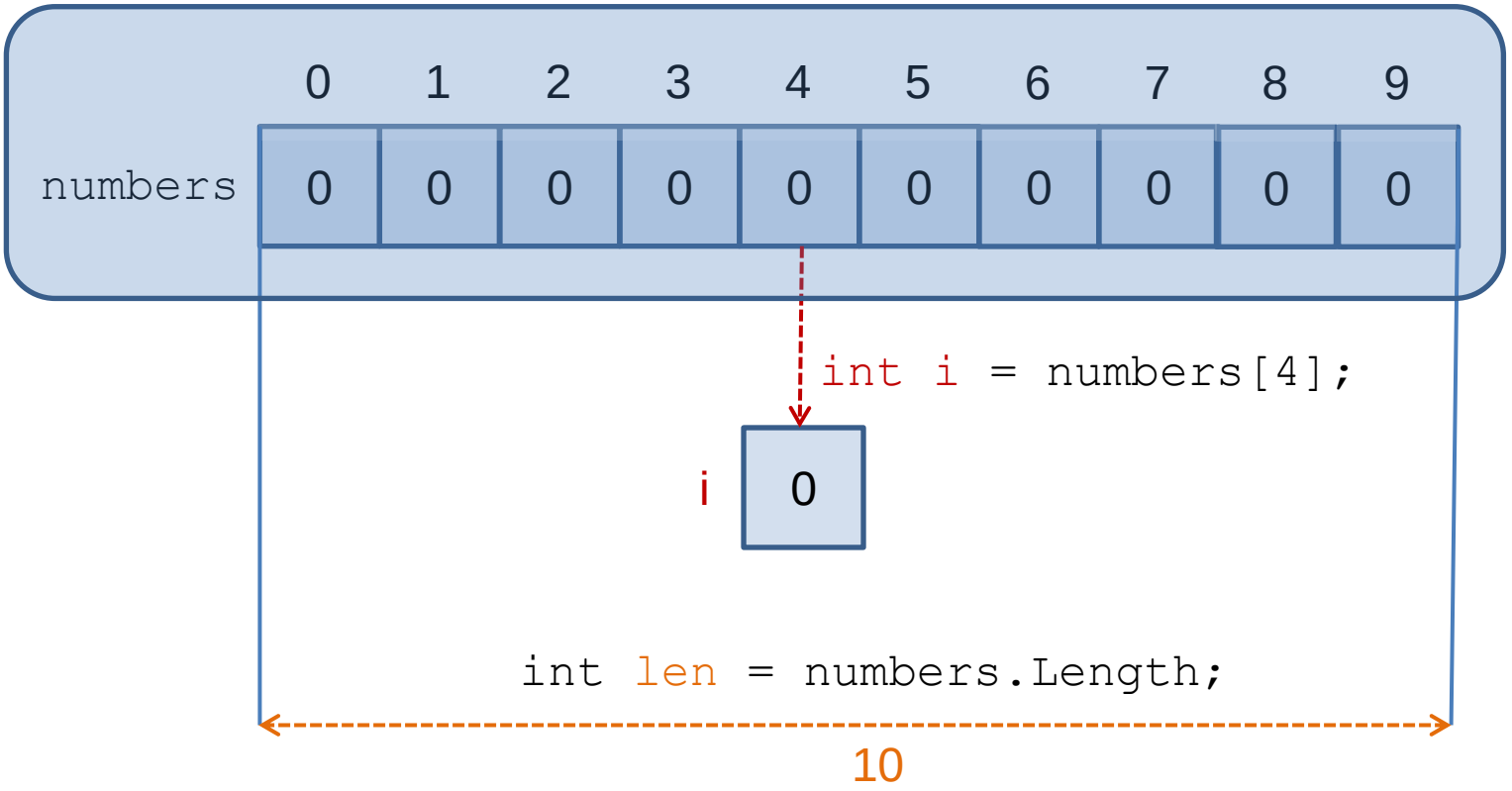
Eindimensionale Arrays / Felder

Arrays, die auch als Datenfelder bezeichnet werden, ermöglichen es, eine nahezu beliebig große Anzahl von Variablen gleichen Namens und gleichen Datentyps zu definieren.
Unterschieden werden die einzelnen Elemente nur anhand einer Indizierung.

```
int[] numbers = new int[10];
```

HTL LEONDING

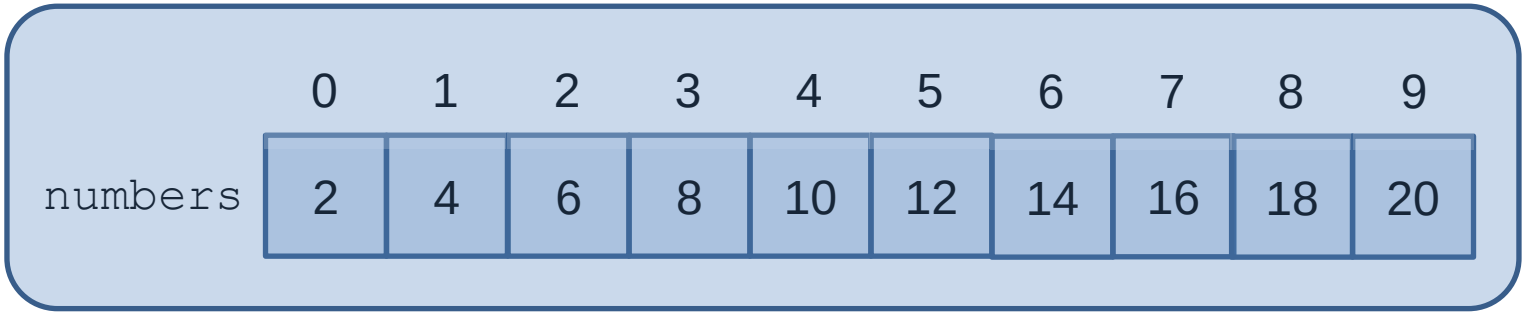
Eindimensionale Arrays



HTL LEONDING

Eindimensionale Arrays

```
int[] numbers = new int[10];

for (int i=0; i<number.Length; i++)
{
   numbers[i] = 2 * (i + 1);
}
```



HTL LEONDING

Eindimensionale Arrays

Arrays erlauben es, idente Operationen auf mehrere Variablen in einer Schleife durchzuführen.
Um 10 Zahlen einzulesen und zu speichern kann nun eine Schleife verwendet werden:

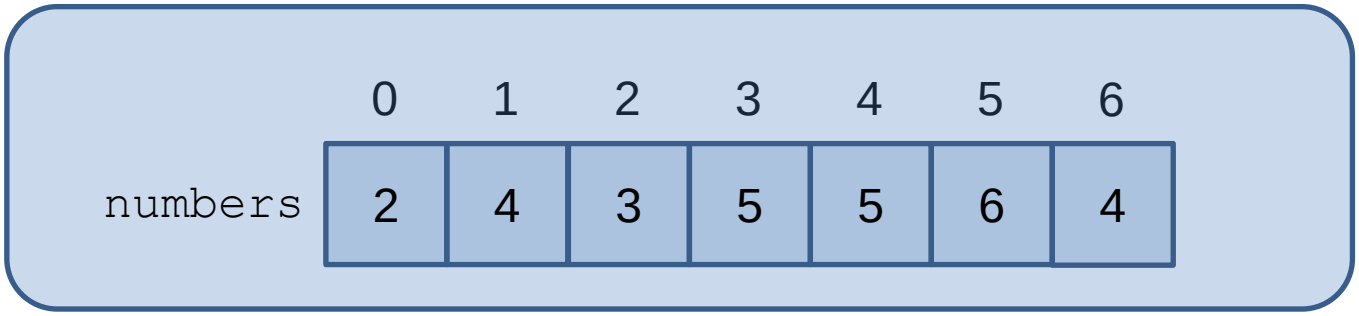
```
int[] numbers = new int[10];

for (int i = 0; i < number.Length; i++)
{
   numbers[i] = int.Parse(Console.ReadLine());
}
```

Wie würde das Einlesen von 10 Zahlen ohne Array aussehen?

Eindimensionale Arrays

Arrays Initialisierung direkt bei der Deklaration

```
int[] numbers = new int[] { 2, 4, 3, 5, 5, 6, 4 };
int[] numbers = { 2, 4, 3, 5, 5, 6, 4 };
var numbers   = new [] { 2, 4, 3, 5, 5, 6, 4 };
```




```
int sum = 0;
for(int i =0;i<numbers.Length;i++)
{
    sum += numbers[i];
}
Console.WriteLine($"Summe aller Zahlen {sum}");
```


HTL LEONDING

Eindimensionale Arrays - foreach

C# foreach

Diese Schleifen-Konstruktion kann verwendet werden:

- Wenn der Index nicht benötigt wird.
- Wenn nur „lesend“ auf die Elemente des Arrays zugegriffen wird.
  Ein Zuweisen der Array-Elemente ist nicht möglich.

```
var numbers   = { 2, 4, 3, 5, 5, 6, 4 };

int sum = 0;

foreach (int number in numbers)
{
    sum += number;
}
Console.WriteLine($"Summe aller Zahlen {sum}");
```