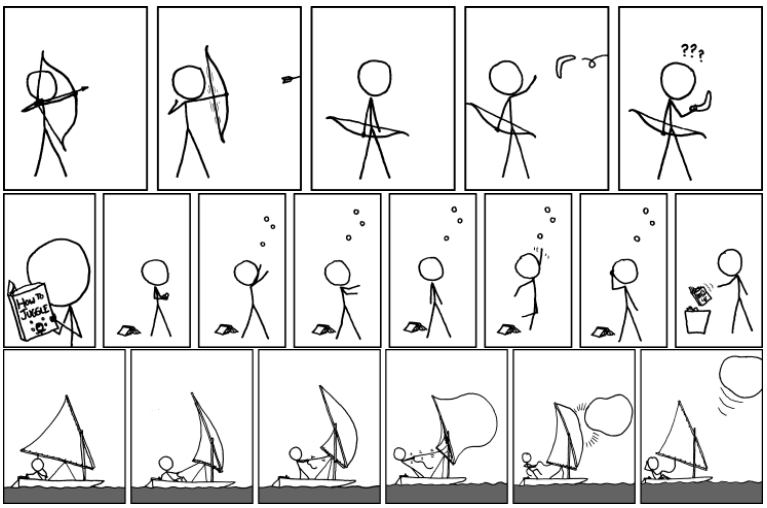
Universiteit van Amsterdam
Inleiding Programmeren

Opdracht 2

Skip Thijssen

De Collatz-reeks



1 Introductie

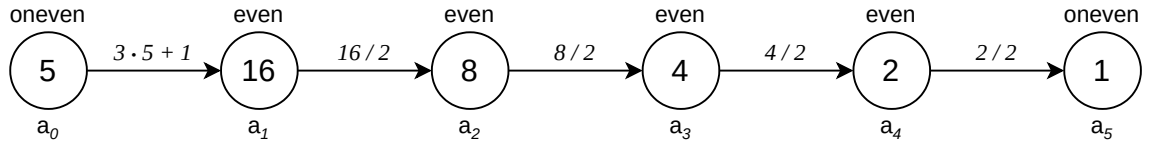
De voorgaande week heb je geoefend met het schrijven van je eerste programma's. Deze week gaan we aan de slag met het maken van een enkel, samenhangend programma. Het onderwerp dat hierbij centraal zal staan is het gebruik van functies. Deze zullen je helpen herhaling in je code te voorkomen, houden de code goed leesbaar, en maken het makkelijker om achteraf aanpassingen te maken. Om te oefenen met het maken van functies gaan wij ze in deze opdracht gebruiken bij het implementeren van een Collatz-reeks.

1.1 De Collatz-reeks

Het vermoeden van Collatz stelt dat alle positieve gehele getallen n na een aantal iteraties volgens de onderstaande regels altijd uitkomen op het getal 1.

$$a_0 = n \qquad\qquad a_{n+1}(n) = \begin{cases} n/2 & \text{als } n \text{ even is} \\ 3 \cdot n + 1 & \text{als } n \text{ oneven is.} \end{cases}$$

De opvolger van n wordt dus telkens bepaald door de pariteit van n: een even waarde wordt gehalveerd, terwijl een oneven waarde juist wordt verdrievoudigd waarna er nog een bij wordt opgeteld.



Voorbeeld De bovenstaande figuur illustreert de Collatz-reeks van het getal 5. Alle reeksen volgen dezelfde twee hierboven genoemde regels. Zodra het getal 1 wordt bereikt gaat de reeks eigenlijk verder met $4, 2, 1, 4, 2, 1, \ldots$, maar het interessante deel van de reeks zijn de waardes tot en met het eerste voorkomen van het getal 1. We noemen de lengte van het beginstuk van de reeks, tot en met de eerste 1, de "stoptijd" van de reeks. Bijvoorbeeld, de stoptijd van de reeks $1, \ldots$ is 1. De stoptijd van de reeks $5, 16, 8, 4, 2, 1, \ldots$ is 6.

Meer informatie De *Collatz conjecture* is genoemd naar Lother Collatz, die in 1937 het probleem benoemde. Het probleem staat ook wel bekend als het "*3n+1 problem*" en onder nog vele andere namen van mensen die destijds toevallig ook met hetzelfde probleem bezig waren.

Het heet een *conjecture* (vermoeden) omdat het er weliswaar op lijkt dat het waar is, maar niemand is tot nu toe in staat geweest om dat ook wiskundig te bewijzen. In 2020 zijn de reeksen die beginnen met alle getallen van 1 tot $2.95 \cdot 10^{20}$ door de computer getest. Je zou kunnen denken dat het vermoeden wel juist moet zijn als het tegendeel nog steeds niet bewezen is met zulke uitgebreide tests. Echter, er zijn veel voorbeelden van wiskundige beweringen die waar lijken maar dat niet zijn, en waar het kleinste tegenvoorbeeld enorm is. (Bijvoorbeeld, het lijkt in eerste instantie waar te zijn dat $n^{17} + 9$ en $(n+1)^{17} + 9$ geen gemeenschappelijke delers hebben, maar het blijkt dat ze die wel degelijk hebben voor $n = 842443292559288932928819732230890067245942046079243 3$.)

Veritasium heeft over dit onderwerp ook een leuke video gemaakt.[1]

1.2 De opdracht

In deze opdracht zul je functies maken om de Collatzreeks uit te rekenen, en vervolgens om te onderzoeken welke begingetallen leiden tot reeksen met een lange stoptijd, of met de grootste getallen erin.

Bij deze opdracht wordt opnieuw gebruik gemaakt van een *Makefile*. Deze makefile heeft verschillende "targets":

[1] `https://youtu.be/094y1Z2wpJg`

`make`

- `make`, compileert `opdracht_2.c`.
- `make clean`, verwijdert de object (`.o`) en executable (`.out` / geen extensie) files.
- `make tarball`, maakt een `.tar.gz` archief van de bestanden die ingeleverd moeten worden.
- `make opdracht_2`, compileert `opdracht_2.c`.

Behalve een `Makefile` krijgen jullie deze week ook een skelet van de uitwerking aangeleverd in het bestand `opdracht_2.c`. In dit bestand staat al wat code gegeven die je wat op weg zal helpen.

1.3 Command line argumenten

Deze opdracht maakt gebruikt van command line argumenten om invoer binnen te krijgen. Dit is in plaats van de `scanf()` functie, die vorige week is gebruikt. De command line argumenten worden achter de naam van het programma toegevoegd op het moment dat je het programma gaat uitvoeren.

```
> ./opdracht_2 5 9
```

Figuur 1: Voorbeeldinvoer van de getallen 5 en 9 als command line argumenten voor *opdracht_2* in de terminal.

Je kunt in `opdracht2.c` zien hoe deze argumenten aan de functie `main` worden aangeboden via de inputs `argc` en `argv`, en naar `int`-waardes worden vertaald met behulp van de functie `atoi`.

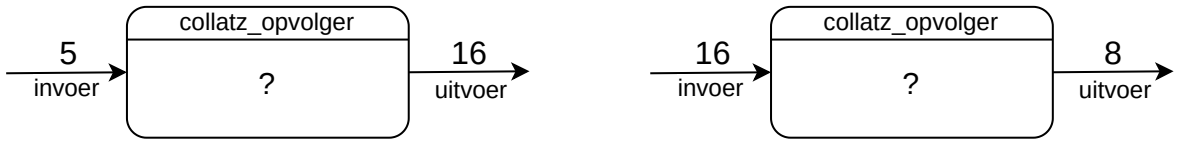
2 Opdracht 2: Implementatie van de Collatz-reeks

De gebruiker geeft het programma twee getallen mee, a en b. Hierbij moet gelden dat $a \leq b$ (controleer dit in je programma). Vervolgens zal jouw programma informatie afdrukken over twee reeksen, allebei met een begingetal tussen a en b. Welke twee reeksen dit zijn, wordt bepaald door de functies `collatz_langste_tussen` en `collatz_grootste_tussen`. Echter, voordat deze twee functies geschreven kunnen worden, moeten de `collatz_opvolger`, `collatz_stoptijd`, en de `collatz_grootste` functies geïmplementeerd worden.

Tip Lees voordat je begint met programmeren eerst de hele opdracht door.

2.1 Functie 1: `collatz_opvolger(n)` **(0.5 pt)**

De eerste functie die geïmplementeerd moet worden is `collatz_opvolger`. Deze functie krijgt een getal binnen en geeft het volgende getal in de Collatz-reeks terug.



Figuur 2: `collatz_opvolger` functie illustratie.

Het verwachte gedrag voor een gegeven invoer is nogmaals geïllustreerd in figuur 2. Aan jou is de taak om deze functie te schrijven. Het begin van de functie is al aangeleverd in `opdracht_2.c`. Let op: wijzig nooit de functiedeclaraties in aangeleverde code!

Tip Maak gebruik van `printf`-statements om te controleren of de teruggegeven waardes overeenkomen met wat je zelf verwacht.

Tip Rondom de `main()` functie van `opdracht_2.c` staat `#ifndef COMPILE_MAIN` en `#endif`. Dit wordt gebruikt om de code te testen en kan je negeren.

2.2 Functie 2: `collatz_stoptijd(n)` (1 pt)

De tweede functie is de `collatz_stoptijd` functie. Deze functie berekent de stoptijd van de Collatz-reeks voor een gegeven begingetal. De functie geeft deze stoptijd als uitvoer terug.

Voorbeeld

Collatz-reeks 1: $1, 4, 2, 1, \ldots$	stoptijd $= 1$.
Collatz-reeks 2: $2, 1, 4, 2, \ldots$	stoptijd $= 2$.
Collatz-reeks 3: $3, 10, 5, 16, 8, 4, 2, 1, 4, \ldots$	stoptijd $= 8$.
Collatz-reeks 4: $4, 2, 1, 4, \ldots$	stoptijd $= 3$.

Hint Denk er aan dat je gebruik kan maken van functies die je eerder hebt geschreven.

2.3 Functie 3: `collatz_grootste(n)` (1 pt)

De derde functie is de `collatz_grootste` functie. Deze functie zoekt het grootste getal dat optreedt vóór de eerste 1 in de Collatz-reeks van een gegeven getal. De functie geeft het grootste getal van de reeks als uitvoer terug.

Voorbeeld

Collatz-reeks 1: $1, 4, 2, \ldots$	grootste $= 1$ (dus niet 4!).
Collatz-reeks 2: $2, 1, 4, \ldots,$	grootste $= 2$ (dus niet 4!)
Collatz-reeks 3: $3, 10, 5, 16, 8, 4, 2, 1, 4, 2, \ldots$	grootste $= 16$.
Collatz-reeks 4: $4, 2, 1, 4, 2, \ldots$	grootste $= 4$.

2.4 Functie 4: `collatz_langste_tussen(a, b)` (1 pt)

De vierde functie is de `collatz_langste_tussen` functie. Deze functie zoekt de Collatz-reeks met de grootste stoptijd tussen alle reeksen die beginnen met een getal tussen a en b (inclusief a en b zelf). Van de Collatz-reeks met de grootste stoptijd wordt het beginnende getal, de stoptijd, en het grootste getal afgedrukt.

Als er meerdere reeksen in het bereik van a t/m b dezelfde lengte hebben, dan wordt alleen de eerste reeks gebruikt.

Zodra je klaar bent kun je in de functie `main` de aanroep van de functie `collatz_langste_tussen(a, b)`, die nu nog als commentaar in het programma staat, actief maken.

Voorbeeld Gegeven worden $a = 5$ en $b = 9$. Dan is de situatie dus als volgt (dit is alleen beschrijving en dus nog niet de gewenste uitvoer van je programma):

Collatz-reeks 5:	$5, 16, 8, 4, 2, 1, \ldots$	stoptijd $= 6$.
Collatz-reeks 6:	$6, 3, 10, 5, 16, 8, 4, 2, 1, \ldots,$	stoptijd $= 9$.
Collatz-reeks 7:	$7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, \ldots,$	stoptijd $= 17$.
Collatz-reeks 8:	$8, 4, 2, 1, \ldots,$	stoptijd $= 4$.
Collatz-reeks 9:	$9, 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, \ldots,$	stoptijd $= 20$.

Collatz-reeks 9 heeft de grootste stoptijd. Het grootste getal dat in deze reeks voorkomt (voordat de eerste 1 optreedt) is 52. Dus de functie drukt de volgende informatie af:

```
9 20 52
```

2.5 Functie 5: `collatz_grootste_tussen(a, b)` **(1 pt)**

De vijfde functie is de `collatz_grootste_tussen` functie. Deze functie zoekt de Collatz-reeks tussen de gegeven getallen a en b die de grootste waarde bevat. Van de grootste Collatz-reeks wordt net zoals hiervoor het beginnende getal, de stoptijd en het grootste getal afgedrukt. Als er meerdere reeksen in het bereik van a t/m b hetzelfde grootste getal hebben, dan wordt alleen de eerste reeks gebruikt.

Zodra je klaar bent kun je de de aanroep van `collatz_grootste_tussen(a, b)` actief maken in de functie `main()`.

Voorbeeld Gegeven $a = 5$ en $b = 9$.

Collatz-reeks 5:	$5, 16, 8, 4, 2, 1, \ldots,$	grootste = 16.
Collatz-reeks 6:	$6, 3, 10, 5, 16, 8, 4, 2, 1, \ldots,$	grootste = 16.
Collatz-reeks 7:	$7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, \ldots,$	grootste = 52.
Collatz-reeks 8:	$8, 4, 2, 1,$	grootste = 8.
Collatz-reeks 9:	$9, 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, \ldots>,$	grootste = 52.

Collatz-reeks 7 heeft de grootste waarde van 52. Deze reeks heeft lengte 17. Dus de functie drukt de volgende informatie af:

```
7 17 52
```

2.6 Functie 6: `collatz_maximum()` **(0.5 pt)**

De zesde functie is de `collatz_maximum` functie. Deze functie staat los van functie 1-5, en zoekt de eerste Collatz-reeks die voor een overflow zorgt (zie hoorcollege 2). Je zult misschien al gemerkt hebben dat sommige hoge reeksen voor problemen zorgen (zoals 1 000 000), terwijl lagere reeksen prima werken (zoals 100). Dit komt omdat de hoge reeksen getallen kunnen bevatten die niet in een `int` passen.

Voor deze laatste opgave is het aan jou om de eerste reeks te vinden die voor een overflow zorgt. Druk vervolgens het kleinste begingetal af waarvoor overflow optreedt.

Zodra je klaar bent kun je de aanroep van `collatz_maximum()` actief maken in `main()`.

Hint Hoe kun je berekenen dat een overflow gaat plaatsvinden voordat dit ook daadwerkelijk gebeurt?

Tip Als je de maximum waarde van een `int` of ander datatype nodig hebt, kun je deze vinden in de `limits` library die al ge-include is. (Bijvoorbeeld onder de naam `INT_MAX`)

2.7 Functie BONUS: collatz_maximum_verlengd() **(1 pt bonus)**

Begin met het maken van een exacte kopie van `collatz_maximum()` genaamd `collatz_maximum_verlengd()`.

Voor deze bonusopgave ga je het datatype `int` in `collatz_maximum_verlengd()` veranderen in een `unsigned long long`. Wat is nu de eerste reeks waarbij een overflow plaatsvindt?

Je zult waarschijnlijk merken het nu erg lang duurt voordat je de eerste overflow bereikt. Kun je aanpassingen bedenken om `collatz_maximum_verlengd` sneller te maken? Zijn er stappen die je kan overslaan of inkorten?

Zodra je klaar bent kun je de aanroep van `collatz_maximum_verlengd()` actief maken in `main()`.

Tip Houd de aanpassingen die je maakt in deze bonusopgave gescheiden van de code uit functies 1-6. Maak als het nodig is een kopie van de functies die je wilt aanpassen, of verplaats de code naar binnen de `collatz_maximum_verlengd` functie. Als het goed is zul je alleen de code uit `collatz_opvolger` nodig hebben. Het is niet de bedoeling dat functies 1-5 niet meer werken nadat je de bonusopgave hebt gemaakt.

3 Voorbeelduitvoer

```
> ./opdracht_2 5 9
9 20 52
7 17 52
> ./opdracht_2 1 100
97 119 9232
27 112 9232
> ./opdracht_2 32 1024
871 179 190996
703 171 250504
```

Figuur 3: Voorbeelduitvoer van `opdracht_2` in de terminal (zonder de uitvoer van opgave 2.6 en 2.7).

4 Inleveren

Als je programma klaar is om ingeleverd te worden, voer dan het commando "`make tarball`" uit. Dit maakt een archiefbestand `opdracht2_submit.tar.gz` met de bestanden `opdracht2.c` en `Makefile` erin. Alleen dit archief hoeft ingeleverd te worden op Canvas.

Tip Doe de compilatietest op Codegrade voor je inlevert om zeker te zijn dat je programma in orde is.