Qbasic - deel 2

Table of Contents

Herhaling qbasic deel 1

Bewerkingen

Algebraïsche bewerkingen

bewerking	betekenis	voorbeeld
+	optelling 2 getallen concatenatie 2 strings	9 + 2 → 11 "hal" + "lo" → "hallo"
-	aftrekking 2 getallen	9 - 2 → 7
*	vermenigvuldiging 2 getallen	9 * 2 → 18
/	deling 2 getallen	9 / 2 → 4.5
^	machtsverheffing 2 getallen	9 ^ 2 → 81
\	gehele deling 2 getallen	9 \ 2 → 4
MOD	modulus deling 2 getallen	9 MOD 2 → 1

Relationele bewerkingen

bewerking	betekenis	voorbeeld
=	is gelijk aan	9 = 2 → F
<	is kleiner dan	9 < 2 → F
>	is groter dan	9 > 2 → T
<>	is ongelijk aan	9 <> 2 → T
<=	is kleiner dan of gelijk aan	9 <= 2 → F
>=	is groter dan of gelijk aan	9 >= 2 → T

T = true = waar F = false = onwaar

Logische bewerkingen

	NOT	AND	OR	XOR	EQV	IMP
expressie 1	T F	T T F F	T T F F	T T F F	T T F F	T T F F
expressie 2		T F T F	T F T F	T F T F	T F T F	T F T F
resultaat	F T	T F F F	T T T F	F T T F	T F F T	T F T T

T = true = waar F = false = onwaar

Volgorde van bewerkingen

bewerkingen	opmerkingen
()	van binnen naar buiten
^	
* / \ MOD	van links naar rechts
+ -	van links naar rechts
NOT	
AND	
OR XOR EQV IMP	van links naar rechts

Data types

type	lengte	suffix	bereik
integer	16 bits (incl. teken)	%	-32,768 .. 32,767
long	32 bits (incl. teken)	&	-2,147,483,648 .. 2,147,483,647
single	32 bits drijvende komma	!	positief: 2.80-45 .. 3.40+38 negatief: -3.40+38 .. -2.28-45
double	64 bits drijvende komma	#	positief: 4.94-324 .. 1.80+308 negatief: -1.80+308 .. -4.94-324
string * n%	vaste lengte van n% bytes		0 .. 32,767 characters
string	variable lengte	$	0 .. 32,767 characters

Syntax conventie

De hier gebruikte syntax is de de syntax conventie die QBASIC aanhoudt voor de beschrijving van de functies en de statements.

HOOFDLETTERS dit zijn sleutelwoorden van QBASIC en zijn zo verplicht in te geven (behalve als ze tussen vierkante haken staan).
QBASIC vertaalt de sleutelwoorden automatisch naar hoofdletters.

kleine letters dit zijn de variabelen die nodig zijn voor de instructie. Het type van de variabele wordt vermeld door de suffix.

[optie] opties staan tussen vierkante haken en zijn dus niet verplicht te gebruiken.

{keuze1 | keuze2} accolades en een verticaal streepje duiden een verplichte keuze tussen 2 of meer items aan (behalve als dit tussen vierkante haken staat).

item, item, ... de 3 puntjes duiden aan dat meerdere items kunnen gebruikt worden.

Voorbeeld

```
DO [{WHILE | UNTIL} condition]
  [statementblock]
LOOP
```

- 'DO' en 'LOOP' zijn verplicht in te geven.
- '{WHILE | UNTIL} condition' en 'statementblock" zijn optioneel in te geven, want ze staan tussen vierkante haakjes.
- Indien '{WHILE | UNTIL} condition' toch gebruikt wordt, dan moet men ofwel het woordje 'WHILE' ofwel het woordje 'UNTIL' gebruiken, want deze staan tussen accolades en zijn gescheiden door een verticaal streepje.

Oefening 1

Maak alle mogelijke combinaties als volgende syntax voor een nederlandse zin opgegeven is:
DEZE HOND { WAS GISTEREN | IS VANDAAG | WORDT MORGEN | LIJKT } [{ ERG | KNAP }] VERVELEND.

Oefening 2

Maak alle mogelijke combinaties als volgende syntax voor het qbasic commando CIRCLE opgegeven is:
CIRCLE [STEP] (x!,y!),radius![,[color%][,[start!][,[end!][,aspect!]]]]

Instructies uit deel 1

Commentaar in programma's

```
REM remark
```

Definiëren van variabelen

```
DIM [SHARED] variable[(subscripts)] [AS type]
  [,variable[(subscripts)] [AS type]]...
```

Wissen van het scherm

```
CLS [{0 | 1 | 2}]
```

Printen van tekst of variabelen

```
PRINT [#filenumber%,] [expressionlist] [{; | ,}]
```

Veranderen van kleuren

```
COLOR [foreground%] [,[background%] [,border%]]
```

Invoeren van variabelen

```
INPUT [;] ["prompt"{; | ,}] variablelist
```

Conditionele uitvoering van instructies

```
IF condition1 THEN
  [statementblock-1]
[ELSEIF condition2 THEN
  [statementblock-2]]...
[ELSE
  [statementblock-n]]
END IF
```

Conditionele uitvoering van instructies

```
SELECT CASE testexpression
CASE expressionlist1
  [statementblock-1]
[CASE expressionlist2
  [statementblock-2]]...
[CASE ELSE
  [statementblock-n]]
END SELECT
```

Een specifiek aantal keren opdrachten uitvoeren

```
FOR counter = start TO end [STEP increment]
  [statementblock]
NEXT [counter [,counter]...]
```

Opdrachten uitvoeren zolang een conditie geldt

```
WHILE condition
  [statementblock]
WEND
```

Opdrachten uitvoeren zolang of totdat een conditie geldt

```
DO [{WHILE | UNTIL} condition]
  [statementblock]
LOOP

DO
  [statementblock]
LOOP [{WHILE | UNTIL} condition]
```

Definiëren, lezen en opnieuwe laten lezen van specifieke gegevens

```
DATA constant[,constant]...
READ variablelist
RESTORE [line]
```

Veranderen van screen-mode

```
SCREEN mode% [,[colorswitch%] [,[activepage%] [,visualpage%]]]
```

Veranderen van kleuren

```
COLOR [background%] [,palette%]    '(Screen mode 1)
COLOR [foreground%]                '(Screen modes 4, 12, 13)
COLOR [foreground%] [,background&] '(Screen modes 7-10)
```

Tekent een puntje op het scherm

```
PRESET [STEP] (x!,y!) [,color%]
PSET [STEP] (x!,y!) [,color%]
```

Tekent een lijn of rechthoek op het scherm

```
LINE [[STEP](x1!,y1!)]-[STEP](x2!,y2!)
  [,[color%] [,[B | BF] [,style%]]]
```

Tekent een circle of ellips op het scherm

```
CIRCLE [STEP] (x!,y!),radius!
  [,[color%] [,[start!] [,[end!] [,aspect!]]]]
```

Vult een vlak met een kleur of een patroon

```
PAINT [STEP] (x!,y!)
  [,[{color% | tile$}] [,[bordercolor%] [,background$]]]
```

Ingebouwde functies

Qbasic kent een aantal ingebouwde functies, zowel numerische als alfanumerische, die we onmiddellijk kunnen gebruiken.

- Numerische functies geven als resultaat een numerische waarde (=getal) terug.
- Alfanumerische functies geven als resultaat een alfanumerische waarde (=string) terug. Deze string kan 0, 1 of meer tekens bevatten.

De meeste functies hebben tenminste 1 argument en geven altijd juist 1 waarde terug. (behalve bij een ongeldig argument vb. de vierkantswortel van een negatief getal (qbasic werkt niet met imaginaire getallen))

Alfanumerische basisfuncties

+

Strings kunnen niet opgeteld worden, maar QBasic plakt ze wel aan elkaar (=concatenatie).

s$ = s1$ + s2$
Plakt s1$ en s2$ aan elkaar en stopt het resultaat in s$.

voorbeeld

```
REM program      : qb0201.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : concatenatie

s1$ = "Computer"
s2$ = "club"
PRINT s1$ + s2$

END
```

LEFT$

Linker gedeelte van een string afzonderen.

l$ = LEFT$(s$, n)
Neemt de n linkse tekens van s$ en stopt het resultaat in l$.

voorbeeld

```
REM program      : qb0202.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : left$

s$ = "Computerclub"
PRINT LEFT$(s$, 8)

END
```

RIGHT$

Rechter gedeelte van een string afzonderen.

r$ = RIGHT$(s$, n)
Neemt de n rechtse tekens van s$ en stopt het resultaat in r$.

voorbeeld

```
REM program      : qb0203.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : right$

s$ = "Computerclub"
PRINT RIGHT$(s$, 4)

END
```

MID$

Middelste gedeelte van een string afzonderen.

m$ = MID$(s$, p, n)
Neemt vanaf positie p n tekens van s$ en stopt het resultaat in m$.

voorbeeld

```
REM program      : qb0204.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : mid$

s$ = "Computerclub"
PRINT MID$(s$, 4, 3)

END
```

Numerische basisfuncties

LEN

Bepalen van de lengte van een string.

l = LEN(s$)
Bepaalt de lengte van s$ en stopt het resultaat in l.

voorbeeld

```
REM program      : qb0205.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : len

s$ = "Computerclub"
PRINT LEN(s$)

END
```

INSTR

Zoekt de positie van een string in een andere string.

p = INSTR(s1$, s2$)

Zoekt de positie van s2$ in s1$ en stopt het resultaat in p.

voorbeeld

```
REM program      : qb0206.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : instr

s$ = "Computerclub"
PRINT INSTR(s$, "put")

END
```

Toepassingen

We geven de naam in van een straat.
Als de naam eindigt op "steenweg", dan korten we dit af tot "stw".

```
REM program      : qb0207.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : afkorten steenweg tot stw

DIM straat AS STRING

INPUT "Geef de naam van de straat : ", straat
IF (RIGHT$(straat, 8) = "steenweg") THEN
  PRINT LEFT$(straat, LEN(straat) - 8) + "stw"
ELSE
  PRINT straat
END IF

END
```

Vraag een woord aan de gebruiker.
Druk dan de eerste letter van het woord af, dan de twee eerste letters, enz, totdat het ganse woord afgedrukt is.

```
REM program      : qb0208.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : druk eerste letters van een string

DIM woord AS STRING
DIM lengte AS INTEGER, i AS INTEGER

INPUT "Geef een woord : ", woord
lengte = LEN(woord)
FOR i = 1 TO lengte
  PRINT LEFT$(woord, i)
NEXT i

END
```

Vraag een woord aan de gebruiker.
Druk dan de laatste letter van het woord af, dan de twee laatste letters, enz, totdat het ganse woord afgedrukt is.
Zorg ervoor dat de laatste letters onder elkaar staan. (rechts-gealligneerd)

```
REM program      : qb0209.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : druk laatste letters van een string

DIM woord AS STRING
```

```
DIM lengte AS INTEGER, i AS INTEGER

INPUT "Geef een woord : ", woord
lengte = LEN(woord)
FOR i = 1 TO lengte
  PRINT TAB(lengte - i + 1); RIGHT$(woord, i)
NEXT i

END
```

Vraag een woord aan de gebruiker.
Druk daarna letter voor letter van dat woord af op het scherm.

```
REM program      : qb0210.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : druk de middelste letters van een string

DIM woord AS STRING
DIM lengte AS INTEGER, i AS INTEGER
DIM x AS INTEGER, y AS INTEGER

INPUT "Geef een woord : ", woord
CLS
lengte = LEN(woord)
FOR i = 1 TO lengte
  x = 1 + RND * 79
  y = 1 + RND * 24
  LOCATE y, x
  PRINT MID$(woord, i, 1);
  SLEEP 1
  LOCATE y, x
  PRINT " ";
NEXT i

END
```

Oefening 3

1. We geven een getal in voorafgegaan door nullen (in een string variabele). Druk de beduidende cijfers af.

2. We geven een zin in en er wordt geheimtaal afgedrukt. Na elke klinker worden er een "p" en deze klinker afgedrukt.

3. We geven de zin in geheimtaal in en drukken dan de gedecodeerde versie af.

4. We drukken de woorden "Computer", "Club", "Masano" en "Schaffen" schuin af.

Alfanumerische - Numerische conversie functies

STR$

Zet een getal om in zijn string-vorm.

s$ = STR$(i)
Zet de waarde van i om in string-vorm en stopt het resultaat in s$.

voorbeeld

```
REM program      : qb0215.bas
REM author       : michel gielens
REM date written : 2001-09-17
```

```
REM description  : str$

i% = -256
PRINT "|"; STR$(i%); "|"

END
```

VAL

Zet een getal in string-vorm om naar een getal.

i = VAL(s$)
Zet het getal in string-vorm in s$ om naar een getal en stopt het resultaat in i.

voorbeeld

```
REM program      : qb0216.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : val

s$ = "-03.140"
PRINT "|"; VAL(s$); "|"

END
```

CHR$

Geeft het karakter of teken van desbetreffende ascii code.

c$ = CHR$(i)
Zet het karakter van ascii code i in c$.

voorbeeld

```
REM program      : qb0217.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : chr$

i% = 77
PRINT "|"; CHR$(i%); "|"

END
```

ASC

Geeft de ascii code van een karakter of teken.

i = ASC(c$)
Zet de ascii code van het karakter c$ in i.

voorbeeld

```
REM program      : qb0218.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : asc

s$ = "Z"
PRINT "|"; ASC(s$); "|"

END
```

Toepassingen

Geef een getal in groter dan 1000.
Bepaal dan met de stringfuncties het aantal duizendtallen.

```
REM program      : qb0219.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : duizendtallen

DIM sgetal AS STRING, igetal AS LONG

DO
  INPUT "Geef een geheel getal groter dan 1000 : "; igetal
LOOP UNTIL (igetal > 1000)

sgetal = LTRIM$(STR$(igetal))
PRINT sgetal; " heeft "; LEFT$(sgetal, LEN(sgetal) - 3); "
duizendtal(len)."

END
```

Vraag de geboortedatum van een persoon in formaat "dd/mm/jjjj".
Controleer of de datum juist ingegeven werd.
Zonder dan de dag, de maand en het jaar af en stop deze in numerische variabelen.

```
REM program      : qb0220.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : geboortedatum

CONST cijfers = "0123456789"

DIM datum AS STRING
DIM dag AS INTEGER, maand AS INTEGER, jaar AS INTEGER
DIM ok AS STRING * 1, i AS INTEGER

DO
  INPUT "Geef de geboortedatum : ", datum
  ok = "Y"
  IF (LEN(datum) <> 10) THEN
    ok = "N"
  ELSE
    FOR i = 1 TO 10
      SELECT CASE (i)
        CASE 1 TO 2, 4 TO 5, 7 TO 10
          IF (INSTR(cijfers, MID$(datum, i, 1)) = 0) THEN ok = "N"
        CASE 3, 6
          IF (MID$(datum, i, 1) <> "/") THEN ok = "N"
      END SELECT
    NEXT i
  END IF
LOOP UNTIL (ok = "Y")

dag = VAL(MID$(datum, 1, 2))
maand = VAL(MID$(datum, 4, 2))
jaar = VAL(MID$(datum, 7, 4))

PRINT "dag   ="; dag
PRINT "maand ="; maand
PRINT "jaar  ="; jaar

END
```

Druk een tabel van 4 kolommen met de karakters en ascii-waarden van alle hoofdletters en kleine letters.

```
REM program      : qb0221.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : lettertabel

CONST hletters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
CONST kletters = "abcdefghijklmnopqrstuvwxyz"

DIM hletter AS STRING * 1, kletter AS STRING * 1
DIM i AS INTEGER
DIM x AS INTEGER, y AS INTEGER

CLS

FOR i = 1 TO 26
  hletter = MID$(hletters, i, 1)
  kletter = MID$(kletters, i, 1)
  y = (i - 1) MOD 13
  x = (i - 1) \ 13
  LOCATE 1 + y, 1 + x * 10
  PRINT USING "! ###"; hletter; ASC(hletter)
  LOCATE 1 + y, 21 + x * 10
  PRINT USING "! ###"; kletter; ASC(kletter)
NEXT i

END
```

Druk een tabel van alle ascii tekens >= 32 af.

```
REM program      : qb0222.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : asciitabel

DIM x AS INTEGER, y AS INTEGER

CLS

FOR y = 2 TO 15
  PRINT USING "###  "; 16 * y;
  FOR x = 0 TO 15
    PRINT CHR$(16 * y + x);
  NEXT x
  PRINT
NEXT y

END
```

Oefening 4

Vraag aan de gebruiker om een getal in te geven (eventueel met een komma).
Stop dit eerst in een string variabele en controleer dan of dit inderdaad een getal is.
Controleer op het teken (+ of -), op het aantal komma's, of er geen vreemde tekens tussen staan, enz...

Veelgebruikte alfanumerische functies

LCASE$

Converteert een string naar kleine letters (lowercase).

s1$ = LCASE$(s2$)
Converteert de string s2$ naar kleine letters en zet het resultaat in s1$.

voorbeeld

```
REM program      : qb0224.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : lcase$

s$ = "Computer Club Masano Schaffen"
PRINT LCASE$(s$)

END
```

UCASE$

Converteert een string naar grote letters (uppercase).

s1$ = UCASE$(s2$)
Converteert de string s2$ naar grote letters en zet het resultaat in s1$.

voorbeeld

```
REM program      : qb0225.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : ucase$

s$ = "Computer Club Masano Schaffen"
PRINT UCASE$(s$)

END
```

LTRIM$

Verwijdert spaties aan het begin van een string.

s1$ = LTRIM$(s2$)
Verwijdert de spaties aan het begin van string s2$ en zet het resultaat in s1$.

voorbeeld

```
REM program      : qb0226.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : ltrim$

s$ = "  Diest  "
PRINT "|"; LTRIM$(s$); "|"

END
```

RTRIM$

Verwijdert spaties aan het einde van een string.

s1$ = RTRIM$(s2$)
Verwijdert de spaties aan het einde van string s2$ en zet het resultaat in s1$.

voorbeeld

```
REM program      : qb0227.bas
REM author       : michel gielens
REM date written : 2001-09-17
```

```
REM description  : rtrim$

s$ = "  Diest  "
PRINT "|"; RTRIM$(s$); "|"

END
```

SPACE$

Maakt een string van allemaal spaties.

s$ = SPACE$(n%)
Maakt een string van n% spaties en zet het resultaat in s$.

voorbeeld

```
REM program      : qb0228.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : space$

PRINT "|"; SPACE$(5); "|"

END
```

STRING$

Maakt een string van allemaal hetzelfde zelfgekozen teken.

s$ = STRING$(n%, c$)
Maakt een string van n% keer het teken c$.

STRING$(x, " ") is dus hetzelfde als SPACE$(x)

voorbeeld

```
REM program      : qb0229.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : string$

PRINT "|"; STRING$(5, "x"); "|"

END
```

INKEY$

Leest een teken van het toetsenbord.

c$ = INKEY$
Leest een teken van het toetsenbord en stopt dit in c$.

- is er geen toetsaanslag, dan is c$ = ""
- bij een gewone toetsaanslag is c$ 1 byte lang (het ingelezen teken)
- bij een extended toetsaanslag is c$ 2 bytes lang (een ASCII 0 + de scan code van de toets)

voorbeeld

```
REM program      : qb0230.bas
REM author       : michel gielens
REM date written : 2001-09-17
```

```
REM description  : inkey$

DIM i AS INTEGER

PRINT "Druk <Esc> om te stoppen..."
c$ = INKEY$
DO UNTIL c$ = CHR$(27)
  FOR i = 1 TO LEN(c$)
    PRINT ASC(MID$(c$, i, 1));
  NEXT i
  c$ = INKEY$
LOOP
PRINT

END
```

Toepassingen

Vraag een zin aan de gebruiker (minder dan 80 tekens) en druk deze dan gecentreerd af in kleine letters op het scherm. Onderlijn deze zin.

```
REM program      : qb0231.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : gecentreerd afdrukken

DIM lengte AS INTEGER
DIM zin AS STRING

DO
  INPUT "Geef een zin korter dan 80 tekens : "; zin
LOOP UNTIL (LEN(zin) < 80)

zin = LCASE$(LTRIM$(RTRIM$(zin)))
lengte = LEN(zin)
PRINT SPACE$((80 - lengte) / 2); zin
PRINT SPACE$((80 - lengte) / 2); STRING$(lengte, "-")

END
```

Om te vermijden dat er steeds een extra <enter> moet gedrukt worden, gaan we de inkey$ gebruiken om de gebruiker op een snelle manier op iets te laten antwoorden.

```
REM program      : qb0232.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : toepassing inkey$

DO
  PRINT "Nog eens (j/n) ? ";
  DO
    c$ = INKEY$
  LOOP UNTIL (c$ = "j" OR c$ = "n")
  PRINT c$
LOOP UNTIL (c$ = "n")

END
```

Oefening 5

Maak een soort paswoord beveiliging. De gebruiker moet een geheim paswoord ingeven.
De letters van het ingegeven paswoord worden als sterretjes weergegeven op het scherm.
Het programma mag pas verder gaan indien het juiste paswoord ingegeven werd.

Veelgebruikte numerische functies

ABS

Geeft de absolute waarde van een numerische expressie.

i = ABS(j)
Zet de absolute waarde van j in i.

voorbeeld

```
REM program      : qb0234.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : val

PRINT ABS(-47.5)

END
```

INT

Geeft het grootste geheel getal kleiner of gelijk aan een numerische expressie.

i = INT(j)
Zet het grootste geheel getal kleiner of gelijk aan j in i.

voorbeeld

```
REM program      : qb0235.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : int

PRINT INT(1.6); INT(-1.4)

END
```

SQR

Geeft de vierkantswortel van een numerische expressie.

i = SQR(j)
Zet de vierkantswortel van j in i.

voorbeeld

```
REM program      : qb0236.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : sqr

PRINT SQR(289)

END
```

RND

Geeft een "willekeurig" getal tussen 0 en 1 (in single precision).

i = RND
Zet een willekeurig getal tussen 0 en 1 in i.

voorbeeld

```
REM program      : qb0237.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : rnd

PRINT INT(RND * 6) + 1;

END
```

Toepassingen

Druk de afstand tussen 2 willekeurige punten af.

```
REM program      : qb0238.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : afstand tussen 2 punten

DIM x1 AS INTEGER, y1 AS INTEGER
DIM x2 AS INTEGER, y2 AS INTEGER
DIM dx AS INTEGER, dy AS INTEGER, a AS SINGLE

x1 = INT(RND * 100 - 50)
y1 = INT(RND * 100 - 50)
x2 = INT(RND * 100 - 50)
y2 = INT(RND * 100 - 50)

dx = (x2 - x1)
dy = (y2 - y1)

a = SQR(dx * dx + dy * dy)

PRINT "De afstand tussen ("; x1; ","; y1; ") en ("; x2; ","; y2; ")
= "; a

END
```

Veelgebruikte goniometrische functies

SIN, COS, TAN

Geeft resp. de sinus, de cosinus en de tangens van een hoek in radialen.

i = SIN(h)
i = COS(h)
i = TAN(h)

voorbeeld

```
REM program      : qb0239.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : sin, cos, tan

DIM h AS INTEGER, r AS DOUBLE

PRINT "   h        r       sin       cos       tan"
FOR h = 0 TO 360 STEP 15
```

```
    r = 3.141592653589793# / 180# * h
    PRINT USING " ### #.######"; h; r;
    PRINT TAB(15); USING "##.######"; SIN(r);
    PRINT TAB(25); USING "##.######"; COS(r);
    IF (((h - 90) MOD 180) <> 0) THEN
      PRINT TAB(35); USING "##.######"; TAN(r)
    ELSE
      PRINT
    END IF
  NEXT h

  END
```

ATN

Geeft de boogtangens van een numerische expressie.

h = ATN(i)

voorbeeld

```
REM program      : qb0240.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : atn

PRINT "pi ="; ATN(1) * 4#

END
```

Toepassingen

Teken een cirkel, gebruik makend van de functies sin en cos.

```
REM program      : qb0241.bas
REM author       : michel gielens
REM date written : 2001-09-17
REM description  : cirkel

REM Schakelt over naar grafische mode.
REM Windows-gebruikers moeten hiervoor qbasic opstarten in full
REM screen mode.

DIM pi AS SINGLE
DIM r AS SINGLE

pi = ATN(1) * 4!

SCREEN 12                              ' schakel over naar 640*480
WINDOW (-4 / 3, 1)-(4 / 3, -1)         ' 4/3 = verhouding beeldscherm

PSET (COS(0), SIN(0))                  ' plaats eerste punt
FOR r = 0 TO 2 * pi STEP pi / 180!
  LINE -(COS(r), SIN(r))               ' trek lijn naar volgende punt
NEXT r
LINE -(COS(0), SIN(0))                 ' trek lijn naar laatste punt

DO
LOOP UNTIL (INKEY$ <> "")              ' wacht op toetsaanslag

SCREEN 0                               ' schakel over naar text mode

END
```

Subroutines & functies

QBasic biedt veel reeds ingebouwde subroutines en functies die volstaan om veel problemen op te lossen.
Nochtans kan bij het oplossen van een probleem de noodzaak zich voordoen om zelf een speciale routine of functie te ontwerpen die QBasic standaard niet kent. In dit hoofdstuk bekijken we hoe we zo'n subroutines en functies kunnen ontwerpen.

Het grootste onderscheid tussen een subroutine en een functie is het volgende: de functie geeft een waarde terug en de subroutine niet.

Subroutines

Stel dat je een programma hebt gemaakt waarbij een bepaald gedeelte van de code nogal dikwijls herhaald wordt in verschillende stukken van dat programma.

Een heel efficiente manier van programmeren is dan om dat gedeelte van de code in een subroutine te stoppen. Het hoofdprogramma doet dan gewoon een oproep naar deze subroutine, waarbij de code uitgevoerd wordt en daarna wordt teruggekeerd waar het hoofdprogramma gebleven was.

voorbeeld

In een programma wordt heel dikwijls een zin gecentreerd op het scherm afgedrukt. De routine om die gecentreerde afdruk te verzorgen, wordt in een subroutine geplaatst.

programma ervoor:

```
REM program      : qb0301.bas
REM author       : michel gielens
REM date written : 2001-09-29
REM description  : subroutine1

DIM l AS INTEGER

CLS
s$ = "er was eens een visje"
l = LEN(s$)
PRINT SPACE$((80 - l) / 2); s$
PRINT
s$ = "dat zwom in het rond"
l = LEN(s$)
PRINT SPACE$((80 - l) / 2); s$
PRINT
s$ = "het zocht of het nergens"
l = LEN(s$)
PRINT SPACE$((80 - l) / 2); s$
PRINT
s$ = "een kruimeltje vond"
l = LEN(s$)
PRINT SPACE$((80 - l) / 2); s$
PRINT

END
```

programma erna:

```
REM program      : qb0302.bas
REM author       : michel gielens
REM date written : 2001-09-29
REM description  : subroutine1
```

```
DECLARE SUB centreer (s AS STRING)

CLS
CALL centreer("er was eens een visje")
CALL centreer("dat zwom in het rond")
CALL centreer("het zocht of het nergens")
CALL centreer("een kruimeltje vond")

END

SUB centreer (s AS STRING)
  DIM l AS INTEGER
  l = LEN(s)
  PRINT SPACE$((80 - l) / 2); s
  PRINT
END SUB
```

Voordelen:
- programma is veel overzichtelijker
- programma is ook korter
- bij een wijziging in de routine, moet deze maar op 1 plaats gedaan worden
- men maakt minder fouten

Functies

Functies verschillen van subroutines enkel door het feit dat ze een waarde kunnen teruggeven.

voorbeeld

Een programma dat het grootste getal uit 2 getallen kiest.

programma ervoor:

```
REM program      : qb0303.bas
REM author       : michel gielens
REM date written : 2001-09-29
REM description  : functie1

DIM getal1 AS INTEGER, getal2 AS INTEGER, grootste AS INTEGER

getal1 = 17
getal2 = 19

IF (getal1 > getal2) THEN
  grootste = getal1
ELSE
  grootste = getal2
END IF

PRINT grootste; "is het grootste getal van"; getal1; "en"; getal2

END
```

programma erna:

```
REM program      : qb0304.bas
REM author       : michel gielens
REM date written : 2001-09-29
REM description  : functie1

DECLARE FUNCTION max% (i1 AS INTEGER, i2 AS INTEGER)

DIM getal1 AS INTEGER, getal2 AS INTEGER

getal1 = 17
```

```
getal2 = 19

PRINT max%(getal1, getal2); "is het grootste getal van"; getal1;
"en"; getal2

END

FUNCTION max% (i1 AS INTEGER, i2 AS INTEGER)
  IF (i1 > i2) THEN
    max% = i1
  ELSE
    max% = i2
  END IF
END FUNCTION
```

Het toevoegen van subroutines en functies doet met als volgt:
Engelse versie: druk <alt-E> voor het Edit menu en daarna <S> voor een nieuwe subroutine (new Sub) of <F> voor een nieuwe functie (new Function).
Nederlandse versie: druk <alt-W> voor het bewerken menu en daarna <S> voor een nieuwe subroutine of <F> voor een nieuwe functie.
Geef een naam aan je subroutine of functie.
Plaats daarna de eventuele parameters tussen ronde haakjes achter de naam.

Variabelen die in de subroutine of functie gedefinieerd worden, zijn enkel zichtbaar in die subroutine of functie.
Bij een tweede aanroep ervan, zijn de waarden van de variabelen 'vergeten'. Moeten ze toch onthouden worden, gebruik dan het woord 'STATIC'.

voorbeeld

Maak een functie om de faculteit van een getal te berekenen.
Vb. 5! = 1 * 2 * 3 * 4 * 5 = 120
Vb. 7! = 1 * 2 * 3 * 4 * 5 * 6 * 7 = 5040
(neem als resultaat een getal van het type 'double')

```
REM program      : qb0305.bas
REM author       : michel gielens
REM date written : 2001-09-29
REM description  : faculteit

DECLARE FUNCTION fac# (i AS INTEGER)

DIM i AS INTEGER

FOR i = 0 TO 9
  PRINT i; TAB(5); fac#((i))
NEXT i

END

FUNCTION fac# (i AS INTEGER)
  DIM f AS DOUBLE
  IF (i < 0) THEN
    PRINT "Error : fac#("; i; ")"
  ELSE
    f = 1
    DO WHILE (i > 1)
      f = f * i
      i = i - 1
    LOOP
    fac# = f
  END IF
END FUNCTION
```

Bij de functieaanroep staat er fac#((i))

Er werd rond de i een extra paar haakjes gezet om de waarde van i door te geven en niet i zelf. Anders zou de functie zelf i telkens veranderen zodat het programma nogal raar gaat doen. Probeer dit maar eens uit: vervang fac#((i)) door fac#(i)

voorbeeld

Maak een functie die een string als parameter heeft en als resultaat een string teruggeeft, ontdaan van spaties aan het begin en het einde van de oorspronkelijke string, en met quotes errond gezet. Deze functie komt goed van pas bij het maken van files in .CSV formaat, waarbij de velden gescheiden zijn door een komma en er rond de strings quotes staan.

```
REM program      : qb0306.bas
REM author       : michel gielens
REM date written : 2001-09-29
REM description  : quoted string

DECLARE FUNCTION qstring$ (s AS STRING)

s$ = "  zeepsop  "
PRINT qstring$(s$)
PRINT qstring$("  wast witter dan wit  ")

END

FUNCTION qstring$ (s AS STRING)
  qstring$ = CHR$(34) + LTRIM$(RTRIM$(s)) + CHR$(34)
END FUNCTION
```

voorbeeld

Maak een procedure om een kadertje te tekenen op het scherm.
Als parameters moeten de coordinaten van de linkerbovenhoek meegeven worden en de hoogte en breedte van het kadertje (inclusief het kadertje zelf). Geef ook als parameter de kleuren van de kader mee.

```
REM program      : qb0307.bas
REM author       : michel gielens
REM date written : 2001-09-29
REM description  : kaders

DECLARE SUB kader (x AS INTEGER, y AS INTEGER, breedte AS INTEGER,
hoogte AS INTEGER, vkleur AS INTEGER, akleur AS INTEGER)

CONST groen = 2
CONST rood = 4

CLS
CALL kader(10, 2, 60, 8, groen, rood)

END

SUB kader (x AS INTEGER, y AS INTEGER, breedte AS INTEGER, hoogte AS
INTEGER, vkleur AS INTEGER, akleur AS INTEGER)
  IF (x < 1 OR y < 1 OR breedte < 2 OR hoogte < 2) THEN EXIT SUB
  IF ((x + breedte) > 80 OR (y + hoogte) > 25) THEN EXIT SUB
  IF (vkleur < 0 OR vkleur > 15 OR akleur < 0 OR akleur > 15) THEN
EXIT SUB
  DIM i AS INTEGER
  COLOR vkleur, akleur
  LOCATE y, x: PRINT "+"; STRING$(breedte - 2, "-"); "+";
  FOR i = 1 TO hoogte - 2
    LOCATE y + i, x: PRINT "|"; STRING$(breedte - 2, " "); "|";
  NEXT i
  LOCATE y + hoogte - 1, x: PRINT "+"; STRING$(breedte - 2, "-");
"+";
  COLOR 7, 0
END SUB
```

Recursieve functies

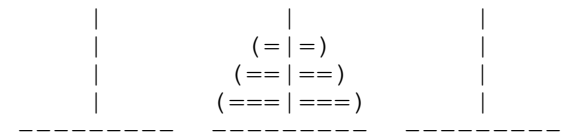
Een functie die zichzelf oproept, wordt een recursieve functie genoemd.
Soms is het met iteratieve statements (loops, ...) erg moeilijk om een probleem op te lossen. Hierbij kan een resursieve functie wel eens helpen.

voorbeeld: de torens van Hanoi

Er zijn 3 stapels: 1 volle en 2 lege. De volle stapel bevat opeengestapelde schijven. Elke schijf heeft een verschillende diameter. De schijven zijn zo gestapeld dat een schijf nooit op een kleinere schijf mag liggen. M.a.w. ze liggen dus gesorteerd met de kleinste boven en de grootste onder.
Het doel is om de volle stapel schijf per schijf te verplaatsen naar de tweede stapel, gebruikmakend van de derde stapel. Daarbij mag nooit uit het oog verloren worden dat een schijf nooit op een kleinere schijf mag liggen.

Beginsituatie:

Eindsituatie:



Oplossing

```
REM program      : qb0308.bas
REM author       : michel gielens
REM date written : 2001-09-29
REM description  : torens van hanoi

DECLARE SUB toon ()
DECLARE SUB verplaats (aantal AS INTEGER, van AS INTEGER, via AS
INTEGER, naar AS INTEGER)

CONST schijven = 5

' test aantal schijven
IF (schijven < 1 OR schijven > 12) THEN
  PRINT "onjuist aantal schijven"
  END
END IF

DIM SHARED toren(1 TO 3, 1 TO schijven) AS INTEGER
DIM i AS INTEGER

' opbouwen toren
FOR i = 1 TO schijven
  toren(1, i) = i
NEXT i

' toon beginsituatie
CLS
CALL toon

' verplaats torens
CALL verplaats(schijven, 1, 3, 2)

END

SUB toon
  DIM s AS INTEGER, t AS INTEGER
  ' teken alle schijven van de 3 torens
  FOR s = 1 TO schijven
    FOR t = 1 TO 3
      LOCATE s, 1 + (t - 1) * (schijven * 2 + 2)
```

```
      PRINT SPACE$(schijven - toren(t, s));
      PRINT STRING$(2 * toren(t, s), "#");
      PRINT SPACE$(schijven - toren(t, s))
    NEXT t
  NEXT s
  ' trek een lijntje onder elke toren
  FOR t = 1 TO 3
    LOCATE schijven + 1, 1 + (t - 1) * (schijven * 2 + 2)
    PRINT STRING$(2 * schijven, "-")
  NEXT t
  ' eventjes wachten
  SLEEP 1
END SUB

SUB verplaats (aantal AS INTEGER, van AS INTEGER, via AS INTEGER,
naar AS INTEGER)
  DIM s1 AS INTEGER, s2 AS INTEGER
  IF (aantal > 0) THEN
    CALL verplaats(aantal - 1, (van), (naar), (via))
    ' zoek bovenste schijf
    FOR s1 = 1 TO schijven
      IF (toren(van, s1) <> 0) THEN EXIT FOR
    NEXT s1
    ' zoek lege plaats
    FOR s2 = schijven TO 1 STEP -1
      IF (toren(via, s2) = 0) THEN EXIT FOR
    NEXT s2
    ' verplaats schijf
    SWAP toren(van, s1), toren(via, s2)
    ' toon de nieuwe situatie
    CALL toon
    CALL verplaats(aantal - 1, (naar), (via), (van))
  END IF
END SUB
```

Voordeel van de recursieve functie is dat ze soms een elegante oplossing biedt voor een anders heel moeilijk te implementeren algoritme.

Nadeel is dat ze heel veel geheugen kan verbruiken. Bij elke oproep moeten er immers parameters doorgegeven worden via de stack, het terugkeeradres (intern) wordt ook doorgegeven via de stack, de variabelen die in deze functie gedefinieerd worden komen ook nog eens op de stack. Indien deze functie verschillende keren na elkaar recursief wordt opgeroepen, kan de computer een stack-overflow krijgen. In het beste geval crasht het programma, in andere gevallen mag je je computer rebooten.

Zoek dus zoveel mogelijk naar een iteratieve oplossing i.p.v. een recursieve.

Voorbeeld van programma dat zal crashen.

```
REM program      : qb0309.bas
REM author       : michel gielens
REM date written : 2001-09-29
REM description  : crash

DECLARE SUB recsub ()
CALL recsub
END

SUB recsub
  CALL recsub
END SUB
```

Oefening 6

Maak een functie die een paswoord vraagt.
De ingedrukte toetsen moeten weergegeven worden als een sterretje.

Bestanden

Alle informatie die we willen bewaren (na het uitzetten van de computer), moeten we op een medium zetten die ook zonder stroom die informatie kan onthouden. De gegevens in het RAM-geheugen zijn we bij stroomonderbreking kwijt. Daarom wordt alles bijgehouden in bestanden op diskette, harddisk, CD, ... Deze bestanden kunnen programma's inhouden, gegevensbanken, voorkeurinstellingen van programma's, enz...
In dit hoofdstuk gaan we de data-bestanden behandelen.

Een eerste onderscheid dat er gemaakt kan worden is deze:
- sequentiële bestanden (= in volgorde)
- random bestanden (= willekeurig op record niveau)
- binaire bestanden (= willekeurig op byte niveau)

Sequentiële bestanden kunnen enkel records in opklimmende volgorde benaderen. De gegevens die in een bepaalde volgorde in deze bestanden geplaatst werden, kunnen enkel in dezelfde volgorde er terug uitgehaald worden. (denk aan magneetbanden).

Random bestanden laten ons toe om gegevens in totale willekeur in het bestand te plaatsen en deze ook in totale willekeur er terug uit te lezen. Dit gebeurt wel op record niveau. Uit een bestand met 50 records van een ledenbestand kunnen we onmiddellijk het 45e lid lezen zonder eerst alle voorgaande records te moeten inlezen.

Binaire bestanden kunnen we manipuleren tot op byte niveau. Elke byte in dit type bestand kunnen we lezen, veranderen, wegschrijven.

Als we een bestand willen gebruiken moeten we de volgende regels respecteren:
1. bestand openen
2. records lezen/schrijven
3. bestand terug sluiten

Om een bestand te openen, moet QBasic een aantal dingen weten:
- de naam van het bestand (eventueel met drive-aanduiding en padnaam)
- hoe het bestand openen (INPUT, OUTPUT, APPEND, RANDOM, BINARY)
- indien men in een netwerk werkt: de ACCESS en LOCK methode
- de file-handle (waarmee het bestand kan benaderd worden)
- voor random-bestanden: de recordlengte

Om een actie te doen op het bestand kunnen we volgende instructies gebruiken:
- schrijven : PRINT, WRITE, PUT
- lezen : INPUT, LINE INPUT, INPUT$, GET

Om een bestand terug te sluiten gebruiken we de instructie close.

Sequentiële bestanden

Zoals eerder gezegd, kunnen we met sequentiële bestanden (net zoals bij magneetbanden) enkel records in één bepaalde volgorde wegschrijven en deze later ook enkel maar in deze volgorde terug inlezen.
De records of de velden hoeven niet allemaal even lang te zijn:
- LINE INPUT leest door tot de eerste return die hij tegenkomt
- INPUT leest een aantal velden in die niet evenlang hoeven te zijn
- INPUT$ leest een aantal tekens in die de programmeur opgeeft

Voorbeeld

```
REM program      : qb0401.bas
REM author       : michel gielens
REM date written : 2001-10-23
REM description  : sequentieel

CLS

PRINT "Geef enkele regels tekst om in het bestand 'tekst.txt' op te
slagen."
PRINT "Sluit af met een lege regel (met gewoon <enter> dus)."

OPEN "tekst.txt" FOR OUTPUT AS #1
DO
  LINE INPUT regel$
  IF (regel$ <> "") THEN
    PRINT #1, regel$
  END IF
LOOP UNTIL regel$ = ""
CLOSE #1

PRINT
PRINT "Geef nu enkele regels tekst om achter het bestand 'tekst.txt'
op te slagen."
PRINT "Sluit af met een lege regel (met gewoon <enter> dus)."

OPEN "tekst.txt" FOR APPEND AS #1
DO
  LINE INPUT regel$
  IF (regel$ <> "") THEN
    PRINT #1, regel$
  END IF
LOOP UNTIL regel$ = ""
CLOSE #1

PRINT
PRINT "Dit is wat nu in het bestand staat:"

OPEN "tekst.txt" FOR INPUT AS #1
DO UNTIL EOF(1)
  LINE INPUT #1, regel$
  PRINT regel$
LOOP
CLOSE #1

END
```

In de eerste OPEN opdracht wordt het bestand "tekst.txt" geopend voor OUTPUT.
Als dit bestand reeds bestond, dan wordt dit eerst gewist. Daarna wordt het aangemaakt en geopend om er naartoe te schrijven.
De file-handle die QBASIC voor dit bestand gebruikt is #1. Alle schrijf-acties naar dit bestand moeten dus via file-handle #1 verlopen.
Het wegschrijven naar dit bestand kan met de PRINT opdracht (of eventueel de WRITE instructie).

De tweede OPEN opdracht gaat het bestand dat we daarnet gemaakt hebben opnieuw openen, maar nu voor APPEND. We willen er dus records achter aan toe te voegen.
Als dit bestand reeds bestond, dan wordt het gewoon geopend. Bestond het nog niet, dan wordt het eerst aangemaakt en dan geopend.
Verdere schrijf-acties gebeuren zoals daarnet.

De derde OPEN opdracht gaat het bestand dat we gemaakt hebben terug openen, maar nu voor INPUT. We wensen nu enkel de records te lezen.
Als dit bestand reeds bestond, dan wordt het gewoon geopend, anders krijg je een foutmelding (file not found).
Records lezen doen we met de LINE INPUT, INPUT of INPUT$ opdracht.

Oefening 7

Maak een programma dat de bestanden c:\autoexec.bat en c:\config.sys inleest en toont op het scherm.
Maak eventeel een reserve-copie van die bestanden vooraleer je het programma uitvoert.

Random bestanden

Met random bestanden kunnen we records in een willekeurige volgorde lezen en schrijven. De computer kan dus op voorhand bepalen waar precies in het bestand een record staat. Om dit te kunnen, moeten alle records even lang zijn.

Stel dat een record 20 tekens lang is.
Record 1 begint dan op positie 1, record 2 begint op positie 21 (=(2-1)*20+1), record 3 begint op positie 41 (=(3-1)*20+1), ..., record 87 begint op positie 1721 (=(87-1)*20+1), enz...

Hoe 'ziet' dit er zo een beetje uit?

file lay-out

	veld1	veld2	veld3	veld4	veld5	veld6
record1						
record2						
record3						
record4						
...						

record lay-out

	veld1 lidnummer	veld2 naam	veld3 straat	veld4 postnr	veld5 woonpl	veld6 telefoon
record	1	Michel Gielens	Kruisstraat 95	3294	Molenstede	013/336972

field lay-out

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
veld6:telefoon	0	1	3	/	3	3	6	9	7	2					

We kunnen deze random files het beste op volgende manier aanpakken:

1) beschrijf het record door een type aan te maken

```
TYPE lidtype
  lidnummer AS INTEGER
  naam AS STRING * 30
  straat AS STRING * 30
  postnr AS STRING * 10
  woonpl AS STRING * 30
  telefoon AS STRING * 15
END TYPE
```

2) definieer dan een variabele van dit type om het ganse record te bevatten

```
DIM lidrecord AS lidtype
```

3) open dan het ledenbestand en vertel qbasic hoelang elk record is

```
OPEN "leden.dat" FOR RANDOM AS #1 LEN = LEN(lidrecord)
```

4) opvullen van de velden kan als volgt

```
lidrecord.lidnummer = 1
lidrecord.naam = "Michel Gielens"
lidrecord.straat = "Kruisstraat 95"
lidrecord.postnr = "3294"
lidrecord.woonpl = "Molenstede"
lidrecord.telefoon = "013/336972"
```

5) dit record kunnen we dan wegschrijven als volgt

```
PUT #1, 1, lidrecord
```

6) record nummer 87 kunnen we dan lezen als volgt

```
GET #1, 87, lidrecord
```

7) vergeet ook niet achteraf het bestand terug af te sluiten

```
CLOSE #1
```

Voorbeeld

```
REM program      : qb0403.bas
REM author       : michel gielens
REM date written : 2001-10-27
REM description  : random

TYPE testtype
  veld AS STRING * 10
END TYPE

DIM testrecord AS testtype

DIM i AS INTEGER

CLS

OPEN "test.dat" FOR RANDOM AS #1 LEN = LEN(testrecord)

PRINT "opvullen van het bestand met 100 records (van achter naar
voor)..."
FOR i = 100 TO 1 STEP -1
  testrecord.veld = "NUMMER " + LTRIM$(STR$(i))
  PUT #1, i, testrecord
NEXT i

PRINT "teruglezen uit dit bestand van elk elfde record..."
FOR i = 1 TO 100 STEP 11
  GET #1, i, testrecord
  PRINT testrecord.veld
NEXT i

CLOSE #1

END
```

Opmerkingen:

Wil je het aantal records weten in een bestand? Gebruik dan volgende truuk:
`aantalrecords = LOF(1) / LEN(record)` Dit deelt de lengte van het bestand in bytes door de lengte van het record in bytes.

Een record wissen uit een file gaat in principe niet. Wat er wel kan, is het volgende:

- Voorzie in het record een extra veldje om te kunnen aanduiden of een record gewist is (met een

sterretje of zo). Dit is ook een handig truukje om de 'gewiste' records terug te heractiveren.
- Voorzie ook een routine om de file te comprimeren, dus om alle 'gewiste' records eruit te gooien. Dit kan door alle actieve records naar een nieuw random bestand uit te schrijven, het oorspronkelijke bestand te wissen en dan het nieuwe bestand te hernoemen naar het oude.

Oefening 8

Maak een programma om vorige opmerkingen uit te proberen.
Vul dus een bestand met een aantal records, wis er enkele door een sterretje in het record te zetten en beeld dan alle actieve records af.
Maak dan een routine om deze 'gewiste' records effectief te verwijderen.

Binaire bestanden

Soms moeten bestanden echt byte per byte benaderd kunnen worden. We denken dan aan bestanden waarvan de structuur helemaal niet als records kunnen beschouwd worden, vb. grafische bestanden, geluidsbestanden, ...
Hiervoor kunnen we -de naam zegt het al- binaire bestanden voor gebruiken.

Met de GET en de PUT opdrachten moeten we hier geen record-nummers opgeven om te lezen of te schrijven, maar bytenummers.
`GET #1, 27, i%` leest 2 bytes vanaf byte 27 in. (2 bytes omdat i% een integer is en dus 2 bytes groot is.)

Voorbeeld

```
REM program      : qb0405.bas
REM author       : michel gielens
REM date written : 2001-10-27
REM description  : binair

DIM i AS INTEGER, c AS STRING * 1

CLS

' een regeltje tekst in een bestand zetten
OPEN "test.dat" FOR OUTPUT AS #1
PRINT #1, "Hallo, allemaal."
CLOSE #1

' hetzelfde bestand terug openen, maar in binary mode
OPEN "test.dat" FOR BINARY AS #1
FOR i = LOF(1) TO 1 STEP -1
  GET #1, i, c
  PRINT USING "### "; ASC(c);   'print ascii-waarde van het teken
  IF (c >= " ") THEN            'is het teken afdrukbaar ?
    PRINT c                     'ja --> print teken
  ELSE
    PRINT                       'nee --> naar nieuwe regel
  END IF
NEXT i
PRINT
CLOSE #1

END
```

Praktische Toepassingen

Morse

Iedereen heeft al wel eens morse gehoord: in een western waar de trein zal overvallen worden, in een oude oorlogsfilm waar de geheime berichten moeten verstuurd worden, of gewoon de bip-bip-bip biep-biep bip-bip-bip van de GSM wanneer er een SMS'je binnenkomt. Met het volgende programma kunt u een regeltje tekst intikken, waarna de computer het zal vertalen (en laten horen) in morsetekens.
Het programma voorziet een array van 256 elementen (om het onszelf gemakkelijk te maken) met voor de gewone letters en cijfers en leestekens een string van puntjes en streepjes. De andere - niet gebruikte - tekens in de tabel hebben een lege string.
Het voortbrengen van het gebiep gebeurt door de `PLAY` instructie. Hiermee kunnen we o.a. de toonhoogte kiezen (octaaf en noot), de lengte, de pauzes, ... Bekijk de help van dit commando voor verdere informatie.

```
REM program      : qb0501.bas
REM author       : michel gielens
REM date written : 2001-11-11
REM description  : morse

'alle ascii codes, hoewel niet alles nodig
DIM morse(0 TO 255) AS STRING
DIM a AS INTEGER, i AS INTEGER, j AS INTEGER

DATA a,.-
DATA b,-...
DATA c,-.-.
DATA d,-..
DATA e,.
DATA f,..-.
DATA g,--.
DATA h,....
DATA i,..
DATA j,.---
DATA k,-.-
DATA l,.-..
DATA m,--
DATA n,-.
DATA o,---
DATA p,.--.
DATA q,--.-
DATA r,.-.
DATA s,...
DATA t,-
DATA u,..-
DATA v,...-
DATA w,.--
DATA x,-..-
DATA y,-.--
DATA z,--..
DATA 1,.----
DATA 2,..---
DATA 3,...--
DATA 4,....-
DATA 5,.....
DATA 6,-....
DATA 7,--...
DATA 8,---..
DATA 9,----.
DATA 0,-----
DATA .,......
DATA ;,-.-.-.
DATA ",",.-.-.-
DATA ":",---..
```

```
DATA ?,..--..
DATA !,--..--
DATA -,-....-
DATA "'",.----.
DATA (,-.--.-
DATA ),-.--.-
DATA " "," "
DATA **,**

'inlezen alle codes
READ teken$, dots$
DO WHILE (teken$ <> "**")
  a = ASC(teken$)
  morse(a) = dots$
  READ teken$, dots$
LOOP

'lengte "-" = 3 x lengte "."
'pauze tussen elk "-" of "." in 1 teken = lengte van "."
'pauze tussen 2 tekens = lengte van "-"

CLS

PLAY "MF" 'Music Foreground
PRINT "Geef een regel tekst en sluit af met <enter>"
PRINT
DO
  LINE INPUT regel$
  regel$ = LCASE$(regel$)
  FOR i = 1 TO LEN(regel$)
    a = ASC(MID$(regel$, i, 1))
    IF (morse(a) <> "") THEN
      PRINT morse(a);
      FOR j = 1 TO LEN(morse(a))
        m$ = MID$(morse(a), j, 1)
        IF (m$ = ".") THEN PLAY "L12C" 'Length 1/12 (=1/12) Note C
        IF (m$ = "-") THEN PLAY "L4C" 'Length 1/4 (=3/12) Note C
        PLAY "P12" 'Pause 1/12 (=1/12)
      NEXT j
      DO WHILE (PLAY(0) > 0): LOOP
      PRINT " ";
      PLAY "P6" 'Pause 1/6 (=2/12)
    END IF
  NEXT i
  PRINT
LOOP UNTIL (regel$ = "")

END
```

Plasma

Kleureffecten op simpele wijze programmeren? Het kan in QBasic.
Dit programma toont een willekeurig plasma op het scherm. (met in zwarte letters wat reclame voor CCMS uiteraard).
Eerst wordt overgeschakeld naar schermmode 13: de resolutie van 320 x 200 pixels is niet zo denderend, maar er zijn wel 256 kleuren beschikbaar en daar is het om te doen.
Dan wordt via een recursieve subroutine het plasma pixel voor pixel getekend in 192 verschillende kleuren (64 rode + 64 groene + 64 blauwe tinten). Het kleurenpalet wordt eerst wel even op zwart gezet, anders krijgen we even 'vieze' kleuren te zien.
Op het plasma wordt dan de reclame over heen geprojecteerd, de tekst in het zwart, de rest buiten de tekst laten we onaangeroerd. Deze informatie hadden we in de file 'ccmsbmp.dat' gestopt.
Vanaf nu laten we het kleurenpalet roteren totdat de gebruiker schele ogen krijgt en op een knopje drukt.

```
REM program      : qb0502.bas
REM author       : michel gielens
REM date written : 2001-11-10
```

```
REM description  : plasma

DECLARE SUB adjust (xa AS INTEGER, ya AS INTEGER, x AS INTEGER, y AS
INTEGER, xb AS INTEGER, yb AS INTEGER)
DECLARE SUB loadpic ()
DECLARE SUB setpalette ()
DECLARE SUB splitbox (x1 AS INTEGER, y1 AS INTEGER, x2 AS INTEGER,
y2 AS INTEGER)

CONST F = 4
DIM SHARED errorcode AS INTEGER
DIM SHARED pal(0 TO 255) AS LONG
DIM pal1 AS LONG
DIM i AS INTEGER
DIM timernow AS SINGLE

SCREEN 13               '320x200 256

PALETTE USING pal       'zet alle kleuren op zwart

RANDOMIZE (-TIMER)      'init random-number generator

PSET (0, 0), RND * 192          'random pixel linksboven
PSET (0, 199), RND * 192        'random pixel linksonder
PSET (319, 0), RND * 192        'random pixel rechtsboven
PSET (319, 199), RND * 192      'random pixel rechtsonder

CALL splitbox(0, 0, 319, 199)   'recursieve splitsing van het scherm

CALL loadpic            'inladen tekening (=tekst) indien aanwezig

CALL setpalette         'init het 'plasma'-palet

DO UNTIL (INKEY$ <> "") 'totdat we op een toets drukken...
  pal1 = pal(1)         'roteer 191 kleuren van het palet
  FOR i = 1 TO 190      '  kleur 0 moet zwart blijven
    pal(i) = pal(i + 1)
  NEXT i
  pal(191) = pal1
  PALETTE USING pal     'kleuren doorgeven
  timernow = TIMER      'wacht even (timer verandert 18.2 x / sec)
  DO WHILE (timernow = TIMER): LOOP
LOOP

SCREEN 0                'text-mode
WIDTH 80

END

handler1:
errorcode = ERR
RESUME NEXT

SUB adjust (xa AS INTEGER, ya AS INTEGER, x AS INTEGER, y AS
INTEGER, xb AS INTEGER, yb AS INTEGER)
  'geeft het punt in het midden van lijnstuk (xa,ya)-(xb,yb) een
  'pseudo willekeurige kleur op basis van de 2 punten van het
  'lijnstuk
  DIM a AS INTEGER, b AS INTEGER
  DIM d AS INTEGER
  DIM r AS SINGLE, v AS INTEGER
  IF (POINT(x, y) = 0) THEN
    r = RND - .5
    d = ABS(xa - xb) + ABS(ya - yb)
    a = POINT(xa, ya)
    b = POINT(xb, yb)
    v = (a + b) / 2 + d * r * F
    IF (v < 1) THEN v = 1
    IF (v > 191) THEN v = 191
    PSET (x, y), v
  END IF
END SUB
```

```
SUB loadpic
  'inladen van de tekening (hier is dit een tekst)
  DIM byte AS INTEGER, char AS STRING * 1
  DIM x AS INTEGER, y AS INTEGER, bit AS INTEGER
  DIM buffer AS STRING * 40
  'kijk of file wel bestaat
  errorcode = 0
  ON ERROR GOTO handler1
  OPEN "ccmsbmp.dat" FOR INPUT AS #1
  ON ERROR GOTO 0
  IF (errorcode) THEN EXIT SUB
  CLOSE #1
  'open de file
  OPEN "ccmsbmp.dat" FOR BINARY AS #1
  IF (LOF(1) <> 8000) THEN  'moet 8000 bytes groot zijn (320/8)*200
    CLOSE #1
    EXIT SUB
  END IF
  FOR y = 0 TO 199
    GET #1, , buffer
    FOR x = 0 TO 39
      byte = ASC(MID$(buffer, x + 1, 1))
      FOR bit = 0 TO 7
        IF ((byte AND 128) = 0) THEN PSET (8 * x + bit, y), 0
        byte = (byte + byte) AND 255
      NEXT bit
    NEXT x
  NEXT y
  CLOSE #1
END SUB

SUB setpalette
  'instellen van de 192 kleuren van 't palet (rood & groen & blauw)
  DIM c AS INTEGER
  FOR c = 0 TO 63
    pal(c) = 65536 * c + 256 * (63 - c) + 0
    pal(64 + c) = 65536 * (63 - c) + 256 * 0 + c
    pal(128 + c) = 65536 * 0 + 256 * c + (63 - c)
  NEXT c
  pal(0) = 0    'kleur 0 blijft zwart (voor de schermrand)
  PALETTE USING pal     'kleuren doorgeven
END SUB

SUB splitbox (x1 AS INTEGER, y1 AS INTEGER, x2 AS INTEGER, y2 AS
INTEGER)
  'recursieve routine om het gebied in 4 stukken te verdelen en te
  'behandelen
  DIM c1 AS INTEGER, c2 AS INTEGER, c3 AS INTEGER, c4 AS INTEGER
  DIM diffx AS INTEGER, diffy AS INTEGER
  DIM x AS INTEGER, y AS INTEGER
  diffx = x2 - x1
  diffy = y2 - y1
  IF (diffx >= 2 OR diffy >= 2) THEN
    x = (x1 + x2) / 2
    y = (y1 + y2) / 2
    CALL adjust(x1, y1, x, y1, x2, y1)
    CALL adjust(x2, y1, x2, y, x2, y2)
    CALL adjust(x1, y2, x, y2, x2, y2)
    CALL adjust(x1, y1, x1, y, x1, y2)
    c1 = POINT(x, y)
    IF (c1 = 0) THEN
      c1 = POINT(x1, y1)
      c2 = POINT(x2, y1)
      c3 = POINT(x1, y2)
      c4 = POINT(x2, y2)
      PSET (x, y), (c1 + c2 + c3 + c4) / 4
    END IF
    CALL splitbox(x1, y1, x, y)
    CALL splitbox(x, y1, x2, y)
    CALL splitbox(x, y, x2, y2)
    CALL splitbox(x1, y, x, y2)
```

```
  END IF
END SUB
```

Het bestand 'ccmsbmp.dat' kunt u downloaden uit dezelfde directory.

Op trage computers zult u merken dat dit programma vrij langzaam loopt. Dit komt hoofdzakelijk door de `PALETTE` instructie die QBasic nogal vrij traag uitvoert. Hetzelfde programma in C loopt veel sneller. Het programma 'plasma4.exe' kunt u downloaden uit dezelfde directory

Sorteren & Zoeken

Over het sorteren van gegevens en het zoeken naar deze gegevens is al veel literatuur verschenen. In dit korte programma wordt een lijst van willekeurige getallen gesorteerd volgens de 'quick-sort' methode en daarna wordt er een element in gezocht via de 'binary-search' methode.
De quick-sort is één van de snelste manieren om een lijst in het geheugen van de computer te sorteren. Er bestaan nog andere algorithmen en zelfs geoptimaliseerde quick-sorts, maar dat is voor de specialisten. Lees hiervoor de boeken van 'The Art of Computer Programming'.
De binary-search is dan weer een zeer snelle methode om iets in een lijst terug te vinden (of niet te vinden).
Beide algorithmen zijn in andere programmeertalen (vb. C) al ingebakken, in QBasic moeten we ze even zelf maken.

```
REM program      : qb0503.bas
REM author       : michel gielens
REM date written : 2001-11-07
REM description  : sorteer- en zoek-routines op arrays

DECLARE FUNCTION srcbin% (x AS INTEGER, l AS INTEGER, r AS INTEGER)
DECLARE SUB mkrandom (l AS INTEGER, r AS INTEGER)
DECLARE SUB srtquick (l AS INTEGER, r AS INTEGER)
DECLARE SUB toonrij (l AS INTEGER, r AS INTEGER)

CONST maxrij = 26
CONST van = 1, tot = van + maxrij - 1

DIM SHARED rij(van TO tot) AS INTEGER
DIM i AS INTEGER, zoek AS INTEGER

CLS

RANDOMIZE (-TIMER)

CALL mkrandom(van, tot)

PRINT "De ongesorteerde rij:"
CALL toonrij(van, tot)
zoek = rij(van)

CALL srtquick(van, tot)

PRINT
PRINT "De gesorteerde rij:"
CALL toonrij(van, tot)

PRINT
i = srcbin(zoek, van, tot)
IF (i = -1) THEN
  PRINT "Getal"; zoek; "is niet gevonden in de rij."
ELSE
  PRINT "Getal"; zoek; "is element"; i; "in de rij."
END IF

END

SUB mkrandom (l AS INTEGER, r AS INTEGER)
  'maakt een lijst met willekeurige waarden
```

```
  DIM i AS INTEGER
  FOR i = l TO r
    rij(i) = INT(RND * 100)
  NEXT i
END SUB

FUNCTION srcbin% (x AS INTEGER, l AS INTEGER, r AS INTEGER)
  'zoekt een element via de binary search
  'de rij moet wel gesorteerd zijn!
  DIM i AS INTEGER, j AS INTEGER, m AS INTEGER
  i = l
  j = r
  srcbin% = -1
  DO WHILE (i <= j)
    m = (i + j) / 2
    IF (x = rij(m)) THEN
      srcbin% = m
      EXIT DO
    END IF
    IF (x < rij(m)) THEN j = m - 1
    IF (x > rij(m)) THEN i = m + 1
  LOOP
END FUNCTION

SUB srtquick (l AS INTEGER, r AS INTEGER)
  'sorteert de lijst volgens de quick sort
  DIM i AS INTEGER, j AS INTEGER
  DIM mrij AS INTEGER
  i = l
  j = r
  mrij = rij(INT((l + r) / 2))
  DO
    DO WHILE (rij(i) < mrij): i = i + 1: LOOP
    DO WHILE (rij(j) > mrij): j = j - 1: LOOP
    IF ((i < j) AND (rij(i) <> rij(j))) THEN SWAP rij(i), rij(j)
    IF (i <= j) THEN i = i + 1
    IF (i <= j) THEN j = j - 1
  LOOP WHILE (i < j)
  IF (j > l) THEN CALL srtquick((l), (j))
  IF (i < r) THEN CALL srtquick((i), (r))
END SUB

SUB toonrij (l AS INTEGER, r AS INTEGER)
  'toont de rij op het scherm
  DIM i AS INTEGER
  FOR i = l TO r
    PRINT USING "## "; rij(i);
  NEXT i
  PRINT
END SUB
```

Andere sorteermethodes

Als laatste programma is hier een toepassing om enkele sorteeralgorithmen te 'tonen' op het scherm. We kunnen hier interactief verschillende opstellingen simuleren:
Sorteren via:

- Bubble Sort
- Insertion Sort
- Quick Sort
- Quick + Insertion Sort

Sorteren van deze soort lijsten:

- gesorteerde lijsten
- omgekeerd gesorteerde lijsten
- willekeurige lijsten
- bijna-gesorteerde lijsten

- upgaande en daarna afgaande lijsten
- afgaande en daarna opgaande lijsten
- lijsten met allemaal dezelfde elementen

We kunnen de diverse lijsten full-speed sorteren of in slow-motion weergeven, zodat je kan zien wat er vergeleken wordt en welke elementen er gewisseld worden.

```
REM program      : qb0504.bas
REM author       : michel gielens
REM date written : 2001-11-07
REM description  : sorteer routines

DECLARE SUB execdelay (pulse AS INTEGER)
DECLARE SUB mkalmost ()
DECLARE SUB mkdownup ()
DECLARE SUB mkequal ()
DECLARE SUB mkforward ()
DECLARE SUB mkrandom ()
DECLARE SUB mkreverse ()
DECLARE SUB mkupdown ()
DECLARE SUB srtbubble (l AS INTEGER, r AS INTEGER)
DECLARE SUB srtinsertion (l AS INTEGER, r AS INTEGER)
DECLARE SUB srtquick (l AS INTEGER, r AS INTEGER)
DECLARE SUB srtquickinsert1 (l AS INTEGER, r AS INTEGER)
DECLARE SUB srtquickinsert2 (l AS INTEGER, r AS INTEGER)
DECLARE SUB status (i AS INTEGER, j AS INTEGER, s AS STRING)
DECLARE SUB toon1rij (i AS INTEGER)

DIM SHARED maxx AS INTEGER, maxy AS INTEGER    'Screen dimensions

REM berekenen scherm afmetingen (maxx en maxy)
DIM sx AS INTEGER, sy AS INTEGER
CLS
maxx = 1
FOR sx = 1 TO 135
  PRINT " ";
  IF (POS(1) > maxx) THEN maxx = POS(1)
NEXT sx
FOR sy = 1 TO 50
  PRINT
NEXT sy
PRINT " ";
maxy = CSRLIN + 1

'Aantal rijen en maximum / rij
DIM SHARED rijen AS INTEGER, maxrij AS INTEGER
rijen = maxx - 17
maxrij = maxy - 2

'Volgende variabelen worden ook in de subroutines gebruikt (SHARED)
DIM SHARED rij(1 TO rijen) AS INTEGER
DIM SHARED delay AS STRING * 1
DIM SHARED compares AS LONG, swaps AS LONG, stack AS INTEGER

'Locale variabelen
DIM i AS INTEGER, choice AS INTEGER

CALL mkrandom   'maakt alvast een willekeurige lijst

delay = "Y"
DO
  CLS
  'verdeel het scherm in 2 stukken
  FOR i = 1 TO maxy - 1
    LOCATE i, 17
    PRINT "|";
  NEXT i
  'toon de ganse lijst
  FOR i = 1 TO rijen
    CALL toon1rij(i)
```

```
    NEXT i
    'initialiseer variabelen
    compares = 0
    swaps = 0
    'toon het menu
    LOCATE 1, 1
    PRINT "Make a list"
    PRINT "1 forward"
    PRINT "2 reverse"
    PRINT "3 random"
    PRINT "4 almost sorted"
    PRINT "5 up & down"
    PRINT "6 down & up"
    PRINT "7 equal"
    PRINT
    PRINT "Sort method"
    PRINT "10 bubble"
    PRINT "11 insertion"
    PRINT "12 quick"
    PRINT "13 quick+insert"
    PRINT
    PRINT "20 delay="; delay
    PRINT
    PRINT "99 end"
    PRINT
    INPUT "Your choice:", choice
    'uitvoeren wat gebruiker wenst
    SELECT CASE choice
      CASE 1: CALL mkforward
      CASE 2: CALL mkreverse
      CASE 3: CALL mkrandom
      CASE 4: CALL mkalmost
      CASE 5: CALL mkupdown
      CASE 6: CALL mkdownup
      CASE 7: CALL mkequal
      CASE 10: CALL srtbubble((1), (rijen))
      CASE 11: CALL srtinsertion((1), (rijen))
      CASE 12: CALL srtquick((1), (rijen))
      CASE 13: CALL srtquickinsert1((1), (rijen))
      CASE 20: IF (delay = "Y") THEN delay = "N" ELSE delay = "Y"
      CASE ELSE
    END SELECT
    'wachten totdat gebuiker op knoppeke drukt
    IF (choice >= 10 AND choice <= 13) THEN
      WHILE INKEY$ = "": WEND
    END IF
  LOOP UNTIL choice = 99

  END

  SUB execdelay (pulse AS INTEGER)
    'wacht een aantal pulsen (1 pulse = 1 / 18.2 seconden)
    DIM i AS INTEGER, timernow AS SINGLE
    FOR i = 1 TO pulse
      timernow = TIMER
      DO WHILE (timernow = TIMER): LOOP
    NEXT i
  END SUB

  SUB mkalmost
    'maakt een bijna gesorteerde lijst
    DIM i AS INTEGER
    FOR i = 1 TO rijen
      rij(i) = CINT(maxrij * i / rijen) + RND * 10 - 5
      IF (rij(i) < 1) THEN rij(i) = 1
      IF (rij(i) > maxrij) THEN rij(i) = maxrij
    NEXT i
  END SUB

  SUB mkdownup
    'maakt een lijst met eerst aflopende en daarna oplopende waarden
    DIM i AS INTEGER
```

```
  FOR i = 1 TO rijen / 2
    rij(i) = (maxrij - CINT(maxrij * i / rijen) * 2) + 2
    rij(rijen - i + 1) = rij(i)
  NEXT i
END SUB

SUB mkequal
  'maakt een lijst met allemaal dezelfde waarden (=1)
  DIM i AS INTEGER
  FOR i = 1 TO rijen
    rij(i) = 1
  NEXT i
END SUB

SUB mkforward
  'maakt een oplopende lijst
  DIM i AS INTEGER
  FOR i = 1 TO rijen
    rij(i) = CINT(maxrij * i / rijen)
  NEXT i
END SUB

SUB mkrandom
  'maakt een lijst met willekeurige waarden
  DIM i AS INTEGER
  FOR i = 1 TO rijen
    rij(i) = INT(RND * maxrij) + 1
  NEXT i
END SUB

SUB mkreverse
  'maakt een lijst met aflopende waarden
  DIM i AS INTEGER
  FOR i = 1 TO rijen
    rij(i) = maxrij - CINT(maxrij * i / rijen) + 1
  NEXT i
END SUB

SUB mkupdown
  'maakt een lijst met eerst oplopende en daarna aflopende waarden
  DIM i AS INTEGER
  FOR i = 1 TO rijen / 2
    rij(i) = CINT(maxrij * i / rijen) * 2
    rij(rijen - i + 1) = rij(i)
  NEXT i
END SUB

SUB srtbubble (l AS INTEGER, r AS INTEGER)
  'sorteert de lijst volgens de bubble-sort
  DIM i AS INTEGER, swapped AS INTEGER
  DO
    swapped = 0
    FOR i = l TO r - 1
      CALL status(i, i + 1, "C")
      IF (rij(i) > rij(i + 1)) THEN
        CALL status(i, i + 1, "S")
        SWAP rij(i), rij(i + 1)
        swapped = 1
        CALL toon1rij(i)
        CALL toon1rij(i + 1)
      END IF
    NEXT i
    r = r - 1
  LOOP WHILE (swapped)
  CALL status(1, 1, " ")
END SUB

SUB srtinsertion (l AS INTEGER, r AS INTEGER)
  'sorteert de lijse volgense de insertion sort
  DIM i AS INTEGER, j AS INTEGER, swapped AS INTEGER
  FOR i = l + 1 TO r
    FOR j = i - 1 TO l STEP -1
```

```
      CALL status(j, j + 1, "C")
      IF (rij(j) > rij(j + 1)) THEN
        CALL status(j, j + 1, "S")
        SWAP rij(j), rij(j + 1)
        swapped = 1
        CALL toon1rij(j)
        CALL toon1rij(j + 1)
      ELSE
        swapped = 0
      END IF
      CALL status(j, j + 1, " ")
      IF (swapped = 0) THEN EXIT FOR
    NEXT j
  NEXT i
END SUB

SUB srtquick (l AS INTEGER, r AS INTEGER)
  'sorteert de lijst volgens de quick sort
  DIM i AS INTEGER, j AS INTEGER
  DIM mrij AS INTEGER
  stack = stack + 1
  LOCATE 24, 1: PRINT "stack   ="; stack;
  i = l
  j = r
  mrij = rij(INT((l + r) / 2))
  DO
    CALL status((i), (i), "C")
    DO WHILE (rij(i) < mrij)
      CALL status((i), (i), "C")
      i = i + 1
    LOOP
    CALL status((j), (j), "C")
    DO WHILE (rij(j) > mrij)
      CALL status((j), (j), "C")
      j = j - 1
    LOOP
    IF ((i < j) AND (rij(i) <> rij(j))) THEN
      CALL status((i), (j), "S")
      SWAP rij(i), rij(j)
      CALL toon1rij(i)
      CALL toon1rij(j)
    END IF
    IF (i <= j) THEN i = i + 1
    IF (i <= j) THEN j = j - 1
  LOOP WHILE (i < j)
  CALL status(1, 1, " ")
  IF (j > l) THEN CALL srtquick((l), (j))
  IF (i < r) THEN CALL srtquick((i), (r))
  stack = stack - 1
  LOCATE 24, 1: PRINT "stack   ="; stack;
END SUB

SUB srtquickinsert1 (l AS INTEGER, r AS INTEGER)
  'sorteert de lijst volgens de quick sort (maar slechts tot linker
  'en rechter grens 10 elementen van elkaar verwijderd zijn), en
  'daarna wordt de rest gesorteerd volgens de insertion sort
  CALL srtquickinsert2((l), (r))
  CALL srtinsertion((l), (r))
END SUB

SUB srtquickinsert2 (l AS INTEGER, r AS INTEGER)
  'sorteert de lijst volgens de quick sort (totdat linker en
  'rechter grens 10 elementen van elkaar verwijderd zijn).
  'maakt dus een almost-sorted lijst.
  DIM i AS INTEGER, j AS INTEGER
  DIM mrij AS INTEGER
  stack = stack + 1
  LOCATE 24, 1: PRINT "stack   ="; stack;
  IF (r - l > 9) THEN
    i = l
    j = r
    mrij = rij(INT((l + r) / 2))
```

```
    DO
      CALL status((i), (i), "C")
      DO WHILE (rij(i) < mrij)
        CALL status((i), (i), "C")
        i = i + 1
      LOOP
      CALL status((j), (j), "C")
      DO WHILE (rij(j) > mrij)
        CALL status((j), (j), "C")
        j = j - 1
      LOOP
      IF ((i < j) AND (rij(i) <> rij(j))) THEN
        CALL status((i), (j), "S")
        SWAP rij(i), rij(j)
        CALL toon1rij(i)
        CALL toon1rij(j)
      END IF
      IF (i <= j) THEN i = i + 1
      IF (i <= j) THEN j = j - 1
    LOOP WHILE (i < j)
    CALL status(1, 1, " ")
    IF (j > l) THEN CALL srtquickinsert2((l), (j))
    IF (i < r) THEN CALL srtquickinsert2((i), (r))
  END IF
  stack = stack - 1
  LOCATE 24, 1: PRINT "stack   ="; stack;
END SUB

SUB status (i AS INTEGER, j AS INTEGER, s AS STRING)
  'toont de tussentijdse statussen
  's="C" : er wordt een 'compare' gedaan
  's="S" : er worden elementen 'geswapt'
  'toont boven op het scherm een "S" of een "C"
  'toont aan de linker kant het aantal swaps en compares
  'wacht eventjes, want anders kunnen we zelf niet meer volgen
  STATIC previ AS INTEGER, prevj AS INTEGER
  DIM pulses AS INTEGER
  SELECT CASE s
    CASE "C"
      compares = compares + 1
      COLOR 10
      pulses = 2
    CASE "S"
      swaps = swaps + 1
      COLOR 12
      pulses = 4
    CASE ELSE
  END SELECT
  IF (previ > 0 AND prevj > 0) THEN
    LOCATE 1, 17 + previ: PRINT " ";
    LOCATE 1, 17 + prevj: PRINT " ";
  END IF
  LOCATE 1, 17 + i: PRINT s;
  LOCATE 1, 17 + j: PRINT s;
  previ = i
  prevj = j
  COLOR 7
  LOCATE 22, 1: PRINT "compares="; compares;
  LOCATE 23, 1: PRINT "swaps   ="; swaps;
  IF (delay = "Y") THEN execdelay (pulses)
END SUB

SUB toon1rij (i AS INTEGER)
  'toont (hertekent) 1 rijtje met allemaal I-s
  DIM y AS INTEGER
  COLOR 7
  FOR y = 1 TO maxrij
    LOCATE maxrij - y + 2, 17 + i
    IF (rij(i) >= y) THEN PRINT "I";  ELSE PRINT " ";
  NEXT y
END SUB
```

Analoog klokje

Ter afsluiting van deze cursus is hier nog een analoog klokje zodat u nooit de tijd uit het oog verliest als u met uw computer aan het werken bent.

```
REM program      : qb0505.bas
REM author       : michel gielens
REM date written : 2001-12-09
REM description  : analoge klok

DECLARE SUB wijzer (kleur AS INTEGER, grootte AS INTEGER, r AS
INTEGER)

DIM SHARED pi AS SINGLE
DIM h AS INTEGER, m AS INTEGER, s AS INTEGER
DIM ho AS INTEGER, mo AS INTEGER, so AS INTEGER
DIM hn AS INTEGER, mn AS INTEGER, sn AS INTEGER
DIM r AS SINGLE

pi = ATN(1) * 4

SCREEN 12
WINDOW (-133, 100)-(133, -100)

' teken de klok
CIRCLE (0, 0), 90
' teken de minuten
FOR r = 0 TO 2 * pi STEP 2 * pi / 60
  PSET (95 * COS(r), 95 * SIN(r))
NEXT r
' teken de uren
FOR r = 0 TO 2 * pi STEP 2 * pi / 12
  CIRCLE (95 * COS(r), 95 * SIN(r)), 2
NEXT r

DO WHILE (INKEY$ = "")
  timenow$ = TIME$
  DO WHILE (timenow$ = TIME$): LOOP
  h = VAL(MID$(timenow$, 1, 2))
  m = VAL(MID$(timenow$, 4, 2))
  s = VAL(MID$(timenow$, 7, 2))
  IF (h >= 12) THEN h = h - 12
  hn = 360 / 12 * h + m \ 2
  mn = 360 / 60 * m + s \ 10
  sn = 360 / 60 * s
  ' wissen oude tijd
  CALL wijzer(0, 40, ho)
  CALL wijzer(0, 60, mo)
  CALL wijzer(0, 80, so)
  ' teken nieuwe tijd
  CALL wijzer(15, 40, hn)
  CALL wijzer(15, 60, mn)
  CALL wijzer(15, 80, sn)
  ho = hn
  mo = mn
  so = sn
LOOP

SCREEN 0
WIDTH 80

END

SUB wijzer (kleur AS INTEGER, grootte AS INTEGER, r AS INTEGER)
  DIM s AS SINGLE, c AS SINGLE
  s = SIN((r - 90) / 180 * pi * -1)
  c = COS((r - 90) / 180 * pi * -1)
  LINE (0, 0)-(grootte * c, grootte * s), kleur
END SUB
```

Oplossingen

oefening 1

DEZE HOND WAS GISTEREN VERVELEND.
DEZE HOND WAS GISTEREN ERG VERVELEND.
DEZE HOND WAS GISTEREN KNAP VERVELEND.
DEZE HOND IS VANDAAG VERVELEND.
DEZE HOND IS VANDAAG ERG VERVELEND.
DEZE HOND IS VANDAAG KNAP VERVELEND.
DEZE HOND WORDT MORGEN VERVELEND.
DEZE HOND WORDT MORGEN ERG VERVELEND.
DEZE HOND WORDT MORGEN KNAP VERVELEND.
DEZE HOND LIJKT VERVELEND.
DEZE HOND LIJKT ERG VERVELEND.
DEZE HOND LIJKT KNAP VERVELEND.

oefening 2

CIRCLE (x!,y!),radius!
CIRCLE (x!,y!),radius!,color%
CIRCLE (x!,y!),radius!,,start!
CIRCLE (x!,y!),radius!,color%,start!
CIRCLE (x!,y!),radius!,,,end!
CIRCLE (x!,y!),radius!,color%,,end!
CIRCLE (x!,y!),radius!,,start!,end!
CIRCLE (x!,y!),radius!,color%,start!,end!
CIRCLE (x!,y!),radius!,,,,aspect!
CIRCLE (x!,y!),radius!,color%,,,aspect!
CIRCLE (x!,y!),radius!,,start!,,aspect!
CIRCLE (x!,y!),radius!,color%,start!,,aspect!
CIRCLE (x!,y!),radius!,,,end!,aspect!
CIRCLE (x!,y!),radius!,color%,,end!,aspect!
CIRCLE (x!,y!),radius!,,start!,end!,aspect!
CIRCLE (x!,y!),radius!,color%,start!,end!,aspect!
CIRCLE STEP (x!,y!),radius!
CIRCLE STEP (x!,y!),radius!,color%
CIRCLE STEP (x!,y!),radius!,,start!
CIRCLE STEP (x!,y!),radius!,color%,start!
CIRCLE STEP (x!,y!),radius!,,,end!
CIRCLE STEP (x!,y!),radius!,color%,,end!
CIRCLE STEP (x!,y!),radius!,,start!,end!
CIRCLE STEP (x!,y!),radius!,color%,start!,end!
CIRCLE STEP (x!,y!),radius!,,,,aspect!
CIRCLE STEP (x!,y!),radius!,color%,,,aspect!
CIRCLE STEP (x!,y!),radius!,,start!,,aspect!
CIRCLE STEP (x!,y!),radius!,color%,start!,,aspect!
CIRCLE STEP (x!,y!),radius!,,,end!,aspect!
CIRCLE STEP (x!,y!),radius!,color%,,end!,aspect!
CIRCLE STEP (x!,y!),radius!,,start!,end!,aspect!
CIRCLE STEP (x!,y!),radius!,color%,start!,end!,aspect!

oefening 3

1)

```
REM program      : oef3_1.bas
REM author       : michel gielens
REM date written : 2001-10-14
REM description  : onderdrukken voorloopnullen

DIM getal AS STRING, letter AS STRING * 1, drukken AS STRING * 1
DIM i AS INTEGER

INPUT "Geef een getal, voorafgegaan door nullen : ", getal
drukken = "N"
FOR i = 1 TO LEN(getal)
  letter = MID$(getal, i, 1)
  IF (letter <> "0") THEN drukken = "Y"
  IF (drukken = "Y") THEN PRINT letter;
NEXT i
PRINT

END
```

2)

```
REM program      : oef3_2.bas
REM author       : michel gielens
REM date written : 2001-10-14
REM description  : coderen geheimschrift

DIM zin AS STRING, letter AS STRING * 1
DIM i AS INTEGER

CLS
INPUT "Geef een zin : ", zin
FOR i = 1 TO LEN(zin)
  letter = MID$(zin, i, 1)
  PRINT letter;
  IF (INSTR("aeiouyAEIOUY", letter) > 0) THEN PRINT "p"; letter;
NEXT i

END
```

3)

```
REM program      : oef3_3.bas
REM author       : michel gielens
REM date written : 2001-10-14
REM description  : decoderen geheimtaal

DIM zin AS STRING, letter AS STRING * 1
DIM lengte AS INTEGER, i AS INTEGER

INPUT "Geef de gecodeerde zin : ", zin
lengte = LEN(zin)
FOR i = 1 TO lengte
  letter = MID$(zin, i, 1)
  PRINT letter;
  ' is het een klinker ?
  IF (INSTR("aeiouyAEIOUY", letter) > 0) THEN
    ' zijn er nog tenminste 2 tekens ?
    IF (i <= lengte - 2) THEN
      ' is de volgende letter een "p" en de daaropvolgende dezelfde
klinker ?
      IF (MID$(zin, i + 1, 1) = "p" AND MID$(zin, i + 2, 1) =
letter) THEN
        ' gewoon 2 tekens overslagen
        i = i + 2
      END IF
    END IF
  END IF
NEXT i

END
```

4)

```
REM program      : oef3_4.bas
```

```
REM author       : michel gielens
REM date written : 2001-10-14
REM description  : schuin afdrukken

DIM woord(0 TO 3) AS STRING
DIM maxlengte AS INTEGER, i AS INTEGER, w AS INTEGER

CLS

woord(0) = "Computer"
woord(1) = "Club"
woord(2) = "Masano"
woord(3) = "Schaffen"

maxlengte = 0
FOR w = 0 TO 3
  IF (LEN(woord(w)) > maxlengte) THEN
    maxlengte = LEN(woord(w))
  END IF
NEXT w

FOR i = 1 TO maxlengte
  FOR w = 0 TO 3
    IF (LEN(woord(w)) >= i) THEN
      PRINT TAB(i + 5 * w); MID$(woord(w), i, 1);
    END IF
  NEXT w
  PRINT
NEXT i

END
```

oefening 4

```
REM program      : oef4.bas
REM author       : michel gielens
REM date written : 2001-10-14
REM description  : controle getal

DIM getal AS STRING, letter AS STRING * 1
DIM teken AS STRING * 1, komma AS STRING * 1, decimaal AS STRING * 1
DIM fout AS STRING * 1
DIM i AS INTEGER

CLS
INPUT "Geef een getal : ", getal

teken = "N"
komma = "N"
decimaal = "N"
fout = "N"

IF (LEN(getal) = 0) THEN
  fout = "Y"
END IF

FOR i = 1 TO LEN(getal)
  letter = MID$(getal, i, 1)
  SELECT CASE (letter)
    CASE "+", "-"
      IF (teken = "Y") THEN
        PRINT "Error : + of - mag maar 1 keer voorkomen."
        fout = "Y"
      ELSEIF (i > 1) THEN
        PRINT "Error : + of - moet eerste teken zijn."
        fout = "Y"
      ELSE
        teken = "Y"
      END IF
    CASE ",", "."
      IF (komma = "Y") THEN
```

```
          PRINT "Error : komma mag maar 1 keer voorkomen."
          fout = "Y"
        ELSE
          komma = "Y"
        END IF
      CASE "0" TO "9"
        decimaal = "Y"
      CASE ELSE
        PRINT "Error : '"; letter; "' is een ongeldig teken."
        fout = "Y"
    END SELECT
  NEXT i

  PRINT
  IF (fout = "N") THEN
    PRINT getal; " lijkt me een goed getal."
  ELSE
    PRINT getal; " lijkt me een geen goed getal."
  END IF

  END
```

oefening 5

```
  REM program      : oef5.bas
  REM author       : michel gielens
  REM date written : 2001-10-14
  REM description  : ingeven paswoord

  REM Opgelet : dit is niet de manier voor een goede
  paswoordbeveiling.
  REM Dit is slechts een voorbeeld van hoe je ermee kunt beginnen.

  DIM paswoord AS STRING
  DIM lengte AS INTEGER

  DO
    CLS
    LOCATE 12, 25
    PRINT "Geef het paswoord : ";
    paswoord = ""
    lengte = 0
    DO
      DO
        letter$ = INKEY$
      LOOP UNTIL (LEN(letter$) = 1)
      SELECT CASE (letter$)
        CASE CHR$(8)   'backspace
          IF (lengte > 0) THEN
            lengte = lengte - 1
            paswoord = LEFT$(paswoord, lengte)
            LOCATE 12, 45
            PRINT STRING$(lengte, "*"); " ";
          END IF
        CASE CHR$(13)  'enter
        CASE ELSE
          IF (lengte < 10) THEN
            lengte = lengte + 1
            paswoord = paswoord + letter$
            LOCATE 12, 45
            PRINT STRING$(lengte, "*");
          END IF
      END SELECT
    LOOP UNTIL (letter$ = CHR$(13))
  LOOP UNTIL (paswoord = "GeHeiM")

  END
```

oefening 6

```
REM program      : oef6.bas
REM author       : michel gielens
REM date written : 2001-11-04
REM description  : ingeven paswoord via een fnctie

REM Opgelet : dit is niet de manier voor een goede
paswoordbeveiling.
REM Dit is slechts een voorbeeld van hoe je ermee kunt beginnen.

DECLARE FUNCTION passwd$ (x AS INTEGER, y AS INTEGER, l AS INTEGER)

DIM paswoord AS STRING

DO
  CLS
  LOCATE 12, 25
  PRINT "Geef het paswoord : ";
  paswoord = passwd$(POS(1), CSRLIN, 10)
LOOP UNTIL (paswoord = "GeHeiM")

END

FUNCTION passwd$ (x AS INTEGER, y AS INTEGER, l AS INTEGER)
  DIM paswoord AS STRING
  DIM lengte AS INTEGER
  LOCATE y, x
  paswoord = ""
  lengte = 0
  DO
    DO
      letter$ = INKEY$
    LOOP UNTIL (LEN(letter$) = 1)
    SELECT CASE (letter$)
      CASE CHR$(8)   'backspace
        IF (lengte > 0) THEN
          lengte = lengte - 1
          paswoord = LEFT$(paswoord, lengte)
          LOCATE y, x
          PRINT STRING$(lengte, "*"); " ";
        END IF
      CASE CHR$(13)  'enter
      CASE ELSE
        IF (lengte < l) THEN
          lengte = lengte + 1
          paswoord = paswoord + letter$
          LOCATE y, x
          PRINT STRING$(lengte, "*");
        END IF
    END SELECT
  LOOP UNTIL (letter$ = CHR$(13))
  passwd$ = paswoord
END FUNCTION
```

oefening 7

```
REM program      : oef7.bas
REM author       : michel gielens
REM date written : 2001-10-23
REM description  : aflijsten autoexec.bat & config.sys

CLS

PRINT "Inhoud van AUTOEXEC.BAT"
PRINT "-----------------------"
OPEN "c:\autoexec.bat" FOR INPUT AS #1
DO UNTIL EOF(1)
  LINE INPUT #1, regel$
  PRINT regel$
LOOP
CLOSE #1
```

```
PRINT
PRINT "Inhoud van CONFIG.SYS"
PRINT "---------------------"
OPEN "c:\config.sys" FOR INPUT AS #1
DO UNTIL EOF(1)
  LINE INPUT #1, regel$
  PRINT regel$
LOOP
CLOSE #1

END
```

oefening 8

```
REM program      : oef8.bas
REM author       : michel gielens
REM date written : 2001-10-27
REM description  : random

DECLARE SUB wacht ()

TYPE arttype
  status AS STRING * 1
  nr AS INTEGER
  naam AS STRING * 20
  prijs AS INTEGER
END TYPE

DIM artrecord AS arttype

DIM i AS INTEGER, aantrecs AS INTEGER

ON ERROR GOTO errorhandler1
KILL "art.dat"
ON ERROR GOTO 0

OPEN "art.dat" FOR RANDOM AS #1 LEN = LEN(artrecord)

CLS
PRINT "Opvullen van het artikelen bestand."
READ naam$, prijs
i = 0
DO WHILE (naam$ <> "*")
  i = i + 1
  artrecord.status = " "
  artrecord.nr = i
  artrecord.naam = naam$
  artrecord.prijs = prijs
  PUT #1, , artrecord
  READ naam$, prijs
LOOP
CALL wacht

CLS
PRINT "Huidig artikelen bestand:"
PRINT "S nr artikel naam         prijs"
SEEK #1, 1
GET #1, , artrecord
DO UNTIL EOF(1)
  PRINT USING "! ## &  ###"; artrecord.status; artrecord.nr;
artrecord.naam; artrecord.prijs
  GET #1, , artrecord
LOOP
CALL wacht

aantrecs = LOF(1) / LEN(artrecord)

' wis artikel 14
IF (aantrecs >= 14) THEN
  GET #1, 14, artrecord
  artrecord.status = "*"
```

```
  PUT #1, 14, artrecord
END IF

' wis artikel 8
IF (aantrecs >= 8) THEN
  GET #1, 8, artrecord
  artrecord.status = "*"
  PUT #1, 8, artrecord
END IF

CLS
PRINT "Huidig artikelen bestand na wissen van record 8 en 14:"
PRINT "S nr artikel naam         prijs"
SEEK #1, 1
GET #1, , artrecord
DO UNTIL EOF(1)
  PRINT USING "! ## &  ###"; artrecord.status; artrecord.nr;
artrecord.naam; artrecord.prijs
  GET #1, , artrecord
LOOP
CALL wacht

CLOSE #1

' reorganisatie bestand
ON ERROR GOTO errorhandler1
KILL "art.tmp"
ON ERROR GOTO 0
OPEN "art.dat" FOR RANDOM AS #1 LEN = LEN(artrecord)
OPEN "art.tmp" FOR RANDOM AS #2 LEN = LEN(artrecord)
GET #1, , artrecord
DO UNTIL EOF(1)
  IF (artrecord.status = " ") THEN PUT #2, , artrecord
  GET #1, , artrecord
LOOP
CLOSE #1
CLOSE #2
KILL "art.dat"
NAME "art.tmp" AS "art.dat"

CLS
OPEN "art.dat" FOR RANDOM AS #1 LEN = LEN(artrecord)
PRINT "Huidig artikelen bestand na reorganisatie"
PRINT "S nr artikel naam         prijs"
GET #1, , artrecord
DO UNTIL EOF(1)
  PRINT USING "! ## &  ###"; artrecord.status; artrecord.nr;
artrecord.naam; artrecord.prijs
  GET #1, , artrecord
LOOP
CLOSE #1
CALL wacht

END

errorhandler1:
RESUME NEXT

DATA lelie,215
DATA krokus,199
DATA tulp,199
DATA hyacint,195
DATA sneeuwklokje,249
DATA narcis,595
DATA iris,187
DATA anemoon,199
DATA gladiool,239
DATA cactus,695
DATA pampasgras,275
DATA lupine,199
DATA ridderspoor,299
DATA pioenroos,395
```

```
DATA margriet,299
DATA zijdeplant,257
DATA anjer,269
DATA strobloem,299
DATA lavendel,295
DATA bruidssluier,495
DATA trompetbloem,395
DATA lelietje-van-dalen,495
DATA zeepkruid,289
DATA *,0

SUB wacht
  COLOR 9
  PRINT "Druk op toets om verder te gaan...";
  COLOR 7
  DO WHILE INKEY$ = ""
  LOOP
END SUB
```