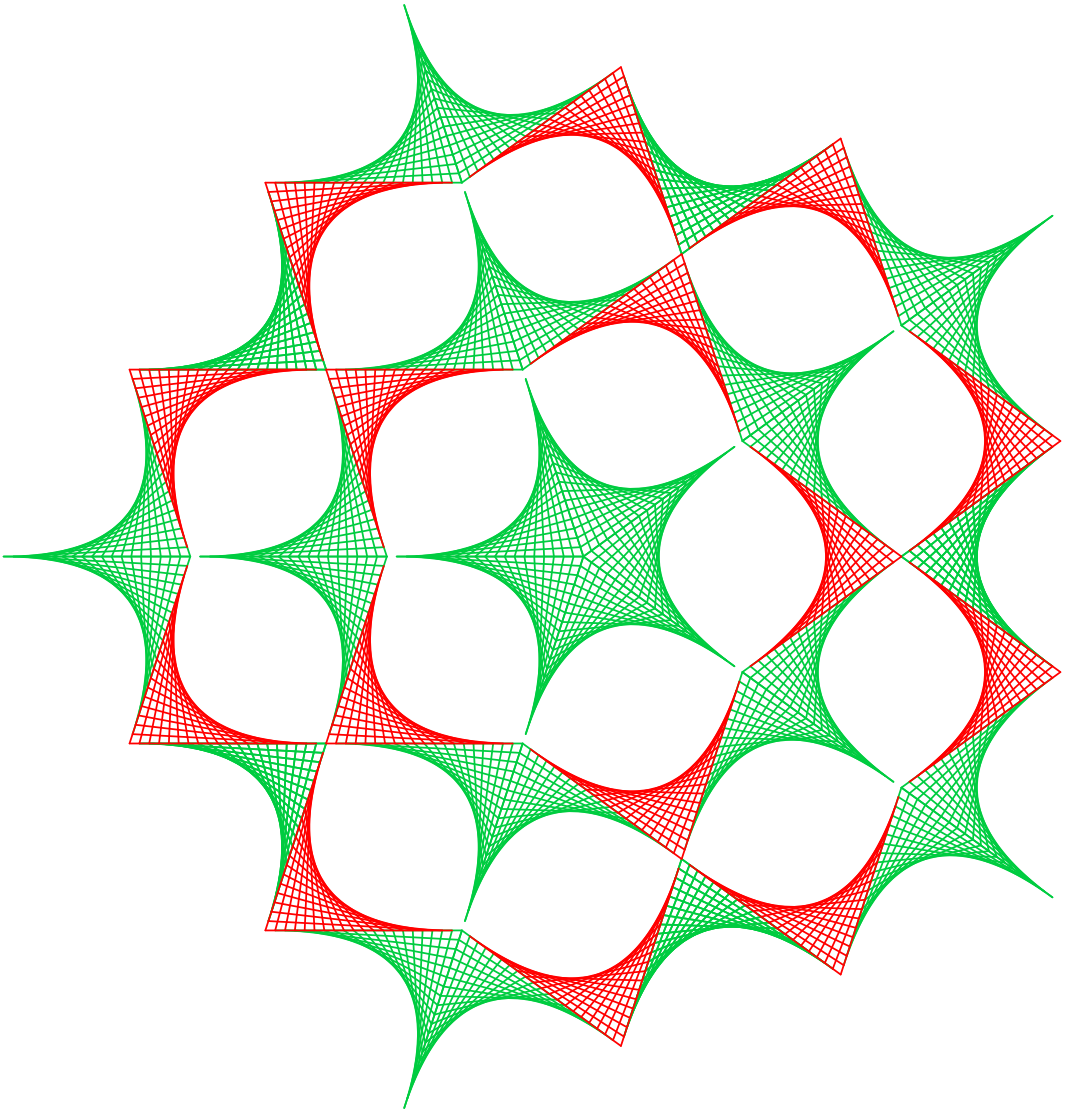
Zie figuur 18.



Figuur 18 – Curve stitching: Spieën

41 Maak een nieuw programma `curvestitchingvliegersvijfhoek.asy`.

Zie figuur 19.



Figuur 19 – Curve stitching: Vijfhoek

42 Schrijf een programm `meervoudigespaties.asy` dat een gegeven zin, bijvoorbeeld `"Ga naar huis."`, herschrijft in een zin waarbij meervoudige spaties vervangen worden door een enkele spatie.

```
//meervoudigespaties.asy
string s="Ga   naar    huis.";
...
write (s);
```

Output Window

```
Ga naar huis.
```

43 Schrijf een programm `zinsontleding.asy` dat een gegeven zin, bijvoorbeeld `"Ga naar de bakker broodjes halen."`, woord voor woord afdrukt op aparte regels. Een nieuw woord begint na een of meer spaties.

```
//zinsontleding.asy
string s="Ga naar de
  bakker broodjes halen";
...
```

Output Window

```
Ga
naar
de
bakker
broodjes
halen.
```

44 Schrijf een programma `dobbelstenensom7.asy` dat 100 keer 'twee dobbelstenen gooien' simuleert, en afdrukt hoeveel keer de som van de twee dobbelstenen gelijk was aan 7.

```
//dobbelstenensom7.asy
...
```

Output Window

```
16 keer som=7
```

45 Schrijf een programma dat de som berekent van de kwadraten van de getallan 1, 2, ... 100.

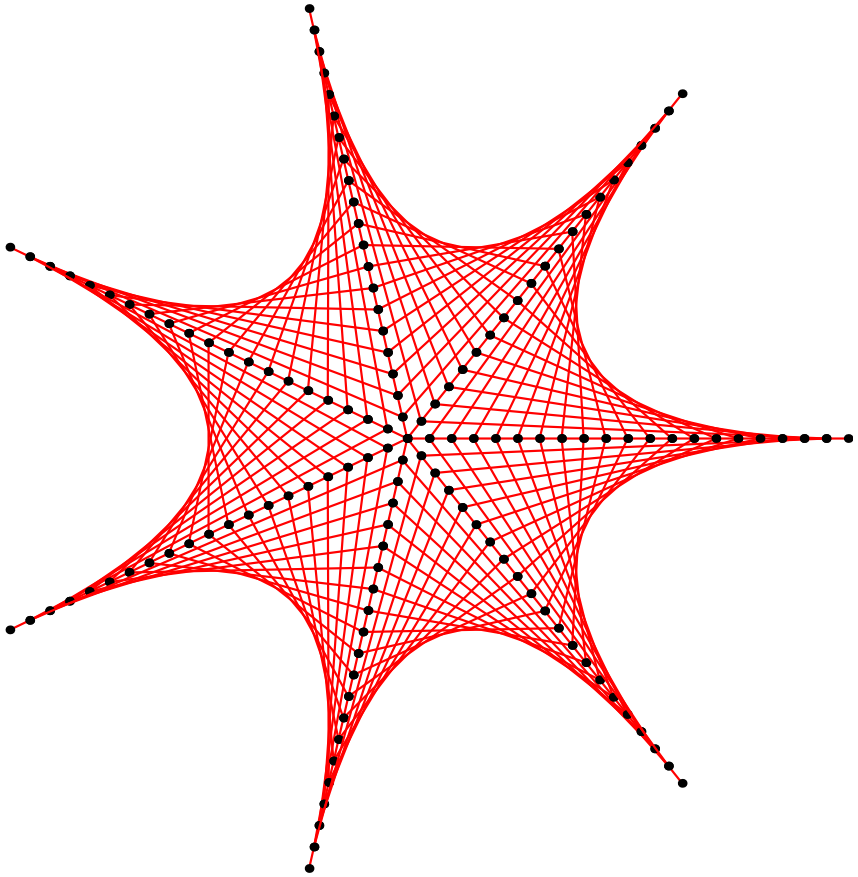
```
//somvankwadraten.asy
...
```

Output Window

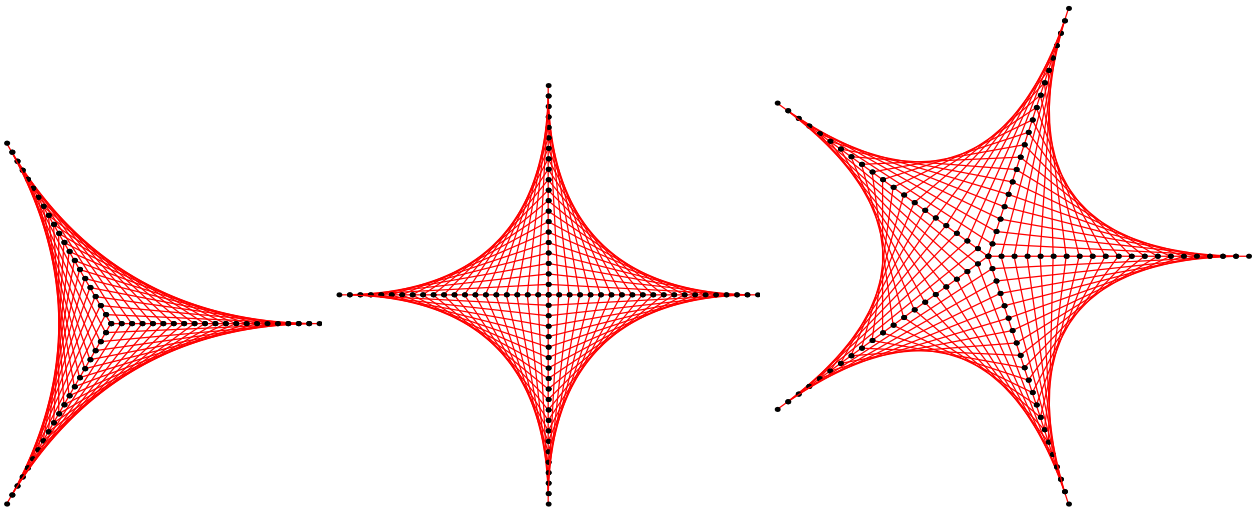
```
338350
```

46 Schrijf een programma `curvestitchingveelhoek.asy`, dat de tekening in figuur 20 produceert.

Maak het programma zodanig dat door een wijziging in slechts één variabele, bijvoorbeeld n het programma een andere veelhoek tekent. Zie figuur 21 en figuur 22.

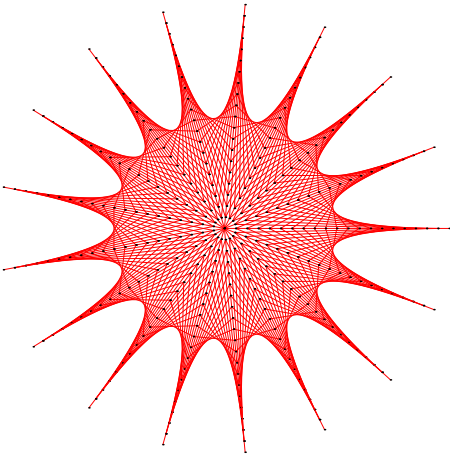


Figuur 20 – Een zevenhoek gevormd door middel van curvestitching



(a) $n = 3$ **(b)** $n = 4$ **(c)** $n = 5$

Figuur 21 – Curvestitchingveelhoeken



Figuur 22 – Curvestitching-veelhoek ($n = 17$)

Referenties

1 *C++ Reference – fprint*. Engels. http://www.cplusplus.com/reference/clibrary/cstdio/printf.html. The C++ Resources network, 2002-2007.

2 Mary Everest Boole. *The Preparation of the Child for Science*. Engels. University of Oxford, 1904.

3 Andy Hammerlindl, John Bowman en Tom Prince. *Asymptote: the Vector Graphics Language*. Engels. http://asymptote.sourceforge.net/. 2007.

4 Wilfried Van Hirtum. *LaTeX voor beginners - deel 1. Direct beginnen met de online editor ShareLaTeX*. 2016. url: https://denkendehanden.be/.

Trefwoordenregister


Symbols


A


B


C

L

M

N

O

P

Q

R

S