**H2: DE ESSENTIE VAN C#**

C# bestaat uit opeenvolging van instructies/commando’s => **statements**

Statements eindigen in ; (puntkomma)

* Volgorde van woorden zijn niet vrijblijvend en moeten voldoen aan grammaticaregels

BELANGRIJKE REGELS:

* Hoofdletter-gevoelig
* Statements afsluiten met ;
* Witruimtes worden genegeerd TENZIJ ze binnen “ “ staan => dus je kunt tabs, spaties en enters gebruiken OM JE CODE OP TE SCHONEN!!
* Commentaar toevoegen: // (voor één regel) OF /\* \*/ (voor meerdere regels)
* **“Main()”** is het startpunt

Keywords: (NIET hetzelfde als commando’s zoals WriteLine)

* If/else true/false string, etc
* Dit is de WOORDENSCHAT, ze heten **reserved keywords**
* Elk keyword heeft een eenduidige, specifieke betekenis
* Er zijn momenteel 80 keywords
* GEBRUIKT **NOOIT** “goto” => dit keyword zorgt ervoor dat er naar een bepaald deel in de code gegaan wordt, en alles dat overgeslagen wordt zal niet geïnterpreteerd worden

Variabelen:

* Syntax: *datatype identifier;*
* Dit gebruiken we om tijdelijke data in op te slaan
* Bvb wanneer de gebruiker een input geeft, willen we die input **opslaan** om er later in het programma/algoritme er iets mee te doen. Dan verwijzen we naar deze variabele.
* **Variabele is een locatie in het geheugen waar je data opslaat, met specifieke NAAM, specifieke GROOTTE en specifiek DATATYPE**
* BVB: “tel 3+4 en vermenigvuldig dan met 5” => het resultaat van 3+4 bewaren we in een variabel, dan doen we *variabel (3+4) \* 5* en het resultaat daarvan bewaren we dan weer in een nieuwe *variabel ((3+4) \* 5).* Dat nieuwe variabel kunnen we dan naar het scherm sturen bvb met Console.WriteLine();
* **Identifier** = de naam van de variabele
* **Datatype** = geeft aan welk type de data is die we willen opslaan (tekst, getal, afbeelding, etc)

Enkel da soort data zal dan opgeslagen kunnen worden in de variabele.

Regels voor **identifiers**:

* Hoofdlettergevoelig (zelfde woord met en zonder hoofdletter worden als 2 verschillende identifiers gezien)
* GEEN keywords (identifiers mogen niet één van de 80 keywords zijn, maar wel als de hoofdletters op andere plaatsen worden gezet)
* Eerste karakter mag enkel kleine/grote letter of \_ zijn (dus GEEN cijfer)
* Alle andere karakters mogen enkel kleine/grote letter, \_ of cijfers zijn (dus GEEN spatie)
* Mag zo lang zijn als je wilt
* CODING RICHTLIJNEN:
* Duidelijke naamgeving: maak je identifiers duidelijk en relevant aan de verwachte input.
* **Camel Casing**: in een identifier met meerdere woorden gebruik je een hoofdletter bij elk nieuw woord, MAAR eerste letter van het eerste woord wel in kleine letter:

bvb: apHogeSchool of groteBoekenInhoud

Commentaar:

* Één regel: //
* Kun je op een eigen regel doen of gewoon NA een stuk code
* Kun je ook VOOR code zetten om deze uit te schakelen
* Meerdere regels: /\* \*/
* Visual Studio heeft ook een “comment/uncomment” knop

DEBUGGING (dit zal nog in een apart hoofdstuk verschijnen)

* Iets wat wij heel veel gaan mogen doen 😊
* Fouten uit de code halen
* Een goede programmeur is een luie programmeur: hoe minder code je moet schrijven om het programma te laten doen wat het moet doen, hoe minder kans er is op bugs/fouten
* Voor iedere 10 minuten dat je code schrijft, zal je minstens DUBBEL ZO LANG bezig zijn met debuggen
* Bug: NIET hetzelfde als syntax/woordenschatfouten.
* Als de code compileert, maar er fouten in het programma zijn = **bug**
* Fixing **runtime** errors (= errors die ontstaan ondanks dat de code compileert)
* Test code, change code, repeat
* Debuggen is een belangrijkere skill dan code schrijven!

Kleine tip:

VS -> Edit -> Advanced -> Format Document

* Dit al proberen je code deftig te schikken

**H2: DATATYPES**

* Ieder datatype kan één welbepaald soort data bewaren en elk datatype zal een bepaalde hoeveelheid geheugen vereisen.

**Strongly typed language**

* Een variabele kan maar aan één datasoort tegelijk gebonden zijn, dus het is belangrijk om dit te specificeren
* Weakly typed language (JavaScript): De variabele kan allerlei soorten datatypes weergeven (dit kan miserie veroorzaken)

PRIMITIEVE DATATYPES

* (in Semester 2 gaan wij onze eigen datatypes leren maken!)
* Basistypes van C#, wij gaan leren werken met datatypes voor:
* Gehele getallen: **sbyte, byte, short, ushort, int, uint, long, ulong** (deze datatypes gaan van weinig ruimte voor geheugen naar veel geheugen)
* Kommagetallen: **double, float, decimal**
* Tekst: **char, string** (deze ken je al)
* Booleans: **bool** (deze is voor true/false statements te maken. (denk aan binair 1/0))
* Enums: **een speciaal soort datatype dat een combinatie van meerdere datatypes bevat**
* Ieder datatype bevat volgende EIGENSCHAPPEN:
* Soort data (tekst, getal, etc)
* Geheugengrootte (hoeveel bits dat 1 element van dit datatype inneemt in het geheugen)
* Schrijfwijze vd **literals** => hoe weet C# of je “2” een kommagetal (2.0) of een geheel getal (2) is?

Basistype voor tekst: *string*

Basistype voor getallen:

* Onderscheid tussen gehele getallen en kommagetallen

GEHELE GETALLEN: *sbyte, byte, short, ushort, int, uint, long, ulong,* ***char*** *(dit datatype bewaart karakters)*

* INTEGERS
* Elk datatype hierboven heft een bepaald bereik en hoeveelheid geheugen nodig.
* **DUS: Kies steeds bewust het datatype dat het best past bij uw probleemoplossing qua bereik, precisie en geheugengebruik**
* **VERMIJD zo verspilling van kostbaar geheugen!**

**Een Bit is een meeteenheid voor één logische staat die ofwel 1, 0, +/-, true/false, yes/no, on/off is**

byte:

* Neemt 8 bits geheugen in
* Bereik van 0 tot 255

sbyte:

* Zoals een byte, maar **signed** (daarom de “s” voor “byte”)
* “signed” betekent dat er een teken voor de waarde staat, deze is dus + of -.
* Er wordt 1 bit gebruikt om het (signed) teken voor de waarde te zetten
* DUS: er zijn nog maar 7 bits over om een waarde te stellen
* DUS: bereik is -128 tot 127. (de helft van 256 maar wel in beide richtingen!!)

short:

* Neemt 16 bits geheugen in
* Deze is al “signed”, dus 1 bit wordt gebruikt voor het +/- teken, 15 bits voor de waarde te stellen.
* DUS: bereik is -32 768 tot 32 767

ushort:

* Neemt 16 bits geheugen in (zoals short)
* Deze is **unsigned** (daarom de “u” voor “short”)
* Er staat GEEN teken voor de waarde
* DUS: **altijd positief**
* DUS: bereik is 65 535 (dubbele van “short” in de positieve richting)
* ONTHOU: sommige datatypes hebben “s” prefix (signed): sbyte, meeste hebben “u” prefix (unsigned): ushort, uint, ulong

int:

* Neemt 32 bits geheugen in
* Deze is “signed” dus 1 bit wordt gebruikt voor een teken, 31 bits worden gebruikt om een waarde te stellen.
* DUS: bereik van -2 147 483 648 tot 2 147 483 647 (2 miljard 150 miljoen)
* Deze wordt onze vriend!

uint:

* Neemt 32 bits geheugen in
* Unsigned
* DUS: bereik is dubbele van “int” in de positieve richting

long:

* Neemt 64 bits geheugen in (zoals veel programma’s these days)
* Signed
* Bereik: belachelijk hoog in beide richtingen

ulong:

* 64 bits
* Unsigned
* Bereik: dubbel zo hoog als “long” in de positieve richting.

KOMMAGETALLEN: *float, double, decimal*

* REALS (reële getallen)
* Maar 3 mogelijkheden, elk datatype hier heeft zijn voordelen en nadelen, alle 3 evenwaardig.
* Er moet afweging gemaakt worden omdat er oneindig veel getallen na de komma zijn.

PRECISIE: hoeveel digits je na de komma te zien krijgt. Hoe meer, hoe preciezer

float:

* 32 bits **(neemt minste plaats in geheugen)**
* Bereik: gemiddeld
* Precisie: 7 digits (na de komma)

double:

* 64 bits
* Bereik: meeste **(grootste en kleinste getallen)**
* Precisie: 15 digits

decimal:

* 128 bits
* Bereik: minste
* Precisie: 28-29 digits **(meeste digits na de komma)**
* MEESTAL gaan wij bij kommagetallen *double* gebruiken en bij gehele getallen *int*.

BOOLEAN DATATYPE: *bool*

* Bevat maar twee mogelijke waarden: true/false.
* Heeft maar 1 bit in het geheugen nodig
* Wij gaan dit datatype veel gebruiken wanneer we met beslissingen werken. Namelijk *if* statements die afhankelijk zijn van een *bool* waarde.
* PAS OP: als je weet dat je variabele maar 2 mogelijke waarden zal hebben, gebruik dan *bool* en niet *int*!!!

TEKST/STRING DATATYPE: *string, char*

string:

* Hier kun je tekst in bewaren
* Duid je aan met “ “

char:

* Één enkel karakter kan hierin bewaard worden.
* Duid je aan met ‘ ‘

Oefening: welke datatype kies je voor volgende variabelen:

* Temperatuur in °C: *short*
* Temperatuur in K: *ushort* of *uint*
* Aantal mensen op aarde: *ulong*
* Aantal klasgenoten: *byte*
* BMI: *sbyte*

**Reële getallen (literals) moeten een aanduiding krijgen om de compiler te laten weten tot welk soort datatype ze behoren!!!**

* DUS: als je 2.5 noteert moet de compiler weten welk datatype het toebehoort:
* Float: 2.5f
* Double: 2.5 (hier ze je niets achter)
* Decimal: 2.5m
* Je mag ook Exponent-notatie gebruiken:
* Float: 9.45E8f
* Double: 9.45e

**H2: VARIABELEN**

Variabele

* Plekje in het geheugen dat in je programma gereserveerd wordt om daarin data te bewaren van een bepaald type (toegekend door het datatype)
* Heeft intern een geheugenadres, wij verwijzen naar dat geheugenadres door middel van een **identifier**

DECLAREREN

* Wij declareren een variabele door het een type (*datatype*) en een naam (*identifier*) te geven
* Optioneel kan de variabele ook een beginwaarde krijgen

*Datatype identifier = value;*

* “= waarde” is optioneel

Je mag ook MEERDERE variabelen met hetzelfde datatype aanmaken in 1 declaratie:

*datatype identifier1, identifier2, identifier3*

vb: string voornaam, achternaam, adres;

Waarde toekennen:

* De toekenning gebeurt van **RECHTS NAAR LINKS!**
* Je kunt een variabele de waarde geven van een andere variabele!

Vb:

Int MijnLeeftijd = 23;

int AndereLeeftijd = MijnLeeftijd;

**LITERALS:**

* Een expliciete waarde die je geeft aan een variabele

Vb: temperatuurGisteren = 20; (20 is de expliciete waarde van “temperatuurGisteren”)

* Als data echter uitgelezen wordt of berekent, en het resultaat daarvan toegekend wordt aan een variabele, dan is dat GEEN literal!

Suffixen voor literals van bepaalde datatypes:

* uint: 125U/u
* long: 125L/l
* ulong: 125 UL/ul
* float: 12.5F/f
* decimal: 12.5M/m
* voor de andere numerieke datatypes zijn er geen suffixen

Andere literals:

bool: true/false

char: ‘z’

string: “z”

Hexadecimale notatie: 0x/0X

* vb: int myAge = 0x0024;

Binaire notatie: 0b/0B

* vb: int myAge = 0b001001000;

**LET OP**:

Je kan NIEUWE waarden overschrijven over OUDE waarden:

Vb:

*int temperatuurGisteren = 25;*

*temperatuurGisteren = 20;*

VEELGEMAAKTE FOUT:

* een variabele meer dan één keer **declareren**:

vb: int leeftijd = 23;

…

Int leeftijd = 24;

* Dit zal een foutmelding opleveren.

Verbetering: int leeftijd = 23;

leeftijd = 24;

**TIPS:**

* Geef je variabelen ALTIJD meteen een beginwaarde!
* Maak je variabelen aan ZO DICHT MOGELIJK bij het punt waar je ze gaat gebruiken!
* *string tekst = “ “;* => dit is ook een ingevulde waarde!

**H2: EXPRESSIES EN OPERATORS**

* eenvoudige tot complexe sequenties van **bewerkingen** die op **1 resultaat** uitkomen met een specifieke datatype. Vb: 3+2 = 5 (int)

Toewijzen:

* Meestal maak je bewerkingen met variabelen

Vb: *int temperatuurVerschil = temperatuurGisteren – temperatuurVandaag;*

* Maar je kan ook bewerkingen maken met literals

Vb: *int temperatuursVerschil = 21 – 25;*

* Je kunt ook combineren

Vb: *int breedte = 15;*

*Int hoogte = 20 \* breedte;*

OPERATORS

* Welgekende wiskundige bewerkingen:
* Optellen (+)
* Aftrekken (-)
* Vermenigvuldigen (\*)
* Delen (/)

Deze bewerkingen volgen de VOLGORDE VAN BEWERKINGEN:

1. Haakjes
2. Vermenigvuldigen, delen, modulo/rest na deling
3. Optellen, aftrekken

Syntax:

*operand operator operand* =

vb: 5(operand) +(operator) 5(operand) =

MODULO OPERATOR

* Syntax: *%*
* Deze operator zal je de rest geven van een deling tussen twee **gehele** getallen
* Geeft de rest terug na deling van de linkse door de rechtse operand!
* 13%3 => je kan 12 delen door 3 en dan hou je 1 over, dus rest =1
* 96%10 = 6 want er passen 90 bananen in 10 dozen
* **Handig om te weten of een getal even/oneven is**
* **Handig om te weten hoeveel je van iets kunt verdelen en wat er overschiet**

Vb: 7%2 zal als rest 1 geven

10%5 zal als rest 0 geven

* Deze operator is handig om te weten of bvb een getal een veelvoud is van iets. Wanneer de rest 0 is weet je dat het een veelvoud is van het getal waar je het door deelde.

**Verkorte notatie voor operators:**

*Getal++; => getal= getal+1;* Variabele verhogen met 1

*Getal--; => getal= getal-1;* Variabele verlagen met 1

*Getal+=3; getal= getal+3;* Variabele verhogen met een getal (in dit geval met 3)

*Getal-=6; getal= getal-6;* Variabele verminderen met een getal (in dit geval met 6)

*Getal\*=7; getal= getal\*7;* Variabele vermenigvuldigen met een getal (in dit geval met 7)

*Getal/=2; getal= getal/2;* Variabele delen door een getal (in dit geval met 2)

**LET OP!!! De datatypes die je in je expressies gebruikt BEPALEN het datatype van het RESULTAAT van die expressies!**

* int = int + int
* NIET int = double + double => ERROR! (vb int result= 3.4 + 2.5)

**DE VARIABELEN BEPALEN HET DATATYPE!**

* int gedeeld door een int zal een int, dus geheel getal, als resultaat geven!
* Alle info achter de komma raakt verloren!
* Int result= 9/2; => dit wordt 4 (maar het eigenlijke antwoord is 4.5)

DATATYPES MENGEN:

* Je kunt bvb een berekening maken met een double en een int.
* C# zal ALTIJD de “grootste” datatype kiezen: in dit geval een double.
* Vb: double result= 3/5.6; => deze code WERKT
* Vb: int result= 3/5.6; => deze code WERKT NIET!

**ONTHOU:** JIJ moet beslissen wat het datatype is, gebaseerd op welk datatype het resultaat van de expressie wordt!!! Dit gebeurt niet automatisch! JIJ MOET DAT DOEN!

BESTE VOORBEELD:

*double helft= 10000 \* (1/2);*

* Geeft als resultaat 0, omdat “(1/2)” genoteerd is in GEHELE GETALLEN (int) en dus het resultaat in int zal printen! 10000 \* 0 = 0

*double helft= 10000 \* (1.0/2);*

* Geeft als resultaat 5000, omdat “(1.0/2)” nu wel genoteerd is in REËLE GETALLEN (double) en C# pakt het ‘grootste’ datatype (double) en print het resultaat in double! 10000 \* 0.5 = 5000

Waarden verwisselen van 2 variabelen:

Int getalA= 6;  
int getalB= 8;

* FOUT: *int getalA= getalB*
* JUIST:

*Int temp = getalA*

*Int getalA= getalB*

*getalB= temp*

* Snap je?

**Keyword**: *const*

* Staat voor “constante”, een waarde die niet kan/mag veranderen, bvb zwaartekracht op aarde, aantal dagen in een jaar, etc
* Vb: *const double MAX\_TEMP= 100.5;*
* Ongeschreven regel: identifiers voor *const* worden in ALL CAPS geschreven!!!
* Ook wel “magic number” genoemd, het kan NIET perongeluk veranderd worden door jouw code!
* Dient puur voor jou als programmeur om je te vrijwaren dat er bugs verschijnen omdat je een constante operand perongeluk verandert!