

# C# Essentials

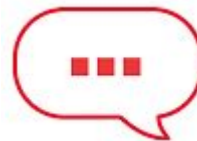
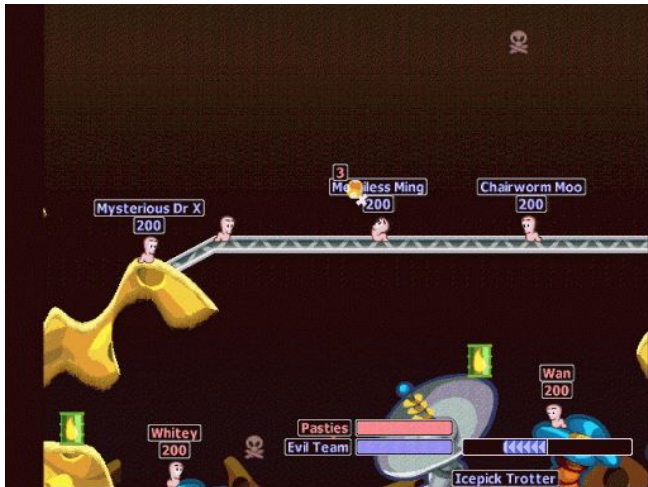
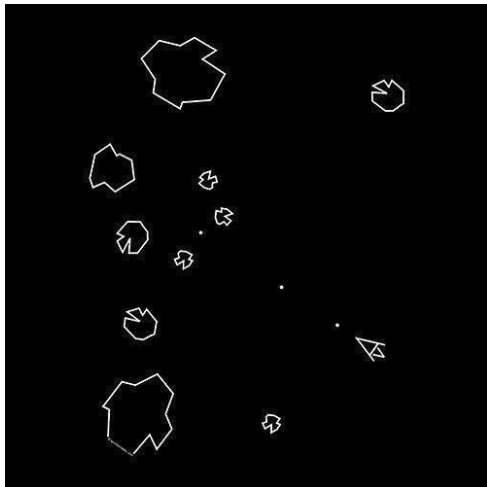
## Arrays

Lector: Tom Quareme


# Voorbeelden

Array: rij of reeks van gegevens (van hetzelfde type)

# Voorbeelden



# Voorbeelden



PLAYLIST

## Blackness and Silence


"All I want is blackness... Blackness, and silence." Intended to be a vast compilation which embraces the melancholy of black metal, ambient, funeral doom and other utterly dark music...

Created by **Tom Quareme** • 133 songs, 17 hr 37 min

**PLAY** ...

Q Filter Do

	TITLE	ARTIST	ALBUM	
♥	Ad Astra	Arcturus	La Masquerade Infer...	2018-11-08
♥	Dunkelheit	Burzum	Filosofem	2018-11-08
♥	Dream Of A Dead Sun	ColdWorld	Melancholie²	2018-11-08
♥	Det Som En Gang Var	Burzum	Hvis Lyset Tar Oss	2018-11-08
♥	Tomhet	Burzum	Hvis Lyset Tar Oss	2018-11-08
♥	Star-Crossed	Arcturus	The Sham Mirrors	2018-11-08

 Puntenadministratie — □ ×

Web Essentials

19

C# Essentials

13

Data Essentials

15

Communication skills

11

IT-organisation

19

C# Advanced

12

Web Advanced

10

Data analysis & SQL

18

C# Mobile

10

Project Management

19

C# Web

10

Security & Privacy

16

71,67%

Berekenen

Wissen

# Probleem

- **Lelijke code...**

```
private static void LelijkeMethod()
{
    float gemiddelde = 0.0f;
    int getal0 = 100;
    int getal1 = 50;
    int getal2 = 20;
    int getal3 = 60;
    int getal4 = 90;
    int getal5 = 80;

    gemiddelde = (getal0 + getal1 + getal2 + getal3 + getal4 + getal5) / 6.0f;
    // Console.WriteLine("Gemiddelde: {0}", gemiddelde);
    Console.WriteLine($"Gemiddelde: {gemiddelde}");
}
```

# Probleem

- **Wat met 10 getallen?**
- **100 getallen?**
- **1000 getallen?**

# Proberen oplossen

- **Lelijke code opschonen**

```
private static void LelijkeMethod()
{
    float gemiddelde = 0.0f;
    // Groepeer tot 1 reeks (array)
    int getal0 = 100;
    int getal1 = 50;
    int getal2 = 20;
    int getal3 = 60;
    int getal4 = 90;
    int getal5 = 80;

    // Teveel herhaling... => gebruik for loop en verander cijfertjes via een index
    gemiddelde = (getal0 + getal1 + getal2 + getal3 + getal4 + getal5) / 6.0f;
    Console.WriteLine("Gemiddelde: {0}", gemiddelde);
}
```

# Oplossing

- **Arrays en lussen!**

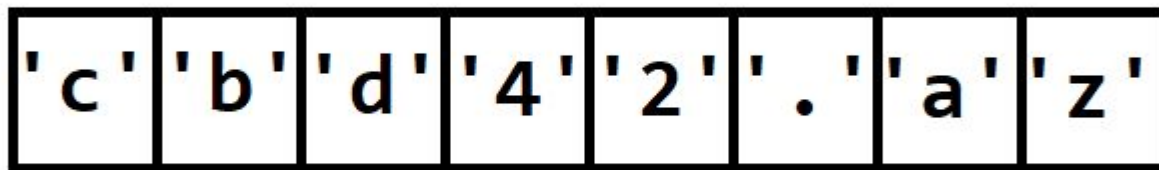
```
private static void MooieMethod()  
{  
    float gemiddelde = 0.0f;  
    int[] getallen = new int[6] { 100, 50, 20, 60, 90, 80 };  
  
    for (int i = 0; i < getallen.Length; i++)  
    {  
        gemiddelde += (float) getallen[i];  
    }  
    gemiddelde /= getallen.Length;  
  
    Console.WriteLine("Gemiddelde: {0}", gemiddelde);  
}
```



# Array

- **Rij of reeks van gegevens van hetzelfde type. Lengte ligt vast.**
  - `char[] letters = new char[8] {'c', 'b', 'd', '4', '2', '.', 'a', 'z'};`

⇒ **Variabelen met index.**



0    1    2    3    4    5    6    7<sub>=length-1</sub>

# Array

- **Rij of reeks van gegevens van hetzelfde type. Lengte ligt vast.**

- `int[] getallen = new int[6] { 100, 50, 20, 60, 90, 80 };`

⇒ **Variabelen met index.**

- `getallen[3]; // 60`
  - `getallen[0]; // 100`
  - `getallen[getallen.Length - 1]; // 80`
  - ~~◦ `getallen[-1];` Waarom?~~
  - ~~◦ `getallen[getallen.Length];` Waarom?~~

- **Gebruik index om variabele van array te lezen:**

- `gemiddelde += getallen[i];`

- **Of aanpassen:**

- `getallen[i] = 100;`

# Array

- **Declaratie:**

```
double[] getallen = new double[7]; // eerste index: 0, laatste index: 6  
string[] namen = new string[9];
```

- **Initialisatie:**

```
string[] namen = new string[5] {"Peter", "Sven", "Piet", "Tom", "Anna"};  
string[] namen = new string[] {"Peter", "Sven", "Piet", "Tom", "Anna"};  
string[] namen = {"Peter", "Sven", "Piet", "Tom", "Anna"};
```

- **Achteraf veranderen:**

```
namen[0] = "Jack";
```

# Array

- **Loopen:**

```
for (int i = 0; i < getallen.Length; i++)  
{  
    Console.WriteLine(getallen[i]);  
}
```

```
foreach (double getal in getallen)  
{  
    Console.WriteLine(getal);  
}
```

- For: als je array element wil lezen of aanpassen.
- Foreach: als je enkel (alle) array elementen wil uitlezen.

# Testoefeningetje

Zoek kleinste en grootste  
getal uit reeks getallen.

Gebruik: arrays, loops (for of foreach)  
Declareer en initialiseer op verschillende manieren

# Array

- **Array returnen uit funciemethode:**

- **De funciemethode gebruiken:**

```
int[] getallen = new int[10];  
getallen = MaakArray();
```

- **De funciemethode:**

```
private int[] MaakArray()  
{  
    int[] getallen = new int[10];  
    for (int i = 0; i < 10; i++)  
    {  
        getallen[i] = i;  
    }  
    return getallen;  
}
```

# Array

- **Array doorgeven als parameter:**

- **De functiemethode gebruiken:**

```
int resultaat = Som(getallen);
```

- **De functiemethode:**

```
private int Som(int[] getallen)
{
    int som = 0;
    foreach (int getal in getallen)
    {
        som += getal;
    }
    return som;
}
```

- **Analoog voor void-procedure.**
- **Gebruik ref om array aan te passen binnen method!**

# Array

- **Wanneer array gebruiken:**
  - Rij nodig van aantal gegevens.
  - Hetzelfde type.
  - Aantal elementen ligt vast.
  - Loopen over gegevens en inlezen of overschrijven.
    - Van begin tot eind.
    - Van eind tot begin.
    - Bepaald segment.
    - Stapgrootte instellen: ( $i+=2$  bijvoorbeeld in plaats van  $i++$ ).



# Testoefeningetje

Zoek kleinste en grootste  
getal uit reeks getallen.

Pas vorige oefening aan: array als parameter, return array  
van kleinste en grootste getal.

# 2D Arrays

- **1D array:** is 1 rij.     [ 'a' ][ 'b' ][ 'c' ][ 'd' ][ 'e' ][ 'f' ]
- **2D array:** tabel van rijen en kolommen.  
    [ 'a' ][ 'b' ][ 'c' ]  
    [ 'd' ][ 'e' ][ 'f' ]
- Bepaal indices?

# 2D Arrays

- **1D array:** is 1 rij.     [ 'a' ][ 'b' ][ 'c' ][ 'd' ][ 'e' ][ 'f' ]
- **2D array:** tabel (of matrix) van rijen en kolommen.  
    [ 'a' ][ 'b' ][ 'c' ]  
    [ 'd' ][ 'e' ][ 'f' ]
- Bepaal indices?
  - 1D: a (0de element), b (1), c (2), d (3), e (4), f (5)
  - 2D: a (0de rij, 0de kolom), b (0, 1), c (0, 2), d (1, 0), e (1, 1), f (1, 2)

# 2D Arrays

- **Declaratie:**

- `int[,]` getallen = `new int[4, 2];` // 4 rijen en 2 kolommen
- `int[,]` getallekes = `new int[2, 4];` // 2 rijen en 4 kolommen

- **Initialisatie:**

```
int[,] getallen = new int[4, 2] { {8, 6}, {9, 1}, {7, 8}, {5, 6} };  
int[,] getallen = new int[, ] { {8, 6}, {9, 1}, {7, 8}, {5, 6} };  
int[,] getallen = { {8, 6}, {9, 1}, {7, 8}, {5, 6} };
```

- **Waarom gaat deze niet?**

```
int[,] getallen = { {8, 6, 4}, {9, 1}, {7, 8, 2, 1}, {5, 6}, {1} };  
  
// Aantal kolommen moet telkens gelijk blijven.
```

# 2D Arrays

- **Loopen:**

```
int aantalRijen = getallen.GetLength(0); // 0 geeft aan: rijen
int aantalKolommen = getallen.GetLength(1); // 1 geeft aan: kolommen
int totaalAantalElementen = getallen.Length;
```

```
for (int r = 0; r < aantalRijen; r++)
{
    for(int k = 0; k < aantalKolommen; k++)
    {
        getallen[r, k] = r + k;
    }
}
```

# Dag 2: Hulpfuncties Arrays

# Hulpfuncties Arrays (Array. ...())

- **IndexOf(): Index van eerste voorkomen**

```
string[] namen = new string[] {"Peter", "Tom", "Piet", "Tom", "Benny", "Anna"};  
int index = Array.IndexOf(namen, "Tom");
```

```
Console.WriteLine(index); // 1
```

```
// Index Piet?
```

```
// Index Jef? ⇒ -1 (als element niet gevonden wordt)
```

```
// Index laatste voorkomen van Tom?
```

# Hulpfuncties Arrays

- **LastIndexOf():** Index van laatste voorkomen

```
string[] namen = new string[] {"Peter", "Tom", "Piet", "Tom", "Benny", "Anna"};  
int index = Array.LastIndexOf(namen, "Tom");
```

```
Console.WriteLine(index); // 3
```



# Hulpfuncties Arrays

- **Copy(): Array kopiëren**

```
string[] namen = {"Peter", "Sven", "Piet", "Tom", "Benny", "Anna"};  
string[] kopie = new string[namen.Length];  
Array.Copy(namen, kopie, namen.Length);
```

```
foreach (string naam in kopie)  
{  
    Console.WriteLine(naam);  
}
```

```
// Peter  
// Sven  
// Piet  
// Tom  
// Benny  
// Anna
```

# Hulpfuncties Arrays

- **Reverse(): Array omkeren**

```
string[] namen = new string[] {"Peter", "Sven", "Piet", "Tom", "Benny", "Anna"};  
Array.Reverse(namen);
```

```
foreach (string naam in namen)  
{  
    Console.WriteLine(naam);  
}
```

```
// Anna  
// Benny  
// Tom  
// Piet  
// Sven  
// Peter
```

# Hulpfuncties Arrays

- **Find(): Zoek element**

```
string[] consoles = new string[] { "PS2", "XBox", "Dreamcast", "N64", "Gamecube", "PS5"};

// Zoekt eerste element dat begint met N.
string console1 = Array.Find(consoles, element => element.StartsWith("N"));
Console.WriteLine(console1); // N64
```

# Hulpfuncties Arrays

- **Find(): Zoek element**

```
string[] consoles = new string[] { "PS2", "XBox", "Dreamcast", "N64", "Gamecube", "PS5"};

// Zoekt eerste element dat begint met N.
string console1 = Array.Find(consoles, element => element.StartsWith("N"));
Console.WriteLine(console1); // N64

// Zoekt eerste element dat 9 karakters heeft.
string console2 = Array.Find(consoles, element => (element.Length == 9));
Console.WriteLine(console2); // Dreamcast
```

# Hulpfuncties Arrays

- **FindAll(): Zoek alle elementen**

```
string[] consoles = new string[] { "PS2", "XBox", "Dreamcast", "N64", "Gamecube", "PS5"};

// Zoekt alle elementen die beginnen met een P.
string[] pConsoles = Array.FindAll(consoles, element => element.StartsWith("P"));

foreach (string console in pConsoles)
{
    Console.WriteLine(console);
}
// PS2
// PS5

// Hoe alle elementen zoeken van lengte 4 of langer?
```

# Hulpfuncties Arrays

- **FindAll(): Zoek alle elementen**

```
string[] consoles = new string[] { "PS2", "XBox", "Dreamcast", "N64", "Gamecube", "PS5"};

// Zoekt alle elementen die beginnen met een P.
string[] pConsoles = Array.FindAll(consoles, element => element.StartsWith("P"));

foreach (string console in pConsoles)
{
    Console.WriteLine(console);
}
// PS2
// PS5

// Hoe alle elementen zoeken van lengte 4 of langer?
string[] langeConsoles = Array.FindAll(consoles, element => (element.Length >= 4));
```

# Hulpfuncties Arrays

- **Exists(): Bestaat het element? (true/false)**

```
string[] consoles = new string[] { "PS2", "XBox", "Dreamcast", "N64", "Gamecube", "PS5"};

bool bestaat = Array.Exists(consoles, element => element.Equals("Phantom"));

Console.WriteLine(bestaat);
// false

// Hoe kijken of er een console is met 3 karakters?
```

# Hulpfuncties Arrays

- **Exists(): Bestaat het element? (true/false)**

```
string[] consoles = new string[] { "PS2", "XBox", "Dreamcast", "N64", "Gamecube", "PS5"};
```

```
bool bestaat = Array.Exists(consoles, element => element == "Phantom");
```

```
Console.WriteLine(console);
```

```
// false
```

```
// Hoe kijken of er een console is met 3 karakters?
```

```
bool bestaat = Array.Exists(consoles, element => (element.Length == 3));
```



# Hulpfuncties Arrays

- **Sort(): Sorteren van klein naar groot**

```
string[] namen = new string[] {"Peter", "Sven", "Piet", "Tom", "Benny", "Anna"};  
Array.Sort(namen);
```

# Hulpfuncties Arrays

- **Sort(): Sorteren van klein naar groot**

```
string[] namen = new string[] {"Peter", "Sven", "Piet", "Tom", "Benny", "Anna"};  
Array.Sort(namen);
```

```
foreach (string naam in namen)  
{  
    Console.WriteLine(naam);  
}
```

```
// Anna  
// Benny  
// Peter  
// Piet  
// Sven  
// Tom
```

# Hulpfuncties Arrays

- **Sorteren van groot naar klein?**

```
string[] namen = new string[] {"Peter", "Sven", "Piet", "Tom", "Benny", "Anna"};  
// ???
```

# Hulpfuncties Arrays

- **Sorteren van groot naar klein?**

```
string[] namen = new string[] {"Peter", "Sven", "Piet", "Tom", "Benny", "Anna"};  
Array.Sort(namen);  
Array.Reverse(namen);
```

```
foreach (string naam in namen)  
{  
    Console.WriteLine(naam);  
}
```

```
// Tom  
// Sven  
// Piet  
// Peter  
// Benny  
// Anna
```

# Hulpfuncties Arrays

- **Sort(): Bepaald gedeelte van array sorteren**

```
string[] namen = new string[] {"Peter", "Sven", "Piet", "Tom", "Benny", "Anna"};
int startIndex = 1;
int length = 4;
Array.Sort(namen, startIndex, length);

foreach (string naam in namen)
{
    Console.WriteLine(naam);
}

// ???
```

# Hulpfuncties Arrays

- **Sort(): Bepaald gedeelte van array sorteren**

```
string[] namen = new string[] {"Peter", "Sven", "Piet", "Tom", "Benny", "Anna"};
int startIndex = 1;
int length = 4;
Array.Sort(namen, startIndex, length);

foreach (string naam in namen)
{
    Console.WriteLine(naam);
}

// {"Sven", "Piet", "Tom", "Benny"} =>
// Peter
// Benny
// Piet
// Sven
// Tom
// Anna
```

# Hulpfuncties Arrays

- **BinarySearch(): SNEL element zoeken in AL GESORTEERDE array**

```
string[] namen = new string[] {"Peter", "Sven", "Piet", "Tom", "Benny", "Anna"};  
Array.Sort(namen);  
// Anna Benny Peter Piet Sven Tom
```

```
int index = Array.BinarySearch(namen, "Tom");  
Console.WriteLine(index);  
// 5
```

```
// Waarom AL op voorhand gesorteerd?
```

# Hulpfuncties Arrays

- **BinarySearch():** SNEL element zoeken in **AL GESORTEERDE** array

```
string[] namen = new string[] {"Peter", "Sven", "Piet", "Tom", "Benny", "Anna"};  
Array.Sort(namen);  
// Anna Benny Peter Piet Sven Tom
```

```
int index = Array.BinarySearch(namen, "Sven");  
Console.WriteLine(index);  
// 4
```

- **Binair zoeken:** telkens kijken in de juiste helften
- Voorbeeld: zoeken naar Sven

```
// Anna Benny Peter Piet Sven Tom  
// Anna Benny Peter [Piet Sven Tom]  
// Anna Benny Peter [Piet Sven] Tom  
// Anna Benny Peter Piet Sven Tom
```



# Hulpfuncties Arrays

- **Clear(): Array leegmaken**

```
// 1D Array leegmaken  
float[] getallen = { 1.1f, 1.2f, 1.3f, 1.4f, 1.5f, 1.6f, 1.7f };  
int startIndex = 2;  
int length = 4;  
Array.Clear(getallen, startIndex, length);  
// { 1.1f, 1.2f, 0.0f, 0.0f, 0.0f, 0.0f, 1.7f }
```

```
// Hoe volledig leegmaken?
```

# Hulpfuncties Arrays

- **Clear(): Array leegmaken**

```
// 1D Array leegmaken  
float[] getallen = { 1.1f, 1.2f, 1.3f, 1.4f, 1.5f, 1.6f, 1.7f };  
int startIndex = 2;  
int length = 4;  
Array.Clear(getallen, startIndex, length);  
// { 1.1f, 1.2f, 0.0f, 0.0f, 0.0f, 0.0f, 1.7f }
```

```
// Hoe volledig leegmaken?  
int startIndex = 0;  
int length = getallen.Length;  
Array.Clear(getallen, startIndex, length);
```

# Hulpfuncties Arrays

- **Clear(): Array leegmaken**

```
// 2D Array leegmaken
float[,] getallen = { {1.1f, 1.2f}, {1.3f, 1.4f}, {1.5f, 1.6f}, {1.7f, 1.8f} };
int startIndex = 3;
int length = 4;
Array.Clear(getallen, startIndex, length);

// Wat is het resultaat?
```

# Hulpfuncties Arrays

- **Clear(): Array leegmaken**

```
// 2D Array leegmaken
float[,] getallen = { {1.1f, 1.2f}, {1.3f, 1.4f}, {1.5f, 1.6f}, {1.7f, 1.8f} };
int startIndex = 3;
int length = 4;
Array.Clear(getallen, startIndex, length);
foreach (float getal in getallen)
{
    Console.WriteLine(getal);
}
```

```
// Wat is het resultaat?
{ {1.1f, 1.2f}, {1.3f, 0.0f}, {0.0f, 0.0f}, {0.0f, 1.8f} }
```

# Hulpfuncties Arrays

- **Resize(): Grootte van array veranderen (ga je niet gauw gebruiken)**

```
// C# alloceert nieuwe array en kopieert inhoud over
```

```
int[] getallen = { 3, 4, 5, 6 };
```

```
// Groter maken
```

```
Array.Resize(ref getallen, 10); // 3 4 5 6 0 0 0 0 0 0
```

```
// Kleiner maken
```

```
Array.Resize(ref getallen, 3); // 3 4 5
```

```
// Automatische resize: gebruik List<T> uit System.Collections.Generic!
```

```
// (zie volgende les!)
```

```
List<int> getallen = new List<int>() {3, 4, 5, 6};
```

```
getallen.Add(0);
```

```
// ...
```