**H3: STRINGS & CHARS**

ASCII

* Eerste standaardisering van tekens die gebruikt worden op computers
* Origineel opgemaakt voor gebruik van telegrafen, waardoor een paar van de onzichtbare tekens niet meer nuttig zijn
* Gebruikt 7 bits, en dus 2^7 = 128 tekens mogelijk (8ste bit wordt gebruikt voor pariteit)
* Werd opgevolgd door Unicode
* Eerste 32 tekens zijn “onzichtbaar”, vb: backspace, enter, del, return, spatie, escape, etc…

Unicode

* Standaard om tekens **binair** voor te stellen met behulp van een **16-bit getal**.
* Superset van ASCII
* Dus: eerste 128 tekens zijn dezelfde als ASCII om comptabiliteit te garanderen
* 137,929/1,111,998 tekens gedefinieerd (15/10/2020)
* https://unicodetable.archive.thomasorlita.com/

***Char***

* Ieder teken dat in **Unicode** staat is een char

Een enkel karakter opslaan als TEKST doe je door het *char* datatype te gebruiken. Vb:

*char eenLetter= ‘X’;*

* Gebruik enkele apostrof ( ‘ )
* Je kan EENDER WELK Unicode-teken in een *char* datatype bewaren
* **Intern wordt de Unicode bewaard in de variabele, namelijk een 16 bit geheel getal**
* Vb: *char eenGetal= ‘7’;* Wij zien dat het cijfer 7 is opgeslagen, maar de compiler zal die ‘7’ als een karakter gebruiken, en is opgeslagen als de overeenkomende Unicode getal. Als je dus werkelijk het cijfer 7 wilt gebruiken, zal je die moeten converteren naar het relevante datatype. (in dit geval *int*)

***String***

* Een reeks van nul, een of meerdere *char*-elementen die je achter elkaar kunt zetten.
* Dit datatype wordt uiteraard gebruikt om tekst voor te stellen.
* Één van de belangrijkste datatypes in eender welke taal, welke software zal niet minstens een paar lijntjes tekst moeten printen. Hiermee communiceer je met de eindgebruiker!

*string eenString= “Dit is een string.”;*

* Gebruikt dubbele apostrof ( “ )

**H2: ESCAPE CHARACTERS**

* Sommige karakters hebben hun eigen functie binnen C#, dus om die alsnog te kunnen printen en geen verwarring te veroorzaken in de compiler, kun je speciale *escape chars* gebruiken om deze karakters aan te duiden zodat ze geprint worden.
* Escape chars worden met **backslash** ( \ ) aangeduid, gevolgd door het gewenste karakter.

Vb:

FOUT: *char karakter= ‘ ‘ ‘;*

JUIST: *char karakter= ‘ \’’;*

Meer voorbeelden:

(volledige lijst: <https://docs.microsoft.com/en-us/archive/blogs/csharpfaq/what-character-escape-sequences-are-available>)

* \’ => enkele apostrof
* \” => dubbele apostrof
* \\ => backslash tonen
* \\\\ => dubbele backslash tonen (dus 4 backslashes zetten, wow!)
* \n => new line (zoals Enter drukken)
* \t => horizontale Tab drukken
* \a => laat je computer een biep-geluid maken
* \uxxxx => toon een specifieke Unicode-karakter

ESCAPE CHARS IN **STRINGS**

* Binnenin een string kun je ook met escape chars werken

Vb:

*string woord= “\’s avonds”;*

*string eenZin= “Hij zei \”Hello World!\””;*

* Dubbele aanhalingstekens in de string zelf

WITREGELS EN TABS

* Escape chars worden vooral gebruikt om in strings witregels (\n) en tabs (\t) te zetten.
* **Tabstop**: tabulatie zal ALTIJD op dezelfde plek stoppen (10 karakters) ongeacht van hoeveel karakters er al staan. Handig voor kolommen te maken.

APENSTAARTJE

* Verbatim/letterlijk karakter
* Plaats een @ voor de string literal **om escape chars te negeren**.
* Dus alle tekens die normaal als escape chars zouden gezien worden zullen gwn deel zijn van de string.
* Voordeel: bvb *string myAdress= @”C:\Temp\Myfile.txt”;*
* Aanhalingstekens moeten nog steeds ge-escaped worden though.
* ASCII art!!!!

OPTELLEN VAN CHAR VARIABELEN

*char letter1= ‘A’;*

*char letter2= ‘B’;*

*Console.WriteLine(letter1 + letter2);*

* Console: 131 (niet “AB” zoals verwacht)

Waarom?

* Variabelen van het *char* datatype worden als **gehele getallen** bewaard! (Unicode-waarde)
* De compiler ziet bij *letter1 + letter2* twee gehele getallen staan en zal die optellen en REMEMBER het resultaat van de variabelen bepalen het datatype DUS 2 gehele getallen (int) opgeteld zullen een int als resultaat geven!
* A = 65

B = 66

65 + 66 = 131

**Concatenatie**

* Het samenvoegen van twee strings (met het + teken)

**H3: STRINGS SAMENVOEGEN**

Er zijn 3 methoden:

* +-operator (hebben we tot nu toe gebruikt)
* $ string interpolation (deze wordt de nieuwe standaard)
* String.Format() (Een oude methode die we nog steeds zullen terugvinden in code)

*int age= 13;  
string name= “Finkelstein”*

* We willen deze variabelen samenvoegen in een *string result* met als resultaat:

*Ik ben Finkelstein en ik ben 13 jaar oud.*

We passen de 3 methoden toe:

METHODE 1: +-OPERATOR

*string result= “Ik ben “+ name + “ en ik ben “+ age + “ jaar oud.”;*

* Langdradige, lastige manier die leidt tot lange, onleesbare strings, bah.
* LET OP! **Concatenatie** volgt de volgorde van bewerkingen! Alles wordt samengevoegd van LINKS NAAR RECHTS.

METHODE2: STRING INTERPOLATION MET $

*string result= $”Ik ben {name} en ik ben {age} jaar oud.”;*

* We gebruiken het $ teken **VOOR** de string om aan te geven dat specifieke delen van de string (die aangegeven zijn met **accolades**) geïnterpoleerd moeten worden
* Interpolatie = invoeging

Berekeningen maken met string interpolatie:

* Je mag EENDER welke expressie zetten tussen de accolades!
* Vb: *string result= $”Ik ben {name} en ik ben {age+5} jaar oud.”;*
* Vb: *Console.WriteLine($3 maal 9 is {3\*9});*

Mooier formatteren:

* Je kan bepalen hoe variabelen en expressies juist weergegeven moeten worden!
* Aantal cijfers na de komma bepalen:

*double number = 12.345;*

*Console.WriteLine($”{number:F2});*

*string result= $“{4.36\*7.6589:F3}”;*

* Er wordt NIET afgerond, cijfers worden simpelweg afgekapt.

(volledige lijst van formats: <https://docs.microsoft.com/en-us/dotnet/standard/base-types/standard-numeric-format-strings>)

* D5: toont een geheel getal als een 5 cijfer getal
* F2: cijfers na de komma tonen
* E2: wetenschappelijke notatie met 2 cijfers precisie (1.20E+007 bvb)
* C: geldbedrag
* N2: getal tonen met . scheidingen

METHODE 3: String.Format()

*string result= String.Format(“Ik ben {0} en ik ben {1} jaar oud.”, name, age);*

* De getallen binnen de accolades geeft aan de hoeveelste parameter na de string hier geplaatst moet worden.
* DUS: 0 = name, 1 = age

*string result= String.Format(“Ik ben {1} en ik ben {1} jaar oud.”, name, age);*

* In dit geval zou de string worden: “Ik ben 13 en ik ben 13 jaar oud.”

Je kan deze methode ook binnen in Console.WriteLine toepassen zonder String.Format:

*Console.WriteLine(“Gratis formateren. {0} maal hoera voor .NET!”, 3);*

**H3: UNICODE TONEN**

* Unicode kan gebruikt worden om kleur en smaak aan je console-applicatie te geven
* Bvb: Dwarf Fortress!

UNICODE KARAKTERS TONEN:

* Je zoekt de tekens die je nodig hebt in een Unicode-tabel (hier kun je er veel van vinden online, of ga naar <https://unicode-table.com/en/>)

Plaats het volgende bovenaan in je *Main*:

*Console.OutputEncoding = System.Text.Encoding.UTF8;*

2 methodes om Unicode karakter te tonen:

METHODE 1: COPY/PASTE

* Simpelweg copypasten van het symbool dat je wilt van de Unicode-tabel naar je VS.
* Let op dat veel lettertypes niet alle Unicode karakters gaan kunnen tonen.

METHODE 2: HEXADECIMALE CODE

* Hexadecimale code casten naar char

*char blokje = (char)0.02e7;*  
*Console.WriteLine(blokje);*

* Dit is de notatie om een hexadecimale code te casten naar het *char* datatype

Je kunt ook hexadecimale getallen rechtstreeks in een string plaatsten dmv een **escape character**:

*Console.WriteLine(“\u02e7”);*

* \uxxxx = is de notatie

UNICODE-ART TONEN

* Gebruik een website om ASCII-art te maken (zoals <http://asciiflow.com/>)

*String myname= @” [plaats hier je Unicode-art]”;*

* REMEMBER: ‘@’ zorgt er voor dat escape chars worden genegeerd.

UNICODE-ART KLEUREN

* Good fucking luck! (veel te veel werk)

CURSOR VERZETTEN

* Met een commando kun je bepalen waar de cursor terechtkomt

*Console.SetCursorPosition(horizontale coörd, verticale coörd);*

**H3: ENVIRONMENT BIBLIOTHEEK**

***Console.****WriteLine();*

* De “Console” stelt een bibliotheek voor die C# aanhaalt om programma’s te gebruiken.

Een andere nuttige bibliotheek is ***Environment.***

* Deze geeft uw applicatie allerlei informatie over de computer waarop het programma op dat moment op draait.
* Bvb:
* Werkgeheugen
* Gebruikersnaam
* Aantal processen
* Hardware
* Etc.
* LET OP: de informatie die gegeven zal worden hangt af OP WELKE COMPUTER HET PROGRAMMA DRAAIT!

Voorbeelden die de *Environment.*-bibliotheek gebruiken:

*bool is64bit= Environment.Is64BitOperatingSystem;*

*string pcname= Environment.MachineName;*

*int proccount= Environment.ProcessorCount;*

*string username= Environment.UserName;*

*long memory= Environment.WorkingSet;* (Zal ongeveer 10 Mb teruggeven)

* *Environment.WorkingSet*: toont hoeveel geheugen het programma toegewezen krijgt van Windows. De hoeveelheid hangt af van de lengte van je code.

Je kunt ook de *Environment.*-bibliotheek rechtstreeks aanhalen in een variabele:

*Console.WriteLine($”Je username is {Environment.UserName}”);*

* Hier gebruiken we dus interpolatie!

<https://docs.microsoft.com/en-us/dotnet/api/system.environment?view=netcore-3.1>

* Hier vindt je meer info over wat je kunt doen met de *Environment.*-bibliotheek.

PROGRAMMA AFSLUITEN

*Environment.Exit(0);*

* Het getal tussen de haakjes bepaal je zelf en staat voor de zogenaamde foutcode die je wilt meegeven bij het afsluiten van het programma (moest dat nodig zijn)
* Bij complexere programma’s zullen we dit commando meer gebruiken
* Nu is het niet erg nuttig, omdat alle code na *Exit* nooit uitgevoerd zal worden.