

C# Essentials - Controlestructuren

Inhoudsopgave

1. Condities	3
1.1. Wat is een conditie?	3
1.2. Waarom zijn condities belangrijk?	3
1.3. Voorbeelden uit het echte leven	3
1.4. Voorbeeld in C#	3
1.5. In het kort	4
2. De <code>if</code> -structuur	5
2.1. Basisvorm	5
2.1.1. Voorbeeld	5
2.2. Zonder accolades {}	5
2.3. Meerdere <code>if</code> -blokken	5
2.4. In het kort	6
3. De <code>if-else</code> -structuur	7
3.1. Basisvorm	7
3.1.1. Voorbeeld	7
3.2. Zonder accolades {}	8
3.3. Veelvoorkomende fouten	8
3.4. Extra voorbeeld	8
3.5. In het kort	8
4. De ternary operator	9
4.1. Vorm	9
4.1.1. Voorbeeld	9
4.2. Wanneer gebruik je de ternary operator?	9
4.3. Vergelijking met <code>if else</code>	9
4.4. In het kort	10
5. Geneste condities	11
5.1. Wat is nesting?	11
5.2. <code>else if</code> gebruiken	11
5.3. Geneste <code>if</code> in praktijk	12
5.3.1. Tips	12
5.4. In het kort	12
6. De <code>switch</code> -structuur	13
6.1. Wat is een <code>switch</code> ?	13
6.2. Basisvorm	13
6.2.1. Voorbeeld	13

6.3. Waarom <code>break</code> ?	14
6.4. Gebruik van <code>default</code>	14
6.5. Wanneer gebruik je <code>switch</code> ?	14
6.6. In het kort	14
7. Scope van variabelen	16
7.1. Wat is scope?	16
7.2. Blok-scope	16
7.3. Scope in <code>Main</code>	16
7.4. Waarom is dit belangrijk?	17
7.5. Samenvatting	17
7.6. Oefening	17

1. Condities

1.1. Wat is een conditie?

Een conditie is een **vraag waarop het antwoord altijd true (waar) of false (onwaar)** is. Op basis van dat antwoord beslist het programma of het een stuk code wel of niet zal uitvoeren.

Bijvoorbeeld:

- "Is het getal groter dan 10?"
- "Heeft de speler nog levenspunten?"
- "Zijn de twee wachtwoorden gelijk?"

1.2. Waarom zijn condities belangrijk?

Condities maken je programma **intelligent**. Zonder condities zou elk programma gewoon een lijstje van instructies zijn die altijd worden uitgevoerd - zonder ook maar iets te controleren of te beslissen.

Denk aan een spel:

- je wil controleren of iemand gewonnen heeft;
- of een actie geldig is;
- of een level voltooid is.

Of aan een webshop:

- is er voldoende voorraad?
- is de kortingscode geldig?

1.3. Voorbeelden uit het echte leven

- Als je honger hebt, dan eet je iets.
- Als het licht op rood staat, dan stop je.
- Als de temperatuur onder nul is, trek je een jas aan.

In een programma vertalen we dat soort situaties naar **if-structuren**, zoals je in de volgende hoofdstukken leert.

1.4. Voorbeeld in C#

```
int temperature = 5;
if (temperature < 0)
{
    Console.WriteLine("Doe een warme jas aan!");
}
```

In dit voorbeeld:

- De conditie is `temperature < 0`
- Als die `true` is, dan toont het programma een bericht op het scherm.

1.5. In het kort

- Een **conditie** is een uitspraak die waar of onwaar kan zijn.
- Je gebruikt condities om **controle** te krijgen over de uitvoering van je programma.
- De belangrijkste sleutelwoorden bij condities zijn `if`, `else`, `switch` en `? :` (ternary operator).
- In de volgende hoofdstukken leer je elk van deze technieken kennen met duidelijke voorbeelden.

2. De if-structuur

2.1. Basisvorm

De algemene vorm van een if-structuur ziet er zo uit:

```
if (voorwaarde)
{
    // Code die wordt uitgevoerd als de voorwaarde waar is
}
```

De **voorwaarde** moet een **boolaansche expressie** zijn. Dat is een uitdrukking die `true` of `false` oplevert.

2.1.1. Voorbeeld

```
Console.WriteLine("Geef een getal...");
string input = Console.ReadLine();
int number = int.Parse(input);

if (number == 10)
{
    Console.WriteLine("Dit getal is gelijk aan 10.");
}
```

In dit voorbeeld:

- De gebruiker geeft een getal in.
- We controleren of het getal gelijk is aan 10 met `number == 10`.
- Als dit zo is, dan tonen we een boodschap.

2.2. Zonder accolades {}

Als er maar **één instructie** hoort bij de if, dan mag je de accolades `{ }` weglaten:

```
if (number == 10)
    Console.WriteLine("Dit getal is gelijk aan 10.");
```

Maar: **we raden aan om altijd accolades te gebruiken**, zeker als beginner. Zo vermijd je verwarring of fouten als je je code gaat uitbreiden.

2.3. Meerdere if-blokken

Je kan meerdere if-blokken probleemloos na elkaar gebruiken:

```
int number = 100;

if (number >= 0)
```

```
{  
    Console.WriteLine("Dit is een positief getal.");  
}  
  
if (number == 10)  
{  
    Console.WriteLine("Dit getal is gelijk aan 10.");  
}
```

2.4. In het kort

- Met een `if`-structuur kan je een beslissing nemen in je programma.
- De voorwaarde binnen de `if` moet een **boolaansche expressie** zijn (`true` of `false`).
- Als de voorwaarde waar is, wordt de code binnen de accolades uitgevoerd.
- Gebruik accolades `{ }` ook bij één lijn code, om fouten te vermijden en leesbaarheid te verhogen.

3. De `if-else`-structuur

3.1. Basisvorm

```
if (voorwaarde)
{
    // Code als voorwaarde true is
}
else
{
    // Code als voorwaarde false is
}
```

3.1.1. Voorbeeld



```
Console.WriteLine("Geef een getal in:");
string input = Console.ReadLine();
int number = int.Parse(input);

if (number == 10)
{
    Console.WriteLine("Dit getal is gelijk aan 10.");
}
else
{
    Console.WriteLine("Dit getal is verschillend van 10.");
}
```

In dit voorbeeld zijn er twee mogelijke uitvoeringen:

- Als `number == 10`, toon je een boodschap dat het gelijk is.
- In alle andere gevallen toon je dat het verschillend is van 10.

3.2. Zonder accolades {}

Als je slechts **één instructie** wil uitvoeren bij `if` en bij `else`, dan mogen de accolades `{ }` weggelaten worden. Toch raden we aan ze altijd te gebruiken voor de leesbaarheid.

```
if (number == 10)
    Console.WriteLine("Gelijk aan 10.");
else
    Console.WriteLine("Niet gelijk aan 10.");
```

3.3. Veelvoorkomende fouten

Let op met inspringing en haakjes:

- `else` hoort **bij de dichtstbijzijnde `if`**
- Vergeet nooit dat de voorwaarde enkel bij de `if` staat, niet bij de `else`.

3.4. Extra voorbeeld

```
int age = 17;

if (age >= 18)
{
    Console.WriteLine("Je bent meerderjarig.");
}
else
{
    Console.WriteLine("Je bent nog minderjarig.");
}
```

3.5. In het kort

- Gebruik `else` om iets te doen **wanneer de `if` niet uitgevoerd wordt**.
- De `if`- en `else`-blokken sluiten elkaar uit: **één van de twee wordt altijd uitgevoerd**.
- Dit is handig voor situaties met een duidelijke **twee-keuze-logica**.

4. De ternary operator

4.1. Vorm

```
voorwaarde ? waarde_als_true : waarde_als_false;
```

Het leest als: "Als de voorwaarde waar is, gebruik dan de eerste waarde, anders de tweede."

4.1.1. Voorbeeld

```
Console.WriteLine("Geef een getal in:");
string input = Console.ReadLine();
int number = int.Parse(input);

string result = (number == 10)
    ? "Dit getal is gelijk aan 10."
    : "Dit getal is verschillend van 10.";

Console.WriteLine(result);
```

In dit voorbeeld:

- `number == 10` is de voorwaarde.
- Als die waar is, wordt "Dit getal is gelijk aan 10." toegewezen aan `result`.
- Anders wordt "Dit getal is verschillend van 10." toegewezen.

4.2. Wanneer gebruik je de ternary operator?

- Als je een korte `if else` wil in één regel
- Als je een waarde wil toekennen aan een variabele op basis van een voorwaarde

Niet geschikt voor:

- Meerdere regels code
- Complexe logica

4.3. Vergelijking met `if else`

```
// if else
string result;
if (number == 10)
{
    result = "Gelijk aan 10.";
}
else
{
    result = "Verschillend van 10.;"
```

```
}

// ternary
string result = (number == 10) ? "Gelijk aan 10." : "Verschillend van 10.;"
```

Beide voorbeelden doen exact hetzelfde. De tweede versie is korter, maar misschien minder leesbaar voor beginners. Gebruik het dus met mate.

4.4. In het kort

- De **ternary operator** is een compacte manier om **if else** te schrijven.
- Vorm: `voorwaarde ? waarde1 : waarde2`
- Handig om **één waarde te kiezen op basis van een conditie**

5. Geneste condities

5.1. Wat is nesting?

Soms wil je **meer dan één voorwaarde na elkaar** controleren of een nieuwe controle doen binnen een bestaande `if`. Dat kan met geneste `if`-structuren.

```
int score = 85;

if (score >= 50)
{
    Console.WriteLine("Geslaagd!");

    if (score >= 80)
    {
        Console.WriteLine("Met onderscheiding!");
    }
}
```

In dit voorbeeld:

- De tweede `if` zit **binnen** de eerste.
- De tweede boodschap wordt enkel getoond als **beide voorwaarden** waar zijn.

5.2. `else if` gebruiken

Met `else if` kan je **meerdere mogelijkheden** controleren in één structuur.

```
double number = 15.2;

if (number < 10)
{
    Console.WriteLine("Kleiner dan 10");
}
else if (number >= 10 && number < 20)
{
    Console.WriteLine("Tussen 10 en 20");
}
else
{
    Console.WriteLine("20 of meer");
}
```

Hierdoor vermijd je dat je meerdere `if`-blokken onder elkaar moet schrijven. `else if` wordt enkel getest als de vorige `if` niet waar was.

5.3. Geneste `if` in praktijk

```
double number = 2.718;

if (number < 3)
{
    if (number > 2)
        Console.WriteLine("Tussen 2 en 3");
    else if (number > 1)
        Console.WriteLine("Tussen 1 en 2");
    else
        Console.WriteLine("Kleiner dan of gelijk aan 1");
}
```

Let op:

- Dit werkt prima, maar **te veel nesting maakt je code moeilijk leesbaar.**
- Als je merkt dat je meerdere lagen diep zit, kan het zinvol zijn om de logica op te splitsen.

5.3.1. Tips

- Gebruik inspringing om nesting overzichtelijk te houden.
- Bij meer dan 2-3 lagen nesting: denk na of het eenvoudiger kan.
- Combineer voorwaarden met logische operatoren (`&&`, `\|\|`) als dat de code eenvoudiger maakt.

5.4. In het kort

- **Nesting** betekent een `if` binnen een andere `if`.
- Gebruik `else if` om meerdere mogelijke gevallen te testen.
- Te veel nesting maakt code onduidelijk — hou het zo eenvoudig mogelijk.

6. De `switch`-structuur

6.1. Wat is een `switch`?

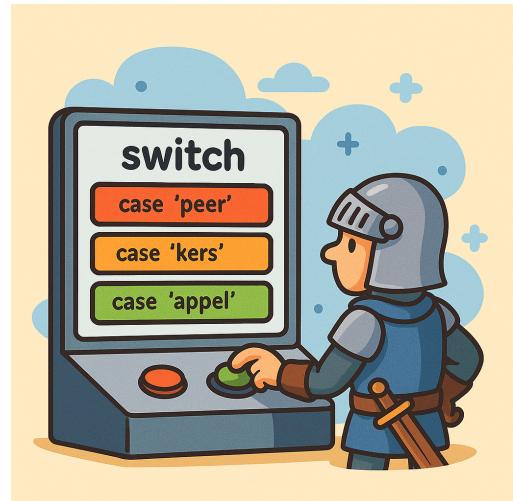
Een `switch` lijkt op een reeks `if else if`-blokken, maar is korter en overzichtelijker als je **veel vaste waarden** wil controleren.

6.2. Basisvorm

```
switch (variabele)
{
    case waarde1:
        // code bij waarde1
        break;
    case waarde2:
        // code bij waarde2
        break;
    default:
        // code als geen enkele case overeenkomt
        break;
}
```

- De `switch` controleert de waarde van één variabele.
- `case` vergelijkt die waarde met een specifieke constante.
- `break` zorgt ervoor dat de `switch` stopt zodra een match is gevonden.
- `default` wordt uitgevoerd als geen enkele `case` overeenkomt (optioneel).

6.2.1. Voorbeeld



```
Console.WriteLine("Geef een fruitsoort:");
string fruit = Console.ReadLine();
bool isDelicious;

switch (fruit)
```

```

{
    case "peer":
        isDelicious = true;
        break;
    case "kers":
        isDelicious = true;
        break;
    case "citroen":
    case "pompelmoes":
        isDelicious = false;
        break;
    default:
        isDelicious = false;
        break;
}

Console.WriteLine("Fruit is " + (isDelicious ? "lekker" : "niet lekker"));

```

In dit voorbeeld: - De gebruiker geeft een fruitsoort in. - We controleren of die soort 'lekker' is op basis van vaste waarden. - `citroen` en `pompelmoes` delen dezelfde `case`.

6.3. Waarom `break`?

Zonder `break` voert C# automatisch ook de volgende cases uit, zelfs als de match al gevonden is. Dit heet **fall-through** en gebeurt alleen als je geen `break` zet. Meestal wil je dit vermijden.

6.4. Gebruik van `default`

Als de ingevoerde waarde **geen enkele `case`** matcht, wordt de `default` uitgevoerd. Dit is vergelijkbaar met de `else` in een `if else`-structuur.

6.5. Wanneer gebruik je `switch`?

- Als je **veel vaste waarden** wil controleren
- Als je één variabele controleert op **gelijkheid**
- Bijvoorbeeld: menu-opties, commando's, getallen met betekenis...

Niet geschikt voor:

- Complexe condities (zoals `number > 10`)
- Meerdere variabelen tegelijk

6.6. In het kort

- Een `switch` is een alternatief voor meerdere `if else`-blokken.
- Je test **gelijkheid** op een waarde.
- Elke `case` eindigt meestal met een `break`.

- Je kan meerdere waarden combineren en een `default` toevoegen.

7. Scope van variabelen

7.1. Wat is scope?

Wanneer je een variabele aanmaakt, bestaat die maar tijdelijk. Ze is enkel zichtbaar **binnen het blok waarin je ze aanmaakt**. Dat bereik noemen we de **scope** van een variabele.

7.2. Blok-scope

Scope wordt bepaald door de accolades `{}`. Als je een variabele in een `if`, of later in een `while` of `for` maakt, bestaat die enkel binnen dat blok.

```
static void Main(string[] args)
{
    int age = 20;

    if (age >= 18)
    {
        string message = "Je bent meerderjarig.";
        Console.WriteLine(message);
    }

    // Console.WriteLine(message); // FOUT! 'message' bestaat hier niet
}
```

De variabele `message` werd aangemaakt **binnen het if-blok**. Buiten dat blok is ze onbekend. Je krijgt dus een fout als je ze daar probeert te gebruiken.

7.3. Scope in `Main`

Een variabele die je rechtstreeks in `Main` aanmaakt, blijft bestaan tot het einde van `Main`.

```
static void Main(string[] args)
{
    int number = 5;
    Console.WriteLine(number); // OK
}
```

Maar als je dezelfde naam probeert te gebruiken binnen een `if`, dan wordt het verwarring:

```
static void Main(string[] args)
{
    int number = 5;

    if (number > 0)
    {
        int number = 10; // FOUT! Je mag 'number' niet opnieuw declareren
    }
}
```

```
}
```

7.4. Waarom is dit belangrijk?

Later in de cursus ga je werken met **loops** en **methodes**. Ook daar gebruik je blokken met `{ }`, en dus gelden dezelfde regels voor scope. Als je een variabele op de verkeerde plaats aanmaakt, dan:

- bestaat ze maar even en verdwijnt ze te snel
- of is ze onbereikbaar op de plaats waar je ze nodig hebt

Daarom is het belangrijk om bij elke variabele goed na te denken **waar** je ze declareert.

7.5. Samenvatting

Regel	Uitleg
Een variabele leeft binnen het blok <code>{ }</code> waarin ze is aangemaakt	Buiten dat blok krijg je een fout als je ze probeert te gebruiken
Je mag geen twee variabelen met dezelfde naam in één scope hebben	De compiler geeft dan een foutmelding
Scope speelt een rol bij condities, loops en methodes	Je leert in elk van die hoofdstukken hoe je er correct mee omgaat

7.6. Oefening

Wat gebeurt er wanneer je dit programma probeert uit te voeren?

```
static void Main(string[] args)
{
    if (true)
    {
        string status = "Actief";
        Console.WriteLine(status);
    }

    Console.WriteLine(status);
}
```

Oplossing

Je krijgt een foutmelding: `status` bestaat niet in de tweede `Console.WriteLine()`. De variabele

`status` werd gedeclareerd binnen het `if`-blok en is dus daarbuiten niet zichtbaar.