# PB notes 02 theory

## 2 STRINGS

### 2.1 Speciale symbolen in een string

`"<string>"` -> string af bakenen `string s = "Programmeren";`

***escape sequences*** beginnen allemaal met `\`

| escape sequence | result in string literal | clarification |
|---|---|---|
| `\"` | `"` | aanhalingsteken |
| `\n` | `newline` | invisible symbol that starts a new line |
| `\\` | `\` | backslash |

### 2.2 Strings bouwen met `+`

***string concatenation***

```
`string s = "<string1>" + "<string2>""
Console.WriteLine(s);

=> <string1><string2>

of

int aantal = 3;
Console.WriteLine("ik kocht " + aantal + " broden");

=> ik kocht 3 broden
```

hier is `+` geen optelling! het plakt strings aan elkaar

## 3 Console invoer en uitvoer

### 3.1 Write en Writeline

Console window keeps track of hidden **cursorpostition** => place where characters will appear next `Console.WriteLine(<var>)` => print `<var>` to console and places a hidden `newline` symbol after `Console.Write(<var>)` => just prints `<var>` without a new line

official documentation:

- https://docs.microsoft.com/en-us/dotnet/api/system.console.write
- https://docs.microsoft.com/en-us/dotnet/api/system.console.writeline

### 3.2 ReadLine

`Console.Readline(<var>)` => used to read input from the console end input with `[ENTER]`
leest de input altijd als **string**!

official documentation:

- https://docs.microsoft.com/en-us/dotnet/api/system.console.readline

## 4 Conversie tussen de datatypes

**Type conversion** = conversion of `<var>` from one type to another implies that the data of `<var>` undergoes a change BUT it actually makes a new `<var>` of the desired type **2 PROBLEMS:**

- no new type can be derived conversion fails bv.: converting the `string` "Hallo" to an `int`
- loss of information = **narrowing conversion** bv.: converting the `double` 8.4 to `int` 8 <=> **widening conversion** bv. `int` to `double`

**IMPLICIT CONVERSION/TYPECASTING** widening conversion bv,:

```
1. int val1 = 10;
2. double val2 =val1;
```

- no data loss
- 2 compatible datatypes

| Comvert from | Convert to |
|---|---|
| byte | short, int, long, double, float |
| short | int, long, float, double |
| int | long, float, double |
| long | float, double |
| float | double |

**EXPLICIT CONVERSIONS/TYPECASTING** you have to typecast for the compiler to allow it = **explicit typecasting**

- incompatible datatypes where automatic conversion can't be done

- target-type specifies desired type to convert to
- sometimes results in **lossy conversion**

`<var>.ToString()` = convert `<var>` to `string` type

`<vartype>.Parse(<string>)` = convert `<string>` to `<varType>` ; bv.:

- `int.Parse(<var>)`
- `double.Parse(<var>)`
- `float.Parse(<var>)`

`Convert.To<varType>(<var>)` = convert any datatype of `<var>` to the datatype `<varType>` bv.:

- `Convert.ToInt32(<var>)`
- `Convert.ToDouble(<var>)`
- `Convert.ToFLoat(<var>)`
- `Convert.ToString(<var>)`

# 6 Controlestructuren

- **sequentie**: opdrachten uitvoeren in de volgorde waarin ze in de broncode staan
- **selectie** : stukken code wel of niet uitvoeren (op basis van één of andere voorwaarde)
- **iteratie** : stukken herhalen (wederom afhankelijk van een voorwaarde)

**controlestructuren** bepalen de volgorde waarin opdrachten in het programma worden uitgevoerd bv.: if/else => werken vaak op meerdere regels code => **codeblock** `{...}` = code afbakenen bv.:

```
static void Main() {
Console.Write("Geef uw leeftijd : ");
string leeftijdAlsTekst = Console.ReadLine();
int leeftijd = int.Parse(leeftijdAlsTekst);
}
```

hier is er een codeblock na `Main()`

# 7 Conrolestructuur : if

=> selectiestructuur kan codeblock wel of niet laten uitvoeren

```
if (<condition>) {
        ...
}

```

```
of

if (<condition>)
{
        ...
}
```

condition always results on a **boolean** TRUE or FALSE

## 8 Waarden vergelijken

| vergelijking | betekenis |
|---|---|
| `x == y` | x gelijk aan y |
| `x != y` | x verschillend van y |
| `x < y` | x kleiner dan y |
| `x <= y` | x kleiner dan of gelijk aan y |
| `x > y` | x groter dan y |
| `x >= y` | x groter dan of gelijk aan y |

strings:

| vergelijking | betekenis |
|---|---|
| `<string1> == <string2>` | is `<string1>` dezelfde tekst als `<string2>` |
| `<string1> != <string2>` | is `<string1>` niet dezelfde tekst als `<string2>` |

## 9 controlestructuur : if/else

```
if (<condition>)
{
         code block 1
}
else
{
        code block 2
}
```

2 code blocks 2 mogelijkheden:

- `<condition>` is true => code block 1 will execute
- `<condition>` is false => code block 2 will execute

# 10 het boolean datatype

boolean datatype = `bool` can only contain:

- true
- false

```
bool trueOrFalse = false;

if(trueOrFalse == true)
{
        Console.Write("True.");
}
else
{
        Console.Write("False.");
}
```