

Oefeningenbundel C#

Syntra West

2018-2019

Inhoudtafel

[Building Block I - Datatypes 3](#_Toc528016509)

[Opdracht 1.1 – Ingave en weergeven van naam en adres **(T)** 3](#_Toc528016510)

[Opdracht 1.2 Berekenen van de omtrek en de oppervlakte van een rechthoek **(T)** 3](#_Toc528016511)

[Opdracht 1.3 Aankopen van groenten en fruit **(T)** 4](#_Toc528016512)

[Opdracht 1.4 – Berekenen van de kostprijs voor het verven van deuren **(T)** 5](#_Toc528016513)

[Building Block II - Operatoren 6](#_Toc528016514)

[Opdracht 2.1 – Berekenen van de absolute waarde van een getal **\* (Z)** 6](#_Toc528016515)

[Opdracht 2.2 – Bepalen of 2 ingevoerde getallen gelijk zijn \* (Z) 6](#_Toc528016516)

[Opdracht 2.3 – Converteer het aantal graden Celcius naar graden Farenheit **\* (Z)** 6](#_Toc528016517)

[Opdracht 2.4 – Converteer het aantal graden Farenheit naar graden Celcius **\* (K)** 7](#_Toc528016518)

[Opdracht 2.5 – Bereken het aantal uren, minuten en seconden **\*\* (K)** 7](#_Toc528016519)

[Opdracht 2.6 – Bepaal duizendtallen, hondertallen tientallen, eenheden van een getal **\*\* (Z)** 7](#_Toc528016520)

[Opdracht 2.7 – Bereken hoeveel dagen tot nieuwjaar **\*\* (T)** 8](#_Toc528016521)

[Opdracht 2.8 – Converteer inch naar centimeter **\* (Z)** 8](#_Toc528016522)

[Opdracht 2.9 – Converteer van cm naar inch **\* (T)** 8](#_Toc528016523)

[Opdracht 2.10 – Omwisselen van 2 getallen **\* (Z)** 8](#_Toc528016524)

[Opdracht 2.11 – bereken prijs inclusief BTW **\*\* (K)** 9](#_Toc528016525)

[Opdracht 2.12 – Bereken prijs exlusief, inclusief BTW met korting **\*\* (Z)** 9](#_Toc528016526)

[Opdracht 2.13 – BTW nummer validatie **\*\* (T)** 10](#_Toc528016527)

[Opdracht 2.14 – Scores **\* (Z)** 11](#_Toc528016528)

[Opdracht 2.15 – Snoepautomaat **\*\*\* (T)** 11](#_Toc528016529)

[Opdracht 2.16 – Even en oneven **\* (Z)** 12](#_Toc528016530)

[Opdracht 2.17 – Kwadraat **\* (Z)** 12](#_Toc528016531)

[Opdracht 2.18 – Uren, overuren, brutoloon **\* (T)** 12](#_Toc528016532)

[Opdracht 2.19 – Bereken het gewicht voor een constructie **\* (T)** 13](#_Toc528016533)

[Opdracht 2.20 – Remafstand **\* (T)** 13](#_Toc528016534)

[Building Block III - Selectie 14](#_Toc528016535)

[Opdracht 3.1 – Absolute waarde **\* (Z)** 14](#_Toc528016536)

[Opdracht 3.2 – hetzelfde getal **\* (Z)** 14](#_Toc528016537)

[Opdracht 3.3 zoek het grootste getal **\* (Z)** 14](#_Toc528016538)

[Opdracht 3.4 even of oneven **\* (Z)** 15](#_Toc528016539)

[Opdracht 3.5 Cinema **\*\* (Z)** 15](#_Toc528016540)

[Opdracht 3.6 Vrachtwagen **\*\* (K)** 16](#_Toc528016541)

[Opdracht 3.7 – Portkosten **\*\* (Z)** 17](#_Toc528016542)

[Opdracht 3.8 Multiple choice **\* (Z)** 18](#_Toc528016543)

[Opdracht 3.9 Dag van de week **\* (Z)** 19](#_Toc528016544)

[Opdracht 3.10 Aantal dagen per maand **\*(\*) (Z)** 19](#_Toc528016545)

[Opdracht 3.11 Leeftijd **\* (Z)** 20](#_Toc528016546)

[Opdracht 3.12 Romeins getal **\* (Z)** 20](#_Toc528016547)

[Opdracht 3.13 Cirkel, rechthoek, vierkant **\*\*\* (T)** 20](#_Toc528016548)

[Opdracht 3.14 decimaal naar binair **\* (K)** 21](#_Toc528016549)

[Opdracht 3.15 decimaal naar hexadecimaal **\* (Z)** 21](#_Toc528016550)

[Opdracht 3.16 Omzetting talstelsels \*\*\* (T) 22](#_Toc528016551)

[Opdracht 3.17 Temperatuur **\* (Z)** 22](#_Toc528016552)

[Opdracht 3.18 Duurtijd **\*\* (Z)** 23](#_Toc528016553)

[Opdracht 3.19 Factuur **\* (Z)** 23](#_Toc528016554)

[Opdracht 3.20 Deling **\*\* (Z)** 23](#_Toc528016555)

[Opdracht 3.21 Eenvoudige sortering **\*\* (K)** 24](#_Toc528016556)

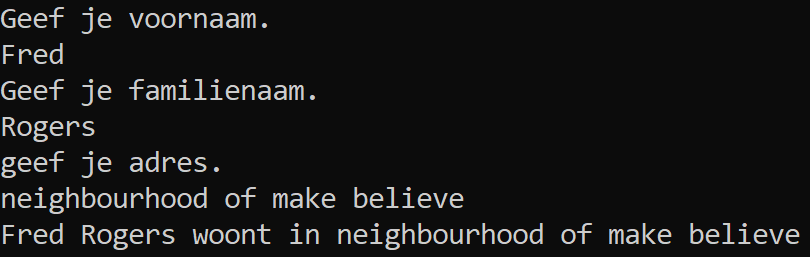
[Opdracht 3.22 Examens **\* (Z)** 24](#_Toc528016557)

[Opdracht 3.23 Kindergeld **\*\* (Z)** 25](#_Toc528016558)

# Building Block I - Datatypes

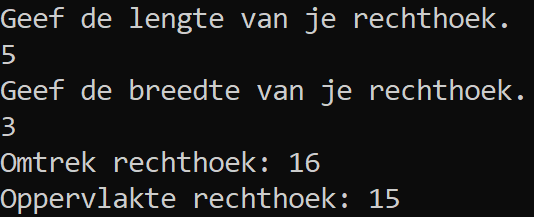
## Opdracht 1.1 – Ingave en weergeven van naam en adres **(T)**

Schrijf een applicatie die aan de gebruiker zijn naam, voornaam en adresgegevens (type string) vraagt en deze ingevoerde data samengevat teruggeeft op het scherm. De uitvoer wordt in één commando geprogrammeerd.



## Opdracht 1.2 Berekenen van de omtrek en de oppervlakte van een rechthoek **(T)**

Lees de lengte (**int**) en de breedte (**int**) van een rechthoek in via het toetsenbord. Bereken de oppervlakte (**double**) en de omtrek (**double**) van deze rechthoek en druk de berekende resultaten af op het scherm.



## Opdracht 1.3 Aankopen van groenten en fruit **(T)**

Theodore koopt 2 kg appels, 5 kg aardappelen, 3.5 kg peren en 1 bakje aardbeien op de markt. De appels kosten 2.4€/kg, de aardappelen 0.75€/kg, de peren 1.8€/kg en de aardbeien 2.84€/bakje.

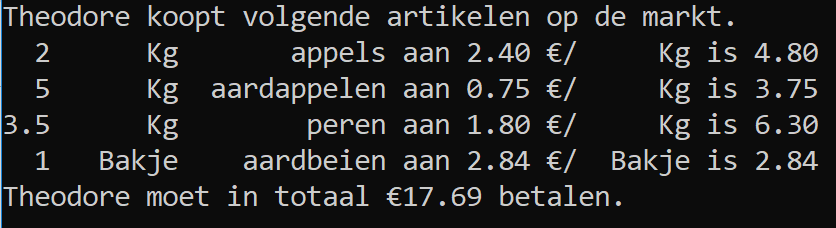
Eisen voor de programmeeromgeving:

* Alle variabelen worden hardgecoded.
* Geef zelf het juiste datatype.

Eisen voor de uitvoer:

* Duidelijk overzicht welke groenten/fruit gekocht (in uitvoer in **drukletters**)
* Prijs/kg of prijs/bakje van de gekochte groente/fruit
* Zorg dat alles overzichtelijk staat (op een nieuwe lijn)
* Het totaal wordt weergegeven met 2 cijfers na de komma.
* Zorg dat in de uitvoerstring de variabelen opgenomen worden.

Zorg ervoor dat bij de output, het aantal kg/bakjes, de naam van de groente/fruit en de prijs/kg, prijs/bakje mooi onder elkaar gerangschikt staan.



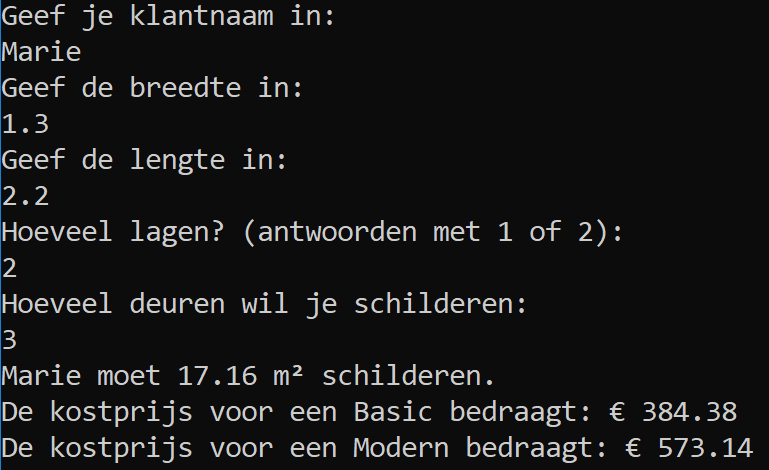
## Opdracht 1.4 – Berekenen van de kostprijs voor het verven van deuren **(T)**

Situering.

We gaan in het lab een kostprijsberekening uitvoeren voor het schilderen van een deur.

De vereisten van het programma zijn:

* De gebruiker moet zijn naam kunnen ingeven
* De gebruiker moet de breedte en hoogte aangeven van de deur
* De gebruiker moet aangeven hoeveel lagen verf hij wil gebruiken (hij kan kiezen tussen 1 of 2 lagen)
* De gebruiker moet aangeven welke uitvoering hij wil (hij kan kiezen tussen basic(€ 22.4) en modern (€ 33.4). Elke uitvoering heeft een andere prijs.
* Het programma rekent de oppervlakte uit.
  + Aanname :
    - alle deuren zijn even breed en hoog
    - er zitten geen ramen in de deuren
    - wanneer de gebruiker een keuze heeft gemaakt voor het aantal lagen en de uitvoering zal deze keuze gelden voor alle deuren.
* Het programma rekent de kostprijs uit die de gebruiker moet betalen. Hiervoor werd rekening gehouden met de oppervlakte en de uitvoering.
* De gebruiker krijgt hiervan een melding.
* Wat is de formule om de oppervlakte van een rechthoek te berekenen?
* Hoe zou je de berekening uitvoeren om de hoeveelheid verf dat je zal nodig hebben? (rekening houdend met het aantal deuren én het aantal lagen)

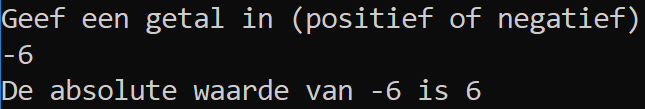


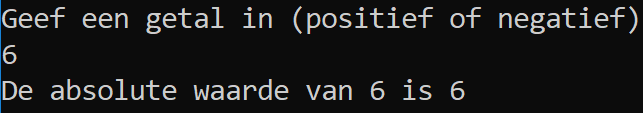
# Building Block II - Operatoren

## Opdracht 2.1 – Berekenen van de absolute waarde van een getal **\* (Z)**

Bepaal de absolute waarde van een getal. (De absolute waarde is de waarde van een getal zonder teken). De gebruiker kan zowel een positief als een negatief getal ingeven. Het resultaat is altijd een positief getal. We maken hierbij gebruik van het type **short** voor ons getal dat we ingeven.

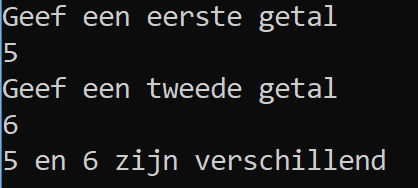
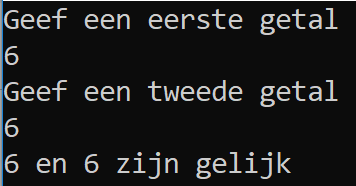
Maak hierbij geen gebruik van de speciale functie Math.Abs() die binnen C# aanwezig is.





## Opdracht 2.2 – Bepalen of 2 ingevoerde getallen gelijk zijn \* (Z)

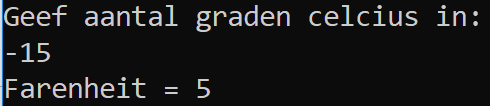
De gebruiker geeft 2 getallen in (zorg dat dit gelijk welk type kan zijn). Ga na of de 2 getallen die ingegeven zijn, hetzelfde getal zijn.

## Opdracht 2.3 – Converteer het aantal graden Celcius naar graden Farenheit **\* (Z)**

Maak een programma waarbij het aantal graden Celsius wordt ingegeven en deze wordt omgezet naar graden Farenheit. Bereken eveneens het aantal graden Kelvin.

Farenheit = celsius \* 1.8 + 32.



## Opdracht 2.4 – Converteer het aantal graden Farenheit naar graden Celcius **\* (K)**

Maak een programma waarbij het aantal graden Farenheit wordt ingegeven en deze wordt omgezet naar graden Celsius. (Rond af naar een geheel getal).

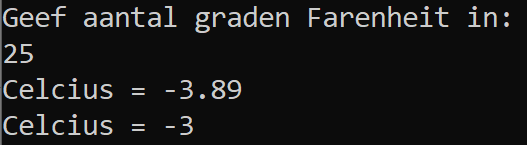
Geef hierbij in commentaar de formule voor de omzetting.

C = 5/9 x (F - 32)

Druk het resultaat af, afgerond naar een geheel getal.

Druk het resultaat vervolgens decimaal af.

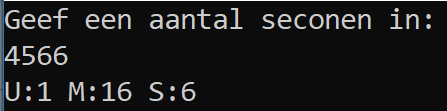
Let op met afrondingsfouten en het gebruik van int versus float voor tussenresultaten!



## Opdracht 2.5 – Bereken het aantal uren, minuten en seconden **\*\* (K)**

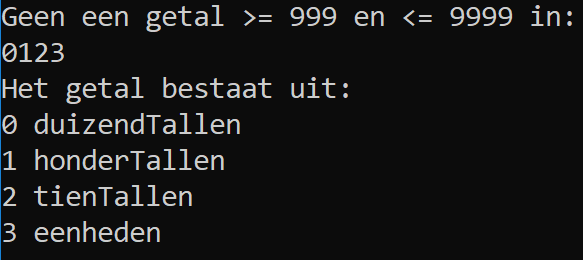
Maak een programma dat een geheel aantal seconden vb 5924 omrekent in uren, minuten en seconden. Het resultaat zou er ongeveer zo moeten uitzien: U:1 M:38 S:44.

Maak dit programma door gebruik te maken van variabelen, alsook door gebruik te maken van de operatoren.



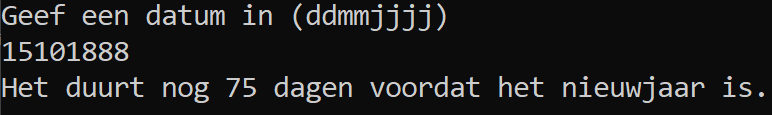
## Opdracht 2.6 – Bepaal duizendtallen, hondertallen tientallen, eenheden van een getal **\*\* (Z)**

Lees een getal in van 4 cijfers. Lees dit getal in als een **string**. Druk op het scherm af welk cijfer het duizendtal, hondertal, tiental en eenheden is van het ingelezen getal. Voor het duizendtal, hondertal, tiental en eenheid maken we gebruik van **int**.



## Opdracht 2.7 – Bereken hoeveel dagen tot nieuwjaar **\*\* (T)**

Geef een datum in via het toetsenbord in de vorm ddmmjjjj. Bereken hoeveel dagen het nog duurt voor het terug nieuwjaar is. Er vanuit gaande dat elke maand in het jaar 30 dagen telt.

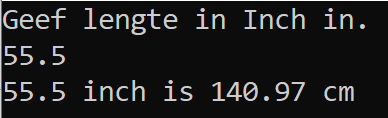


## Opdracht 2.8 – Converteer inch naar centimeter **\* (Z)**

Schrijf een programma dat de gebruiker om een lengte in inch vraagt en de lengte in centimeter afbeeldt.

Maak hierbij gebruik van de constante:

const float cmPerInch = 2.54f

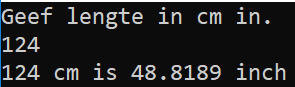


## Opdracht 2.9 – Converteer van cm naar inch **\* (T)**

Schrijf een programma dat de gebruiker om een lengte in cm vraagt en de lengte in inch afbeeldt

Maak hierbij weerom gebruik van de constante:

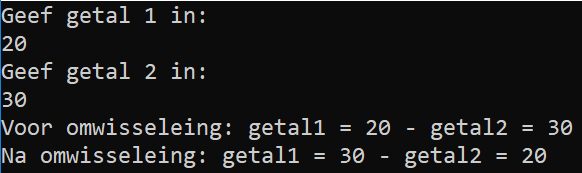
const float cmPerInch = 2.54f



## Opdracht 2.10 – Omwisselen van 2 getallen **\* (Z)**

Schrijf een programma met twee int variabelen a en b dat vervolgens hun waarden omwisselt

Tip : Denk hoe je een glas cola en een glas limonade kunt omwisselen.

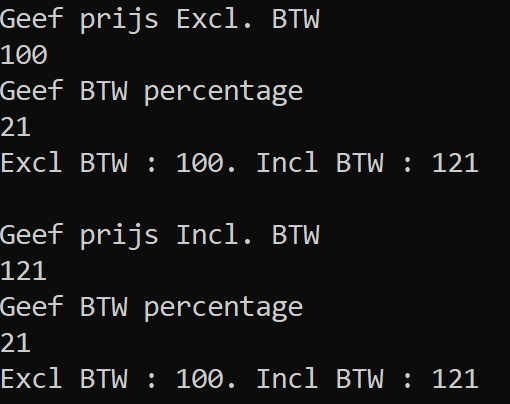


## Opdracht 2.11 – bereken prijs inclusief BTW **\*\* (K)**

Schrijf een programma dat de gebruiker vraagt om een prijs exclusief BTW en het BTW percentage (bv. 21).

Het programma toont dan de prijs excl. btw, het btw bedrag en de prijs incl. BTW.

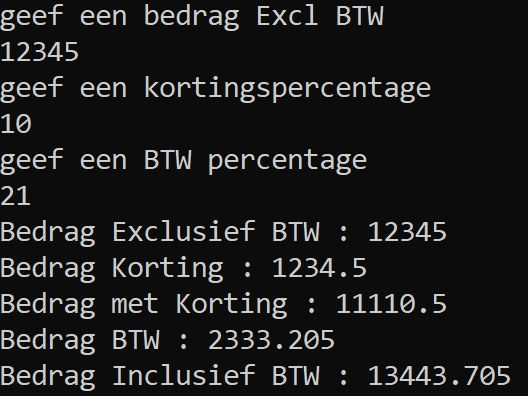
Geef vervolgens een bedrag in inclusief BTW en bereken het bedrag exclusief BTW. Druk het resultaat af.



## Opdracht 2.12 – Bereken prijs exlusief, inclusief BTW met korting **\*\* (Z)**

Schrijf een programma dat de gebruiker vraagt om een prijs exclusief BTW, een kortingspercentage (vb. 4.5) en het BTW percentage (vb. 21).

Het programma toont dan de prijs exclusief BTW, het kortingsbedrag, de prijs exclusief BTW, het BTW bedrag en de prijs inclusief. BTW.



## Opdracht 2.13 – BTW nummer validatie **\*\* (T)**

Schrijf een programma waarin je een BTW nummer moet ingeven. Geef weer of dit een geldig of een ongeldig BTW nummer betreft.

De controle van een BTW-nummer:

Een BTW-nummer bestaat uit 9 cijfers. Daarvan zijn de eerste 7 cijfers het eigenlijke BTW-nummer, de laatste 2 cijfers vormen een controlegetal.

Om een BTW-nummer te valideren neemt men de rest van de deling van het eigenlijke BTW-nummer (de eerste 7 cijfers) door 97.

Het controlegetal moet gelijk zijn aan 97 – de rest.

Voorbeeld: BTWnummer 476306127

BTWH

-nummer = 4763061

Controlegetal = 27

Rest na deling door 97. 4763061 % 97 = 70

97 – 70 = 27 = controlegetal

Geldige BTW-nummers:

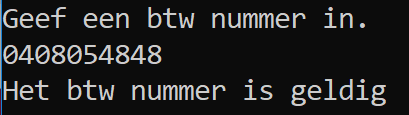
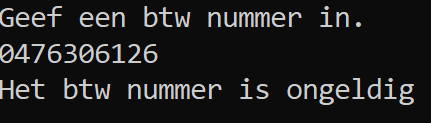
0408054848

0476306127

0673362122

0426996671

0213252520

## Opdracht 2.14 – Scores **\* (Z)**

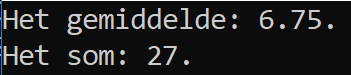
Een student behaalt op een examen volgende scores: 8, 6, 9 en 4.

Maak een programma dat het gemiddelde berekent (als decimaal getal) en het behaalde percentage (als je weet dat de maximum score op ieder vak 10 is).

Bereken en toon eveneens de som van deze 4 getallen.

Bereken ook het percentage als je weet dat de maximum score gelijk is aan 10.

Gebruik geen array’s mocht je daar al kennis van hebben.



## Opdracht 2.15 – Snoepautomaat **\*\*\* (T)**

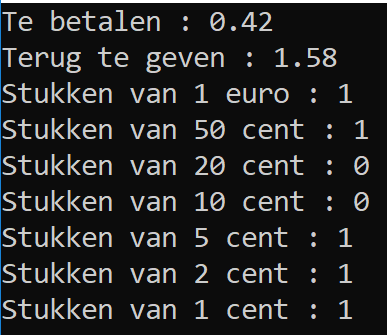
Gegeven : een te betalen bedrag voor een snoep uit een snoepautomaat.

De kostprijs van de verschillende snoepen varieert tussen € 0,30 en € 1,20.

De klant kan enkel betalen met een stuk van € 2. Het programma moet het wisselgeld uitrekenen : hoeveel stukken van € 1, € 0,50, € 0,20, € 0,10, € 0,05, € 0,02 en € 0,01 moeten er teruggegeven worden. Steeds met zo weinig mogelijk munten!

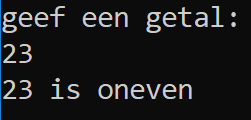
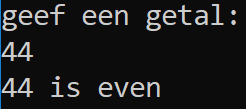
Test het programma voor een aankoop die € 0,42 kost, voor een aankoop die € 1,02 kost, …

Opgelet: bij bewerkingen met decimale getallen (float, double) geven bewerkingen niet altijd een correct resultaat: 10.0 / 2.0 = 4.9999993 (of zo iets). Gebruik dus de gepaste datatypes.



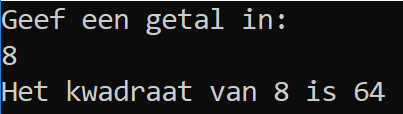
## Opdracht 2.16 – Even en oneven **\* (Z)**

Schrijf een programma die een integer inleest en vervolgens bepaald of dit getal een even of een oneven getal betreft.

## Opdracht 2.17 – Kwadraat **\* (Z)**

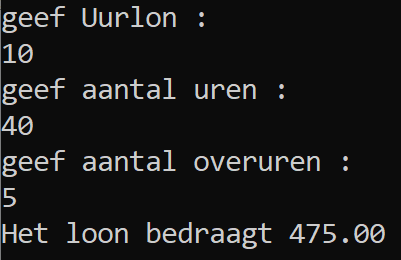
Vraag de gebruiker om een getal. Lees dat getal in en bereken het kwadraat. Toon het kwadraat.



## Opdracht 2.18 – Uren, overuren, brutoloon **\* (T)**

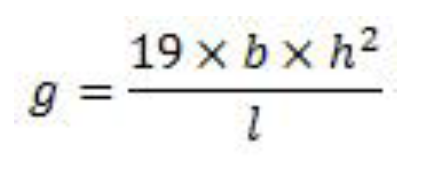
Lees het aantal uren, het aantal overuren en het bruto uurloon in.

Bepaal het effectieve brutoloon, als je weet dat overuren aan 150% worden uitbetaald. (Bijv.: 38 uren, 6 overuren, € 25,00 uurloon geeft een brutoloon van € 1 175,00)



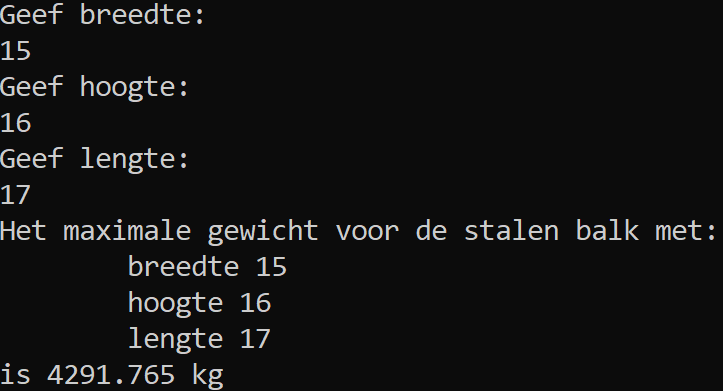
## Opdracht 2.19 – Bereken het gewicht voor een constructie **\* (T)**

Een stalen balk met hoogte h en breedte b, die voor een lengte l uit een (stevige) constructie steekt, kan aan het uiteinde maximaal een gewicht g dragen. Dat gewicht wordt berekend als :



Hierbij worden de lengtematen b, h en l in cm uitgedrukt en het gewicht g in kg.

Lees de drie nodige maten in en bereken het maximale gewicht. Druk het resultaat af.



## Opdracht 2.20 – Remafstand **\* (T)**

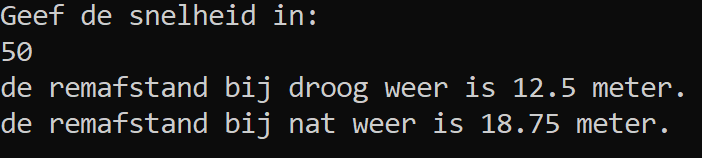
De gebruiker kan de snelheid in km/uur ingeven waarop hij rijdt met de auto.

Aan de hand van deze waarde zal het programma de remafstand berekenen.

De remafstand wordt berekend als volgt:

* Bij droog weer is dit het kwadraat van de snelheid/10 en dit gedeeld door 2 : (S / 10)² / 2
* Bij nat weer is dit het kwadraat van de snelheid/10 en dit gedeeld door 2 vermenigvuldigt met 1.5 : (S / 10)² / 2 \* 1.5

Gebruik hierbij de de Pow method uit de Math class.



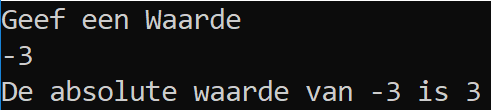
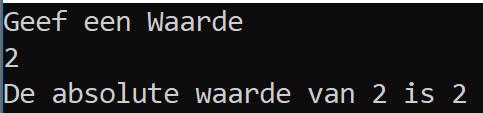
# Building Block III - Selectie

## Opdracht 3.1 – Absolute waarde **\* (Z)**

Bepaal de absolute waarde van een getal. (De absolute waarde is de waarde van een getal zonder teken). De gebruiker kan zowel een positief als een negatief getal ingeven. Het resultaat is altijd een positief getal.

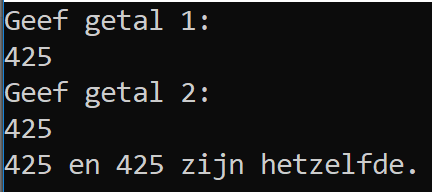
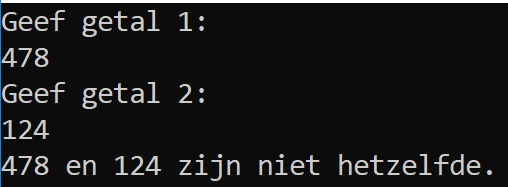
Maak hierbij geen gebruik van de speciale functie abs() die binnen C# aanwezig is.

Gebruik een if-else structuur.

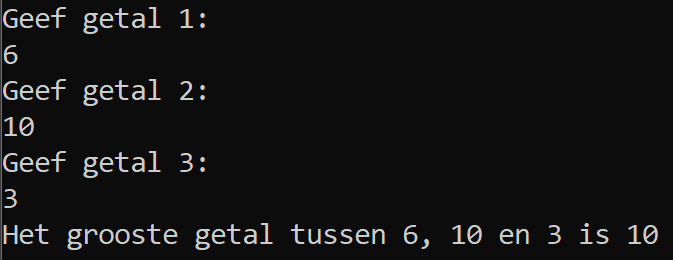
## Opdracht 3.2 – hetzelfde getal **\* (Z)**

De gebruiker geeft 2 getallen in. Ga na of de 2 getallen die ingegeven zijn, hetzelfde getal zijn.

## Opdracht 3.3 zoek het grootste getal **\* (Z)**

De gebruiker heeft de mogelijkheid om 3 getallen in te geven. Wij willen als resultaat het grootste getal van de 3 als uitkomst hebben. We maken gebruik van **int**getallen.

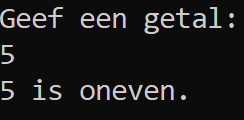
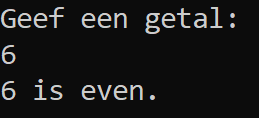


## Opdracht 3.4 even of oneven **\* (Z)**

We willen van een getal dat we ingeven weten of dit getal even of oneven is. We maken gebruik van het type **int** als ingegeven getal.

Hoe ga je dit programmeren? Aan welke voorwaarde wordt voldaan om te weten of een getal even of oneven is?

Als resultaat wordt er weergegeven: “Het getal is even” of “Het getal in oneven”, waarbij we het woord getal vervangen door het ingegeven getal.

## Opdracht 3.5 Cinema **\*\* (Z)**

Een toegangsticket voor de cinema kost 3€.

Kinderen beneden de 11 jaar moeten maar 1€ betalen.

Naargelang de leeftijd van de bezoeker kan er een korting worden verkregen.

Als men tussen de 11 en 15 jaar is bedraagt de korting 15%

Als men tussen de 15 en 18 jaar is bedraagt de korting 10%

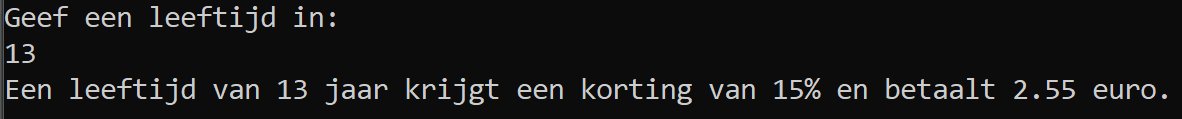
Als men tussen de 18 en 21 jaar is bedraagt de korting 5%

Als men ouder is dan 21 jaar krijgt met geen korting

Schrijf een programma dat toelaat dat aan de gebruiker zijn leeftijd vraagt en op basis van deze leeftijd zal de kostprijs worden berekend.

Als uitvoer krijgen we het volgende:

Je bent 12 jaar je krijgt 15% korting dus je betaalt nog 2.55€



## Opdracht 3.6 Vrachtwagen **\*\* (K)**

Een vrachtwagen heeft een lengte van 3 meter lang, 1,5 meter breed en 2 meter hoog. Bereken de inhoud van deze vrachtwagen.(breedte \* hoogte \* lengte).

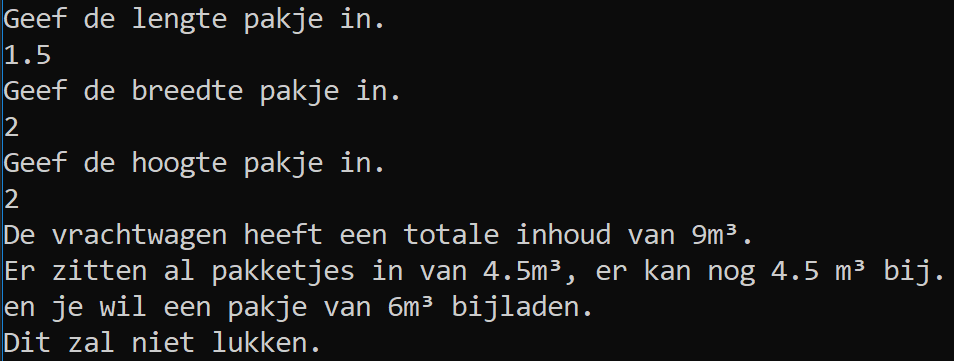
De vrachtwagen is reeds voor 4,5 m3 geladen met rechthoekige verpakkingen.

De gebruiker wil nog een verpakking bijgooien.

Het programma vraagt aan de gebruiker de breedte, hoogte en lengte van het pakje dat hij er nog wil bijgooien.

Het programma zal berekenen of dat pakje nog op de vrachtwagen kan worden geladen.

Als uitvoer krijg je: de vrachtwagen is reeds voor….m3 geladen en je wil een pakje van …m3 bijladen. Dat zal (niet of nog) lukken.



## Opdracht 3.7 – Portkosten **\*\* (Z)**

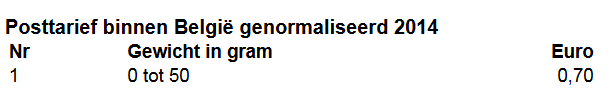
Wanneer we een Brief gaan versturen via de Post moeten daar portkosten op betaald worden. Alles hangt af van de volgende factoren:

Is je brief al dan niet genormaliseerd.

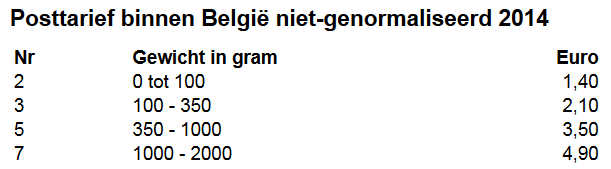
Wordt je brief verstuurd naar het buitenland of binnenland.

Volgende business regels gelden bij post:

Voor de genormaliseerde post in het Binnenland



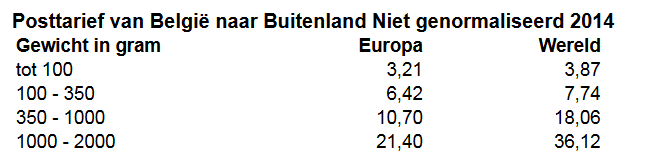
Voor de niet genormaliseerde post in het Binnenland



Voor de genormaliseerde post van Belgie naar Buitenland



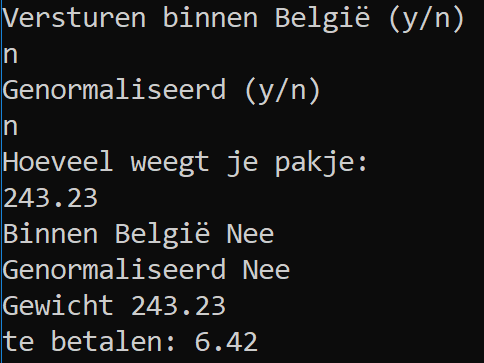
Voor de niet-genormaliseerde post van België naar Buitenland



We vragen aan de gebruiker of hij een genormaliseerde brief zal versturen.

Verder vragen we aan de gebruiker of het een brief is dat hij wenst te versturen naar iemand in het Binnenland of naar het Buitenland. Tenslotte vragen we het gewicht van de brief (in gram).

Op basis van deze parameters zal de prijs worden berekend (rekening houdend met de business regels van de post) die een gebruiker moet betalen voor het versturen van de post.



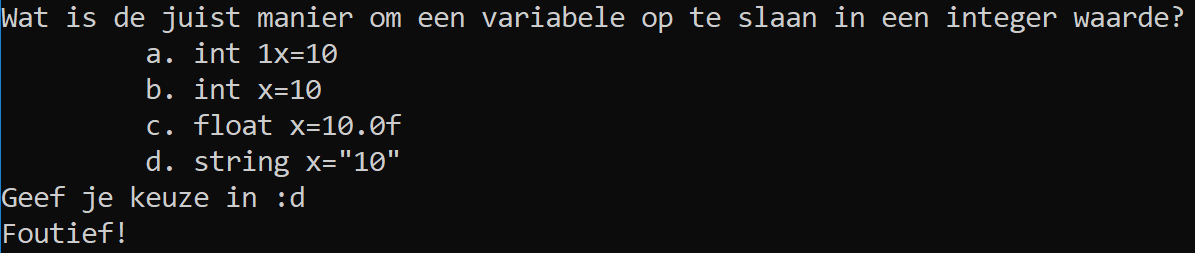
## Opdracht 3.8 Multiple choice **\* (Z)**

Schrijf een programma dat de vraag stelt wat de correcte manier is om een variabele op te slaan in een integer variabele. Mogelijke antwoorden zijn (deze worden in de vraagstelling opgenomen. Zorg hierbij dat deze onder elkaar geplaatst worden, alsook dat deze inspringen):

* int 1x = 10
* int w = 10
* float x = 10.0f
* string x = “10”

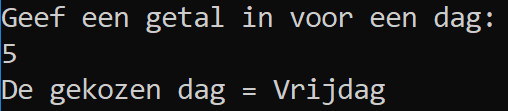
Indien je als gebruiker een foutief antwoord kiest, krijg je dit ook te weergegeven. Indien correct wordt dit ook weergegeven. Indien er niets ingevuld wordt, heeft deze een andere output die zegt dat er niets ingegeven is.

Los op aan de hand van een switch.



## Opdracht 3.9 Dag van de week **\* (Z)**

Schrijf een programma waarbij je een getal (1 tem 7) ingeeft die een dag van de week voorstelt. Via een switch statement, wordt het getal omgezet naar de naam van de dag. Voorzie ook een output indien er een getal groter dan 7 wordt ingegeven. Zorg hierbij dat er maar één Console.WriteLine() geplaatst wordt voor de output van de gekozen dag. (We gaan ervan uit dat de week start op maandag)



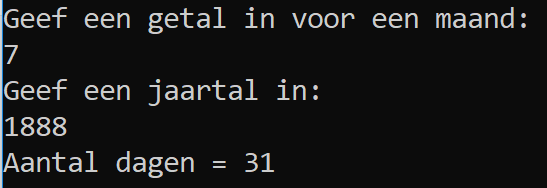
## Opdracht 3.10 Aantal dagen per maand **\*(\*) (Z)**

Schrijf een programma die weergeeft hoeveel dagen er zijn bij de ingegeven maand. Januari, Maart, Mei, Juli, Augustus, Oktober en December hebben 31 dagen. April, Juni, September, November hebben 30 dagen. Februari heeft 28 of 29 dagen, afhankelijk of het een schrikkeljaar is.

Een schrikkeljaar is elk jaar dat deelbaar is door 4 met uitzondering van de eeuwjaar, dan is het pas een schrikkeljaar indien ze deelbaar zijn door 400.

Zo is 2000 een schrikkeljaar, maar 1900 was geen schrikkeljaar.

Los op door gebruik te maken van een switch.



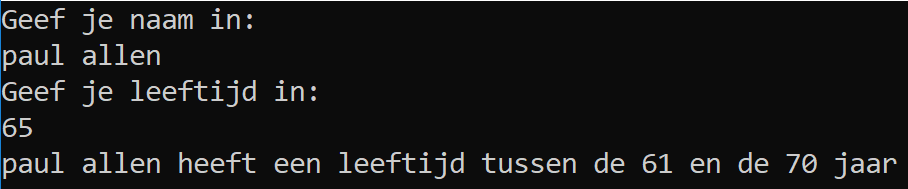
## Opdracht 3.11 Leeftijd **\* (Z)**

De gebruiker wordt gevraagd naar zijn naam en leeftijd. Deze kan hij ook ingeven. Naargelang de leeftijd die wordt ingegeven kan de output verschillen.

* Naam heeft een leeftijd tussen 0 en 10 jaar
* Naam heeft een leeftijd tussen 11 en 20 jaar
* Naam heeft een leeftijd tussen 21 en 30 jaar
* …

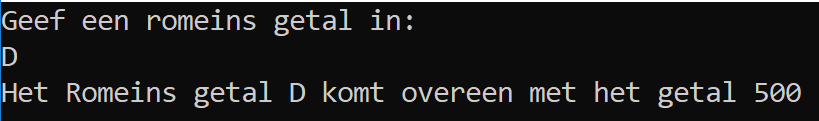
Los op tot een leeftijd van 100 jaar.

Los op door gebruik te maken van een switch.



## Opdracht 3.12 Romeins getal **\* (Z)**

De gebruiker wordt gevraagd om een Romeins getal in te geven. Dit Romeins getal is van het type char. Via een switch constructie wordt dit Romeins getal omgezet naar een decimaal getal. Doe hierbij de opzoeking van de romeinse getallen. Zorg hierbij dat de output Console.WriteLine maar één keer geschreven wordt in het programma



## Opdracht 3.13 Cirkel, rechthoek, vierkant **\*\*\* (T)**

Stanley wil een programma maken waarmee hij de omtrek en de oppervlakte van een cirkel, rechthoek en vierkant kan bepalen.

Hij wil zelf kunnen opgeven wat hij wilt berekenen. Oppervlakte, omtrek? Alsook van welke vorm: cirkel, rechthoek, vierkant.

De gegevens lengte, breedte, hoogte, diameter kunnen ook iedere keer verschillend zijn.

Zorg voor onderstaande output:

Je hebt gekozen om de oppervlakte van een rechthoek te bepalen.

Je hebt hierbij gekozen voor volgende gegevens:

Lengte: x m

Breedte: x m

Hoogte: x m

De oppervlakte = x m2

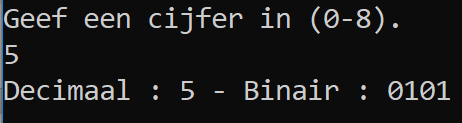


## Opdracht 3.14 decimaal naar binair **\* (K)**

We willen een programma maken, waarbij we de decimale waarde (0->8) omzetten naar binaire waarden.

Je kan een decimaal getal ingeven, het binair getal wordt als output weergegeven.

Gebruik een switch.

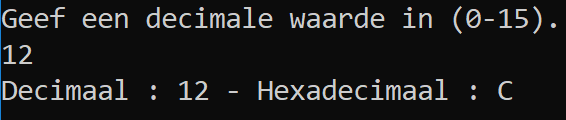


## Opdracht 3.15 decimaal naar hexadecimaal **\* (Z)**

We willen een programma maken, waarbij we de decimale waarde (unsigned short) (0->15) omzetten naar hexadecimale waarden.

Je kan een decimaal getal ingeven, het hexadecimaal getal wordt als output weergegeven.

Gebruik een switch.



## Opdracht 3.16 Omzetting talstelsels \*\*\* (T)

Je kan in het programma kiezen of je een omzetting van binair/decimaal/hex naar hex/decimaal/binair kan maken. Deze keuze wordt gemaakt door de user die dit ingeeft (menu met switch).

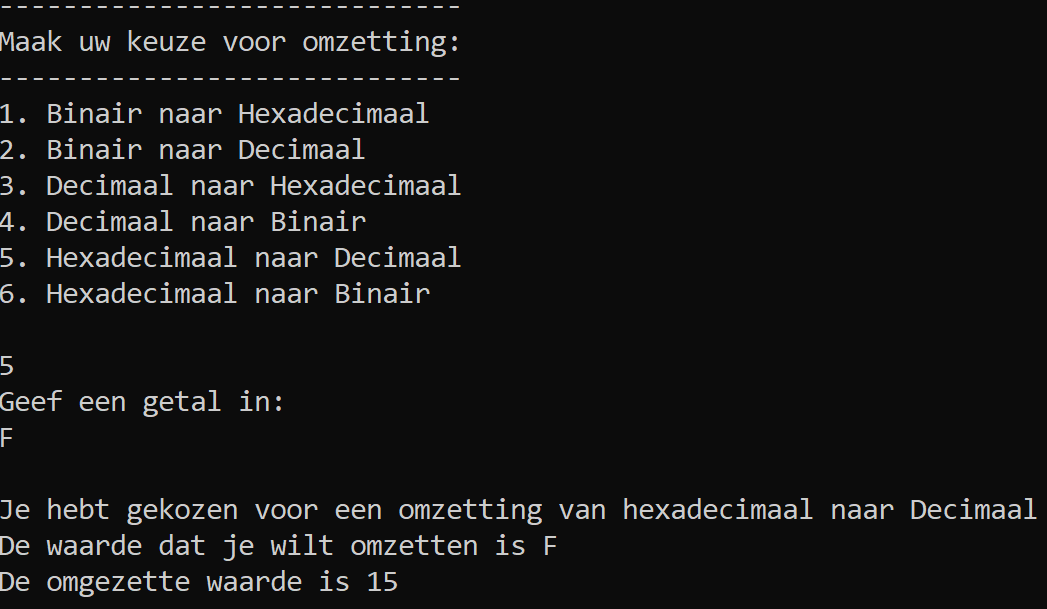
Als output krijgen we:

Je hebt gekozen voor een omzetting van … naar …

De waarde dat je wilt omzetten is …

De omgezette waarde is …

Maak gebruik van switches, je kan ook switches nesten in elkaar.

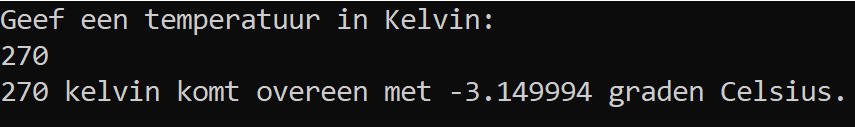


## Opdracht 3.17 Temperatuur **\* (Z)**

Tussen een temperatuur gemeten in Kelvin en een temperatuur gemeten in graden Celsius bestaat het volgende verband:

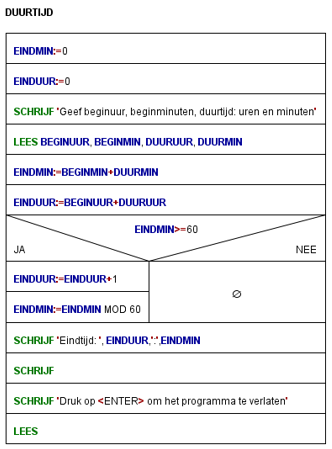
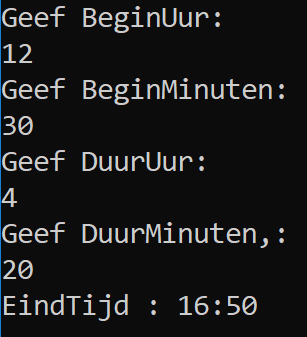
C=K-273,15

Hierbij mag K niet negatief zijn (Geef hierbij een gepaste boodschap). Ontwerp een programma dat een temperatuur in Kelvin inleest en de temperatuur in graden celsius afdrukt.



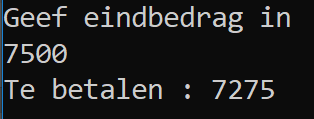
## Opdracht 3.18 Duurtijd **\*\* (Z)**

Lees een begintijd in (uren en minuten afzonderlijk). Lees een duurtijd in (ook in uren en minuten). Bereken dan de eindtijd, in uren en minuten.

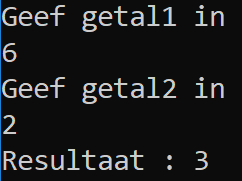
## Opdracht 3.19 Factuur **\* (Z)**

Lees een eindbedrag (bijv. van een factuur) in. Indien het bedrag groter is dan € 5 000,00, dan wordt een korting van 3% toegestaan. Toon het uiteindelijk te betalen bedrag.



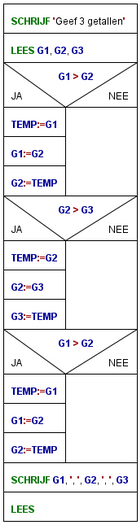
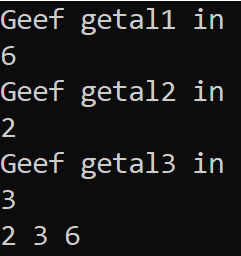
## Opdracht 3.20 Deling **\*\* (Z)**

Lees twee getallen in. Deel het grootste door het kleinste en toon het resultaat op het scherm.



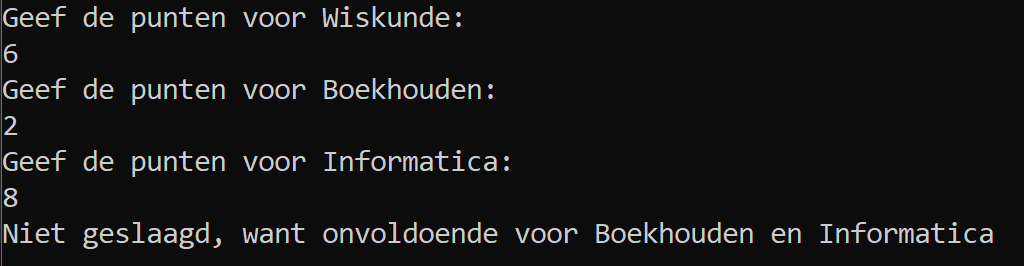
## Opdracht 3.21 Eenvoudige sortering **\*\* (K)**

Lees drie getallen in. Rangschik ze van groot naar klein en druk het resultaat af.

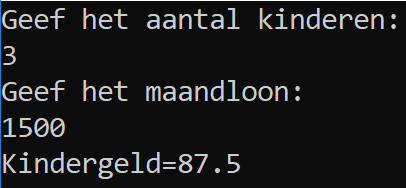
## Opdracht 3.22 Examens **\* (Z)**

Lees de examenuitslagen in voor drie vakken (wiskunde, boekhouden en informatica). Elk van de vakken staat op 10 punten. De student is geslaagd indien hij/zij voor wiskunde minstens 6/10 haalt, en voor boekhouden en informatica samen minstens 12/20. Toon op het scherm of de student geslaagd is, en indien de student niet geslaagd is, toon je ook de reden daarvoor.



## Opdracht 3.23 Kindergeld **\*\* (Z)**

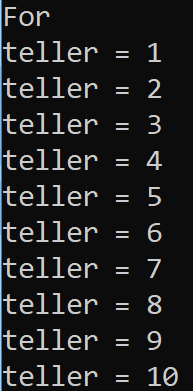
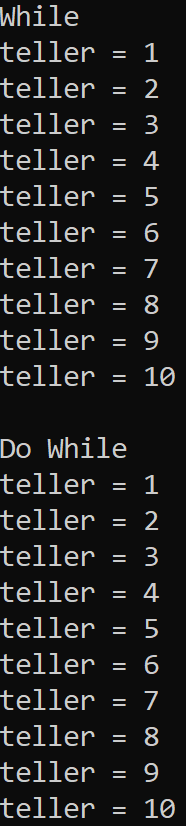
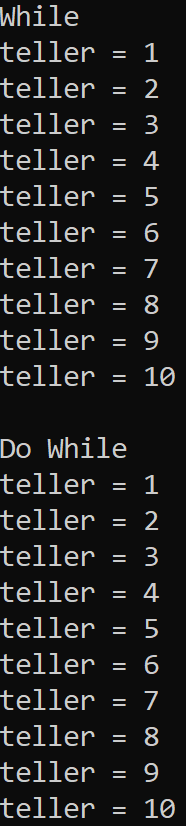
Een moeder heeft recht op € 25,00 kindergeld per kind. Voor het derde kind (en elk volgend kind) krijgt ze een toeslag van € 12,50. Voor het vijfde (en elk volgend) kind krijgt ze nog eens een toeslag van € 7,50. Als het maandloon van de moeder kleiner is dan of gelijk aan € 500,00, dan krijgt ze 25% toeslag op het kindergeld. Maar als haar maandloon groter is dan € 2000,00, dan krijgt ze 25% minder kindergeld. Minimaal heeft een moeder altijd recht op € 25,00 per kind. Lees het aantal kinderen en maandloon in, en toon het kindergeld waar de moeder recht op heeft. (Bijv. : 6 kinderen en € 1 500,00 maandloon geeft € 215,00 kindergeld)



# Building Block IV - Iteratie

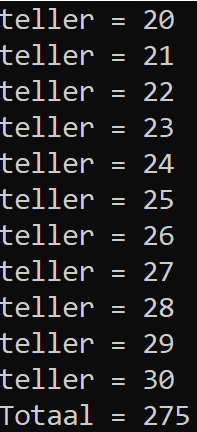
## Opdracht 4.1 - Teller

Maak een teller van 1 tem 10. Maak de oefening 3 keer (mag in hetzelfde opdracht) zowel met een **for**-loop, een **while**-loop en een **do-while** loop.

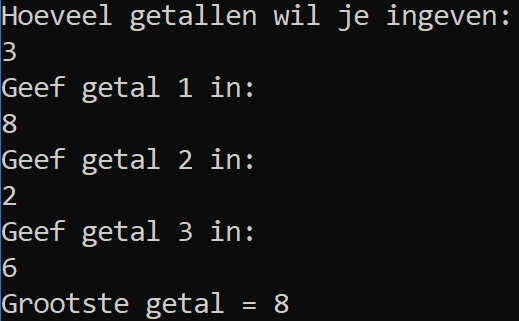
## Opdracht 4.2 – Teller + Som

Maak een teller van 20 tem 30. Geef hierbij de getallen terug, alsook de som van de getallen. De som van de getallen wordt één keer getoond, nl op het einde.



## Opdracht 4.3 – Grootste getal

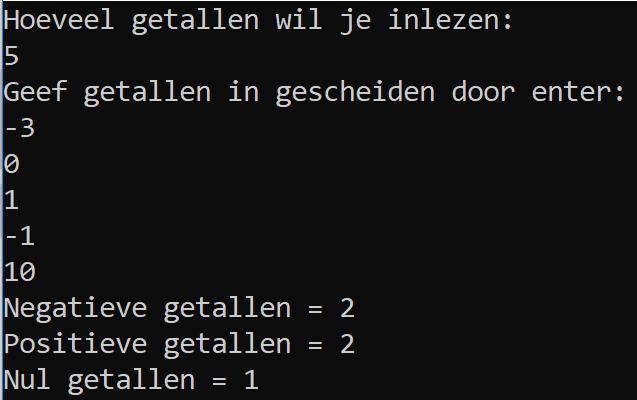
De gebruiker geeft het aantal getallen in (aantal > 1) die hij wenst in te geven. Voer daarna een reeks getallen in. Bepaal het grootste getal in de rij. Gebruik geen array!



## Opdracht 4.4 – Aantal positieve, negatieve en 0-getallen

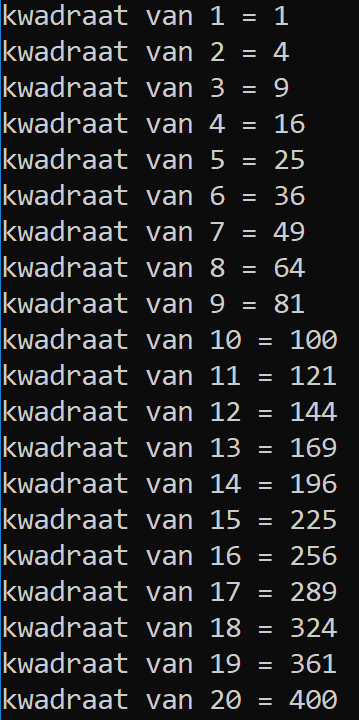
De gebruiker geeft het aantal getallen in (aantal > 1) dat hij zal ingeven. De ingegeven getallen kunnen zowel positief als negatief zijn.

De output zal het aantal negatieve getallen, positieve getallen en nulgetallen weergeven.



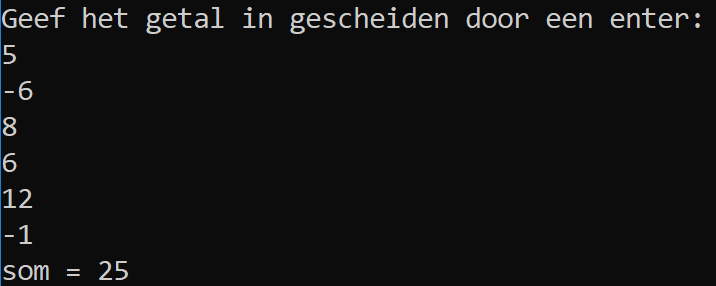
## Opdracht 4.5 - Kwadraat

Schrijf een programma dat het kwadraat van de getallen van 1 tot 20 afdruk door middel van **while**-lus.



## Opdracht 4.6 – Som van oneindig aantal getallen

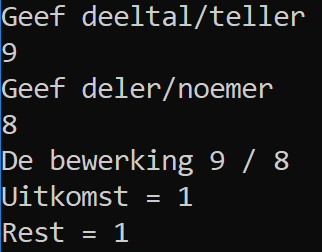
We moeten een oneindig aantal getallen kunnen ingeven. Van het aantal ingegeven getallen wordt de som berekend en weergegeven. De getallen die ingegeven worden zijn van het type **int**. De getallenreeks eindigt met de ingave van het getal “-1”. Dit getal wordt niet bij de som geteld. Indien we als eerst getal -1 ingeven, dan is de som = 0.



## Opdracht 4.7 – Delen

We delen 2 getallen, deeltal en deler, elk van het type **unsigned int (uint)**, door elkaar zonder gebruik te maken van de deeloperator (/). Hierbij nemen we wel aan dat deler > 0.

Als resultaat worden 2 zaken weergegeven, nl. het quotient en de rest van de deling.



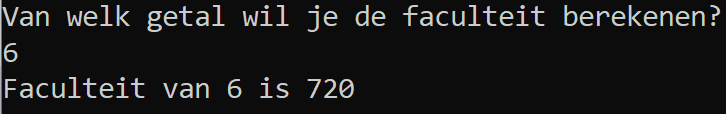
## Opdracht 4.8 – Faculteit

Bereken de faculteit van een getal.

Bv. Faculteit van 5 = 5! = 5 \* 4 \* 3 \* 2 \* 1 = 120.

Opgelet: de faculteit van 0 (0!) is 1, en de faculteit van een negatief getal bestaat niet.

De faculteit is een long.



## Opdracht 4.9 – Halve kerstboom

Schrijf een programma met een geneste for loop die volgend patroon van sterretjes op het scherm plaatst.

\*

\*\*

\*\*\*

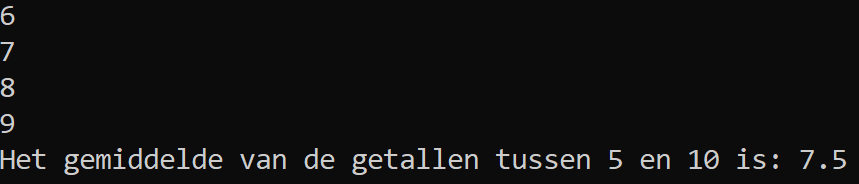
\*\*\*\*

\*\*\*\*\*

## Opdracht 4.10 - Gemiddelde

We vragen aan de gebruiker een startwaarde en eindwaarde op. (integer getallen). Het programma berekent het gemiddelde van de getallen die tussen deze startwaarde en eindwaarde liggen.

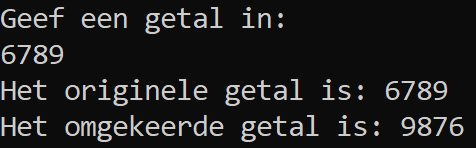
De uitvoer is als volgt wanneer de gebruiker 5 als startwaarde en 10 als eindwaarde heeft opgegeven.



## Opdracht 4.11 – Omgekeerd Getal

We vragen aan de gebruiker om een geheel getal in te geven. Het programma zal het omgekeerde getal berekenen en dat op het scherm plaatsen.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **getal** | **kopieGetal>0** | **copyGetal** | **digit** | **copyGetal** | **omgekeerdGetal** |
| **1234** |  | 1234 |  | 1234 |  |
| **start while** |  |  |  |  |  |
| 1234 | true | 1234 | 1234 % 10=4 | 1234 / 10=123 | 0 \* 10 + 4 = 4 |
| 1234 | true | 123 | 123 % 10=3 | 123 / 10=12 | 4 \* 10 + 3 = 43 |
| 1234 | true | 12 | 12 % 10=2 | 12 / 10=1 | 43 \* 10 + 2 = 432 |
| 1234 | true | 1 | 1 % 10=1 | 1 / 10=**0** | 432 \* 10 + 1 = 4321 |
| 1234 | **false=STOP while** |  |  |  | 4321 |
| 1234 |  | 0 |  |  | **4321** |



## Opdracht 4.12 – Berekening intrest

Charlie Chaplin investeert 1000 EUR. In de gouden jaren kreeg hij nog 5% interest per jaar van de bank. We gaan ervan uit dat Charlie het geld 10 jaar onaangeroerd op de bank laat staan. Geef hierbij als output in welke jaar Charlie zit, en voor welk bedrag Charlie al op de bank heeft staan.

Gebruik onderstaande formule voor het berekenen van de intrest:

a = p (1+r)n

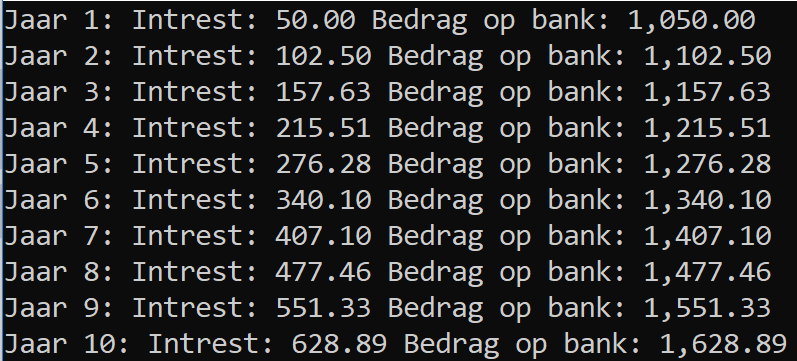
a: bedrag op de bank op het einde van jaar n

p: origineel geïnvesteerd bedrag

r: jaarlijkse intrest

n: aantal jaar

tip: bekijk hierbij eens de Math.Pow bewerking binnen .NET.

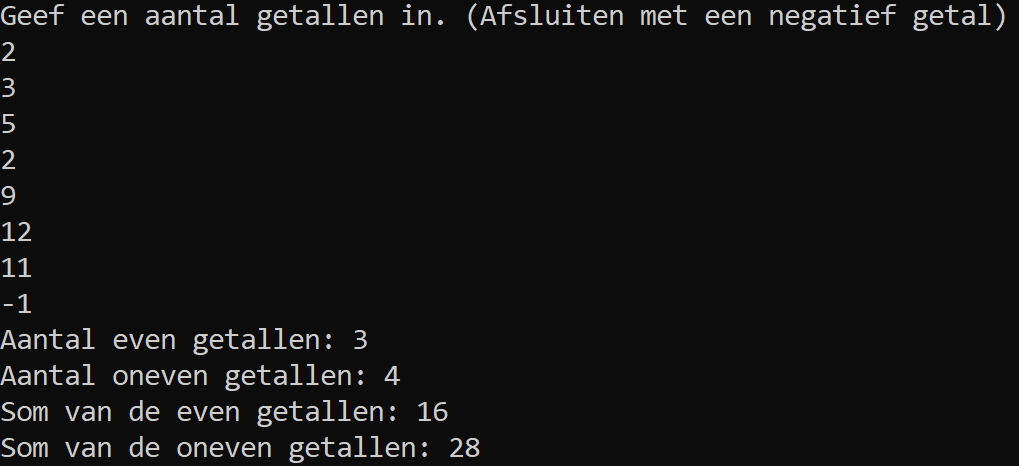


## Opdracht 4.13 – berekening resultaat even getallen

Andrei T. kan een oneindig aantal positieve getallen ingeven. De invoer stopt met het ingeven van een negatief getal.

Van de getallen die ingegeven worden, wordt weergegeven hoeveel even getallen er ingegeven werden alsook hoeveel oneven getallen. De som van de even en oneven getallen wordt eveneens weergegeven.

Gebruik hierbij een **do-while** lus.



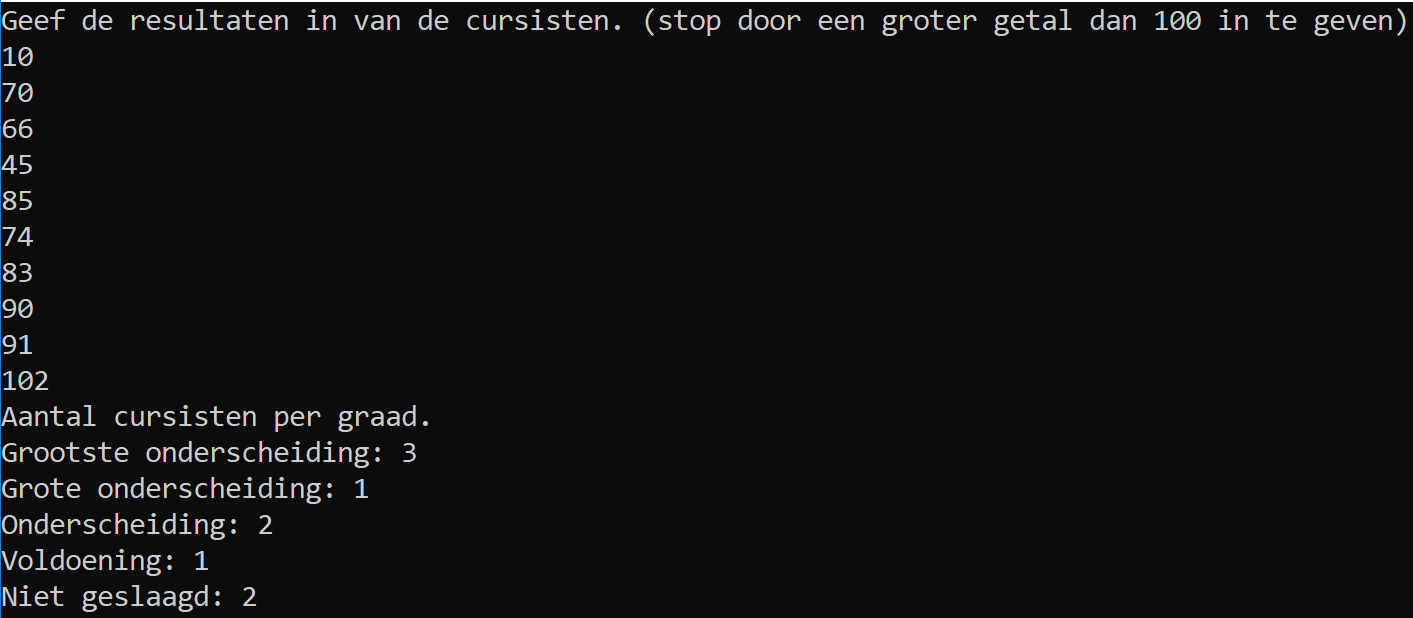
## Opdracht 4.14 – Omzetting cijfer resultaten naar codes

Een school wil een programma waarmee ze direct zien na het invoeren van de resultaten hoeveel cursisten er grootste onderscheiding, grote onderscheiding, onderscheiding, voldoening en niet geslaagd zijn.

Als input worden de resultaten ingegeven van de cursisten. De invoer wordt gestopt door een getal groter dan 100 in te geven.

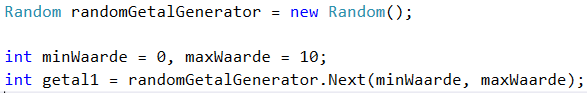
Als output wordt het aantal resultaten die ingegeven zijn getoond, alsook het totaal van die resultaten. Ook het gemiddelde wordt getoond. En hoeveel cursisten er iedere graad hebben behaald.

* Grootste onderscheiding >=85%
* Grote onderscheiding >= 77%
* Onderscheiding >= 68%
* Voldoening >= 50%
* Niet geslaagd < 50%

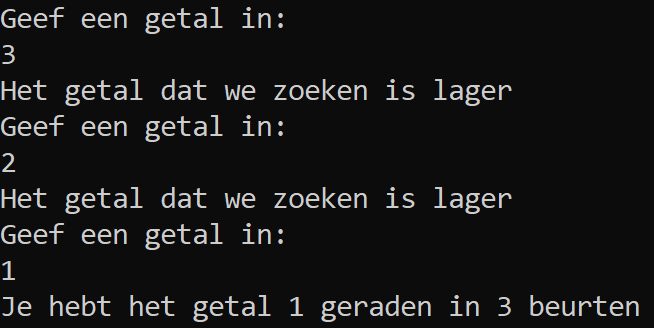


## Opdracht 4.15 - Hoger-Lager

1. We genereren een willekeurig getal via de random functie die in de class Math aanwezig is. Hiervoor kan je de volgende code gebruiken:



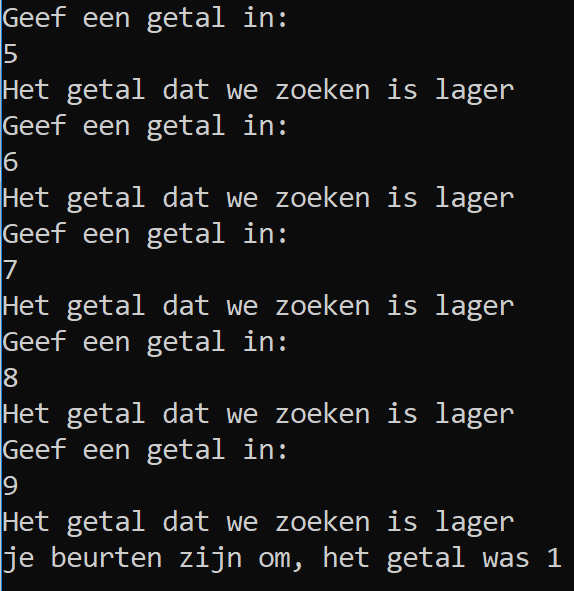
1. De gebruiker kan een getal ingeven.
2. Het systeem bekijkt of het getal dat de gebruiker heeft ingegeven gelijk is aan datgene dat het systeem heeft uitgerekend.
3. Het systeem geef een evaluatie in de vorm van “je hebt een hogere waarde ingegeven”, “je hebt een kleinere waarde ingegeven” , “je hebt de juiste waarde ingegeven”
4. Zolang de gebruiker niet het juiste getal heeft geraden kan hij opnieuw een getal ingeven.
5. Het systeem houdt het aantal pogingen bij.
6. Wanneer de gebruiker het juiste getal heeft geraden krijgt hij het getal dat door het systeem werd gegenereerd te zien én het aantal pogingen dat er werd ondernomen om dat getal te raden.



## 

## Opdracht 4.16 – Hoger-Lager + uitbreiding

Maak een uitbreiding op de vorige oefening door het raden naar het juiste getal te laten stoppen wanneer de gebruiker het getal heeft gevonden of dat het aantal beurten groter is dan 5. Wanneer het programma stopt zal de gebruiker een melding krijgen waarom het is gestopt. (geraden of maximaal aantal beurten is verstreken).



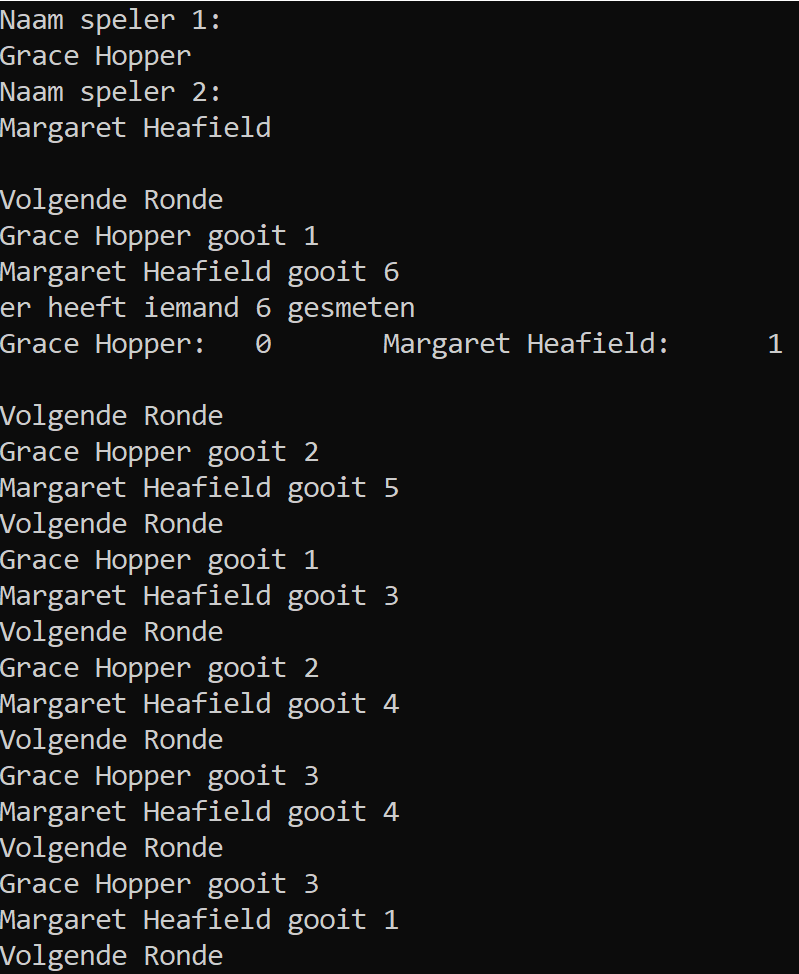
## Opdracht 4.17 - Dobbelsteen

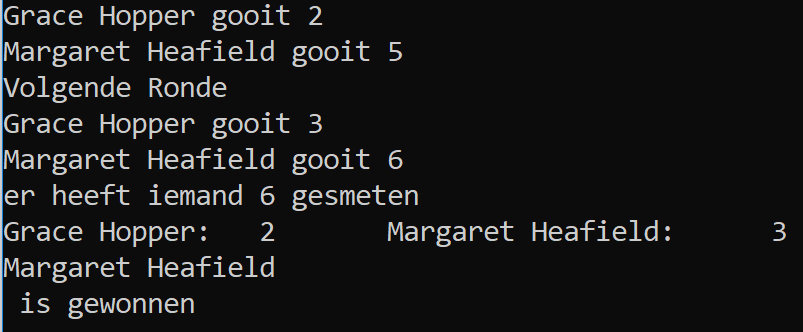
We schrijven een programma dat twee spelers laat spelen met een dobbelsteen. Het programma moet de volgende zaken kunnen doen:

1. De namen van twee gebruikers opvragen en bewaren
2. De gebruiker gooit de eerste dobbelsteen. Dus dat getal zal willekeurig door het programma worden berekend. (Zie oefening hoger lager voor voorbeeld om random getallen te laten genereren)
3. De waarde van de dobbelsteen die de eerste gebruiker heeft gegooid wordt bewaard in een variabele
4. De tweede gebruiker gooit de dobbelsteen. (maak hiervan weer gebruik van de methode van de willekeurige getallen). Dat getal zal weer door het systeem worden berekend.
5. De beide spelers kunnen blijven gooien tot er één van de twee of beide zes heeft gegooid. Wie zes heeft gegooid krijgt één punt.
6. Het spel is afgelopen als er één van de spelers 3 punten heeft behaald. (dus 3 maal de waarde 6 heeft gegooid).

In de uitvoer moet er telkens worden weergegeven wat de speler heeft gegooid. Indien hij een 6 heeft gegooid moet daar een melding van worden gegeven én de tussenscore moet worden getoond.

Bij afloop plaats je op het scherm wie heeft gewonnen.

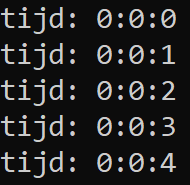
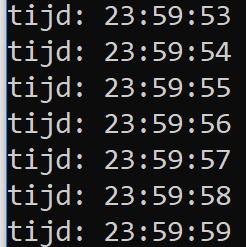
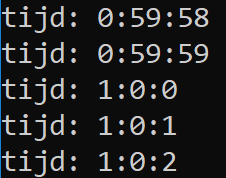
 



## Opdracht 4.18 – Uurwerk

Schrijf een programma dat een uurwerk simuleert. We starten de tijd met 00:00:00. Dit is in de vorm hh:mm:ss.

We gebruiken hiervoor drie for loops die genest zijn. Wanneer het een volledige 24 uur heeft gelopen hoef je niet meer te herhalen. Het programma zal dan stoppen.

## Opdracht 4.19 – Shopping

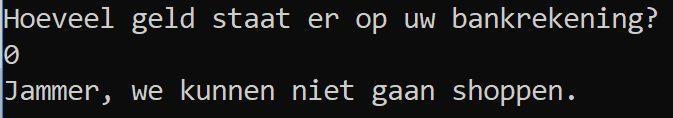
We vragen aan de klant hoeveel geld er op zijn bankrekening staat.

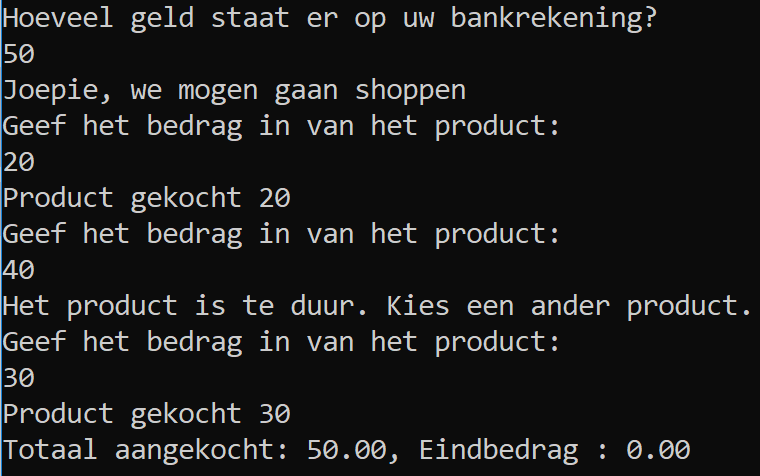
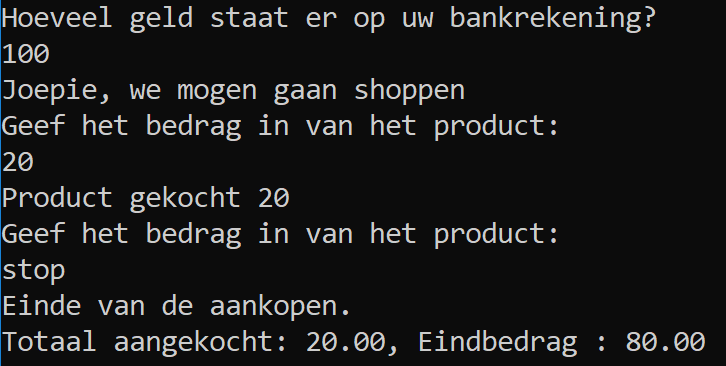
Als er geld opstaat kunnen we gaan shoppen.

De klant gaat de winkel binnen en legt een product in zijn winkelmandje. De verkoopster kan in het programma aangeven hoeveel het product kost. Zolang er voldoende geld op de bankrekening staat om het product te komen kan de klant het product kopen. Indien er geen genoeg geld meer beschikbaar is krijgt de gebruiker daarvan een melding.

Geef “Stop” in om te stoppen.

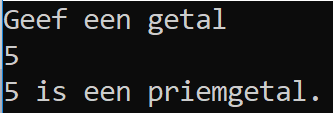
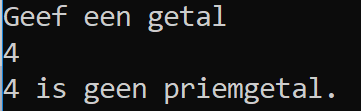
Op het einde krijgt de klant de totale prijs te zien die hij moet betalen én wat hij nog op zijn rekening over heeft.



## Opdracht 4.20 – Priemgetallen

Maak een programma die als input een getal vraag groter dan 0, en als output heeft als het een priemgetal is of niet.

## Opdracht 4.21 – Rij van Fibonacci

Ontwerp een consoletoepassing waarmee je de eerste 10 getallen van de rij van Fibonacci toont.

1 1 2 3 5 8 13 21 ..

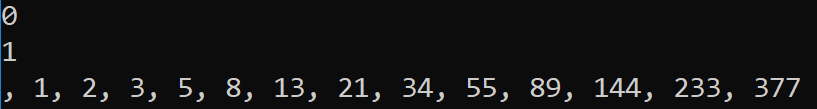
X1 = 1

X2 = 2

Xn = Xn-2 + Un-1 voor n > 2

De “rij van Fibonacci” is een wiskundige rij, waarbij elk element de som van de twee voorgaande is. De eerste twee elementen zijn 0 en 1. Het begin van de rij is dus: 0, 1, 1, 2, 3, 5, 8, 13, 21, …

Maak een programma dat 15 elementen van deze rij afdrukt.



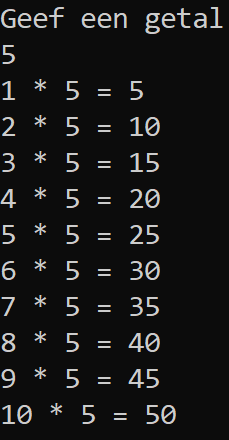
## Opdracht 4.22 – Ratten

In New York leven naar schatting 2 miljoen ratten. De populatie ratten groeit ieder jaar met 15 %. Ontwerp een consoletoepassing waarmee je kan berekenen hoelang het duurt vooraleer New York 10 miljoen ratten telt.



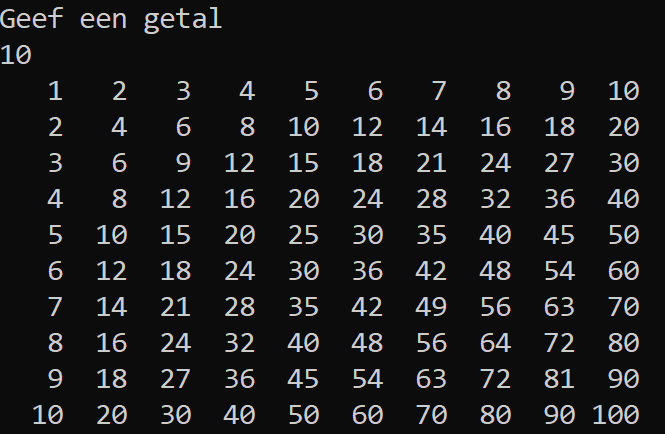
## Opdracht 4.23 – Tafels van vermenigvuldiging

Ontwerp een consoletoepassing waarmee de gebruiker de tafels van vermenigvuldiging verkrijgt van een ingegeven getal tussen 1 en 10.



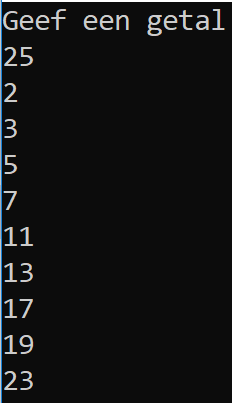
## Opdracht 4.24 – Vermenigvuldingsvierkant

Schrijf een programma dat een vermenigvuldigingsvierkant afdrukt. De grootte van het vierkant is afhankelijk van een in te lezen getal.



## Opdracht 4.25 – Priemgetal

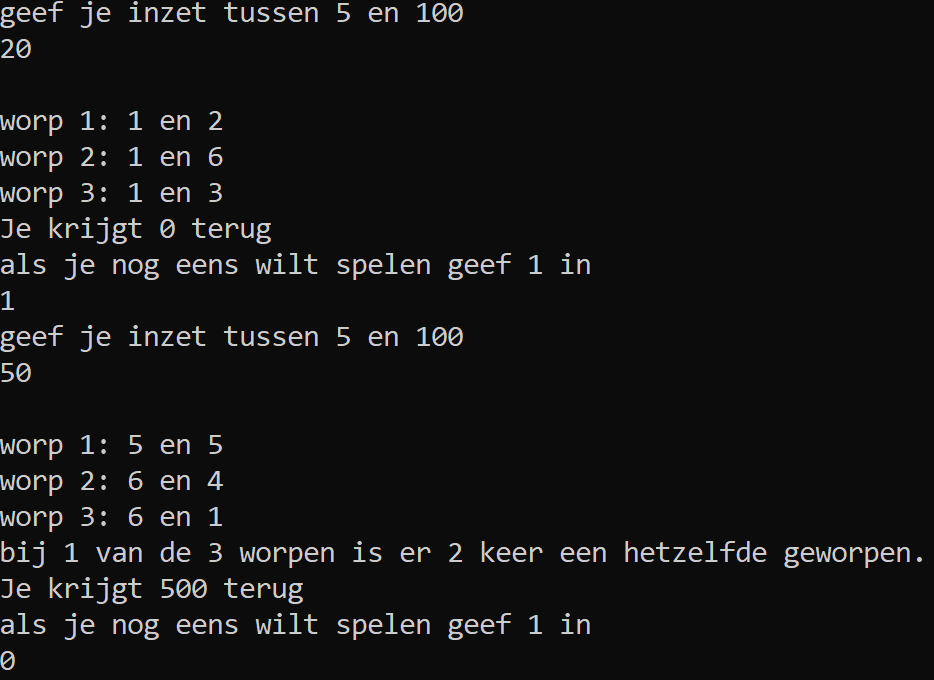
Ontwerp een consoletoepassing waarin de gebruiker een geheel getal kan invoeren met een maximum van 100. De console toont de getallen van 1 tot en met het ingevoerde.



## Opdracht 4.26 – Dobbelstenen

Ontwerp een consoletoepassing waarmee je 3 worpen met 2 dobbelstenen nabootst. Het resultaat in de console toont wat de deelnemer wint. De deelnemer kan zelf kiezen hoeveel hij inzet. De minimum inzet is 5 EUR en de maximum inzet is 100 EUR. Het spel kan herhaald worden na de laatste worp.

* bij 1 van de 3 worpen is er 2 keer een 6 geworpen: winst = inzet x 50
* bij 1 van de 3 worpen is er 2 keer hetzelfde geworpen: winst = inzet x 10
* bij de 3 worpen is er 2 keer een zes geworpen: winst = inzet x 2



## Opdracht 4.27 – Temperatuur

Het KMI ontvangt iedere dag temperatuurwaarnemingen uit het hele land. Deze gegevens worden per computer verwerkt. Er zijn niet elke dag evenveel waarnemingen.

Lees een willekeurig aantal temperaturen in (wanneer 99 wordt ingetikt, stopt de ingave). Druk hierna volgende resultaten af:

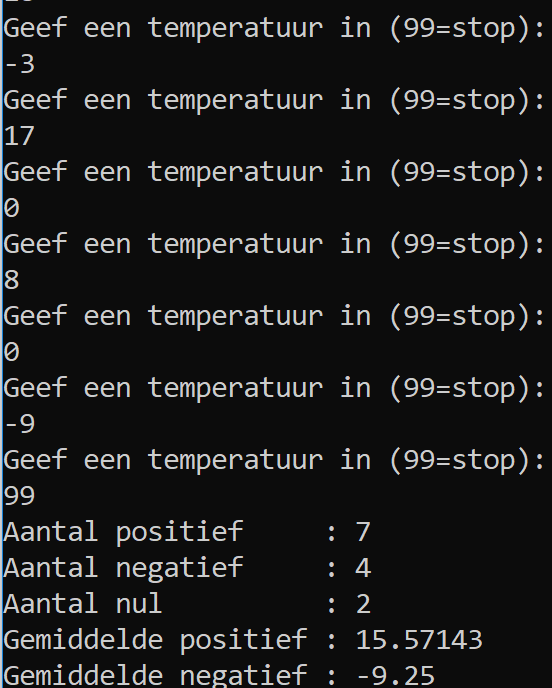
aantal positieve temperaturen,

aantal negatieve temperaturen,

gemiddelde van de positieve temperaturen,

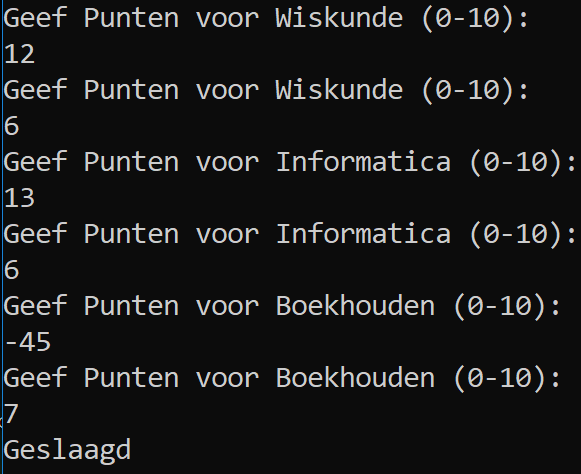
gemiddelde van de negatieve temperaturen,

aantal nul-temperaturen.

## Opdracht 4.28 – Examens

We hernemen opnieuw de opgave Examens (Zie opdracht 3.22). Lees de examenuitslagen in voor drie vakken (wiskunde, boekhouden en informatica). Elk van de vakken staat op 10 punten. Voeg de validatie toe bij het inlezen van de punten, zodat je punten kan blijven ingeven totdat je een juiste waarde hebt voor de punten van wiskunde, boekhouden en informatica. De student is geslaagd indien hij/zij voor wiskunde minstens 6/10 haalt, en voor boekhouden en informatica samen minstens 12/20. Toon op het scherm of de student geslaagd is, en indien de student niet geslaagd is, toon je ook de reden daarvoor.



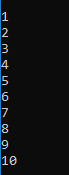
## Opdracht 4.29 – Lidgeld

Een vereniging vraagt € 10,00 lidgeld per jaar. Leden ouder dan 50 jaar krijgen € 2,00 reductie. Per kind ten laste wordt € 1,00 reductie gegeven (met een maximum van € 5,00). Indien het jaarinkomen onder € 12 500,00 ligt, wordt € 2,50 korting gegeven. De maximale reductie per lid is € 8,50. Maak een programma dat de nodige gegevens inleest en het te betalen lidgeld toont. Het programma eindigt wanneer bij de naam ‘stop’ of ‘STOP’ ingegeven wordt.

# Building Block V - Arrays

## Opdracht 5.1 – Stop waarden in array en druk ze af

Maak een nieuwe array van het datatype int van 10 groot, vul de waarden in; index 0 heeft een waarde van 1, index 1 heeft een waarde van 2, …, index 9 heeft een waarde van 10. Druk daarna alle entries af op het scherm.

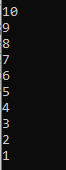
|  |  |
| --- | --- |
| Index | Value |
| 0 | 1 |
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |
| 4 | 5 |
| 5 | 6 |
| 6 | 7 |
| 7 | 8 |
| 8 | 9 |
| 9 | 10 |

## Opdracht 5.2 – Stop eerste 10 even getallen in array

Maak een nieuwe array van het datatype int van 10 groot, vul de waarden in; index 0 heeft een waarde van 2, index 1 heeft een waarde van 4, …., index 9 heeft een waarde van 20 Druk daarna alle entries af op het scherm.

|  |  |
| --- | --- |
| Index | Value |
| 0 | 2 |
| 1 | 4 |
| 2 | 6 |
| 3 | 8 |
| 4 | 10 |
| 5 | 12 |
| 6 | 14 |
| 7 | 16 |
| 8 | 18 |
| 9 | 20 |

## Opdracht 5.3 – Druk waarden omgekeerd af

Maak een nieuwe array van het datatype int van 10 groot, vul de waarden in; index 0 heeft een waarde van 1, index 1 heeft een waarde van 2, …., index 9 heeft een waarde van 10 Druk daarna alle entries omgekeerd af op het scherm.

|  |  |
| --- | --- |
| Index | Value |
| 0 | 1 |
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |
| 4 | 5 |
| 5 | 6 |
| 6 | 7 |
| 7 | 8 |
| 8 | 9 |
| 9 | 10 |

## Opdracht 5.4 – Druk negative waarden

Maak een nieuwe array van het datatype int van 10 groot, vul de waarden in; index 0 heeft een waarde van 1, index 1 heeft een waarde van 2, …., index 9 heeft een waarde van 10 Druk daarna de negatieve waarden van elke entry af, zonder de waarden van de array zelf negatief te zetten.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Voor |  | Print | Na |  |
| Index | Value |  | Index | Value |
| 0 | 1 | -1 | 0 | 1 |
| 1 | 2 | -2 | 1 | 2 |
| 2 | 3 | -3 | 2 | 3 |
| 3 | 4 | -4 | 3 | 4 |
| 4 | 5 | -5 | 4 | 5 |
| 5 | 6 | -6 | 5 | 6 |
| 6 | 7 | -7 | 6 | 7 |
| 7 | 8 | -8 | 7 | 8 |
| 8 | 9 | -9 | 8 | 9 |
| 9 | 10 | -10 | 9 | 10 |

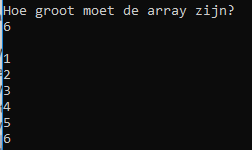
## Opdracht 5.5 – Som van entries uit een array

Maak een nieuwe array van het datatype int van 10 groot, vul de waarden in; index 0 heeft een waarde van 1, index 1 heeft een waarde van 2, …., index 9 heeft een waarde van 10. Druk daarna de som af. Gebruik hiervoor niet de method Sum.



## Opdracht 5.6 – Geef lengte van array

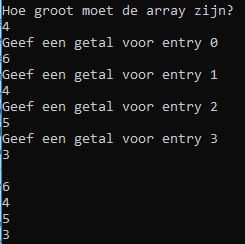
Maak een nieuwe array van het datatype int. Laat de gebruiker het aantal grootte van de array bepalen. Vul daarna elke waarde van de array in en druk het af op het scherm.

Vb. de gebruiker heeft 6 als lengte in.

|  |  |
| --- | --- |
| Index | Value |
| 0 | 1 |
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |
| 4 | 5 |
| 5 | 6 |

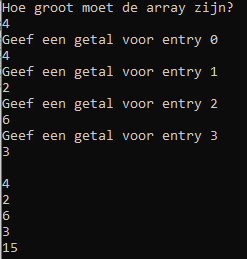
## Opdracht 5.7 – Geef lengte van array en geef waarde

Maak een nieuwe array van het datatype int. Laat de gebruiker het aantal entries van de array bepalen. Ga doorheen elke entry van de array en vraag aan de gebruiker om een waarde. Druk daarna de array af.



## Opdracht 5.8 – Geef lengte van array en geef waarde en som

Maak een nieuwe array van het datatype int. Laat de gebruiker het aantal entries van de array bepalen. Ga doorheen elke entry van de array en vraag een de gebruiker om een waarde. Druk daarna de array af. Maak de array 1 entry groter dan de gebruiker gevraagd heeft. De laatste entry gebruik je om de som te bewaren.

Vb. De gebruiker heeft 5 als lengte van de array (+1 entry voor de som)

|  |  |  |
| --- | --- | --- |
| Index | Value | PRINT |
| 0 | 2 | 2 |
| 1 | 5 | 5 |
| 2 | 3 | 3 |
| 3 | 1 | 1 |
| 4 | 5 | 5 |
| 5 (som) | 16 | 16 |

## Opdracht 5.9 – Grootste getal, kleinste getal, gemiddelde

Laat de gebruiker 10 waarden ingeven, de waardes kunnen een komma getal bevatten. Nadat de gebruiker de getallen ingegeven heeft druk je de array nogmaals af, je toont ook het grootste en kleinste ingegeven getal, je toont ook het gemiddelde.



## Opdracht 5.10 EvenOneven

* We declareren een reeks getallen van het datatype int waar 20 getallen kunnen worden gestockeerd.
* We vullen deze getallen op met random getallen tussen 0 en 30
* We gaan nog twee extra array’s aanmaken waar ook 20 getallen kunnen in gestockeerd worden. De ene noemen we even de andere oneven. Initieel zullen in deze twee laatste array’s **geen data** aanwezig zijn.
* Het programma loopt onze reeks getallen af en zal de even getallen in de array even plaatsen en de oneven getallen in de array oneven.
* Als uitvoer krijgen we op scherm de waarden van onze originele array, de waarden van onze array even en de waarden van onze array oneven.

Let wel :

Onze twee extra’s array’s zullen natuurlijk ledige elementen bevatten. Als er bijvoorbeeld maar 8 even getallen aanwezig zijn heb ik ruimte over want we hadden voorzien van 20 elementen. Om dat op te vangen ga je als laatste element van deze array’s de waarde -99 ingeven (=einde).

TIP:

je zal dus een mechanisme moeten inbouwen om te weten hoeveel even én hoeveel oneven getallen er aanwezig zijn. Dus stel de array even heeft 4 even getallen dan krijgen we:

4 10 2 8 -99 -99 -99 …

Bij het uitlezen van deze array moet je dan ook stoppen op index 5.

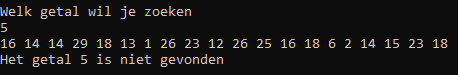
Vb. Voor 6 Entries

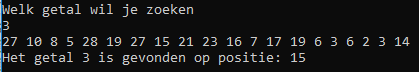
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Getallen |  | Even |  | Oneven |  |
| Index | Waarde | Index | Waarde | Index | Waarde |
| 0 | 2 | 0 | 2 | 0 | 3 |
| 1 | 3 | 1 | 12 | 1 | 15 |
| 2 | 12 | 2 | -99 | 2 | 9 |
| 3 | 15 | 3 | -99 | 3 | 7 |
| 4 | 9 | 4 | -99 | 4 | -99 |
| 5 | 7 | 5 | -99 | 5 | -99 |



## Opdracht 5.11: Zoeken

* We declareren een reeks getallen van het datatype int waar 20 getallen kunnen worden gestockeerd.
* We vullen deze getallen op met random getallen 0 en 30
* We vragen aan de gebruiker welk getal hij in de array wil zoeken
* Het programma zoekt het getal.
  + Indien het getal is gevonden krijgen we de melding : “gevonden op positie X” [X is natuurlijk de index waar het getal staat dat we zochten]
  + Indien het getal niet is gevonden krijgen we de melding: “getal staat niet in de reeks”.





## Opdracht 5.12 – Som met opvolger

Begin met **int[] a={5, 3, 1, -1, -3};**

en schrijf een programma die de waarden in a vervangt door de som van het element en zijn opvolger.

Bv. het eerste element (op positie 0) wordt 8 (5+3). Het tweede wordt 4 (3+1), enz. Het laatste element kun je niet vervangen bij gebrek aan een opvolger.

Uiteindelijk zal het array de waarden 8, 4, 0, -4, -3 bevatten.

Toon het eindresultaat op het scherm (getallen onder elkaar).

|  |  |  |  |
| --- | --- | --- | --- |
| Voor |  | Na |  |
| Index | Waarde | Index | Waarde |
| 0 | 5 | 0 | 5 + 3 = 8 |
| 1 | 3 | 1 | 3 + 1 = 4 |
| 2 | 1 | 2 | 1 + -1 = 0 |
| 3 | -1 | 3 | -1 + -3 = -4 |
| 4 | -3 | 4 | -3 |



## Opdracht 5.13 – Lijst waarden gescheiden door komma’s

Begin met **int[] a={5, 3, 1, -1, -3};**

en schrijf een programma dat de waarden netjes achtereen op het scherm zet met komma's en spaties ertussen :

5, 3, 1, -1, -3

Gebruik hiervoor een loop die zich aanpast aan de lengte van het array, dus als we array a zouden opvullen met meer of minder waarden, dan moet het programma nog steeds correct werken.

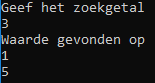


## Opdracht 5.14 – Zoek waarde in een array

Begin met **int[] a={5, 3, 1, -1, -3, 3, 9, -4};**

En schrijf een programma dat de gebruiker om een waarde vraagt en die waarde zoekt in het array. Telkens de waarde gevonden wordt, toont het programma de array index (=positie) waarop dit gebeurde.

Bv. indien de gebruiker 3 intypt, toont het programma 1 5. Indien de waarde niet gevonden werd, toont het programma niets.



## Opdracht 5.15 - Boerderijdieren

Begin met **String[] boerderijDieren={"kip", "koe", "paard", "geit", "schaap"};**

en schrijf een programma dat de gebruiker om een dier vraagt. Het programma toont vervolgens of het wel of niet om een boerderij dier gaat.

## Opdracht 5.16 – Punten voor een vak

Begin met

**String[] vakken={"Frans", "Engels", "Wiskunde", "Duits", "L.O."};**

**int[] scores={34, 55, 20, 10, 80};**

|  |  |  |  |
| --- | --- | --- | --- |
| **Vakken** |  | **Scores** |  |
| Index | Waarde | Index | waarde |
| 0 | Frans | 0 | 34 |
| 1 | Engels | 1 | 55 |
| 2 | Wiskunde | 2 | 20 |
| 3 | Duits | 3 | 10 |
| 4 | L.O. | 4 | 80 |

De punten voor een vak staan steeds op de overeenkomstige positie in het punten array.

Bv. op index 2 vinden we dat de student voor “Wiskunde” en 20 punten scoorde (op 100).

Het programma toont voor welke vakken de student onvoldoende behaalde.

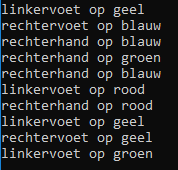


## Opdracht 5.17 - Twister

Schrijf een programma dat 10 willekeurige Twister opdrachten toont. Zo'n opdracht is steeds de combinatie van een kleur en een lichaamsdeel.

**kleuren : rood, geel, groen, blauw**

**lichaamsdelen : linkerhand, rechterhand, linkervoet, rechtervoet**



## Opdracht 5.18 – Invoegen in een array

Begin met **int[] a={5, 3, 1, -1, -3, 7, 9, 10, 23};**

en schrijf een programma dat de gebruiker om een positie en een getal vraagt en dan dat getal op die positie invoegt.

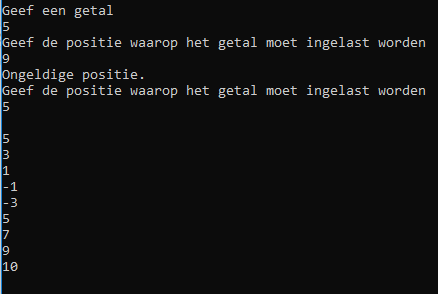
Het laatste getal zal dan noodgedwongen niet meer in de array voorkomen (een array kan immers niet 'groeien').

Vervolgens toont het programma de getallen in de array (naast elkaar gescheiden door spaties). Mocht de gebruiker een ongeldige positie opgeven meldt het programma dit en probeert opnieuw.

Bv. indien de gebruiker 15 wil inlassen op positie 7 krijgen we

5, 3, 1, -1, -3, 7, 9, 15, 10

|  |  |  |  |
| --- | --- | --- | --- |
| Voor |  | Na |  |
| Index | Waarde | Index | Waarde |
| 0 | 5 | 0 | 5 |
| 1 | 3 | 1 | 3 |
| 2 | 1 | 2 | 1 |
| 3 | -1 | 3 | -1 |
| 4 | -3 | 4 | -3 |
| 5 | 7 | 5 | 7 |
| 6 | 9 | 6 | 9 |
| 7 | 10 | 7 | 15 |
| 8 | 23 | 8 | 10 |



## Opdracht 5.19 – Verwijderen uit een array

Begin met String[] **boerderijDieren={"kip", "koe", "paard", "geit", "schaap"};**

en schrijf een programma dat de gebruiker om een positie vraagt en dan het dier op die positie verwijderd. Op het einde van het array zullen we noodgedwongen een tekst "LEEG" introduceren (een array kan immers niet 'krimpen').

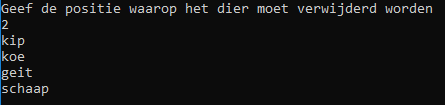
Mocht de gebruiker een ongeldige positie opgeven meldt het programma dit en probeert opnieuw.

Vervolgens toont het programma de lijst met dieren gescheiden door spaties (maar "LEEG" wordt niet getoond!).

Bv. indien de gebruiker op positie 2 wil wissen toont het programma

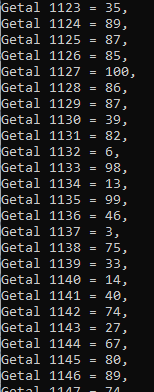
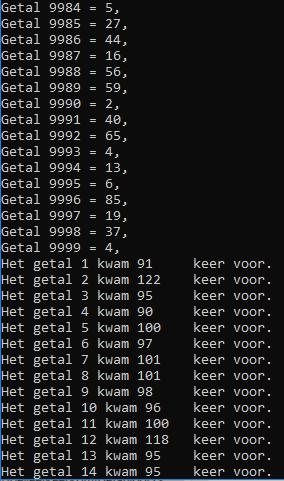
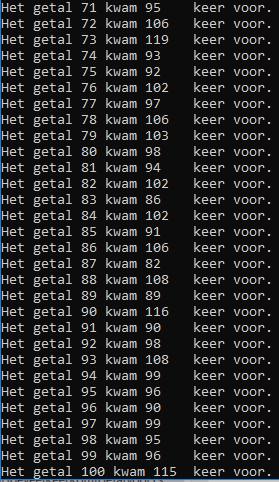
kip koe geit schaap

|  |  |  |  |
| --- | --- | --- | --- |
| Voor |  | Na |  |
| Index | Waarde | Index | Waarde |
| 0 | “kip” | 0 | “kip” |
| 1 | “koe” | 1 | “koe” |
| 2 | “paard” | 2 | “geit” |
| 3 | “geit” | 3 | “schaap” |
| 4 | “schaap” | 4 | “LEEG” |



## Opdracht 5.20 – Bepaal hoeveel keer een getal voorkomt

Maak een programma die 10000 random getallen genereert tussen 1 en 100. Bepaal daarna hoeveel keer elk random getal voorkomt.

# Building BLock VI – OOP

## Opdracht 6.1 – kerstmis

Inhoud:

* Probleemstelling
* Model
* Input - Output

**Probleemstelling**

De kerstman heeft één fabriek. In een fabriek kunnen er twee feeën werken. De fabriek heeft één kast waarin er speelgoed geplaatst kan worden. Het is de taak van de feeën om speelgoed te maken en in te pakken. Hierbij heeft elke Fee zijn eigen job, een fee kan een ‘Inpakker’ zijn of een fee kan een ‘maker’ zijn. Telkens een fee één stuk speelgoed verwerkt heeft krijgt hij hiervoor beloont met een bedrag in euro. Elke job word verschillend betaalt, een ‘Maker’ verdient meer dan een ‘Inpakker’. Een stuk speelgoed kan drie statussen hebben; ‘niet gemaakt’, ‘gemaakt’ en ingepakt’. Een stuk speelgoed is gemaakt van zodra een Fee (de maker) er een naam aan gegeven heeft. Op het einde van de dag wil de kerstman een rapport krijgen waar volgende informatie opstaat:

* Alle feeën (hun volledige naam, hun job en hun verdiend bedrag).
* Een lijst van alle stukken speelgoed die niet gemaakt zijn.
* Een lijst van alle stukken speelgoed die gemaakt zijn maar niet ingepakt.
* Een lijst van alle stukken speelgoed die ingepakt zijn.

**Model**

**Job**

Fields:

* Private decimal bedragPerVerwerking
* Private string naam

Methods:

* Public Job(string naam, decimal bedragPerVerwerking)
* Getters en Setters

**Speelgoed**

Fields

* Private string naam (default value = “/”)
* Private bool isGemaakt (default value = false)
* Private bool isIngepakt (default value = false)

Methods:

* Public Speelgoed()
* Getters en Setters
  + SetIsIngepakt(bool isIngepakt)
    - Zorg ervoor dat *isIngepakt* enkel op true komt te staan als het stuk speelgoed al gemaakt is. Als het stuk speelgoed nog niet gemaakt is toon je een gepaste boodschap.
  + SetIsGemaakt(bool isGemaakt)
    - Zorg ervoor dat *IsGemaakt* enkel veranderd kan worden als *isIngepakt* op op false staat
* Public override string ToString()
  + Return de naam van het stuk speelgoed, de value van *isGemaakt* en de value van *isIngepakt.*

**Fee**

Fields:

* Private string naam
* Private string voornaam
* Private readonly Job job
* Private int aantalSpeelgoedVerwerkt (default value = 0)

Methods:

* Public Fee(string naam, string voornaam, Job job)
* Public Job GetJob() (Getter)
* Public string GetVolledigeNaam()
  + Er bestaat geen getter van naam en voornaam
  + Return de samengevoegde waarde van naam en voornaam
* Public decimal VerdiendBedrag()
  + Het verdiend bedrag is het aantalSpeelgoedVerwerkt maal bedragPerVerwerking (dit hangt af van de job, ‘maker’ of ‘inpakker’)
* Public void MaakSpeelgoed(Speelgoed speelgoed, string speelgoedNaam)
  + Validatie bij begin van de method
    - Als de fee een inpakker is, toon een gepaste boodschap en ga uit de method
    - Als het stuk speelgoed al gemaakt is, toon een gepaste boodschap en ga uit de method
  + Geef het stuk speelgoed een naam
  + Zet het stuk speelgoed op als ‘gemaakt’
  + Het aantalSpeelgoedVerwerkt vermeerderd met één
  + Toon een boodschap dat het stuk speelgoed succesvol gemaakt is
* Public void PakSpeelgoedIn(Speelgoed speelgoed)
  + Validatie bij begin van method
    - Als de fee een maker is, toon een gepaste boodschap en ga uit de method
    - Als het stuk speelgoed nog niet gemaakt is, toon een gepaste boodschap en ga uit de method
    - Als het stuk speelgoed al ingepakt is, toon een gepaste boodschap en ga uit de method
  + Zet het stuk speelgoed op als ‘ingepakt’
  + Het aantalSpeelgoedVerwerkt vermeerderd met één
  + Toon een boodschap dat het stuk speelgoed succesvol ingepakt is

**Fabriek**

Fields:

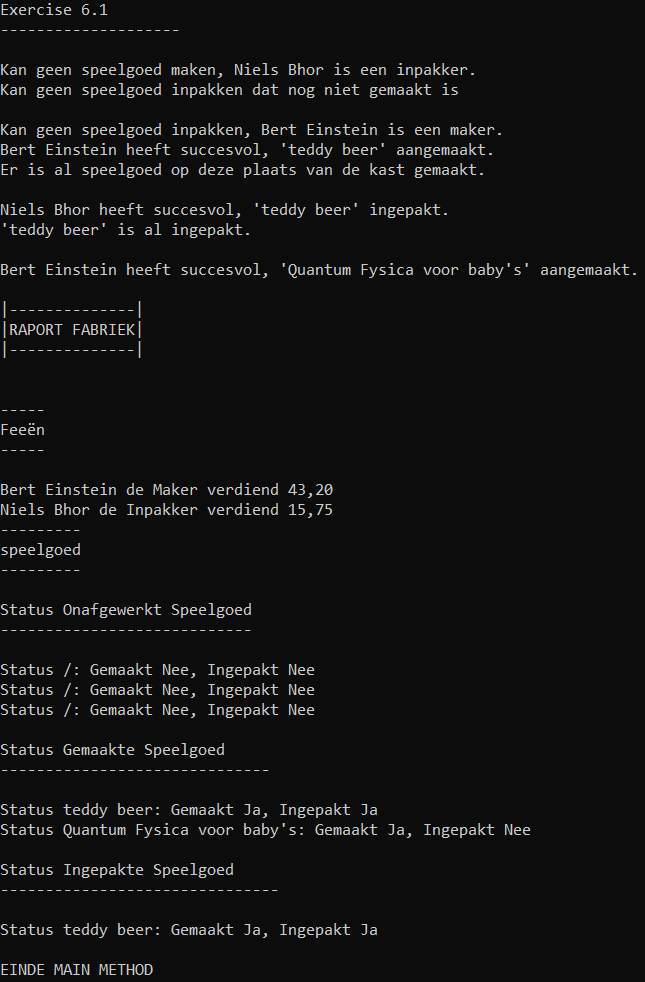
* Private Fee fee1
* Private Fee fee2
* Private Speelgoed[] kast

Methods

* Fabriek(Fee fee1, Fee fee2, int kastGrootte)
  + Initsialiseer ‘kast’, de lengte is de kastGrootte
  + Loop door elke entry van de array, stop in elke entry een nieuw stuk speelgoed
* Getters en Setters
  + Enkel getters
* Public decimal TotaleKost()
  + De totale kost is de som van het verdiend bedrag van elke fee
* private void PrintGemaaktSpeelgoed()
  + Druk een lijst af van alle gemaakte stukken speelgoed
    - Gebruik de **ToString()** method van het stuk speelgoed.
* private void PrintIngepaktSpeelgoed()
  + Druk een lijst af van alle ingepakte stukken speelgoed
    - Gebruik de **ToString()** method van het stuk speelgoed.
* private void PrintOnafgewerktSpeelgoed()
  + Druk een lijst af van alle onafgewerkte stukken speelgoed
    - Gebruik de **ToString()** method van het stuk speelgoed.
* Public void PrintRaportFabriek()
  + Toon de gegevens van elke fee
    - Volledige naam
    - Job; ‘maker’ of ‘inpakker’
    - Verdiend bedrag
  + Toon al het onafgewerkt speelgoed (reeds aangemaakte method)
  + Toon al het gemaakt speelgoed (reeds aangemaakte method)
  + Toon al het ingepakt speelgoed (reeds aangemaakte method)

Input – Output





# Building BLock VII – OOP

## Opdracht 7.1 Videotheek

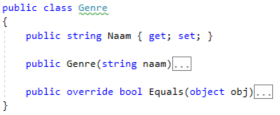
Inhoud:

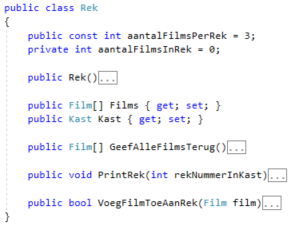
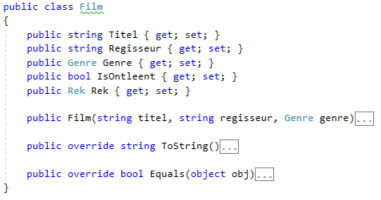
* Probleemstelling
* Model
* Input - Output

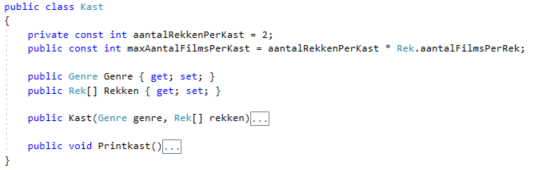
**Probleemstelling**

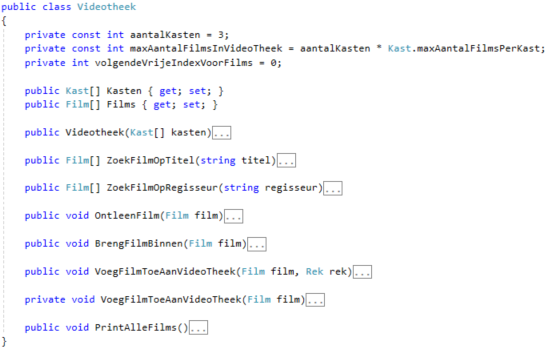
Een eigenaardige man ‘Cosmos’ genaamd die tijdreist vanuit de toekomst vraagt je om een een ‘simpele’ console applicatie in C# te schrijven. Je hebt Cosmos met handen en voeten proberen uit te leggen dat videotheken niet meer van deze tijd zijn. Hij wil alsnog deze applicatie. Cosmos zegt dat hij drie kasten heeft om in zijn videotheek te plaatsen. Elke kast heeft twee rekken waar hij de films kan bewaren. Een film heeft een titel, regisseur en een genre. Het moet mogelijk zijn om Films in de videotheek te plaatsen. Aan een kast hangt ook een genre, Cosmos wil dat je enkel films in een rek kan plaatsen waarvan de film hetzelfde genre heeft als de kast waar het rek aanhangt. Bv. Een film met genre ‘Sci-Fi’ mag enkel in een Rek komen waarvan de kast aan dit rek het genre ‘Sci-Fi’ heeft. Je moet ook films kunnen vergelijken met elkaar, een film is hetzelfde wanneer de titel de regisseur en het genre hetzelfde zijn. Klanten moeten films kunnen ontlenen, je kan geen geen films ontlenen die nog niet in een rek ziten of als die film al ontleent is. Je moet ook films kunnen terugbrengen, je kan geen films binnenbrengen die nog niet in een rek zitten of als de film nog niet ontleent is. Het moet mogelijk zijn om een film te zoeken met een bepaalde titel of een regisseur. Als laatste feature wil Cosmos dat je alle films in de videotheek kunt afdrukken.

**Model**



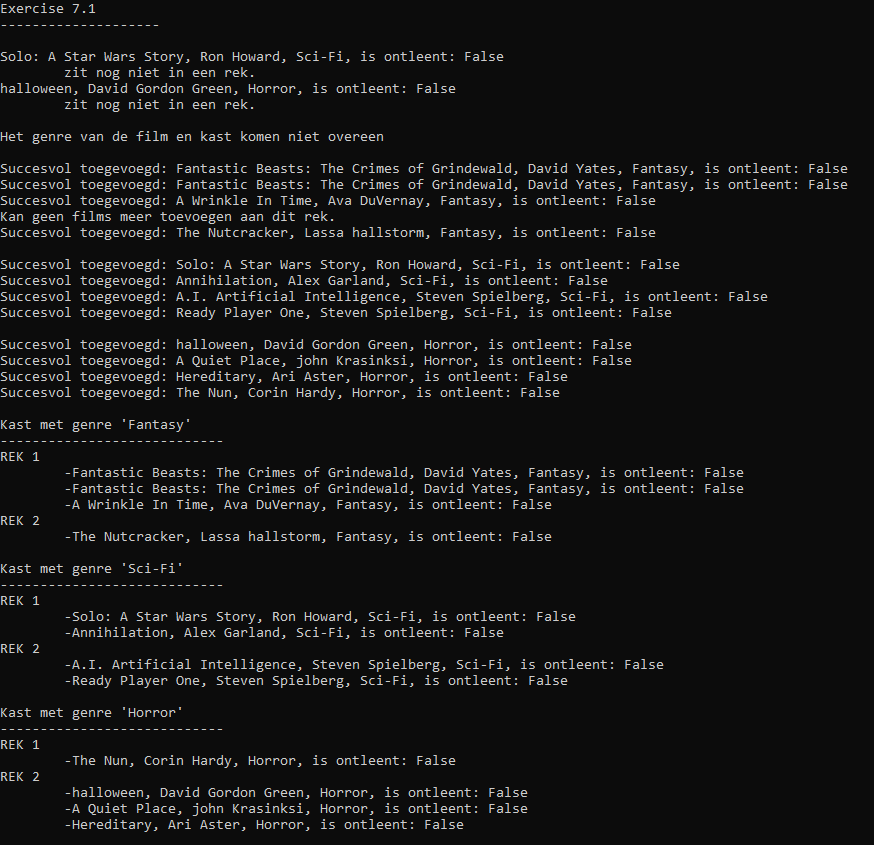


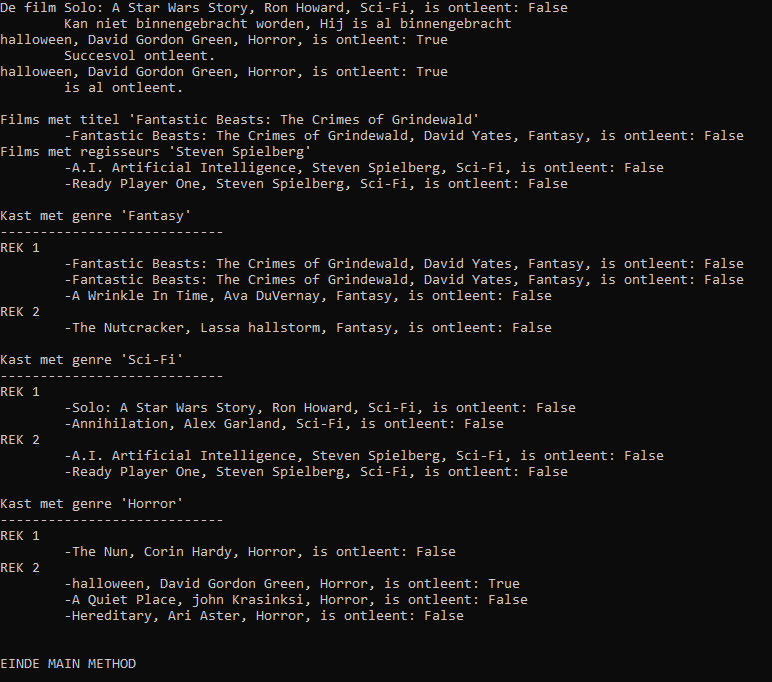


**Input – Output**









# Building BLock VIII – Methods

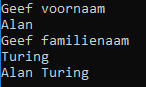
## Opdracht 8.1 VolledigeNaam

**RunExercise**

1. Vraag aan de gebruiker om zijn voornaam in te geven
2. Vraag aan de gebruiker om zijn familienaam in te geven
3. Roep *PrintVolledigeNaam* op

void **PrintVolledigeNaam(**string voornaam, string familienaam)

Print de volledige naam



## Opdracht 8.2 Volledige naam met return string

**RunExercise**

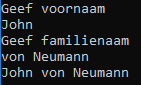
Vraagt aan de gebruiker om hun voornaam en familienaam in te geven en sla ze op in memory.

Roep de method *GetVolledigeNaam* en sla zijn return waarde op in een nieuwe variable.

Druk vervolgens de gereturneerde waarde van *GetVolledigeNaam* af.

**GetVolledigenaam**

Deze method verwacht twee argumenten, een voornaam en een familienaam en heeft de volledige naam terug.



## Opdracht 8.3 Volledige Naam vraag naam en voornaam in method

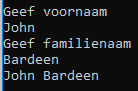
**RunExercise**

Roep de method *PrintVolledigeNaam* en sla zijn return waarde op in een nieuwe variable.

**PrintVolledigeNaam**

Vraagt aan de gebruiker om hun voornaam en familienaam in te geven en sla ze op in memory.

Druk vervolgens de volledige naam af.



## Opdracht 8.4 Volledige naam Vraag Naam en voornaam in method en return

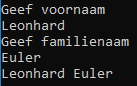
**RunExercise**

Roep de method *GetVolledigeNaam* en bewaar zijn return waarde in een variable.

Print de gereturneerde waarde af.

**GetVolledigenaam**

Deze method verwacht twee argumenten, een voornaam en een familienaam en heeft de volledige naam terug.



## Opdracht 8.5 Valideer op een lege string

**RunExercise**

Vraag aan de gebruiker om hun lievelingsdier.

Controleer aan de hand van *IsStringLeeg* als de gebruiker effectief iets heeft meegegeven.

Als hij iets meegegeven heeft, druk je zijn lievelingsdier af op het scherm, zo niet, toon je een gepaste boodschap

**IsStringLeeg**

Deze method verwacht één string argument.

Als return waarde heeft hij terug als de meegegeven string leeg of ingevuld is (bool).

Maak hiervan gebruik van de ingebouwde functie die in C# zit. **string.IsNullOrEmpty(*value)*.**





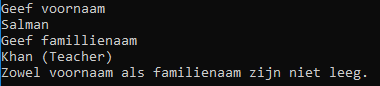
## Opdracht 8.6 Valideer Naam en voornaam

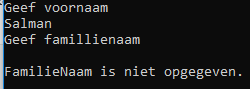
**RunExercise**

1. Vraag aan de gebruiker om zijn voornaam
2. Controleer aan de hand van *IsStringLeeg* als de voornaam ingevuld is
   1. Ja, ga door met stap 3
   2. Nee, Geef een gepaste boodschap en stop deze method met **return**
3. Vraag aan de gebruiker om zijn famileinaam
4. Controleer aan de hand van *IsStringLeeg* als de familienaam ingevuld is
   1. Ja, Ga door met stap 5
   2. Nee, Geef een gepaste boodschap en stop deze method met **return**
5. Print een boodschap die zegt dat zowel de voornaam als famileinaam inguvld zijn.

**IsStringLeeg**

Zie opdracht 8.5





## Opdracht 8.7 Valideer Naam en Voornaam En Print

**RunExercise**

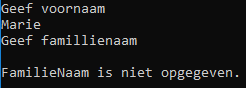
1. Vraag aan de gebruiker om zijn voornaam
2. Controleer aan de hand van *IsStringLeeg* als de voornaam ingevuld is
   1. Ja, ga door met stap 3
   2. Nee, Geef een gepaste boodschap en stop deze method met **return**
3. Vraag aan de gebruiker om zijn famileinaam
4. Controleer aan de hand van *IsStringLeeg* als de familienaam ingevuld is
   1. Ja, Ga door met stap 5
   2. Nee, Geef een gepaste boodschap en stop deze method met **return**
5. Print de volledige naam met *PrintVolledigeNaam*

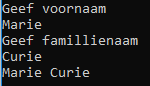
**IsStringLeeg**

Zie opdracht 8.5

**PrintVolledigeNaam**

Zie opdracht 8.1



## Opdracht 8.8 Print volledige naam en valideer method

**RunExercise**

1. Roep de method *PrintEnValideerVolledigeNaam* op

**PrintEnValideerVolledigeNaam**

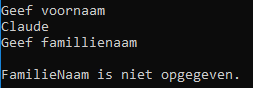
1. Vraag aan de gebruiker om zijn voornaam
2. Controleer aan de hand van *IsStringLeeg* als de voornaam ingevuld is
   1. Ja, ga door met stap 3
   2. Nee, Print een gepaste boodschap en stop deze method met **return**
3. Vraag aan de gebruiker om zijn famileinaam
4. Controleer aan de hand van *IsStringLeeg* als de familienaam ingevuld is
   1. Ja, Ga door met stap 5
   2. Nee, Print een gepaste boodschap en stop deze method met **return**
5. Print de volledige naam met *PrintVolledigeNaam*

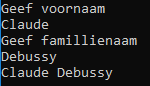
**IsStringLeeg**

Zie opdracht 8.5

**PrintVolledigeNaam**

Zie opdracht 8.1



## Opdracht 8.9 Get volledige Naam en valideer

**Class Opdracht08\_09GetVolledigeNaamEnValideer**

Plaats twee constanten, deze zijn bedoeld om een error message terug te geven aan de gebruiker.

private const string VoornaamNietGegeven = "Voornaam is niet opgegeven";

private const string FamilieNaamNietGegeven = "FamilieNaam is niet opgegeven";

**RunExercise**

1. Roep de method *GetAndValidateVolledigeNaam* en bewaar zijn resultaat in een variable genaamd **naam**.
2. Controleer als **naam** gelijk is aan aan één van de twee constanten
   1. Zo ja, toon een gepaste boodschap en de reden;
   2. Zo Nee, print de volledige naam

**GetAndValidateVolledigenaam**

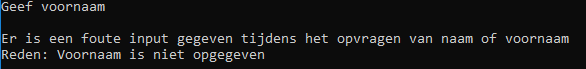
1. Vraag aan de gebruiker om de voornaam
2. Controleer als de voornaam ingevuld is
   1. Ja, Ga door met stap 3
   2. Nee, return de correcte constante
3. Vraag aan de gebruiker om de familienaam
   1. Ja, Ga door met stap 4
   2. Nee, return de correct constante
4. Returneer de volledige naam a.d.h.v.

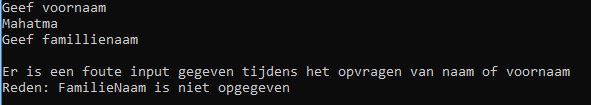
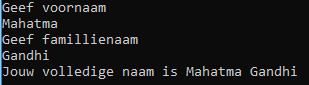
**IsStringLeeg**

Zie opdracht 8.5

**GetVolledigeNaam**

Zie opdracht 8.2



## Opdracht 8.10 Print Lijn

**RunExercise**

1. Vraag aan de gebruiker hoelang zijn lijn moet zijn
2. Roep de method *TrekLijn* op

void **TrekLijn** (int lengte)

1. Druk een *lengte* aantal keer het karakter ‘-‘ af.

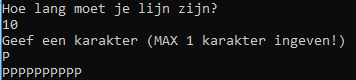
## Opdracht 8.11 Print lijn met zelf gekozen karakter

**RunExercise**

1. Vraag aan de gebruiker hoe lang de lijn moet zijn
2. Vraag aan de gebruiker **1** karaketer
3. Roep de method *Treklijn* Op

void **TrekLijn** (int lengte, char karakter)

1. Druk een *Lengte* aantal keer *karakter* af.



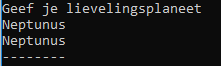
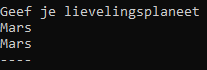
## Opdracht 8.12 Print titel en lijn

**RunExercise**

1. Vraag aan de gebruiker om zijn lievelingsplaneet op te geven.
2. Controleer als de gebruiker iets ingegeven heeft
   1. Ja, Ga naar stap 3
   2. Nee, Toon een gepaste boodschap en stop de method met **return**
3. Druk de planeet af
4. Roep de method *TrekLijn* Op. (zorg ervoor dat er zoveel lijntjes gezet worden als er karakters zijn in de planeet.

**TrekLijn**

Zie opdracht 8.10

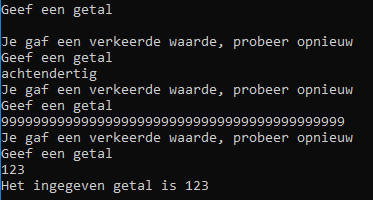
## Opdracht 8.13 Try Parse Integer

**RunExercise**

1. Roep de method *GetGetal* en bewaar zijn return waarde.
2. Druk het ingegeven getal af op het scherm

int **GetGetal**()

Maak een method die de gebruiker om een getal vraagt. Als het getal niet correct geparsed kan worden vraag je de gebruiker opnieuw om een getal in te geven tot zolang hij een correct getal meegeeft. Gebruik hiervoor de ingebouwde functie int.TryParse van het .NET framework.



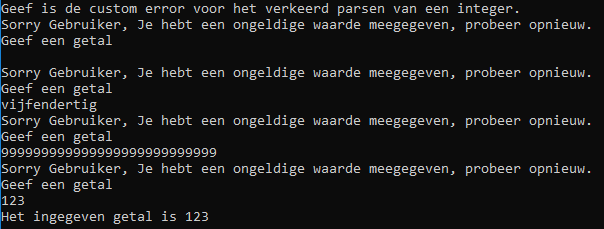
## Opdracht 8.14 Try Parse Integer Custom Error Message

**RunExercise**

1. Vraag aan de gebruiker wat voor custom error er gegeven moet worden als een getal niet geparsed kan worden.
2. Bewaar de return waarde van *GetGetal*
3. Druk het getal af

Int **GetGetal**(string customError)

Uitbreiding op GetGetal van opdracht 8.13



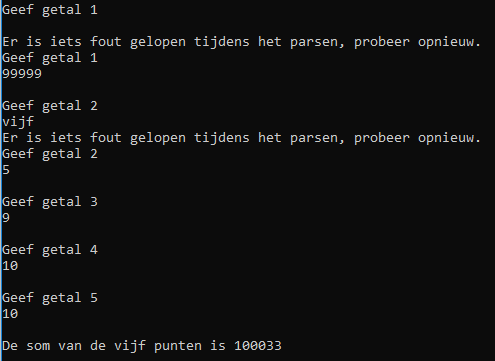
## Opdracht 8.15 Som Punten en Check De punten met de GetGetal method

**RunExercise**

1. Loop 5 keer door een lus
   1. Vraag een getal via de *GetGetal* Method
   2. Tel het getal op bij de som
2. Druk de som af

int **GetGtal(**int getalNummer)

uitbreiding op GetGetal van opdracht 8.13



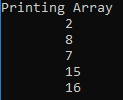
## Opdracht 8.16 Print array

**RunExercise**

1. Initialiseer een array { 2, 8, 7 ,15 ,16 }
2. Roep de method *PrintArray*

void **PrintArray(**int[] array)

1. Itereer door elke entry van de array en druk de waarde af



## Opdracht 8.17 Zoek Waarde In Array

**RunExercise**

1. Initialiseer een array { 2, 6, 2, 8, 9, 12, 7 }
2. Roep de method *PrintArray* op
3. Vraag aan de gebruiker om een ‘te zoeken getal’ a.d.h.v. de *GetGetal* method
4. Bewaar het resultaat van method *GeefIndexVanWaardeInArray*
5. Controleer als het resultaat gelijk is aan -1
   1. Ja, Print een boodschap dat de te zoeken waarde niet gevonden is
   2. Nee, Print de te zoeken waarde en de gevonden index af

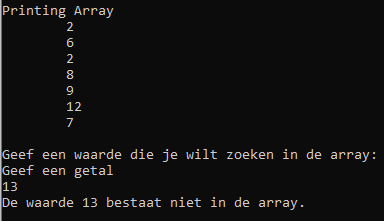
int **GeefIndexVankWaardeInArray (**int[] array, int teZoekenWaarde)

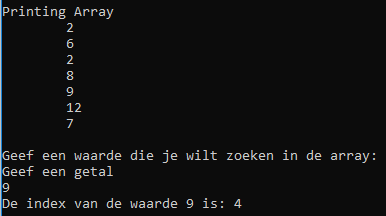
1. Loop door elke waarde in de array
   1. Als de waarde van de index in array gelijk is aan de *teZoekenWaarde* return je de index
2. Als de teZoekenWaarde Niet gevonden is return je **-1**

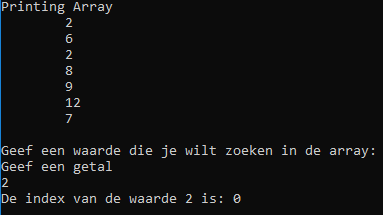
**GetGetal**

Zie Opdracht 8.13

**PrintArray**

Zie Opdracht 8.16





## Opdracht 8.18 Maak Array

**RunExercise**

1. Vraag aan de gebruiker de lengte van een array a.d.h.v. *GetGetal*
2. Vraag aan de gebruiker om een default waarde dat elke value van de array moet hebben
3. Bewaar de return van *MaakArray* Method
4. Print de array met *PrintArray*

Int[] **MaakArray(**int lengte, int defaultWaarde = 0)

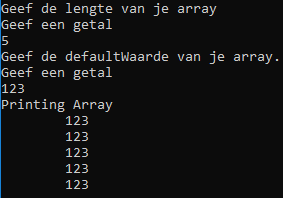
1. Initialiseer een array met grootte van *lengte*
2. Itereer door elke entry van de array en zet de waarde gelijk aan de default waarde
3. Return de array

**GetGetal**

Zie Opdracht 8.13

**PrintArray**

Zie Opdracht 8.16



## Opdracht 8.19 Geef Waarde aan entry van array

**RunExercise**

1. Bewaar een array a.d.h.v. *MaakArray* method, lengte = 5, defaulte waarde mag je leeg laten
2. Print de Array af (BEFORE)
3. Vraag aan de gebruiker welke index je wilt veranderen met *GetGetal*
4. Vraag aan de gebruiker naar welke waarde je de entry op die index wilt veranderen
5. Roep *ChangeEntryArray* op
6. Print de Array af (AFTER)

void **ChangeEntryArray(**int[] array, int changeIndex, int value)

1. Controleer met de *NumberBetweenTwoValues* method als je een geldig index hebt meegegeven (Een geldige index is groter dan -1 en kleiner dan de lengte van je array)
   1. Geldig, verander de value van de array op plaats *index*
   2. Ongeldig, Toon een gepaste boodschap en stop de method met return

bool **NumberBetweenTwoValues(**int numberToCheck, int min, int max)

1. Return true als de *numberToCheck* groter of gelijk is aan *min* EN numberToCheck kleiner of gelijk is aan *max*

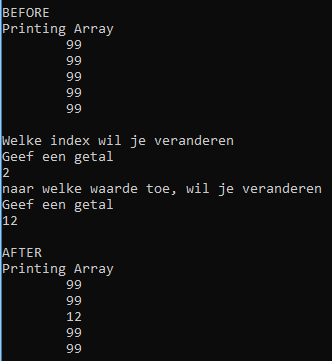
**GetGetal**

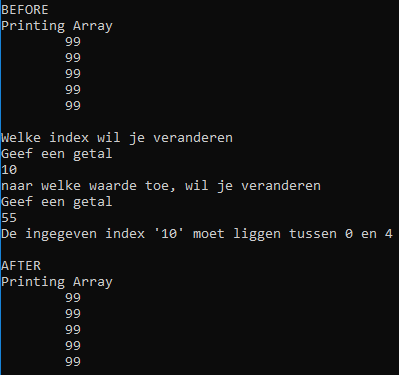
Zie Opdracht 8.13

**PrintArray**

Zie Opdracht 8.16

**MaakArray**

Zie opdracht 8.18



## Opdracht 8.20 Vul Elke Entry Van Array in

**RunExercise**

1. Vraag aan de gebruiker om de lengte van een array op te geven a.d.h.v. *GetGetal*
2. Bewaar een array a.d.h.v. *MaakArray*
3. Roep *ChangeEveryEntryOfArray* Op
4. Roep *PrintArray* Op

void **ChangeEveryEntryOfArray(**int[] array)

1. Itereer door elke entry van de array
   1. Vraag aan de gebruiker Welke waarde je in de index wilt opslaan a.d.h.v. *GetGetal*
   2. Roep *ChangeEntryArray* op

**GetGetal**

Zie Opdracht 8.13

**PrintArray**

Zie Opdracht 8.16

**MaakArray**

Zie opdracht 8.18

**NumberBetweenTwoValues**

Zie opdracht 8.19

**ChangeEntryArray**

Zie opdracht 8.19



# Building BLock IX – Methods

## Opdracht 9.1 Vrienden

### KlassDiagram



**Zwaai Vriend Uit**Print de tekst “{naam van persoon} zwaait {naam van vriend} uit.”

**Verwelkom Vriend**Print de tekst “{naam van persoon} Verwelkomt {naam van vriend}.”

### Doel

In **Opdracht09\_01Vrienden** in de **RunExercise** method.

Maak een nieuw object aan op basis van de **Persoon** class.

Roep de method **ZwaaiVriendUit** op.  
Roep de method **VerwelkomVriend** op

### Voorbeeld



## Opdracht 9.2 Dagboek

### KlassDiagram



**Schrijf Nieuwe Pagina**  
Vraag aan de gebruiker om een nieuwe pagina te schrijven.Voeg de nieuwe pagina toe aan de lijst **Boek.**

**PrintBoek**Itereer door het hele boek en print elke pagina af.  
Geef hier ook telkens bij welke pagina er afgedrukt word en hoeveel paginas zijn.

### Doel

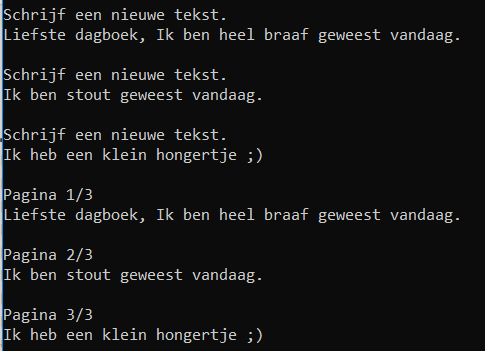
In **Opdracht09\_02Dagboek** in de **RunExercise** method.

Maak een nieuw object aan van de class **Dagboek.**

Roep een aantal keer de **SchrijfNieuwePagina** method op

Roep daarna de method **PrintBoek** op

### Voorbeeld



## Opdracht 9.3 Vuilbak

### ClassDiagram



**Vuilbak (Constructor)**Ze de property **GrootteVuilbak** gelijk aan de inkomende parameter **grootteVuilbak**

**WerpVuilnis(string vuilnis)**

* ALS de vuilnisbak vol zit
  + Print je een bericht naar de console dat deze vol zit
* ANDERS
  + Zeg je dat het {vuilnis} weg gesmete is
  + Voeg je {vuilnis} toe aan de **Inhoud van de vuilnisbak**

**LeegVuilnis**

* Print hoeveel stukken vuilnis geleegd werden
* Maak de inhoud van **InhoudVuilnisBak** leeg (zoek op welke method je hiervoor kunt gebruiken)

### Doel

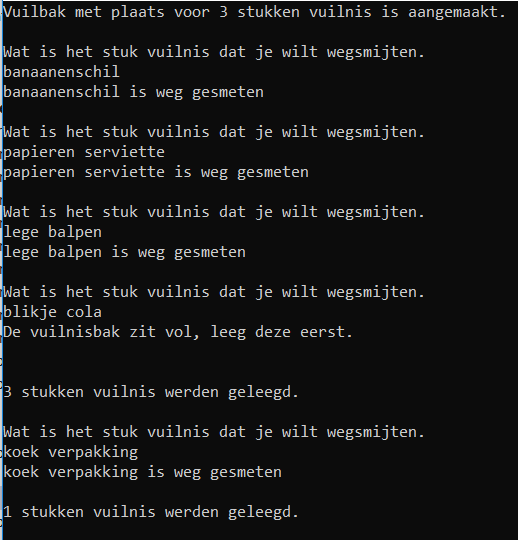
In **Opdracht09\_03Vuilbak** in de **RunExercise** method.

Maak een nieuwe instantie aan van de classe **Vuilbak**

Roep de method **WerpVuilnis** een paar keer op todat je op het punt komt dat je vuilnisbak te vol zit.

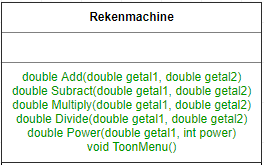
Leeg de vuilbak met **LeegVuilnis** method

### Voorbeeld



## Opdracht 9.4 Rekenmachine

### Class Diagram



**Add**return de som van getal1 en getal2

**Subtract**return het verschil van getal1 en getal2

**Multiply**return het product van getal1 en getal2

**Divide**geef een message als getal2 = 0 (je kan niet delen door nul)return Het quotient van getal1 en getal2

**Power**return getal1 tot de macht van power

**ToonMenu**Druk het menu af op het scherm

### Doel

In **Opdracht09\_04Rekenmachine** in de **RunExercise** method.

Geef de gebruiker telkens de keuze om een berekening uit de voeren. Dit doet hij door een optie te kiezen in het menu.

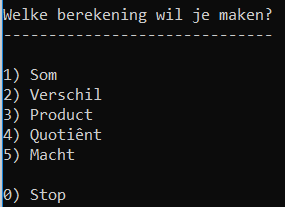
Voor elke berekening vraag je aan de gebruiker de nodige waarden om de bereking uit te voeren.

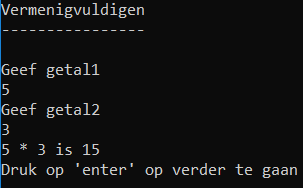
Je toont de berekening op het scherm.

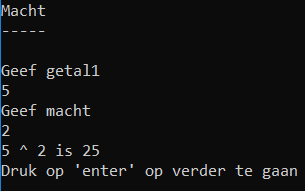
De gebruiker krijgt opnieuw het menu te zien en moet een nieuwe keuze maken.

Als de gebruiker **0** als keuze geeft stopt het programma.

### Voorbeeld







## Opdracht 9.5 Pretpark

### Classdiagram



**Volgendeklant(……)**

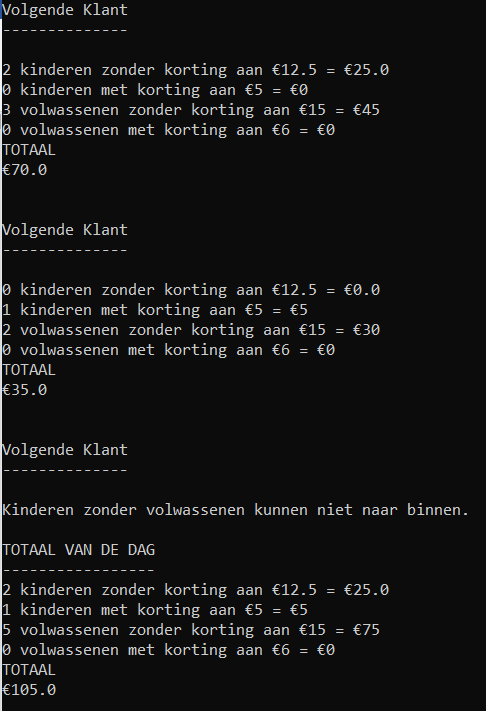
* ALS er enkel kinderen zijn
  + Print een message dat er minstens 1 volwassene moet bijzijn
* ANDERS
  + Tel aantalKinderen (parameter) op bij de property **AantalKinderenZonderKorting**
  + Tel aantalKinderenMetKorting (parameter) op bij de property **AantalKinderenMetKorting**
  + Tel aantalVolwassenen (parameter) op bij de property **AantalVolwassenenZonderKorting**
  + Tel aantalVolwassenenMetKorting (parameter) op bij de property **AantalVolwassenenMetKorting**
  + Print voor elke van de vier groepen (kinder, kinderenKorting, volwassenen, volwassenenKorting). Aantal en de totale kost voor die groep
  + Print de totale kost voor alle groepen

**PrintDag()**

Print voor elke groep het totaal aantal van die groep en de totale winst van die groep

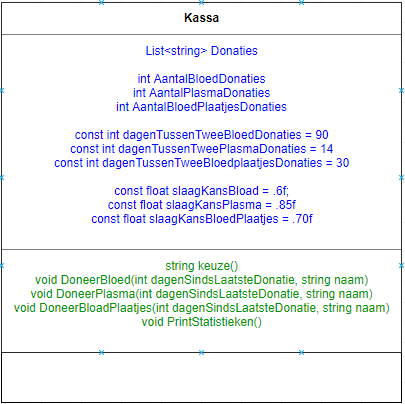
Print de totale winst voor alle groepen

### Voorbeeld



## Opdracht 9.6 RodeKruis

### ClassDiagram



**Keuze()**

Print het menu en returned de gemaakte keuze

**DoneerBloed(int dagenSindsLaatsteDonatie, string naam)  
DoneerPlasma(int dagenSindsLaatsteDonatie, string naam)  
DoneerBloedPlaatjes(int dagenSindsLaatsteDonatie, string naam)  
PrintStatistieken()**

Zie FlowChart. In solution

### Doel

In **Opdracht09\_06RodeKruis** in de **RunExercise** method.

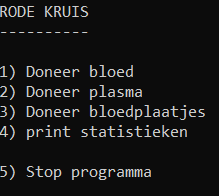
Maak een nieuwe instantie van **rode kruis** class

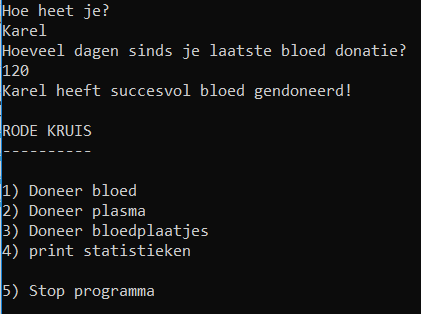
Roep de **Keuze** method op.

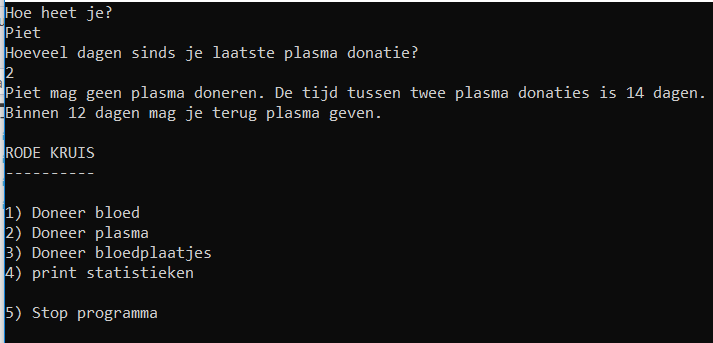
Maak gebruik van een switch om het resultaat van **Keuze** to redirecten naar de juiste method.

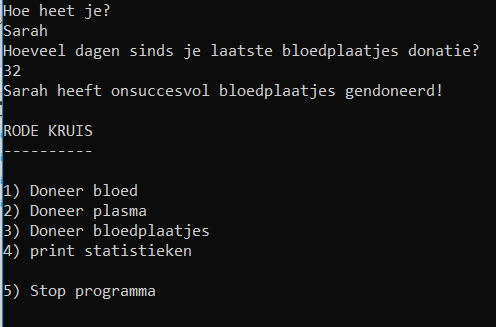
Vb. Als de Keuze **1** is dan moet er bloed gedoneerd worden.

### Voorbeeld









STATESTIEKEN

