**H1: WAT IS PROGRAMMEREN?**

PROGRAMMEREN?

* Programmeren = computer opdracht geven
* Algoritme = recept dat je geeft aan de computer om op te lossen

Programmeertaal

* Ondubbelzinning: computer kan elk woord en opdracht maar op één manier interpreteren
* Bepaalde woordenschat
* Grammaticaregels

C# (C sharp)

= hogere programmeertaal (hoe hoger, hoe leesbaarder, hoe lager, hoe dichter tegen de machinetaal)

COMPILER

* Vertaalt het algoritme (geschreven in hogere programmeertaal) naar de machinetaal die de computer verstaat. Maakt dus de code gebruiksklaar voor de computer.
* Functie van compiler voor dit vak: **Het omzetten van C# code naar een uitvoerbaar bestand geschreven in IL code.** -> IL : Intermediate Language

COMPUTERS

* “A device which processes information according to instructions it has been given.”
* Zit tegenwoordig in allerlei hardware
* Wij willen spreken met de hardware:

Hogere programmeertaal (C#, Jave, etc) -> Machinetaal (Assembler) -> 0’en en 1’en (binair stelsel) -> stroom & spanning

Booleaanse algebra

* Bits
* 0 & 1 (binair)
* Digitalisering

Machinetaal, Assembler:

* Letterlijk de operaties/opdrachten die de computer moet doen om een werkend programma te hebben
* Heel specifiek

Hogere en lagere programmeertaal:

* Hoe lager, hoe dichter tegen de machinetaal, hoe meer 1 op 1 interactie met de machinetaal en accuratere performantie
* Hoe hoger, hoe leesbaarder in de eigen taal, hoe meer overhead en minder performantie

Data vs Information:

* Data: bits op een harde schijf zonder betekenis
* Information: interpretatie van die data
* Data kan op verschillende manieren geïnterpreteerd worden
* Jij als programmeur bepaalt waar de data voor staat!!!

Computer = DDV (Domme DataVerwerker)

* Data -> Computer -> Data (verwerkt)
* Analogie met de worstendraaier

De Software geeft een instructie aan het OS, en de OS moet dat vertalen naar de hardware. De software bekommert zich niet met de hardware.

* Het grootste deel van programmeren is NIET code schrijven, maar **probleemoplossing**!!!
* Probleembeschrijving -> Oplossing voor probleem -> programmeren van oplossing

**Algoritme** = logische stappenplan, de instructies die je geeft aan de computer. HET RECEPT.

* VOLGORDE is belangrijk
* Een eindige reeks instructies die vanuit een gegeven begintoestand naar een beoogd doel leidt.

**Onderdelen van een algoritme:**

* Opeenvolgingen (statements)
* Herhalingen (loops/lussen)
* Selecties (if/als)
* Methode-oproepen op objecten
* Kant-en-klaar
* Zelf geschreven

Wanneer programmeren:

* Jij moet de computer doen begrijpen wat je het wil laten doen
* Zelf het probleem eerst oplossen voor je programma kan schrijven dat het probleem oplost voor jou
* Computertaal is enkel een middel, en moet je niet loyaal aan zijn.
* **COMPUTERS ZIJN ALTIJD DOM**

Programmeertalen:

Waarom programmeertaal? Waarom geen Engels?

* Computers zijn te dom om Engels te verstaan
* Engels zou een slechte programmeertaal zijn
* Spreektalen zijn dubbelzinnig
* Er zijn honderden programmeertalen, **de taal op zich is niet zo belangrijk**

**Keywords** = eigen woordenschat, eigen sleutelwoorden die op één manier geïnterpreteerd worden

**Syntax** = eigen grammaticale regel

**C#**

* Lijkt heel hard op Java (en je zal Java dan ook begrijpen)
* Zeer relevante taal in het bedrijfsleven
* Heel veel dingen draaien op C#
* Heel toegankelijke taal (tegenover academische C – maar Je zal C ook kunnen leren)
* C# vind je overal in Microsoft: Windows 10, Azure, Xamarin, Android, zelfs iPhone

**.NET omgeving**

* Framework waarin C# onderdeel is
* Gigantische groep technologieën die toelaat voor ontwikkelaars om software te maken
* Gaat veel werk voor ons doen
* Vb: vroeger moest je zelf geheugen teruggeven aan de machine, vandaag doet .NET dat voor jou
* Runtime omgeving
* Beveiliging
* Netwerking
* .NET 5 komt uit in November 2020 (upgrade from .NET ore 3.0)

**Hoe wordt een programma gemaakt?**

* **C# Compiler** => zet je code om naar uitvoerbaar bestand in een taal die de OS/hardware begrijpt (machinecode => dit heet Low Level Instructions (bytecode) niet belangrijk)
* Je zou je code in Notepad kunnen schrijven (of VSC)
* Maar wij gebruiken **Visual Studio**
* De .NET Compiler zal je code kunnen compileren en omzetten naar uitvoerbaar bestand (.exe)

**H1: WERKEN MET VISUAL STUDIO**

IDE = Integrated Development Environment (bvb: Visual Studio)

* VSC = lightweight, minder features

IN VS:

* Wij gaan altijd werken met Console App (.NET Core)
* Er zijn 2 versies: Console APP C# en VB (visual basic)
* Wij kiezen altijd C#

Nieuw Project:

* Project name: altijd duidelijk maken wat voor project en goed organiseren!!!
* Solution name: hetzelfde als Project Name
* Vakje NIET aanvinken

In het projectvenster:

* Groene startknop: code door de compiler sturen en .exe genereren.
* Tabblad: elk bestand krijgt zijn eigen tabblad
* Solution explorer: alle bestanden en elementen die tot het huidige project horen
* Properties: alle eigenschappen van het huidige geselecteerde element

NOOIT “save as…” doen, altijd alles binnen de gemaakte Solution (map) houden!!!

**H1: JE EERSTE PROGRAMMA**

Typische flow van programma:

Input: Console.ReadLine (de user brengt dingen in, bvb via toetsenbord) -> C# Applicatie (verwerking) -> Console.WriteLine (oplossing tonen)

2 belangrijke functies:

* Console.ReadLine
* Console.WriteLine

Console => Shell/DOS applicatie (cmd.exe)

**Syntax:**

Console.WriteLine(“tekst”); -> aanhalingstekens heel belangrijk

Of

Console.WriteLine(5 + 10);

“;” -> elke opdracht eindigt met puntkomma

Static void Main(string[] args)  
{  
}

* Tussen de accolades staat het codeblok, hier schrijf je code binnen de “methode” (Main)

Alles boven de methode, de codeblok, daar BLIJF JE AF!

* Je schrijft je code enkel binnen het codeblok

Using System;

* Alle Console. Commando’s zitten in de System bibliotheek, zo hoeven we niet “System.Console.” te schrijven heel de tijd

Namespace Demo1

* De naam van het project

Class Program

* Hier start het programma, alles hierbinnen wordt gecompileerd.
* Je moet ALLES juist schrijven, inclusief hoofdletters!

Console.WriteLine(“tekst”);

* schrijft één regel aan tekststring of andere output
* de cursor zal altijd naar het begin van de volgende lijn gaan

Console.Write(“tekst”);

* Output tekst maar niet op één enkele regel, gewoon na elkaar. Verschillende Console.Write() commando’s zullen op dezelfde regel output geven

string [input];

* Je vertelt het programma om geheugen vrij te maken voor user input, je geeft een naam aan de input van de user, het wordt met die naam opgeslagen in het geheugen
* LET OP: het lijkt erop dat je geen cijfers kunt steken in de naamgeving!
* zal outputten wat de user input in stringvorm (tekst dus)
* string: cijfers, letters, speciale karakters. Alles uit het toetsenbord zijn strings.

String [input] = “tekst”;

* de input wordt opgeslagen als “tekst” en er kan zo verwezen worden naar [input] in andere delen van de code.

[input] = Console.ReadLine();

* deze commando zorgt ervoor dat de user [input] kan inputten die door de “string” command in tekstvorm zal outputten
* als je deze alleen schrijft, kun je iets schrijven maar het wordt nergens opgeslagen
* het zal de input van de user gebruiken tot aan Enter, waar het opgeslagen wordt moet JIJ programmeren.

**Je kunt de twee vorige commando’s verkort schrijven door:**

String [input]= Console.ReadLine();

Console.WriteLine(“Tekst” + [input] + “tekst”);

* met de + tekens kun je het resultaat van de input toevoegen in een tekstregel. Dit bespaart ruimte in het codeblok

**H1: FOUTEN IN JE CODE**

* squiggly line in de code zelf toont foute of ontbrekende code aan.
* onderaan is er een Error List waar jouw errors staan, je kunt er ook op klikken om naar de relevante regel te gaan.
* Dubbelklik op de error om de cursor op de relevante regel te zetten.
* Lightbulb links van de code geeft suggesties voor verbeteringen (ook al is er geen fout), wees wel kritisch want VS is niet alwetend.

Als je code probeert te runnen met een fout in, krijg je waarschuwingsvenster dat je vraagt of je in de plaats de laatste CORRECTE code wilt runnen. NIET DOEN.

**H1: KLEUREN IN CONSOLE**

* Enkele commando’s om kleuren in je console te zetten. Let op dat dit vrij gelimiteerd is en geen nut heeft buiten esthetiek voor de eindgebruiker.

Console.ForegroundColor= ConsoleColor.[kleur];

* De kleur die de letters krijgen

Console.BackgroundColor= ConsoleColor.[kleur];

* De kleur die de achtergrond zal krijgen

Console.ResetColor();

* Reset de kleuren terug naar de standaardwaarden van de computer.