PB1 notes 01 theory

3 ENKELE TAALELEMENTEN

3.1 Statements and terminators:

- instructies = statements
- wordt afgesloten met een `;`
- -> statement terminator

3.2 Literals

- wordt letterlijk gelezen door compiler
- onveranderd
- `"Hello world!"` -> string literal

3.3 Commentaar

- **verduidelijking** van code voor andere mensen of jezelf
- wordt niet gecompiled
- over meerdere regels -> start met `/*` en eindigt met `*/`
- over 1 regel -> `//` voor de regel

3.4 Variabelen:

tijdens uitvoer van program **informatie onthouden** bij aanmaak **reserveert** dit **geheugen** vb.: - letterlijke waarden die we in ons algoritme nodig hebben, dit noemen we literal values - resultaten van berekeningen - input van de gebruiker - waardes ingelezen uit een bestand - informatie die binnenkomt over een netwerkverbinding - … variabele kan nooit echt leeg zijn **declaratie** - naam van info - soort info dat het wenst te houden **analogie, geheugen en datatype** string, float, int, … = datatypes vb.: `string s = "hello";` alles tussen `"..."` = string/tekst `s` stores "hello" als een string `Console.WriteLine("12")` -> print 12 als tekst `Console.WriteLine(12)` -> print 12 als getal

3.5 String, int en double waardes:

string = text **int** = gehele getallen/ integers **float**/**double** = kommagetallen verschil is aantal getallen na komma dat opgeslagen worden -> toont welke **soort** info de var. kan opslaan ->

welke **acties** we met/op de var. kan toepassen

3.6 Static type checking

compilefouten -> krijgen we als we handeling proberen die niet kunnen/ondersteund zijn. vb.: string *int: `12* "tekst" `**expressie** = stuk code dat de waarde uitdrukt vb.: - rechts van = (toekenningsoperator) ->` x = `- tussen haakjes bij` WriteLine `->` WriteLine() `- links en rechts van (vermenigvuldiginsoperator) ->` *x =* `-> kan voorkomen in: - literal vorm (vb.:` "hello" `) - uitlezen van een variable (vb.:` x `) - complexere vormen (vb.:` x * 2`) dat subexpressies combineert met operatoren

3.7 Berekeningen maken

- `+` = som
- `-` = verschil
- `*` = vermenigvuldigen
- `/` = delen
- `%` = modulo (rest na deling) alles tussen haakjes `()` wordt eerst gedaan voor andere berekeningen + leesbaarheid **MATH RULES!!**

FUNCTIONS

- `Convert.ToDouble(<var>)` -> converts var. to double
- `Console.WriteLine(<var>)` -> writes var. to the console + new line
- `console.Write(<var>)` -> writes var to the console / no new line
- `Console.ReadLine(<var>)` -> reads input from the console
- `<var>.ToUpper()` -> converts characters of a string to uppercase
- `int.Parse(<var>)` / `double.Parse(<var>)` -> converts string to double/int/... (similar to convert)

EXTRA

om in een string een `"` te tonen -> `\` voor `"` plaatsen = `\"` => meestal `\` voor speciale tekens die stuff doen in strings om ze toch te tonen in de string vb.: `\"` , `\$` , `\\`

om meerdere strings aan elkaar te hangen: + gebruiken vb.:

```
int leeftijd = 22;
Console.WriteLine("Ik ben " + leeftijd + "jaar oud.");
```

VERDIEPING

I Dot notatie, using en namespaces

dot notatie

`System.Console.WriteLine("Hello");` `Writeline` = **method** -> gedefinieerd in `System.Console` of gewoon `Console` = **klasse** -> gedefinieerd in `System` = **namespace**

namespaces en using

```
using System;

namespace HelloWorldApp {
        class Program {
                static void Main(string[] args)
                {
                Console.WriteLine("Hello World!");
                }
        }
}
```

Namespace = rubriek/map waar klassen worden uitgeschreven hier `HelloWorldApp`

- verschillende classes met dezelfde naam maken (in verschillende namespaces) -> naamconflicten vermijden
- namespaces + subnamespaces => hiërarchische structuur -> bv voor classes te organiseren die samen horen of zoiets

`System.IO` = alles over lezen uit/schrijven naar bestanden `System.Drawing` = grafische vormen tekenen

-> definieer je adhv namespace **statementblok** sleutelwoord namespace + naam van rubriek + {...}

```
namespace MijnRubriek {
        namespace SubRubriek {
                ...
        }
}
```

of

```
namespace MijnRubriek.SubRubriek {
        ...
}
```