Introductie C#

door Martine Vrijhof

December 2022

Dit is een introductie naar C# voor wiskundestudenten die bekend zijn met Python en C# willen leren.

Contents

Figure 1: Opstarten van Visual Studio 2022

1 Installeren van Visual Studio en eerste gebruik

1.a Installatie

1. Ga naar `https://visualstudio.microsoft.com/downloads/`.
2. Download de 'Visual Studio Community' versie (gratis voor studenten)
3. Start het installatieprogramma
4. Kruis tijdens de installatie aan dat je *.NET desktop development* nodig hebt
5. Als je Visual Studio voor het eerst opstart, dan zal het programma vragen wat je standaardontwikkeltaal is. Kies hier voor 'Visual C# Development Settings'. Deze optie zal de ontwikkelomgeving optimaal instellen voor het gebruik van de taal C#. Zodra dit gedaan is, zie je het opstartscherm van Visual Studio.

bron: dictaat Imperatief programmeren - Bijlage A - Jeroen Fokker 2018

1.b Opstarten

Bij het openen van Visual Studios zie je het volgende scherm Om een nieuw project te

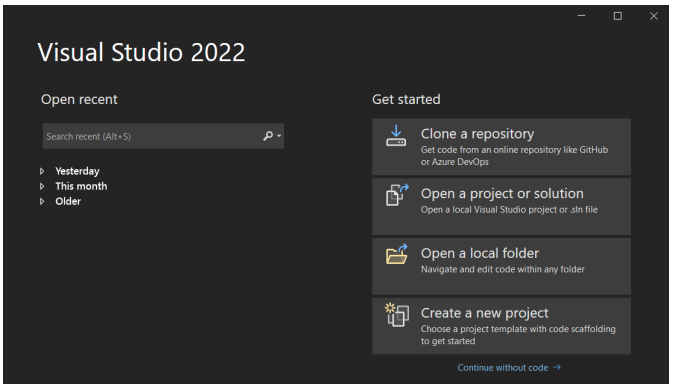

Figure 2: Beginscherm

maken, kies je voor de onderste optie waarna je op het volgende scherm uitkomt: Aan de rechterkant van het scherm zijn er veel verschillende opties waaruit je kan kiezen. Hier kies je voor de optie *Console App (.NET Framework)*. Vervolgens geef je het project een naam en kan je een bestandslocatie kiezen. Het veld met de *Solution name* heeft altijd dezelfde naam als je project. Onder het kopje *Framework* kies je de meest recente versie, dus in dit geval *.NET Framework 4.8* (zie Figuur 4). Als het project is gemaakt, zie je het programma zoals in Figuur 5. Het belangrijkste onderdeel is hier het bestand *program.cs*

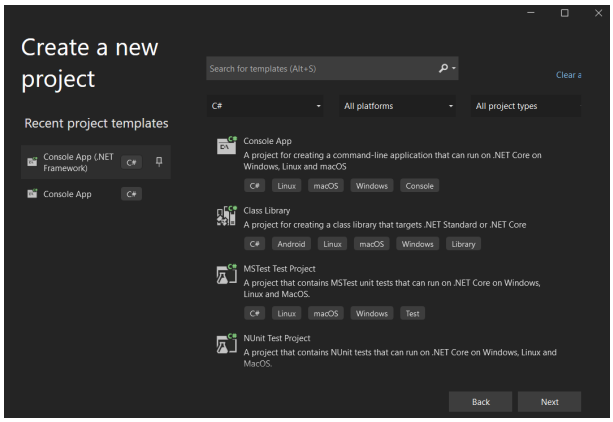

Figure 3: Create a new project.

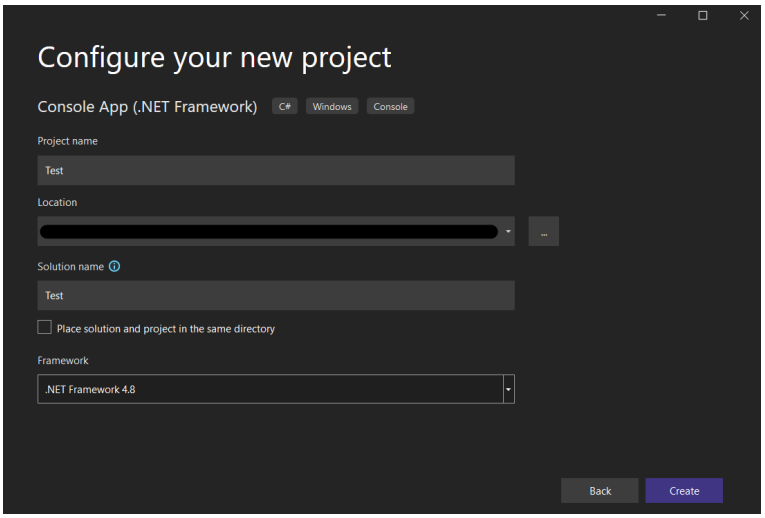

Figure 4: Bestandsnaam en -locatie kiezen.

dat geopend staat. Dit is je script waar de code komt te staan en dit script zal dus gerunt worden, net als in Python. Het grote verschil met Python is wel, dat er geen live console is waar je het bestand in kan runnen. Het script in C# zal de console opstarten.

1.b.1 Het script

Libaries
Net als in Python wordt er in C# gebruik gemaakt van libaries met bepaalde methoden en objecten. In C# worden deze libaries geïmporteerd met de regel *using System;*.

Namespace en Class
De namespace is de titel die je aan het project hebt gegeven.
Zoals je in Python niet hoeft te werken in een klasse, moet dat in C# wel. Als je al wat meer weet over C# kan je er goed gebruik van maken, maar voor nu is alleen de *internal class Program* nodig. Deze kan je een andere naam geven, maar dat is niet nodig.

Methode Main()
Elk programma dat in C# geschreven wordt, heeft de methode *static void Main(string[] args)* nodig. Zonder deze methode zal je programma niet gerunt kunnen worden.

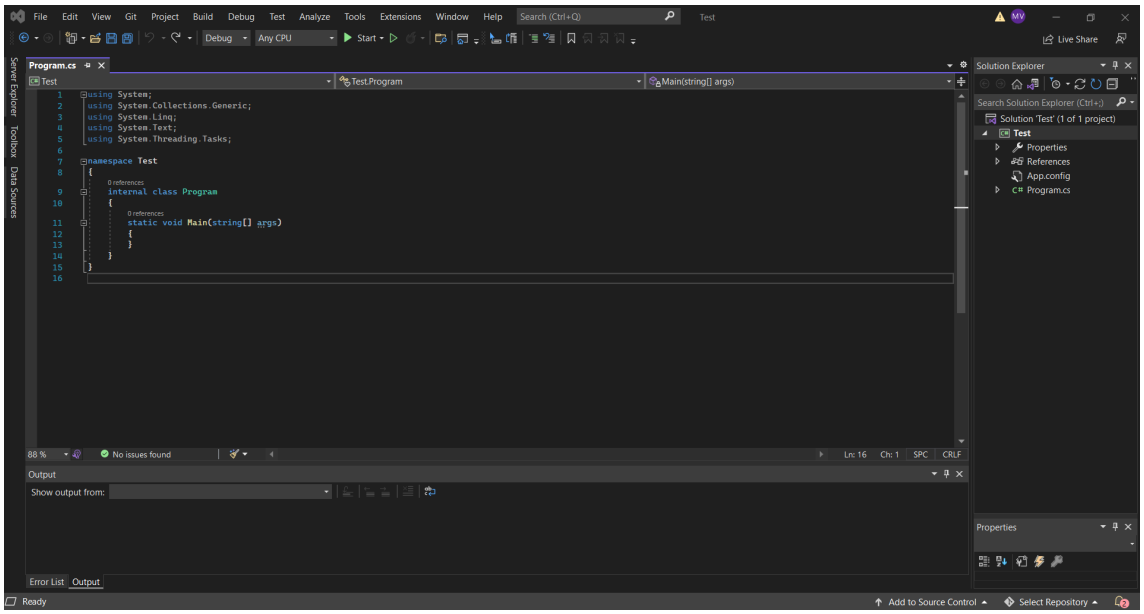

Figure 5: Het eerste project

Syntax

Zoals misschien al is opgevallen, staat alles wat in de namespace hoort tussen accolades, net als alles wat in de klasse hoort en als straks de methode geschreven wordt, zal dat ook tussen accolades staan. In Python werden methodes gescheiden door indents, maar in C# is het belangrijk om daar zelf voor te zorgen. Bij het importeren van de libaries eindigt de regel met een puntkomma (;). Dit hoort na (bijna elke) regel met code. Er zijn een paar uitzonderingen, zoals in het geval van loops.

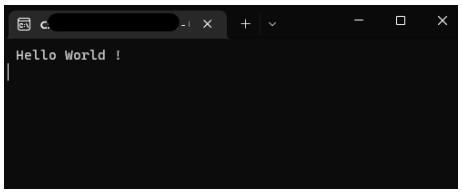

Figure 6: *Hello World!* als output in de console.

2 Hello World!

Nu kan het eerste programma geschreven worden: Het printen van de string "Hello World!" in de console. Hiervoor gebruik je de methode *Console.WriteLine()*. Dan ziet je programma er als volgt uit:

```
internal class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello World!");
    }
}
```

Door vervolgens op *Start* te klikken, midden boven in het scherm, zal het programma gerunt worden. Als het goed is zie je nu een console verschijnen, maar ook weer gelijk sluiten. Dit komt doordat het script, zodra het volledig gerunt is, de console gelijk afsluit. Als we de methode *Console.ReadLine()* aan het programma toevoegen, zal er gewacht worden op input van de 'gebruiker' die dan vervolgens op de Enter-toets drukt, voordat de console afgesloten wordt.

```
internal class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello World!");
        Console.ReadLine();
    }
}
```

Opmerking: Er zijn ook andere methodes om hetzelfde te bereiken als Console.ReadLine(), *bijvoorbeeld* Console.ReadKey().

3 Variabelen declareren

In C# is het nodig om alle variabelen te declareren. Dit houdt in dat je, voordat je een variabele gebruikt, aangeeft om wat voor variabele het gaat. In Python gebeurt dit automatisch, maar in C# dus niet.
Het volgende programma declareert eerst de variabelen en vervolgens geeft hij deze variabelen een waarde mee.

```
static void Main(string[] args)
{
    int x;
    int y;
    int z;

    x = 4;
    y = 9;
    z = x + y;

    Console.WriteLine(z);
    Console.ReadLine();
}
```

De waarde van een variabele kan ook direct toegekend worden als je hem declareert, zoals

```
static void Main(string[] args)
{
    string x = "Tom";
    string y = "Jerry";
    string z = x + " en " + y;

    Console.WriteLine(z);
    Console.ReadLine();
}
```

Type variabelen	
int	Gehele getallen
float	Kommagetallen
double	Kommagetallen
char	1 letter of teken
string	Letter- of tekenreeks

Table 1: Typen Variabelen

Om twee gehele getallen door elkaar te delen, wordt er gebruik gemaakt van de standaard /-operator. Als we x en y hebben gedeclareerd als gehele getallen, zal de regel *int* $z = 3/4$; de uitkomst 0 geven. Dit komt doordat 3 en 4 gehele getallen zijn, en dus de deling wordt uitgevoerd als integer-deling. In dat geval, als er een *float* uit de deling komt, wordt de ondergrens van dit getal genomen. Daarom zullen z en x en/of y gedeclareerd moeten worden als een float.

4 Eigen input

In C# kan je eigen input geven in de console, die vervolgens gebruikt kan worden om het programma te laten runnen. Hiervoor gebruik je de methode *Console.ReadLine()*. In het volgende programma wordt hier van gebruik gemaakt.

```
static void Main(string[] args)
    {
        Console.WriteLine("Wat is je naam?");

        string naam = Console.ReadLine();

        Console.WriteLine("Hallo " + naam + "!");
        Console.ReadLine();
    }
```

De input die ingelezen wordt met de methode Console.ReadLine(), is altijd een string. Dit betekent dat je de string moet omzetten naar een integer als je een getal invoert waar berekeningen mee gedaan moeten worden. Het omzetten van de ene variabele naar de andere variabele gebeurt met de *Parse* methode zoals in het volgende voorbeeld.

```
static void Main(string[] args)
    {
        Console.WriteLine("Tel er 18 bij op:");

        string input = Console.ReadLine();
        int getal = int.Parse(input);

        int antwoord = getal + 18;

        Console.WriteLine("De som van 18 en " + getal + " is:
            " + antwoord);
        Console.ReadLine();
    }
```

Zoals Console.ReadLine() een string inleest, zo print de methode Console.WriteLine() altijd een string in de console. Voor sommige typen variabelen wordt de variabele 'automatisch' omgezet naar een string, zoals in dit geval voor een getal gebeurt.

In het geval dat een regel meerdere getallen bevat die ingelezen moeten worden, kan de input gesplitst worden door de *Split()* methode. Tussen de haakjes wordt het teken meegegeven waar de input op gesplitst zal worden. De output zal teruggegeven worden in een array. Hier wordt later verder op ingegaan.

```
Console.WriteLine("Twee getallen optellen:");

string input = Console.ReadLine().Split(' '); // input: 12 16
int x = int.Parse(input[0]);
int y = int.Parse(input[1]);

int antwoord = x + y;

Console.WriteLine("De som van " + x + " en " + y + " is: " +
    antwoord);
Console.ReadLine();
```

5 Belangrijke functies/statements

5.a For- en While-loop

For-loop

In C# heeft de for-loop een iets andere notatie dan in Python. In de aanroep van de for-loop geef je aan met welk getal de loop begint, tot welk getal hij loopt en de stapgrootte. Een eenvoudig voorbeeld is het uitrekenen van de som $\sum_{i=0}^{10} 3i + 4$.

```
    static void Main(string[] args)
        {
            int som = 0;
            int deelsom;

            for (int i = 0; i <= 10; i++)
            {
                deelsom = 3 * i + 4;
                som = som + deelsom;
                Console.WriteLine(som);
            }
            Console.WriteLine("Het antwoord is: " + som);
            Console.ReadLine();
        }
```

Het antwoord dat uit dit programma zou moeten komen is 209. De som kan ook berekend worden vanaf $i = 10$ tot $i = 20$. Dan zou aanroep van de for-loop er uitzien als

```
    for (int i = 10; i < 21; i++)
```

In deze voorbeelden is gebruik gemaakt van gehele getallen, maar de loop kan ook over komma-getallen lopen, zoals in het volgende voorbeeld een rij getallen van 0.1 tot 2.2 wordt afgedrukt.

```
    for (float i = 0.1; i < 2.3; i = i + 0.1)
            {
                Console.WriteLine(i);
            }
            Console.ReadLine();
```

Let wel op, als je werkt met komma-getallen is de kans groter dat er afrondfouten gemaakt worden.

While-loop

De while-loop in C# werkt hetzelfde als in Python. Dus

```
    static void Main(string[] args)
        {
            int i = 0;
            while (i < 6)
            {
                Console.WriteLine(i);
                i++;
            }
            Console.ReadLine();
        }
```

geeft een lijst met getallen van 0 tot 5.

5.b If-statements

In C# kan er ook gebruik worden gemaakt van if-statements. Dat ziet er als volgt uit:

```
if (...)
{}
else if (...)
{}
else
{}
```

Tussen de haakjes staan de voorwaarden... Belangrijke operatoren voor while-loops en if-statements zijn te vinden in Tabel 2.

Een voorbeeld met && en || operatoren is gegeven in het volgende voorbeeld.

Operator	Teken
gelijk aan	==
kleiner/groter dan	< / >
kleiner/groter gelijk aan	<= / >=
ongelijk aan	!=
en	&&
of	||
niet	!

Table 2: Operatoren

```
if (a == 20 && b == 20)
{
    Console.WriteLine("a en b zijn gelijk aan 20");
}
else if (a == 20 || b == 20)
{
    Console.WriteLine("a of b is gelijk aan 20");
}
else
{
    Console.WriteLine("a en b zijn niet gelijk aan 20");
}
```

Continue en break

Om een loop voortijdig te beëindigen als variabelen aan een bepaalde voorwaarden voldoen, kan een if-statement gebruikt worden met de functie *'break'*. Door deze functie wordt de loop gestopt.

Met de functie *'continue'* kan de loop verder gaan naar de volgende waarde van i, zonder nog de andere if-statements langs te gaan.

6 Lists en Arrays

In C# kan gebruik gemaakt worden van twee soorten lijst objecten, lists en arrays. Een list is een opgebouwd uit arrays die omgestapeld kunnen worden. Ze zijn echter wel verschillend in gebruik. Het belangrijkste verschil is dat een array een vaste lengte heeft en een list niet.

Het zijn beide objecten, dus ze moeten beide geïnitialiseerd worden.

6.a Array

Om een array van gehele getallen willen maken met lengte 4, dan wordt deze geïnitialiseerd door:

```
int[] array = new int[4];
```

Als er direct waarden in deze array opgeslagen kunnen worden, kan dat op een van de volgende manieren:

```
int[] array = new int[4] {0, 1, 2, 3};
int[] array = new int[] {0, 1, 2, 3};
int[] array = {0, 1, 2, 3};
```

Als je eenmaal de array gemaakt hebt, kan hieraan geen element meer aan toegevoegd of verwijderd worden. Wel kunnen de elementen makkelijk gewijzigd worden als het indexnummer bekend is. Het indexnummer van het eerste element is altijd 0, dit geldt ook voor een list.
De initialisatie

```
int[] array = new int[4] {0, 1, 2, 3};
```

geeft de array [0,1,2,3]. Om het eerste element in een 4 te veranderen, wordt de waarde met index 0 aangepast, zoals

```
array[0] = 4;
```

De lengte van een array kan gevonden worden door de functie *array.Count()*.
Een voorbeeld waarbij het heel handig is om met arrays te werken, zijn matrices. Er wordt dan gebruik gemaakt van twee-dimensionale arrays. Deze worden geïnitialiseerd door

```
int[,] array = new int[4, 2];
```

Dit geeft je een matrix met vier rijen en twee kolommen. Een andere manier om dit te doen gaat als volgt:

```
int[,] array5 = new int[,] { { 1, 2 }, { 3, 4 }, { 5, 6 }, {
    7, 8 } };
```

Waarbij de waarden gelijk meegegeven worden aan de matrix.

6.b List

Een List object wordt geïnitialiseerd door

```
List<T> lijst = new List<T>();
```

waar T het type is van de variabele die de lijst zal bevatten. In een lijst worden alle elementen een voor een toegevoegd. Ze kunnen niet bij het initialiseren van de lijst gelijk worden meegegeven. Het toevoegen van elementen gaat door middel van de functie

lijst.Add(element).

Om een element op een bepaalde index van een lijst te vinden, wordt er gebruik gemaakt van dezelfde notatie als bij arrays, namelijk *lijst[i]*.
Een aantal operaties die op een lijst uitgevoerd kunnen worden, zijn te zien in Tabel 3.

Operatie	List
elementen toevoegen	lijst.Add(i)
elementen verwijderen	lijst.Remove(i)
check of element in de lijst zit	lijst.Contains(i)
check hoeveel elementen de lijst bevat	lijst.Count()
lijst sorteren	lijst.Sort()

Table 3: List operaties

Zo zijn er nog meer operaties die je op een lijst kan doen, zoals het gemiddelde nemen of het maximum vinden.

7 Methoden en Classes

Methoden
In Python heb je geleerd om functies te schrijven die in het programma gebruikt kunnen worden. In C# kan dit natuurlijk ook. Het is hierin belangrijk om aan te geven wat voor een variabele de functie terug zal geven en ook welk soort variabelen je aan de functie zelf meegeeft. Een methode schrijven om de norm van twee getallen te berekenen, zal er als volgt uitzien

```
public static double Norm(int a, int b)
    {
        int x = a * a + b * b;
        double y = Math.Sqrt(x);
        return y;
    }
```

Het woord *public* voor de functie staat voor welke klassen toegang hebben tot deze functie. In dit geval zal elke klasse toegang hebben tot deze methode. Laat je *public* weg of zet je er *private* neer, dan kan de methode alleen gebruikt worden in de klasse zelf. Wat *static* doet is iets ingewikkelder, dus daar zal verder niet veel aandacht aan besteed worden.

Als je ook van floats en doubles de norm wil berekenen, zal je natuurlijk van a, b en x floats (of doubles) moeten maken.

Classes
In C# kan handig gebruik gemaakt worden van een zelfgemaakte *class* met zijn eigen objecten en methoden. Om een lijst bijhouden met alle studenten Wiskunde en Wiskunde en Toepassingen met de studentnummers en opleiding. Dan kan er een eigen klasse gemaakt worden, bijvoorbeeld de klasse 'Student'.

```
class Student { }
```

Het is de bedoeling dat van elke student het studentnummer en de bacheloropleiding opgeslagen wordt. Dit wordt gedaan door de klasse membervariabelen mee te geven. Een mem-

bervariabele hoort bij een specifiek object, in dit geval dus een student, en is toegankelijk voor alle methoden in een klasse.

```
class Student
{
    public int studentnr;
    public string bachelor;
}
```

Om nu daadwerkelijk een object 'Student' aan te maken, hebben we een zogeheten constructormethode nodig. Deze methode heeft altijd dezelfde naam als de klasse en het is mogelijk om meerdere constructormethoden te hebben. Er *hoeft* niets in deze methode te staan, al kan dat wel handig zijn. De klasse zal er dan als volgt uitzien:

```
class Student
{
    public int studentnr;
    public string bachelor;

    public Student() { }
}
```

Let op dat de klasse 'Program' met de methode 'Main' nog steeds bestaat.

Omdat de klasse twee membervariabelen heeft, heeft elk object dat je aanmaakt de twee variabelen opgeslagen, alleen hebben ze nog geen waarde. Een object aanmaken en een studentnummer en opleiding toevoegen gaat als volgt

```
static void Main(string[] args)
    {
        Student student1 = new Student();
        student1.bachelor = "Wiskunde en Toepassingen";
        student1.studentnr = 123;
    }
```

Het is ook mogelijk om meerdere constructormethoden te hebben en argumenten in de aanroep mee te geven.

```
class Student
{
    public int studentnr;
    public string bachelor;

    // eerste constructormethode
    public Student() { }

    // tweede construcormethode
    public Student(string opleiding, int nr)
    {
        studentnr = nr;
        bachelor = opleiding;
    }
}
```

Met deze methoden kan een object 'Student' in een keer aangemaakt worden met de opleiding en studentnummer door de volgende regel:

```
Student student2 = new Student("Wiskunde", 234);
Student student3 = new Student("Wiskunde", 345);
```

Er kan nu bijvoorbeeld een lijst gemaakt worden met *Student* als type.

```
List<Student> students = new List<Student>();
students.Add(student1);
students.Add(student2);
students.Add(student3);
```

8 Opmerkingen

- Let op, het maximale getal voor een integer is 2 147 483 647, dit komt omdat een integer 32 bits is en geen grotere getallen kan opslaan. Moet je met grotere getallen werken, dan heb je de klasse BigInteger nodig in de System.Numerics libary.
- Bij het maken van het eerste programma kies je voor de Console App (.NET Framework). Dit zorgt ervoor dat er al een framework programma is. Oftewel er bestaat al een voorgeschreven main programma hebt, interface etc.
- Als je geen static voor een methode hebt staat en hij geeft een error, werkt het vaak om static er wel voor te zetten. Wil je weten wat dit precies doet, kan je dat op de volgende site lezen: `https://www.tutorialsteacher.com/csharp/csharp-static`

9 Om te oefenen

Probeer nu een programma te schrijven dat de input hieronder kan sorteren op de volgende manieren:

- Schrijf een programma dat studenten sorteert op opleiding (eerst de wiskunde studenten, dan de toepassingen studenten) en de naam van de studenten teruggeeft.
- Schrijf een programma dat het aantal studenten per opleiding teruggeeft.
- Schrijf een programma die het gemiddelde/maximum/minimum/som cijfer van alle studenten teruggeeft.
- Schrijf een programma dat studenten sorteert op studentennummer en eventueel de naam van de student teruggeeft.

Tip: Zet alle studenten in een lijst (gebruik hiervoor een for-loop) en schrijf verschillende methoden om te sorteren.

De volgende lijst is de input die gegeven is. Het eerste getal is het totaal aantal studenten in de lijst.

```
24
Jan Wiskunde 235 7.5
Piet Toepassingen 126 6.9
Klaas Wiskunde 156 9.1
Jaap Wiskunde 159 7.5
Mies Toepassingen 147 6.3
Tim Wiskunde 268 6.8
Mark Wiskunde 984 7.4
Lisa Toepassingen 754 6.9
Eva Toepassingen 896 7.8
Tom Toepassingen 865 8.0
Rianne Wiskunde 123 8.1
Pien Wiskunde 234 6.8
Bart Wiskunde 456 6.5
Kees Toepassingen 567 6.9
Paul Toepassingen 789 6.3
Petra Toepassingen 890 7.1
Fien Wiskunde 900 7.5
Sofie Toepassingen 458 7.4
Gijs Toepassingen 588 6.9 7.6
Rien Wiskunde 225 6.6
Roel Wiskunde 365 5.6
Cor Toepassingen 360 6.8
Bas Toepassingen 314 7.0
Roos Toepassingen 154 7.2
```

Output:

Het sorteren op opleiding:

```
Jan
Klaas
Jaap
Tim
Mark
Rianne
```

```
Pien
Bart
Fien
Rien
Roel
Piet
Mies
Lisa
Eva
Tom
Kees
Paul
Petra
Sofie
Gijs
Cor
Bas
Roos
```

Het aantal studenten per opleiding:

```
Aantal wiskunde studenten: 11
Aantal toepassingen studenten: 13
```

Het gemiddelde, laagste en hoogste cijfer (met eventueel de naam erbij):

```
Gemiddelde: 7.120834
Laagste: 5.6        (naam: Roel)
Hoogste: 9.1        (naam: Klaas)
```

En alle studenten gesorteerd op studentennummer:

```
123
126
147
154
156
159
225
234
235
268
314
360
365
456
458
567
588
754
789
865
890
896
900
984
```