Stap voor stap uitwerking van een eenvoudige C opdracht

Bij het vak CPL01 krijg je regelmatig een opdracht die als volgt begint: “schrijf een programma dat . . . ”. Hoe pak je zoiets, als beginnend programmeur, nu aan? In TafelsStapVoorStap.pdf leg ik uit hoe je de volgende opdracht aanpakt: “schrijf een programma dat een geheel getal $0 < n < 7$ inleest en vervolgens de tafels van vermenigvuldiging”. Ik raad je aan om dat voorbeeld eerst te bestuderen! Op deze webpagina leg ik uit hoe je de volgende opdracht aanpakt: “benader $\cos(x)$ met de volgende formule $\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$”. In deze formule is x de hoek in radialen waar je de cosinus van wilt benaderen en betekent 4! vier faculteit ($1 \cdot 2 \cdot 3 \cdot 4 = 24$). Het aantal termen dat gebruikt moet worden om de cosinus te benaderen is variabel en moet door het programma worden ingelezen.

Stap voor stap

De methode die ik gebruik om tot een werkend programma te komen dat aan de opdracht voldoet is stapsgewijze verfijning (Engels: stepwise refinement). Ik begin met een eenvoudig programma en breid het programma stap voor stap uit tot ik uiteindelijk het gewenste programma heb. Ik test het programma na elke stap. De bovenstaande opdracht zou ik als volgt aanpakken.

Stap 0: Bezint eer gij begint

De eerste stap heeft nog niets met programmeren te maken. Voordat ik begin te programmeren denk ik eerst over de opdracht na. Snap ik wat ik moet maken? De formule waarmee de cosinus benaderd kan worden is gegeven:

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots \tag{1}$$

Bron: Wikipedia

In de opdracht wordt gesproken over “termen” en “faculteit”. Als ik deze begrippen niet ken dan zoek ik ze op. Volg indien nodig deze links naar Wikipedia: termen en faculteit. Ik stel mezelf de vraag hoe ik in C x^n uit kan rekenen. Met behulp van Google vind ik dat ik x^n in C kan uitrekenen met behulp van de standaard functie pow, zie eventueel cppreference.com. Vervolgens vraag ik mezelf af hoe ik in C $n!$ (n faculteit) uit kan rekenen. Omdat $n!$ gelijk is aan $1 \cdot 2 \cdot 3 \cdot \cdots \cdot n$ ligt het voor de hand om een herhalingsinstructie te gebruiken.

Als ik naar de formule kijk dan zie ik dat ik de eerste term (1) ook kan vervangen door $\frac{x^0}{0!}$. Dat geeft een mooie regelmaat in de termen.

$$\cos(x) = \frac{x^0}{0!} - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots \tag{2}$$

Om vertrouwd te raken met de benaderingsformule reken ik de eerste vier termen uit van cos(1.0). Deze termen zijn:

$$+\frac{1^0}{0!} = +\frac{1}{1} = +1, \tag{3}$$

$$-\frac{1^2}{2!} = -\frac{1}{1 \cdot 2} = -\frac{1}{2} = -0.5, \tag{4}$$

$$+\frac{1^4}{4!} = +\frac{1}{1 \cdot 2 \cdot 3 \cdot 4} = +\frac{1}{24} = +0.0416667 \text{ en} \tag{5}$$

$$-\frac{1^6}{6!} = -\frac{1}{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6} = -\frac{1}{720} = -0.0013889 \tag{6}$$

De benadering van cos(1.0) met vier termen is dus:

$$1 - 0.5 + 0.0416667 - 0.0013889 = 0.5402778 \tag{7}$$

De werkelijke waarde van cos(1.0) is volgens mijn rekenmachine: 0.5403023. Ik zie dat bij een benadering met vier termen van cos(1.0) al drie cijfers achter de decimale punt correct zijn.

Stap 1: Alle begin is ~~moeilijk~~ makkelijk

Elk C-programma heeft hetzelfde begin en einde. Ik heb een bestandje gemaakt waarin een C-programma staat dat niets doet. Ik begin met een kopietje van dat bestand en noem dat `cos_stap0.c`, zie listing 1.

```
1 #include <stdio.h>
2
3 /* © 2015 Harry Broeders */
4
5 int main(void)
6 {
7
8     return 0;
9 }
```

Listing 1: Stap 0: `cos_stap0.c`.

Stap 2: Invoer

Het is altijd handig om met de invoer te beginnen. Ik begin dus met het inlezen van `x`, het floating point getal waar ik de cosinus van wil benaderen. Omdat ik dit zo nauwkeurig mogelijk wil doen definieer ik de variabele `x` van het type `double`. Zie eventueel hier. Ik herinner mij dat ik een variabele van het type `double` kan inlezen met behulp van `scanf`. Voordat ik de waarde van `x` inlees wil ik de gebruiker eerst vertellen wat er van hem/haar verwacht wordt door op het scherm te zetten: `Geef x: `, dat kan ik doen met behulp van `printf`. Om te kijken of het inlezen is gelukt druk ik de waarde van `x` af. Zie listing 2.

Na elke stap compileer en run ik het programma om te testen of de code die ik heb toegevoegd goed werkt. Het voordeel van deze manier van werken is dat ik fouten meteen ontdek en ze dan ook meteen kan corrigeren.

```c
#include <stdio.h>

int main(void)
{
    double x;

    printf("Geef x: ");
    scanf("%lf", &x);

    printf("cos(%f) = %f\n", x, 0.0); /* dit is nog niet het goede antwoord! */

    return 0;
}
```

Listing 2: Stap 1: cos_stap1.c.

Als ik het programma uit listing 2 uitvoer verschijnt de uitvoer die gegeven is in figuur 1 op mijn scherm. De door mij ingevoerde waarde is groen en onderstreept weergegeven.

```
Geef x: -1.32
cos(-1.320000) = 0.000000
```

Figuur 1: Uitvoer van het programma uit listing 2.

Het aantal termen dat gebruikt moet worden om de cosinus te benaderen is variabel en moet door het programma worden ingelezen. Ik bedenk me dat het aantal termen groter dan nul moet zijn en dat het netjes is om te controleren of het ingevoerde getal aan deze voorwaarde voldoet. Wat moet er gebeuren als de gebruiker een getal invoert dat kleiner dan of gelijk aan nul is? Ik besluit om de gebruiker opnieuw te vragen om de waarde in te voeren (zonder foutmelding) en de ingevoerde waarde opnieuw in te lezen. Ik wil dus het programmadeel dat gegeven is in listing 3 herhalen totdat een geldige waarde is ingevoerd.

Ik herinner me dat er verschillende herhalingsinstructies zijn (`for`, `do while` en `while`). Het programmadeel uit listing 3 moet minstens 1x herhaald worden en

```
        printf("Geef het aantal termen: ");
        scanf("%d", &aantalTermen);
```

Listing 3: Programmadeel dat herhaald moet worden zolang een waarde kleiner dan of gelijk aan nul wordt ingevoerd.

het is afhankelijk van de gebruiker hoe vaak het herhaald moet worden. Het programmadeel moet dus minstens 1x herhaald worden, maar het aantal herhalingen is verder niet bekend. Om die reden moet ik kiezen voor een `do while`-lus. Zie listing 4.

```
#include <stdio.h>

int main(void)
{
    double x;
    int aantalTermen;

    printf("Geef x: ");
    scanf("%lf", &x);

    do
    {
        printf("Geef het aantal termen: ");
        scanf("%d", &aantalTermen);
    }
    while (aantalTermen <= 0);

    printf("cos(%f) benaderd met %d termen = %f\n", x, ↩
     ↪ aantalTermen, 0.0); /* dit is nog niet het goede ↩
     ↪ antwoord! */

    return 0;
}
```

Listing 4: Stap 2: `cos_stap2.c`.

Bij het testen probeer ik eerst een aantal ongeldige waarden en constateer dat het programma correct werkt, zie figuur 2.

```
Geef x: -1.397
Geef het aantal termen: -3
Geef het aantal termen: 0
Geef het aantal termen: 5
cos(-1.397000) benaderd met 5 termen = 0.000000
```

Figuur 2: Uitvoer van het programma uit listing 4.

Stap 3. Nummer de termen

De waarde van `aantalTermen` wordt nu dus succesvol ingelezen en op de juiste wijze gecontroleerd. Nu moet ik de termen van de benadering gaan berekenen. Dit is voor een beginnend programmeur misschien te moeilijk om in één stap uit te voeren. Ik bedenk dat het niet zo moeilijk is om de termen te nummeren: 0, 1, 2, 3, 4, ... Dit nummer kan ik dan later gebruiken om de verschillende onderdelen van de termen te berekenen. Ik maak een variabele `termNummer` en die krijgt achtereenvolgens (in een herhaling) de waarde: 0, 1, 2, 3, 4, ... Het aantal herhalingen is bekend als de herhaling begint (want de waarde van `aantalTermen` is dan al ingelezen). Omdat het aantal herhalingen bekend is moet ik kiezen voor een `for`-lus. Om te testen druk ik de waarde van de variabele `termNummer` voor elke term af. Zie listing 5.

Bij het testen blijkt dat het programma correct werk, zie figuur 3.

```
Geef x: -1.39
Geef het aantal termen: 6
Test: termNummer = 0
Test: termNummer = 1
Test: termNummer = 2
Test: termNummer = 3
Test: termNummer = 4
Test: termNummer = 5
cos(-1.390000) benaderd met 6 termen = 0.000000
```

Figuur 3: Uitvoer van het programma uit listing 5.

```c
#include <stdio.h>

int main(void)
{
    double x;
    int aantalTermen;

    printf("Geef x: ");
    scanf("%lf", &x);

    do
    {
        printf("Geef het aantal termen: ");
        scanf("%d", &aantalTermen);
    }
    while (aantalTermen <= 0);

    for (int termNummer = 0; termNummer < aantalTermen; ↩
     ↪ termNummer = termNummer + 1)
    {
        printf("Test: termNummer = %d\n", termNummer);
    }

    printf("cos(%f) benaderd met %d termen = %f\n", x, ↩
     ↪ aantalTermen, 0.0); /* dit is nog niet het goede ↩
     ↪ antwoord! */

    return 0;
}
```

Listing 5: Stap 3: cos_stap3.c.

Stap 4. Bepaal de constante waarde van elke term

Elke term gebruikt een bepaalde constante waarde. Term nummer 0 gebruikt de constante waarde 0, term nummer 1 gebruikt de constante waarde 2, term nummer 2 gebruikt de constante waarde 4, enzovoort. Deze constante waarde noem ik `termConstante` en deze is eenvoudig te berekenen voor elke term met de formule: `termConstante = termNummer * 2;`. Om te testen druk ik de waarde van de variabele `termConstante` voor elke term af. Zie listing 6.

Bij het testen blijkt dat het programma correct werk, zie figuur 4.

```
Geef x: -1.39
Geef het aantal termen: 6
Test: termConstante = 0
Test: termConstante = 2
Test: termConstante = 4
Test: termConstante = 6
Test: termConstante = 8
Test: termConstante = 10
cos(-1.390000) benaderd met 6 termen = 0.000000
```

Figuur 4: Uitvoer van het programma uit listing 6.

Stap 5. Bepaal de teller van elke term

Elke term bestaat uit een breuk. Elke breuk heeft een teller (het gedeelte boven de breukstreep) en een noemer (het gedeelte onder de breukstreep). De tellers van de termen zijn x^0, x^2, x^4, x^6, ... Elke term heeft dus als teller $x^{\texttt{termConstante}}$. Deze waarde kan berekend worden met de standaard functie `pow`. Om deze functie te kunnen gebruiken moet ik `math.h` includen. Om te testen druk ik de waarde van de teller (variabele `termTeller`) voor elke term af. Zie listing 7.

Bij het testen blijkt dat het programma correct werk, zie figuur 5.

Op zich is dit een goede oplossing maar ik vraag me af of ik de benadering van de cosinus ook kan bereken zonder gebruik te maken van de C math library. Ik

```c
#include <stdio.h>

int main(void)
{
    double x;
    int aantalTermen;

    printf("Geef x: ");
    scanf("%lf", &x);

    do
    {
        printf("Geef het aantal termen: ");
        scanf("%d", &aantalTermen);
    }
    while (aantalTermen <= 0);

    for (int termNummer = 0; termNummer < aantalTermen; ↩
     ↪ termNummer = termNummer + 1)
    {
        int termConstante = termNummer * 2;
        printf("Test: termConstante = %d\n", termConstante);
    }

    printf("cos(%f) benaderd met %d termen = %f\n", x, ↩
     ↪ aantalTermen, 0.0); /* dit is nog niet het goede ↩
     ↪ antwoord! */

    return 0;
}
```

Listing 6: Stap 4: cos_stap4.c.

```c
#include <stdio.h>
#include <math.h>

int main(void)
{
    double x;
    int aantalTermen;

    printf("Geef x: ");
    scanf("%lf", &x);

    do
    {
        printf("Geef het aantal termen: ");
        scanf("%d", &aantalTermen);
    }
    while (aantalTermen <= 0);

    for (int termNummer = 0; termNummer < aantalTermen; ↩
     ↪ termNummer = termNummer + 1)
    {
        int termConstante = termNummer * 2;
        double termTeller = pow(x, termConstante);
        printf("Test: termTeller = %f\n", termTeller);
    }

    printf("cos(%f) benaderd met %d termen = %f\n", x, ↩
     ↪ aantalTermen, 0.0); /* dit is nog niet het goede ↩
     ↪ antwoord! */

    return 0;
}
```

Listing 7: Stap 5a: cos_stap5a.c.

```
Geef x: 2
Geef het aantal termen: 5
Test: termTeller = 1.000000
Test: termTeller = 4.000000
Test: termTeller = 16.000000
Test: termTeller = 64.000000
Test: termTeller = 256.000000
cos(2.000000) benaderd met 5 termen = 0.000000
```

Figuur 5: Uitvoer van het programma uit listing 7.

kijk nog eens goed naar de tellers van de termen: x^0, x^2, x^4, x^6, x^8 .... Deze termen kan ik ook als volgt schrijven: x^0, $x^0 \cdot x^2$, $x^2 \cdot x^2$, $x^4 \cdot x^2$, $x^6 \cdot x^2$, ... Als ik een teller heb berekend dan kan ik de volgende teller berekenen door de laatst berekende teller te vermenigvuldigen met x^2. Als ik x^2 uitrekenen als $x \cdot x$ dan heb ik met deze aanpak de pow-functie niet meer nodig. Zie listing 8. De waarde van x^n groeit snel voor grote waarden van x. De waarde van $10^10!$ past al niet meer in een variabele van het type `int` (een 32 bits two's complement getal). Om deze reden definiëren we de variabele `termTeller` van het type `double`.

Bij het testen blijkt dat het programma (nog steeds) correct werk, zie figuur 6.

```
Geef x: 2
Geef het aantal termen: 5
Test: termTeller = 1.000000
Test: termTeller = 4.000000
Test: termTeller = 16.000000
Test: termTeller = 64.000000
Test: termTeller = 256.000000
cos(2.000000) benaderd met 5 termen = 0.000000
```

Figuur 6: Uitvoer van het programma uit listing 8.

```c
#include <stdio.h>

int main(void)
{
    double x;
    int aantalTermen;

    printf("Geef x: ");
    scanf("%lf", &x);

    do
    {
        printf("Geef het aantal termen: ");
        scanf("%d", &aantalTermen);
    }
    while (aantalTermen <= 0);

    double termTeller = 1;
    for (int termNummer = 0; termNummer < aantalTermen; ↩
     ↪ termNummer = termNummer + 1)
    {
        int termConstante = termNummer * 2;
        printf("Test: termTeller = %f\n", termTeller);
        /* bereken volgende termTeller */
        termTeller = termTeller * x * x;
    }

    printf("cos(%f) benaderd met %d termen = %f\n", x, ↩
     ↪ aantalTermen, 0.0); /* dit is nog niet het goede ↩
     ↪ antwoord! */

    return 0;
}
```

Listing 8: Stap 5b: cos_stap5b.c.

Bepaal de noemer van elke term

Elke term bestaat uit een breuk. Elke breuk heeft een teller (het gedeelte boven de breukstreep) en een noemer (het gedeelte onder de breukstreep). De noemers van de termen zijn 0!, 2!, 4!, 6!, ... Elke term heeft dus als noemer `termConstante`! (de faculteit van de `termConstante`). De waarde van $n!$ kan berekend worden als $1 \cdot 2 \cdot 3 \cdot 4 \cdot \cdots \cdot n$. Omdat het aantal vermenigvuldigingen voor een bepaalde waarde van n bekend is (namelijk n) kun je een eenvoudige `for`-lus gebruiken om $n!$ uit te rekenen, zie listing 9.

```
1     n_fac = 1;
2     for (i = 2; i <= n; i = i + 1) {
3         n_fac = n_fac * i;
4     }
```

Listing 9: Het berekenen van $n!$.

De waarde van $n!$ groeit heel snel. De waarde van 13! past al niet meer in een variabele van het type `int` (een 32 bits two's complement getal). Om deze reden definiëren we de variabele `termNoemer` van het type `double`. Om te testen druk ik de waarde van de noemer (variabele `termNoemer`) voor elke term af. Zie listing 10.

Bij het testen blijkt dat het programma correct werk, zie figuur 7.

```
Geef x: 1.52
Geef het aantal termen: 5
Test: termNoemer = 1.000000
Test: termNoemer = 2.000000
Test: termNoemer = 24.000000
Test: termNoemer = 720.000000
Test: termNoemer = 40320.000000
cos(2.000000) benaderd met 5 termen = 0.000000
```

Figuur 7: Uitvoer van het programma uit listing 10.

Op zich is dit een goede oplossing maar ik vraag me af of ik de waarde van `termNoemer` (= `termConstante`!) tekens opnieuw uit moet rekenen. Ik kijk nog eens goed naar de noemers van de termen: 0!, 2!, 4!, 6!, 8! .... Deze termen

```c
#include <stdio.h>

int main(void)
{
    double x;
    int aantalTermen;

    printf("Geef x: ");
    scanf("%lf", &x);

    do
    {
        printf("Geef het aantal termen: ");
        scanf("%d", &aantalTermen);
    }
    while (aantalTermen <= 0);

    double termTeller = 1;
    for (int termNummer = 0; termNummer < aantalTermen; ↩
     ↪ termNummer = termNummer + 1)
    {
        int termConstante = termNummer * 2;
        double termNoemer = 1;
        for (int i = 2; i <= termConstante; i = i + 1)
        {
            termNoemer = termNoemer * i;
        }
        printf("Test: termNoemer = %f\n", termNoemer);
        /* bereken volgende termTeller */
        termTeller = termTeller * x * x;
    }

    printf("cos(%f) benaderd met %d termen = %f\n", x, ↩
     ↪ aantalTermen, 0.0); /* dit is nog niet het goede ↩
     ↪ antwoord! */

    return 0;
}
```

Listing 10: Stap 6a: cos_stap6a.c.

kan ik ook als volgt schrijven: $0!$, $0! \cdot 1 \cdot 2$, $2! \cdot 3 \cdot 4$, $4! \cdot 5 \cdot 6$, $6! \cdot 7 \cdot 8$, ... Als ik een noemer heb berekend dan kan ik de volgende noemer berekenen door de laatst berekende noemer (met de waarde `termConstante`!) te vermenigvuldigen met $(\texttt{termConstante} + 1) \cdot (\texttt{termConstante} + 2)$. Zie listing 11.

Bij het testen blijkt dat het programma (nog steeds) correct werk, zie figuur 8.

```
Geef x: 1.52
Geef het aantal termen: 5
Test: termNoemer = 1.000000
Test: termNoemer = 2.000000
Test: termNoemer = 24.000000
Test: termNoemer = 720.000000
Test: termNoemer = 40320.000000
cos(2.000000) benaderd met 5 termen = 0.000000
```

Figuur 8: Uitvoer van het programma uit listing 11.

Stap 7. Bepaal het teken van elke term

Elke term heeft een teken. Het teken is afwisselend positief en negatief dus: $+1$, -1, $+1$, -1, $+1$, ... Als je dus het teken van een term kent dan kun je het teken van de volgende term berekenen door het bekende teken te vermenigvuldigen met -1. Om te testen druk ik het teken (variabele `termTeken`) voor elke term af. Zie listing 12.

Bij het testen blijkt dat het programma correct werk, zie figuur 9.

Stap 8. Bepaal elke term

We hebben nu de teller, de noemer en het teken van elke term berekend. Het is nu eenvoudig om de waarde van elke term te berekenen: `term = termTeken * termTeller / termNoemer`. Zie listing 13.

```c
#include <stdio.h>

int main(void)
{
    double x;
    int aantalTermen;

    printf("Geef x: ");
    scanf("%lf", &x);

    do
    {
        printf("Geef het aantal termen: ");
        scanf("%d", &aantalTermen);
    }
    while (aantalTermen <= 0);

    double termTeller = 1;
    double termNoemer = 1;
    for (int termNummer = 0; termNummer < aantalTermen; ↩
     ↪ termNummer = termNummer + 1)
    {
        int termConstante = termNummer * 2;
        printf("Test: termNoemer = %f\n", termNoemer);
        /* bereken volgende termTeller */
        termTeller = termTeller * x * x;
        /* bereken volgende termNoemer */
        termNoemer = termNoemer * (termConstante + 1) * ↩
         ↪ (termConstante + 2);
    }

    printf("cos(%f) benaderd met %d termen = %f\n", x, ↩
     ↪ aantalTermen, 0.0); /* dit is nog niet het goede ↩
     ↪ antwoord! */

    return 0;
}
```

Listing 11: Stap 6b: cos_stap6b.c.

```c
#include <stdio.h>

int main(void)
{
    double x;
    int aantalTermen;

    printf("Geef x: ");
    scanf("%lf", &x);

    do
    {
        printf("Geef het aantal termen: ");
        scanf("%d", &aantalTermen);
    }
    while (aantalTermen <= 0);

    double termTeller = 1;
    double termNoemer = 1;
    int termTeken = 1;
    for (int termNummer = 0; termNummer < aantalTermen; termNummer = termNummer + 1)
    {
        int termConstante = termNummer * 2;
        printf("Test: termTeken = %d\n", termTeken);
        /* bereken volgende termTeller */
        termTeller = termTeller * x * x;
        /* bereken volgende termNoemer */
        termNoemer = termNoemer * (termConstante + 1) * (termConstante + 2);
        /* bereken volgende termTeken */
        termTeken = termTeken * -1;
    }

    printf("cos(%f) benaderd met %d termen = %f\n", x, aantalTermen, 0.0); /* dit is nog niet het goede antwoord! */

    return 0;
}
```

Listing 12: Stap 7: cos_stap7.c.

```c
#include <stdio.h>

int main(void)
{
    double x;
    int aantalTermen;

    printf("Geef x: ");
    scanf("%lf", &x);

    do
    {
        printf("Geef het aantal termen: ");
        scanf("%d", &aantalTermen);
    }
    while (aantalTermen <= 0);

    double termTeller = 1;
    double termNoemer = 1;
    int termTeken = 1;
    for (int termNummer = 0; termNummer < aantalTermen; ↩
     ↪ termNummer = termNummer + 1)
    {
        int termConstante = termNummer * 2;
        double term = termTeken * termTeller / termNoemer;
        printf("Test: term = %f\n", term);
        /* bereken volgende termTeller */
        termTeller = termTeller * x * x;
        /* bereken volgende termNoemer */
        termNoemer = termNoemer * (termConstante + 1) * ↩
         ↪ (termConstante + 2);
        /* bereken volgende termTeken */
        termTeken = termTeken * -1;
    }

    printf("cos(%f) benaderd met %d termen = %f\n", x, ↩
     ↪ aantalTermen, 0.0); /* dit is nog niet het goede ↩
     ↪ antwoord! */
    return 0;
}
```

Listing 13: Stap 8: cos_stap8.c.

```
Geef x: -1.23
Geef het aantal termen: 7
Test: termTeken = 1
Test: termTeken = -1
Test: termTeken = 1
Test: termTeken = -1
Test: termTeken = 1
Test: termTeken = -1
Test: termTeken = 1
cos(-1.230000) benaderd met 7 termen = 0.000000
```

Figuur 9: Uitvoer van het programma uit listing 12.

Bij het testen blijkt dat het programma de eerste 4 termen van cos(1) correct berekent (ik heb de uitkomsten vergeleken met de in stap 0 met behulp van de rekenmachine berekende waarden, zie vergelijkingen (3) tot (6)), zie figuur 10.

```
Geef x: 1
Geef het aantal termen: 4
Test: term = 1.000000
Test: term = -0.500000
Test: term = 0.041667
Test: term = -0.001389
cos(1.000000) benaderd met 4 termen = 0.000000
```

Figuur 10: Uitvoer van het programma uit listing 13.

Stap 9. Bepaal de som van alle termen

We kunnen nu de benadering van de cosinus eenvoudig berekenen door alle berekende termen bij elkaar op te tellen. Zie listing 14.

Bij het testen blijkt dat het programma de eerste 4 termen van cos(1) correct optelt (ik heb de uitkomsten vergeleken met de in stap 0 met behulp van de rekenmachine berekende waarden, zie vergelijking (7)), zie figuur 11.

```c
#include <stdio.h>

int main(void)
{
    double x;
    int aantalTermen;

    printf("Geef x: ");
    scanf("%lf", &x);

    do
    {
        printf("Geef het aantal termen: ");
        scanf("%d", &aantalTermen);
    }
    while (aantalTermen <= 0);

    double termTeller = 1;
    double termNoemer = 1;
    int termTeken = 1;
    double cos = 0;
    for (int termNummer = 0; termNummer < aantalTermen; ↩
     ↪ termNummer = termNummer + 1)
    {
        int termConstante = termNummer * 2;
        double term = termTeken * termTeller / termNoemer;
        cos = cos + term;
        /* bereken volgende termTeller */
        termTeller = termTeller * x * x;
        /* bereken volgende termNoemer */
        termNoemer = termNoemer * (termConstante + 1) * ↩
         ↪ (termConstante + 2);
        /* bereken volgende termTeken */
        termTeken = termTeken * -1;
    }

    printf("cos(%f) benaderd met %d termen = %f\n", x, ↩
     ↪ aantalTermen, cos);
    return 0;
}
```

Listing 14: Stap 9: cos_stap9.c.

```
Geef x: 1
Geef het aantal termen: 4
cos(1.000000) benaderd met 4 termen = 0.540278
```

Figuur 11: Uitvoer van het programma uit listing 14.

Stap 10. Een laatste verbetering

Als ik nog eens goed naar het bovenstaande programma kijk, dan zie ik dat het helemaal niet zo handig is om term 0 in de `for`-lus uit te rekenen. Het is veel eenvoudiger om gewoon te beginnen met term 0 (die altijd de waarde 1 heeft). Verder druk ik het antwoord met 15 decimalen na de punt af zodat ik kan zien of de benadering nauwkeurig genoeg is. Een `double` heeft minimaal 15 en maximaal 17 significante cijfers dus meer cijfers achter de punt afdrukken is zinloos, zie eventueel hier. Uiteindelijk kom ik dus tot het programma dat gegeven is in listing 15.

Bij het testen blijkt dat het programma de cosinus correct benaderd. De waarde van cos(0.5) is 0.8775825618903727 volgens mijn rekenmachine. Als we cos(0.5) met 6 termen berekenen dan is het antwoord tot op 10 cijfers achter de punt correct. Als we 7 termen gebruiken dan zijn alle afgedrukte 15 cijfers achter de punt correct, zie figuur 12.

```
Geef x: 0.5
Geef het aantal termen: 6
cos(0.500000) benaderd met 6 termen = 0.877582561889864

Geef x: 0.5
Geef het aantal termen: 7
cos(0.500000) benaderd met 7 termen = 0.877582561890373
```

Figuur 12: Uitvoer van het programma uit listing 15.

```c
#include <stdio.h>

int main(void)
{
    double x;
    int aantalTermen;

    printf("Geef x: ");
    scanf("%lf", &x);

    do
    {
        printf("Geef het aantal termen: ");
        scanf("%d", &aantalTermen);
    }
    while (aantalTermen <= 0);

    double termTeller = 1;
    double termNoemer = 1;
    int termTeken = 1;
    double cos = 1;
    for (int termNummer = 1; termNummer < aantalTermen; termNummer = termNummer + 1)
    {
        int termConstante = termNummer * 2;
        termTeller = termTeller * x * x;
        termNoemer = termNoemer * (termConstante - 1) * termConstante;
        termTeken = termTeken * -1;
        double term = termTeken * termTeller / termNoemer;
        cos = cos + term;
    }

    printf("cos(%f) benaderd met %d termen = %.15f\n", x, aantalTermen, cos);

    return 0;
}
```

Listing 15: Stap 10: cos_stap10.c.