1. Array principes

H8. Arrays

Beeld je in....

- Stel, je moet de scores van 11 voetbalwedstrijden invoeren:

```
int score1, score2, score3, score4, score5, score6, score7
    score8, score9, score10, score11 ;

Console.WriteLine("Enter the scores ");
string score1String = Console.ReadLine();
int score1 = int.Parse(score1String);
string score2String = Console.ReadLine();
int score2 = int.Parse(score2String);
string score3String = Console.ReadLine();
int score3 = int.Parse(score3String);
string score4String = Console.ReadLine();
int score4 = int.Parse(score4String);
string score5String = Console.ReadLine();
int score5 = int.Parse(score5String);
string score6String = Console.ReadLine();
int score6 = int.Parse(score6String);
string score7String = Console.ReadLine();
int score7 = int.Parse(score7String);
string score8String = Console.ReadLine();
int score8 = int.Parse(score8String);
string score9String = Console.ReadLine();
int score9 = int.Parse(score9String);
string score10String = Console.ReadLine();
int score10 = int.Parse(score10String);
string score11String = Console.ReadLine();
int score11 = int.Parse(score11String);
```

Maar wat als het 100 wedstrijden zijn?
Of 1000?
Of 1 miljoen?!

Beeld je in...

- Je hebt een lijst van alle Steam-spelers nodig
- Een lijst van alle dagelijkse temperatuurmetingen van de vorige eeuw
- Quiz antwoorden
- Een DNA sequentie van het menselijk genoom
- Etc etc etc

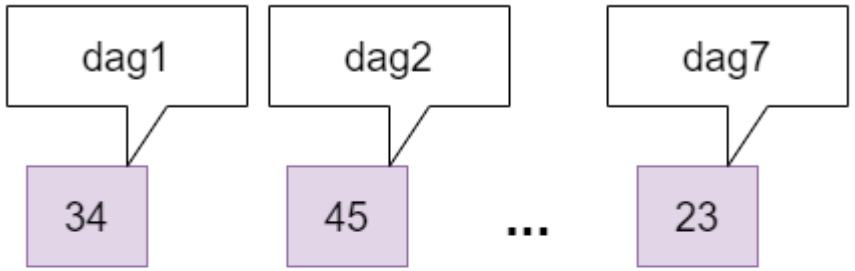
Arrays to the rescue

Via arrays kunnen we een grote hoeveelheid data op eenvoudige manier gebruiken.

Met vs zonder

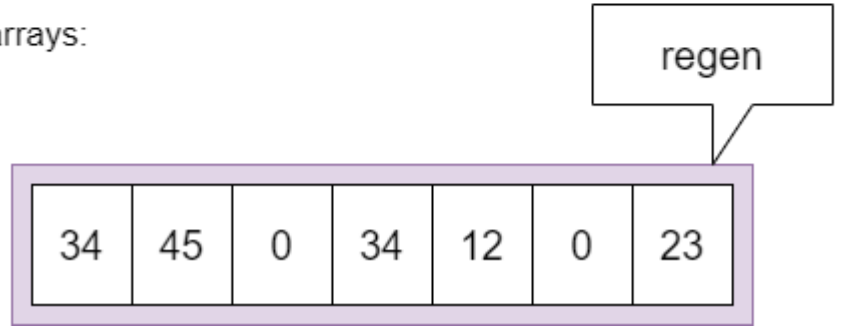
```
int dag1 = 34;
int dag2 = 45;
int dag3 = 0;
int dag4 = 34;
int dag5 = 12;
int dag6 = 0;
int dag7 = 23;
```

Zonder arrays:

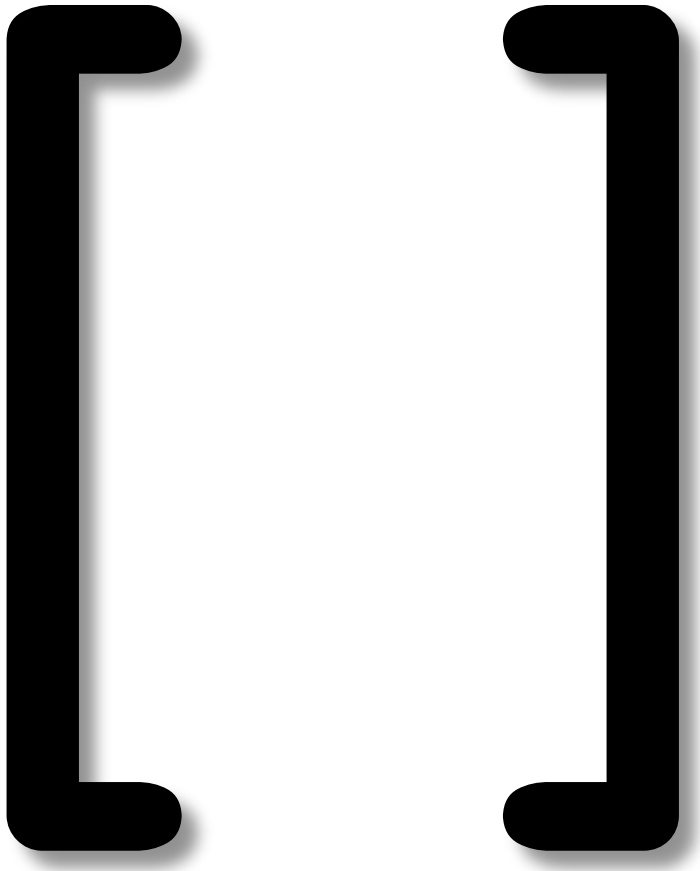



```
int[] regen  = {34, 45, 0, 34, 12, 0, 23};
```

Met arrays:



Vierkante haken in C# == arrays incoming



Array

Lijst van waarden die allemaal hetzelfde **same data type** en **naam** hebben

Ieder apart element wordt geïdentificeerd m.b.v. een **index**

Arrays aanmaken

- Zelfde als we al kenden, maar nu met vierkante haken achter het type:

```
type[] variabelenaam
```

Voorbeelden:

```
int[] getallen;
```

Lees als: “array van ints, genaamd getallen”

```
float[] gewichten;
```

Lees als: “array van floats, genaamd gewichten”

Alle datatypes kunnen als array type gebruikt worden

- **“If it’s a *type*, it can be an array”**

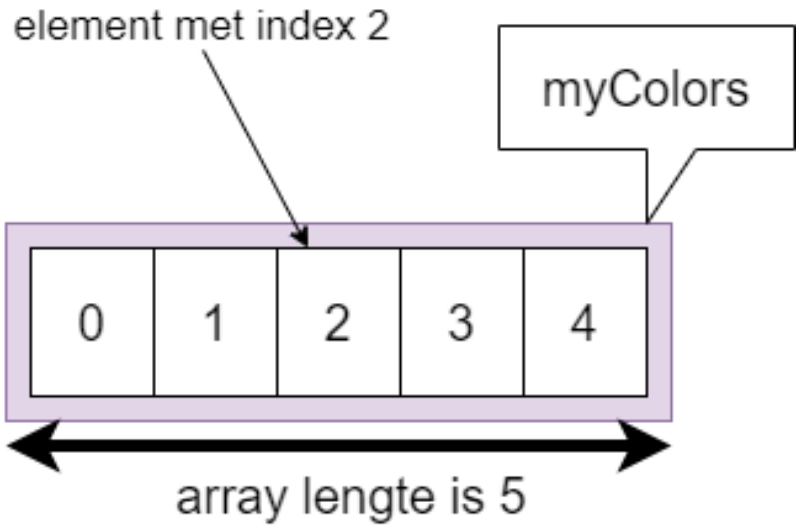
```
string[] Zinnen;

bool[] AntwoordenQuiz;

ConsoleColor[] LievelingsKleuren;

Etc.
```

Array aanmaken: lengte vereist

```
string[] myColors;
myColors = new double[5];
```

"You must provide a size once you initialize"

(*Poëzie met Tim*)



Array initialization met gekende waarden:

- Nuttige syntax indien je al weet wat er in array moet. Lengte hoef je dan niet mee te geven.
- Volgende manieren zijn identiek, kies wat je handigste vindt:

```
int[] myScores = new int[5] { 100, 76, 88, 100, 90 };
int[] myScores = new int[] { 100, 76, 88, 100, 90 };
int[] myScores = { 100, 76, 88, 100, 90 };
```

De lengte van array kan ook dynamisch ontstaan

Array lengte hoeft niet hardcoded te zijn:

```
double[] sales;

Console.WriteLine("How many sales did you make this year ?");

int sizereq = Convert.ToInt32(Console.ReadLine());

sales = new double[sizereq];
```

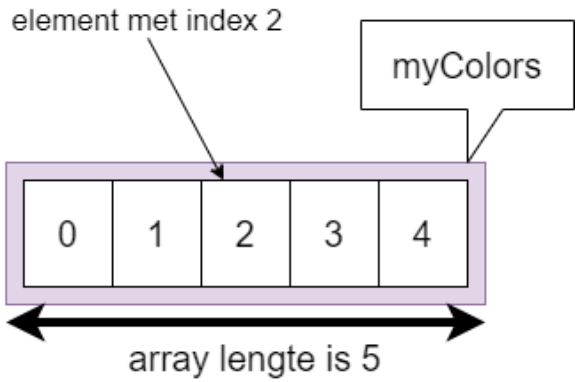
Arrays kunnen niet groeien

- Van zodra je een array een lengte hebt gegeven is deze vast

- Wat als je array moet groeien of krimpen?
  - Maak een nieuwe array en kopiëer de waarden die je nodig hebt.

Array elementen

- De indexering bij arrays gebruik je om individuele elementen te bereiken. **Telt vanaf 0**



0 1 2 3 4 5

« Eentje is geentje »

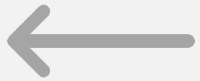
Lengte array = 6

Laatste pintje = index 5

Waarden schrijven en uitlezen

Schrijven

```
sales[5] = 2100.00;
```



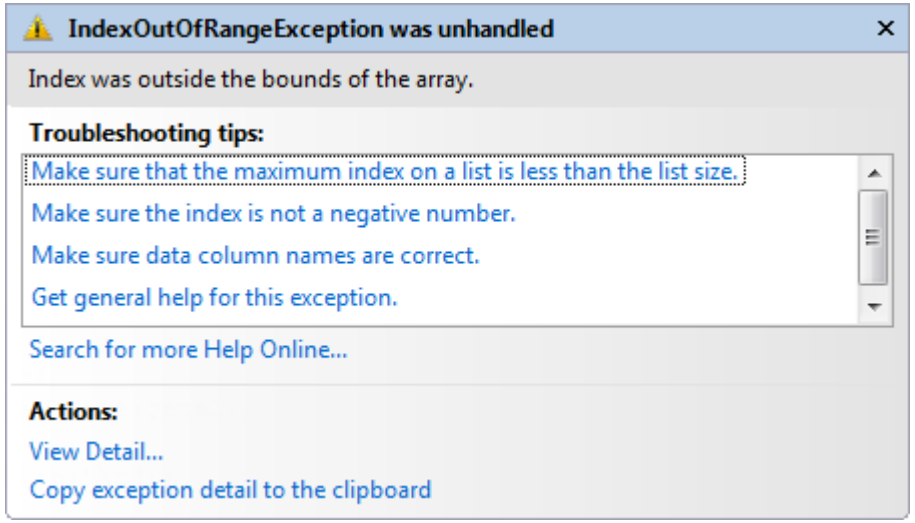
Lezen

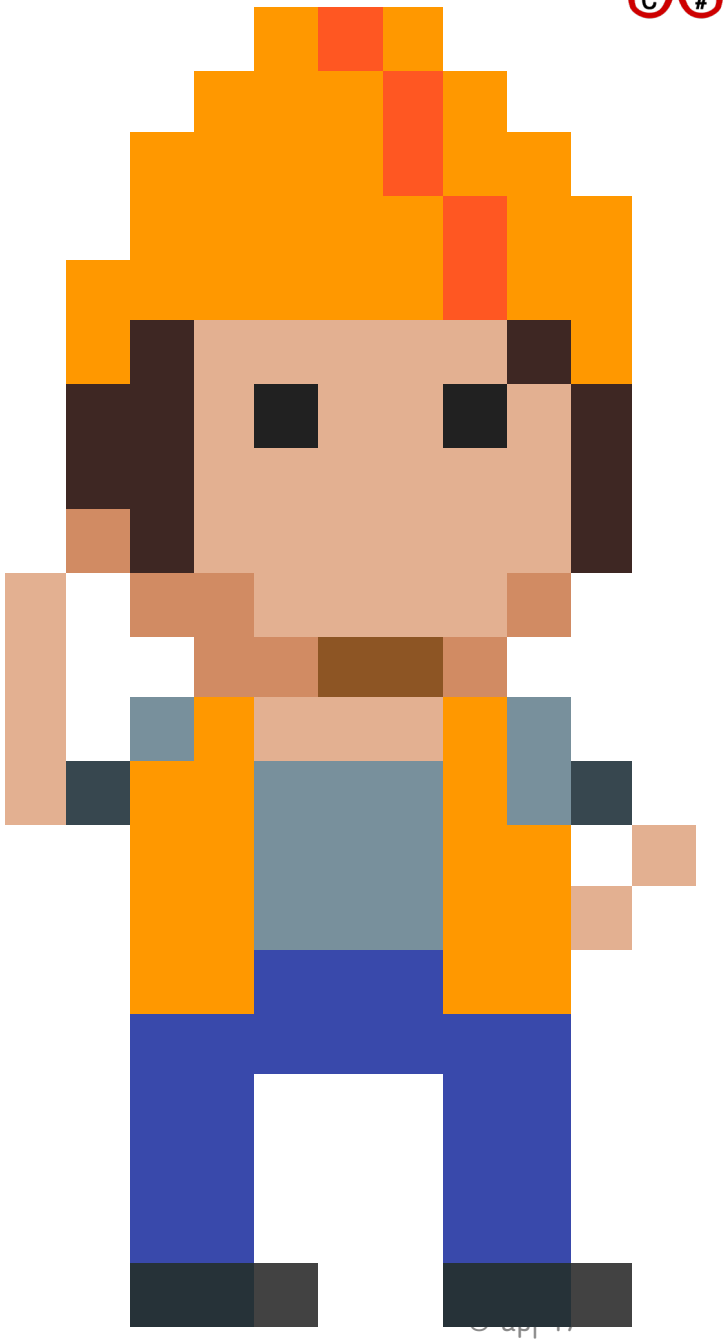
```
double mySale= sales[16];
```

“Out of range exception”

Veel gemaakte fout!

Je laatste element heeft index *lengte-1*





Elementen benaderen met loops

- Arrays en loops zijn dikke vrienden
  - Voorbeeld om de eerste 5 elementen telkens met 3 te verhogen:

```
for (int sub = 0; sub < 5; sub++)
    myScores[sub] += 3;
```

.Length property

- Alle arrays hebben de .Length eigenschap.
- Zeer handig om lengte van array te weten.

- Vaak gebruikt in loop om alle elementen van array te benaderen

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

Een volledig programma

- Test zelf!

```
//Array aanmaken
int[] getallen = new int[100];
//Array vullen
for (int i = 0; i < getallen.Length; i++)
{
    getallen[i] = i;
}
//Alle elementen met 3 vermenigvuldigen
for (int i = 0; i < getallen.Length; i++)
{
    getallen[i] = getallen[i] * 3;
}
//Enkel veelvouden van 4 op het scherm tonen
for (int i = 0; i < getallen.Length; i++)
{
    if(getallen[i] % 4 == 0)
        Console.WriteLine(getallen[i]);
}
```