



# Programmeren 2

# Programma periode 1.2

01 (wk-46)	Enumeraties / structures / classes
02 (wk-47)	2-dim arrays / Flow Control
03 (wk-48)	Lists / Dictionaries
04 (wk-49)	File I/O / error handling
05 (wk-50)	opbouw / structuur
06 (wk-51)	opbouw / structuur
07 (wk-52)	kerstvakantie
08 (wk-53)	kerstvakantie
<hr/>	
09 (wk-01)	herhaling/vragen/oefententamen
10 (wk-02)	<i>tentamens</i>
11 (wk-03)	<i>herkansingen</i>
12 (wk-04)	<i>herkansingen</i>

# 2-dimensional arrays

# 1-dimensional arrays

- één index-getal: 0 .. lengte - 1

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

    Random rnd = new Random();
    for (int i = 0; i < 20; i++)
    {
        getallen[i] = rnd.Next(1, 101);
    }
}
```

Array met int-waarden,  
dus gehele getallen.

Array met float-waarden  
(komma getallen) en array  
met bool-waarden  
(true/false).

één index-getal nodig.

```
static void Main(string[] args)
{
    float[] maandOmzet = new float[12];
    bool[] maandWinst = new bool[12];

    // ...
}
```

# 1-dimensional arrays

## ■ Length (property)

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

    Random rnd = new Random();
    for (int i = 0; i < getallen.Length; i++)
    {
        getallen[i] = rnd.Next(1, 101);
    }
}
```

Met property 'Length' kunnen we het (totaal) aantal elementen in een array bepalen.

# 2-dimensional arrays

- 2 index-getallen: rij en kolom

```
static void Main(string[] args)
{
    int[,] matrix = new int[5, 5]; // [aantal rijen, aantal kolommen]
    Random rnd = new Random();

    // doorloop alle rijen
    for (int r = 0; r < 5; r++)
    {
        // (binnen elke rij) doorloop alle kolommen
        for (int k = 0; k < 5; k++)
        {
            matrix[r, k] = rnd.Next(1, 101);
        }
    }
}
```

2 index-getallen, r (rij)  
en k (kolom).

*volgorde van verwerking*

1	2	3	4	5	r=0
6	7	8	9	10	r=1
11	12	13	14	15	r=2
16	17	18	19	20	r=3
21	22	23	24	25	r=4
k=0	k=1	k=2	k=3	k=4	

# 2-dimensional arrays

## ■ GetLength (methode)

```
static void Main(string[] args)
{
    int[,] matrix = new int[5, 5]; // [aantal rijen, aantal kolommen]
    Random rnd = new Random();

    // doorloop alle rijen
    for (int r = 0; r < matrix.GetLength(0); r++)
    {
        // (binnen elke rij) doorloop alle kolommen
        for (int k = 0; k < matrix.GetLength(1); k++)
        {
            matrix[r, k] = rnd.Next(1, 101);
        }
    }
}
```

Met methode 'GetLength(...)' kunnen we de grootte/length van een dimensie (1<sup>ste</sup>, 2<sup>de</sup>, ...) bepalen.

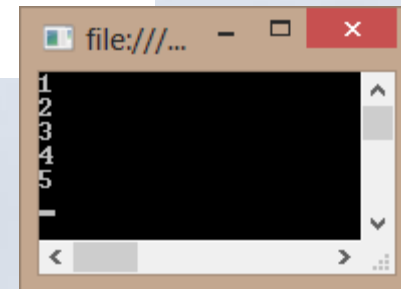
# Flow Control



# break – stop lus


*Dit stopt  
de while-  
loop.*

```
// break
int i = 1;
while (i <= 10)
{
    if (i == 6)
        break;
    Console.WriteLine("{0}", i++);
}
Console.WriteLine();
```

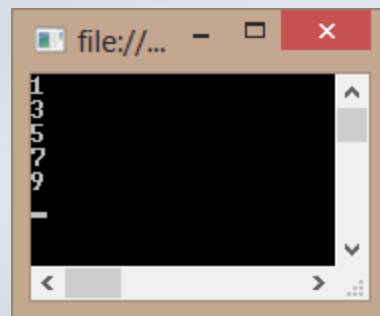


# continue – spring naar begin lus

```
// continue
for (i = 1; i <= 10; i++)
{
    if ((i % 2) == 0)
        continue;
    Console.WriteLine(i);
}
Console.WriteLine();
```



Dit stopt de huidige iteratie en start de volgende. We blijven dus in de lus.



# return – stop huidige methode

```
// return  
for (i = 1; i <= 10; i++)  
{  
    if ((i % 2) == 0)  
        return;  
    Console.WriteLine(i);  
}  
Console.WriteLine();
```

Een return zorgt er voor dat de rest van de methode overgeslagen wordt.

# return – stop huidige methode

```
bool IsPrimeNumber(int number)
{
    if (number < 2) return false;

    bool isPrime = true;
    int i = 2;
    while ((i < number) && (isPrime))
    {
        if ((number % i) == 0)
            isPrime = false;
        else
            i++;
    }
    return isPrime;
}
```

*We kunnen de IsPrimeNumber  
iets efficiënter implementeren.*

```
bool IsPrimeNumber(int number)
{
    if (number < 2) return false;

    int i = 2;
    while (i < number)
    {
        if ((number % i) == 0)
            return false;
        i++;
    }
    return true;
}
```

*Maar... gebruik niet teveel return-  
statements in 1 methode!!*

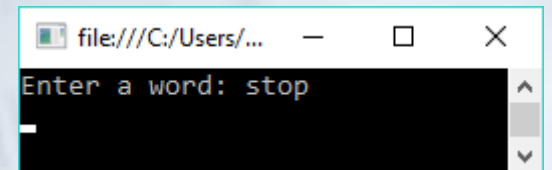
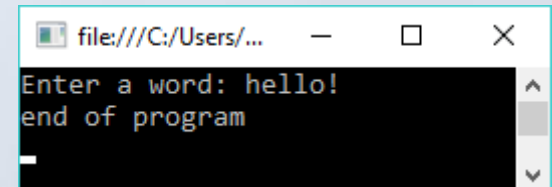
# return – stop huidige methode

```
static void Main(string[] args)
{
    Console.Write("Enter a word: ");
    string input = Console.ReadLine();

    if (input == "stop")
    {
        return;
    }

    Console.WriteLine("end of program");
    Console.ReadKey();
}
```

*Een return in de main-methode stopt het programma.*



# break – stop omsluitende lus

```
int[,] matrix = new int[8, 10]; // 8 rows, 10 columns
Random rnd = new Random();

for (int row = 0; row < matrix.GetLength(0); row++)
{
    for (int col = 0; col < matrix.GetLength(1); col++)
    {
        int number = rnd.Next(1, matrix.Length + 1);
        matrix[row, col] = number;
        Console.Write("{0,3} ", number);

        if (number == (row * 10 + col + 1))
            break;
    }
    Console.WriteLine();
}
```

Dit stopt  
alleen de  
'huidige rij'.

4	63	48	69	71	79	25	39	37	67
11	8	77	35	3	25	27	54	52	50
25	70	77	46	39	48	17	45	30	34
15	9	61	35	57	59	46	28	51	12
56	68	54	1	63	2	1	55	57	77
23	23	2	36	2	14	66	2	29	22
62	4	63	48	69	71	79	25	39	37
67	11	8	77	35	3	25	27	54	52

Op welke plekken stopt de rij???

# Stoppen van een geneste lus

- Stel we zoeken een getal in een matrix (2-dimensionale array); hoe kunnen we, zodra het getal gevonden is, zowel de binnenste als de buitenste lus afbreken?

```
for row = 0 to number_of_rows - 1
    for col = 0 to number_of_columns - 1
        if matrix[row, col] = number
            // found, stop both loops...
        end_if
    end_for
end_for
// we want to continue here...
```

# Verkeerd gebruik van break

```
bool IsNumberPresent(int number, int[] numbers)
{
    bool found = false;
    int i = 0;
    while (i < numbers.Length)
    {
        if (numbers[i] == number)
        {
            found = true;
            break;
        }
        else
        {
            i++;
        }
    }
    return found;
}
```

*Na een break is een  
else-block niet nodig...*



# Verkeerd gebruik van continue

```
void PrintEvenNumbers(int[] numbers)
{
    for (int i = 0; i < numbers.Length; i++)
    {
        if (numbers[i] % 2 == 0)
        {
            Console.WriteLine("{0} ", numbers[i]);
        }
        else
        {
            continue;
        }
    }
}
```

*continue aan het  
einde van een lus  
is zinloos!!*

# Debug - demo

- Breakpoint
- Step into
- Step over
- Watches

The screenshot shows a C# code editor with a breakpoint set at line 31. The code is as follows:

```
17 void Start()  
18 {  
19     int[,] matrix = new int[8, 10]; // 8 rows, 10 columns  
20     Random rnd = new Random();  
21  
22     for (int row = 0; row < matrix.GetLength(0); row++)  
23     {  
24         for (int col = 0; col < matrix.GetLength(1); col++)  
25         {  
26             int number = rnd.Next(1, 80);  
27             matrix[row, col] = number;  
28             Console.Write("{0,3} ", number);  
29  
30             if (number == (row * 10 + col + 1))  
31                 break;  
32         }  
33         Console.WriteLine();  
34     }  
35  
36     Console.ReadKey();  
37 }  
38 }
```

A breakpoint is set at line 31, and the execution has paused. The Locals window shows the following variables and their values:

Name	Value
matrix	{int[8, 10]}
rnd	{System.Random}
row	4
col	6
number	47

# Command line arguments

# Command line arguments

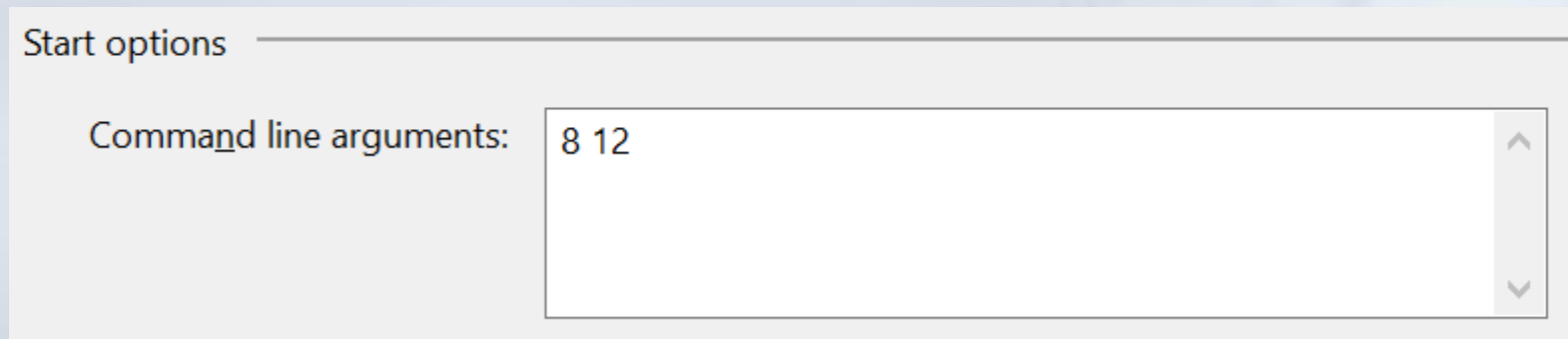
- Om onze programma's flexibeler te maken, kunnen we ze input geven via "Command line arguments"
- Een voorbeeld: we willen dat een programma met een 2 dimensionale array werkt met een willekeurig aantal rijen en kolommen. *(bv 4 rijen en 8 kolommen, of 9 rijen en 6 kolommen, of ...)*

```
void Start()
{
    int[,] matrix = new int[8, 12];
    // ...
}
```

Steeds als we een ander aantal rijen of een ander aantal kolommen willen gebruiken, moeten we deze code aanpassen...

# Command line arguments

- In Visual Studio kunnen we input geven aan een programma via zogenaamde "command line arguments"
- Ga naar de properties van het programma (rechtermuis klik op het project | Properties) en selecteer menu-item Debug
- Vul de arguments in (voorbeeld hieronder heeft 2 argumenten: 8 en 12)



Start options

Command line arguments: 8 12

# Command line arguments

```
static void Main(string[] args)
{
    if (args.Length != 2)
    {
        Console.WriteLine("invalid number of arguments!");
        Console.WriteLine("usage: assignment <nrOfRows> <nrOfColumns>");
        return;
    }

    int nrOfRows = int.Parse(args[0]);
    int nrOfColumns = int.Parse(args[1]);

    Program myProgram = new Program();
    myProgram.Start(nrOfRows, nrOfColumns);
}

void Start(int nrOfRows, int nrOfColumns)
{
    int[,] matrix = new int[nrOfRows, nrOfColumns];

    // your code here...
}
```

Dit programma verwacht 2 argumenten.

We kunnen de 2 command line argumenten aan de Start method doorgeven, ...

... en ze gebruiken om de 2-dimensionale array aan te maken.

# Huiswerk

- Bestudeer de aangegeven paragrafen uit het 'Yellow Book' (zie Moodle)
- Week 2 opdrachten (zie Moodle)