Inleiding programmeren natuurkundigen

Ragnar Groot Koerkamp

11 juni 2015

Contents

1 Overzicht

In deze cursus leer je de basis van programmeren in C++. We beginnen we bij het opstarten van een project in een tekstverwerker. Daarna gaan we in op de structuur van een programma en wat de regels zijn bij het schrijven ervan. Vervolgens wordt aan de hand van veel voorbeelden en simpele opdrachten de basis van C++zelf uitgelegd. We gaan hierbij bijvoorbeeld in op variabelen, `if`-statements en `for`-loops.
Na afloop van deze cursus zou je van onderstaande code moeten kunnen achterhalen wat het doet en hoe het werkt:[1]

```
1 #include <iostream>
2 using namespace std;
3
4 int main(){
```

[1] of gewoon meteen hier uitleggen?

```
5
6       int aantal;
7       cout << "Voer het aantal personen in:" << endl;
8       cin >> aantal;
9
10      int som = 0;
11      int kinderen = 0;
12
13      for(int i=0; i<aantal; i++){
14          int leeftijd;
15          cout << "Wat is je leeftijd?" << endl;
16          cin >> leeftijd;
17          cout << "Dan ben je al meer dan ";
18          cout << 356 * leeftijd;
19          cout << " dagen oud!" << endl;
20
21          som += leeftijd;
22          if(leeftijd < 18)
23              kinderen++;
24      }
25
26      cout << "De totale leeftijd is " << som << " jaar!" << endl;
27      cout << "Er zijn " << kinderen << " kinderen"<<endl;
28
29      return 0;
30  }
```

2 Visual studio/ of online?

Voor Manon Hier ook iets over compileren

3 Basis

We bekijken nu eerst hoe een programma globaal in elkaar zit. Elk programma heeft een `main` functie waar het altijd begint. Vandaag staat alle code in de main functie, dus tussen de accolades die aangeven welke code bij de `main` hoort. Bovenaan het programma zie je groene regels commentaar staan die beginnen met `//`. Deze regels worden altijd overgeslagen bij het uitvoeren. Het commentaar begint pas bij de `//`, dus ervoor kan nog wel code staan. Het is handig om commentaar bij de code te zetten om te verduidelijken wat de code doet. Dat is handig als je ergens later weer naar terug kijkt en dan niet opnieuw hoeft uit te zoeken wat de code nou eigenlijk deed. Commentaar kan ook meer dan een regel omvatting. Dan gebruik je aan het begin `/*` en aan het einde `*/`.

```
1  // dit pakketje is nodig om straks
```

```
2  // naar de console te kunnen schrijven
3  #include <iostream>
4
5  // en deze regel maakt die functies alvast
6  // toegankelijk voor de rest van het programma
7  using namespace std;
8
9  /*
10  * Het programma begint in de ‘main‘.
11  * Dit is de functie die wordt aangeroepen
12  * zodra je programmma begint.
13  * De main moet altijd een geheeltallige waarde
14  * retouneren, en meestal 0.
15  * Elk getal ongelijk aan 0 betekent dat er een error is geweest.
16  */
17 int main(){
18     return 0; // no error
19 }
```

Zoals je misschien al is opgevallen eindigt bijna elke regel van het voorbeeld met een puntkomma (;). Dit is nodig zodat het programma dat de code uitvoert weet wanneer een nieuwe regel met nieuwe code begint. In sommige gevallen zijn geen puntkomma's nodig, zoals voor of na een accolade.
In C++mag je overal zo veel witruimte (spaties, tabs, enters) neerzetten als je maar wilt. De enige eis is dat sommige dingen niet direct aan elkaar geschreven morgen worden, en er dus minstens een spatie tussen moet zitten.
Verder zie je ook nog `#include <iostream>` staan. `<iostream>` is een pakketje dat we later nodig zullen hebben om tekst te schrijven naar de console (dat zwarte schermpje, zie verderop) en om data in te lezen. Alle functies in dat pakketje behoren tot de C++-standaard-library[2]. Om ze toegankelijk te maken moeten we daarom ook `using namespace std;` bovenaan onze code zetten.
Het programma eindigt met `return 0;`[3]. Het getal dat achter `return` staat, staat voor een mogelijke error die het programma geeft. 0 betekent hier dat er geen errors zijn geweest.

3.1 Opdrachten

1. Type de code uit het voorbeeld over en start het programma, maar zonder het commentaar.[4] Zorg ervoor dat de console niet meteen weer uit beeld verdwijnt, maar daar hij blijft staan en pas weg gaat als je op een toets drukt.

[2] beter kan ik het niet maken. Suggesties?
[3] of dit gewoon weglaten?
[4] dat doe je door ... te doen, afhankelijk IDE.

2. Voeg nu ook commentaar toe op de plekken waar dat in het voorbeeld ook is gedaan. Check dat het programma nog steeds werkt.

3. Probeer nu ook op binnen `main` je eigen commentaar toe te voegen van beide soorten.

4. Voeg nu een blok code (dus van de soort `/* */`) toe binnen een code regel, dus bijvoorbeeld tussen `int` en `main(){` of tussen `return` en `0;`. Omdat witruimte niet van belang is en het commentaar wordt weggegooid, zou je programma nog steeds moeten werken.

4 Console uitvoer

Wanneer je het programma start, zie je steeds een zwart scherm verschijnen. Dat heet de *console* of *terminal*. Daar kunnen we tekst neerzetten of juist van lezen. Het neerzetten van tekst kan met het commando `cout`, oftewel, Console-out. Je kan achter de `cout <<` bijvoorbeeld een string of een getal zetten. Ook is er het speciale commando `endl`, dat voor *endline* staat. Hiermee beëindig je dus een regel. Het is handig om te onthouden dat je de tekst 'in de console stopt' en de pijltjes dus richting `cout` staan.

```
#include <iostream>
using namespace std;

int main(){
    // print 'Hello world!' to the console
    // together with a newline character
    cout << "Hello world!" << endl;
}
```

5 Variabelen

Er zijn meerdere soorten variabelen. De belangrijkste zijn getallen en strings. Een getal kan of geheel zijn, of reëel. In het eerste geval gebruiken we een `int`, wat staat voor *integer*. Als een getal niet altijd geheel is, hebben we een `double` nodig. Een double kan elke mogelijke waarde aannemen, dus bijvoorbeeld ook π.
Een `string` is een stuk tekst. In computertaal is dat gewoon een rij van tekens.

```
#include <iostream>
#include <string> // nodig voor strings
using namespace std;

int main(){
    /*
```

```
7       * Weinig inspiratie hier
8       * TODO:
9       * + - * /
10      * integer division
11      * (%)
12      */
13      int a = 10;
14      cout << "a = " << a << endl;
15      cout << "2*a = " << 2*a << endl;
16
17      int b = 3.5; // een int is altijd geheel
18      cout << "b = " << b << endl;
19
20      double c = 3.5; // een double kan elk getal zijn
21      cout << "c = " << c << endl;
22      cout << "c kwadraat = " << c*c << endl;
23
24      string s = "Hello world!";
25      cout << s << endl;
26
27      return 0; // no error
28  }
```

6 Console invoer

[5] Het is soms ook handig om juist invoer van de gebruiker te lezen. Daarvoor hebben we `cin`, console-in. Hieronder lezen we een getal in, wat we in de variabele (zie het volgende hoofdstuk) `leeftijd` stoppen. De pijltjes staan hier naar rechts, omdat het getal uit de console komt, en in de variabele wordt gestopt.

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      cout << "Hoe oud ben je?" << endl;
6      int leeftijd;
7      cin >> leeftijd;
8      cout << "Je bent " << leeftijd << " jaar oud" << endl;
9  }
```

[5] Ik heb nu maar uitvoer-variabelen-invoer gedaan, omdat er op die manier geen vooruit-referenties nodig zijn.

7 For

Als je een stukje code vaker dan één keer wilt uitvoeren, kan je hem natuurlijk een aantal keer onder elkaar kopiëren en plakken. Als je echter heel vaak (≥ 3 keer) iets wilt doen, is dat niet meer handig. Daarvoor is de `for`-loop. Deze herhaalt iets een bepaald aantal keer. Hieronder staat een voorbeeldje.

```
#include <iostream>
using namespace std;

int main(){
    cout << "drie hoeratjes:" << endl;
    for(int i = 0; i < 3; i++)
        cout << "hoera" << endl;

    cout << "tellen tot 10:" << endl;
    for(int getal = 0; getal <= 10; getal++)
        cout << getal << endl;

    return 0;
}
```

Als je dit intikt en uitvoert, zie je dat er eerst drie keer 'hoera' wordt geschreven, en daarna de getallen van 0 tot en met 10. De werking van een `for`-loop is als volgt:

```
for(initialisatie; conditie; verhoging){
    code die herhaalt word
}
```

In de `initialisatie` wordt meestal een variabele geïnitialiseerd, zoals hier `int i=0`. In de `conditie` kijken we steeds of we moeten doorgaan met de volgende iteratie, of dat we moeten stoppen met herhalen. Zolang de conditie waar is gaan we door, en als hij niet meer waar is stoppen we met herhalen. Hier gaan we dus alleen maar door wanneer `i` kleiner of gelijk is dan 3. In de `verhoging` verhogen we de variabele waarover we lopen. Meestal verhogen we hem steeds met 1 tegelijk maar dat hoeft niet per se.

8 If

Tot nu toe lag het altijd precies vast welke code werd uitgevoerd en hoe vaak dat gebeurde. Het is ook mogelijk om iets alleen uit te voeren als aan een bepaalde voorwaarde is voldaan. Daarvoor gebruiken we het `if`-statement. De syntax is

```
if(conditie){
    code wanneer conditie waar (true) is;
} else {
```

```
4       code wanneer conditie niet waar (false) is;
5   }
```

Dit kunnen we bijvoorbeeld gebruiken in het volgende voorbeeld, waarin de uitvoer afhangt van de invoer van de gebruiker.

```
1   #include <iostream>
2   using namespace std;
3
4   int main(){
5       cout << "voer je leeftijd in:" << endl;
6       int leeftijd;
7       cin >> leeftijd;
8       if(leeftijd >= 18){
9           cout << "Je bent volwassen!" << endl;
10      } else {
11          cout << "Je bent nog een kind!" << endl;
12      }
13
14      return 0;
15  }
```

9 While

Tot slot hebben we nog de `while`-loop. Deze lijkt een beetje op de `for`-loop, maar is toch anders. Bij een `for`-loop weten we meestal van tevoren al hoe vaak we de code willen herhalen. Als we dat niet precies weten, is een `while`-loop handig. We herhalen de code nu net zo lang tot `conditie` onwaar wordt in

```
1   while(conditie){
2       code die herhaald wordt;
3   }
```

In de code hieronder zie je een voorbeeld. We beginnen hier met $n = 0$, en vervolgens schrijven we steeds n^2 op, net zo lang tot n^2 groter wordt dan 1000.

```
1   #include <iostream>
2   using namespace std;
3
4   int main(){
5       cout << "alle kwadraten onder 1000" << endl;
6
7       int n = 0;
8       while(n * n <= 1000){
9           cout << (n * n) << endl;
10          n = n+1;
11      }
12
```

```
13      return 0;
14  }
```

10 Arrays [optioneel]

```
1   #include <iostream>
2   using namespace std;
3
4   int main(){
5
6       int kwadraten[10];
7
8       for(int i = 0; i < 10; i++){
9           kwadraten[i] = i*i;
10      }
11
12      int som = 0;
13      for(int i = 0; i < 10; i++){
14          som += kwadraten[i];
15      }
16
17      cout << "de som is " << som << endl;
18
19      return 0;
20  }
```