1 Getallen en codes

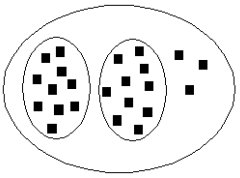
1.1 Het decimaal talstelsel

Om een aantal elementen van een verzameling weer te geven, gebruiken we een code. Die wordt gevormd door symbolen. Bijvoorbeeld :

- Het alfabet bestaat uit 26 afgesproken tekens, van A naar Z
- Voor het weergeven van numerieke waarden, gebruiken we reeksen getallen, samengesteld uit cijfers van 0 tot 9

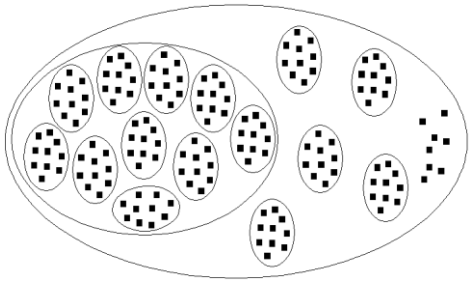
In dat laatste geval, gebruiken we het tiendelig of decimaal talstelsel. We noemen dit dus zo, omdat het stelsel 10 verschillende symbolen telt. Dat aantal verschillende symbolen, noemen we het grondtal (of *radix*) van het talstelsel. Voor het decimaal talstelsel is dat grondtal dus 10. Dat betekent dat de plaats voor één symbool 10 verschillende waarden bevatten.

Hoe gaat het dan wanneer je meer waarden wil weergeven? Daarvoor keren we even terug naar de verzamelingetjes van de lagere school.



3	2	1
0	2	3

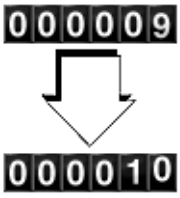
Er zitten dus **23** elementen in de verzameling

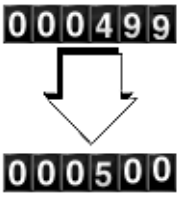


3	2	1
1	5	8

Er zitten dus **158** elementen in de verzameling

Het bepalen van het aantal elementen kan je ook vergelijken met een kilometerteller van een wagen. Wanneer je meer dan 9 kilometer gereden hebt, draait het laatste cijfertje door naar 0. Tegelijk draait het voorlaatste cijfertje ook eentje verder:





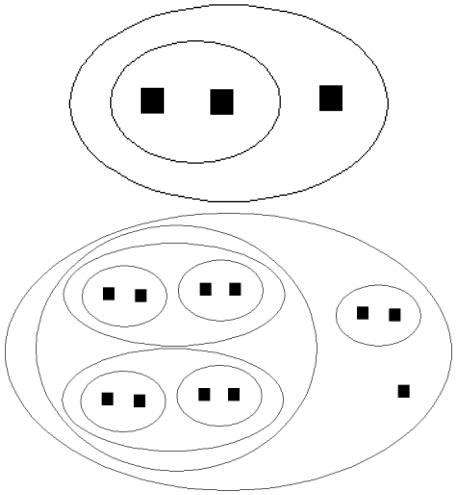
De waarde van een cijfer in een getal, wordt dus niet enkel bepaald door zijn eigen waarde, maar ook door de plaats van dat cijfer in het getal. De plaatswaarden in het decimaal talstelsel worden eenheden, tientallen, honderdtallen, duizendtallen, enz. genoemd. Niet toevallig worden die waarden gevormd door een machtsverheffing van het grondtal van het tiendelig stelsel (10), waarbij de exponent de plaatswaarde + 1 in het getal voorstelt :

Eenheden →	10^0 →	1
Tientallen →	10^1 →	10
Honderdtallen →	10^2 →	100
Duizendtallen →	10^3 →	1000
enz.		

Dit lijkt allemaal heel erg logisch voor een decimaal talstelsel, omdat we als kind al in dit stelsel hebben leren tellen. We kunnen hetzelfde principe ook toepassen in andere talstelsels, en dan wordt het heel wat makkelijker om ook daarmee te werken.

1.2 Het binair talstelsel

In het binair of tweedelig talstelsel is het grondtal 2. Het vormt de basis voor de werking van computers, waar de kleinste eenheid een bit is – een geheugenplaats om slechts 2 waarden te bewaren : 1 of 0.



4	3	2	1
0	0	1	1

4	3	2	1
1	0	1	1

In de eerste verzameling zitten 3 elementen. Binair wordt dit blijkbaar geschreven als “11”.
De 11 elementen uit de tweede verzameling worden blijkbaar geschreven als “1011”.

Een binaire kilometerteller in een auto zou telkens na een 1 verder draaien naar een 0, en de 0 op de plaats ervoor, verandert in een 1.

Het probleem is dat een binair geschreven waarde moeilijk door mensen kan gelezen worden. Je zal ze dus moeten omzetten naar een decimaal geschreven waarde. Daarvoor bestaan verschillende methoden.

Conversie van de binaire naar decimale schrijfwijze

Optellen van machten

- Neem het cijfer op de eerste plaatswaarde, en vermenigvuldig het met 2^0.
- Vermenigvuldig het cijfer op de tweede plaatswaarde met 2^1.
- Ga zo verder tot het cijfer op de laatste plaatswaarde.
- Tel de producten van alle vermenigvuldigingen met elkaar op.

1011 $= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
$= 2^3 + 0 + 2^1 + 2^0$
$= 8 + 0 + 2 + 1$
$=$ **11**

Vermenigvuldigen en tellen

- Vermenigvuldig het cijfer op de hoogste plaatswaarde met 2.
- Tel het product bij het cijfer op de eerstvolgende lagere plaatswaarde, en noteer het resultaat.
- Vermenigvuldig dat getal opnieuw met 2 en tel het bij het cijfer op de eerstvolgende lagere plaatswaarde.
- Ga op die manier verder tot je het laatste cijfer in de binaire waarde hebt bereikt. Het getal dat je laatst opgeschreven hebt, is de decimale waarde.

1	0	1	1
	2	5	**11**

Ook in de andere richting zijn er verschillende methoden :

Conversie van de decimale naar binaire schrijfwijze

Machten van 2

- Zoek de grootste macht van 2 die in het decimale getal past. Noteer evenveel puntjes als de exponent plus 1.
- Noteer een 1 op het eerste puntje.
- Trek het resultaat van de grootste macht van 2 af van het decimale getal. Noteer het verschil op een klad-blaadje.
- Indien de eerstvolgende lagere macht van 2 in de rest van de deling past, trek je het resultaat daarvan af van het verschil dat op het kladblaadje staat, en je noteert een 1 op het volgende puntje. Zoniet, noteer je een 0, en je probeert het met een lagere macht.
- Het nieuwe verschil vervangt het eerste die je op het kladblaadje had staan.
- Blijf herhalen tot je slechts een 1 of een 0 overhoudt, en die vormt dan het laatste cijfer van het binaire getal.

$11 \rightarrow 8 = 2^3 = 3+1$ puntjes	. . . .
$11 - 8 = 3$	1 . . .
$2^2 = 4 > 3$	1 0 . .
$3 - 2^1 = 3 - 2 = 1$	**1 0 1 1**

Delen door 2

- Deel het decimaal getal door 2, en noteer het gehele quotiënt links ervan.
- Noteer de rest van de deling onder het gehele quotiënt.
- Deel het gehele quotiënt opnieuw door 2 en noteer het nieuwe gehele quotiënt weer links.
- Noteer opnieuw de rest van de deling onder het nieuwe gehele quotiënt.
- Herhaal dit tot het gehele quotiënt 0 is. De onderste rij cijfers vormen de binaire waarde.

0	1	2	5	11
1	**0**	**1**	**1**	

Oefening

Binair	Decimaal	Decimaal	Binair
110	……	14	……
1001	……	29	……
10110	……	55	……
10001	……	77	……
110011	……	112	……
101101	……	199	……
1011011	……	245	……
1001111	……	300	……
11011010	……	364	……
11101110	……	525	……

Rekenen in het binair talstelsel

Het principe van optellen en aftrekken in het binair talstelsel is precies hetzelfde als in het decimaal talstelsel, alleen is het aantal symbolen dus beperkt tot 1 en 0. Enkele voorbeelden :

```
   1
  1 0 1
+ 1 0 0 1
  1 1 1 0
```

Wanneer je de laagste plaatswaarden wil optellen, tel je 1 + 1 op. Aangezien 1 de hoogste waarde is in het binair talstelsel, wordt het resultaat van die optelling 10. Je noteert de 0, en je onthoudt de 1 op de volgende plaatswaarde.

```
  1 1
 1 1 1 1
    1 1
  1 1 0
+   1 1 1
 1 0 0 0 0
```

Wanneer je meerdere binaire waarden moet optellen, wordt het iets moeilijker. Wanneer je in dit voorbeeld de cijfers op de tweede plaatswaarde optelt, krijg je 1 + 1 + 1 + 1. Het resultaat is 11. Die twee eentjes onthoud je beiden voor de volgende plaatswaarde (dus boven elkaar, en niet achter elkaar).

```
  0 0
  1 1 0
-   1 1
  0 1 1
```

Wanneer je op de laatste plaatswaarde 1 aftrekt van 0, zal er bij het cijfer op de volgende plaatswaarde eentje “geleend” worden. 10 - 1 heeft als resultaat 1. Het cijfer waarvan eentje “geleend” werd, wordt dan met één verminderd.

Oefeningen

```
  1 1 1          1 0 1 0 1 1          1 0 1 1          1
+ 1 0 1 0      + 1 1 0 1 1 1 0      + 1 0 1 1        1 1
                                                    1 1 0
                                                  + 1 0 1 0

  1 1 0 1        1 1 1 0 1          1 0 0 0 0 0      1 0 1 1 0 1 0 1
-   1 1 0      -   1 0 0 1 1      -   1 0 1 0 1    -   1 1 1 1 1 1
```

Computers werken in het binair talstelsel. De plaats waarin één eentje of één nulletje bewaard wordt, heet een bit (kort voor binary digit). In zo’n bit kan je dus slechts twee verschillende waarden bewaren: een 1 of een 0. Dat is natuurlijk weinig, en daarom worden 8 bits samengenomen tot één byte. Daarmee kan je dan 2^8 of 256 verschillende waarden maken.
Een byte vormt de basiseenheid voor geheugenopslagruimte van een computer. Moderne computers beschikken over zeer uitgebreide geheugens. Daarom worden – naar analogie met de bekende metrische stelsels – telkens met een tiende macht nieuwe eenheden benoemd :

1 Kilobyte	= 1 KB	= 2^{10} bytes	= 1 024 bytes
1 Megabyte	= 1 MB	= 2^{20} bytes	= 1 048 576 bytes
1 Gigabyte	= 1 GB	= 2^{30} bytes	= 1 073 741 824 bytes
1 Terabyte	= 1 TB	= 2^{40} bytes	= 1 099 511 627 776 bytes
1 Petabyte	= 1 PB	= 2^{50} bytes	= 1 125 899 906 842 624 bytes
1 Exabyte	= 1 EB	= 2^{60} bytes	= 1 152 921 504 606 846 976 bytes
1 Zettabyte	= 1 ZB	= 2^{70} bytes	= 1 180 591 620 717 411 303 424 bytes
1 Yottabyte	= 1 YB	= 2^{80} bytes	= 1 208 925 819 614 629 174 706 176 bytes

1.3 Het hexadecimaal talstelsel

In sommige computertoepassingen wordt ook wel gebruik gemaakt van het hexadecimale of zestiendelige talstelsel, waarbij het grondtal 16 is. Een probleem is dat wij in onze schrijftaal geen symbolen hebben voor de zes waarden boven 9. Daarvoor worden in het hexadecimaal talstelsel de letters A tot en met F gebruikt.

Kleurencodes in webpagina's

Op een computerscherm is een kleur altijd een samenstelling van de kleuren rood, groen en blauw (RGB). Elke kleur kan 256 schakeringen (van donker naar licht) hebben, zodat een combinatie van de verschillende schakeringen van de drie kleuren een palet van meer dan 16 miljoen kleuren vormt. Niet toevallig is 256 gelijk aan 16^2.
In de HTML-taal wordt elke kleur weergegeven met de hexadecimale waarde van haar schakering, en dit telkens met exact 2 plaatswaarden. De kleurcode in een HTML-pagina bestaat dus altijd uit 6 hexadecimale tekens, waarvan de eerste twee de roodschakering aangeeft, de twee volgende de groenschakering en de laatste twee de blauwschakering.

```
...
<body bgcolor="#FF207A">
<p><font color="#00D1F1">Welkom op mijn website,</p>
...
```

De conversiemethoden lopen parallel aan de conversiemethoden voor het binair talstelsel :

Conversie van de hexadecimale naar decimale schrijfwijze

Optellen van machten

- Neem het symbool op de eerste plaatswaarde, en indien het een symbool een letter van A tot F is, zet je die eerst om naar de decimale waarde ervan.
- Vermenigvuldig dat getal met 16^0.
- Vermenigvuldig het cijfer op de tweede plaatswaarde met 16^1.
- Ga zo verder tot het cijfer op de laatste plaatswaarde.
- Tel de producten van alle vermenig-vuldigingen met elkaar op.

23B7 $= 2 \times 16^3 + 3 \times 16^2 + 11 \times 16^1 + 7 \times 16^0$
$= 2 \times 4096 + 3 \times 256 + 11 \times 16 + 7 \times 1$
$= 8192 + 768 + 176 + 7$
$=$ **9143**

Vermenigvuldigen en tellen

- Vermenigvuldig het cijfer op de hoogste plaatswaarde met 16 (indien het symbool een letter is, zet je die eerst om naar de decimale waarde ervan).
- Tel het product bij het cijfer op de eerst-volgende lagere plaatswaarde, en noteer het resultaat.
- Vermenigvuldig dat getal opnieuw met 16 en tel het bij het cijfer op de eerstvolgen-de lagere plaatswaarde.
- Ga op die manier verder tot je het laatste cijfer in de binaire waarde hebt bereikt. Het getal dat je laatst opgeschreven hebt, is de decimale waarde.

2	3	B	7
	35	571	**9143**

Conversie van de decimale naar hexadecimale schrijfwijze

Machten van 16

- Zoek de grootste macht van 16 die in het decimale getal past. Noteer evenveel puntjes als de exponent plus 1.
- Deel het decimale getal door het resultaat van de grootste macht van 16, en noteer het gehele quotiënt op het eerste puntje (indien dit groter is dan 9 zet je dit eerst om naar het hexadecimale symbool).
- Trek het resultaat van de grootste macht van 16 vermenigvuldigd met het gehele quotiënt dat je had berekend, af van het decimale getal. Noteer het verschil op een kladblaadje.
- Indien de eerstvolgende lagere macht van 16 in de rest van de deling past, volg je dezelfde procedure. Zoniet, noteer je een 0, en je probeert het met een lagere macht.
- Het nieuwe verschil vervangt het eerste die je op het kladblaadje had staan.
- Blijf herhalen tot je een waarde lager of gelijk aan F hebt genoteerd, en die vormt dan het laatste cijfer van het hexadecimale getal.

9143 > 4096 = 16^3	
9143 / 4096 = **2** ; rest = 951	2 . . .
951 > 256 = 16^2	
951 / 256 = **3** ; rest = 183	2 3 . .
183 > 16 = 16^1	
183 / 16 = 11 = **B** ; rest = 7	2 3 B .
7 <= F	**2 3 B 7**

Delen door 16

- Deel het decimaal getal door 16, en noteer het gehele quotiënt links ervan.
- Noteer de rest van de deling onder het gehele quotiënt (indien dit groter is dan 9 zet je dit eerst om naar het hexadecimale symbool).
- Deel het quotiënt opnieuw door 16 en noteer het nieuwe quotiënt weer links.
- Noteer opnieuw de rest van de deling onder het nieuwe quotiënt.
- Herhaal dit tot het gehele quotiënt niet meer deelbaar is door 16. Het laatste gehele qoutiënt noteer je voor de laatst bekomen rest.
- Het onderste getal dat je nu krijgt, vormt het hexadecimale getal.

0	2	35	571	9143
2	**3**	**B**	**7**	

Oefening

Hexadecimaal	Decimaal
5F	……
2BD	……
A23	……
3799	……
C4A5	……

Decimaal	Hexadecimaal
175	……
287	……
1855	……
6799	……
25252	……

Overzicht

Binair	Decimaal	Hexadecimaal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

Rekenen in het hexadecimaal talstelsel

Het principe van optellen en aftrekken in het hexadecimaal talstelsel is precies hetzelfde als in het decimaal talstelsel, alleen zijn er nu 16 verschillende symbolen. Dat betekent dat de overdracht van eenheden slechts gebeurt bij een getal groter dan F. Enkele voorbeelden :

```
  1
  3 A
+ C 9
  1 0 3
```

Op de hoogste plaatswaarde vind je een A – dat is in het decimaal talstelsel de waarde 10 (zie tabel op de vorige bladzijde). Wanneer je 10 en 9 optelt, is het resultaat 19. In het hexadecimaal talstelsel zijn er echter maximaal 16 waarden. We trekken dus 16 af van 19, en houden 3 over. Er wordt er eentje overgedragen naar de volgende plaatswaarde.

```
  C
  ~~D~~ 7
- 3 C
  9 B
```

Wanneer men van het getal op de hogere plaatswaarde “leent”, is dat geen 10 maar wel 16.

Oefeningen

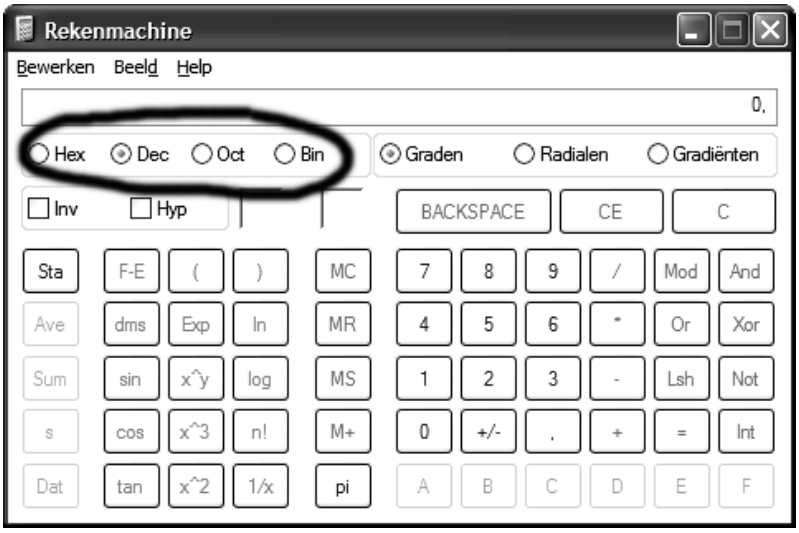
		1 A	8 6 E
5 7 D	6 0 A 4	2 B D	9 1 7
+ 2 F 4 5	+ 4 C B 5	+ 3 C E F	+ 5 D A C
5 D 1	7 A 2 6	1 2 3 4	2 8 C 4
- 1 2 3	- D 5 1	- C B A	- 1 9 D 5

Machtsverheffingen

Exponent	2	10	16
0	1	1	1
1	2	10	16
2	4	100	256
3	8	1 000	4 096
4	16	10 000	65 536
5	32	100 000	1 048 576
6	64	1 000 000	16 777 216
7	128	10 000 000	268 435 456
8	256	100 000 000	4 294 967 296
9	512	1 000 000 000	68 719 476 736
10	1024	10 000 000 000	1 099 511 627 776

Hoewel je zonder hulpmiddelen getallen vanuit het ene talstelsel naar het andere moet kunnen omzetten, vind je in de computer uiteraard een handig hulpmiddel : de rekenmachine van Windows.

Kies via het menu “Beeld” voor de wetenschappelijke interface, en typ een decimaal getal in. Door de selectierondjes Hex of Bin (Oct staat voor het Octaal of achtdelig talstelsel) aan te vinken, bekom je respectievelijk de hexadecimale of de binaire waarde van het decimale getal. En uiteraard werkt dat ook in de andere richting.



Aanduiding van het talstelsel

Wanneer getallen uit verschillende talstelsels door elkaar worden gebruikt, is het noodzakelijk om aan te geven in welk talstelsel een bepaald getal werd geschreven. Immers, het getal 1000 heeft in het binair talstelsel een heel andere waarde dan hetzelfde getal in het decimaal of het hexadecimaal talstelsel.

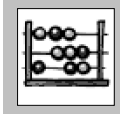
Om expliciet aan te duiden om welk talstelsel het gaat, wordt een specaal suffix achter het getal geschreven :

$$1111101000_2 = 1000_{10} = 3E8_{16}$$

Om de leesbaarheid te verhogen, wordt soms ook gebruik gemaakt van haakjes :

$$(1111101000)_2 = (1000)_{10} = (3E8)_{16}$$

1.4 De EBCDIC-code



Afkorting voor "*Extended Binary Coded Decimal Interchange Code*".

Dit is de alfanumerieke code die gebruikt werd op de IBM-mainframe computers. Het was een 8-bits code, wat betekent dat er 2^8 of 256 verschillende tekens konden gevormd worden. Dat was meer dan strikt nodig, en daarom werden er een aantal niet gebruikt.

				0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	b7
				0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	b6
				0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	b5
b3	b2	b1	b0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	b4
0	0	0	0	NUL	DLE	DS		SP	&	-						{	}		0	
0	0	0	1	SOH	DC1	SOS				/		a	j	~		A	J		1	
0	0	1	0	STX	DC2	FS	SYN					b	k	s		B	K	S	2	
0	0	1	1	ETX	DC3							c	l	t		C	L	T	3	
0	1	0	0	PF	RES	BYP	PN					d	m	u		D	M	U	4	
0	1	0	1	HT	NL	LF	RS					e	n	v		E	N	V	5	
0	1	1	0	LC	BS	ETB	UC					f	o	w		F	O	W	6	
0	1	1	1	DEL	IL	PRE	EOT					g	p	x		G	P	X	7	
1	0	0	0		CAN							h	q	y		H	Q	Y	8	
1	0	0	1	RLF	EM						\	i	r	z		I	R	Z	9	
1	0	1	0	SMM	CC	SM		¢	!	□	:									
1	0	1	1	VT				.	$	,	#									
1	1	0	0	FF	IFS		DC4	<	*	%	@									
1	1	0	1	CR	IGS	ENQ	NAK	(	)		'									
1	1	1	0	SO	IRS	ACK		+	;	>	=									
1	1	1	1	SI	IUS	BEL	SUB	|	¬	?	"									

De eerste 64 codes werden gebruikt voor speciale toepassingen, zoals schuifbewerkingen, codes voor datatransmissie, … enz.
Daarna had de spatie de laagste waarde, gevolgd door een aantal speciale tekens.
Hierna kwamen eerst de kleine letters, en dan de hoofdletters. De hoogste waarden in de EBCDIC-tabel werden ingenomen door de cijfers.

http://www.legacyj.com/cobol/ebcdic.html

1.5 De ASCII-code

Afkorting voor : "*American Standards Code of Information Interchange*"

Dit is een 7-bits alfanumerieke code, die tegenwoordig door het allergrootste deel van de computers gehanteerd wordt. Met deze code kunnen slechts 128 verschillende lettertekens worden gevormd, maar dat bleek genoeg voor alle tekens uit het standaard Engels en de andere symbolen die op een toetsenbord aanwezig zijn.

Dec	Hex	Sym	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	NUL	32	20		64	40	@	96	60	`
1	1	SOH	33	21	!	65	41	A	97	61	a
2	2	STX	34	22	"	66	42	B	98	62	b
3	3	ETX	35	23	#	67	43	C	99	63	c
4	4	EOT	36	24	$	68	44	D	100	64	d
5	5	ENQ	37	25	%	69	45	E	101	65	e
6	6	ACK	38	26	&	70	46	F	102	66	f
7	7	BEL	39	27	'	71	47	G	103	67	g
8	8	BS	40	28	(	72	48	H	104	68	h
9	9	TAB	41	29	)	73	49	I	105	69	i
10	A	LF	42	2A	*	74	4A	J	106	6A	j
11	B	VT	43	2B	+	75	4B	K	107	6B	k
12	C	FF	44	2C	,	76	4C	L	108	6C	l
13	D	CR	45	2D	-	77	4D	M	109	6D	m
14	E	SO	46	2E	.	78	4E	N	110	6E	n
15	F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	□

De eerste 32 tekens van standaard ASCII zijn **stuurtekens** (bv.: 27=ESC, 9=TAB, …).

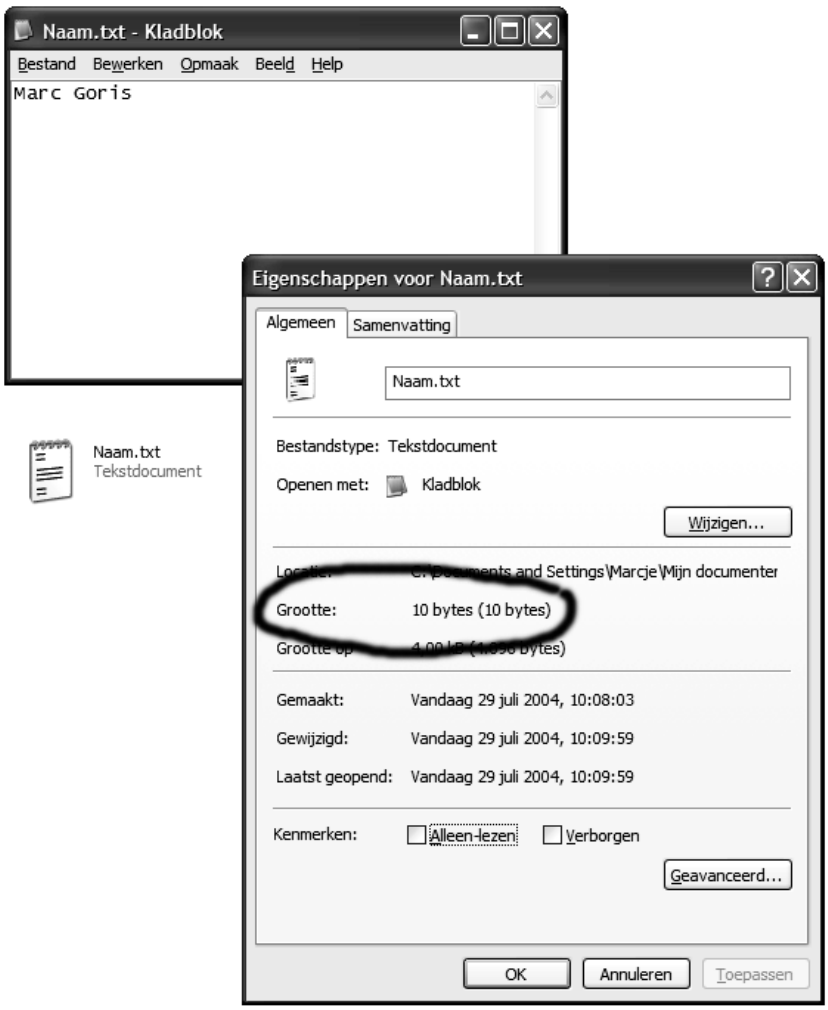
Het probleem is echter dat andere talen van andere tekens gebruik maken dan voorzien in de ASCII-tabel. Daarom werden 8-bits ASCII-tabellen gemaakt, die men dan ***Extended ASCII*** noemt, of bij PC's **ANSI-code** (*American National Standards Institute*). Helaas bestaat er geen algemene standaard. In een Windows-omgeving verschillen de 128 laatste tekens zelfs per lettertype. Dat maakt de uitwisseling van gegevens die in de verschillende codes zijn opgemaakt, soms wat moeilijk.

http://www.asciitable.com
http://www.ansi.org

Wanneer je een onopgemaakte tekst bewaart, zal voor elk teken precies één byte worden gebruikt. Probeer het volgende maar eens uit :

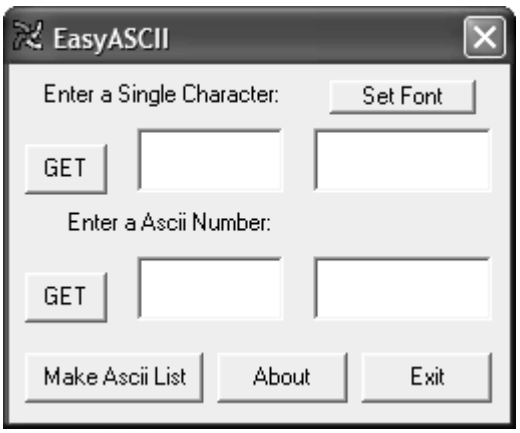
- Open een leeg document in Kladblok.
- Typ je naam in. Tel alle tekens (een spatie is ook een teken!)
- Bewaar het document met als naam “Naam.txt”.
- Klik met de rechtermuisknop op het bestand, en kies “Eigenschappen” uit het menu.

Je zal merken dat het bestand net evenveel bytes telt als het aantal tekens dat het bevat.



Uiteraard werkt dit alleen maar met onopgemaakte tekst. Een tekst die je bijvoorbeeld in MS Word intypt, zal meer bytes beslaan dan het aantal tekens die erin staan. De informatie van de opmaakcodes (lettertype, kleuren, lijnen, … enz.) wordt immers eveneens mee bewaard, en dat vraagt natuurlijk ook opslagruimte.

Hulpmiddelen



Op het internet zijn allerlei hulpprogramma’tjes te vinden om informatie uit de ASCII-code te bekijken, meestal is dat freeware (gratis software, die vrij mag verspreid worden). Een voorbeeld daarvan is het programma EasyASCII van Breezin Software (www.breezin.net), waarin je de plaats van een karakter in de ASCII-code kan nagaan, of een karakter op een bepaalde plaats in de ASCII-code kan bekijken.

Maar er bestaat nog een ander truukje : in elk Windows-gebaseerd tekstverwerkingsprogramma kan je rechtstreeks een karakter uit de ASCII-code ingeven, door de ALT-toets ingedrukt te houden, en dan de decimale waarde van de ASCII-code in te typen. Wanneer je de ALT-toets weer loslaat, verschijnt het overeenkomstige karakter. (vb. ALT+80 = “P”). Probeer maar eens uit in MS Word…

Wanneer tekstgegevens gesorteerd worden, gebeurt dit altijd op basis van de volgorde van de tekens in de ASCII-code.

- “F” < “T” want de ASCII-code van het teken “F” is 70 en die van het teken “T” is 84.
- “D” < “d” want de ASCII-code van het teken “D” is 68 en die van het teken “d” is 100.
- “2” < “6” want de ASCII-code van het teken “2” is 50 en die van het teken “6” is 54.
- “2” < “D” want de ASCII-code van het teken “2” is 50 en die van het teken “D” is 68.

Wanneer het eerste teken hetzelfde is, zal gesorteerd worden op het eerstvolgende teken dat verschillend is :

- “Jan” < “Jef” omdat het eerste teken hetzelfde is, en de ASCII-code van het teken “a” 97 is, en dat van het teken “e” 101.
- “Schrik” < “Schroom” omdat de eerste vier tekens dezelfde zijn, en de ASCII-code van het teken “i” 105 is, en dat van het teken “o” 111.
- “365” < “39” omdat het eerste teken hetzelfde is, en de ASCII-code van het teken “6” 54 is, en dat van het teken “9” 57.

Wanneer gegevens strikt volgens de ASCII-code worden gesorteerd, betekent dit :

- dat gegevens die beginnen met een hoofdletter, altijd hoger zullen staan dan gegevens die beginnen met kleine letters ;
- dat eenzelfde tekst mét een spatie erin altijd hoger zal staan dan die tekst zonder een spatie ;
- Dat cijfers die als tekstgegeven worden bewaard, niet volgens hun getalwaarde worden gesorteerd.

Softwaretoepassingen waarin gegevens moeten gesorteerd worden, zullen meestal een bepaalde correctie op het ASCII-sorteringssysteem uitvoeren : hoofdletters en kleine letters worden daarin gelijkgeschakeld, in vele gevallen worden spaties (en soms zelfs andere tekens) genegeerd bij de sortering, en door sommige toepassingen worden cijfers in een tekstgegeven automatisch omgezet naar hun getalwaarde.

Merk trouwens op dat de sorteervolgorde van de ASCII-code volledig verschillend is van die van de EBCDIC-code, waar de kleine letters een lagere waarde hebben dan de hoofdletters …

1.6 De Unicode

De Extended ASCII-code heeft als groot nadeel dat een bestand opgeslagen in één codetabel, er onder een andere gedaante uitkomt bij gebruik van een andere codetabel. Daarom ontstond de behoefte om een codering te ontwikkelen die àlle tekens in één keer zou bevatten. Dan moet zo'n code wel meer dan 256 tekens bevatten, en dat kan alleen maar wanneer er meer bits per karakter worden gebruikt dan bij de ASCII-code.

Op dit moment ondersteunt de huidige Unicode-standaard meer dan 100.000 verschillende gecodeerde tekens uit tientallen verschillende talen, en daar komen regelmatig nog nieuwe tekens bij. Daardoor kunnen zowat alle geschreven talen ter wereld gecodeerd worden.

Om dat mogelijk te maken gebruikte Unicode oorspronkelijk 16 bits i.p.v. 8 bits (UTF-16). Daarmee konden dus 2^{16} ofwel 65 536 verschillende tekens gecodeerd worden, wat al gauw te weinig bleek. Daarom werd Unicode uitgebreid tot een 32-bits code (UTF-32), waardoor het in theorie mogelijk wordt om meer dan 4 miljard tekens te coderen.

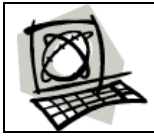
Het gevolg daarvan is uiteraard dat bestanden die bewaard worden in de Unicode veel groter zijn dan diezelfde bestanden opgeslagen in de ASCII-code ; de opslagruimte die nodig is voor één karakter is in de Unicode immers dubbel tot vier keer zo groot (2 of 4 bytes per karakter) dan in de ASCII-code (1 byte per karakter).

De Unicode-standaard is overgenomen door toonaangevende bedrijven als Apple, HP, IBM, JustSystem, Microsoft, Oracle, SAP, Sun, Sybase, Unisys en vele andere. Unicode wordt vereist door moderne standaards als XML, Java, JavaScript, WML. Unicode wordt ook ondersteund door een groot aantal besturingssystemen en alle moderne browsers. De komst van de Unicode-standaard en de beschikbaarheid van programma's die deze standaard ondersteunen vormt een van de belangrijkste recente ontwikkelingen in de globalisering van softwaretechnologie.

Unicode zorgt er dus voor dat één softwareproduct of één website zonder verdere aanpassingen kan worden gebruikt op meerdere platforms, in meerdere talen en landen. Bestanden kunnen worden uitgewisseld tussen verschillende systemen zonder dat hierbij gegevens verloren gaan of veranderen.

Het Unicode Consortium

Het Unicode Consortium is een non-profitorganisatie die is opgericht om de Unicode-standaard te ontwikkelen en uit te breiden en om het gebruik ervan te stimuleren. De leden van dit consortium bestaan uit een breed spectrum van bedrijven en organisaties uit de computerindustrie en de IT-wereld. Het consortium wordt financieel alleen ondersteund door de contributie van de leden. Het lidmaatschap van het Unicode Consortium staat open voor organisaties en personen uit de hele wereld die de Unicode-standaard steunen en willen helpen bij de uitbreiding en implementatie ervan.



http://www.unicode.org

1.7 Getalvoorstellingen

Getallen kunnen worden bewaard als alfanumeriek gegeven (een tekstgegeven waarmee niet gerekend wordt) of als numeriek gegeven (een getalwaarde waarmee wel gerekend kan worden). We hebben reeds geleerd dat dit verschil een belangrijke invloed heeft op de sorteervolgorde waarmee gegevens gesorteerd worden.

Getallen worden bij voorkeur bewaard als numerieke gegevens, niet alleen omdat er dan mee kan gerekend worden, maar ook omdat dit een belangrijke plaatsbesparing betekent. Nemen we als voorbeeld het getal 25000 :

- Alfanumerieke voorstelling in de ASCII-code:
  0011 0010 0011 0101 0011 0000 0011 0000 0011 0000
- Numerieke voorstelling in binaire code: 110 0001 1010 1000

De numerieke voorstelling (slechts 15 bits) neemt dus minder plaats in beslag dan de alfanumerieke (40 bits). Vandaar dat we getallen in het interne geheugen het best kunnen bewaren als numerieke gegevens.

Een woord

Men reserveert steeds een vast aantal bytes om de numerieke gegevens voor te stellen, afhankelijk van het gegevenstype dat door de programmeur van een toepassing gekozen wordt. Nemen we bijvoorbeeld de gegevenstypes die we kennen uit MS Access. Indien we kiezen voor een Lange integer, dan weten we dat dit type per waarde een geheugenruimte van 4 bytes (32 bits) in beslag neemt. Voor elk getal, klein of groot, worden 4 bytes gereserveerd. Het getal 1 wordt dan als volgt voorgesteld:

0000 0000 0000 0000 0000 0000 0000 0001

Op het eerste gezicht lijkt dit plaatsverlies (alfanumeriek zijn voor het getal 1 slechts 8 bits nodig) maar van zodra er gewerkt wordt met grote getallen betekent dit systeem uiteraard een grote plaatswinst.

Dit vast aantal bytes dat wordt gereserveerd, noemt men een ***woord***. Men spreekt van woorden van 4 bytes, van woorden van 8 bytes, enz. Bij woorden van 4 bytes (= 32 bits) is het hoogste gehele getal dat op deze wijze kan worden voorgesteld in principe 2^{32} ofwel 4 294 967 296. Maar omdat het eerste bit doorgaans gereserveerd wordt voor het tekenbit (positief of negatief getal), is het grootste getal in een woord van 4 bytes beperkt tot 2^{31} of 2 147 483 647.

Negatieve getallen

In de informatica worden de gehele getallen vanaf nul, positieve gehele getallen genoemd (0 wordt dus als een positief getal beschouwd). Deze definitie komt overeen met die van de verzameling $\mathbb{N}$ van de natuurlijke getallen in de wiskunde.

Negatieve getallen worden in hun decimale schrijfwijze voorafgegaan door een minteken, positieve getallen door een plusteken. Omdat we meestal met positieve getallen werken, hebben we de gewoonte aangenomen het plusteken automatisch weg te laten in positieve getallen.
Computers werken echter met bits, en die kunnen slechts twee waarden bevatten : 0 en 1. Daarom werden er afspraken gemaakt om in bijvoorbeeld geheugenregisters van computers, de eerste bit te gebruiken om aan te duiden of het om een positief of een negatief getal gaat:

0 = positief getal
1 = negatief getal

Die eerste bit wordt in dat geval het **tekenbit** genoemd.

Werken met kommagetallen : fixed en floating point

Huidige computers werken met geheugenregisters van 32 bits. Dat wil zeggen dat men slechts over 32 plaatsjes voor eentjes en nulletjes beschikt om een getal te bewaren. Wanneer men een vaste afspraak zou maken over de plaats waar de komma zou moeten komen (bijvoorbeeld vastgelegd op de 24^{ste} bit), dan beperkt dat het getalbereik dat kan worden gekozen, zowel wat betreft de grootte van het gehele getal, als van het decimale gedeelte. Die manier van werken noemt men ***fixed point***.

Wanneer door een bewerking een getal wordt berekend, dat groter is dan de waarden die mogelijk zijn binnen de grenzen van het fixed point-getal, wordt dit de fixed point overflow genoemd. Je zou verwachten dat een computer dan een foutmelding genereert, maar dat gebeurt niet automatisch (uiteraard wel als de programmeur van een toepassing in dat geval zelf een foutmelding voorziet). De waarde die bewaard wordt zal echter niet correct zijn : immers de hoogste bit(s) zullen niet worden bewaard.

Omdat het werken met een fixed point getal zo weinig efficiënt is, wordt in de informatica meestal met een ***floating point*** gewerkt. Stel dat ik het getal 158,65 wil bewaren in de computer. De computer zal van het kommagetal een geheel getal maken, door de komma zoveel plaatsen op te schuiven als er cijfers na de komma staan – dat wil dus zeggen : 2 plaatsen. Dat getal noemen we in de floating point-weergave de ***exponent***. Het oorspronkelijke getal werd 15 865. Dat getal noemen we de ***mantisse***.

Om nu het oorspronkelijke getal terug te krijgen, delen we de mantisse door het grondtal van het talstelsel (in dit geval 10, want het gaat om een decimaal getal) tot de exponent. In ons voorbeeld : 15 865 / 10^2 ofwel 15 865 / 100 = 158,65. Bij deze manier om een kommagetal in de computer op te bouwen, worden twee waarden onthouden : de mantisse en de exponent. Daarmee wordt het getalbereik dat kan bewaard worden veel groter.

Computers werken echter niet met het decimaal, maar met het binair talstelsel. Hetzelfde principe blijft van kracht. Rest er enkel nog een vaste afspraak over de plaatsen in het 32-bit adres waar de verschillende delen (mantisse en exponent) van het floating point-getal worden geschreven. Tegenwoordig houdt men zich in de computerwereld aan de ***IEEE 754 standaard***.(IEEE staat voor *Institute of Electrical and Electronics Engineers*, op z'n Engels uitgesproken *I-triple-E*). Daarbij is het volgende bepaald :

- ✓ De eerste bit (0) : ***tekenbit***
- ✓ De volgende 8 bits (1-8) : ***exponent***
- ✓ De resterende 23 bits (9-32) : ***mantisse***

Tegenwoordig wordt het floating point principe ook toegepast op 64-bits codes. Een floating point getal van 32 bits (4 bytes) wordt daarom tegenwoordig een ***short floating point*** (ook : single precision) genoemd ; een floating point getal van 64 bits (8 bytes) wordt dan een ***long floating point*** (ook : double precision) genoemd.

1.8 De pariteitscontrole

In de tijd dat men nog diende te programmeren in machinetaal (eentjes en nulletjes), kon een kleine vergissing tussen een eentje en een nulletje verstrekkende gevolgen hebben. Het was dus absoluut noodzakelijk dat een controle bestond op eventuele fouten.
De pariteitscontrole is de meest eenvoudig en meest gebruikte manier om eventuele fouten op te sporen. Voor een reeks bits werd een extra bit (pariteitsbit) geplaatst. Is het aantal enen in de rij bits even, dan moet de pariteitsbit de waarde 0 bevatten, is het aantal enen oneven, dan krijgt de pariteitsbit de waarde 1.

0 0110101
1 0010110

Wanneer een foutje werd gemaakt (een eentje was per ongeluk ingegeven als een nulletje), dan klopt de pariteitsbit niet meer en kon de computer een signaal geven dat er een fout gemaakt was.

Er bestaan twee vormen van pariteitscontrole : even en oneven pariteitscontrole. Bij een even pariteitscontole krijgt de pariteitsbit de waarde 0 indien de bitreeks een even aantal eentjes heeft, terwijl bij een oneven pariteitscontrole een waarde 1 oplevert in hetzelfde geval. Bij datacommunicatie is het uiteraard belangrijk dat vooraf tussen zender en ontvanger afgesproken wordt welke vorm van pariteitscontrole er wordt gehanteerd.

De controle van de pariteit van één bitreeks wordt ook ***Verticale Pariteitscontrole*** (in het Engels : ***Vertical Redundancy Check*** of kortweg ***VRC***) genoemd, en wordt vaak toegepast bij seriële verzending van gegevens.

Bloksomcontrole

Er is natuurlijk wel één groot nadeel : wanneer in één reeks een even aantal fouten zitten, dan wordt de bitreeks toch als correct aanzien. Om dat nadeel op te vangen, werd de bloksomcontrole (block sum check) in het leven geroepen. De overdracht van bytes gebeurt immers vaak in blokken. Door nu niet alleen horizontaal in de byte zelf, maar ook verticaal de bits in hun positie in de bytes te controleren, ontstaat een dubbele controle.

1 1 0 0 1 0 1 0
0 0 0 1 0 1 1 1
1 1 0 1 1 0 1 1
0 0 0 0 0 1 1 0
0 0 0 1 0 1 1 1
0 0 1 1 0 1 1 0

0 0 1 0 0 0 0 1

De bloksomcontrole wordt ook wel ***Horizontale Pariteitscontrole*** (in het Engels : ***Longitudinal Redundancy Check*** of kortweg ***LRC***) genoemd.

Maar ook deze controle is niet waterdicht. Wanneer tegelijk horizontaal en verticaal eenzelfde even aantal fouten voorkomen (dus zowel horizontaal als verticaal 2 fouten, ofwel zowel horizontaal als verticaal 4 fouten), wordt het blok toch nog als correct aanzien.

Cyclische pariteitscontrole

De meest betrouwbare methode van foutenbeheersing is de cyclische pariteitscontrole (in het Engels : ***Cyclic Redundancy Check*** of ***CRC***). Bij CRC-controle wordt een bericht beschouwd als één stroom van bits die één na één (cyclisch) worden verstuurd, en worden aangeboden aan een vooraf afgesproken CRC-generator. In deze generator wordt op de reeks bits een een deling door een afgesproken getal uitgevoerd. Het controlegetal (de restwaarde) dat op die manier ontstaat, wordt aan het einde van het bericht meegestuurd.

Aan de ontvangende zijde worden de binnenstromende bits eveneens door dit afgesproken getal gedeeld. De restwaarde van deze deling wordt dan vergeleken met het meegezonden controlegetal.

Een voorbeeld : stel dat het getal 1498 moet verzonden worden, en de CRC-generator gebruikt 73 als deeltal. Het gehele resultaat van 1498 gedeeld door 73 is 20, en de rest is 38. Die rest wordt door de zender als controlegetal meegestuurd, uiteraard samen met het te verzenden getal.
De ontvanger gaat nu met hetzelfde deeltal (73) de ontvangen waarde delen en de rest berekenen. Indien de restwaarde die de ontvanger zelf berekend heeft, overeenkomt met het meegestuurde controlegetal (38), dan wordt de communicatie tussen zender en ontvanger als foutloos beschouwd.

CRC is de meest zekere van alle methoden van pariteitscontrole. Deze foutcontrole heeft bovendien het bijkomende voordeel dat niet voor elk karakter een aparte pariteitsbit nodig is, waardoor de ‘overhead’ aan bits ten behoeve van de foutcontrole beduidend kleiner wordt.

http://docweb.khk.be/digitech/digitech.html

Praktische wenken bij de cursus

1. Doel van de cursus en doelpubliek

Dit is het eerste hoofdstuk van een uitgebreide cursus computertechniek, bruikbaar binnen de 3de graad TSO Informaticabeheer, sommige informatica-opleidingen in het hoger onderwijs of in het volwassenenonderwijs.

Ik heb voor het publiceren de ganse cursus dubbel nagekeken, maar het blijft natuurlijk mogelijk dat er hier en daar nog kleine foutjes instaan. Gelieve mij daarvoor dan te verontschuldigen. U mag ze mij altijd signaleren, dan kan ik ze verbeteren.

2. Gebruiksrecht

Deze cursus kon u gratis downloaden van het internet. Mijn idee is: ik heb er zelf redelijk wat tijd ingestoken, en voor mij mogen die inspanningen gerust ook elders lonen. Ik hoef hiervoor geen vergoeding (mocht u mij toch met een attentie willen bedenken, dan hou ik u niet tegen, en mailt u me maar voor de modaliteiten). Wel wil ik u vriendelijk verzoeken om, als u de cursus daadwerkelijk gaat gebruiken, mij een mailtje te sturen om mij te laten weten aan welk publiek en in welk kader u mijn cursus gebruikt. Zo kan ik mijn statistieken up-to-date houden. Bovendien mag u mij gerust uw bemerkingen of kritieken op deze cursus doormailen. Ik sta altijd open voor positief geformuleerde suggesties, en kan op die manier ook de kwaliteit van deze cursus verbeteren.

U bent vrij om gratis van mijn cursus gebruik te maken, maar dat betekent niet dat ik daarmee afstand doe van mijn auteursrecht. Het vermenigvuldigen van de cursus kan, op voorwaarde dat het enkel om educatieve redenen - en zeker niet met commerciële bedoelingen - gebeurt, en onder deze voorwaarden:

- dat u de cursus, of een deel daarvan, ongewijzigd gebruikt
- dat u de bron vermeld laat (die staat zowiezo onderaan elke bladzijde)
- **dat u mij via e-mail laat weten voor wie en in welk kader u de cursus gebruikt (zie hierboven)**

De cursus werd zo opgesteld dat ze best recto-verso wordt afgedrukt.

Mocht u deze cursus op een andere manier dan via het internet onder ogen krijgen, weet dan dat de volledige cursus in PDF-formaat te downloaden is via de website **www.marcgoris.be**.

Marc Goris
marc@sitebuilder.be
www.marcgoris.be

7 juli 2008