C# Essentials - Veelgebruikte Functies

Inhoudsopgave

1. Introductie

1.1. Inleiding

Je hebt ondertussen al kennisgemaakt met variabelen en controlestructuren. Daarmee kan je al wel wat doen. Maar in de praktijk kom je vaak terugkerende taken tegen zoals:

- een getal afronden
- een stukje tekst uit een zin halen
- een random getal genereren

Gelukkig moet je die zaken niet zelf programmeren. C# heeft hiervoor al heel wat functies klaarstaan. Deze zitten in zogenaamde **klassen** die deel uitmaken van het .NET-framework.

1.2. Wat bedoelen we met “veelgebruikte functies”?

Veelgebruikte functies zijn stukjes code die al kant-en-klaar in het .NET-platform zitten en die je met één lijn code kan oproepen. Je hoeft dus zelf niet te weten hoe ze exact werken, je moet gewoon weten **hoe je ze kan gebruiken**.

Voorbeelden van zulke functies zijn:

- `Math.Round(…)` om af te ronden
- `"tekst".Substring(…)` om een deel uit een string te halen
- `rnd.Next(…)` om een random getal te krijgen

In deze module leren we je hoe je deze functies efficiënt inzet aan de hand van voorbeelden.

1.3. Hoe weet je welke functies beschikbaar zijn?

In Visual Studio kan je:

- een klasse intypen (bv. `Math.`) en wachten op de suggesties
- met `CTRL + klik` op een methode klikken om naar de documentatie te springen

Je kan ook opzoeken via de officiële documentatie: https://learn.microsoft.com/dotnet/api/

1.4. Wat leer je in deze module?

In deze module ontdek je de belangrijkste functies van de volgende klassen:

- `System.Math` – voor wiskundige berekeningen
- `System.String` – voor tekstverwerking
- `System.Text.StringBuilder` – voor het efficiënt opbouwen van tekst
- `System.Random` – voor willekeurige getallen

Bij elke klasse krijg je:

- een korte uitleg
- een overzicht van veelgebruikte functies
- duidelijke voorbeelden
- praktische tips

Zo leer je om deze functies zelf toe te passen in je programma’s. Ze vormen een goede basis voor alles wat je later nog zal bouwen.

2. System.Math

2.1. Wat is `System.Math`?

Soms moet je in je programma berekeningen uitvoeren zoals afronden, het minimum van twee getallen bepalen of een vierkantswortel nemen. Je hoeft dit niet zelf te programmeren. C# bevat hiervoor de klasse `Math` met allerlei handige functies.

Om `Math` te gebruiken, hoef je enkel bovenaan je code te schrijven:

```
using System;
```

2.2. Constantes

Sommige wiskundige constanten zoals π (pi) of e (de grondslag van de natuurlijke logaritme) zijn al voorzien:

```
Console.WriteLine(Math.PI);   // 3.14159265358979
Console.WriteLine(Math.E);    // 2.71828182845905
```

2.3. Functies

2.3.1. Abs

Geeft het positieve equivalent van een getal terug, ook als het negatief is:

```
double value = -12.34;
Console.WriteLine(Math.Abs(value));  // 12.34
```

2.3.2. Min & Max

Geeft het kleinste (min) of grootste (max) getal van 2 gegeven getallen:

```
int number1 = 17;
int number2 = 35;

Console.WriteLine(Math.Min(number1, number2));  // 17
Console.WriteLine(Math.Max(number1, number2));  // 35
```

2.3.3. Round

Afronden naar het dichtstbijzijnde geheel getal of op een bepaald aantal decimalen.

```
Console.WriteLine(Math.Round(10.5));          // 10  → afronden naar
dichtstbijzijnde even getal
Console.WriteLine(Math.Round(10.51));         // 11
Console.WriteLine(Math.Round(-12.3456, 2));   // -12,35 → afronden op 2
decimalen
```

Bij de standaardinstelling gebruikt C# **Banker's Rounding**: bij `.5` kiest hij voor het dichtstbijzijnde **even getal** (dus 10.5 → 10 en 11.5 → 12).

Je kan echter ook zelf bepalen **hoe** er wordt afgerond via een extra parameter van het type `MidpointRounding`:

```
Console.WriteLine(Math.Round(10.5, MidpointRounding.ToEven));       // 10
(standaard in C#)
Console.WriteLine(Math.Round(10.5, MidpointRounding.AwayFromZero)); // 11
(klassiek afronden)
Console.WriteLine(Math.Round(-1.25, 1, MidpointRounding.AwayFromZero)); // -1,3
```

- `MidpointRounding.ToEven` → afronden naar het dichtstbijzijnde **even** getal bij .5 (standaardgedrag)
- `MidpointRounding.AwayFromZero` → afronden **weg van nul**, dus 0.5 wordt 1 en -0.5 wordt -1

NOTE: Gebruik `AwayFromZero` als je klassieke wiskundige afronding verwacht, bijvoorbeeld bij geldbedragen.

2.3.4. Ceiling & Floor

Afronden naar boven (ceiling) of beneden (floor):

```
Console.WriteLine(Math.Ceiling(11.5));   // 12
Console.WriteLine(Math.Floor(11.5));     // 11
Console.WriteLine(Math.Ceiling(-11.5));  // -11
Console.WriteLine(Math.Floor(-11.5));    // -12
```

2.3.5. Pow

Een getal tot een bepaalde macht verheffen:

```
Console.WriteLine(Math.Pow(2, 3));  // 8 (2³ = 2 * 2 * 2)
```

2.3.6. Log

Geeft terug tot welke macht je een getal moet verheffen om een ander te bekomen:

```
Console.WriteLine(Math.Log(8, 2));  // 3 (want 2³ = 8)
```

2.3.7. Sqrt

Geeft de vierkantswortel van een getal terug:

```
Console.WriteLine(Math.Sqrt(16));  // 4
```

2.4. Oefeningen

2.4.1. Circel

Maak een programma dat de straal van een cirkel vraagt aan de gebruiker en zowel de oppervlakte als de omtrek berekent. Gebruik hiervoor `Math.PI` en `Math.Pow`. Zorg ervoor dat zowel de oppervlakte als de omtrek afgerond worden op 4 cijfers na de komma.

```
Geef de straal van de cirkel: 5

Oppervlakte:     78,5398
Omtrek:          31,4159
```

Oplossing

```
Console.Write("Geef de straal van de cirkel: ");
double radius = double.Parse(Console.ReadLine());

double area = Math.PI * Math.Pow(radius, 2);
double perimeter = 2 * Math.PI * radius;

Console.WriteLine($"Oppervlakte: {area}");
Console.WriteLine($"Omtrek: {perimeter}");
```

2.4.2. Temperatuur

Schrijf een programma dat twee temperaturen inleest (bijvoorbeeld gisteren en vandaag) en:

- het verschil tussen deze twee temperaturen toont
- de maximum temperatuur toont
- de minimum temperatuur toont

```
Geef temperatuur in van gisteren: 24
Geef temperatuur in van vandaag:  19

Het verschil is 5°C
Het maximum is 24°C
Het minimum is 19°C
```

Het verschil tussen 2 temperaturen wordt altijd als een positief getal getoond.

Oplossing

```
Console.Write("Geef temperatuur in van gisteren: ");
double yesterday = double.Parse(Console.ReadLine());

Console.Write("Geef temperatuur in van vandaag:  ");
double today = double.Parse(Console.ReadLine());

Console.WriteLine();

double difference = Math.Abs(yesterday - today);
double maximum = Math.Max(yesterday, today);
double minimum = Math.Min(yesterday, today);
```

```
Console.WriteLine($"Het verschil is {difference}°C");
Console.WriteLine($"Het maximum is {maximum}°C");
Console.WriteLine($"Het minimum is {minimum}°C");
```

3. System.String

3.1. Wat is `System.String`?

Een string is een reeks van karakters, zoals `"Hallo"` of `"C# is leuk"`. Je zal merken dat strings een belangrijke rol spelen in veel programma's, zeker als je werkt met invoer van de gebruiker.

Strings worden in C# voorgesteld met het type `string`. Je hoeft daarvoor geen extra namespace toe te voegen, want `System.String` is standaard beschikbaar.

3.2. String vergelijken

Gebruik nooit `==` om strings te vergelijken. Dat kan onverwachte fouten geven. Gebruik liever de functie `Equals()` of `IsNullOrEmpty()`

3.2.1. Equals

Vergelijkt twee strings en geeft `true` als ze exact gelijk zijn, inclusief hoofdletters.

```
string name = "C Sharp";
bool isEqual = name.Equals("C Sharp");
Console.WriteLine(isEqual);  // true
```

Je kan ook aangeven dat hoofdletters genegeerd moeten worden. In dat geval gebruik je `StringComparison.OrdinalIgnoreCase`.

```
string name = "C Sharp";
bool isEqual = name.Equals("c sharp", StringComparison.OrdinalIgnoreCase);
Console.WriteLine(isEqual);  // true
```

3.2.2. IsNullOrEmpty

Geeft `true` als de string `null` of leeg is (`""`). Deze functie wordt vaak gebruikt om gebruikersinvoer te controleren.

```
string input = "";
if (string.IsNullOrEmpty(input))
{
    Console.WriteLine("Ongeldige invoer.");
}
```

3.2.3. IsNullOrWhiteSpace

Doet hetzelfde als `IsNullOrEmpty(…)`, maar beschouwt ook spaties als leeg.

```
string input = "   ";
if (string.IsNullOrWhiteSpace(input))
```

```
{
    Console.WriteLine("Er werd niets ingevuld.");
}
```

3.3. String onderzoeken

3.3.1. Length

Geeft het aantal tekens in een string terug.

```
string sentence = "Welkom!";
Console.WriteLine(sentence.Length);  // 7
```

3.3.2. Contains

Controleert of een string een bepaalde tekst bevat. Je krijgt `true` of `false` als resultaat.

```
string text = "Welkom in de les C#";
Console.WriteLine(text.Contains("C#"));   // true
Console.WriteLine(text.Contains("Java")); // false
```

3.3.3. IndexOf

Zoekt naar de positie van een bepaalde tekst in een string. Als de tekst niet voorkomt, krijg je `-1`.

```
string text = "Welkom in de les C#";
Console.WriteLine(text.IndexOf("les"));   // 13
Console.WriteLine(text.IndexOf("Python")); // -1
```

3.4. String bewerken

> String objects are immutable: they can't be changed after they're created.
>
> — Bron: learn.microsoft.com

Technisch gezien kan je een string niet wijzigen. Onderstaande functies maken dus eigenlijk een nieuwe string aan.

3.4.1. Substring

Haalt een deel van een string uit een langere tekst. Je geeft de startpositie op (vanaf 0), en eventueel ook het aantal tekens.

```
string text = "Visual CSharp";
Console.WriteLine(text.Substring(2));    // sual CSharp
Console.WriteLine(text.Substring(2, 4)); // sual
```

3.4.2. Concat

Voegt meerdere strings samen tot één nieuwe string

```
string text1 = "C# ";
string text2 = "is ";
string text3 = "super leuk!"
string result = String.Concat(text1, text2, text3);
Console.WriteLine(result);    // C# is super leuk
```

3.4.3. Trim

Verwijdert spaties (en andere witruimte) aan het begin en het einde van een string. Je kan ook alleen het begin (`TrimStart`) of alleen het einde (`TrimEnd`) bewerken.

```
string input = "   hallo wereld   ";
Console.WriteLine(input.Trim());       // "hallo wereld"
Console.WriteLine(input.TrimStart());  // "hallo wereld   "
Console.WriteLine(input.TrimEnd());    // "   hallo wereld"
```

3.4.4. Replace

Vervangt alle voorkomens van een teken of tekst in een string.

```
string sentence = "C# is leuk";
Console.WriteLine(sentence.Replace("leuk", "tof"));  // C# is tof
```

3.4.5. Insert

Voegt tekst in op een bepaalde positie.

```
string sentence = "C# is leuk!";
Console.WriteLine(sentence.Insert(6, "super "));
// C# is super leuk!
```

3.4.6. Remove

Verwijdert een deel van een string. Je geeft de startpositie, en optioneel ook hoeveel tekens je wil verwijderen.

```
string sentence = "C# is leuk!";
Console.WriteLine(sentence.Remove(4));      // C# i
Console.WriteLine(sentence.Remove(4, 3));   // C# ieuk!
```

3.4.7. ToUpper en ToLower

`ToUpper()` zet alle letters in hoofdletters. `ToLower()` zet alles om naar kleine letters.

```
string word = "Test";
Console.WriteLine(word.ToUpper());  // TEST
Console.WriteLine(word.ToLower());  // test
```

3.5. Oefeningen

3.5.1. Gebruikersnaam en wachtwoord

Vraag een gebruikersnaam en wachtwoord aan de gebruiker en controleer:

- of de gebruikersnaam exact `"admin"` is. Negeer hoofdletters en overbodige spaties
- of het wachtwoord minstens 6 tekens bevat

```
Voer je gebruikersnaam in: ADmin
Voer je wachtwoord in: 12345

Toegang geweigerd.
```

```
Voer je gebruikersnaam in: ADmin
Voer je wachtwoord in: abc123

Welkom, administrator!
```

Oplossing

```
Console.Write("Voer je gebruikersnaam in: ");
string username = Console.ReadLine();
username = username.Trim();

Console.Write("Voer je wachtwoord in: ");
string password = Console.ReadLine();

Console.WriteLine();

if (username.Equals("admin", StringComparison.OrdinalIgnoreCase) &&
password.Length >= 6)
{
    Console.WriteLine("Welkom, administrator!");
}
else
{
    Console.WriteLine("Toegang geweigerd.");
}
```

4. System.DateTime

4.1. Wat is `System.DateTime`?

Of je nu de huidige datum wil opvragen, een verjaardag wil vergelijken of wil weten hoeveel dagen er nog resten tot het einde van het jaar — de klasse `DateTime` biedt alles wat je nodig hebt.

`DateTime` maakt deel uit van de namespace `System` en is dus meteen beschikbaar in je project.

4.1.1. Now

Geeft de volledige datum en het tijdstip van dit moment.

```
DateTime now = DateTime.Now;
Console.WriteLine(now);  // bv. 21/05/2025 13:45:12
```

4.1.2. Today

Geeft enkel de datum (zonder tijdstip) van dit moment.

```
DateTime today = DateTime.Today;
Console.WriteLine(today);  // bv. 21/05/2025 00:00:00
```

4.1.3. new DateTime

Maakt een nieuwe datum aan op basis van een gegeven jaar, maand en dag

```
DateTime birthday = new DateTime(2000, 12, 15);  // 15 december 2000
```

4.1.4. Parse

Converteert tekst naar een datum.

```
DateTime parsedDate = DateTime.Parse("2024-04-01");
```

4.1.5. AddDays, AddMonths, AddYears

Past een bestaande datum aan door er dagen, maanden of jaren aan toe te voegen

```
DateTime today = DateTime.Today;
Console.WriteLine(today.AddDays(5));    // 5 dagen later
Console.WriteLine(today.AddMonths(1));  // 1 maand later
Console.WriteLine(today.AddYears(-1));  // 1 jaar vroeger
```

4.1.6. DayOfWeek

Toont op welke dag een bepaalde datum valt:

```
DateTime date = new DateTime(2025, 5, 21);
```

```
Console.WriteLine(date.DayOfWeek);  // bv. Wednesday
```

Als je het nummer van de dag wil (maandag = 1), gebruik dan `(int)`:

```
int dayNumber = (int)date.DayOfWeek;
Console.WriteLine(dayNumber);  // bv. 3
```

Let op: in C# is zondag = 0, maandag = 1, …, zaterdag = 6

4.1.7. TimeOfDay

Geeft enkel het tijdstip van een `DateTime`-waarde:

```
DateTime now = DateTime.Now;
Console.WriteLine(now.TimeOfDay);  // bv. 13:45:12.546
```

4.1.8. ToString

Zet een `DateTime` om naar een string in verschillende formaten:

```
DateTime now = DateTime.Now;

Console.WriteLine(now.ToLongDateString());   // woensdag 21 mei 2025
Console.WriteLine(now.ToShortDateString());  // 21/05/2025
Console.WriteLine(now.ToLongTimeString());   // 13:45:12
Console.WriteLine(now.ToShortTimeString());  // 13:45
```

4.2. System.TimeSpan

De klasse `TimeSpan` stelt een tijdsinterval voor. Je gebruikt `TimeSpan` om bijvoorbeeld het verschil tussen twee tijdstippen te berekenen, of om zelf een bepaalde duur aan te maken.

4.2.1. new TimeSpan

Maakt een nieuw tijdsinterval aan

```
TimeSpan interval = new TimeSpan(1, 30, 45);  // 1 uur, 30 minuten, 45 seconden
Console.WriteLine(interval);  // 01:30:45
```

4.2.2. From…

Maakt een nieuw tijdsinterval op basis van een aantal dagen, uren, minuten of seconden.

```
TimeSpan ts1 = TimeSpan.FromDays(12);
TimeSpan ts2 = TimeSpan.FromHours(8);
TimeSpan ts3 = TimeSpan.FromMinutes(20);
```

4.2.3. Verschil

Wanneer je twee `DateTime`-objecten van elkaar aftrekt, krijg je een `TimeSpan`.

```
DateTime start = new DateTime(2025, 5, 21, 9, 0, 0);
DateTime end = new DateTime(2025, 5, 21, 12, 45, 30);

TimeSpan duration = end - start;
Console.WriteLine(duration);          // 03:45:30
```

4.2.4. Hours, Minutes, Seconds

Met eigenschappen kan je een `TimeSpan` analyseren:

```
DateTime start = new DateTime(2025, 5, 21, 9, 0, 0);
DateTime end = new DateTime(2025, 5, 21, 12, 45, 30);

TimeSpan duration = end - start;

Console.WriteLine(duration.Hours);         // 3
Console.WriteLine(duration.Minutes);       // 45
Console.WriteLine(duration.Seconds);       // 30
Console.WriteLine(duration.TotalSeconds);  // 13530
```

5. System.Text.StringBuilder

5.1. Wat is `System.Text.StringBuilder`?

In C# zijn strings **immutable**, wat betekent dat je ze technisch gezien niet kan wijzigen. Wanneer je een bestaande string verandert (bijvoorbeeld via `Replace`, `Insert`, …), wordt er onder de motorkap eigenlijk telkens een **nieuwe** string aangemaakt.

Als je vaak tekst aan elkaar plakt of bewerkt (en dus telkens een nieuwe string maakt), kan dat je programma trager maken. In die gevallen gebruik je beter de klasse `StringBuilder`. Die laat je toe om tekst stapsgewijs op te bouwen zonder telkens een nieuwe string te maken.

Om `StringBuilder` te gebruiken, voeg je bovenaan je code toe:

```
using System.Text;
```

5.2. StringBuilder aanmaken

Je start met een nieuwe instantie van `StringBuilder`. Deze is eerst leeg.

```
StringBuilder sb = new StringBuilder();
```

5.3. Append

Voegt tekst toe aan het einde van de `StringBuilder`.

```
sb.Append("Score: ");
sb.Append(42);
```

De tekst groeit dus telkens aan.

5.4. AppendLine

Doet hetzelfde als `Append(…)`, maar voegt automatisch een nieuwe regel (`\n`) toe na elke lijn.

```
sb.AppendLine("Result:");
sb.AppendLine("Passed");
```

5.5. ToString

Op het einde zet je de `StringBuilder` om naar een gewone `string` met `ToString()`.

```
string result = sb.ToString();
Console.WriteLine(result);
```

5.6. Insert

Voegt tekst toe op een specifieke positie.

```
sb.Insert(0, "Report:\n");
```

5.7. Replace

Vervangt één of meerdere tekens of woorden in het resultaat.

```
sb.Replace("Score", "Points");
```

5.8. Remove

Verwijdert een aantal tekens vanaf een bepaalde positie.

```
sb.Remove(0, 7);  // verwijder de eerste 7 tekens
```

5.9. Clear

Maakt de volledige inhoud van de `StringBuilder` leeg.

```
sb.Clear();
```

5.10. AppendFormat

Je kan ook meerdere stukjes tekst combineren in één lijn met `AppendFormat(…)`.

```
sb.AppendFormat("Student {0} heeft {1} op {2}", "Emma", 17, 20);
```

Dezelfde functionaliteit kan je bereiken met de `Append()`-methode in combinatie met interpolatie

5.11. Voorbeeld

```
StringBuilder sb = new StringBuilder();

sb.AppendLine("Result Overview:");
sb.AppendLine("--------------------");
sb.Append("Name:\t").AppendLine("Alice");
sb.Append("Score:\t").AppendLine("18/20");

string output = sb.ToString();
Console.WriteLine(output);
```

6. System.Random

6.1. Wat is `System.Random`?

Willekeurige getallen worden in heel wat toepassingen gebruikt: een dobbelsteen, een quiz met willekeurige vragen, een wachtwoordgenerator, een spel waarin je vijanden op een willekeurige plaats laat verschijnen, enzovoort.

De klasse `Random` hoort bij de namespace `System` en is dus meteen beschikbaar in je project.

6.1.1. Random aanmaken

Je start altijd met een nieuwe instantie van de klasse `Random`:

```
Random random = new Random();
```

Deze instantie kan je dan gebruiken om verschillende soorten getallen te genereren.

Je moet maar één instantie aanmaken van de `Random`-klasse. Je kan deze telkens hergebruiken om een nieuw willekeurig getal te genereren.

6.1.2. Next

De methode `Next(…)` geeft een willekeurig geheel getal terug.

```
int number = random.Next();  // willekeurig positief getal
```

Je kan ook grenzen opgeven:

```
int number1 = random.Next(10);     // getal van 0 t.e.m. 9
int number2 = random.Next(1, 7);   // getal van 1 t.e.m. 6
```

De bovengrens is **exclusief**, de ondergrens is **inclusief**.

6.1.3. NextDouble

De methode `NextDouble()` geeft een willekeurig **kommagetal** terug tussen 0.0 (inclusief) en 1.0 (exclusief):

```
double value = random.NextDouble();
Console.WriteLine(value);  // bv. 0.72358123
```

Je kan dit combineren met andere berekeningen om een willekeurig kommagetal binnen een bepaald bereik te maken:

```
double scaled = random.NextDouble() * 10;  // tussen 0.0 en 10.0
```

6.1.4. Dezelfde random-getallen (seed)

Je kan een **seed** meegeven bij het aanmaken van `Random`. Dan krijg je telkens dezelfde volgorde van getallen — handig voor testen of debugging.

```
Random seededRandom = new Random(42);

Console.WriteLine(seededRandom.Next(1, 101));
Console.WriteLine(seededRandom.Next(1, 101));
Console.WriteLine(seededRandom.Next(1, 101));
```

Als je geen seed opgeeft, gebruikt `Random` automatisch de systeemklok, zodat je telkens andere waarden krijgt.

6.2. Oefeningen

6.2.1. Raden

Schrijf een programma dat een willekeurig getal tussen 1 en 10 genereert en de gebruiker vraagt om het te raden. Zeg of het juist is of niet.

```
Raad het getal (1 t.e.m. 10): 8
Fout! Het juiste getal was 2.
```

```
Raad het getal (1 t.e.m. 10): 8
Juist geraden!
```

Oplossing

```
Random random = new Random();
int secret = random.Next(1, 11);

Console.Write("Raad het getal (1 t.e.m. 10): ");
int guess = int.Parse(Console.ReadLine());

if (guess == secret)
{
    Console.WriteLine("Juist geraden!");
}
else
{
    Console.WriteLine($"Fout! Het juiste getal was {secret}.");
}
```