

# 1.Methoden

H7. Methoden



# Methoden

- Manier om code éénmalig te typen en veelvuldig te (her)gebruiken.
- Ook wel functies genoemd (Method=.NET benaming)

## Soorten methoden

### 2 types

- **Bestaande** bv  
Console.WriteLine(),  
Convert.ToInt32()
- **Zelfgemaakte**

Steeds herkenbaar aan ()

# Methoden : voor de luie, slimme programmeur



- Nut van methoden:
  1. Stukken code kunnen herbruikt worden, zonder dat ze hertypet moeten worden.
  2. Lange stukken code kunnen opgedeeld worden in kleinere, losse stukken (“verdeel en heers”-principe)
- Methoden lossen niets zelf op, maar maken het programmeren eenvoudiger
  - (minder typwerk, minder kans op fouten, etc).

# Methode syntax

- Aanmaken methode:

```
static returntype MethodeNaam(parameters)
{
    //code van methode
}
```

- Elders oproepen:

```
MethodeNaam();
```

# Eenvoudige methode

- Methodenaam: doit
- Parameters: geen
- Returntype: void
  - wil zeggen: deze methode geeft niets terug (zie later)

```
static void doit () {  
    Console.WriteLine ("Hello");  
}
```

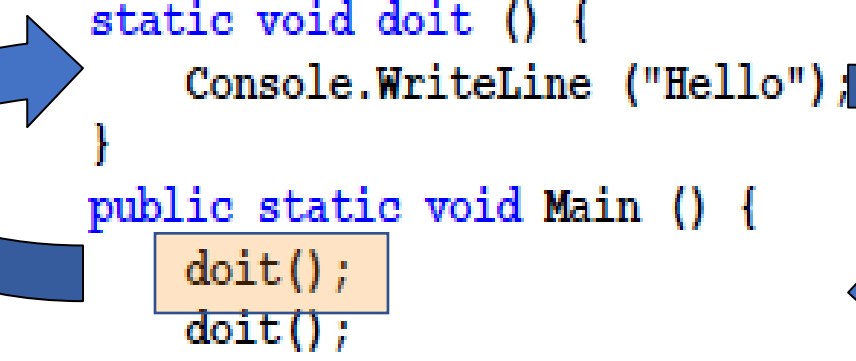
# Een methode maken

```
using System ;  
class MethodDemo {  
    static void doit () {  
        Console.WriteLine ("Hello");  
    }  
    public static void Main () {  
        doit();  
        doit();  
    }  
}
```

Zelf gemaakte methode, naam doit()

# Een methode maken

```
using System ;  
class MethodDemo {  
    static void doit () {  
        Console.WriteLine ("Hello") ;  
    }  
    public static void Main () {  
        doit();  
        doit();  
    }  
}
```



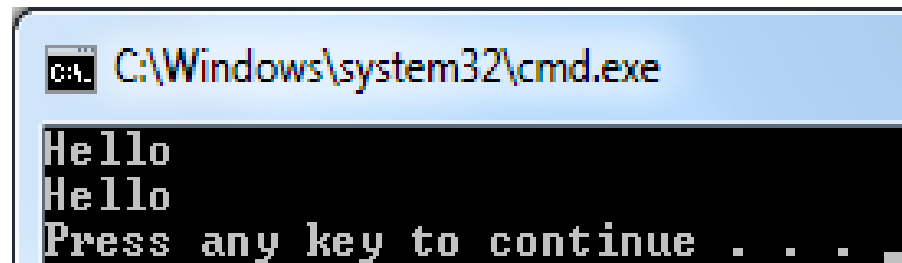
Aanroep van methode doit



# Een methode maken

```
using System ;  
class MethodDemo {  
    static void doit () {  
        Console.WriteLine ("Hello");  
    }  
    public static void Main () {  
        doit();  
        doit();  
    }  
}
```

Tweede aanroep van methode doit().



```
C:\Windows\system32\cmd.exe  
Hello  
Hello  
Press any key to continue . . . _
```

# Parameters

- Parameters: extra informatie die we meegeven wanneer we methode aanroepen

- Voorbeelden:

- `Console.WriteLine("Schijf deze zin");`
- `Math.Pow(4,3);`

Vervangen door lijst  
van type+naam,  
bv  
int getal, string naam

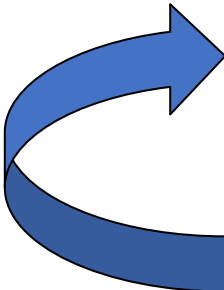
```
static returnType MethodeNaam(parameters)
{
    //code van methode
}
```

# Voorbeeld: methode met 1 parameter

- Methodenaam: silly
- Parameters:
  - 1° int i
- Returntype: void

```
static void silly (int i) {  
    Console.WriteLine ( "i is : " + i ) ;  
}
```

# Parameters



```
using System ;  
class MethodDemo {  
    static void silly ( int i ) {  
        Console.WriteLine ( "i is : " + i ) ;  
    }  
    public static void Main () {  
        silly ( 101 ) ;  
        silly ( 500 ) ;  
    }  
}
```

Aanroepen van methode silly().  
We geven als parameter de integer 101 mee.

# Parameters

- Parameters: extra informatie die we meegeven wanneer we methode

```
using System ;  
class MethodDemo {  
    static void silly ( int i) {  
        Console.WriteLine ( "i is : " + i ) ;  
    }  
    public static void Main () {  
        silly ( 101 ) ;  
        silly ( 500 ) ;  
    }  
}
```

Methode silly() verwacht één parameter van het type integer.

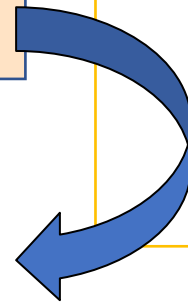
Binnen de methode zal deze parameter als i door het leven gaan

# Parameters

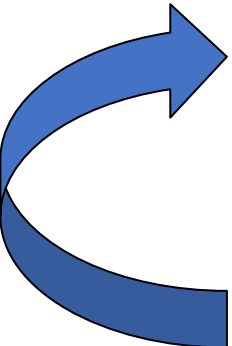
```
using System ;  
class MethodDemo {  
    static void silly ( int i) {  
        Console.WriteLine ( "i is : " + i ) ;  
    }  
    public static void Main () {  
        silly ( 101 ) ;  
        silly ( 500 ) ;  
    }  
}
```

Op scherm verschijnt

i is : 101



# Parameters



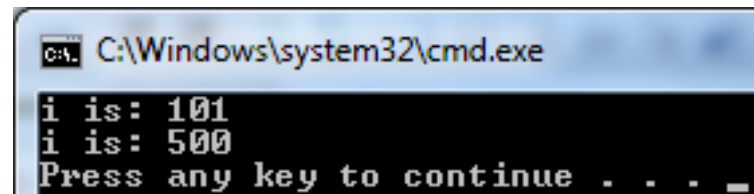
```
using System ;  
class MethodDemo {  
    static void silly ( int i ) {  
        Console.WriteLine ( "i is : " + i ) ;  
    }  
    public static void Main () {  
        silly ( 101 ) ;  
        silly ( 500 ) ;  
    }  
}
```

Tweede aanroep van methode silly(). We geven nu een parameter met waarde 500 mee.

# Parameters

```
using System ;  
class MethodDemo {  
    static void silly ( int i) {  
        Console.WriteLine ( "i is : " + i ) ;  
    }  
    public static void Main () {  
        silly ( 101 ) ;  
        silly ( 500 ) ;  
    }  
}
```

Op scherm verschijnt  
i is : 500



```
C:\Windows\system32\cmd.exe  
i is: 101  
i is: 500  
Press any key to continue . . . _
```



# Returntypes van methoden

# Returntype

- Aangeven wat voor type data uit de methode komt, mogelijkheden:
  - **Void**: niets komt terug uit de methode
  - Eender welk type (int, string, double)
    - MOET eindigen met **return** in methode-body

```
static returntype MethodeNaam(parameters)
{
    //code van methode
}
```

# Methode met return

- Naam: sillyReturnPlus
- Parameters:
  - 1° int i
- Returntype: int

```
static int sillyReturnPlus ( int i) {  
    i = i + 1;  
    Console.WriteLine ( "i is : " + i ) ;  
    return i;  
}
```

# Methode met return

- Naam: sillyReturnPlus
- Parameters:
  - 1° int i
- Returntype: int

Type na return moet overeenkomen met return-type in header methode.


Enkele voorbeelden in deze methode die geldig zijn:

```
return i;  
return 5;  
return i*3;
```

```
static int sillyReturnPlus ( int i) {  
    i = i + 1;  
    Console.WriteLine ( "i is : " + i ) ;  
    return i;  
}
```

# Waarden terugkrijgen van methodes

**Return value:** waarde teruggeven als resultaat van een methode.



```
using System ;
class ReturnDemo {
    static int sillyReturnPlus ( int i) {
        i = i + 1;
        Console.WriteLine ( "i is : " + i ) ;
        return i;
    }

    public static void Main () {
        int res;
        res = sillyReturnPlus (5);
        Console.WriteLine ( "res is : " + res ) ;
    }
}
```

Aanroep van de methode `sillyReturnPlus()`, met extra parameter de integer 5.

Het resultaat van de methode wordt in `res` bewaard.

# Waarden terugkrijgen van methodes

```
using System ;  
class ReturnDemo {  
  
    static int sillyReturnPlus ( int i) {  
        i = i + 1;  
        Console.WriteLine ( "i is : " + i ) ;  
        return i;  
    }  
  
    public static void Main () {  
        int res;  
        res = sillyReturnPlus (5);  
        Console.WriteLine ( "res is : " + res ) ;  
    }  
}
```

Type van de return waarde die deze methode heeft.

# Waarden terugkrijgen van methodes

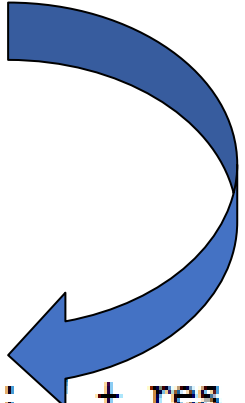
```
using System ;  
class ReturnDemo {  
  
    static int sillyReturnPlus ( int i) {  
        i = i + 1;  
        Console.WriteLine ( "i is : " + i ) ;  
        return i;  
    }  
  
    public static void Main () {  
        int res;  
        res = sillyReturnPlus (5);  
        Console.WriteLine ( "res is : " + res ) ;  
    }  
}
```

i wordt met één verhoogd en op het scherm getoond.



# Waarden terugkrijgen van methodes

```
using System ;  
class ReturnDemo {  
  
    static int sillyReturnPlus ( int i) {  
        i = i + 1;  
        Console.WriteLine ( "i is : " + i ) ;  
        return i;  
    }  
  
    public static void Main () {  
        int res;  
        res = sillyReturnPlus (5);  
        Console.WriteLine ( "res is : " + res ) ;  
    }  
}
```



i wordt ge'return'd als resultaat van deze methode



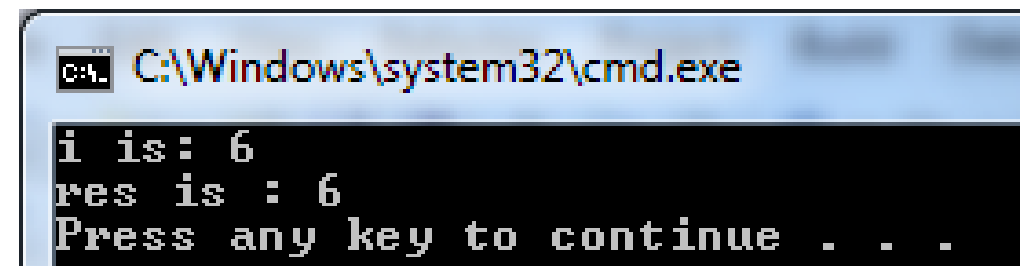
# Waarden terugkrijgen van methodes

```
using System ;  
class ReturnDemo {  
  
    static int sillyReturnPlus ( int i) {  
        i = i + 1;  
        Console.WriteLine ( "i is : " + i ) ;  
        return i;  
    }  
  
    public static void Main () {  
        int res;  
        res = sillyReturnPlus (5);  
        Console.WriteLine ( "res is : " + res ) ;  
    }  
}
```

Returnwaarde van aangeroepen waarde wordt in res geplaatst.

# Waarden terugkrijgen van methodes

```
using System ;  
class ReturnDemo {  
  
    static int sillyReturnPlus ( int i) {  
        i = i + 1;  
        Console.WriteLine ( "i is : " + i ) ;  
        return i;  
    }  
  
    public static void Main () {  
        int res;  
        res = sillyReturnPlus (5);  
        Console.WriteLine ( "res is : " + res ) ;  
    }  
}
```



C:\Windows\system32\cmd.exe

```
i is: 6  
res is : 6  
Press any key to continue . . .
```

# Voorbeeldmethode

- Faculteitsmethode:

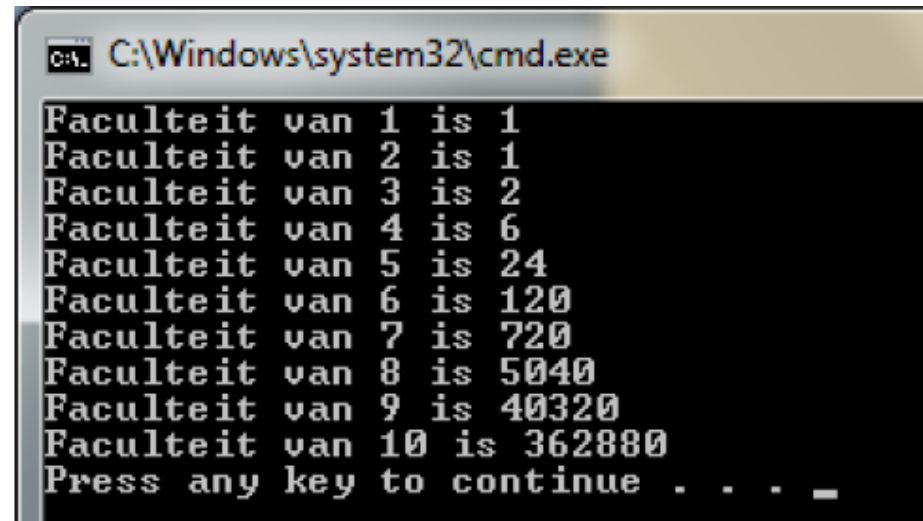
```
static int BerekenFaculteit(int grens)
{
    int resultaat = 1;
    for (int i = 1; i <= grens; i++)
    {
        resultaat *= i;
    }
    return resultaat;
}
```
- Gebruik:

```
static void Main(string[] args)
{
    int getal = 6;
    Console.WriteLine("Faculteit van {0} is {1}", getal, BerekenFaculteit(getal))
}
```

  - Op scherm verschijnt:
    - “Faculteit van 6 is 720”

# Methode hergebruiken

```
for (int i = 1; i < 11; i++)  
{  
    Console.WriteLine("Faculteit van {0} is {1}", i, BerekenFaculteit(i));  
}
```



```
C:\Windows\system32\cmd.exe  
Faculteit van 1 is 1  
Faculteit van 2 is 2  
Faculteit van 3 is 6  
Faculteit van 4 is 24  
Faculteit van 5 is 120  
Faculteit van 6 is 720  
Faculteit van 7 is 5040  
Faculteit van 8 is 40320  
Faculteit van 9 is 362880  
Press any key to continue . . . _
```

# Programmer's Point

## “Design with methods”

1° Je moet minder vaak dezelfde code opnieuw schrijven

2° Indien er een fout in je code is, moet je die maar op 1 plek oplossen.

# Parameters doorgeven

A SOCIETY GROWS GREAT  
WHEN OLD MEN PLANT TREES  
WHOSE SHADE THEY KNOW THEY  
SHALL NEVER SIT IN.



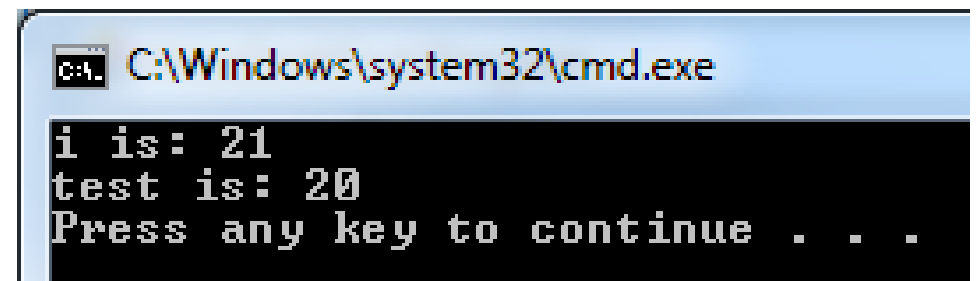
# Parameters 'by value' doorgeven

- Standaard wordt enkel een **kopie van de waarde** van een variabele meegegeven aan een methode en NIET de variabele in z'n geheel zelf. (=pass by value)

- Voor 

```
static void addOneToParam(int i)
{
    i = i + 1;
    Console.WriteLine("i is: " + i);
}

static void Main(string[] args)
{
    int test = 20;
    addOneToParam(test);
    Console.WriteLine("test is: " + test);
}
```



```
C:\Windows\system32\cmd.exe
i is: 21
test is: 20
Press any key to continue . . .
```



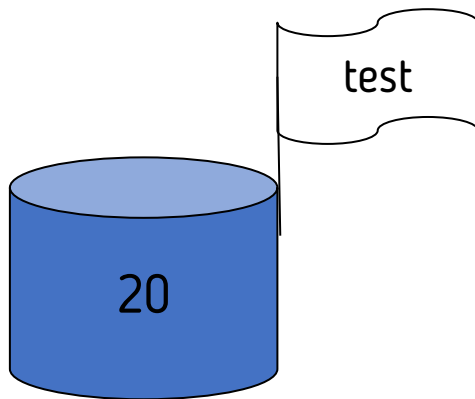
# Voorbeeld 'by value passing'

```
static void addOneToParam(int i)
{
    i = i + 1;
    Console.WriteLine(("i is: " + i));
}

static void Main(string[] args)
{
    int test = 20;
    addOneToParam(test);
    Console.WriteLine("test is: "+test);
}
```

Startpunt!

We maken variabele test aan met waarde 20.

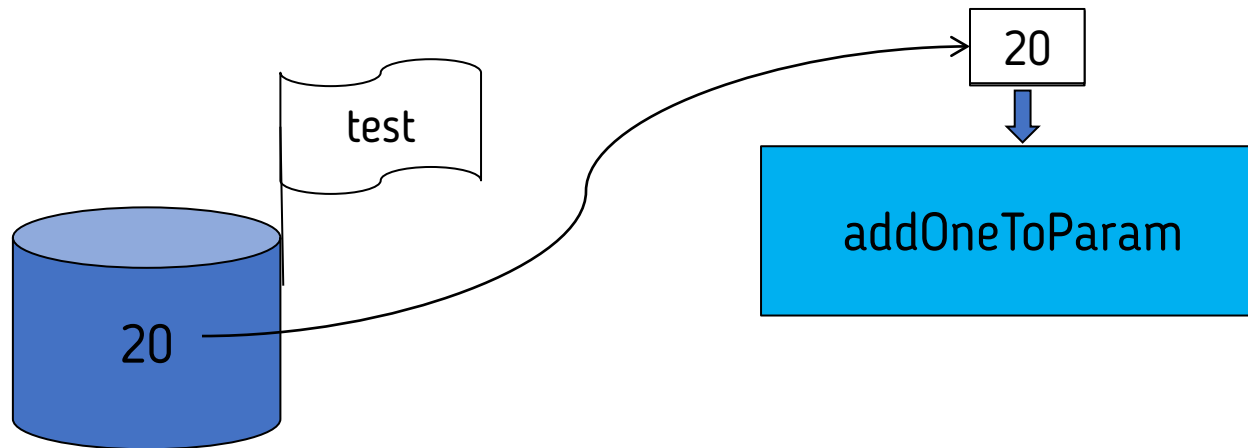


# Voorbeeld 'by value passing'

```
static void addOneToParam(int i)
{
    i = i + 1;
    Console.WriteLine(("i is: " + i));
}

static void Main(string[] args)
{
    int test = 20;
    addOneToParam(test);
    Console.WriteLine("test is: "+test);
}
```

Inhoud van test wordt meegegeven aan addOneToParam



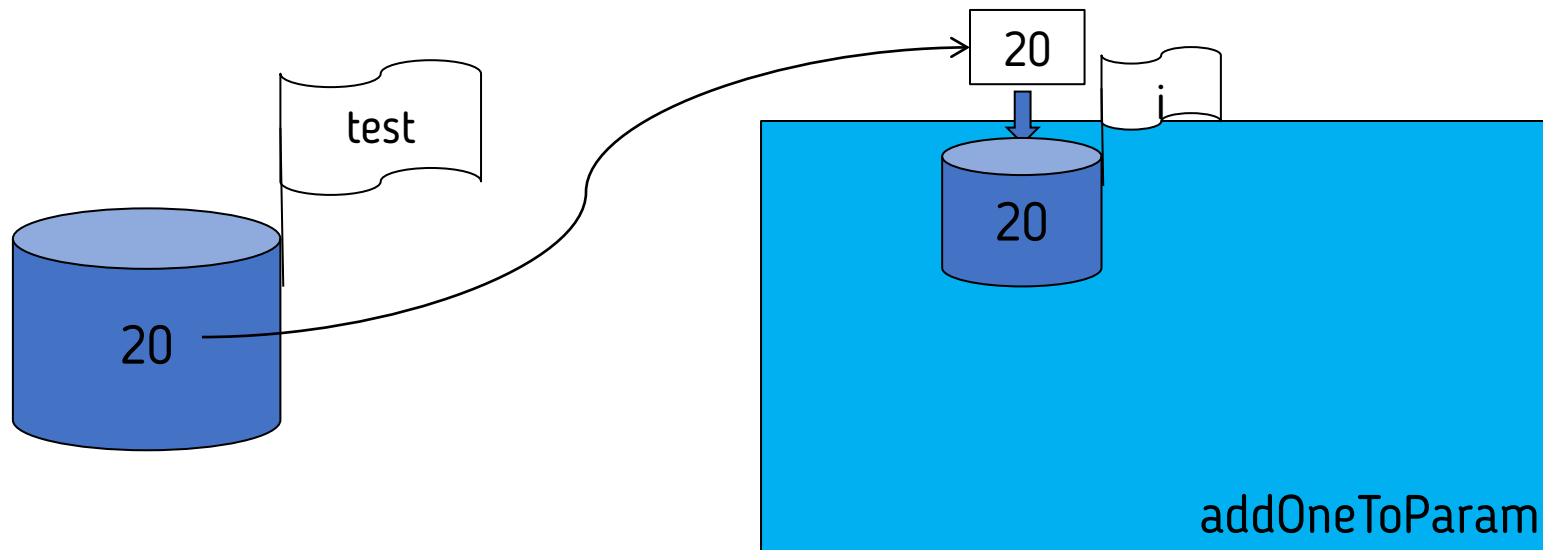
# Voorbeeld 'by value passing'

```
static void addOneToParam(int i)
{
    i = i + 1;
    Console.WriteLine("i is: " + i);
}

static void Main(string[] args)
{
    int test = 20;
    addOneToParam(test);
    Console.WriteLine("test is: " + test);
}
```

We komen in addOneToParam:

Variabele int i wordt aangemaakt

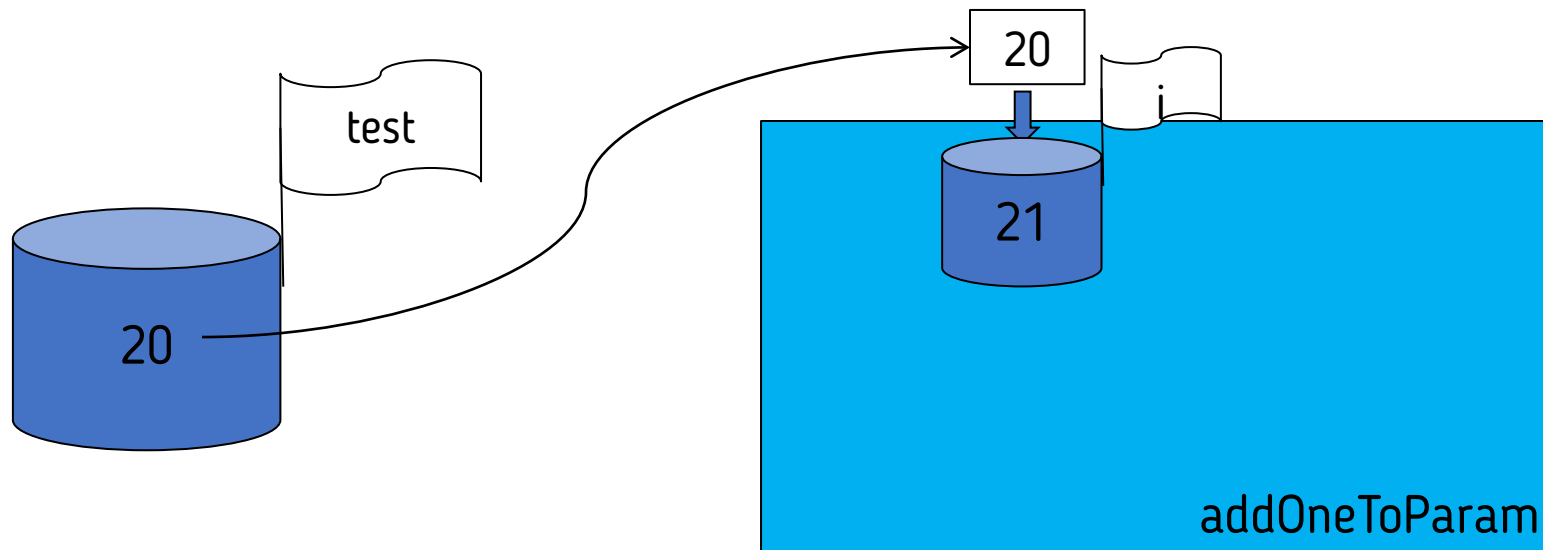


# Voorbeeld 'by value passing'

```
static void addOneToParam(int i)
{
    i = i + 1;
    Console.WriteLine("i is: " + i);
}

static void Main(string[] args)
{
    int test = 20;
    addOneToParam(test);
    Console.WriteLine("test is: " + test);
}
```

i wordt met eentje verhoogt

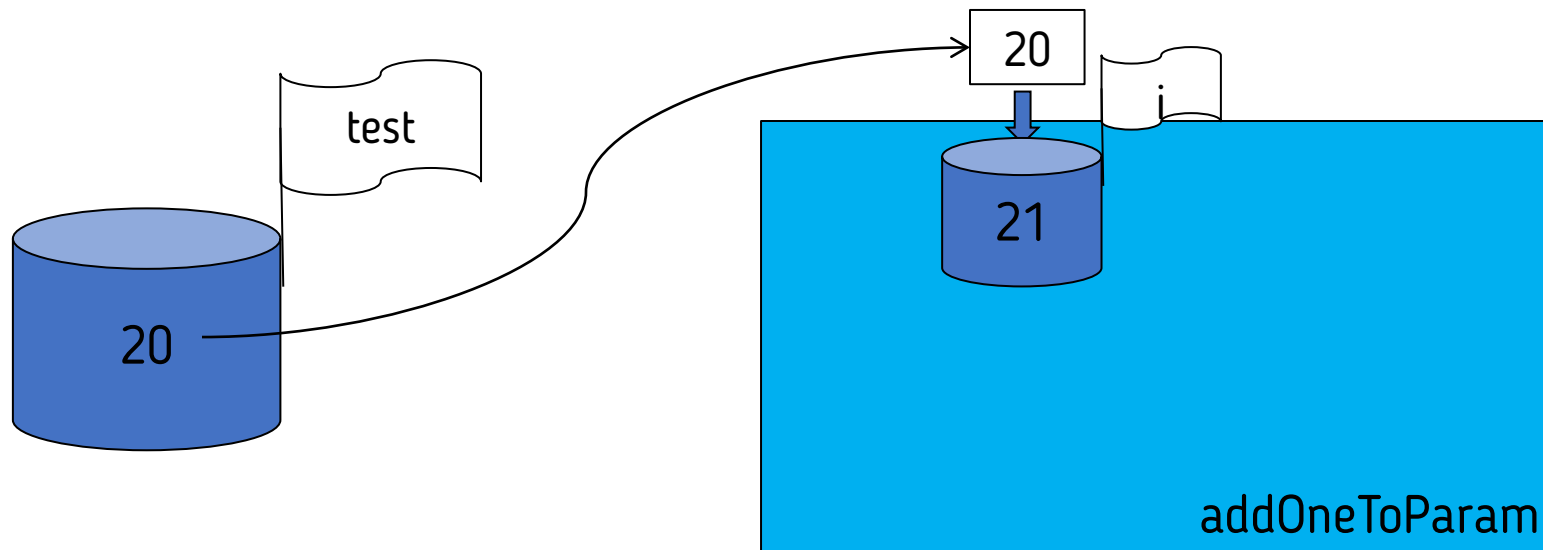


# Voorbeeld 'by value passing'

```
static void addOneToParam(int i)
{
    i = i + 1;
    Console.WriteLine(("i is: " + i));
}

static void Main(string[] args)
{
    int test = 20;
    addOneToParam(test);
    Console.WriteLine("test is: " + test);
}
```

Op scherm verschijnt:  
i is 21

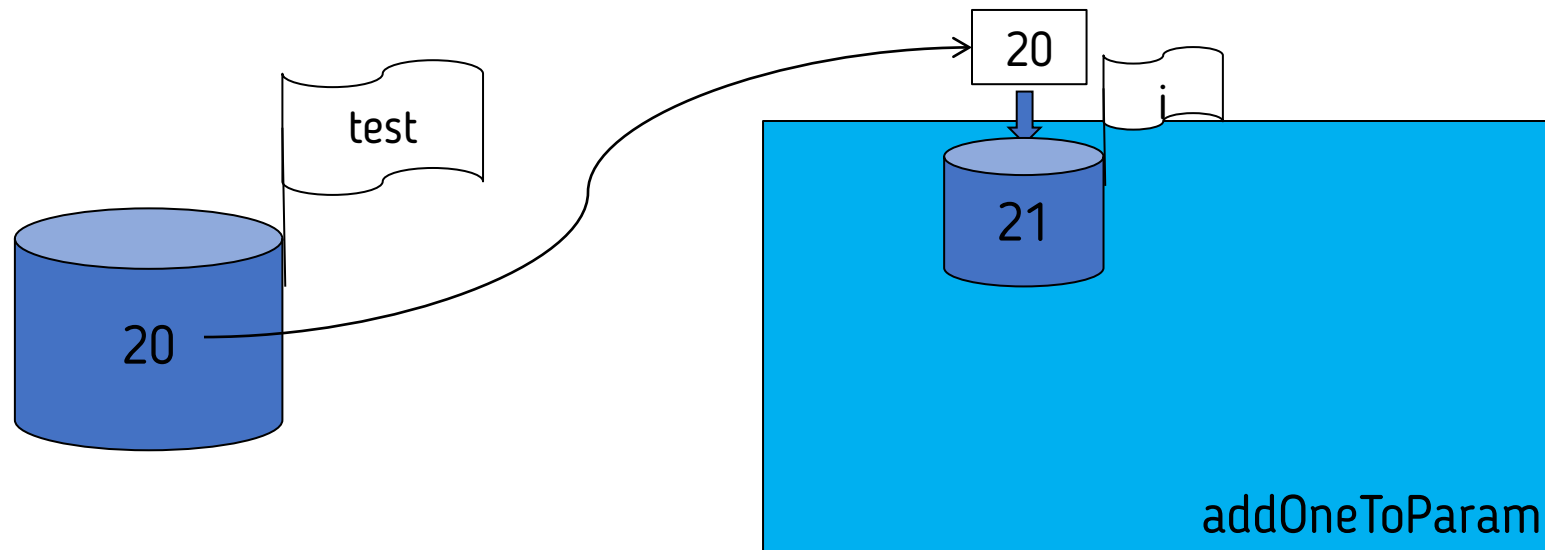


# Voorbeeld 'by value passing'

```
static void addOneToParam(int i)
{
    i = i + 1;
    Console.WriteLine(("i is: " + i));
}

static void Main(string[] args)
{
    int test = 20;
    addOneToParam(test);
    Console.WriteLine("test is: " + test);
}
```

Alles van addOneToParam verdwijnt

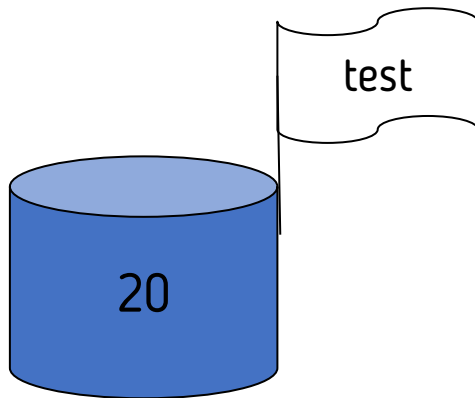


# Voorbeeld 'by value passing'

```
static void addOneToParam(int i)
{
    i = i + 1;
    Console.WriteLine(("i is: " + i));
}

static void Main(string[] args)
{
    int test = 20;
    addOneToParam(test);
    Console.WriteLine("test is: "+test);
}
```

Op scherm verschijnt:  
test is 20

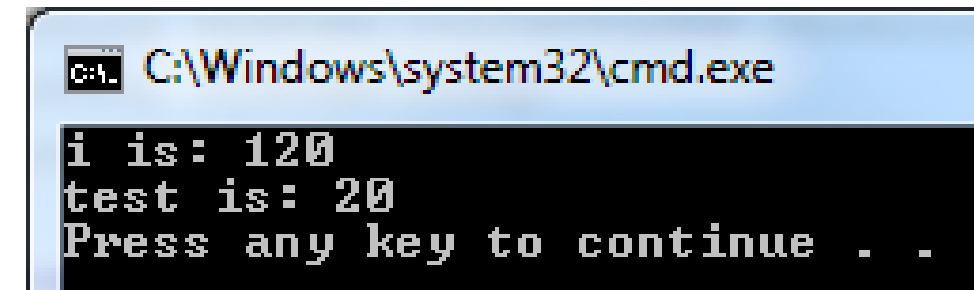


# Parameters 'by value' doorgeven

- Enkel waarde van variabele wordt meegegeven, niet de variabele zelf!!!!
- We kunnen dus ook doen:

```
static void addOneToParam(int i)
{
    i = i + 1;
    Console.WriteLine(("i is: " + i));
}

static void Main(string[] args)
{
    int test = 20;
    addOneToParam(test+99);
    Console.WriteLine("test is: "+test);
}
```



```
C:\Windows\system32\cmd.exe
i is: 120
test is: 20
Press any key to continue . .
```