

# 2019 oplossingen oefeningen

Met dank aan de [Github van Martijn](#)

## Bestanden:

Invoer: een bestand met 1 lijn tekst van 70 karakters, bvb.:

Het spaanse graan heeft de orkaan doorstaan...

Schrijf een programma dat telt hoeveel keer de letter 'a' voorkomt. Toon de uitvoer aan de gebruiker.

```
%include 'gt.asm'
```

```
covar
```

```
inarea: resb 70
```

```
uitvoer: resd 1
```

```
inleiding
```

```
openin
```

```
lees
```

```
cld
```

```
sub eax, eax
```

```
sub esi, esi
```

```
sub ebx, ebx
```

```
mov al, 'a'
```

```
mov ecx, 70
```

```
terug:
```

```
cmp al, [inarea + esi]
```

```
jne verder
```

```
add ebx, 1
```

verder:

add esi, 1

sub ecx, 1

cmp ecx, 0

je einde

jmp terug

einde:

mov [uitvoer], ebx

uit [uitvoer]

slot

Herschrijf de vorige oefening om alle klinkers (a,e,i,o,u) te tellen.

%include 'gt.asm'

covar

inarea: resb 70

uitvoer: resd 1

inleiding

openin

lees

cld

sub eax, eax

sub esi, esi

sub ebx, ebx

mov al, 'a'

mov ecx, 70

terug:

mov al, 'a'

cmp al, [inarea + esi]

je verder

mov al, 'e'

cmp al, [inarea + esi]

```
je verder
mov al, 'i'
cmp al, [inarea + esi]
je verder
mov al, 'o'
cmp al, [inarea + esi]
je verder
mov al, 'u'
cmp al, [inarea + esi]
je verder
jmp skip
verder:
add ebx, 1
skip:
add esi, 1
sub ecx, 1
cmp ecx, 0
je einde
jmp terug

einde:
mov [uitvoer], ebx
uit [uitvoer]

slot
```

Schrijf een programma dat een woord uitleest uit het invoerbestand. Het woord staat op de eerste lijn (die verder uit allemaal spaties bestaat). Lees dan het bestand verder uit, en tel hoe vaak het woord nog voorkomt in de rest van de tekst. Toon het resultaat aan de gebruiker. (Elke lijn bevat één woord gevolgd door spaties.)

```
%include 'gt.asm'
cvar

inarea: resb 70
uitvoer: resd 1

inleiding
```

openin

lees

cld

sub eax, eax

sub esi, esi

sub ebx, ebx

sub edx, edx

mov al, 'a'

mov ecx, 10

terug:

mov al, [inarea + edi]

cmp al, [inarea + esi]

jne verder

add ebx, 1

verder:

add esi, 1

sub ecx, 1

cmp ecx, 0

je next

jmp terug

next:

add edi, 70

sub edx, 1

cmp edx, 0

je einde

einde:

mov [uitvoer], ebx

uit [uitvoer]

slot

## Rijen:

Schrijf een programma dat 9 getallen inleest en de volgende getallen berekent en toont:

Het gemiddelde

Hoeveel getallen groter zijn dan het gemiddelde

Hoeveel getallen kleiner zijn dan het gemiddelde

```
%include "gt.asm"
```

```
covar
```

```
getal: resd 9
```

```
ding: resd 1
```

```
meer: resd 1
```

```
minder: resd 1
```

```
negen: dd 9
```

```
een: dd 1
```

```
inleiding
```

```
sub ecx, ecx
```

```
sub eax, eax
```

```
sub edi, edi
```

```
mov ecx, 9
```

```
hoger:
```

```
    cmp ecx, 0
```

```
    jle verder
```

```
    inv[getal + edi]
```

```
    add eax, [getal + edi]
```

```
    add edi, 4
```

```
    sub ecx, 1
```

```
    jmp hoger
```

```
verder:
```

```
mov [ding], eax
```

```
imul dword [een]
```

```
idiv dword [negen]
```

```
mov [ding], eax
```

```
mov ecx, 9
```

```
sub edi, edi
```

terug:

```
    mov eax, [ding]
    cmp eax, [getal + edi]
    jl nogverder
    je natel
    sub eax, eax
    add eax, 1
    add [meer], eax
    add edi, 4
    jmp na
```

nogverder:

```
    mov eax, [ding]
    cmp eax, [getal + edi]
    sub eax, eax
    add eax, 1
    add [minder], eax
```

natel:

```
    add edi, 4
```

na:

```
    loop terug
```

uit [ding]

uit [minder]

uit [meer]

slot

Schrijf een programma dat 6 getallen inleest en deze gesorteerd van klein naar groot afdruckt. Om te sorteren kan men als volgt te werk gaan:

doe 5 maal:

```
{ i = 0;
doe 5 maal
{ vergelijk element(i) met element(i+1);
  if (element(i) > element(i+1)) {
    verwissel;
  }
```

```
    i = i + 1;  
}  
}
```

```
%include "gt.asm"
```

```
covar
```

```
getal: resd 6
```

```
nul: dd 0
```

```
hulp1: resd 1
```

```
hulp2: resd 2
```

```
teller: resd 1
```

```
inleiding
```

```
sub esi, esi
```

```
sub ecx, ecx
```

```
inv [getal]
```

```
inv [getal + 4]
```

```
inv [getal + 8]
```

```
inv [getal + 12]
```

```
inv [getal + 16]
```

```
inv [getal + 20]
```

```
begin:
```

```
sub ecx, ecx
```

```
sub esi, esi
```

```
vergelijk:
```

```
cmp ecx, 5
```

```
je verder
```

```
mov eax, [getal + esi]
```

```
mov edx, [getal + esi + 4]
```

```
cmp eax, edx
```

```
jle skip
```

```
mov [getal + esi + 4], eax
```

```
mov [getal + esi], edx
```

```
skip:
```

```
add ecx, 1
add esi, 4
jmp vergelijk

verder:
sub ecx, ecx
mov ecx, [teller]
add ecx, 1
mov [teller], ecx
cmp ecx, 5
jle begin
```

```
uit [getal]
uit [getal + 4]
uit [getal + 8]
uit [getal + 12]
uit [getal + 16]
uit [getal + 20]
```

slot

## Strings in bestanden:

Schrijf een programma dat 2 getallen leest via inv en de som afdruckt in het uitvoerbestand. De getallen bestaan uit niet meer dan 5 cijfers. Als uitvoer willen we:

HET EERSTE GETAL IS: 39



HET TWEEDE GETAL IS: 40

=====

DE SOM IS: 79

%include "gt.asm"

covar

hulp: resd 1

getal1: resd 1

getal2: resd 2

inarea: resb 70

outarea: resb 70

db 13,10

vb1: dd "HET EERSTE GETAL IS:"

vb2: dd "HET TWEEDE GETAL IS:"

vb3: dd "DE SOM IS: "

spatie: dd " "

lijn: dd "=====

som: dd "DE SOM IS:"

inleiding

openuit

inv[getal1]

inv[getal2]

# EERSTE GETAL

cld

mov ecx, 70

mov al, " "

mov edi, outarea

rep stosb

mov ecx, 20

mov esi, vb1

mov edi, outarea

rep movsb

mov eax, [getal1]

```
mov edi, outarea + 27
std
mov ebx, 10
lus: mov edx, 0
idiv ebx
add dl, 30h
xchg al, dl
stosb
xchg al, dl
cmp eax, 0
jne lus
schrijf
```

```
# TWEEDE GETAL
```

```
cld
mov ecx, 70
mov al, " "
mov edi, outarea
rep stosb
```

```
mov ecx, 20
mov esi, vb2
mov edi, outarea
rep movsb
```

```
mov eax, [getal2]
```

```
mov edi, outarea + 27
std
mov ebx, 10
lus1: mov edx, 0
idiv ebx
add dl, 30h
xchg al, dl
stosb
xchg al, dl
cmp eax, 0
jne lus1
schrijf
```

# STREEP

cld

mov ecx, 70

mov al, " "

mov edi, outarea

rep stosb

cld

mov ecx, 27

mov al, "="

mov edi, outarea

rep stosb

schrijf

# SOM

cld

mov ecx, 70

mov al, " "

mov edi, outarea

rep stosb

mov ecx, 20

mov esi, vb3

mov edi, outarea

rep movsb

mov eax, [getal1]

add eax, [getal2]

mov edi, outarea + 27

std

mov ebx, 10

lus2: mov edx, 0

idiv ebx

add dl, 30h

xchg al, dl

stosb

xchg al, dl

cmp eax, 0

jne lus2

schrijf

slot

;BELASTING BEREKENEN

Invoer (meerdere lijnen):

kol 1-20: naam

kol 31-40: inkomen

Uitvoer (voor ieder invoerrecord):

kol 1-20: naam (idem als invoer)

kol 31-40: inkomen (idem als invoer)

kol 42-50: belasting

Deze belasting wordt als volgt berekent:

Als inkomen  $\leq 2500$ , dan is belasting = 0

Als inkomen  $> 2500$  en  $\leq 5000$ , dan is belasting =  $(\text{inkomen} - 2500) * 10\%$

Als inkomen  $> 5000$  en  $\leq 10000$ , dan is belasting =  $(\text{inkomen} - 5000) * 20\% + 250$

Als inkomen  $> 10000$ , dan is belasting =  $(\text{inkomen} - 10000) * 40\% + 1250$

%include "gt.asm"

covar

hulp: resd 1

inarea: resb 70

outarea: resb 70

db 13,10

spatie: dd " "

tien: dd 10

vijf: dd 5

twee: dd 2

een: dd 1

inleiding

sub esi, esi

openuit

openin

lus:

cld

mov ecx, 70

mov al, " "

mov edi, outarea

rep stosb

lees

cmp eax, 0

je einde

cld

mov ecx, 70

mov esi, inarea

mov edi, outarea

rep movsb

mov esi, inarea + 30

mov ecx, 10

tekstbin

mov [hulp], eax

uit [hulp]

imul dword [een]

cmp eax, 2500

jg next

sub eax, eax

jmp past

next:

cmp eax, 5000

jg next1

sub eax, 2500

idiv dword [tien]

jmp past

next1:

```
cmp eax, 10000
jg next2
sub eax, 5000
idiv dword [vijf]
add eax, 250
jmp past
```

```
next2
sub eax, 10000
imul eax, 2
idiv dword [vijf]
add eax, 1250
past:
```

```
mov edi, outarea + 49
std
mov ebx, 10
luss: mov edx, 0
idiv ebx
add dl, 30h
xchg al, dl
stosb
xchg al, dl
cmp eax, 0
jne luss
schrijf
```

```
jmp lus
```

```
einde:
```

slot

Maak een invoerbestand met meerdere lijnen als volgt:

kol 1-20: naam

kol 21-25: aantal dagen

kol 31-35: dagloon

1) Schrijf een programma dat de eerste record van dit invoerbestand leest en afdruckt. De uitvoer wordt dus:

kol 1-35: idem als op invoerrecord

kol 36-70: blanco

2) Wijzig uw programma zodat nu het invoerbestand record per record afgedrukt wordt.

3) Voeg volgende functie toe aan uw programma: voor ieder invoerrecord wordt nu ook het brutoloon (= aantal-dagen \* dagloon) berekend en afgedrukt in kol 42-50.

4) Voeg volgende functie toe aan uw programma: voor ieder invoerrecord wordt nu ook de afhouding (= brutoloon \* 40%) berekend en afgedrukt in kol 52-60.

5) Voeg volgende functie toe aan uw programma: nadat de gegevens van het laatste invoerrecord verwerkt (en afgedrukt) zijn, wordt er: (a) een blanco lijn gedrukt, en (b) het totaal van de kolommen brutoloon en afhouding afgedrukt.

Het is verboden de bevelen voor een volgend punt in te typen zonder dat men er zich van vergewist heeft dat de voorgaande punten perfect opgelost zijn!

```
%include "gt.asm"
```

```
covar
```

```
hulp: resd 1
```

```
hulp1: resd 1
```

```
inarea: resb 70
```

```
outarea: resb 70
```

```
    db 13,10
```

```
spatie: dd " "
```

```
tien: dd 10
```

```
vijf: dd 5
```

```
twee: dd 2
```

```
een: dd 1
```

```
totaalbruto: resd 1
```

totaalafhouding: resd 1

inleiding

sub esi, esi

openuit

openin

lus:

cld

mov ecx, 70

mov al, " "

mov edi, outarea

rep stosb

lees

cmp eax, 0

je einde

cld

mov ecx, 36

mov esi, inarea

mov edi, outarea

rep movsb

cld

mov esi, inarea + 21

mov ecx, 4

tekstbin

mov [hulp], eax

uit [hulp]

cld

mov esi, inarea + 31

mov ecx, 4

tekstbin

mov [hulp1], eax

uit [hulp1]

imul dword [hulp]



```
mov [hulp], eax
add [totaalbruto], eax
```

```
mov edi, outarea + 48
std
mov ebx, 10
luss: mov edx, 0
idiv ebx
add dl, 30h
xchg al, dl
stosb
xchg al, dl
cmp eax, 0
jne luss
```

```
mov eax, [hulp]
imul dword [een]
imul dword [twee]
idiv dword [vijf]
add [totaalafhouding], eax
```

```
mov edi, outarea + 58
std
mov ebx, 10
lus2: mov edx, 0
idiv ebx
add dl, 30h
xchg al, dl
stosb
xchg al, dl
cmp eax, 0
jne lus2
```

```
schrijf
jmp lus
einde:
```

```
cld
mov ecx, 70
mov al, " "
```

```
mov edi, outarea
```

```
rep stosb
```

```
schrijf
```

```
mov eax, [totaalbruto]
```

```
mov edi, outarea + 48
```

```
std
```

```
mov ebx, 10
```

```
lusbruto: mov edx, 0
```

```
idiv ebx
```

```
add dl, 30h
```

```
xchg al, dl
```

```
stosb
```

```
xchg al, dl
```

```
cmp eax, 0
```

```
jne lusbruto
```

```
mov eax, [totaalafhouding]
```

```
mov edi, outarea + 58
```

```
std
```

```
mov ebx, 10
```

```
lusafhouding: mov edx, 0
```

```
idiv ebx
```

```
add dl, 30h
```

```
xchg al, dl
```

```
stosb
```

```
xchg al, dl
```

```
cmp eax, 0
```

```
jne lusafhouding
```

```
schrijf
```

```
slot
```

## Strings:

Schrijf een programma dat afdrukt:

WHO IS MY TAILOR?

MY TAILOR IS CHRISTIAN DIOR

MY TAILOR IS RICH

IS MY TAILOR RICH?

```
%include "gt.asm"
```

```
covar
```

```
outarea: resb 70
```

```
db 0Dh, 0Ah
```

```
vb1: db 'WHO IS MY TAILOR?'
```

```
vb2: db 'MY TAILOR IS CHRISTIAN DIOR'
```

```
vb3: db 'MY TAILOR IS RICH'
```

```
vb4: db 'IS MY TAILOR RICH?'
```

```
spatie: db ' '
```

```
inleiding
```

```
openuit
```

```
mov ecx, 70
```

```
mov esi, spatie
```

```
mov edi, outarea
```

```
rep movsb
```

```
mov ecx, 17
```

```
mov esi, vb1
```

```
mov edi, outarea
```

```
rep movsb
```

```
schrijf
```

```
mov ecx, 70
```

```
mov esi, spatie
```

```
mov edi, outarea
```

```
rep movsb
```

```
sub esi, esi
```

```
mov ecx, 27
mov esi, vb2
mov edi, outarea
rep movsb
schrijf
```

```
mov ecx, 70
mov esi, spatie
mov edi, outarea
rep movsb
```

```
sub esi, esi
```

```
mov ecx, 17
mov esi, vb3
mov edi, outarea
rep movsb
schrijf
```

```
mov ecx, 70
mov esi, spatie
mov edi, outarea
rep movsb
```

```
sub esi, esi
```

```
mov ecx, 18
mov esi, vb4
mov edi, outarea
rep movsb
schrijf
```

```
slot
```

---

Revision #1

Created 17 June 2021 13:54:26 by Jasper G.

Updated 3 December 2021 22:13:09 by Jasper G.