

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ  
ΜΕ  
**FreeBASIC**



- ΔΗΜΟΣΘΕΝΗΣ ΚΟΠΤΣΗΣ -



**ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ**  
**ΜΕ**  
**FreeBASIC**



**ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ  
ΜΕ  
FreeBASIC**

Το βιβλίο αυτό το αφιερώνω στους  
αγαπημένους μου γονείς



Copyright © 2021 Δημοσθένης Κόπτης

Διατηρούνται ορισμένα δικαιώματα.

Το έργο αυτό αδειοδοτείται υπό την Άδεια Creative Commons Αναφορά προέλευσης 4.0 (Atribution-4.0).

Προκειμένου να δείτε ένα αντίγραφο αυτής της άδειας, δείτε το Παράρτημα Β, ή επισκεφτείτε τη διεύθυνση

<https://creativecommons.org/licenses/by/4.0/legalcode.el>

ή στείλτε γράμμα στην Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.



# Πίνακας Περιεχομένων

Εισαγωγή.....	17
Εγκατάσταση.....	17
Για Windows.....	17
Για Linux.....	18
Για το DOS.....	19
Εγκατάσταση IDE.....	19
Εγκατάσταση του poseidonFB.....	20
Ρυθμίζοντας το poseidonFB.....	21
Συμβάσεις.....	23
Μια σύντομη ανασκόπηση στις γλώσσες προγραμματισμού. .....	25
Κεφάλαιο 1ο - Το πρώτο μου πρόγραμμα.....	27
Ανάλυση.....	28
Κεφάλαιο 2ο - Τα βασικά μέρη της γλώσσας.....	29
Αλφάβητο και Λεξιλόγιο της γλώσσας προγραμματισμού. .....	29
Σύνταξη της γλώσσας.....	30
Σχόλια μέσα σε ένα πρόγραμμα.....	30
Κεφάλαιο 3ο - Μεταβλητές.....	32
Τύποι δεδομένων (Datatypes).....	33
Δεκαδικοί, δεκαεξαδικοί, οκταδικοί, δυαδικοί ακέραιοι. .	36
Δηλώνοντας Μεταβλητές.....	37
Αρχικοποίηση Μεταβλητών κατά την δήλωση τους.....	39
Εμβέλεια μεταβλητών.....	39
Local Scope.....	40
Shared Scope.....	41
Common Scope.....	42
Common Shared.....	43
Ανακύκλωση μεταβλητών.....	45
Ειδικοί χαρακτήρες συμβολοσειρών (strings).....	47
Μετατροπή τιμών μεταβλητών.....	49
CBool.....	50
CByte.....	51
CUByte.....	52
CShort.....	52
CUShort.....	53
CInt.....	54
CUInt.....	55

CLng.....	55
CULng.....	56
CLngInt.....	57
CULngInt.....	58
CSng.....	59
CDbl.....	59
Str.....	60
Val.....	61
Συμβολοσειρές (Strings) στη FreeBASIC.....	62
String.....	62
ZString.....	63
WString.....	64
Κεφάλαιο 3ο - Σταθερές.....	65
Const.....	65
Κεφάλαιο 4ο - Προτάσεις και εκφράσεις.....	67
Προτάσεις (statements).....	67
Μπλοκ κώδικα.....	68
Εκφράσεις (expressions).....	68
Κεφάλαιο 5ο - Πίνακες.....	69
Στοιχεία και θέσεις.....	69
Δηλώνοντας Πίνακες.....	70
Πίνακες σταθερού και μεταβλητού μεγέθους.....	72
Πίνακες πολλαπλών διαστάσεων.....	72
Οι συναρτήσεις LBound και UBound.....	74
LBound.....	74
UBound.....	74
Κεφάλαιο 6ο - Ροή προγράμματος.....	75
Εντολές λήψης λογικών αποφάσεων.....	75
If.....	75
IF ... THEN ... ELSE IF ... THEN ... ELSE ... ENDIF.....	77
Select Case ... End Select.....	80
Εντολές βρόχων.....	81
For...Next.....	82
Λίγα λόγια για τις ένθετες προτάσεις.....	83
Do...Loop.....	84
While ... Wend.....	86
Εντολή μετάβασης Goto.....	86
Διαχείριση λαθών.....	88
Εντολές END, RETURN, STOP, SLEEP, EXIT.....	90
End.....	91



Return.....	91
Stop.....	92
Sleep.....	92
Exit.....	92
Κεφάλαιο 7ο - Ρουτίνες, συναρτήσεις, διαδικασίες.....	93
SUB.....	93
Function.....	96
Αναδρομή συναρτήσεων.....	99
Κεφάλαιο 8ο Δείκτες.....	102
Η μνήμη του υπολογιστή.....	102
Δήλωση δεικτών σε μεταβλητές.....	105
Δήλωση δεικτών σε διαδικασίες.....	106
Η συνάρτηση Allocate.....	108
Αριθμητική Δεικτών.....	110
Πρόσθεση και αφαίρεση από τους δείκτες.....	110
Αύξηση και μείωση δεικτών.....	111
Προβλήματα με τους δείκτες.....	111
Διαρροές μνήμης (Memory Leak).....	111
Αλλοίωση δεδομένων στην μνήμη (Memory Corruption)	
.....	112
Περιπλανώμενοι δείκτες (dangling pointers).....	112
Μη απελευθέρωση δείκτη οδηγεί σε διαρροή μνήμης..	114
Άλλα προγραμματιστικά λάθη με τους δείκτες.....	115
Κεφάλαιο 9ο - Τελεστές.....	116
Τελεστές εκχώρησης.....	117
Τελεστής = [=] (Assignment).....	117
Τελεστής &= (Concatenate And Assign).....	117
Τελεστής += (Add And Assign).....	118
Τελεστής -= (Subtract And Assign).....	118
Τελεστής *= (Multiply And Assign).....	119
Τελεστής /= (Divide And Assign).....	119
Τελεστής \= (Integer Divide And Assign).....	120
Τελεστής ^= (Exponentiate And Assign).....	120
Τελεστής Let (Assignment).....	121
Τελεστής Let() (Assignment).....	121
Τελεστής Mod= (Modulus And Assign).....	121
Τελεστής And= (Conjunction And Assign).....	122
Τελεστής Eqv= (Equivalence And Assign).....	122
Τελεστής Imp= (Implication And Assign).....	123
Τελεστής Or= (Inclusive Disjunction And Assign).....	123

Τελεστής Xor= (Exclusive Disjunction And Assign)....	123
Τελεστής Shl= (Shift Left And Assign).....	124
Τελεστής Shr= (Shift Right And Assign).....	124
Αριθμητικοί τελεστές.....	125
Τελεστής + (Add).....	125
Τελεστής - (Subtract).....	125
Τελεστής * (Multiply).....	126
Τελεστής / (Divide).....	126
Τελεστής \ (Integer Divide).....	127
Τελεστής ^ (Exponentiate).....	127
Τελεστής Mod (Modulus).....	128
Τελεστής - (Negate).....	128
Τελεστής Shl (Shift Left).....	129
Τελεστής Shr (Shift Right).....	130
Σχεσιακοί τελεστές.....	130
Τελεστής = (Equal).....	131
Τελεστής <> (Not Equal).....	131
Τελεστής < (Less Than).....	132
Τελεστής <= (Less Than Or Equal).....	132
Τελεστής >= (Greater Than Or Equal).....	132
Τελεστής > (Greater Than).....	133
Τελεστής Is (Run-Time Type Information).....	133
Λογικοί τελεστές.....	134
Τελεστές Short Circuit.....	137
Τελεστής Andalso (Short Circuit Conjunction).....	137
Τελεστής Orelse (Short Circuit Inclusive Disjunction).....	138
Τελεστές indexing, θέσης.....	139
Τελεστής () (Array Index).....	139
Τελεστής [] (String Index).....	140
Τελεστής [] (Pointer Index).....	141
Τελεστές συμβολοσειρών.....	141
Τελεστής + (String Concatenation).....	141
Τελεστής & (String Concatenation With Conversion).....	142
Τελεστές προεπεξεργαστή.....	143
Τελεστής # (Stringize).....	143
Τελεστής ## (Concatenation).....	144
Τελεστής ! (Escaped String Literal).....	144
Τελεστής \$ (Non-Escaped String Literal).....	145
Τελεστές δεικτών.....	146
Τελεστής Varptr (Variable Pointer).....	146

Τελεστής Strptr (String Pointer).....	146
Τελεστής @ (Address Of).....	148
Τελεστής * (Value Of).....	149
Τελεστές επαναληπτών.....	150
Τελεστές For, Next, Step.....	150
Τελεστές UDT.....	150
Τελεστές . (Member Access), -> (Pointer To Member Access), Is (Run-Time Type Information).....	150
Τελεστές Μνήμης.....	150
Τελεστές New Expression, New Overload, Placement New, Delete Statement, Delete Overload.....	150
Κεφάλαιο 10ο - Τύποι Χρηστών - UDT (User Defined Types) .....	151
Δήλωση UDT.....	151
Μέλη UDT.....	153
Δείκτες σε UDT.....	155
Constructors και Destructors, συναρτήσεις Δόμησης και Αποδόμησης.....	156
Δήλωση συναρτήσεων Δόμησης και Αποδόμησης.....	156
Μέλη Ρουτίνες.....	160
Δήλωση και ορισμός.....	160
Η λέξη κλειδί This.....	161
Υπερφόρτωση μελών ρουτινών.....	164
Ιδιότητες - Properties.....	165
Υπερφόρτωση setters.....	167
Δικαιώματα πρόσβασης μελών και ευθυλάκωση (Member Access Rights and Encapsulation).....	168
Public μέλη.....	169
Protected μέλη.....	169
Private μέλη.....	169
Συναρτήσεις Δόμησης και Αποδόμησης, Constructors και destructors.....	169
Ευθυλάκωση.....	170
Κληρονομικότητα και Πολυμορφισμός (Inheritance Polymorphism).....	172
Η λέξη-κλειδί Base (Member Access).....	174
Ο τελεστής Is (Run-Time Type Information).....	177
Virtual και Abstract μέθοδοι.....	180
Abstract μέθοδοι.....	183
Κεφάλαιο 11ο - Run Time Functions.....	185

Συναρτήσεις πινάκων - Array Functions.....	185
ReDim.....	185
Preserve.....	187
Erase.....	189
LBound και UBound.....	191
Διαχείριση bit - Bit Manipulation.....	192
LoByte.....	192
HiByte.....	192
LoWord και HiWord.....	193
Bit.....	194
BitReset.....	195
BitSet.....	196
Συναρτήσεις κοσνόλας - Console Functions.....	197
Cls.....	197
Width.....	198
View Print.....	199
Color.....	200
CsrLin.....	201
Pos.....	201
Locate.....	201
Screen.....	202
Print   ?.....	203
Write.....	203
Spc.....	204
Tab.....	205
Συναρτήσεις ημερομηνίας και ώρας - Date and Time Functions.....	206
Now.....	206
DateSerial.....	208
TimeSerial.....	209
DateValue.....	210
TimeValue.....	212
Second.....	213
Minute.....	214
Hour.....	215
Day.....	216
Weekday.....	217
Month.....	218
Year.....	219
DatePart.....	220

DateAdd.....	223
DateDiff.....	224
IsDate.....	227
MonthName.....	228
WeekdayName.....	229
Date.....	231
Time.....	232
SetDate.....	232
SetTime.....	233
Timer.....	233
Συναρτήσεις διαχείρισης λαθών - Error Handling	
Functions.....	235
Erl.....	235
Erfn.....	236
Ermn.....	237
Err.....	238
Error.....	239
On Error.....	240
Resume.....	241
Resume Next.....	242
Συναρτήσεις εισόδου/εξόδου αρχείων - File IO Functions	
.....	243
Άνοιγμα αρχείων ή συσκευών.....	244
FreeFile.....	244
Open.....	246
Open Com.....	250
Open Cons.....	254
Open Err.....	256
Open Lpt.....	257
Open Pipe.....	259
Open Scrn.....	263
Close.....	264
Reset.....	266
Καταστάσεις I/O αρχείου.....	268
INPUT (FILE MODE).....	268
Output.....	273
Append.....	276
Binary.....	279
Random.....	282
Δικαιώματα πρόσβασης αρχείων.....	286

Access.....	286
Read (File Access).....	287
Write (File Access).....	287
Read Write (File Access).....	288
Κωδικοποίηση χαρακτήρων.....	288
Encoding.....	288
Ανάγνωση και εγγραφή σε αρχεία ή συσκευές.....	289
Input #.....	289
Write #.....	291
Input().....	293
Winput().....	294
Line Input #.....	295
(Print   ?) #.....	296
Put (File I/O).....	298
Get (File I/O).....	302
Θέση αρχείου και άλλες πληροφορίες.....	304
LOF.....	304
LOC.....	305
EOF.....	306
Seek (Statement).....	308
Seek (Function).....	309
Lock και Unlock.....	310
Μαθηματικές συναρτήσεις - Mathematical Functions... 312	312
Αλγεβρικές Συναρτήσεις.....	312
Τριγωνομετρικές Συναρτήσεις.....	314
Συναρτήσεις Τυχαίων Αριθμών.....	315
Randomize.....	315
Rnd.....	318
Συναρτήσεις μνήμης - Memory Functions.....	320
Allocate.....	320
CAllocate.....	323
Reallocate.....	324
Deallocate.....	327
Peek.....	329
Poke.....	330
Clear.....	332
fb_memcopy.....	334
fb_MemCopyClear.....	336
fb_memmove.....	338
Swap.....	340

SAdd.....	341
Συναρτήσεις λειτουργικού συστήματος - Operating System Functions.....	343
Exec.....	343
Chain.....	344
Run.....	345
Kill.....	346
Name.....	347
FileAttr.....	348
FileCopy.....	351
FileDateTime.....	352
FileExists.....	354
FileLen.....	355
FileSetEof.....	356
FileFlush.....	358
Συναρτήσεις συμβολοσειρών - String Functions.....	360
String (Function).....	360
Wstring (Function).....	361
Space.....	363
WSpace.....	364
Len.....	365
Asc.....	367
Chr.....	368
WChr.....	370
Bin.....	371
WBin.....	373
Hex.....	374
WHex.....	376
Oct.....	377
WOct.....	378
Str.....	380
WStr.....	381
Format.....	383
Val.....	388
ValInt.....	390
ValLng.....	391
ValUInt.....	393
ValULng.....	394
MKD.....	396
MKI.....	397

MKL.....	398
MKLongInt.....	399
MKS.....	401
MKShort.....	402
CVD.....	403
CVI.....	405
CVL.....	407
CVLongInt.....	409
CVS.....	410
CVShort.....	412
Left.....	413
Right.....	414
Mid (Function).....	415
LCase.....	417
UCase.....	418
LTrim.....	419
RTrim.....	420
Trim.....	422
InStr.....	423
InStrRev.....	425
Mid (Statement).....	426
LSet.....	428
RSet.....	429
Συναρτήσεις υποστήριξης νήματος - Threading Support	
Functions.....	431
ThreadCall.....	431
ThreadCreate.....	434
ThreadWait.....	439
ThreadDetach.....	440
ThreadSelf.....	442
MutexCreate.....	444
MutexLock.....	445
MutexUnlock.....	446
MutexDestroy.....	446
CondCreate.....	447
CondWait.....	450
CondSignal.....	452
CondBroadcast.....	455
CondDestroy.....	456
Συναρτήσεις εισόδου χρήστη - User Input Functions....	457



Input.....	457
Line Input.....	459
Input().....	460
Winput().....	462
Inkey.....	464
GetKey.....	467
Κεφάλαιο 12ο - Διάλεκτοι FreeBASIC.....	469
Κεφάλαιο 13ο - Προεπεξεργαστής (Preprocessor).....	470
Εντολές προεπεξεργαστή.....	470
Υποθετική Μεταγλώττιση (Conditional Compilation)....	471
#if...#elseif...#else...#endif.....	471
defined.....	472
#ifdef.....	473
#ifndef.....	474
Αντικατάσταση κειμένου (Text Replacement).....	475
#define.....	475
#Macro...#Endmacro.....	478
#undef.....	480
Τελεστής # (Preprocessor Stringize).....	481
Τελεστής ## (Preprocessor Concatenate).....	482
Τελεστής ! (Escaped String Literal).....	483
Τελεστής \$ (Non-Escaped String Literal).....	484
Οδηγίες αρχείων (File Directives).....	486
#include.....	486
#include.....	488
#include.....	488
Οδηγίες ελέγχου (Control Directives).....	489
#pragma.....	489
#lang.....	491
#print.....	492
#error.....	493
#assert.....	493
#line.....	494
Μεταεντολές (Metacommands).....	495
\$Dynamic.....	495
\$Static.....	496
\$Lang.....	497
Κεφάλαιο 14ο - Εγγενείς Ορισμοί (Intrinsic Defines).....	498
Κεφάλαιο 15ο Εξωτερικές βιβλιοθήκες (External Libraries)	
.....	501

Κεφάλαιο 16ο Εφαρμογές της FreeBASIC.....	506
Εφαρμογή 1: Μετατροπή θερμοκρασίας σε Fahrenheit/Celsius.....	506
Εφαρμογή 2 - Αλγόριθμος Ταξινόμησης Φυσαλίδας (Bubble Sort).....	509
Εφαρμογή 3 - Αλγόριθμος Ταξινόμησης με Εισαγωγή (Insertion Sort).....	511
Εφαρμογή 4 - Αλγόριθμος Ταξινόμησης με Επιλογή (Selection sort).....	513
Εφαρμογή 5 - Αλγόριθμος Γρήγορης Ταξινόμησης (Quick Sort).....	516
Εφαρμογή 6 - Αλγόριθμος Ταξινόμησης με Συγχώνευση (Merge Sort).....	519
Εφαρμογή 7 - Μάντεψε τον αριθμό.....	521
Εφαρμογή 8 - Διαχείριση βάσης δεδομένων sqlite3.....	523
Βιβλιογραφία.....	527
Παράρτημα.....	527
Creative Commons Αναφορά 4.0.....	527

# Εισαγωγή

Καλώς ήρθατε στο κόσμο της FreeBASIC !

Η FreeBASIC είναι ένας ελεύθερος μεταγλωττιστής (compiler) για τα Windows 32bit και 64bit, για το 32bit protected mode DOS και για το Linux x86, x86\_64 και ARM.

Είναι έργο ανοιχτού κώδικα (open source) και [αδειοδοτείται με την άδεια GPL](#).

Έχει σχεδιαστεί να είναι συμβατή με την [QuickBASIC](#) ωστόσο έχει πολλές επεκτάσεις ειδικά στην γλώσσα C.

Με την FreeBASIC μπορείτε να αναπτύξετε προγράμματα για MS-Windows, DOS, Linux.

Για τις απαιτήσεις του κάθε συστήματος για να τρέξουν τον μεταγλωττιστή δείτε στην σελίδα [-απαιτήσεις-](#).

## Εγκατάσταση

Για αρχεία της FreeBASIC δείτε στο <https://www.freebasic.net/>

### Για Windows

Ανάλογα ποια Windows έχετε, 32bit/64bit μεταφορτώστε το αρχείο FreeBASIC-x.xx.x-win32.zip ή FreeBASIC-x.xx.x-win64.zip.

Όπου x.xx.x η έκδοση της FreeBASIC.

Η τελευταία έκδοση κατά την συγγραφή αυτού του βιβλίου είναι η FreeBASIC-1.08.1.

Αποσυμπέστε το zip αρχείο στην θέση που επιθυμείτε.

Για να τρέξετε τον μεταγλωττιστή, τρέξτε από γραμμή εντολών το αρχείο fbc.exe

Αν θέλετε μπορείτε να κατεβάσετε και τις δύο εκδόσεις της FreeBASIC στο ίδιο σύστημα, αν έχετε Windows 64bit.

Σε αυτή την περίπτωση θα μπορείτε να τρέχετε όποια έκδοση θέλετε και να χτίζετε 32bit ή 64bit εφαρμογές.

## Για Linux

Μεταφορτώστε την τελευταία έκδοση της FreeBASIC-x.xx.x-linux-x86.tar.gz (32bit) ή FreeBASIC-x.xx.x-linux-x86\_64.tar.gz (64bit) για το σύστημα σας.

Αποσυμπιέστε το αρχείο με “δεξί κλικ”, “Αποσυμπίηση εδώ” ή γράφοντας στο τερματικό τις παρακάτω εντολές.

```
$ cd Downloads
$ tar xzf FreeBASIC-x.xx.x-linux-x86.tar.gz
ή
$ tar xzf FreeBASIC-x.xx.x-linux-x86_64.tar.gz
```

Στην συνέχεια μπείτε στον φάκελο που αποσυμπιέσατε και τρέξτε το script εγκατάστασης.

```
$ cd FreeBASIC-x.xx.x-linux-x86
$ sudo ./install.sh -i
```

Η FreeBASIC απαιτεί αρκετά πακέτα στο Linux πριν χρησιμοποιήσετε τον μεταγλωττιστή.

Αυτά τα πακέτα είναι:

- binutils
- libc development files
- gcc
- libncurses development files
- libtinfo development files
- X11 development files (για FB γραφικά)
- libffi development files (για την χρήση του ThreadCall)
- gpm (general purpose mouse) daemon και libgpm

Για απεγκατάσταση της FreeBASIC από το /usr/local, μπορείτε να τρέξετε το install.sh script ξανά αλλά με τον διακόπτη -u:

```
sudo ./install.sh -u
```

## Για το DOS

Μεταφορτώστε την τελευταία έκδοση  
FreeBASIC-x.xx.x-dos.zip archive

Αποσυμπίστε σε μία θέση με τουλάχιστον 13 MiB ελεύθερου χώρου.

Ο αρχικός φάκελος αποσυμπίεσης ονομάζεται FreeBASIC-x.xx.x-dos (θα συγκοφτεί σε "FREEBASI" στο DOS), έτσι ίσως θα θέλατε να τον μετονομάσετε σε κάτι πιο συνηθισμένο σε μια λέξη με 8 χαρακτήρες όπως "FB".

Για περισσότερες πληροφορίες σχετικά με την εγκατάσταση της FreeBASIC μπορείτε να δείτε στην σελίδα στο [wiki](#).

## Εγκατάσταση IDE

Η FreeBASIC έρχεται χωρίς κάποιο περιβάλλον ανάπτυξης IDE.

Ωστόσο αν θέλετε μπορείτε να εγκαταστήσετε ένα για Windows/Linux.

Παρακάτω μπορείτε να δείτε μερικά από τα πιο γνωστά IDE που υπάρχουν για FreeBASIC.

### **FBIde (2004)**

<https://fbide.freebasic.net/>

### **FbEdit (2009-04-09)**

<https://github.com/CherryDT/FbEditMOD>

### **wxFBE (Aug 21, 2012)**

<https://freebasic.net/forum/viewtopic.php?f=8&t=20284>

### **poseidonFB (Sep 13, 2015)**

<https://freebasic.net/forum/viewtopic.php?f=8&t=23935>

### **WinFBE Editor (Nov 27, 2016)**

<https://freebasic.net/forum/viewtopic.php?f=8&t=25215>

## IUP\_FB\_EDITOR (Oct 17, 2017)

<https://freebasic.net/forum/viewtopic.php?f=8&t=26030>

## VisualFBEditor (Dec 27, 2018)

<https://github.com/XusinboyBekchanov/VisualFBEditor>

## IUP\_FB\_Editor+ (Mar 07, 2020)

<https://freebasic.net/forum/viewtopic.php?f=8&t=28375>

Αναλυτικά μπορείτε να δείτε στο forum της FreeBASIC.

<https://www.freebasic.net/forum/viewtopic.php?t=28347>

Σε αυτό το βιβλίο θα χρησιμοποιήσουμε τον VisualFBEditor σε περιβάλλον Linux Ubuntu 20.04 LTS.

## Εγκατάσταση του poseidonFB

Μεταφορτώστε το poseidonFB από το bitbucket.org ως AppImage αρχείο από τον σύνδεσμο

<https://bitbucket.org/KuanHsu/poseidonfb/downloads/>

Σιγουρευτείτε ότι έχετε κατεβάσει την έκδοση

[https://bitbucket.org/KuanHsu/poseidonfb/downloads/poseidonFB\\_x64\\_rev462.AppImage](https://bitbucket.org/KuanHsu/poseidonfb/downloads/poseidonFB_x64_rev462.AppImage)

Ανοίξτε τερματικό στην τοποθεσία που βρίσκεται το poseidonFB\_x64\_rev462.AppImage και δώστε του δικαιώματα εκτελέσιμου αρχείου με την παρακάτω εντολή.



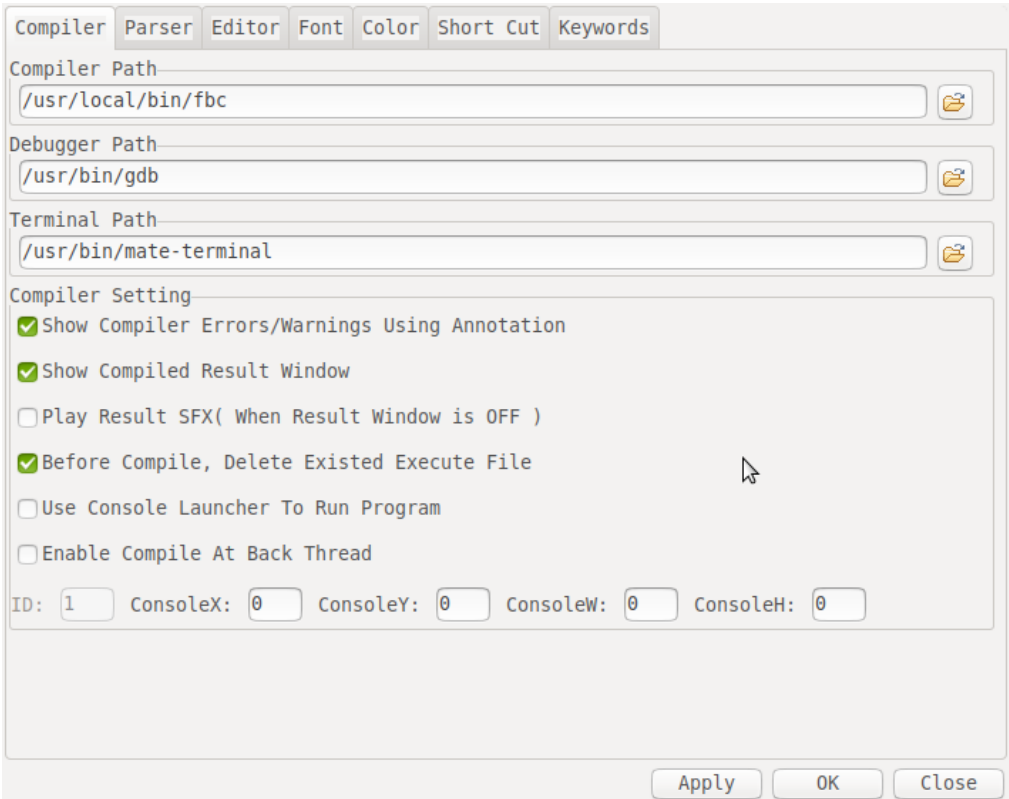
```
chmod +x poseidonFB_x64_rev462.AppImage
```

Τώρα μπορείτε να κάνετε διπλό κλικ στο αρχείο poseidonFB\_x64\_rev462.AppImage για να τρέξετε το IDE.

## Ρυθμίζοντας το poseidonFB

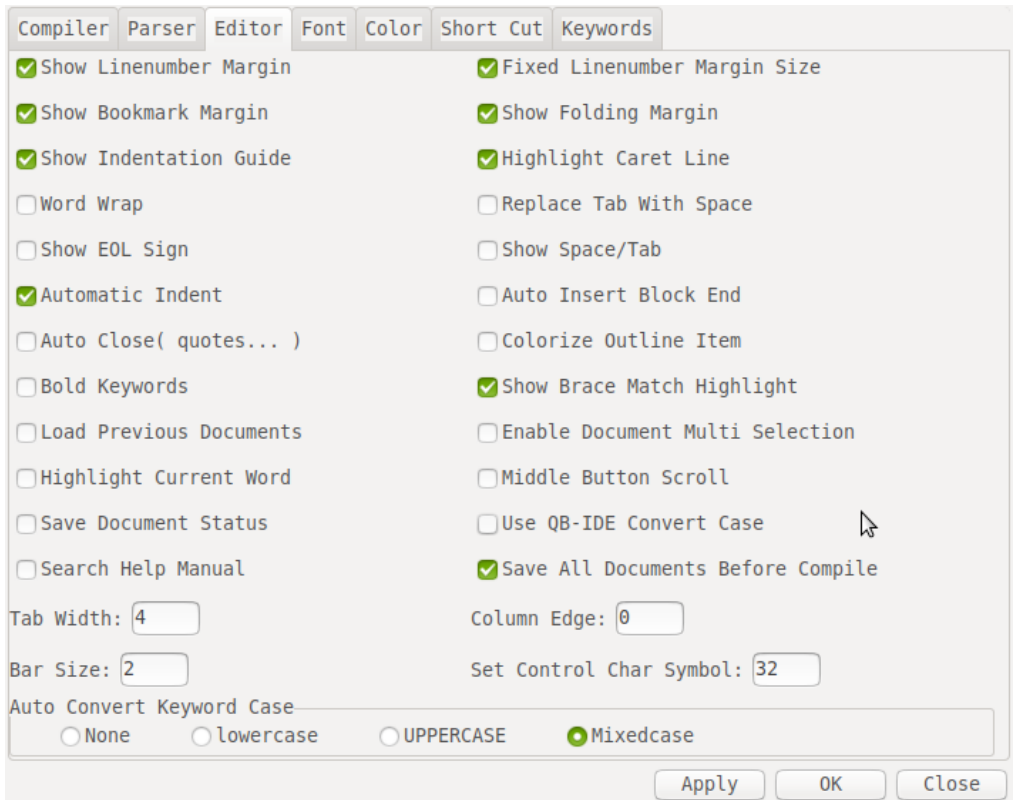
Μετά την εγκατάσταση του poseidonFB πρέπει να γίνουν κάποιες απαραίτητες ρυθμίσεις.

Από το μενού Options->Preference->Compiler επιλέξτε τις ρυθμίσεις όπως φαίνεται στην παρακάτω εικόνα.



Στην επιλογή Terminal Path ορίστε το τερματικό του συστήματός σας. Εγώ στο σύστημα μου έχω το mate-terminal, στο δικό σας μπορεί να είναι κάποιο άλλο.

Πηγαίνετε στην καρτέλα Editor και επιλέξτε τις ρυθμίσεις που φαίνονται στην παρακάτω εικόνα.



Μετά από αυτές τις ρυθμίσεις μπορείτε να χρησιμοποιήσετε το IDE poseidonFB για το Linux Ubuntu Mate 20.04 LTS.

## Συμβάσεις

Μέσα στο βιβλίο αυτό ορισμένα σημεία έχουν ιδιαίτερη απεικόνιση για να τονιστούν κάποια χαρακτηριστικά όπως ότι τώρα αυτό που φαίνεται είναι κώδικας, ή εντολές στο τερματικό κτλ.



## Κώδικας



Όταν θα παρουσιάζεται αυτό το εικονίδιο τότε σημαίνει ότι αυτό που ακολουθεί είναι κώδικας που γράφεται στον επεξεργαστή κειμένου του περιβάλλοντος ανάπτυξης εφαρμογών.

## Παράδειγμα:



1. `Print "Hello World"`

## Τερματικό



Όταν θα παρουσιάζεται αυτό το εικονίδιο τότε σημαίνει ότι αυτό που ακολουθεί είναι κείμενο που γράφεται μέσα σε τερματικό.

## Παράδειγμα:



```
user@laptop:~/Bin/FreeBASIC-1.08.1$ ls
```

```
bin doc FB-manual-1.08.1.chm include lib  
changelog.txt examples install.sh readme.txt
```

## Προσοχή



Όταν θα παρουσιάζεται αυτό το εικονίδιο τότε σημαίνει ότι αυτό που ακολουθεί είναι κείμενο που χρειάζεται ιδιαίτερη προσοχή ή ότι είναι ορισμός ή σύνταξη οδηγιών της γλώσσας FreeBASIC

# Μια σύντομη ανασκόπηση στις γλώσσες προγραμματισμού.

Οι γλώσσες προγραμματισμού έχουν μια ραγδαία εξέλιξη από τον καιρό που πρωτοεμφανίστηκαν.

Στις αρχές τους, οι προγραμματιστές δούλευαν με τον κώδικα μηχανής. Αυτές οι οδηγίες προς τον υπολογιστή αποτελούνταν από μεγάλες συμβολοσειρές από 0 και 1.

Αργότερα ανακάλυψαν τους συμβολομεταφραστές (assemblers) για να αντιστοιχούν τις οδηγίες προς τον υπολογιστή σε μικρές αναγνώσιμες και εύκολα κατανοητές λέξεις από τον άνθρωπο. Τέτοιες λέξεις είναι για παράδειγμα οι ADD, MOV της γλώσσας Assembly.

Με τον καιρό η τεχνολογία και επιστήμη των συμβολομεταφραστών προχώρησε και αναπτύχθηκαν γλώσσες ικανές να δέχονται λέξεις και προτάσεις όπως Let X=10. Αυτές οι γλώσσες ονομάστηκαν, γλώσσες υψηλού επιπέδου και ήταν γλώσσες όπως η BASIC, FORTRAN, COBOL, ALGOL, PASCAL, LISP, PROLOG.

Οι γλώσσες αυτές χρησιμοποιούν έναν διερμηνευτή (interpreter) και έναν μεταγλωττιστή (compiler).

Ο διερμηνευτής (interpreter) μεταφράζει ένα πρόγραμμα όπως το διαβάζει και μετατρέπει τις οδηγίες του προγράμματος άμεσα σε γλώσσα μηχανής.

Ο μεταγλωττιστής (compiler) μεταφράζει των κώδικα σε μια ενδιάμεση μορφή και δημιουργεί ένα αντικειμενικό αρχείο και στην συνέχεια χρειάζεται ένα πρόγραμμα σύνδεσης (linker) για να μετατρέψει το αντικειμενικό αρχείο σε εκτελέσιμο πρόγραμμα.

Η διαφορά των γλωσσών που χρησιμοποιούν διερμηνευτές ή μεταγλωττιστές έγκειται στο συμβάν ότι σε μια γλώσσα με διερμηνευτή είναι πιο εύκολο στον προγραμματιστή να δουλεύει με αυτούς αλλά ο τελικός χρήστης θα πρέπει να έχει τον διερμηνευτή για να τρέξει το εκτελέσιμο αρχείο.

Ενώ στις γλώσσες με μεταγλωττιστή το εκτελέσιμο αρχείο μπορεί να τρέξει χωρίς την ανάγκη της ύπαρξης της γλώσσας προγραμματισμού στον τελικό προγραμματιστή.

Η FreeBASIC μας παρέχει έναν μεταγλωττιστή τον `fbcc`.

Ο μεταγλωττιστής `fbcc` παίρνει ένα αρχείο προέλευσης ως όρισμα γραμμής εντολών και παράγει ένα εκτελέσιμο αρχείο.

Αυτό το κάνει μεταγλωττίζοντας το αρχείο προέλευσης (`.bas`) σε ένα αρχείο `assembly` (`.asm`), στη συνέχεια το μεταγλωττίζει σε ένα αρχείο αντικειμένου (`.o`) χρησιμοποιώντας το GAS (GNU Assembler) και τελικά συνδέει χρησιμοποιώντας το LD αυτό το αρχείο αντικειμένου με άλλα αρχεία αντικειμένων και βιβλιοθήκες που χρειάζεται για εκτέλεση, παράγοντας το τελικό εκτελέσιμο αρχείο.

Τα αρχεία `assembly` και μεταγλωττισμένων αντικειμένων διαγράφονται σε αυτό το σημείο από προεπιλογή.

# Κεφάλαιο 1ο - Το πρώτο μου πρόγραμμα

Τρέξτε το εκτελέσιμο του IDE, το `roseidonFB_x64_rev462.AppImage` για να ανοίξετε το περιβάλλον ανάπτυξης.

Αν και με το IDE μπορείτε να δημιουργείτε Projects που θα περιέχουν πολλά αρχεία `bas`, για αρχή για εκμάθηση της γλώσσας FreeBASIC δεν θα χρειαστεί αυτό.

Αρχικά θα δημιουργούμε μικρά μεμονωμένα αρχεία `bas` τα οποία θα τα τροφοδοτούμε στον μεταγλωττιστή `fbc` και θα παράγουμε το εκτελέσιμο αρχείο που θα τρέχουμε στο τερματικό.

Επιλέγουμε **File**→ **New** ή πατάμε το εικονίδιο **New**. Αμέσως δημιουργείτε ένα αρχείο `bas` με όνομα `NONAME#0.bas`.

Γράφουμε στον editor τα εξής:



1. **Print "Hello World!"**
2. **Sleep**
3. **End**

Επιλέγουμε **File**→ **Save** ή πατάμε **CTRL+S**. Αποθηκεύουμε το αρχείο με όνομα `hello.bas` στον φάκελο που επιθυμούμε.

Επιλέγουμε **Build**→ **Quick Run**

Το IDE μας ενημερώνει με ένα μήνυμα `Compile Success`.

Το εκτελέσιμο `hello.bas`

```
File Edit View Search Terminal Help
Hello World!
END
```

Εικόνα 1: Το πρόγραμμα hello στο mate-terminal

## Ανάλυση



1. Print "Hello World!"
2. Sleep
3. End

Η πρώτη γραμμή καλεί την διαδικασία Print η οποία τυπώνει στην οθόνη ότι έπεται μέσα σε διπλά εισαγωγικά.

Εντός των εισαγωγικών είναι η συμβολοσειρά Hello World! Αυτή η συμβολοσειρά ή αλλιώς string τυπώνεται στην οθόνη.

Η δεύτερη γραμμή καλεί την διαδικασία Sleep η οποία θέτει το πρόγραμμα σε αναμονή μέχρι ο χρήστης να πατήσει Enter. Έτσι το τερματικό δεν κλείνει αμέσως και περιμένει τον χρήστη να πατήσει Enter.

Η τρίτη γραμμή δηλώνει το τέλος του προγράμματος με την δήλωση End. Εδώ το πρόγραμμα τερματίζει και το τερματικό κλείνει.

## **Κεφάλαιο 2ο - Τα βασικά μέρη της γλώσσας**

### **Αλφάβητο και Λεξιλόγιο της γλώσσας προγραμματισμού.**

Κάθε γλώσσα έχει τους δικούς της χαρακτήρες που με συνδυασμό αυτών παράγονται οι διάφορες λέξεις. Το σύνολο των λέξεων μιας γλώσσας είναι το λεξιλόγιο της.

Το ίδιο συμβαίνει και σε μια γλώσσα προγραμματισμού. Ο κατασκευαστής της γλώσσας έχει επιλέξει ένα σετ χαρακτήρων με τους οποίους θα δομήσει τις λέξεις της γλώσσας.

Κάθε λέξη της γλώσσας είναι ορισμένη να τελεί μια λειτουργία στην γλώσσα προγραμματισμού.

Οι λέξεις αυτές είναι δεσμευμένες από τον κατασκευαστή της γλώσσας και δεν επιτρέπεται να τις χρησιμοποιήσει ο προγραμματιστής ως δικές του, για παράδειγμα να ονομάσει μια μεταβλητή με το ίδιο όνομα μιας δεσμευμένης λέξης.

Για παράδειγμα στην FreeBASIC το αλφάβητο αποτελείται από όλους τους αλφαριθμητικούς χαρακτήρες της Αγγλικής δλδ τα γράμματα από a-z και τα νούμερα από 0-9 και κάποιους ειδικούς χαρακτήρες όπως το ' ή () κτλ.

Από αυτούς τους χαρακτήρες παράγονται δεσμευμένες λέξεις όπως το IF, SUB, END, PUBLIC κτλ

Αν λοιπόν ο προγραμματιστής επιχειρήσει να ορίσει ως όνομα μεταβλητής μία δεσμευμένη λέξη ο μεταγλωττιστής θα του εμφανίσει ένα λάθος σύνταξης και το πρόγραμμα δεν θα ξεκινήσει.

# Σύνταξη της γλώσσας

Κάθε γλώσσα έχει την δικιά της σύνταξη. Αυτό σημαίνει ότι για να λειτουργήσει ένα μέρος κώδικα πρέπει να είναι γραμμένο σωστά.

Με άλλα λόγια αν θέλετε να γράψετε μια δημόσια ρουτίνα ο σωστός τρόπος είναι:



1. | PUBLIC SUB Button1\_Click()  
Και όχι  
1. | SUB PUBLIC Button1\_Click()

# Σχόλια μέσα σε ένα πρόγραμμα

Καθώς θα γράφετε προγράμματα θα δείτε ότι ένα πρόγραμμα μπορεί να έχει εκατοντάδες γραμμές.

Έτσι σε μερικές γραμμές θα θέλετε να γράψετε κάποια σχόλια.

Τα σχόλια είναι γραμμένα μέσα στο κώδικα του προγράμματος αλλά ο διερμηνευτής δεν τα υπολογίζει σαν εντολές της γλώσσας.

Τα σχόλια είναι απλά ένα κείμενο ανενεργό μέσα στον κώδικα του προγράμματος.

Όποια γραμμή του κώδικα ξεκινάει με μια απόστροφο ' ή την λέξη REM είναι ένα σχόλιο μιας γραμμής. Αν θέλουμε να κάνουμε σχόλια πολλαπλών γραμμών τότε γράφουμε τα σχόλια ανάμεσα σε /' και '/

## Παράδειγμα - comments.bas



```
1.  /' this is a multi line
2.  comment as a header of
3.  this example '/
4.
5.  Rem This Is a Single Line comment
6.  ' this is a single line comment
7.  Print "Hello" : Rem This is a comment
8.  Print "World" ' comment following a statement
9.  Print "FreeBASIC" : ' also acceptable
10. Sleep
11. End
```

## Έξοδος



```
Hello
World
FreeBASIC
```

Μάθετε να σχολιάζετε τον κώδικα σας είναι χρήσιμο για την κατανόηση του προγράμματος.



## Κεφάλαιο 3ο - Μεταβλητές

Τα προγράμματα χρειάζονται κάποιο τρόπο για να διαχειρίζονται δεδομένα.

Ένας τρόπος είναι να αποθηκεύουν δεδομένα στην μνήμη του η/υ και αργότερα να ανακτούν τα δεδομένα αυτά και να τα επεξεργάζονται.

Για να αποθηκεύσουν τα προγράμματα δεδομένα στην μνήμη χρησιμοποιούν μεταβλητές.



Μια μεταβλητή είναι μια θέση στην μνήμη του υπολογιστή.  
Σε μια μεταβλητή αποθηκεύουμε μία τιμή ενός τύπου δεδομένων.

Η μεταβλητή έχει ένα όνομα με το οποίο την καλούμε.  
Όταν ορίζουμε μια μεταβλητή ορίζουμε και τον τύπο των δεδομένων που αυτή θα δεχθεί.

Τα δεδομένα σε ένα πρόγραμμα μπορεί να είναι αριθμοί, κείμενο, ημερομηνία, αναφορά σε αντικείμενο, τιμή διεύθυνσης μνήμης και μια λογική έκφραση.  
Κάθε τύπος δεδομένων έχει ένα συγκεκριμένο μέγεθος στην μνήμη του υπολογιστή.

Μερικοί τύποι δεδομένων διαφέρουν στο μέγεθος της μνήμης όταν ο επεξεργαστής είναι τεχνολογίας 32bits ή 64bits.

## Τύποι δεδομένων (Datatypes)

Παρακάτω φαίνεται ο πίνακας των εγγενών τύπων δεδομένων της FreeBASIC.

<b>Τύπος (Datatype)</b>	<b>Περιγραφή τιμών</b>	<b>Μέγεθος στην μνήμη</b>
BOOLEAN	Αλήθεια (True) ή Ψέμα (False).	1 bit
BYTE	-128 ... +127	8 bits
UBYTE	0...255	8 bits
SHORT	-32.768 ... +32.767	16 bits
USHORT	0 ... 65535	16 bits
LONG	-2.147.483.648 ... +2.147.483.647	32 bits
ULONG	0 ... +4294967295	32 bits
INTEGER	32bit -2147483648 ... +2147483647  64bit -9223372036854775808 ... +9223372036854775807	32/64 bits
UINTEGER	0 ...  32bit +4294967295  64bit +18446744073709551615	32/64 bits
LONGINT	-9223372036854775808 ... +9223372036854775807	64 bits

ULONGINT	0 ... +18446744073709551615	64 bits
SINGLE	+/-1.401 298 E-45 ... +/-3.402 823 E+38	32 bits
DOUBLE	+/-4.940 656 458 412 465 E-324 ... +/-1.797 693 134 862 316 E+308	64 bits
enums	32bit: -2147483648, ... +2147483647  64bit: -9223372036854775808 ... +9223372036854775807	32/64 bits
String	Minimum characters 0 ...  Maximum characters 32bit: +2147483647,  64bit: +9223372036854775807	1 byte
Zstring	Minimum characters 0 ...  Maximum characters 32bit:	1 bbyte

	+2147483647,  64bit: +9223372036854775807	
Wstring	Minimum characters 0 ...  Maximum characters 32bit: +2147483647,  64bit: +9223372036854775807	*

\*) Οι ευρείς χαρακτήρες (wide characters) ή οι χαρακτήρες Unicode διαφέρουν σε μέγεθος από σύστημα σε σύστημα.

### **Τύποι δεδομένων πινάκων**

Platform	Maximum Subscript Range	Maximum Elements per Dimension	Minimum/ Maximum Dimensions	Maximum Size (in bytes)
32bit	-2147483648, +2147483647	+2147483647	1/8	+2147483647
64bit	- 92233720368547 75808,  +9223372036854 775807	+9223372036 854775807	1/8	+9223372036 854775807

# Δεκαδικοί, δεκαεξαδικοί, οκταδικοί, δυαδικοί ακέραιοι

Οι ακέραιοι Integers μπορούν να γράφονται σε δεκαδική (decimal), δεκαεξαδική (hexadecimal), οκταδική (octal) ή δυαδική (binary) μορφή.

## Δεκαδική μορφή:

Η δεκαδική μορφή είναι γνωστή σε όλους.

Ένας ακέραιος αριθμός σε δεκαδική μορφή αποτελείται από τα ψηφία 0 έως 9. Αν είναι προσημασμένος έχει και το πρόσημο - ή + .

## Παράδειγμα



- |    |  |           |
|----|--|-----------|
| 1. |  | X = 1     |
| 2. |  | X = -1    |
| 3. |  | X = 124   |
| 4. |  | X = -1245 |

## Δεκαεξαδική μορφή:

Η δεκαεξαδική μορφή ενός αριθμού αποτελείται από τα ψηφία 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Ένας δεκαεξαδικός αριθμός αρχίζει με &H ή &h

## Παράδειγμα



- |    |  |            |
|----|--|------------|
| 1. |  | X = &H001A |
| 2. |  | X = &h001A |
| 3. |  | X = &HFF0A |

## Οκταδική μορφή:

Η οκταδική μορφή ενός αριθμού αποτελείται από τα ψηφία 0, 1, 2, 3, 4, 5, 6, 7

Ένας δεκαεξαδικός αριθμός αρχίζει με &O ( Ο όπως στην λέξη "Octal" )

## Παράδειγμα



```
1. | X = &O361100  
2. | X = &O200
```

## Η δυαδική μορφή

Η δυαδική μορφή ενός αριθμού αποτελείται από τα ψηφία 0, 1.

Ένας δυαδικός αριθμός αρχίζει με &B

## Παράδειγμα



```
1. | X = &B0101  
2. | X = &B1101
```

# Δηλώνοντας Μεταβλητές

Σύνταξη δήλωσης μεταβλητών



```
Dim [Shared] name1 As DataType [, name2 As DataType, ...]
```

ή

```
Dim [Shared] As DataType name1 [, name2, ...]
```

Οι μεταβλητές είναι θέσεις μνήμης στις οποίες έχουμε πρόσβαση μέσω ενός προσδιοριστή ο οποίος είναι το όνομα της μεταβλητής.

## Παράδειγμα



1. `'' One variable per DIM statement`
2. `Dim text As String`
3. `Dim x As Double`

Για να ορίσουμε μια μεταβλητή τοποθετούμε την λέξη Dim και στην συνέχεια το όνομα της μεταβλητής ακολουθούμενη από την λέξη As και τον τύπο δεδομένων της μεταβλητής.

## Παράδειγμα



1. `'' More than one variable declared,`
2. `' different data types`
- 3.
4. `Dim k As Single, factor As Double, s As String`
- 5.
6. `'' More than one variable declared,`
7. `' all same data types`
- 8.
- 9.
10. `Dim As Integer mx, my, mz ,mb`

Όπως φαίνεται στο παραπάνω παράδειγμα με την χρήση της λέξεως Dim μπορούν να οριστούν παραπάνω από μια μεταβλητή διαφορετικών ή ίδιων τύπων δεδομένων.

# Αρχικοποίηση Μεταβλητών κατά την δήλωση τους

Μπορούμε κατά την δήλωση μεταβλητής να ορίσουμε και αρχική τιμή που θα έχει. Δηλαδή να την αρχικοποιήσουμε.

Σύνταξη αρχικοποίησης μεταβλητής



Dim scalar\_symbol As DataType = expression | Any

## Παράδειγμα



1. '' Variable having an initializer
2. `Dim x As Double = 1.23`

## Εμβέλεια μεταβλητών

Η εμβέλεια μιας μεταβλητής αναφέρεται στην ορατότητα της μέσα στο πρόγραμμα.

Μια μεταβλητή δεν είναι ορατή (δεν μπορεί να προσπελαστεί) εκτός της εμβέλειας της από την οποία δηλώθηκε.

Που και πώς μια μεταβλητή έχει δηλωθεί ορίζει την εμβέλεια της (scope).

Στη FreeBASIC, υπάρχουν 4 κατηγορίες εμβέλειας: **local**, **shared**, **common** και **common shared**.

Κάθε μία από αυτές τις εμβέλειες έχει διαφορετική ορατότητα και κανόνες οι οποίοι περιγράφονται στην συνέχεια.



## Local Scope

Μεταβλητές οι οποίες έχουν δηλωθεί σε local scope είναι ορατές μόνο τοπικά σε διαδικασίες όπως IF, SELECT, WITH, FOR, WHILE, DO, SCOPE, ή στο μπλοκ του αρθρώματος (module) στο οποίο έχουν δηλωθεί.

Οι ρουτίνες SUB, και συναρτήσεις FUNCTION ορίζουν για τον εαυτό τους τοπική εμβέλεια μεταβλητών.

Στο local scope δεν υπάρχει ορατότητα μεταβλητών μεταξύ των ρουτινών και των συναρτήσεων, ούτε μεταξύ των modules.

Ως module εννοούμε ένα αρχείο με επέκταση bas της FreeBASIC.

### Παράδειγμα - local.bas



```
1. ' visible only in this module
2. Dim As Integer local_moduleLevel1 = 10
3. ' OK.
4. Print local_moduleLevel1
5.
6. Scope
7. ' OK; SCOPE Blocks inherit outer scope
8. Print local_moduleLevel1
9.
10. ' visible only in this SCOPE Block
11. Dim As Integer local_moduleLevel2 = 20
12. ' OK.
13. Print local_moduleLevel2
14. End Scope
15.
16.
17. ' Error; can't see inner-SCOPE vars
18. ' Print local_moduleLevel2
19.
20. Function some_function( ) As Integer
21. ' visible only in this function
22. Dim As Integer local_functionLevel=30
23. ' OK.
24. Print local_functionLevel
25. ' Error; can't see local module-level vars
26. ' Print local_moduleLevel1
27. ' Error; can't see local module-level vars
28. ' Print local_moduleLevel2
29. Function = 0
30. End Function
```

```

31.
32. '' Print local_functionLevel
33. '' Error; can't see function_level vars
34.
35. Sleep
36. End 0

```

## Shared Scope

Οι μεταβλητές που δηλώνονται ως Shared έχουν εμβέλεια σε όλο το module στο οποίο υπάρχουν όσο και μεταξύ των διαδικασιών του module.

### Παράδειγμα - shared.bas



```

1. '' visible throughout this module
2. Dim Shared As Integer shared_moduleLevel1 = 10
3. '' OK.
4. Print shared_moduleLevel1
5.
6. Scope
7. '' OK; can see outer-scope vars
8.   Print shared_moduleLevel1
9.
10. '' Error; SCOPE-level vars cannot be shared
11. '' dim shared as integer shared_ModuleLevel2
12. End Scope
13.
14. Function some_function( ) As Integer
15. '' OK; can see shared module-level vars
16.   Print shared_moduleLevel1
17. '' Error; function-level vars cannot be shared
18. '' dim shared as integer sharedFunctionLevel
19.   Function = 0
20. End Function
21.
22.
23. Sleep
24. End

```

## Common Scope

Οι μεταβλητές που έχουν δηλωθεί ως Common έχουν εμβέλεια σε όλα τα modules.

### Παράδειγμα - module1.bas



```
1.  '' compile with:
2.  ''      fbc -lang qb module1.bas module2.bas
3.  '$lang: "qb"
4.  Declare Sub Print_Values()
5.  Common m1 As Integer
6.  Common m2 As Integer
7.
8.  ' This is executed after all other modules
9.  m1 = 1
10. Print "Module1"
11. Print "m1 = "; m1      ' m1 = 1 as set in this
12. module
13. Print "m2 = "; m2      ' m2 = 2 as set in module2
14. Print_Values
```

### module2.bas



```
1.  Common m1 As Integer
2.  Common m2 As Integer
3.  m2 = 2
4.  Print "Module2"      ' This is executed first
5.  Print "m1 = "; m1    ' m1 = 0 (by default)
6.  Print "m2 = "; m2    ' m2 = 2
7.  Sub Print_Values()
8.      Print "Module2.Print_Values"
9.      Print "m1 = "; m1 ' Implicit variable = 0,
10. because '-lang qb' use
11.      Print "m2 = "; m2 ' Implicit variable = 0,
12. because '-lang qb' use
13. End Sub
```

## Έξοδος



```
Module2
m1 = 0
m2 = 2
Module1
m1 = 1
m2 = 2
Module2.Print_Values
m1 = 0
m2 = 0
```

## Common Shared

Μεταβλητές που δηλώνονται ως Common Shared έχουν εμβέλεια σε όλα τα modules και σε όλες τις διαδικασίες αυτών των modules.

## Παράδειγμα - module3.bas



```
1.  ' compile with:
2.  '   fbc module3.bas module4.bas
3.  Declare Sub Print_Values()
4.  Common m1 As Integer
5.  Common m2 As Integer
6.
7.  ' This is executed after all other modules
8.  m1 = 1
9.  Print "Module3"
10. Print "m1 = "; m1 ' m1 = 1 as set in this module
11. Print "m2 = "; m2 ' m2 = 2 as set in module2
12. Print_Values
13.
```

## module4.bas



```
1. Common Shared m1 As Integer
2. Common Shared m2 As Integer
3. m2 = 2
4. Print "Module4"           '' This is executed first
5. Print "m1 = "; m1        '' m1 = 0 (by default)
6. Print "m2 = "; m2        '' m2 = 2
7. Sub Print_Values()
8.     Print "Module4.Print_Values"
9.     Print "m1 = "; m1     '' m1 = 1
10.    Print "m2 = "; m2     '' m2 = 2
11. End Sub
```

## Έξοδος



```
Module4
m1 = 0
m2 = 2
Module3
m1 = 1
m2 = 2
Module4.Print_Values
m1 = 1
m2 = 2
```

## Ανακύκλωση μεταβλητών

Στις αριθμητικές μεταβλητές θα προσέξατε ότι ανάλογα του τύπου δεδομένων τους μπορούν να πάρουν μία τιμή ανάμεσα σε ένα εύρος τιμών.

Για παράδειγμα ένας Short οι τιμές που μπορεί να πάρει είναι από -32.768 έως +32.767.

Αν λοιπόν έχουμε μια μεταβλητή τύπου δεδομένων Short με την τιμή +32.767 και την αυξήσουμε κατά μία μονάδα τι γίνεται;

Προφανώς δεν μπορεί να τεθεί η τιμή σε +32.768. Αυτό που γίνεται είναι η μεταβλητή να λειτουργεί σαν ένα κοντέρ μέτρησης. Η επόμενη τιμή της μεταβλητής θα είναι το -32.768, όπως ακριβώς όταν ένα κοντέρ μέτρησης από 9999 πάει στην αρχή του στο 0000.

Σε μια μεταβλητή Short η αρχή μέτρησης είναι το -32.768 δλδ η μικρότερη τιμή από το εύρος τιμών.

Έτσι το ίδιο ακριβώς συμβαίνει και σε μια μεταβλητή UByte. Το εύρος τιμών είναι από 0 έως 255.

Η κατώτατη τιμή είναι το 0 και αυτή θα είναι η αρχή μέτρησης της μεταβλητής.

Αν σε μια μεταβλητή UByte αυξήσετε την μέγιστη τιμή της κατά ένα, δλδ από 255 σε 255+1 τότε αυτή θα έχει όχι 256 αλλά το 0. Δλδ θα έχει γυρίσει στην αρχή.

Αυτό το φαινόμενο ονομάζεται **“Ανακύκλωση μεταβλητής ή υπερχείλιση μεταβλητής (overflow)”**.



Υπερχείλιση ή ανακύκλωση μεταβλητών ονομάζεται το φαινόμενο όταν η τιμή μιας μεταβλητής υπερβαίνει το μέγεθος της θέσης αποθήκευσης που έχει προβλεφθεί για την καταχώρηση της.

## Παράδειγμα - overflow.bas



```
1. Dim i As UByte
2. i = 255
3. Print i
4. i = i + 1
5. Print i
6. Sleep
7. End
```

## Έξοδος



```
255
0
```

## Ανάλυση

Στην γραμμή 1 ορίζουμε την μεταβλητή *i* ως μη προσημασμένο byte.

Στην γραμμή 2 θέτουμε σε αυτήν την τιμή 255.

Στην γραμμή 3 τυπώνουμε στο τερματικό την μεταβλητή *i*.

Στην γραμμή 4 αυξάνουμε κατά μία μονάδα την μεταβλητή *i*.

Στο σημείο αυτό συμβαίνει η υπερχείλιση. Η μεταβλητή *i* από την τιμή 255 επιστρέφει στην τιμή 0.

Στην γραμμή 5 τυπώνουμε στο τερματικό, άλλη μια φορά την μεταβλητή *i*. Αυτή την φορά τυπώνεται το 0.

Τέλος στις γραμμές 6,7 έχουμε την αναμονή του χρήστη και το τέλος του προγράμματος.

## Ειδικοί χαρακτήρες συμβολοσειρών (strings)

Όταν θέτουμε μια τιμή σε μια μεταβλητή ή σταθερά string συνήθως είναι λέξεις, αριθμοί ή λέξεις με αριθμούς οι οποίες σχηματίζονται από σετ αλφαριθμητικών χαρακτήρων [a-z0-9].

Υπάρχουν όμως και οι λεγόμενοι ειδικοί χαρακτήρες που μπορούμε να έχουμε σε μια συμβολοσειρά.

Αυτοί οι ειδικοί χαρακτήρες είναι για παράδειγμα ο χαρακτήρας νέας γραμμής, το διπλό διάστημα (tab) κτλ.

Αυτοί οι χαρακτήρες ονομάζονται χαρακτήρες διαφυγής (Escape characters) και είναι οι παρακάτω.

```
\a beep  
\b backspace  
\f formfeed  
\l or \n newline  
\r carriage return  
\t tab  
\unnnn unicode char in hex  
\v vertical tab  
\nnn ascii char in decimal  
\&hnn ascii char in hex  
\&onnn ascii char in octal  
\&bnnnnnnnnn ascii char in binary  
\\ backslash  
\" double quote  
' single quote
```

Σημείωση: Ο χαρακτήρας 0 (\000 = \&h00 = \&o000 = \&b00000000) είναι ο χαρακτήρας null.

Μόνο χαρακτήρες πριν τον πρώτο null terminator είναι ορατοί σε ένα String. Για να πάρουμε ένα μηδενικό χαρακτήρα σε μια συμβολοσειρά χρησιμοποιήστε το Chr(0).



Χαρακτήρες διαφυγής μπορούν να χρησιμοποιηθούν σε ένα string χρησιμοποιώντας τον χαρακτήρα !.

## Σύνταξη



```
result = !"text"
```

## Παράδειγμα - escape.bas



```
1. Print "Some escape sequence examples:"
2. Print !"1.\tsingle quote (\\') : \'
3. Print !"2.\tdouble quote (\\") : \"
4. Print !"3.\tbackslash (\\) : \\
5. Print !"4.\tascii char (\\65): \65"
6. Sleep
7. End
```

## Έξοδος



```
Some escape sequence examples:
1. single quote (\') : '
2. double quote (\") : "
3. backslash (\\) : \
4. ascii char (\65): A
```

## Μετατροπή τιμών μεταβλητών

Υπάρχουν περιπτώσεις που χρειάζεται μια τιμή μιας μεταβλητής να την εκχωρήσουμε σε μια άλλη μεταβλητή διαφορετικού τύπου δεδομένων.

Για παράδειγμα έχουμε μια φόρμα με ένα κουτί κειμένου TextBox.

Ο χρήστης χρησιμοποιεί το κουτί κειμένου για να εισάγει έναν αριθμό ο οποίος μετά θα χρησιμοποιηθεί σε μια μαθηματική πράξη.

Το TextBox όμως έχει την ιδιότητα Text και ό,τι εισάγεται σε αυτό αποτελεί ένα κείμενο που εκχωρείται στην ιδιότητα Text. Έτσι η FreeBASIC τον αριθμό που πληκτρολογεί ο χρήστης τον αναγνωρίζει ως κείμενο.

Θα πρέπει πριν χρησιμοποιηθεί αυτό το κείμενο να μετατραπεί σε αριθμό.

Για τέτοιες περιπτώσεις υπάρχουν οι συναρτήσεις μετατροπής τιμών μεταβλητών.

Αυτές είναι:

<b>Συναρτήσεις μετατροπής.</b>	
CBool	Μετατρέπει μια τιμή σε Boolean.
CByte	Μετατρέπει μια τιμή σε Byte.
CUByte	Μετατρέπει μια τιμή σε UByte.
CShort	Μετατρέπει μια τιμή σε Short
CUShort	Μετατρέπει μια τιμή σε UShort
CInt	Μετατρέπει μια τιμή σε Integer.
CUInt	Μετατρέπει μια τιμή σε UInteger.
CLng	Μετατρέπει μια τιμή σε Long.
CULng	Μετατρέπει μια τιμή σε ULong.
CLngInt	Μετατρέπει μια τιμή σε Long Integer.

CULngInt	Μετατρέπει μια τιμή σε ULong Integer.
CSng	Μετατρέπει μια τιμή σε Single.
CDbl	Μετατρέπει μια τιμή σε Double.
Str	Μετατρέπει μια τιμή σε String
Val	Μετατρέπει μια τιμή string σε αριθμό κινούμενης υποδιαστολής (double)

## CBool

### Σύνταξη



Declare Function CBool ( ByVal expression As datatype ) As Boolean

### Χρήση

result = CBool( expression )

### Παράδειγμα - cbool.bas



```

1.  'Create an BOOLEAN variable
2.  Dim b As Boolean
3.  'Convert a numeric value
4.  b = CBool(1)
5.  'Print the result, should return True
6.  Print b
7.  Sleep
8.  End

```

### Έξοδος



true

# CByte

## Σύνταξη



Declare Function CByte ( ByVal expression As datatype ) As Byte

## Χρήση

result = CByte( expression )

## Παράδειγμα - cbyte.bas



```
1. | Create an BYTE variable
2. | Dim numeric_value As Byte
3. | Convert a numeric value
4. | numeric_value = CByte(-66.30)
5. | Print the result, should return -66
6. | Print numeric_value
7. | Sleep
8. | End
```

## Έξοδος



-66

# CUByte

## Σύνταξη



Declare Function CUByte ( ByVal expression As datatype ) As UByte

## Χρήση

result = CUByte( expression )

## Παράδειγμα - cubyte.bas



```
1. | 'Create an UNSIGNED BYTE variable
2. | Dim numeric_value As UByte
3. | 'Convert a numeric value
4. | numeric_value = CUByte(123.55)
5. | 'Print the result, should return 124
6. | Print numeric_value
7. | Sleep
8. | End
```

## Έξοδος



124

# CShort

## Σύνταξη



Declare Function CShort ( ByVal expression As datatype ) As Short

## Χρήση

result = CShort( expression )

## Παράδειγμα - cshort.bas



```
1. 'Create an SHORT variable
2. Dim numeric_value As Short
3. 'Convert a numeric value
4. numeric_value = CShort(-4500.66)
5. 'Print the result, should return -4501
6. Print numeric_value
7. Sleep
8. End
```

## Έξοδος



```
-4501
```

## CUShort

### Σύνταξη



Declare Function CUShort ( ByVal expression As datatype ) As UShort

### Χρήση

result = CUShort( expression )

## Παράδειγμα - cushort.bas



```
1. 'Create an USHORT variable
2. Dim numeric_value As UShort
3. 'Convert a numeric value
4. numeric_value = CUShort(36000.4)
5. 'Print the result, should return 36000
6. Print numeric_value
7. Sleep
8. End
```

## Έξοδος



```
36000
```

## CInt

### Σύνταξη



Declare Function CInt ( ByVal expression As datatype ) As Integer

### Χρήση

result = CInt( expression )

### Παράδειγμα - cint.bas



```
1. 'Create an INTEGER variable
2. Dim numeric_value As Integer
3.
4. 'Convert a numeric value
5. numeric_value = CInt(300.5)
6. 'Print the result, should return 300,
7. 'because 300 is even
8. Print numeric_value
9.
10. numeric_value = CInt(301.5)
11. 'Print the result, should return 302,
12. 'because 301 is odd
13. Print numeric_value
14. Sleep
15. End
```

## Έξοδος



```
300
302
```

## CUInt

### Σύνταξη



Declare Function CUInt ( ByVal expression As datatype ) As UInteger

### Χρήση

result = CUInt( expression )

### Παράδειγμα - `cuint.bas`



```
1. | 'Create an UNSIGNED INTEGER variable
2. | Dim numeric_value As UInteger
3. | 'Convert a numeric value
4. | numeric_value = CUInt(300.23)
5. | 'Print the result = 300
6. | Print numeric_value
7. | Sleep
8. | End
```

### Έξοδος



```
300
```

## CLng

### Σύνταξη



Declare Function CLng ( ByVal expression As datatype ) As Long



## Χρήση

result = CLng( expression )

## Παράδειγμα - clng.bas



```
1. | 'Create an LONG variable
2. | Dim numeric_value As Long
3. | 'Convert a numeric value
4. | numeric_value = CLng(-300.23)
5. | 'Print the result, should return -300
6. | Print numeric_value
7. | Sleep
8. | End
```

## Έξοδος



-300

## CULng

### Σύνταξη



Declare Function CULng ( ByVal expression As datatype ) As ULong

## Χρήση

result = CULng( expression )

## Παράδειγμα - culng.bas



```
1. | 'Create an UNSIGNED LONG variable
2. | Dim numeric_value As ULong
3. | 'Convert a numeric value
4. | numeric_value = CULng(300.23)
5. | 'Print the result = 300
6. | Print numeric_value
7. | Sleep
8. | End
```

## Έξοδος



```
300
```

## CLngInt

### Σύνταξη



Declare Function CLngInt ( ByVal expression As datatype ) As LongInt

### Χρήση

result = CLngInt( expression )

### Παράδειγμα - clngint.bas



```
1. 'Create an LONG INTEGER variable
2. Dim numeric_value As LongInt
3. 'Convert a numeric value
4. numeric_value = CLngInt(-12345678.123)
5. 'Print the result, should return -12345678
6. Print numeric_value
7. Sleep
8. End
```

## Έξοδος



```
-12345678
```

# CULngInt

## Σύνταξη



Declare Function CULngInt ( ByVal expression As datatype ) As ULongInt

## Χρήση

result = CULngInt( expression )

## Παράδειγμα - culngint.bas



```
1. | 'Create an UNSIGNED LONG INTEGER variable
2. | Dim numeric_value As ULongInt
3. | 'Convert a numeric value
4. | numeric_value = CULngInt(12345678.123)
5. | 'Print the result, should return 12345678
6. | Print numeric_value
7. | Sleep
8. | End
```

## Έξοδος



```
12345678
```

## CSng

### Σύνταξη



Declare Function CSng ( ByVal expression As datatype ) As Single

### Χρήση

result = CSng( expression )

### Παράδειγμα - csng.bas



```
1. | 'Create an SINGLE variable
2. | Dim numeric_value As Single
3. | 'Convert a numeric value
4. | numeric_value = CSng(-12345.123)
5. | 'Print the result, should return -12345.123
6. | Print numeric_value
7. | Sleep
8. | End
```

### Έξοδος



```
-12345.123
```

## CDbl

### Σύνταξη



Declare Function CDbl ( ByVal expression As datatype ) As Double

### Χρήση

result = CDbl( expression )

## Παράδειγμα - cdbl.bas



```
1. 'Create an DOUBLE variable
2. Dim numeric_value As Double
3. 'Convert a numeric value
4. numeric_value = CDbI(-12345678.123)
5. 'Print the result, should return -12345678.123
6. Print numeric_value
7. Sleep
8. End
```

## Έξοδος



```
-12345678.123
```

## Str

### Σύνταξη



```
Declare Function Str ( ByVal n As Byte ) As String
Declare Function Str ( ByVal n As UByte ) As String
Declare Function Str ( ByVal n As Short ) As String
Declare Function Str ( ByVal n As UShort ) As String
Declare Function Str ( ByVal n As Long ) As String
Declare Function Str ( ByVal n As ULong ) As String
Declare Function Str ( ByVal n As LongInt ) As String
Declare Function Str ( ByVal n As ULongInt ) As String
Declare Function Str ( ByVal n As Single ) As String
Declare Function Str ( ByVal n As Double ) As String
Declare Function Str ( ByVal b As Boolean ) As String
Declare Function Str ( ByVal str As Const String ) As String
Declare Function Str ( ByVal str As Const WString ) As String
```

### Χρήση

```
result = Str( string )
```

## Παράδειγμα - str.bas



```
1. Dim a As Integer
2. Dim b As String
3. a = 8421
4. b = Str(a)
5. Print a, b
6. Sleep
7. End
```

## Έξοδος



```
8421 8421
```

## Val

### Σύνταξη



```
Declare Function Val ( ByRef str As Const String ) As Double
Declare Function Val ( ByRef str As Const WString ) As Double
```

### Χρήση

```
result = Val( strnum )
```

## Παράδειγμα - csng.bas



```
1. Dim a As String, b As Double
2. a = "2.1E+30xa211"
3. b = Val(a)
4. Print a, b
5. Sleep
6. End
```

## Έξοδος



```
2.1E+30xa211 2.1e+030
```

## Συμβολοσειρές (Strings) στη FreeBASIC

Υπάρχουν τρεις τύποι συμβολοσειρών στη FreeBASIC.

### String

Σταθερού και μεταβλητού μήκους μεταβλητή με ενσωματωμένη διαχείριση μνήμης.

### Σύνταξη



```
Dim variable As String [ * size]
```

Μια συμβολοσειρά είναι ένας πίνακας χαρακτήρων. Μια συμβολοσειρά δηλωμένη χωρίς το size αλλάζει μέγεθος δυναμικά. Το μέγεθος μπορεί να κυμαίνεται από 0 bytes μέχρι 2 gigabytes.

### Παράδειγμα



```
1.  '' Variable length
2.  Dim a As String, b As String
3.  a = "Hello"
4.  Print a
5.  a += ", world!"
6.  Print a
7.  b = "Welcome to FreeBASIC"
8.  Print b + "! " + a
9.  Sleep
10. End
```

## Έξοδος



```
Hello  
Hello, world!  
Welcome to FreeBASIC! Hello, world!
```

Για να ενώσουμε δύο συμβολοσειρές χρησιμοποιούμε το +

## ZString

Σταθερού και μεταβλητού μήκους μεταβλητή με τερματικό χαρακτήρα το NULL.

Μια συμβολοσειρά ZString είναι μια κλασική συμβολοσειρά τής γλώσσας C. Ο τελευταίος χαρακτήρας της συμβολοσειράς είναι το 0 ή ένας χαρακτήρας NULL και τοποθετείται αυτόματα από την γλώσσα FreeBASIC.

## Σύνταξη



```
Dim variable As ZString * size  
Dim variable As ZString Ptr
```

## Παράδειγμα



```
1. Dim As ZString * 13 str1 => "hello, world"  
2. Print str1  
3. Print Len(str1)      'returns 12, the size of the  
4.                    'string it contains  
5. Print SizeOf(str1)  'returns 13, the size of the  
6.                    'variable  
7. Sleep  
8. End
```



## Έξοδος



```
hello, world  
12  
13
```

### Ανάλυση

Στην γραμμή 1 δηλώνεται ένα ZString 13 χαρακτήρων και αρχικοποιείται με την τιμή “hello, world”.

Η εκχώρηση της τιμής στην συμβολοσειρά γίνεται με τον τελεστή =>

Στην γραμμή 2 τυπώνεται στην οθόνη το str1

Στην γραμμή 3 καλείται η συνάρτηση Len() με όρισμα το str1 και επιστρέφει το πλήθος των χαρακτήρων που περιέχει το str1. Αυτή η τιμή τυπώνεται στην οθόνη με την συνάρτηση Print.

Ομοίως στην γραμμή 4 καλείται η συνάρτηση SizeOf() με όρισμα το str1 και επιστρέφει το μέγεθος της συμβολοσειράς str1. Αυτή η τιμή τυπώνεται στην οθόνη με την συνάρτηση Print.

### WString

Σταθερού και μεταβλητού μήκους μεταβλητή με τερματικό χαρακτήρα το NULL και υποστήριξη για χαρακτήρες ευρείας μορφής. Υποστήριξη χαρακτήρων UNICODE.

Η συμβολοσειρά WString είναι όμοια τεχνικά με την ZString μόνο που εδώ μπορούμε να εκχωρήσουμε χαρακτήρες UNICODE ως τιμές.

Όταν η FreeBASIC προσπελάζει πηγαία αρχεία bas μπορεί να αναλύσει ASCII αρχεία με τον χαρακτήρα διαφυγής Unicode που είναι ο \u ή να διαβάσει απευθείας αρχεία UTF-8, UTF-16LE, UTF-16BE, UTF-32LE and UTF-32BE.

# Κεφάλαιο 3ο – Σταθερές

Οι σταθερές είναι αριθμοί, booleans, ή συμβολοσειρές strings οι οποίες δεν μπορούν να αλλάξουν από την στιγμή που θα δηλωθούν.

Για παράδειγμα, το 5 πάντα θα είναι ο ίδιος αριθμός.

Στη FreeBASIC μια δήλωση σταθεράς διαφέρει από μια δήλωση μεταβλητής από την χρήση της λέξης Const.

Τέτοιες σταθερές είναι ορατές παντού, που σημαίνει ότι μιας και δηλωθούν μπορείτε να προσπελάζετε την σταθερά με το όνομα της οπουδήποτε στον κώδικα.

Από την στιγμή που θα δηλωθεί μια σταθερά με την λέξη Const έπειτα δεν μπορεί να αλλάξει.

Αν προσπαθήσετε να αλλάξετε μια δηλωμένη σταθερά ένα μήνυμα λάθους θα προκύψει από τον μεταγλωττιστή

## Const

### Σύνταξη



```
Const symbolname1 [AS DataType] = value1 [,  
symbolname2 [AS DataType] = value2, ...]
```

ή

```
Const [AS DataType] symbolname1 = value1 [,  
symbolname2 = value2, ...]
```

### Παράδειγμα - const.bas



```
1. Const FirstNumber = 1  
2. Const SecondNumber = 2  
3. Const FirstString = "First string."  
4. Const FirstBoolean = False  
5. Const SecondBoolean = True  
6.  
7. Print FirstNumber, SecondNumber
```

```

8. | 'This will Print 1 2
9. | Print FirstString 'This will Print First string.
10. | Print FirstBoolean, SecondBoolean
11. | 'This will Print false true
12. |
13. | Sleep
14. | End

```

## Έξοδος



```

1 2
First string
false true

```

## Παράδειγμα - constcolors.bas



```

1. | Const Red = RGB(252, 2, 4)
2. | Const Black As UInteger = RGB(0, 0, 0)
3. | Const Text = "This is red text on a black bkgnd."
4. | Locate 1, 1
5. | Color Red, Black
6. | Print Text
7. | Sleep
8. | End

```

## Έξοδος



```

This is red text on a black bkgnd.

```

# Κεφάλαιο 4ο – Προτάσεις και εκφράσεις

Ένα πρόγραμμα αποτελείται από πλήθος εντολών που εκτελούνται γραμμή προς γραμμή.

Στην FreeBASIC υπάρχουν οι προτάσεις (**statements**) και οι εκφράσεις (**expressions**).

Μια πρόταση εκτελεί τρεις λειτουργίες, ελέγχει την σειρά της εκτέλεσης, υπολογίζει μια τιμή ή δεν κάνει τίποτα.

## Προτάσεις (statements)

Μια συνηθισμένη πρόταση για παράδειγμα είναι η εξής:

$a=b+c$

Σε αντίθεση με την άλγεβρα αυτή η πρόταση δεν σημαίνει ότι το  $a$  ισούται με το αποτέλεσμα της προσθέσεως του  $b$  και  $c$ .

Αυτή η πρόταση σημαίνει, πρόσθεσε το  $b$  και  $c$  και το αποτέλεσμα εκχώρησε το στην μνήμη που συμβολίζεται από την μεταβλητή  $a$ .

Αυτή η πρόταση αποτελείται από:

- τις μεταβλητές  $a,b,c$
- τους τελεστές  $=, +$ .

Τις μεταβλητές τις είδαμε σε προηγούμενο κεφάλαιο.

Οι τελεστές τελούν μια ενέργεια στις μεταβλητές.

Εδώ το  $+$  τελεί την πρόσθεση των  $b,c$  και το  $=$  θέτει το αποτέλεσμα στην μνήμη στην θέση που έχει οριστεί από την μεταβλητή  $a$ .

## Μπλοκ κώδικα

Ένα μπλοκ κώδικα είναι ένα μέρος κώδικα με αρχή και τέλος. Ένα μπλοκ κώδικα ξεκινά με μια πρόταση και τελειώνει με μια πρόταση τέλους.

Στην FreeBASIC υπάρχουν έξι είδη μπλοκ κώδικα.

- Μπλοκ ρουτίνας
- Μπλοκ συναρτήσεων
- Μπλοκ επιλογής
- Μπλοκ υπόθεσης
- Μπλοκ “Με”
- Μπλοκ επανάληψης

Μέσα σε κάθε μπλοκ γράφεται κώδικας όπου εκτελείται από την αρχή του μπλοκ μέχρι το τέλος.

Εδώ απλά θα αναφέρουμε τα είδη των μπλοκ και σε επόμενο κεφάλαιο θα γίνει μελέτη κάθε μπλοκ χωριστά.

Τα μπλοκ που αναφέρθηκαν είναι:

SUB ... END

FOR ... NEXT

DO .. LOOP

FUNCTION ... END

WHILE ... WEND

SELECT CASE ... END

SELECT

IF ... THEN ... END IF

WITH ... END WITH

## Εκφράσεις (expressions)

Οτιδήποτε υπολογίζεται ως μία τιμή αποτελεί μια έκφραση. Μια έκφραση επιστρέφει μια τιμή ενός τύπου δεδομένων.

Παραδείγματα εκφράσεων:

4.5        *'επιστρέφει την αριθμητική τιμή 4.5*

Pi        *'επιστρέφει την τιμή της double μεταβλητής Pi*

MyName   *'επιστρέφει κείμενο*

a=b+c    *'υπολογίζει το άθροισμα και επιστρέφει την τιμή στην μεταβλητή a.*

## Κεφάλαιο 5ο – Πίνακες

Οι πίνακες (**Arrays**) είναι ειδικές μεταβλητές οι οποίες δρουν σαν δοχεία για έναν αριθμό τιμών ή στοιχείων. Ένας πίνακας μπορεί να αποθηκεύσει στοιχεία κάθε τύπου δεδομένων και όλα τα στοιχεία του πίνακα είναι πάντα του ίδιου τύπου δεδομένων.

Για παράδειγμα ένας πίνακας μπορεί να αποθηκεύσει στοιχεία τύπου Integer ή Single αλλά όχι και τα δύο μαζί. Τα στοιχεία αυτά προσπελάζονται για διάβαση ή γράψιμο μέσω ενός ακεραίου αριθμού που σηματοδοτεί την θέση κάθε στοιχείου στο πίνακα. Οι πίνακες έχουν μήκος ή μέγεθος το οποίο είναι ίσο με το πλήθος των στοιχείων που έχουν. Υπάρχουν δύο είδη πινάκων. Πίνακες σταθερού μεγέθους και πίνακες μεταβλητού μεγέθους.

Οι πίνακες σταθερού μεγέθους έχουν για όλη την διάρκεια ύπαρξης τους σταθερό αριθμό στοιχείων και άρα σταθερό μέγεθος.

Οι πίνακες μεταβλητού μεγέθους έχουν μεταβλητό αριθμό στοιχείων και άρα μεταβλητό μέγεθος. Οι πίνακες αυτοί μπορούν να αλλάζουν το μέγεθος τους δυναμικά.

### Στοιχεία και θέσεις

Οι τιμές τις οποίες ένας πίνακας αποθηκεύει είναι τα στοιχεία του. Κάθε στοιχείο του πίνακα έχει συγκεκριμένη θέση η οποία είναι ένας ακεραίος αριθμός που κυμαίνεται από το κατώτερο όριο του πίνακα έως το ανώτερο όριο του πίνακα.

Η προσπέλαση των θέσεων ενός πίνακα γίνεται γράφοντας το όνομα του πίνακα ακολουθούμενο από ( ) και εντός των παρενθέσεων την τιμή της θέσης που θέλουμε να προσπελάσουμε.

# Δηλώνοντας Πίνακες

## Σύνταξη



Dim name ( [lbound To] ubound [, ...] ) As DataType  
Dim name ( Any [, Any...] ) As DataType  
Dim name ( ) As DataType

## Αρχικοποίηση



Dim array\_symbol (arraybounds) As DataType =  
{ expression [, ...] } | Any

## Παράδειγμα - array1.bas



```
1. ' Create an array of 3 elements.  
2. Dim array(1 To 3) As Single  
3. ' Assign a value to the first element.  
4. array(1) = 1.2  
5. ' Output the values of all the elements ("1.2 0  
6. '0").  
7. Print array(1)  
8. Print array(2)  
9. Print array(3)  
10. Sleep  
11. End
```

## Έξοδος



```
1.2  
0  
0
```

## Ανάλυση

Στην γραμμή 2 ορίζεται ένας πίνακας με το όνομα array και κατώτερο όριο το 1 και ανώτερο όριο το 3. Ο πίνακας array δηλδ έχει 3 στοιχεία το 1,2,3.

Στην γραμμή 4 προσπελάζεται ο πίνακας στην θέση 1 για γράψιμο. Τίθεται στην θέση 1 η τιμή 1.2

Στις γραμμές 6,7,8 τυπώνεται στην οθόνη η τιμή που έχει ο πίνακας για τα στοιχεία 1,2,3.

## Παράδειγμα - array2.bas



```
1. | ' Declares and initializes an array of four  
2. | ' integer elements.  
3. | Dim array(3) As Integer = { 10, 20, 30, 40 }  
4. | Print array(0)  
5. | Print array(1)  
6. | Print array(2)  
7. | Print array(3)  
8. | Sleep  
9. | End
```

## Έξοδος



```
10  
20  
30  
40
```

## Ανάλυση

Στην γραμμή 3 δηλώνεται ένας πίνακας Integer 4 θέσεων. Το ανώτερο όριο του πίνακα είναι το 3 αλλά το κατώτερο όριο το οποίο είναι εξ' ορισμού 0 όταν δεν ορίζεται.

Η δήλωση: Dim array(3) ισοδυναμεί με την δήλωση Dim array(0 To 3).

Έτσι ο πίνακας array έχει 4 στοιχεία τα 0,1,2,3,4.

Αυτά τα στοιχεία αρχικοποιούνται επίσης στην γραμμή 3.

Στις γραμμές 4,5,6,7 τυπώνονται στην οθόνη οι τιμές των στοιχείων του πίνακα.



# Πίνακες σταθερού και μεταβλητού μεγέθους

Μέχρι τώρα είδαμε δηλώσεις πινάκων σταθερού μεγέθους.

Η δήλωση `Dim array(1 To 3) As Integer` δηλώνει έναν πίνακα σταθερού μεγέθους τύπου `Integer`.

Η δήλωση `Dim array() As Integer` δηλώνει έναν πίνακα δυναμικού μεγέθους.

Οι πίνακες δυναμικού μεγέθους μετά την δήλωση τους πρέπει να ορίσουν τις διαστάσεις τους με την λέξη `ReDim`.

## Παράδειγμα



1. `' Creates an empty variable-length array`
2. `' that holds integer values.`
3. `Dim v1array() As Integer`
4. `' Resizes the array to 10 elements.`
5. `ReDim v1array(1 To 10)`

## Πίνακες πολλαπλών διαστάσεων

Μέχρι τώρα είδαμε πίνακες με μία διάσταση όπου τα στοιχεία τους προσπελάζονταν με μια τιμή.

Οι πίνακες μονής διάστασης μπορεί να ειπωθούν σαν πίνακες στοιχείων απλής γραμμής.

Οι πίνακες μπορούν να έχουν από μία μέχρι 8 διαστάσεις στην `FreeBASIC`. Πίνακες με πάνω από μία διάσταση ονομάζονται πίνακες πολλαπλών διαστάσεων.

Οι πίνακες 2 διαστάσεων μπορούν να ειπωθούν σαν πίνακες που έχουν μια γραμμή και μια κολώνα.

Οι πίνακες 3 διαστάσεων επιπλέον έχουν και το βάθος.

Οι πίνακες 4 διαστάσεων μπορούν να θεωρηθούν ως πολλοί τρισδιάστατοι πίνακες κτλ.

## Παράδειγμα - multyarray.bas



```
1. ' Take Care while initializing
2. ' multi-dimensional array
3. Dim As Integer multidim(1 To 2, 1 To 5) = _
4. {{1,2,3,4,5},{6,7,8,9,10}}
5.
6. Print multidim(1, 3)
7. Print multidim(2, 3)
8. Sleep
9. End
```

## Έξοδος



```
3
8
```

## Ανάλυση

Στην γραμμή 3 ορίζεται ένας δισδιάστατος πίνακας με διαστάσεις 1 μέχρι 2 για την πρώτη διάσταση και 1 μέχρι 5 για την δεύτερη.

Το σύμβολο \_ (κάτω παύλα) σημαίνει ότι η γραμμή συνεχίζεται παρακάτω επειδή δεν χωράει στον επεξεργαστή κειμένου.

Στην γραμμή 4 γίνεται η αρχικοποίηση του πίνακα.

Η πρώτη διάσταση έχει 2 στοιχεία ενώ η δεύτερη διάσταση έχει 5. Έτσι ο πίνακας έχει 2 5άδες συνολικά.

Στην γραμμή 6 τυπώνεται στην οθόνη από την πρώτη πεντάδα το τρίτο στοιχείο.

Στην γραμμή 7 τυπώνεται στην οθόνη από την δεύτερη πεντάδα το τρίτο στοιχείο.

# Οι συναρτήσεις LBound και UBound

## LBound

### Σύνταξη



result = LBound( array [, dimension ] )

Επιστρέφει την κατώτερη τιμή της διάστασης ενός πίνακα.

### Παράδειγμα - lbound.bas



```
1. 'Declare an array
2. Dim array(1 To 5) As Integer
3. Print LBound(array)
4. Sleep
5. End
```

### Έξοδος



```
1
```

## UBound

### Σύνταξη



result = UBound( array [, dimension ] )

Επιστρέφει την ανώτερη τιμή της διάστασης ενός πίνακα.

## Παράδειγμα - ubound.bas



```
1. 'Declare an array
2. Dim array(1 To 5) As Integer
3. Print UBound(array)
4. Sleep
5. End
```

## Έξοδος



```
5
```

# Κεφάλαιο 6ο - Ροή προγράμματος

## Εντολές λήψης λογικών αποφάσεων

### IIf

Ελέγχει αν μια έκφραση είναι αληθής και επιστρέφει την πρώτη τιμή όταν είναι αληθής αλλιώς την δεύτερη τιμή αν είναι ψευδής η έκφραση.

### Σύνταξη



```
value = IIf ( condition, expr_if_true, expr_if_false )
```

value: Η επιστρεφόμενη τιμή, μπορεί να είναι οποιαδήποτε αριθμητική τιμή ή συμβολοσειρά ή Τύπος δεδομένων του χρήστη UDT.

Τους UDT (User Defined Types) θα τους δούμε στην συνέχεια παρακάτω.

condition: η συνθήκη η οποία ελέγχεται για το αν είναι μη μηδενική ώστε να επιστρέψει η συνάρτηση την τιμή `expr_if_true` ή αν είναι μηδενική επιστρέφει την τιμή `expr_if_false`

Οι `expr_if_true` και `expr_if_false` είναι οποιαδήποτε αριθμητική τιμή ή συμβολοσειρά ή UDT.

## Παράδειγμα - `iff.bas`



```
1. Dim value As Integer
2. Dim b As Boolean
3. b = True
4. value = IIf(b, 10, 20)
5. Print value
6. b = False
7. value = IIf(b, 10, 20)
8. Print value
9. Sleep
10. End
```

## Έξοδος



```
10
20
```

## Ανάλυση

Στην γραμμή 1 δηλώνεται η μεταβλητή `value` ως ακέραιος.

Στην γραμμή 2 δηλώνεται η μεταβλητή `b` ως `boolean`

Στην γραμμή 3 τίθεται η τιμή `True` στην μεταβλητή `b`.

Στην γραμμή 4 καλείται η `IIf` που ελέγχει την μεταβλητή `b`.

Η `IIf` επιστρέφει την τιμή 10 διότι η μεταβλητή `b` έχει την τιμή `True`.

Στην γραμμή 5 τυπώνεται η τιμή της μεταβλητής `value` στην οθόνη.

Στην γραμμή 6 θέτουμε την τιμή `False` στην μεταβλητή `b`.

Στην γραμμή 7 ξανακαλείται η `IIf` και αυτήν την φορά επιστρέφει το 20 διότι η μεταβλητή `b` έχει την τιμή `False`.

Στην γραμμή 8 τυπώνεται στην οθόνη η τιμή της `value`.

Τέλος έχουμε τις εντολές αναμονής του χρήστη, Sleep και τέλος του προγράμματος End.

## **IF ... THEN ... ELSE IF ...THEN ... ELSE ... ENDIF**

Το IF...THEN...ELSE IF...THEN...ENDIF ονομάζεται έκφραση συνθήκης.

Μετά το IF ελέγχεται μία πρόταση αν είναι αληθής.  
Αν είναι εκτελείται ο κώδικας μετά το THEN.

Διαφορετικά μπορούμε να κάνουμε και άλλον λογικό έλεγχο αληθείας με την ELSE IF.  
Μπορούμε να έχουμε όσους λογικούς ελέγχους θέλουμε με το ELSE IF.  
Αν κάποια έκφραση είναι αληθής από το ELSE IF εκτελείται ο κώδικας μετά το THEN.

Αν τελικά καμία IF ή ELSE IF δεν είναι αληθής μπορούμε να εκτελέσουμε τον κώδικα που ακολουθεί την ELSE.

Το μπλοκ κώδικα τελειώνει με την λέξη ENDIF.

### **Σύνταξη**



```
If expression Then  
    [statement(s)]  
[ ElseIf expression Then ]  
    [statement(s)]  
[ Else ]  
    [statement(s)]  
End If
```

## Παράδειγμα - if1.bas



```
1. Dim As Integer a = 10
2. If a = 10 Then
3.     Print "Η τιμή του a είναι 10"
4. End If
5.
6. If a > 20 Then
7.     Print "Η τιμή του a είναι μεγαλύτερη του 20"
8. Else
9.     Print "Η τιμή του a είναι μικρότερη του 20"
10. End If
11. Sleep
12. End
```

## Έξοδος



```
Η τιμή του a είναι 10
Η τιμή του a είναι μικρότερη του 20
```

## Ανάλυση

Στη γραμμή 1 δηλώνεται ένας ακέραιος και αρχικοποιείται με την τιμή 10.

Στις γραμμές 2-4 υπάρχει ένα If...End If το οποίο ελέγχει αν η μεταβλητή a ισούται με την τιμή 10. Αν ισούται τυπώνεται ένα μήνυμα στην οθόνη.

Στις γραμμές 5-10 υπάρχει ένα If...Else...End If το οποίο ελέγχει αν η μεταβλητή a είναι μεγαλύτερη από την τιμή 20. Αν είναι μεγαλύτερη από το 20 τότε τυπώνεται ένα αντίστοιχο μήνυμα στην οθόνη. Διαφορετικά το πρόγραμμα περνάει στην γραμμή 9 όπου τυπώνεται ένα μήνυμα στην οθόνη που αναφέρει ότι η τιμή είναι μικρότερη του 20.

Στην γραμμή 10 τελειώνει το If.

Τέλος έχουμε την εντολή αναμονής του χρήστη και το τέλος του προγράμματος.

## Παράδειγμα - if2.bas



```
1. Dim a As String
2. Line Input "Δώστε έναν ακέραιο: ", a
3. If Cint(a) > 10 Then
4.     Print "Δώσατε τιμη μεγαλύτερη του 10"
5. ElseIf Cint(a) < 10 Then
6.     Print "Δώσατε τιμη μικρότερη του 10"
7. ElseIf Cint(a) = 10 Then
8.     Print "Δώσατε τιμη ίση με 10"
9. End If
10. Sleep
11. End
```

### Ανάλυση

Στη γραμμή 1 δηλώνουμε μια μεταβλητή τύπου String  
Στη γραμμή 2 η εντολή Line Input προτρέπει τον χρήστη με το μήνυμα “Δώστε έναν ακέραιο: “ και περιμένει είσοδο από τον χρήστη. Ο χρήστης πληκτρολογεί ένα ακέραιο και η τιμή του εκχωρείται στην μεταβλητή a.

Στην γραμμή 3 ξεκινάει ένας έλεγχος If...End If.

Θέλουμε να ελέγξουμε αν ο χρήστης έχει βάλει τιμή μεγαλύτερη του 10. Η μεταβλητή a είναι τύπου String και η τιμή που εκχώρησε ο χρήστης στο πρόγραμμα υπολογίζεται ως συμβολοσειρά και όχι ως αριθμός.

Για τον λόγο αυτό μετατρέπουμε την τιμή της συμβολοσειράς σε ακέραιο με την συνάρτηση Cint().

Αν η τιμή που έβαλε ο χρήστης είναι μεγαλύτερη από 10 τότε εκτελείται η γραμμή 4 και τυπώνεται στην οθόνη το αντίστοιχο μήνυμα.

Αν η τιμή που έβαλε ο χρήστης είναι μικρότερη από 10, έλεγχος της γραμμής 5, τότε εκτελείται η γραμμή 6 και τυπώνεται στην οθόνη το αντίστοιχο μήνυμα.

Τέλος στην γραμμή 7 ελέγχεται αν η τιμή είναι ίση με το 10. Τότε το πρόγραμμα εκτελεί τη γραμμή 8 και τυπώνει στην οθόνη το αντίστοιχο μήνυμα.

Τέλος έχουμε την αναμονή του χρήστη και το τέλος του προγράμματος.



## Select Case ... End Select

Η Select είναι η έκφραση επιλογής.

Συγκρίνει μία έκφραση με περιπτώσεις άλλων εκφράσεων και όταν βρεθεί η σύγκριση να είναι αληθής εκτελείται ο κώδικας σε εκείνο το μπλοκ κώδικα.

### Σύνταξη



```
Select Case expression  
[ Case expressionlist]  
[statements]  
[ Case Else ]  
[statements]  
End Select
```

Μπορεί να χρησιμοποιηθεί η έκφραση **Exit Select** για να βγει η ροή του προγράμματος από ένα μπλοκ **Select Case**.

Η έκφραση σε κάθε Case μπορεί να είναι από μέχρι ή να έχει συγκεκριμένη τιμή.

### Σύνταξη **expressionlist**



```
{ expression | expression To expression | Is  
relational operator expression }[, ...]
```

## Παράδειγμα - select.bas



```
1. Dim choice As Integer
2. Input "Choose a number between 1 and 10: ";_
3. choice
4. Select Case As Const choice
5. Case 1
6.     Print "number is 1"
7. Case 2
8.     Print "number is 2"
9. Case 3, 4
10.    Print "number is 3 or 4"
11. Case 5 To 10
12.    Print "number is in the range of 5 to 10"
13. Case Else
14.    Print "number is outside the 1-10 range"
15. End Select
16. Sleep
17. End
```

### Ανάλυση

Στη γραμμή 1 δηλώνεται ένας ακέραιος.

Στην γραμμή 2 έχουμε είσοδο από τον χρήστη ενός ακεραίου από το 1 έως το 10.

Στην γραμμή 4 αρχίζει το Select Case.

Στην γραμμή 5 ελέγχεται αν η μεταβλητή choice έχει την τιμή 1. Αν την έχει τότε εκτελείται η γραμμή 6 όπου τυπώνει ένα μήνυμα στην οθόνη.

Ομοίως στην γραμμή 7 ελέγχεται αν η τιμή είναι 2, στην γραμμή 9 ελέγχεται αν έχει τις τιμές 3 ή 4. Στην γραμμή 11 ελέγχεται αν έχει την τιμή από 5 έως 10.

Τέλος στην γραμμή 13 υπάρχει ένα Case Else για οποιαδήποτε άλλη περίπτωση.

Οι γραμμές 16 και 17 είναι η αναμονή του χρήστη και το τέλος του προγράμματος.

### Εντολές βρόχων

Ο βρόχος είναι μια επανάληψη. Ο κώδικας που βρίσκεται μέσα σε έναν βρόχο επαναλαμβάνεται μέχρι να ικανοποιήσει μια συνθήκη και η ροή του προγράμματος να βγει από τον βρόχο.

## For...Next

Επαναλαμβάνει ένα μπλοκ κώδικα καθώς μεταβάλλει την τιμή μιας μεταβλητής.

### Σύνταξη



```
For iterator [ As datatype ] = startvalue To  
endvalue [ Step stepvalue ]  
[ statement block ]  
Next [ iterator ]
```

### Παράδειγμα - for1.bas



```
1. For i As Integer = 1 To 10  
2.     Print i  
3. Next i  
4. Sleep  
5. End
```

### Έξοδος



```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

### Ανάλυση

Στις γραμμές 1-3 βρίσκεται ο βρόγχος επανάληψης For...Next.

Στην γραμμή 1 δηλώνεται η μεταβλητή i ως ακέραιος που θα πάρει τις τιμές από 1 έως 10.

Στην γραμμή 2 τυπώνεται η τιμή της μεταβλητής i στην οθόνη. Η γραμμή 2 θα εκτελεστεί 10 φορές όπως έχει οριστεί από το βρόγχο επανάληψης.

Η γραμμή 3 αυξάνει την μεταβλητή i κατά μία μονάδα.

Τέλος υπάρχει η αναμονή του χρήστη και το τέλος του προγράμματος.

## Λίγα λόγια για τις ένθετες προτάσεις

Όλες οι δομές ροής προγράμματος συνήθως είναι μπλοκ κώδικα που έχουν μια αρχή και ένα τέλος.

Σε μια δομή μπλοκ όλες οι προτάσεις που υπάρχουν εντός της ονομάζονται ένθετες προτάσεις ή αλλιώς φωλιασμένες (nested) όπως λένε οι προγραμματιστές απλά.

Το φώλιασμα (nesting) πρόκειται για την ενσωμάτωση μιας δομής μέσα σε μια άλλη, όπως για παράδειγμα η ένθεση ενός βρόχου σε άλλους βρόχους.

Κατά την συγγραφή του κώδικα μια φωλιασμένη πρόταση γράφεται με μια εσοχή προς τα δεξιά σε σχέση με την στοίχιση της αρχής του μπλοκ που ενυπάρχει.

## Παράδειγμα - for2.bas



```
1. ' External For loop
2. For i As Integer = 1 To 5
3. ' Nested For loop
4.     For a As Integer = 1 To 2
5.         Print "i="; i; " a="; a
6.     Next a
7. Next i
8. Sleep
9. End
```

Στο παράδειγμα for2.bas φαίνεται το φώλιασμα ενός βρόγχου For μέσα σε άλλο βρόγχο For.

## Έξοδος



```
i= 1 a= 1
i= 1 a= 2
i= 2 a= 1
i= 2 a= 2
i= 3 a= 1
i= 3 a= 2
i= 4 a= 1
i= 4 a= 2
i= 5 a= 1
i= 5 a= 2
```

## Do...Loop

Επαναλαμβάνει ένα μπλοκ κώδικα όσο η αρχική πρόταση παραμένει αληθής, ή μέχρι η τελική πρόταση γίνει αληθής

### Σύνταξη



```
Do [ { Until | While } condition ]
  [ statement block ]
Loop
```

ή

```
Do
  [ statement block ]
Loop [ { Until | While } condition ]
```

Αν υπάρχει μια **Exit Do** πρόταση στον βρόγχο τότε η ροή του προγράμματος βγαίνει από τον βρόγχο αμέσως

Αν υπάρχει μια **Continue Do** πρόταση στον βρόγχο τότε παραλείπεται η εκτέλεση του statement block και η ροή μεταφέρεται στο **Do** για να εκτελεστεί ο βρόγχος κατά την επόμενη επανάληψη.

## Παράδειγμα - `getkey.bas`



```
1. Dim As Long key
2. Do
3.     Print "Press any key from keyboard"
4.     Sleep
5.     key = GetKey
6.     Print "key: " & Chr(key) & "=" & key
7.     Print "Press q for quit"
8. Loop Until Chr(key) = "q"
9. End
```

## Έξοδος



```
Press any key from keyboard
key: a=97
Press q for quit
Press any key from keyboard
key: q=113
Press q for quit
```

## Ανάλυση

Στην γραμμή 1 δηλώνεται μια μεταβλητή Long με το όνομα `key`.

Στην γραμμή 2 ξεκινάει ο βρόγχος `Do` και τελειώνει στην γραμμή 8.

Στην γραμμή 3 τυπώνεται ένα μήνυμα στην οθόνη να πατήσει ο χρήστης ένα πλήκτρο από το πληκτρολόγιο.

Στην γραμμή 4 το πρόγραμμα μπαίνει σε αναμονή.

Στην γραμμή 5 η συνάρτηση `GetKey` λαμβάνει το πάτημα του πληκτρολογίου και εκχωρεί τον κωδικό ASCII ως αριθμό Long στη μεταβλητή `key`.

Στη γραμμή 6 η συνάρτηση `Chr()` μετατρέπει την αριθμητική τιμή `key` στον αντίστοιχο χαρακτήρα ASCII.

Έτσι εμφανίζεται στην οθόνη η συμβολοσειρά `key: a=97` αν ο χρήστης πληκτρολογήσει το `a`. Οπου `key=97` και `Chr(97) = a`.

Στη γραμμή 7 εμφανίζεται στην οθόνη ένα μήνυμα προς τον χρήστη ότι αν πατήσει το πλήκτρο `q` θα τερματίσει το πρόγραμμα.

Στη γραμμή 8 είναι το τέλος του βρόγχου. Ελέγχεται αν η μεταβλητή key έχει αντίστοιχα το γράμμα q από τον ASCII πίνακα.

Τέλος έχουμε το τέλος του προγράμματος στην γραμμή 9.

## While ... Wend

Ο βρόχος αυτό εκτελεί το μπλοκ κώδικα όσο η συνθήκη ελέγχου είναι αληθής.

Αν η συνθήκη ελέγχου είναι ψευδής δεν εκτελεί καθόλου το μπλοκ κώδικα που περικλείει.

### Σύνταξη



```
While [condition]
  [statement block]
Wend
```

Αν υπάρχει μια **Exit While** πρόταση στον βρόγχο τότε η ροή του προγράμματος βγαίνει από τον βρόγχο αμέσως

Αν υπάρχει μια **Continue While** πρόταση στον βρόγχο τότε παραλείπεται η εκτέλεση του statement block και η ροή μεταφέρεται στο **While** για να εκτελεστεί ο βρόγχος κατά την επόμενη επανάληψη.

## Εντολή μετάβασης Goto

Οδηγεί την ροή του προγράμματος σε ένα σημείο με μια ετικέτα.

### Σύνταξη



```
Goto label
```

Goto, απαραίτητη λέξη που είναι η εντολή μετάβασης Label, το όνομα μιας ετικέτας.

Μια ετικέτα είναι ένα στόχος για την εντολή Goto.

Μια εντολή Goto μπορεί να χρησιμοποιηθεί για να αλλάξει η ροή του προγράμματος αφήνοντας μια δομή ελέγχου όπως ένα μπλοκ π.χ. For...Next.

## Παράδειγμα - goto.bas



```
1. Dim i As Integer
2. For i = 0 To 10
3.     Print i
4.     If i = 5 Then Goto OutOfFor
5. Next i
6.
7. OutOfFor:
8.     Print "I am out of FOR...NEXT now."
9.
10. Print "Next lines continue here."
11. Sleep
12. End
```

## Έξοδος



```
0
1
2
3
4
5
I am out of FOR...NEXT now.
Next lines continue here.
```

## Ανάλυση

Στην γραμμή 1 δηλώνεται ένας ακέραιος.

Στις γραμμές 2-5 υπάρχει ένα For...Next από το 0 μέχρι το 10.

Στην γραμμή 3 τυπώνεται η τιμή του μετρητή i στην οθόνη.

Στην γραμμή 4 γίνεται ένας έλεγχος αν ο μετρητής είναι

ίσος με 5. Αν η πρόταση αληθεύει μεταφέρεται η ροή με την Goto στην ετικέτα OutOfFor.



Έτσι τυπώνονται στην οθόνη οι αριθμοί από το 1 έως το 5 και μετά η ροή του προγράμματος μεταφέρεται στην ετικέτα OutOfFor όπου στην γραμμή 8 τυπώνεται στην οθόνη το μήνυμα ότι έχουμε μεταφερθεί εκτός βρόγχου For...Next. Η ροή του προγράμματος συνεχίζει μετά την ετικέτα OutOfFor στην γραμμή 10 όπου τυπώνεται το αντίστοιχο μήνυμα στην οθόνη. Τέλος έχουμε την αναμονή του χρήστη και το τέλος του προγράμματος.

## Διαχείριση λαθών

Η FreeBASIC διαχειρίζεται τα λάθη κατά την εκτέλεση του προγράμματος με τους παρακάτω τρόπους:

- Εξ' ορισμού το πρόγραμμα δεν κάνει κάτι με τα λάθη. Σιωπηρά αυτά αγνοούνται και ο κώδικας συνεχίζει.
- Εάν μεταγλωττίζετε το πρόγραμμα με τους διακόπτες -e, -ex ή -exx, FreeBASIC χρησιμοποιεί την διαχείριση λαθών όμοια με αυτήν της QB.
- Μελλοντικά οι OOP εκδόσεις της FreeBASIC θα έχουν διαχείριση λαθών όμοια με αυτήν της java, δλδ το μπλοκ TRY..CATCH...FINALLY.

## Παράδειγμα - onerror.bas



```
1. #lang "fblite"
2. On Error Goto FAILED
3. Open "xzxwz.zwz" For Input As #1
4. On Error Goto 0
5. Print "Program continue here..."
6. Sleep
7. End
8.
9. FAILED:
10. Dim e As Integer
11. e = Err
12. Print "Error " & e & " on line " & ErL
13. Resume Next
14. End
```

Εάν έχουν οριστεί οι διακόπτες του μεταγλωττιστή -e, -ex ή -exx τότε το πρόγραμμα περιμένει να έχει μία ετικέτα ως χειριστή λαθών. Όπως φαίνεται στο παραπάνω παράδειγμα η ετικέτα FAILED είναι ο διαχειριστής λαθών.

Στην γραμμή 1 ορίζουμε ότι θα χρησιμοποιήσουμε την διάλεκτο fblite έτσι ώστε να μας επιτραπεί η χρήση της πρότασης Resume Next.

Στην γραμμή 2 υπάρχει η πρόταση On Error Goto η οποία δηλώνει την ετικέτα FAILED ως διαχειριστή λάθους. Στην περίπτωση που εμφανιστεί ένα λάθος η ροή του προγράμματος πηγαίνει στην ετικέτα FAILED.

Στην γραμμή 3 προσπαθούμε να ανοίξουμε ένα αρχείο που δεν υπάρχει για γράψιμο εντός του. Επειδή το αρχείο δεν υπάρχει ο μεταγλωττιστής εμφανίζει ένα λάθος. Η ροή του προγράμματος μεταβαίνει στην ετικέτα FAILED.

Στην γραμμή 10 δηλώνεται ένας ακέραιος.

Στην γραμμή 11 η συνάρτηση Err εκχωρεί στην μεταβλητή e έναν ακέραιο αριθμό που δείχνει ποιο λάθος προέκυψε.

Στην γραμμή 12 η συνάρτηση Err επιστρέφει την γραμμή στο κώδικα που εμφανίστηκε το λάθος. Έτσι τυπώνεται στην οθόνη το μήνυμα "Error 2 on line 2"

Στην γραμμή 13 υπάρχει εντός του διαχειριστή λάθους η πρόταση Resume Next ή οποία επιστρέφει την ροή του προγράμματος μετά το λάθος στην επόμενη γραμμή από την οποία εμφανίστηκε το λάθος, δηλ την γραμμή 5.

Έτσι στην γραμμή 5 εμφανίζεται ένα μήνυμα επιστροφής στην οθόνη.

Η FreeBASIC ορίζει τα παρακάτω λάθη εκτέλεσης.

- 0 No error
- 1 Illegal function call
- 2 File not found signal
- 3 File I/O error
- 4 Out of memory
- 5 Illegal resume
- 6 Out of bounds array access
- 7 Null Pointer Access
- 8 No privileges
- 9 interrupted signal
- 10 illegal instruction signal
- 11 floating point error signal
- 12 segmentation violation signal
- 13 Termination request signal
- 14 abnormal termination signal
- 15 quit request signal
- 16 return without gosub
- 17 end of file

Έτσι βλέπουμε ότι το λάθος με την τιμή 2 είναι το File not found signal.

## **Εντολές END, RETURN, STOP, SLEEP, EXIT**

Υπάρχουν κάποιες εντολές της FreeBASIC που τερματίζουν ή αλλάζουν την ροή του προγράμματος είτε μιας συνάρτησης είτε όλου του προγράμματος, είτε ενός βρόχου κτλ.

Αυτές οι προτάσεις λίγο πολύ μοιάζουν και ένας αρχάριος μπορεί να μπερδευτεί με τις παραπλήσιες ονομασίες τους.

Αυτές οι εντολές είναι οι **END, RETURN, STOP, SLEEP, EXIT**

Σε αυτή την ενότητα θα αναφέρουμε τις εντολές αυτές και θα εξηγηθεί η χρήση κάθε μίας.

## End

Η πρόταση End χρησιμοποιείται με δύο τρόπους στη FreeBASIC.

Ο πρώτος τρόπος είναι για να δηλώσει το τέλος του προγράμματος.

### Σύνταξη



End [ retval ]

Όπου retval ο κωδικός επιστροφής λάθους.

Συνήθως όταν δεν υπάρχει λάθος η τιμή είναι 0.

Ο δεύτερος τρόπος είναι για να δηλώσει το τέλος ενός μπλοκ κώδικα.

### Σύνταξη



End { Sub | Function | If | Select | Type | Enum |  
Scope | With | Namespace | Extern | Constructor |  
Destructor | Operator | Property }

## Return

Επιστρέφει από μια υπορουτίνα ή συνάρτηση.

### Σύνταξη



Return expression

Όταν βρίσκεται εντός μιας υπορουτίνας απλά επιστρέφει από αυτήν χωρίς να επιστρέφει κάποια τιμή.

Όταν βρίσκεται εντός συνάρτησης επιστρέφει μια τιμή από την συνάρτηση.

Περισσότερα για τις υπορουτίνες και συναρτήσεις στην συνέχεια στο επόμενο κεφάλαιο.

## Stop

Σταματάει την εκτέλεση του προγράμματος και περιμένει να πατηθεί ένα πλήκτρο από τον χρήστη πριν τερματίσει το πρόγραμμα.

### Σύνταξη



Stop

## Sleep

Αναμένει για την παρέλευση ενός χρονικού διαστήματος ή το πάτημα ενός πλήκτρου από το πληκτρολόγιο.

### Σύνταξη



Sleep [ amount [, keyflag ] ]  
result = Sleep ( amount, keyflag )

amount, ο χρόνος αναμονής σε milisecond  
keyflag, όταν είναι 0 είναι εξ' ορισμού η αναμονή για το πάτημα ενός πλήκτρου του πληκτρολογίου, όταν είναι 1 ορίζει ότι η αναμονή δεν μπορεί να περιμένει να πατηθεί κάποιο πλήκτρο από το πληκτρολόγιο.

## Exit

Πρόταση ελέγχου ροής του προγράμματος. Η πρόταση συνοδεύεται με άλλες λέξεις που δηλώνουν μπλοκ κώδικα και εξέρχεται από το μπλοκ αυτό.

## Σύνταξη



Exit {Do | For | While | Select }  
Exit {Sub | Function | Operator | Constructor |  
Destructor | Property }

## Κεφάλαιο 7ο – Ρουτίνες, συναρτήσεις, διαδικασίες

Μια ρουτίνα είναι ένα κομμάτι (block) κώδικα που έχει αρχή και τέλος, αποτελεί μέρος του κώδικα μέσα σε ένα μεγάλο πρόγραμμα και τελεί μια συγκεκριμένη λειτουργία.

Οι ρουτίνες σε διάφορες γλώσσες ονομάζονται ως procedure, method, function, routine, ή subroutine.

Στην FreeBASIC έχουμε τα εξής είδη ρουτινών.

Ρουτίνες που δημιουργεί ο προγραμματιστής. Αυτές προσπελάζονται ως SUB, FUNCTION.

## SUB

### Σύνταξη



```
[Public|Private] Sub identifier [cdecl|pascal|stdcall]  
[Overload] [Alias external_identifier]  
[( [parameter_list] )] [Static] [Export]  
statements  
...  
[Return]  
...  
End Sub
```

Η λέξη **Sub** σηματοδοτεί την αρχή μιας υπορουτίνας και το τέλος της ορίζεται από τις λέξεις **End Sub**.

Κάθε υπορουτίνα έχει ένα όνομα με το οποίο την καλούμε από διάφορα σημεία μέσα στον κώδικα. Όταν καλούμε μια υπορουτίνα με το όνομα της η εκτέλεση του προγράμματος μεταφέρεται στην υπορουτίνα και με το τέλος της επιστρέφει στο σημείο από το οποίο εκλήθη.

Στην υπορουτίνα μπορούν να περάσουν τιμές μέσω των παραμέτρων. πχ. **Sub name(foo, bar)**. Τα **foo** και **bar** είναι παράμετροι της υπορουτίνας **name**. Αν μια παράμετρος έχει αρχική τιμή τότε αυτή η παράμετρος είναι προαιρετική.

Πίνακες ως παράμετροι γράφονται με κάποιο προσδιοριστή με άδειες παρενθέσεις. Σημείωση, ότι οι πίνακες ως παράμετροι περνιούνται κατά αναφορά ByRef, η οποία ByRef λέξη δεν απαιτείται ούτε επιτρέπεται για πίνακες παραμέτρους. Όταν καλείται μια υπορουτίνα με παράμετρο έναν πίνακα τότε οι άδειες παρενθέσεις απαιτούνται.

Μια υπορουτίνα μπορεί επίσης να καθορίσει πως θα περνιούνται οι παράμετροι, είτε κατά αναφορά ByRef, είτε κατά τιμή ByVal.

Εάν η παράμετρος είναι κατα αναφορά "ByRef", το όνομα της παραμέτρου γίνεται μια αναφορά για την αρχική μεταβλητή που περνιέται στην υπορουτίνα. Οποιαδήποτε αλλαγή γίνει στην μεταβλητή αυτή θα απεικονιστεί εξωτερικά από την υπορουτίνα στην μεταβλητή.

Αν η παράμετρος περάσει κατά τιμή "ByVal", η τιμή οποιαδήποτε μεταβλητής που περνιέται αντιγράφεται σε νέα μεταβλητή και έτσι όποια αλλαγή γίνει σε αυτή δεν επηρεάζει την μεταβλητή που πέρασε σαν παράμετρος.

Η λέξη **Static** ορίζει ότι οι τιμές όλων των τοπικών μεταβλητών που ορίζονται στην υπορουτίνα θα πρέπει να διατηρούνται μεταξύ των κλήσεων της υπορουτίνας. Η υπορουτίνα **Sub** είναι ίδια με την συνάρτηση **Function** με την εξαίρεση ότι δεν επιτρέπει να επιστραφεί μια τιμή από αυτήν. Η λέξη **Return** εντός μιας υπορουτίνας σημαίνει την

άμεση έξοδο από την υπορουτίνα και δεν επιστρέφει κάποια τιμή.

## Παράδειγμα - sub.bas



```
1. Sub AddNumbers (a As Integer, b As Integer)
2.     Print a + b
3. End Sub
4.
5. AddNumbers(10, 6)
6. AddNumbers(10, 5)
7. AddNumbers(10, 16)
8. Sleep
9. End
```

## Έξοδος



```
16
15
26
```

## Ανάλυση

Μεταξύ των γραμμών 1 και 3 βρίσκεται η **υπορουτίνα** AddNumbers() η οποία έχει δύο **ορίσματα** ως ακεραίους, το a και το b. Σε αυτά τα **ορίσματα** θα εκχωρηθούν **τιμές** των **παραμέτρων** της υπορουτίνας.

Όταν δηλώνεται μια **υπορουτίνα** λέμε ότι έχει **ορίσματα** και όταν καλείται και διοχετεύονται **τιμές** λέμε ότι διοχετεύονται **παραμέτροι** στην υπορουτίνα.

Η γραμμή 2 είναι το **σώμα** της υπορουτίνας. Το σώμα της υπορουτίνας έχει μόνο μία πρόταση την Print a+b, η οποία τυπώνει στην οθόνη το άθροισμα δύο ακεραίων.

Στις γραμμές 5,6,7 καλείται η υπορουτίνα με παραμέτρους διάφορους ακεραίους. Η υπορουτίνα λαμβάνει τις παραμέτρους και τυπώνει το άθροισμα τους στην οθόνη.

Τέλος έχουμε την αναμονή του χρήστη και το τέλος του προγράμματος.



# Function

## Σύνταξη



```
[Public|Private] Function identifier [cdecl|pascal|
stdcall] [Overload] [Alias external_identifier]
[([parameter_list])] [ ByRef ] [As return_type]
[Static] [Export]
statements
```

...

```
{ {Return [return_value]}|{Function =
return_value}|{identifier = return_value} }
```

...

```
End Function
```

## Περιγραφή

Μια συνάρτηση ορίζει ένα μπλοκ κώδικα τον οποίο μπορείτε με μια πρόταση να καλέσετε και να τρέξει ο κώδικας του σώματος της συνάρτησης. Κάθε συνάρτηση επιστρέφει μια τιμή πίσω στον προγραμματιστή.

## Δικαιώματα πρόσβασης (Access Rights)

Οι λέξεις `Public` / `Private` δηλώνουν τα δικαιώματα πρόσβασης της συνάρτησης. Με άλλα λόγια δηλώνουν αν η συνάρτηση θα είναι ορατή από άλλα αρθρώματα (modules) ή θα έχει εμβέλεια μόνο εντός του αρθρώματος του οποίου έχει δηλωθεί. Ως αρθρώματα θεωρούνται τα αρχεία `*.bas` ενός έργου (Project).

Όταν μια συνάρτηση είναι `Public` τότε είναι ορατή και σε άλλα αρχεία `bas` του ίδιου έργου.

Όταν μια συνάρτηση είναι `Private` τότε είναι ορατή μόνο στο αρχείο `bas` το οποίο έχει δηλωθεί.

Όταν δεν αναγράφεται αν είναι `Public/Private` τότε εξ' ορισμού είναι `Public`.

## **Σύμβαση κλήσης (Calling Convention )**

Η σύμβαση κλήσης, ή αλλιώς η σειρά με την οποία περνούν τα παράμετροι σε μια συνάρτηση, ορίζεται με τις λέξεις `cdecl`, `pascal` και `stdcall`. Αν δεν ορίζεται κάποια εξ' ορισμού είναι η `stdcall`.

Όταν οριστεί η λέξη `cdecl` τότε οι παράμετροι της συνάρτησης περνούν από δεξιά προς τα αριστερά με αντίστροφη σειρά στην στοίβα (`stack`). Η κλήση αυτή είναι η εξ' ορισμού κλήση παραμέτρων για το Linux και το DOS.

Όταν οριστεί η λέξη `pascal` τότε οι παράμετροι της συνάρτησης περνούν από αριστερά προς τα δεξιά με την σειρά που ορίστηκαν, στην στοίβα (`stack`).

Όταν οριστεί η λέξη `stdcall` τότε οι παράμετροι της συνάρτησης περνούν από δεξιά προς τα αριστερά με αντίστροφη σειρά στην στοίβα (`stack`). Η κλήση αυτή είναι η εξ' ορισμού κλήση παραμέτρων για τα Windows.

## **Πέρασμα παραμέτρων (Passing Arguments)**

Οι συναρτήσεις μπορούν να έχουν μία ή περισσότερες παραμέτρους. Αυτές οι παράμετροι μπορούν να περάσουν κατά αναφορά ή κατά τιμή στην συνάρτηση, `ByRef`, `ByValue`. Οι πίνακες περνούνται πάντα κατά αναφορά. Για να περάσει ένας πίνακας ως παράμετρος γράφουμε το όνομα του και άδειες παρενθέσεις.

## **Υπερφόρτωση συναρτήσεων**

Μια συνάρτηση μπορεί να υπερφορτωθεί. Αυτό σημαίνει ότι μπορούμε να έχουμε πολλές συναρτήσεις με το ίδιο όνομα αλλά με διαφορετικά ορίσματα ή αλλιώς όπως λέγεται με διαφορετική υπογραφή.

## **Παράδειγμα**

```
Function AddNumbers(a As Integer, b As Integer) As Integer  
Function AddNumbers(a As Single, b As Single) As Single  
Function AddNumbers(a As Double, b As Double) As Double
```

## **Επιστρεφόμενη τιμή**

Με την λέξη `Return` η συνάρτηση επιστρέφει μια τιμή πίσω στον προγραμματιστή.

## Διατήρηση τοπικών μεταβλητών

Με την λέξη `Static` διατηρούνται οι τοπικές μεταβλητές της συνάρτησης μεταξύ των κλήσεων της.

Παρακάτω θα δούμε το παράδειγμα με την συνάρτηση `AddNumbers()` αυτή την φορά αντί της υπορουτίνας από το προηγούμενο παράδειγμα.

### Παράδειγμα - `func.bas`



```
1.  Function AddNumbers Overload (a As Integer, b As
    Integer) As Integer
2.      Return a + b
3.  End Function
4.
5.  Function AddNumbers Overload (a As Double, b As
    Double) As Double
6.      Return a + b
7.  End Function
8.
9.  Dim i As Integer
10. Dim d As Double
11.
12. i = AddNumbers(10, 5)
13. Print i
14. d = AddNumbers(10.3, 5.4)
15. Print d
16.
17. Sleep
18. End
19.
```

### Έξοδος



```
15
15.7
```

## **Ανάλυση**

Στις γραμμές 1-3 και 5-7 έχουμε την συνάρτηση `AddNumbers()` υπερφορτωμένη την μια φορά να προσθέτει 2 ακέραιους και την άλλη να προσθέτει δύο αριθμούς με κινητή υποδιαστολή.

Το σώμα της συνάρτησης επιστρέφει με την λέξη `Return` το άθροισμα.

Στις γραμμές 9,10 δηλώνονται δύο μεταβλητές. Μία ακέραιος και μία κινητής υποδιαστολής.

Στη γραμμή 12 και 14 εκχωρείται στις μεταβλητές η τιμή του αθροίσματος της συνάρτησης `AddNumbers()`.

Στις γραμμές 13,15 τυπώνονται οι τιμές των μεταβλητών στην οθόνη.

Τέλος έχουμε την αναμονή του χρήστη και το τέλος του προγράμματος.

## **Αναδρομή συναρτήσεων**

Όταν μια συνάρτηση καλεί τον εαυτό της αυτό ονομάζεται αναδρομή (recursion). Η αναδρομή είναι δύο ειδών. Η άμεση όπου μία συνάρτηση καλεί την ίδια και έμμεση όταν μια συνάρτηση καλεί μια άλλη και η δεύτερη καλεί την πρώτη.

Όταν μια συνάρτηση ενεργεί ξανά επί του αποτελέσματος της είναι ευκολότερο να γραφτεί κώδικας αναδρομής. Οι αναδρομές μπορεί να τελειώνουν και να επιστρέφουν μία τιμή αλλά μπορεί να μην τελειώνουν και ποτέ. Οι δεύτερες είναι συνήθως ένα προγραμματιστικό λάθος.

Όταν καλείτε μια συνάρτηση με αναδρομή ένα νέο αντίγραφο της συνάρτησης με τις τοπικές μεταβλητές της δημιουργείται και ο μεταγλωττιστής το διαχειρίζεται χωριστά χωρίς να επηρεάζει την πρώτη συνάρτηση.

Για να δούμε ένα παράδειγμα αναδρομής θα χρησιμοποιήσουμε την ακολουθία Fibonacci.

Η ακολουθία αυτή είναι η εξής: 1,1,2,3,5,8,13,21,34,55...

Τα πρώτα δύο νούμερα είναι 1 και 1. Τα επόμενα παράγονται από την πρόσθεση των δύο προηγούμενων.

Έτσι ο 3ος όρος παράγεται από το  $1+1=2$ .

Ο 4ος από το  $1+2=3$

Ο 5ος από το  $2+3=5$

...

Ο 8ος όρος από το  $8+13=21$

...

κτλ.

Με άλλα λόγια ο νιοστός όρος παράγεται από το άθροισμα  $n-1 + n-2$  εφόσον  $n > 2$ .

Το πρόγραμμα που θα γράψουμε θα βρίσκει τον νιοστό όρο της σειράς Fibonacci.

Για να τερματίσει το πρόγραμμα θα πρέπει  $n < 3$ .

Ο αλγόριθμος του fib.bas είναι:

- Δώστε την θέση του όρου προς εύρεση. π.χ. Θέλω να βρεθεί ο 10ος όρος.
- Κάλεσε την συνάρτηση Fibonacci() και διοχετεύστε την τιμή.
- Αν το όρισμα είναι  $x < 3$  επιστρέφει 1 διότι ο 1ος και 2ος όρος είναι το 1.
- Διαφορετικά η Fibonacci() καλεί αναδρομικά τον εαυτό της διοχετεύοντας τα ορίσματα  $x-2$  και  $x-1$  και τέλος αφού τα υπολογίσει τα προσθέτει, εφόσον κάθε όρος παράγεται από την πρόσθεση των δύο προηγούμενων.

Ας δούμε τον αλγόριθμο αυτόν:

Αν καλέσετε την Fibonacci(1) επιστρέφει 1.

Η Fibonacci(2) επιστρέφει 2.

Η Fibonacci(3) επιστρέφει την Fibonacci(1) + Fibonacci(2)

δλδ το  $1+1=2$

Η Fibonacci(4) επιστρέφει την Fibonacci(3)+Fibonacci(2) δλδ

το  $2+1=3$

...

έτσι κάθε φορά που καλείτε μια Fibonacci() με όρισμα μεγαλύτερο του 2 υπολογίζονται όλοι οι προηγούμενοι όροι με πρόσθεση κάθε δύο προηγούμενων όρων.

## Παράδειγμα - fib.bas



```
1. Function Fibonacci (a As Integer) As Integer
2.     Print "Calculate Fibonacci("; a; ")..."
3.     If a < 2 Then
4.         Print "Return "; a
5.         Return a
6.     Else
7.         Print "Call Fibonacci("; (a - 1); " and
8.         Fibonacci("; (a - 2); ")"
9.         Return (Fibonacci(a - 1) + Fibonacci(a -
10.    2))
11.     End If
12. End Function
13.
14. Dim i As Integer
15. Dim result As Integer
16. Input "Enter which Fibonacci to find:", i
17. result = Fibonacci(i)
18. Print "The "; i; " Fibonacci is: "; result
19.
20. Sleep
21. End
```

## Έξοδος



```
Enter which Fibonacci to find:3
Calculate Fibonacci( 3)...
Call Fibonacci( 2 and Fibonacci( 1)
Calculate Fibonacci( 2)...
Call Fibonacci( 1 and Fibonacci( 0)
Calculate Fibonacci( 1)...
Return 1
Calculate Fibonacci( 0)...
Return 0
Calculate Fibonacci( 1)...
Return 1
The 3 Fibonacci is: 2
```

## Ανάλυση

Στις γραμμές 1-10 υπάρχει η συνάρτηση Fibonacci όπως περιγράφηκε προηγουμένως.

Στην γραμμή 8 καλούμε αναδρομικά την συνάρτηση Fibonacci με παραμέτρους  $a-1$ ,  $a-2$  για να βρούμε τους προηγούμενους όρους της ακολουθίας.

Έτσι για παράδειγμα αν τρέξουμε το πρόγραμμα και θέλουμε να βρούμε τον 3ο όρο της ακολουθίας Fibonacci έχουμε την έξοδο που φαίνεται παραπάνω.

## Κεφάλαιο 8ο Δείκτες

### Η μνήμη του υπολογιστή

Η γλώσσα FreeBASIC εκτός από συναρτήσεις και μεταβλητές έχει και ένα άλλο χαρακτηριστικό, να μπορεί να έχει πρόσβαση στην μνήμη του υπολογιστή και να γράφει και να διαβάζει δεδομένα απευθείας από αυτήν. Για να γίνει αυτό η γλώσσα έχει ένα τύπο μεταβλητής που ονομάζεται δείκτης και αποθηκεύει σε αυτήν την διεύθυνση της μνήμης, που το πρόγραμμα αργότερα θα χρησιμοποιήσει για να διαβάσει ή να γράψει δεδομένα στην μνήμη.



Ένας δείκτης είναι μια μεταβλητή που της εκχωρείται μια διεύθυνση της ελεύθερης μνήμης. Γνωρίζοντας την διεύθυνση μνήμης μπορούμε να γράψουμε δεδομένα στην ελεύθερη μνήμη ή να διαβάσουμε δεδομένα από αυτήν.

Για να καταλάβουμε τους δείκτες πρώτα θα πρέπει να πούμε λίγα λόγια για την οργάνωση της μνήμης του υπολογιστή.

Η μνήμη του υπολογιστή, αν θέλουμε να την περιγράψουμε σχηματικά θα λέγαμε ότι, αποτελείται από μικρές περιοχές σαν κυψέλες (cells) που κάθε μία έχει μια διεύθυνση. Οι διευθύνσεις αναπαριστώνται σε δεκαεξαδική μορφή αλλά εμείς για απλούστευση θα χρησιμοποιήσουμε το δεκαδικό σύστημα.

<b>RAM</b>	<b>Διεύθυνση</b>	1567	<b>1568</b>	1569	1570	1571	1572
	<b>Δεδομένα</b>		h	e	l	l	o

Σε κάθε κυψέλη αποθηκεύονται δεδομένα.

Όταν ορίζουμε μεταβλητές ο τύπος δεδομένων της μεταβλητής καθορίζει πόσα bytes θα δεσμευθούν στην μνήμη για να αποθηκευτούν τα δεδομένα.

Η μνήμη που δεσμεύεται ξεκινά από μια διεύθυνση και καταλαμβάνει τόσες κυψέλες όσες ικανοποιούν τον αριθμό bytes του τύπου δεδομένων της μεταβλητής.

Ένας δείκτης όπως είπαμε είναι και αυτός μια μεταβλητή, η οποία όπως κρατά σαν τιμή μια διεύθυνση μνήμης. Έτσι ενώ η μεταβλητή **sString** κρατά την τιμή **“hello”** η μεταβλητή **pPointer** μπορεί να κρατά την διεύθυνση της **sString** δηλ την **“1568”**

Η μνήμη του υπολογιστή διαιρείται σε διάφορες περιοχές που άλλες είναι προσβάσιμες στον προγραμματιστή για το πρόγραμμα του και άλλες είναι δεσμευμένες από άλλα προγράμματα του συστήματος και δεν είναι προσβάσιμες για το πρόγραμμα το οποίο δημιουργούμε.

Πολύ γενικά θα πούμε ότι κάποιες περιοχές της μνήμης του υπολογιστή είναι:

Τοποθεσία καθολικών μεταβλητών
Σωρός (stack)
Τοποθεσία κώδικα προγράμματος
Καταχωρητές
Ελεύθερη μνήμη



Οι καθολικές μεταβλητές ενός προγράμματος βρίσκονται στην μνήμη στην περιοχή των καθολικών μεταβλητών. Οι τοπικές μεταβλητές ενός προγράμματος βρίσκονται στην περιοχή μνήμης που ονομάζεται σωρός. Οι εντολές του κώδικα του προγράμματος μας βρίσκονται στην τοποθεσία κώδικα προγράμματος. Οι καταχωρητές βρίσκονται στην δικιά τους περιοχή μνήμης. Τέλος υπάρχει κάποια περιοχή μνήμης που ένα πρόγραμμα μπορεί να χρησιμοποιήσει και ονομάζεται ελεύθερη μνήμη.

Σε αυτές τις περιοχές μνήμης καθώς ένα πρόγραμμα τρέχει εκτελούνται κάποιες αλλαγές.

Οι τοπικές μεταβλητές όταν μια συνάρτηση επιστρέφει απαλείφονται και ο σωρός καθαρίζει.

Οι καθολικές μεταβλητές από την άλλη έχουν απεριόριστη πρόσβαση από το πρόγραμμα αλλά αυτό δημιουργεί κώδικα δύσκολα συντηρήσιμο.

Αυτά τα προβλήματα η FreeBASIC τα λύνει τοποθετώντας δεδομένα στην ελεύθερη μνήμη.

Η ελεύθερη μνήμη έχει το πλεονέκτημα ότι ό,τι δεδομένο τοποθετείται σε αυτήν παραμένει διαθέσιμο και μετά το τέλος μιας συνάρτησης.

Αυτό αρχικά μπορεί να μοιάζει με τις καθολικές μεταβλητές αλλά δεν είναι.

Οι καθολικές μεταβλητές επιτρέπουν πρόσβαση σε όλες τις συναρτήσεις της κλάσης.

Τα δεδομένα στην ελεύθερη μνήμη επιτρέπουν πρόσβαση σε αυτά μόνο για τις συναρτήσεις που έχουν πρόσβαση στον δείκτη τους.

Αυτή η τεχνική είναι πιο αυστηρή σχετικά με την διαχείριση των δεδομένων και εξαλείφει το πρόβλημα της κατά λάθος αλλαγής των δεδομένων από τις συναρτήσεις με τρόπο που είναι δύσκολο να ανιχνευτεί σε ένα πρόγραμμα.

Αναλόγως την αρχιτεκτονική του υπολογιστή, αν είναι 32 bit ή 64 bit, μια θέση μνήμης μπορεί να έχει μήκος 32 ή 64 bit.

Έτσι ένας δείκτης μπορεί να είναι οποιαδήποτε τύπου δεδομένων, ακόμα και αυτά που καθορίζονται από τον χρήστη ως UDT και θα τα δούμε σε επόμενο κεφάλαιο.

# Δήλωση δεικτών σε μεταβλητές

Η δήλωση ενός δείκτη γίνεται με τον εξής τρόπο.

Σύνταξη:



```
DIM pVar As DataType {Pointer | Ptr}
```

Τα ονόματα των δεικτών έχουν συνήθως ένα p για πρώτο χαρακτήρα του ονόματος τους.

Ο τελεστής @ (Address of) ή η συνάρτηση VarPtr χρησιμοποιούνται για να πάρουμε την διεύθυνση του δείκτη.

Ο τελεστής \* (Value of) χρησιμοποιείται για να πάρουμε την τιμή στην οποία δείχνει ο δείκτης.

## Παράδειγμα - pointer.bas



```
1. ' Create the pointer.
2. Dim p As Integer Ptr
3. ' Create an integer value that we will point to
   using pointer "p"
4. Dim num As Integer = 98845
5. ' Point p towards the memory address that
   variable "num" occupies.
6. p = @num
   ' Print the value stored in memory pointed to by
   pointer "p"
7. Print "Pointer 'p' ="; *p
8. Print
9. ' Print the actual location in memory that
   pointer "p" points at.
10. Print "Pointer 'p' points to memory location:"
11. Print p
12.
13.
14. Sleep
15. End
```

## Έξοδος



```
Pointer 'p' = 98845
```

```
Pointer 'p' points to memory location:  
140730687114616
```

## Ανάλυση

Στην γραμμή 2 δηλώνεται ένας δείκτης σε ακέραιο.

Στην γραμμή 4 δηλώνεται και αρχικοποιείται μια μεταβλητή τύπου ακεραίου.

Στην γραμμή 6 ο δείκτης λαμβάνει την διεύθυνση της μεταβλητής num.

Στην γραμμή 8 τυπώνεται στην οθόνη η τιμή στην οποία δείχνει ο δείκτης.

Στην γραμμή 12 τυπώνεται στην οθόνη η διεύθυνση που έχει αποθηκευμένη ο δείκτης.

## Δήλωση δεικτών σε διαδικασίες

### Σύνταξη



```
Dim pointerToProcedure As Sub
```

```
Dim pointerToProcedure As Function
```

Για να πάρουμε την διεύθυνση ενός δείκτη σε διαδικασία χρησιμοποιούμε τον τελεστή @ (Address Of) ή τον τελεστή ProcPtr().

### Σύνταξη ProcPtr



```
result = ProcPtr ( procedure )
```

## Παράδειγμα - fpointer.bas



```
1. ' This example uses ProcPtr to demonstrate
   function pointers
2.
3. Declare Function Subtract( x As Integer, y As
   Integer) As Integer
4.
5. Declare Function Add( x As Integer, y As Integer)
   As Integer
6.
7. Dim myFunction As Function( x As Integer, y As
   Integer) As Integer
8.
9. ' myFunction will now be assigned to Add
10. myFunction = ProcPtr( Add )
11. Print myFunction(2, 3)
12.
13. ' myFunction will now be assigned to Subtract.
   Notice the different output.
14. myFunction = ProcPtr( Subtract )
15. Print myFunction(2, 3)
16.
17. Function Add( x As Integer, y As Integer) As
   Integer
18.     Return x + y
19. End Function
20.
21. Function Subtract( x As Integer, y As Integer) As
   Integer
22.     Return x - y
23. End Function
24.
25. Sleep
26. End
```

## Έξοδος



```
5
-1
```

# Η συνάρτηση Allocate

Η συνάρτηση Allocate δεσμεύει μνήμη και επιστρέφει την διεύθυνση της μνήμης που δέσμευσε.

## Σύνταξη



```
result = Allocate( count )
```

Όπου count το μέγεθος της μνήμης σε bytes που πρόκειται να δεσμευθεί.

Όπου result η διεύθυνση μνήμης που δεσμεύεται ή αν η κλήση είναι ανεπιτυχής επιστρέφεται το 0.

## Παράδειγμα - allocate.bas



```
1.  Const integerCount As Integer = 15
2.  ' Try allocating memory for a number of
    integers.
3.
4.  Dim buffer As Integer Ptr
5.  buffer = Allocate(integerCount * SizeOf(Integer))
6.  If (0 = buffer) Then
7.      Print "Error: unable to allocate memory,
    quitting."
8.      End - 1
9.  End If
10.
11. ' Prime and fill the memory with the fibonacci
    sequence.
12.
13. buffer[0] = 0
14. buffer[1] = 1
15.
16. For i As Integer = 2 To integerCount - 1
17.     buffer[i] = buffer[i - 1] + buffer[i - 2]
18. Next
19.
20. ' Display the sequence.
21.
22.
```

```

23. | For i As Integer = 0 To integerCount - 1
24. |     Print buffer[i] ;
25. | Next
26. |
27. | Deallocate(buffer)
28. |
29. | Sleep
30. | End 0

```

## Έξοδος



```
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

## Ανάλυση

Το πρόγραμμα αυτό χρησιμοποιεί την συνάρτηση ALLOCATE(...) για να δημιουργήσει ένα buffer μνήμης 15 ακεραίων.

Στην συνέχεια εκχωρεί στην μνήμη αυτή τους 15 πρώτους αριθμούς της ακολουθίας Fibonacci.

Τυπώνει στην οθόνη τις τιμές από τις 15 αυτές θέσεις μνήμης.

Τέλος απελευθερώνει την μνήμη με την συνάρτηση DEALLOCATE(...).

Όταν δεσμεύουμε μνήμη με την Allocate() ελέγχουμε αν ο δείκτης είναι διάφορος του 0. Αν ο δείκτης έχει τιμή 0 σημαίνει ότι δεν ήταν ικανό να δεσμευτεί η μνήμη.

Αυτό γίνεται στην γραμμή 6.

Τέλος με κάθε μία κλήση της Allocate() κάνουμε και μια κλήση της Deallocate() για να απελευθερώσουμε την μνήμη που δεσμεύσαμε. Αυτό γίνεται στην γραμμή 27.

## Αριθμητική Δεικτών

Με τον όρο αριθμητική δεικτών εννοούμε την διαχείριση των τιμών των διευθύνσεων των δεικτών με μαθηματικό τρόπο.

Είναι σχετικά απαιτητό μερικές φορές να προσπελάζουμε τις θέσεις μνήμης προς τα εμπρός ή προς τα πίσω και να ανακτούμε τις τιμές των διευθύνσεων.

Ο τύπος δεδομένων ενός δείκτη δείχνει και την απόσταση της μνήμης από την αρχική της θέση. Για παράδειγμα ένας Short καταλαμβάνει 2 bytes στην μνήμη ενώ ένας Single χρειάζεται 4 bytes.

## Πρόσθεση και αφαίρεση από τους δείκτες

Οι δείκτες μπορούν να προστεθούν και να αφαιρεθούν από τιμές όπως οποιοσδήποτε αριθμός.

Το αποτέλεσμα αυτών των πράξεων είναι ένας αριθμός που στην πραγματικότητα είναι μια διεύθυνση μνήμης.

## Παράδειγμα

```
Dim p As Integer Ptr = New Integer[2]
```

```
*p = 1
```

```
*(p + 1) = 2
```

Ο p είναι ένας δείκτης δύο θέσεων τύπου ακεραίου.

Η πρόταση \*p=1 θέτει στην διεύθυνση του δείκτη την τιμή 1.

Για να προχωρήσουμε μια θέση μπροστά σε μέγεθος ενός ακεραίου γράφουμε p+1.

Η έκφραση \*(p + 1) = 2 θέτει στην νέα θέση την τιμή 2.

## Αύξηση και μείωση δεικτών

Μερικές φορές θέλουμε να μετακινηθούμε στην μνήμη μπροστά η πίσω.

### Παράδειγμα

```
Dim array(5) As Short = { 32, 43, 66, 348, 112, 0 }
```

```
Dim p As Short Ptr = @array(0)
```

```
While (*p <> 0)
```

```
    If (*p = 66) Then Print "found 66"
```

```
    p += 1
```

```
Wend
```

Αρχικά δηλώνεται ένας πίνακας 6 θέσεων και αρχικοποιείται με τιμές.

Στην συνέχεια δηλώνεται ένας δείκτης και του εκχωρείται η διεύθυνση του πρώτου στοιχείου του πίνακα.

Έχουμε έναν βρόγχο While...Wend όπου ελέγχει αν η τιμή που έχει η διεύθυνση του δείκτη είναι διαφορετική του 0.

Στην συνέχεια αν η τιμή που έχει η διεύθυνση του δείκτη είναι 66 τυπώνεται ένα μήνυμα στην οθόνη.

Στην συνέχεια αυξάνεται ο δείκτης κατά μία θέση στον επόμενο Short από τον πίνακα, δηλαδή στο επόμενο στοιχείο του πίνακα.

Με τον τρόπο αυτό μετακινούμαστε σε διαδοχικές θέσεις μνήμης προς τα εμπρός με βήμα όσο ο τύπος δεδομένων του δείκτη.

## Προβλήματα με τους δείκτες

Η χρήση των δεικτών πολλές φορές οδηγεί σε κατάρρευση του προγράμματος αν δεν χρησιμοποιηθούν σωστά.

### Διαρροές μνήμης (Memory Leak)

Όταν η μνήμη δεσμεύεται αλλά δεν απελευθερώνεται οδηγεί μια εφαρμογή στο να καταναλώνει μνήμη μειώνοντας την διαθέσιμη μνήμη για τις άλλες εφαρμογές και κάνει το σύστημα να χρησιμοποιεί τον σκληρό δίσκο για να καταγράψει την εικονική μνήμη επιβραδύνοντας την



εφαρμογή ή κάνοντας την εφαρμογή να καταρρεύσει. Το σύστημα μπορεί να σταματήσει να εργάζεται καθώς αγγίζει τα όρια λειτουργίας του.

Για τον λόγο αυτό όποτε δεσμεύεται μνήμη με την `Allocate()` θα πρέπει να την απελευθερώνετε μετά την χρήση των δεδομένων με την `Deallocate()`.

## **Αλλοίωση δεδομένων στην μνήμη (Memory Corruption)**

Υπάρχουν περιπτώσεις που τα δεδομένα στην μνήμη αλλάζουν από λάθος και αλλοιώνονται από λάθος χρήση των δεικτών. Για παράδειγμα να έχει δεσμευθεί 4 bytes στην μνήμη και εμείς να γράψουμε δεδομένα μεγαλύτερα από 4 bytes.

## **Περιπλανώμενοι δείκτες (dangling pointers)**

Όταν απελευθερώσετε μια διεύθυνση μνήμης ή αλλιώς έναν δείκτη η τιμή που περιέχει η μεταβλητή δείκτης δεν μηδενίζεται. Ο δείκτης θα συνεχίσει να έχει την διεύθυνση της μνήμης εκχωρημένη ως τιμή του. Αν χρησιμοποιήσετε ένα δείκτη που έχει ελευθερωθεί τότε αυτός ο δείκτης ονομάζεται περιπλανώμενος δείκτης και αποτελεί ένα σφάλμα διότι χρησιμοποιείτε μια περιοχή μνήμης που είναι είτε ελεύθερη είτε την χρησιμοποιεί κάποιο άλλο πρόγραμμα.

Τα συνηθέστερα προβλήματα από την λάθος χρήση των δεικτών είναι:

- Η εκχώρηση νέας τιμής σε υπάρχων δείκτη δημιουργεί διαρροή μνήμης.
- Μη απελευθέρωση δείκτη οδηγεί σε διαρροή μνήμης.
- Εγγραφή δεδομένων με μεγαλύτερο μέγεθος από αυτό που έχει δεσμευθεί.
- Χρησιμοποίηση δείκτη ήδη έχει απελευθερωθεί.
- Απελευθέρωση δείκτη που ήδη έχει απελευθερωθεί.
- Απελευθέρωση μνήμης που δεν έχει δεσμευτεί.

- Απόπειρα εγγραφής σε δείκτη που έχει ελευθερωθεί.
- Εγγραφή σε περιοχή μνήμης που ποτέ δεν δεσμεύτηκε.

Τα παρακάτω παραδείγματα καλό είναι να μην τα τρέξετε ή αν τα τρέξετε αυτό να γίνει σε κάποιο εικονικό μηχάνημα (virtual machine) σε περίπτωση που καταρρεύσει το σύστημα σας.

## Παράδειγμα - memoryleak.bas



```

1. Dim pPointer1 As Integer Pointer
2. Dim pPointer2 As Integer Pointer
3.
4. pPointer1 = Allocate(4 * SizeOf(Integer))
5. pPointer2 = Allocate(4 * SizeOf(Integer))
6.
7.
8. Print "Address of pPointer1: "; pPointer1
9. Print "Address of pPointer2: "; pPointer2
10.
11. pPointer2 = pPointer1
12.
13. Print "Address of pPointer1: "; pPointer1
14. Print "Address of pPointer2: "; pPointer2
15.
16. Deallocate(pPointer1)
17. Deallocate(pPointer2)
18.
19. Sleep
20. End

```

## Έξοδος



```

Address of pPointer1: 23911280
Address of pPointer2: 23911328
Address of pPointer1: 23911280
Address of pPointer2: 23911280
free(): double free detected in tcache 2

```

```

Aborting due to runtime error 14 ("abnormal
termination" signal) in memoryleak.bas::()

```

## **Ανάλυση**

Στις γραμμές 1,2 έχουμε την δήλωση 2 δεικτών τύπου ακεραίου.

Στις γραμμές 4,5 δεσμεύεται μνήμη για τους δείκτες.

Στις γραμμές 7,8 τυπώνονται οι διευθύνσεις μνήμης των δεικτών.

Στην γραμμή 10 ο δεύτερος δείκτης λαμβάνει την διεύθυνση του πρώτου. Έτσι και οι 2 δείκτες δείχνουν στην ίδια διεύθυνση. Η πρόσβαση στην πρώτη διεύθυνση που έδειχνε ο πρώτος δείκτης είναι ανέφικτη πλέον και εδώ έχουμε ένα σφάλμα προγραμματισμού.

Στις γραμμές 12,13 τυπώνονται στην οθόνη οι διευθύνσεις των δεικτών. Σημείωση ότι και οι δύο δείκτες δείχνουν στην ίδια διεύθυνση.

Στην γραμμή 15 απελευθερώνεται η μνήμη από τον πρώτο δείκτη.

Στην γραμμή 16 επιχειρείται να απελευθερωθεί ξανά η ίδια διεύθυνση που έχει απελευθερωθεί προηγουμένως, έτσι δημιουργείται ένα λάθος και το πρόγραμμα διακόπτεται με μηνύματα λάθους από το λειτουργικό σύστημα.

```
free(): double free detected in tcache 2
```

```
Aborting due to runtime error 14 ("abnormal termination"  
signal) in memoryleak.bas::()
```

## **Μη απελευθέρωση δείκτη οδηγεί σε διαρροή μνήμης.**

Όταν δεσμεύσετε μια περιοχή μνήμης με την Allocate() σε μια συνάρτηση και η συνάρτηση τελειώσει αλλά εσείς δεν έχετε απελευθερώσει την μνήμη με την Deallocate() ο δείκτης που είναι μια τοπική μεταβλητή παύει να ισχύει και χάνεται.

Η μνήμη όμως δεν απελευθερώνεται αμέσως. Μετά το τέλος της συνάρτησης δεν υπάρχει τρόπος να έχετε πρόσβαση στην μνήμη και έτσι η μνήμη είναι σαν να διαρρέει από το σύστημα.

## Παράδειγμα - memoryleak2.bas



```
1. Sub UsePointer()  
2.     Dim pPointer1 As Integer Pointer  
3.     pPointer1 = Allocate(4)  
4.     Print "Address of pPointer1"; pPointer1  
5. End Sub  
6.  
7. UsePointer()  
8.  
9. Sleep  
10. End
```

## Έξοδος



```
Address of pPointer1: 17304432
```

## Ανάλυση

Στις γραμμές 1-5 έχουμε μια υπορουτίνα.

Στην γραμμή 2 δηλώνεται ένας δείκτης και στην γραμμή 3 τυπώνεται στην οθόνη η διεύθυνση που έχει ο δείκτης.

Στην γραμμή 7 καλείται η υπορουτίνα.

Σημείωση ότι στην υπορουτίνα ο δείκτης που δηλώθηκε δεν απελευθερώνεται και έτσι μετά το τέλος της υπορουτίνας δεν έχουμε πρόσβαση σε αυτόν για να τον απελευθερώσουμε.

Έτσι είναι σαν να διαρρέει μνήμη από το πρόγραμμα μας.

Εδώ βλέπουμε ένα από τα πιο συνηθισμένα λάθη προγραμματισμού αν και δεν υπάρχει λάθος μεταγλώττισης ή το πρόγραμμα δεν καταρρέει.

## Άλλα προγραμματιστικά λάθη με τους δείκτες.

- Εγγραφή δεδομένων με μεγαλύτερο μέγεθος μνήμης από αυτό που έχει δεσμευθεί.
- Χρησιμοποίηση δείκτη που ήδη έχει απελευθερωθεί.
- Απελευθέρωση δείκτη που ήδη έχει απελευθερωθεί.

- Απελευθέρωση μνήμης που δεν έχει δεσμευτεί από μετακίνηση προς τα εμπρός στην μνήμη
- Απόπειρα εγγραφής σε δείκτη που έχει ελευθερωθεί
- Εγγραφή σε περιοχή μνήμης που ποτέ δεν δεσμεύτηκε

## Κεφάλαιο 9ο - Τελεστές

Ένας τελεστής είναι ένα σύμβολο που λέει στον μεταγλωττιστή να κάνει κάποια ενέργεια.

Οι τελεστές ενεργούν πάνω σε τελεστέους οι οποίοι είναι οι εκφράσεις.

Υπάρχουν οι εξής ομάδες τελεστών:

Τελεστές εκχώρησης, οι οποίοι εκχωρούν μια έκφραση σε μια μεταβλητή.

Τελεστές αριθμητικοί, οι οποίοι τελούν μαθηματικές πράξεις μεταξύ εκφράσεων.

Τελεστές λογικοί, οι οποίοι τελούν λογικές μαθηματικές πράξεις μεταξύ εκφράσεων.

Τελεστές αλφαριθμητικών, οι οποίοι τελούν πράξεις σε strings.

Τελεστές σχεσιακοί οι οποίοι συγκρίνουν τους τελεστέους.

Τελεστές short circuit, οι οποίοι κάνουν μια εκτίμηση μεταξύ των τελεστέων

Τελεστές indexing, οι οποίοι επιστρέφουν αναφορές σε μεταβλητές ή αντικείμενα βασισμένα στην τιμή θέσης τους.

Τελεστές προεπεξεργαστή, οι οποίοι ελέγχουν την λειτουργία του προεπεξεργαστή.

Τελεστές δεικτών, οι οποίοι λειτουργούν πάνω σε δείκτες και διευθύνσεις.

Τελεστές τύπων ή κλάσεων, οι οποίοι παρέχουν πρόσβαση στα μέλη των τύπων δεδομένων του χρήστη.

Τελεστές μνήμης, οι οποίοι δεσμεύουν μνήμη και δημιουργούν αντικείμενα.

Τελεστές επαναληπτών, οι οποίοι χρησιμοποιούν επαναλήπτες για τα μπλοκ For...Next.

## Τελεστές εκχώρησης

Τελεστές οι οποίοι εκχωρούν τιμές σε τελεστέους.

Οι τελεστές εκχώρησης εκχωρούν στον πρώτο ή left-hand side, τελεστέο την τιμή του δεύτερου τελεστέου ή αλλιώς right-hand side, τελεστέο.

### Τελεστής [=] (Assignment)

Απευθείας εκχώρηση μιας τιμής σε μεταβλητή.

#### Χρήση:

lhs = rhs

ή

lhs => rhs

Εδώ lhs είναι η μεταβλητή και rhs η τιμή η οποία εκχωρείται.

#### Παράδειγμα:

```
i=420
```

### Τελεστής &= (Concatenate And Assign)

Εκχώρηση με συνένωση

#### Χρήση:

lhs &= rhs

#### Παράδειγμα:

```
Dim s As String = "Hello, "
```

```
s &= " world!"
```

```
Print s
```

θα παράγει την έξοδο:

```
Hello, world!
```

## Τελεστής += (Add And Assign)

Εκχώρηση με πρόσθεση

### Χρήση:

lhs += rhs

### Παράδειγμα:

```
Dim n As Double
```

```
n = 6
```

```
n += 1
```

```
Print n
```

```
Sleep
```

### Έξοδος:

7

## Τελεστής -= (Subtract And Assign)

Εκχώρηση με αφαίρεση

### Χρήση:

lhs -= rhs

### Παράδειγμα:

```
Dim n As Double
```

```
n = 6
```

```
n -= 2.2
```

```
Print n
```

```
Sleep
```

### Έξοδος:

3.8

## Τελεστής \*= (Multiply And Assign)

Εκχώρηση με πολλαπλασιασμό

### Χρήση:

lhs \*= rhs

### Παράδειγμα:

Dim n As Double

n = 6

n \*= 2

Print n

Sleep

### Έξοδος:

12

## Τελεστής /= (Divide And Assign)

Εκχώρηση με διαίρεση

### Χρήση:

lhs /= rhs

### Παράδειγμα:

Dim n As Double

n = 6

n /= 2.2

Print n

Sleep

### Έξοδος:

2.727272727272727



## Τελεστής \= (Integer Divide And Assign)

Εκχώρηση με ακέραιη διαίρεση

### Χρήση:

lhs \= rhs

### Παράδειγμα:

Dim n As Double

n = 6

n \= 2.2

Print n

Sleep

### Έξοδος:

3

## Τελεστής ^= (Exponentiate And Assign)

Εκχώρηση με ύψωση σε δύναμη

### Χρήση:

lhs ^= rhs

### Παράδειγμα:

Dim n As Double

n = 6

n ^= 2

Print n

Sleep

### Έξοδος:

36

## Τελεστής Let (Assignment)

Εκχώρηση τιμής UDT (User Defined Type) σε άλλο UDT

### Χρήση:

Στην διάλεκτο της QB έχουμε,

```
[ Let ] lhs = rhs  
ή  
[ Let ] lhs => rhs
```

Περισσότερα για τον τελεστή αυτό στο κεφάλαιο για τα UDT.

## Τελεστής Let() (Assignment)

Εκχώρηση τιμής πεδίου UDT σε μεταβλητή.

### Χρήση:

```
Let( variable1 [, variable2 [, ... ]] ) = UDT_var  
ή  
Let( variable1 [, variable2 [, ... ]] ) => UDT_var
```

Περισσότερα για τον τελεστή αυτό στο κεφάλαιο για τα UDT.

## Τελεστής Mod= (Modulus And Assign)

Εκχώρηση υπολοίπου διαίρεσης

### Χρήση:

```
lhs Mod= rhs
```

### Παράδειγμα:

```
Dim n As Integer  
n = 11  
n Mod= 3  
" The result is 2  
Print n  
Sleep
```

**Έξοδος:**

2

## **Τελεστής And= (Conjunction And Assign)**

Εκχώρηση και λογικό And

**Χρήση:**

lhs And= rhs

**Παράδειγμα:**

```
' Using the AND= operator on two numeric values
Dim As UByte numeric_value1, numeric_value2
numeric_value1 = 15 " 00001111
numeric_value2 = 30 " 00011110
numeric_value1 And= numeric_value2
" Result = 14 = 00001110
Print numeric_value1
Sleep
```

## **Τελεστής Eqv= (Equivalence And Assign)**

Εκχώρηση και λογική ισοδυναμία

**Χρήση:**

lhs Eqv= rhs

**Παράδειγμα:**

```
Dim As UByte a = &b00110011
Dim As UByte b = &b01010101
a Eqv= b
" Result a = &b10011001
Print Bin(a)
```

## **Τελεστής Imp= (Implication And Assign)**

Εκχώρηση και λογικό implication

### **Χρήση:**

lhs Imp= rhs

### **Παράδειγμα:**

```
Dim As UByte a = &b00110011
Dim As UByte b = &b01010101
a Imp= b
" Result  a = &b11011101
Print Bin(a)
```

## **Τελεστής Or= (Inclusive Disjunction And Assign)**

Εκχώρηση και λογικό Or

### **Χρήση:**

lhs Or= rhs

### **Παράδειγμα:**

```
Dim As UByte a = &b00110011
Dim As UByte b = &b01010101
a Or= b
" Result  a = &b01110111
Print Bin(a)
```

## **Τελεστής Xor= (Exclusive Disjunction And Assign)**

Εκχώρηση και λογικό Xor

### **Χρήση:**

lhs Xor= rhs

### **Παράδειγμα:**

```
Dim As UByte a = &b00110011
Dim As UByte b = &b01010101
a Xor= b
" Result   a = &b01100110
Print Bin(a)
```

### **Τελεστής Shl= (Shift Left And Assign)**

Εκχώρηση και λογική μετατόπιση προς τα αριστερά

#### **Χρήση:**

```
lhs shl= rhs
```

### **Παράδειγμα:**

```
Dim i As Integer
i = &b00000011  " = 3
i Shl= 3       " = i*2^3
" Result: 11000      24      24
Print Bin(i), i, 3*2^3
Sleep
```

### **Τελεστής Shr= (Shift Right And Assign)**

Εκχώρηση και λογική μετατόπιση προς τα δεξιά

#### **Χρήση:**

```
lhs shr= rhs
```

### **Παράδειγμα:**

```
Dim i As Integer
i = &b00011000  " = 24
i Shr= 3       " = i\2^3
" Result: 11      3      3
Print Bin(i), i, 24\2^3
Sleep
```

# Αριθμητικοί τελεστές

Τελεστές που χρησιμοποιούνται σε μαθηματικές πράξεις

## Τελεστής + (Add)

Πρόσθεση δύο τελεστών

### Χρήση:

result = lhs + rhs

### Παράδειγμα:

Dim n As Single

n = 4.75 + 5.25

Print n

### Έξοδος:

10

## Τελεστής - (Subtract)

Αφαίρεση δύο τελεστών

### Χρήση:

result = lhs - rhs

### Παράδειγμα:

Dim n As Single

n = 4 - 5

Print n

### Έξοδος:

-1

## Τελεστής \* (Multiply)

Επιστρέφει τον πολλαπλασιασμό δύο τελεστών

### Χρήση:

```
result = lhs * rhs
```

### Παράδειγμα:

```
Dim n As Double
```

```
n = 4 * 5
```

```
Print n
```

```
Sleep
```

### Έξοδος:

```
20
```

## Τελεστής / (Divide)

Επιστρέφει την διαίρεση δύο τελεστών

### Χρήση:

```
result = lhs / rhs
```

### Παράδειγμα:

```
Dim n As Double
```

```
n = 6 / 2.3
```

```
Print n
```

```
Sleep
```

### Έξοδος:

```
2.608695652173913
```

## Τελεστής \ (Integer Divide)

Επιστρέφει την ακέραιη διαίρεση μεταξύ δύο τελεστών

### Χρήση:

result = lhs \ rhs

### Παράδειγμα:

Dim n As Double

Print n \ 5

n = 7 \ 2.6 " => 7 \ 3 => 2.33333 => 2

Print n

n = 7 \ 2.4 " => 7 \ 2 => 3.5 => 3

Print n

Sleep

### Έξοδος:

0

2

3

## Τελεστής ^ (Exponentiate)

Ύψωση σε δύναμη

### Χρήση:

result = lhs ^ rhs

### Παράδειγμα:

Dim As Double n

Input "Please enter a positive number: ", n

Print

Print n;" squared is "; n ^ 2

Print "The fifth root of "; n;" is "; n ^ 0.2

Sleep

### Έξοδος:

Please enter a positive number: 3.4

3.4 squared is 11.56

The fifth root of 3.4 is 1.27730844458754



## Τελεστής Mod (Modulus)

Υπόλοιπο διαίρεσης

### Χρήση:

result = lhs Mod rhs

### Παράδειγμα:

Print 47 Mod 7

Print 5.6 Mod 2.1

Print 5.1 Mod 2.8

### Έξοδος:

5

0

2

## Τελεστής - (Negate)

Αρνητικοί αριθμοί

### Χρήση:

result = - rhs

### Παράδειγμα:

Dim n As LongInt

Print -5

n = 65432568459

n = - n

Print n

Sleep

### Έξοδος:

-5

-65432568459

## Τελεστής Shl (Shift Left)

Λογική μετατόπιση προς τα αριστερά

### Χρήση:

```
result = lhs Shl rhs
```

### Παράδειγμα:

```
'Double a number
```

```
For i As Integer = 0 To 10
```

```
    Print 5 Shl i, Bin(5 Shl i, 16)
```

```
Next i
```

### Έξοδος:

```
5          00000000000000101
10         0000000000001010
20         0000000000010100
40         0000000000101000
80         000000001010000
160        000000010100000
320        000000101000000
640        000001010000000
1280       000010100000000
2560       000010100000000
5120       000101000000000
```

## Τελεστής Shr (Shift Right)

Λογική μετατόπιση προς τα δεξιά

### Χρήση:

```
result = lhs Shr rhs
```

### Παράδειγμα:

```
'Halve a number
```

```
For i As Integer = 0 To 10
```

```
    Print 1000 Shr i, Bin(1000 Shr i, 16)
```

```
Next i
```

### Έξοδος:

```
1000    0000001111101000
500     0000000111110100
250     0000000011111010
125     0000000001111101
62      0000000000111110
31      0000000000011111
15      0000000000001111
7       0000000000000111
3       0000000000000011
1       0000000000000001
0       0000000000000000
```

## Σχισιακοί τελεστές

Τελεστές που συγκρίνουν την σχέση των τελεστών

Οι σχισιακοί τελεστές εκτελούν συγκρίσεις μεταξύ των τιμών δύο τελεστών. Κάθε τελεστής επιστρέφει έναν boolean που είναι true (-1)

Η σχέση κρατά το αληθές ή το ψέμα (0).

## Τελεστής = (Equal)

Ισότητα μεταξύ δύο τελεστών

### Χρήση:

result = lhs = rhs

### Παράδειγμα:

```
Dim i As Integer
```

```
i = 420
```

" assignment: assign to i the value of 420

```
If (i = 69) Then
```

```
    Print "serious error: i should equal 420"
```

" equation: compare the equality of the value of i and 69

```
End If
```

```
End -1
```

```
End If
```

## Τελεστής <> (Not Equal)

Ανισότητα

### Χρήση:

result = lhs <> rhs

### Παράδειγμα:

```
Dim As String a = "hello", b = "world"
```

```
Dim As Integer i = 10, j = i
```

```
If (a <> b) Then
```

```
    Print a & " does not equal " & b
```

```
End If
```

```
If (i <> j) Then
```

```
    Print "error: " & i & " does not equal " & j
```

```
End If
```

## **Τελεστής < (Less Than)**

Μικρότερο από

### **Χρήση:**

```
result = lhs < rhs
```

### **Παράδειγμα:**

```
Const size As Integer = 4  
Dim array(size - 1) As Integer = { 1, 2, 3, 4 }  
Dim index As Integer = 0  
While (index < size)  
    Print array(index)  
    index += 1  
Wend
```

## **Τελεστής <= (Less Than Or Equal)**

Μικρότερο ή ίσο από

### **Χρήση:**

```
result = lhs <= rhs
```

### **Παράδειγμα:**

```
If (69 <= 420) Then Print "(69 <= 420) is true."
```

## **Τελεστής >= (Greater Than Or Equal)**

Μεγαλύτερο ή ίσο από

### **Χρήση:**

```
result = lhs >= rhs
```

### **Παράδειγμα:**

```
If (420 >= 69) Then Print "(420 >= 69) is true."
```

## **Τελεστής > (Greater Than)**

Μεγαλύτερο από

### **Χρήση:**

result = lhs > rhs

### **Παράδειγμα:**

If (420 > 69) Then Print "(420 > 69) is true."

## **Τελεστής Is (Run-Time Type Information)**

Ελέγχει αν ένα αντικείμενο είναι συγκεκριμένου τύπου

### **Χρήση:**

result = expression Is typename

Περισσότερα για τον τελεστή αυτό στην συνέχεια στο κεφάλαιο για τα UDT

# Λογικοί τελεστές

Τελεστές που τελούν λογικές δυαδικές πράξεις μεταξύ των τελεστών

Οι λογικοί τελεστές τελούν λογικές μαθηματικές πράξεις σε αριθμούς.

Οι λογικοί τελεστές είναι AND, OR, NOT, XOR, IMP, EQV

Οι λογικοί τελεστές βασίζονται στους πίνακες αληθείας.

Ένας πίνακας αληθείας δείχνει το αποτέλεσμα τιμών με κάθε λογικό τελεστή.

Έτσι έχουμε για δύο τιμές A, B

**AND, επιστρέφει TRUE όταν και οι δύο τιμές είναι TRUE**

A	B	Αποτέλεσμα
FALSE	TRUE	FALSE
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	FALSE	FALSE

Παραδείγματα:

```
Print FALSE AND FALSE 'returns FALSE  
Print FALSE AND TRUE 'returns FALSE  
Print TRUE AND FALSE 'returns FALSE  
Print TRUE AND TRUE 'returns TRUE
```

## **NOT, επιστρέφει την αντίθετη τιμή**

<b>A</b>	<b>Αποτέλεσμα</b>
TRUE	FALSE
FALSE	TRUE

Παραδείγματα:

Print NOT TRUE      'return FALSE

Print NOT FALSE     'return TRUE

## **OR, επιστρέφει TRUE όταν μία από τις τιμές είναι TRUE**

<b>A</b>	<b>B</b>	<b>Αποτέλεσμα</b>
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

Παραδείγματα:

Print FALSE OR FALSE   'returns FALSE

Print FALSE OR TRUE    'returns TRUE

Print TRUE OR FALSE    'returns TRUE

Print TRUE OR TRUE     'returns TRUE



**XOR, επιστρέφει TRUE όταν μία από τις τιμές και μόνο μία είναι TRUE**

<b>A</b>	<b>B</b>	<b>Αποτέλεσμα</b>
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE

Παραδείγματα:

```
Print FALSE XOR FALSE 'returns FALSE
Print FALSE XOR TRUE  'returns TRUE
Print TRUE XOR FALSE  'returns TRUE
Print TRUE XOR TRUE   'returns FALSE
```

**Imp**

<b>A</b>	<b>B</b>	<b>Αποτέλεσμα</b>
FALSE	FALSE	TRUE
FALSE	TRUE	TRUE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

Παραδείγματα:

```
Print FALSE Imp FALSE 'returns TRUE
Print FALSE Imp TRUE  'returns TRUE
Print TRUE Imp FALSE  'returns FALSE
Print TRUE Imp TRUE   'returns TRUE
```

**Eqv**, επιστρέφει TRUE όταν οι δύο τιμές είναι ίδιες αλλιώς FALSE

<b>A</b>	<b>B</b>	<b>Αποτέλεσμα</b>
FALSE	FALSE	TRUE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

Παραδείγματα:

```
Print FALSE Imp FALSE 'returns TRUE
Print FALSE Imp TRUE 'returns FALSE
Print TRUE Imp FALSE 'returns FALSE
Print TRUE Imp TRUE 'returns TRUE
```

## Τελεστές Short Circuit

Οι τελεστές short circuit τελούν μια εκτίμηση στο αριστερό μέρος του τελεστή και ανάλογα το αποτέλεσμα μπορεί να τελέσουν εκτίμηση στο δεξί μέρος του τελεστή.

Οι εκτιμήσεις evaluations λαμβάνουν μέρος λογικά σε σύγκριση με το μηδέν.

## Τελεστής Andalso (Short Circuit Conjunction)

Επιστρέφει την σύζευξη δύο τελεστών

Η σύγκριση γίνεται βάσει του παρακάτω πίνακα αληθείας

Lhs Value	Rhs Value	Result
0	-	0
Nonzero	0	0
Nonzero	Nonzero	-1

**Χρήση:**

```
result = lhs AndAlso rhs
```

**Παράδειγμα:**

```
" Using the ANDALSO operator to guard against array access
" when the index is out of range
```

```
Dim As Integer isprime(1 To 10) = { _
    ' 1 2 3 4 5 6 7 8 9 10
    - 0, 1, 1, 0, 1, 0, 1, 0, 0, 0 _
}
```

```
Dim As Integer n
```

```
Input "Enter a number between 1 and 10: ", n
```

```
" isprime() array will only be accessed if n is in range
```

```
If (n >= 1 AndAlso n <= 10) AndAlso isprime(n) Then
```

```
    Print "n is prime"
```

```
Else
```

```
    Print "n is not prime, or out of range"
```

```
End If
```

**Τελεστής OrElse (Short Circuit Inclusive Disjunction)**

Επιστρέφει το αποτέλεσμα ενός short circuit inclusive των δύο τελεστών.

Η σύγκριση γίνεται βάσει του παρακάτω πίνακα αληθείας

Lhs Value	Rhs Value	Result
0	0	0
0	Nonzero	-1
Nonzero	-	-1

**Χρήση:**

```
result = lhs OrElse rhs
```

### **Παράδειγμα:**

```
' Using the ORELSE operator on two numeric values
Dim As Integer numeric_value1, numeric_value2
numeric_value1 = 15
numeric_value2 = 30
'Result = -1
Print numeric_value1 OrElse numeric_value2
Sleep
```

## **Τελεστές indexing, θέσης**

### **Τελεστής () (Array Index)**

Θέση πίνακα

#### **Χρήση:**

```
result = lhs ( rhs [, ...] )
ή
lhs ( rhs [, ...] ) = value
```

### **Παράδειγμα:**

```
Dim array(0 To 4) As Integer = { 0, 1, 2, 3, 4 }
For index As Integer = 0 To 4
    Print array(index);
Next
Print
```

#### **Έξοδος:**

0 1 2 3 4

## Τελεστής [] (String Index)

Θέση σε συμβολοσειρά

### Χρήση:

```
result = lhs [ rhs ]
```

ή

```
lhs [ rhs ] = value
```

### Παράδειγμα:

```
Dim a As String = "Hello, world!"
```

```
Dim i As Integer
```

```
For i = 0 To Len(a) - 1
```

```
    Print Chr(a[i]) & " ";
```

```
Next i
```

```
Print
```

```
Print
```

```
For i = 1 To 4
```

```
    a[i] = a[i] - 32 ' converting lowercase alphabetic  
characters to uppercase
```

```
Next i
```

```
For i = 7 To 11
```

```
    a[i] = a[i] - 32 ' converting lowercase alphabetic  
characters to uppercase
```

```
Next i
```

```
Print a
```

### Έξοδος:

```
Hello, world!
```

```
HELLO, WORLD!
```

## Τελεστής [] (Pointer Index)

Θέση σε δείκτη

### Χρήση:

```
result = lhs [ rhs ]
```

ή

```
lhs [ rhs ] = value
```

### Παράδειγμα:

```
" initialize a 5-element array
```

```
Dim array(4) As Integer = { 0, 1, 2, 3, 4 }
```

```
" point to the first element
```

```
Dim p As Integer Ptr = @array(0)
```

```
" use pointer indexing to output array elements
```

```
For index As Integer = 0 To 4
```

```
    Print p[index];
```

```
Next
```

```
Print
```

### Έξοδος:

```
0 1 2 3 4
```

## Τελεστές συμβολοσειρών

### Τελεστής + (String Concatenation)

Ένωση συμβολοσειρών

### Χρήση:

```
result = lhs + rhs
```

### Παράδειγμα:

```
Dim As String a = "Hello, ", b = "World!"
```

```
Dim As String c
```

```
c = a + b
```

```
Print c
```

### Έξοδος:

```
Hello, World!
```

## Τελεστής & (String Concatenation With Conversion)

Ένωση συμβολοσειρών με μετατροπή

### Χρήση:

```
result = lhs & rhs
```

### Παράδειγμα:

```
Dim As String A,C
Dim As Single B
A="The result is: "
B=124.3
C=A & B
Print C
Sleep
```

### Έξοδος:

```
The result is: 124.3
```

Τελεστής StrPtr (String Pointer)

Επιστρέφει την διεύθυνση μιας συμβολοσειράς

### Χρήση:

```
result = StrPtr ( lhs )
```

### Παράδειγμα:

```
" This example uses StrPtr to demonstrate using pointers
" with strings
```

```
Dim myString As String
Dim toMyStringDesc As Any Ptr
Dim toMyString As ZString Ptr
```

```
" Note that using standard VARPTR notation will return a
" pointer to the descriptor, not the string data itself
myString = "Improper method for Strings"
toMyStringDesc = @myString
```

```
Print myString
Print Hex( toMyStringDesc )
Print
```

```
" However, using StrPtr returns the proper pointer
myString = "Hello World Examples Are Silly"
toMyString = StrPtr(myString)
Print myString
Print *toMyString
Print
```

```
" And the pointer acts like pointers to other types
myString = "MyString has now changed"
Print myString
Print *toMyString
Print
```

## Τελεστές προεπεξεργαστή

### Τελεστής # (Stringize)

Επιστρέφει έναν τελεστή κειμένου που έχει μετατραπεί σε γραμματοσειρά String.

#### Χρήση:

```
#macro_argument
```

#### Παράδειγμα:

```
#define SEE(x) Print #x ;" = "; x
Dim variable As Integer, another_one As Integer
variable=1
another_one=2
SEE(variable)
SEE(another_one)
```

#### Έξοδος:

```
variable = 1
another_one = 2
```



## Τελεστής ## (Concatenation)

Ένωση δύο τελεστέων

### Χρήση:

```
text##text
```

### Παράδειγμα:

```
#define Concat(t,n) t##n
```

```
Print concat (12,34)
Dim Concat (hello,world) As Integer
Concat (hello,world)=99
Print helloworld
```

### Έξοδος:

```
1234
99
```

## Τελεστής ! (Escaped String Literal)

Χαρακτήρες διαφυγής

### Χρήση:

```
!"text"
```

### Παράδειγμα:

```
Print "Some escape sequence examples:"
Print !"1.\ttsingle quote (\\') : \'
Print !"2.\ttdouble quote (\\") : '\"
Print !"3.\tbackslash (\\) : \
Print !"4.\tascii char (\\65): \65"
```

### Έξοδος:

```
Some escape sequence examples:
1. single quote (\') : '
2. double quote (\") : "
3. backslash (\\) : \
4. ascii char (\\65): A
```

## **Τελεστής \$ (Non-Escaped String Literal)**

Μη χαρακτήρες διαφυγής

### **Χρήση:**

```
$"text"
```

### **Παράδειγμα:**

```
" Compile with -lang fblite or qb  
#lang "fblite"
```

```
Print "Default"  
Print "Backslash : \\  
Print !"Backslash !: \\  
Print $"Backslash $: \\  
Print  
Option Escape  
Print "Option Escape"  
Print "Backslash : \\  
Print !"Backslash !: \\  
Print $"Backslash $: \\  
Print
```

### **Έξοδος:**

```
Default  
Backslash : \\  
Backslash !: \  
Backslash $: \\  
Option Escape  
Backslash : \  
Backslash !: \  
Backslash $: \\  

```

# Τελεστές δεικτών

## Τελεστής Varptr (Variable Pointer)

Επιστρέφει την διεύθυνση μιας μεταβλητής

### Χρήση:

```
result = VarPtr ( lhs )
```

### Παράδειγμα:

```
Dim a As Integer, addr As Integer
```

```
a = 10
```

```
" place the address of a in addr
```

```
addr = CInt( VarPtr(a) )
```

```
" change all 4 bytes (size of INTEGER) of a
```

```
Poke Integer, addr, -1000
```

```
Print a
```

```
" place the address of a in addr (same as above)
```

```
addr = CInt( @a )
```

```
" print the least or most significant byte, depending on the
```

```
" CPU endianness
```

```
Print Peek( addr )
```

## Τελεστής Strptr (String Pointer)

Επιστρέφει την διεύθυνση μιας συμβολοσειράς

### Χρήση:

```
result = StrPtr ( lhs )
```

### Παράδειγμα:

```
" This example uses Strptr to demonstrate using pointers
```

```
" with strings
```

```
Dim myString As String
```

```
Dim toMyStringDesc As Any Ptr
```

```
Dim toMyString As ZString Ptr
```

```
" Note that using standard VARPTR notation will return a  
" pointer to the descriptor, not the string data itself  
myString = "Improper method for Strings"  
toMyStringDesc = @myString
```

```
Print myString  
Print Hex( toMyStringDesc )  
Print
```

```
" However, using StrPtr returns the proper pointer  
myString = "Hello World Examples Are Silly"  
toMyString = StrPtr(myString)
```

```
Print myString  
Print *toMyString  
Print
```

```
" And the pointer acts like pointers to other types  
myString = "MyString has now changed"
```

```
Print myString  
Print *toMyString  
Print
```

Τελεστής ProcPtr (Procedure Pointer)  
Επιστρέφει την διεύθυνση μιας ρουτίνας

Χρήση:  
result = ProcPtr ( lhs )

Παράδειγμα:

```
' This example uses ProcPtr to demonstrate function pointers  
Declare Function Subtract( x As Integer, y As Integer) As  
Integer
```

```
Declare Function Add( x As Integer, y As Integer) As Integer
```

```
Dim myFunction As Function( x As Integer, y As Integer) As  
Integer
```

```
' myFunction will now be assigned to Add
myFunction = ProcPtr( Add )
Print myFunction(2, 3)
```

```
' myFunction will now be assigned to Subtract. Notice the
' different output.
myFunction = ProcPtr( Subtract )
Print myFunction(2, 3)
```

```
Function Add( x As Integer, y As Integer) As Integer
    Return x + y
End Function
```

```
Function Subtract( x As Integer, y As Integer) As Integer
    Return x - y
End Function
```

## **Τελεστής @ (Address Of)**

Επιστρέφει την διεύθυνση μιας μεταβλητής, ρουτίνας, ή αντικειμένου

Χρήση:  
result = @ rhs

Παράδειγμα:

'This program demonstrates the use of the @ operator.

```
Dim a As Integer
Dim b As Integer
Dim addr As Integer Ptr
```

```
a = 5
b = 10
```

```
' Here, we print the value of the variables, then where in
' memory they are stored.
```

```
Print "The value in A is ";a;" but the pointer to a is ";@a
Print "The value in B is ";b;" but the pointer to b is ";@b
```

' Now, we will take the integer ptr above, and use @ to place  
' a value into it.  
'Note that the \* will check the value in the ptr, just as @  
' checked the ptr for a normal variable.

```
addr = @a  
Print "The pointer addr is now pointing at the memory  
address to a, value: ";*addr
```

```
addr = @b  
Print "The pointer addr is now pointing at the memory  
address to b, value: ";*addr
```

## **Τελεστής \* (Value Of)**

Επιστρέφει την τιμή αναφοράς από έναν δείκτη

Χρήση:  
result = \* rhs  
ή  
\* rhs = value

Παράδειγμα:

' This program demonstrates the use of \* to utilize the value a  
' pointer points to.

```
Dim a As Integer  
Dim pa As Integer Ptr
```

```
pa=@a 'Here point our integer ptr at 'a'.  
a=9   'Here we give 'a' a value of 9.  
Print "The value of 'a' is"; *pa
```

```
*pa = 1 'Here we use our pointer to change the value of 'a'  
Print "The new value of 'a' is"; a  
'Here we display the new value of 'a'.
```

## **Τελεστές επαναληπτών**

### **Τελεστές For, Next, Step**

Τους τελεστές αυτούς θα τους δούμε στο κεφάλαιο για τα UDT

### **Τελεστές UDT**

#### **Τελεστές . (Member Access), -> (Pointer To Member Access), Is (Run-Time Type Information)**

Τους τελεστές αυτούς θα τους δούμε στο κεφάλαιο για τα UDT

### **Τελεστές Μνήμης**

#### **Τελεστές New Expression, New Overload, Placement New, Delete Statement, Delete Overload**

Τους τελεστές αυτούς θα τους δούμε στο κεφάλαιο για τα UDT

# Κεφάλαιο 10ο - Τύποι Χρηστών - UDT (User Defined Types)

Οι Τύποι δηλωμένοι από τον χρήστη ή αλλιώς User-Defined Types (UDT) είναι ειδικού τύπου μεταβλητές οι οποίες δημιουργούνται από τον προγραμματιστή.

Ένα UDT είναι ένα container το οποίο περιέχει άλλες μεταβλητές όπως οι πίνακες με την διαφορά ότι οι μεταβλητές του UDT μπορεί να είναι διαφορετικών τύπων δεδομένων, ενώ οι θέσεις ενός πίνακα είναι του ίδιου τύπου δεδομένων. Τα UDTs μπορούν ακόμη να περιέχουν και ρουτίνες.

## Δήλωση UDT

### Σύνταξη



```
Type typename
    fieldname1 As DataType
    fieldname2 As DataType
As DataType fieldname3, fieldname4
...
End Type
ή
Type typename [Alias "alternatename"] [Extends
base_typename] [Field = alignment]
[Private:|Public:|Protected:]
Declare Sub|Function|Constructor|Destructor|
Property|Operator ...
Static variablename As DataType
ReDim arrayname(array dimensions) As DataType
fieldname As DataType [= initializer]
fieldname(array dimensions) As DataType [=
initializer]
fieldname(Any [, Any...]) As DataType
fieldname : bits As DataType [= initializer]
As DataType fieldname [= initializer], ...
As DataType fieldname(array dimensions) [=
initializer], ...
```



```
As DataType fieldname(Any [, Any...])
As DataType fieldname : bits [= initializer], ...
```

```
Union
    fieldname As DataType
```

```
Type
    fieldname As DataType
```

```
...
End Type
```

```
...
End Union
```

```
...
End Type
```

Η πιο απλή μορφή UDT είναι η εξής:

```
Type typename
    fieldname1 As DataType
    fieldname2 As DataType
    As DataType fieldname3, fieldname4
```

```
...
End Type
```

### Παράδειγμα - type1.bas



```
1. Type car
2.     speed As Integer
3. End Type
4.
5. Dim mycar As car
6. mycar.speed = 10
7. Print mycar.speed
8.
9. Sleep
10. End
11.
```

## Έξοδος:



10

### Ανάλυση:

Στις γραμμές 1-3 δηλώνεται ο τύπος δεδομένων του χρήστη, UDT, με το όνομα `car`.

Στην γραμμή 2 έχουμε ένα μέλος του UDT μια μεταβλητή ακεραίου τύπου με το όνομα `speed`.

Στην γραμμή 5 δηλώνουμε ένα UDT τύπου `car`, ως μια μεταβλητή με το όνομα `mycar`.

Στην γραμμή 6, εκχωρούμε την τιμή 10 στην μεταβλητή `mycar.speed`. Για να προσπελάσουμε την μεταβλητή `speed` του UDT `mycar` γράφουμε το όνομα "`mycar`" του UDT ακολουθούμενο από μια τελεία "." και στην συνέχεια το όνομα του μέλους που θέλουμε να έχουμε πρόσβαση.

Στην γραμμή 7 τυπώνουμε την μεταβλητή `speed` του UDT `mycar`.

Τέλος έχουμε την αναμονή του χρήστη και το τέλος του προγράμματος.

Η τελεία "." που χρησιμοποιήθηκε για να έχουμε πρόσβαση σε ένα μέλος του UDT ονομάζεται τελεστής πρόσβασης μέλους (Member Access).

## Μέλη UDT

Οι διαφορετικές μεταβλητές ή και ρουτίνες που αποθηκεύονται μέσα σε ένα UDT ονομάζονται "μέλη". Τα μέλη μπορεί να είναι μεταβλητές κάθε τύπου δεδομένων συμπεριλαμβανομένου αριθμητικών μεταβλητών, συμβολοσειρών, δεικτών, πινάκων, απαριθμήσεων κτλ. Οι μεταβλητές των UDT δημιουργούνται όπως κάθε άλλη μεταβλητή παραλείποντας την λέξη `Dim` η οποία είναι προαιρετική.

Τα μέλη ενός UDT προσπελάζονται με τον τελεστή πρόσβασης που είναι η τελεία, "." !

## Παράδειγμα - color.bas



```
1. Type clr
2.     red As UByte
3.     green As UByte
4.     blue As UByte
5. End Type
6.
7. Dim mycolor As clr
8. mycolor.red = 255
9. mycolor.green = 128
10. mycolor.blue = 64
11.
12. Print mycolor.red
13. Print mycolor.green
14. Print mycolor.blue
15.
16.
17. Sleep
18. End
```

### Έξοδος:



```
255
128
64
```

### Ανάλυση:

Στις γραμμές 1-5 δηλώνουμε ένα UDT με το όνομα `clr`. Θέλουμε να δημιουργήσουμε ένα τύπο για να κρατάμε το χρώμα της οθόνης.

Έτσι στις γραμμές 2,3,4 έχουμε τις μεταβλητές τύπου `UByte` για τα τρία χρώματα της οθόνης `red`, `green`, `blue`.

Στην γραμμή 7 δηλώνουμε μια μεταβλητή UDT τύπου `clr` με το όνομα `mycolor`.

Στις γραμμές 8,9,10 δίνουμε τιμές στα μέλη `red`, `green`, `blue`.

Στις γραμμές 12,13,14 τυπώνουμε στην οθόνη τις τιμές των μελών `red`, `green`, `blue` του UDT `mycolor`.

Τέλος έχουμε την αναμονή του χρήστη και το τέλος του προγράμματος.

## Δείκτες σε UDT

Όπως έχουμε δείκτες σε μεταβλητές, έτσι μπορούμε να έχουμε δείκτες που να δείχνουν σε UDT.

Δηλώνονται σαν κανονικοί δείκτες αλλά υπάρχει ειδικός τρόπος χρήσης τους.

Για να προσπελάσουμε ένα μέλος ενός UDT που δείχνεται από έναν δείκτη χρησιμοποιούμε το τελεστή -> (Pointer To Member Access).

### Παράδειγμα - udtpointer1.bas



```
1. Type rect
2.     x As Integer
3.     y As Integer
4. End Type
5.
6. Dim r As rect
7. Dim rp As rect Pointer = @r
8.
9.
10. rp->x = 4
11. rp->y = 2
12.
13. Print "x = " & rp->x & ", y = " & rp->y
14. Print "x = " & (*rp).x & ", y = " & (*rp).y
15. Sleep
    End
```

### Έξοδος:



```
x = 4, y = 2
x = 4, y = 2
```

### Ανάλυση:

Στις γραμμές 1-4 δηλώνουμε ένα UDT με μέλη τις μεταβλητές x,y ως ακεραίους.

Στην γραμμή 6 δηλώνουμε ένα UDT τύπου rect.

Στην γραμμή 7 δημιουργούμε έναν δείκτη για το UDT.

Στις γραμμές 9,10 εκχωρούμε στα μέλη τιμές κάνοντας χρήση του τελεστή ->.

Στις γραμμές 12,13 τυπώνουμε στην οθόνη τις τιμές των μελών του UDT.

Στην γραμμή 12 χρησιμοποιούμε τον τελεστή ->, ενώ στην γραμμή 13 χρησιμοποιούμε την πρόταση (\*gr).x που είναι ισοδύναμη με την πρόταση gr->x

## **Constructors και Destructors, συναρτήσεις Δόμησης και Αποδόμησης**

Οι συναρτήσεις Δόμησης (Constructors) και Αποδόμησης (Destructors) είναι υπεύθυνες για την δημιουργία και την καταστροφή αντικειμένων.

Όταν δημιουργούμε μια μεταβλητή από μια δηλωμένη UDT τότε λέμε ότι δημιουργούμε ένα αντικείμενο (object).

Κάθε φορά που δηλώνεται ένα αντικείμενο καλείται εξ' ορισμού μια συνάρτηση Δόμησης του αντικειμένου.

Και κάθε φορά που περνιέται ένα αντικείμενο ως παράμετρος και τελειώνει η συνάρτηση στην οποία πέρασε, τότε το αντικείμενο βγαίνει εκτός εμβέλειας της συνάρτησης και καταστρέφεται. Όταν ένα αντικείμενο καταστρέφεται συνήθως απελευθερώνονται όλοι οι πόροι οι οποίοι έχει δεσμεύσει.

Ένα αντικείμενο μπορεί να δομηθεί με την λέξη Dim ή New και να καταστραφεί από την λέξη Delete ή αυτόματα όταν βγει εκτός εμβέλειας.

## **Δήλωση συναρτήσεων Δόμησης και Αποδόμησης**

Αν δεν έχουν δηλωθεί από τον προγραμματιστή συναρτήσεις Δόμησης και Αποδόμησης ο μεταγλωττιστής αυτόματα δημιουργεί αυτές για εσάς.

Ωστόσο η γλώσσα FreeBASIC μας δίνει το πλεονέκτημα να γράψουμε τις δικές μας συναρτήσεις Δόμησης και Αποδόμησης.

Οι συναρτήσεις αυτές δηλώνονται όπως οποιαδήποτε ρουτίνα μέλος του UDT αλλά αντί για την λέξη Sub/Function χρησιμοποιούμε τις λέξεις Constructor/Destructor.

## Σύνταξη:



```
Type typename  
  Declare Constructor ( )  
  Declare Constructor ( [ ByRef | ByVal ] parameter  
  As datatype [ = default ] [, ... ] )  
End Type
```

```
Constructor typename ( [ parameters ] ) [ Export ]  
  statements  
End Constructor
```

## Παράδειγμα - constructor1.bas



```
1. Type car  
2.     speed As Integer  
3.     Declare Constructor()  
4.     Declare Destructor()  
5. End Type  
6.  
7. Constructor car()  
8.     Print "I am in constructor"  
9.     Print "I initialize speed member"  
10.    This.speed = 10  
11.    Print "My car's speed is " & This.speed  
12. End Constructor  
13.  
14. Destructor car()  
15.     Print "I am in destructor"  
16. End Destructor  
17.  
18. Scope  
19.     Print "I am in scope..."  
20.     Dim mycar As car  
21. End Scope  
22.  
23. Print "I am out of scope..."  
24.  
25.  
26. Sleep  
27. End
```

## Έξοδος:



```
I am in scope...  
I am in constructor  
I initialize speed member  
My car's speed is 10  
I am in destructor  
I am out of scope...
```

## Ανάλυση:

Στις γραμμές 1-5 δηλώνουμε ένα UDT με το όνομα car.  
Στην γραμμή 2 έχουμε το μέλος speed ως ακέραιο.  
Στις γραμμές 3,4 δηλώνουμε τις συναρτήσεις Δόμησης και Αποδόμησης.

Στις γραμμές 7-12 έχουμε το σώμα της συνάρτησης Δόμησης.  
Στις γραμμές 14-16 έχουμε το σώμα της συνάρτησης Αποδόμησης.

Στην συνάρτηση Δόμησης τυπώνουμε μερικά μηνύματα στην οθόνη και αρχικοποιούμε για το τρέχον αντικείμενο την μεταβλητή μέλος speed. Για να έχουμε πρόσβαση στην μεταβλητή μέλος speed του εκάστοτε αντικειμένου χρησιμοποιούμε την λέξη This. Η λέξη This επιστρέφει την αναφορά στο τρέχον αντικείμενο και έχουμε πρόσβαση στα μέλη του με τον τελεστή πρόσβασης τελεία. Έτσι για να εκχωρήσουμε ή να προσπελάσουμε το μέλος speed του αντικειμένου γράφουμε This.speed  
Στην γραμμή 10 εκχωρούμε μια τιμή ακεραίου στο μέλος speed.

Στην γραμμή 11 τυπώνουμε ένα μήνυμα στην οθόνη προσπελάζοντας το μέλος speed.

Στην γραμμή 15 βρισκόμαστε στην συνάρτηση Αποδόμησης και τυπώνουμε ένα μήνυμα στην οθόνη.

Στις γραμμές 18-21 δημιουργούμε ένα μπλοκ εμβέλειας με τις λέξεις Scope...End Scope.

Στην γραμμή 19 τυπώνουμε το μήνυμα ότι βρισκόμαστε εντός μπλοκ εμβέλειας.

Στην γραμμή 20 δηλώνουμε ένα αντικείμενο τύπου car

Με την δήλωση αυτή δημιουργείται το αντικείμενο mycar και καλείται η συνάρτηση Δόμησης. Έτσι έχουμε τα μηνύματα στην οθόνη:

```
I am in constructor  
I initialize speed member  
My car's speed is 10
```

Στην γραμμή 21 τελειώνει το μπλοκ εμβέλειας και το αντικείμενο mycar καταστρέφεται. Έτσι καλείται η συνάρτηση Αποδόμησης και εμφανίζεται το μήνυμα στην οθόνη:

```
I am in destructor
```

Στην γραμμή 23 βρισκόμαστε πλέον εκτός μπλοκ εμβέλειας και τυπώνουμε το μήνυμα:

```
I am out of scope...
```

Τέλος έχουμε την αναμονή του χρήστη και το τέλος του προγράμματος.



# Μέλη Ρουτίνες

## Δήλωση και ορισμός

Τα μέλη ρουτίνες δηλώνονται όπως οι υπορουτίνες και συναρτήσεις του αρθρώματος με την εξαίρεση ότι δηλώνονται εντός ενός `Type...End Type` και ορίζονται εκτός ενός `Type....End Type`.

## Παράδειγμα - `subfunc.bas`



```
1. Type foo
2.     Declare Sub f (As Integer)
3.     Declare Function g As Integer
4.         i As Integer
5. End Type
6.
7. Sub foo.f (n As Integer)
8.     Print n
9. End Sub
10.
11. Function foo.g () As Integer
12.     Return 420
13. End Function
14.
15. Dim bar As foo
16. bar.f (5)
17. bar.i = bar.g()
18. Print bar.i
19.
20.
21. Sleep
22. End
```

## Έξοδος:



```
5
420
```

## **Ανάλυση:**

Στις γραμμές 1-5 έχουμε την δήλωση ενός UDT με το όνομα foo.

Στις γραμμές 2,3 έχουμε τις δηλώσεις δύο ρουτινών του UDT. Υπόψιν οι δηλώσεις των ρουτινών του UDT γίνονται εντός του σώματος του UDT και οι ορισμοί των ρουτινών εκτός, όπως φαίνεται στις γραμμές 7,11.

Στην γραμμή 15 δηλώνουμε ένα αντικείμενο τύπου foo με το όνομα bar.

Στην γραμμή 16 καλούμε την υπορουτίνα του αντικειμένου bar.f με όρισμα την τιμή 5.

Έτσι τυπώνεται στην οθόνη η τιμή 5.

Στην γραμμή 17 εκχωρούμε στο μέλος i του αντικειμένου bar την τιμή που επιστρέφει η συνάρτηση μέλος bar.g(). Δηλαδή εκχωρείται η τιμή 420 στην μεταβλητή μέλος i.

Στην γραμμή 18 τυπώνουμε στην οθόνη την τιμή που έχει η μεταβλητή μέλος i του αντικειμένου bar.

## **Η λέξη κλειδί This**

Όταν δηλώνουμε ένα UDT είναι σαν να δημιουργούμε ένα σχέδιο για τα αντικείμενα που θα δημιουργηθούν από αυτό το UDT.

Στην συνέχεια δηλώνουμε ένα αντικείμενο τύπου UDT.

Είναι το ίδιο σαν να έχουμε ένα σχέδιο σπιτιού (το UDT) και να χτίζουμε ένα σπίτι βασισμένοι στο σχέδιο αυτό.

Το σχέδιο είναι το δηλωμένο UDT και το σπίτι είναι το δηλωμένο αντικείμενο τύπου UDT.

Έτσι από ένα σχέδιο μπορούμε να χτίσουμε πολλά σπίτια.

Το ίδιο συμβαίνει και με τα UDT. Μπορούμε να έχουμε πολλά αντικείμενα του ίδιου τύπου UDT.

Όταν θέλουμε να προσπελάσουμε μέλη ενός αντικειμένου τύπου UDT χρησιμοποιούμε την λέξη κλειδί This.

Η λέξη This αναφέρεται στο αντικείμενο που έχει δηλωθεί τύπου UDT.

## Παράδειγμα - this.bas



```
1. Type car
2.     speed As Integer
3.     Declare Sub getspeed ()
4. End Type
5.
6. Sub car.getspeed ()
7.     Print This.speed
8. End Sub
9.
10. Dim bmw As car
11. Dim porche As car
12.
13.
14. bmw.speed = 100
15. bmw.getspeed
16.
17. porche.speed = 200
18. porche.getspeed
19.
20. Sleep
21. End
```

### Έξοδος:



```
100
200
```

### Ανάλυση:

Στις γραμμές 1-4 έχουμε την δήλωση ενός UDT με το όνομα car.

Στην γραμμή 2 έχουμε το μέλος μεταβλητή ακεραίου speed.

Στην γραμμή 3 δηλώνεται το μέλος υπορουτίνα getspeed.

Στις γραμμές 6-8 ορίζουμε την υπορουτίνα μέλος getspeed().

Στην γραμμή 7 τυπώνουμε στην οθόνη την μεταβλητή μέλος του εκάστοτε αντικειμένου που θα δημιουργηθεί από το τύπο UDT.

Στις γραμμές 10,11 δηλώνουμε δύο αντικείμενα τύπου car με τα ονόματα bmw και porche.

Στην γραμμή 13 εκχωρούμε την τιμή 100 στο μέλος μεταβλητή speed του αντικειμένου bmw.

Στην γραμμή 14 καλούμε την υπορουτίνα μέλος getspeed του αντικειμένου bmw. Η υπορουτίνα bmw.getspeed τυπώνει στην οθόνη την τιμή της μεταβλητής speed για το αντικείμενο bmw. Αυτό γίνεται μέσω της λέξης This που επιστρέφει το αντικείμενο bmw.

Το ίδιο ακριβώς γίνεται και στις γραμμές 16,17 για το αντικείμενο porche.

Κάθε φορά που καλείται η υπορουτίνα μέλος getspeed η λέξη This επιστρέφει το αντικείμενο το οποίο κάλεσε την υπορουτίνα μέλος. Έτσι στη γραμμή 7 η πρόταση This.speed αναφέρεται κάθε φορά σε διαφορετικό αντικείμενο, την μια φορά αναφέρεται στο αντικείμενο bmw και την άλλη στο porche.

## Υπερφόρτωση μελών ρουτινών

Η FreeBASIC υποστηρίζει την υπερφόρτωση των μελών ρουτινών ενός UDT.

Με το όρο υπερφόρτωση εννοούμε την πολλαπλή δήλωση ρουτινών με το ίδιο όνομα αλλά με διαφορετικούς παραμέτρους.

### Παράδειγμα - overloading.bas



```
1. Type car
2.     Declare Sub printspeed (i As Integer)
3.     Declare Sub printspeed (d As Double)
4.     colour As String
5. End Type
6.
7. Sub car.printspeed (i As Integer)
8.     Print i
9. End Sub
10.
11. Sub car.printspeed (d As Double)
12.     Print d
13. End Sub
14.
15.
16. Dim bmw As car
17. bmw.printspeed(100)
18. bmw.printspeed(100.5)
19.
20. Sleep
21. End
```

### Έξοδος:



```
100
100.5
```

## Ανάλυση:

Στις γραμμές 1-5 δηλώνεται ένα UDT με το όνομα car.

Στις γραμμές 2,3 δηλώνεται η υπορουτίνα μέλος printspeed ως υπερφορτωμένη υπορουτίνα με διαφορετικές παραμέτρους κάθε φορά.

Στις γραμμές 7-13 έχουμε τους ορισμούς των υπορουτινών printspeed.

Στην γραμμή 15 δηλώνουμε ένα αντικείμενο bmw τύπου car.

Στην γραμμή 16 καλούμε την υπορουτίνα μέλος printspeed με παράμετρο έναν ακέραιο. Έτσι τυπώνεται στην οθόνη η τιμή 100.

Στην γραμμή 17 καλούμε την υπορουτίνα μέλος printspeed με παράμετρο ένα double. Έτσι τυπώνεται στην οθόνη η τιμή 100,5.

## Ιδιότητες - Properties

Ένα UDT μπορεί να έχει ιδιότητες. Οι ιδιότητες είναι όμοιες με τα μέλη ρουτινών με την διαφορά ότι δηλώνονται με την λέξη Property αντί των λέξεων Sub, Function.

### Παράδειγμα:



```
1. Type Window
2.     Declare Property title(ByRef s As String)
3. Private:
4.     As String title_
5. End Type
6. Property Window.title(ByRef s As String)
7.     this.title_ = s
8. End Property
9. Dim As Window w
10. w.title = "My Window"
```

## Ανάλυση:

Στις γραμμές 1-5 έχουμε ένα UDT με το όνομα Window.

Στην γραμμή 2 δηλώνουμε μια ιδιότητα με το όνομα title και να παίρνει κατά αναφορά μια μεταβλητή τύπου String.

Στην γραμμή 3 έχουμε την λέξη Private που σημαίνει ότι το UDT έχει Private μεταβλητές.

Στην γραμμή 4 δηλώνουμε την Private μεταβλητή title\_ τύπου String.

Στην γραμμή 6-8 έχουμε τον ορισμό της ιδιότητας Window.title

Στην γραμμή 9 δηλώνουμε ένα αντικείμενο τύπου Window.

Στην γραμμή 10 εκχωρούμε στην ιδιότητα το όνομα του παραθύρου ως "My Window".

Είναι πολύ όμοια η χρήση μιας ιδιότητας με αυτήν μιας ρουτίνας μέλους, καθώς λαμβάνει μια παράμετρο και ανανεώνει την νέα κατάσταση του αντικείμενου.

Ωστόσο η σύνταξη για την εκχώρηση της παραμέτρου είναι μια βασική εκχώρηση και όχι μια κλήση σε ρουτίνα.

Εκχωρώντας την νέα τιμή στην ιδιότητα title, η ιδιότητα αυτόματα καλείται και αποδίδει την νέα τιμή.

Μετά από τον ορισμό του τίτλου του παραθύρου, είναι εφικτό επίσης να την προσπελάσει με μια ιδιότητα getter.

## Παράδειγμα:



```
1. Type Window
2.     '' setter
3.     Declare Property title(ByRef s As String)
4.     '' getter
5.     Declare Property title() As String
6. Private:
7.     As String title_
8. End Type
9.
10. '' setter
11. Property Window.title(ByRef s As String)
12.     this.title_ = s
13. End Property
14.
15. '' getter
16. Property Window.title() As String
17.     Return this.title_
18. End Property
19.
20.
21. Dim As Window w
22. w.title = "My Window"
23. Print w.title
```

## Ανάλυση:

Στις γραμμές 1-8 έχουμε το UDT με το όνομα Window.  
Στην γραμμή 3 έχουμε την δήλωση της ιδιότητας ως setter.  
Στην γραμμή 5 έχουμε την δήλωση της ιδιότητας ως getter.  
Στις γραμμές 11-13 έχουμε τον ορισμό της ιδιότητας setter.  
Στις γραμμές 16-18 έχουμε τον ορισμό της ιδιότητας getter.  
Στην γραμμή 20 δηλώνεται ένα αντικείμενο τύπου Window.  
Στην γραμμή 21 εκχωρείται μια τιμή στην ιδιότητα.  
Στην γραμμή 22 προσπελάζεται η ιδιότητα και τυπώνεται στην οθόνη η τιμή της.

Η ιδιότητα getter είναι πολύ όμοια με μια συνάρτηση. Επιστρέφει την τρέχουσα τιμή της ιδιότητας και επιτρέπει τον υπολογισμό της τρέχουσας τιμής.  
Σημείωση ότι και οι δύο setter και getter ιδιότητες έχουν του ίδιο προσδιοριστή δείχνοντας ότι χειρίζονται την ίδια ιδιότητα.

## Υπερφόρτωση setters.

Όπως στην υπερφόρτωση μεθόδων έτσι μπορούμε να έχουμε και υπερφόρτωση των setters μιας ιδιότητας.

## Παράδειγμα:



```
1. Type Window
2.     Declare Property title(ByRef s As String)
3.     Declare Property title(ByVal i As Integer)
4.     Declare Property title() As String
5. Private:
6.     As String title_
7. End Type
8.
9. Property Window.title(ByRef s As String)
10.    this.title_ = s
11. End Property
12.
13. Property Window.title(ByVal i As Integer)
14.    this.title_ = "Number: " & i
15. End Property
16.
17.
```



```

18. Property Window.title() As String
19.     Return this.title_
20. End Property
21.
22. Dim As Window w
23. w.title = "My Window"
24. Print w.title
25. w.title = 5
26. Print w.title

```

### Ανάλυση:

Το παράδειγμα είναι όμοιο με το προηγούμενο.

Στις γραμμές 2,3 έχουμε τις δηλώσεις της υπερφορτωμένης ιδιότητας.

Στις γραμμές 9 και 13 έχουμε τους ορισμούς της υπερφορτωμένης ιδιότητας title.

Στην γραμμή 22 εκχωρούμε μια συμβολοσειρά ως τιμή της ιδιότητας.

Στην γραμμή 24 εκχωρούμε έναν ακέραιο ως τιμή της ιδιότητας.

## Δικαιώματα πρόσβασης μελών και ενθυλάκωση (Member Access Rights and Encapsulation)

Με το όρο δικαιώματα πρόσβασης μελών ενός UDT εννοούμε τον περιορισμό πρόσβασης των μελών σε ορισμένα σημεία του κώδικα.

Όλα τα μέλη ενός UDT συμπεριλαμβανομένου μέλη μεταβλητές, ρουτίνες, απαριθμήσεις κτλ ανήκουν σε μια από τις παρακάτω κατατάξεις που η κάθε μία έχει τους δικούς της κανόνες για την πρόσβαση του κώδικα σε ένα UDT. Αυτοί οι κανόνες ονομάζονται, δικαιώματα πρόσβασης.

Υπάρχουν public, protected και private μέλη σε ένα UDT και ορίζονται ως τέτοια μέσα σε ένα μπλοκ κώδικα Type...End Type, από αντίστοιχες ετικέτες public, protected και private.

Όταν δεν υπάρχουν ετικέτες τα μέλη ενός UDT είναι public.

## **Public μέλη**

Τα public μέλη μπορούν να έχουν πρόσβαση οπουδήποτε, για παράδειγμα σε μέλη ρουτίνες, ή κώδικα του αρθρώματος ή ρουτίνες.

## **Protected μέλη**

Τα protected μέλη μπορούν μόνο να προσπελάζονται από μέλη ρουτίνες από τον τύπο που έχουν δηλωθεί ή από μέλη ρουτίνες από ένα παράγωγο τύπο. Δεν είναι ορατά σε εξωτερικό κώδικα.

## **Private μέλη**

Τα private μέλη μπορούν μόνο να είναι προσβάσιμα από μέλη ρουτίνες του τύπου που είναι δηλωμένα. Δεν είναι ορατά σε εξωτερικό κώδικα ή σε μέλη ρουτίνες παράγωγων τύπων.

## **Συναρτήσεις Δόμησης και Αποδόμησης, Constructors και destructors.**

Οι Constructors και destructors ακολουθούν τους ίδιους κανόνες όπως οποιοδήποτε άλλο μέλος:

- Όταν είναι public, τα αντικείμενα μπορούν να αρχικοποιούνται και να καταστρέφονται οπουδήποτε στον κώδικα.
- Όταν είναι protected, τα αντικείμενα μπορούν να αρχικοποιούνται και να καταστρέφονται μόνο από μέλη ρουτίνες του τύπου τους ή παράγωγου τύπου.
- Οι private constructors και destructors περιορίζουν την αρχικοποίηση του αντικειμένου μοναδιαία σε μέλη ρουτίνες του ίδιου τύπου.

# Ενθυλάκωση

## Γενικά

Η ενθυλάκωση (encapsulation) είναι η διαδικασία της απόκρυψης λεπτομερειών του πώς ένα αντικείμενο υλοποιεί διάφορες εργασίες του.

Αντί της άμεσης πρόσβασης των μελών, ο χρήστης έχει πρόσβαση στο αντικείμενο μέσω μιας δημόσιας διεπαφής. Με αυτόν τον τρόπο, της απόκρυψης, ο χρήστης είναι ικανός να χρησιμοποιεί το αντικείμενο χωρίς να χρειάζεται να κατανοεί το πώς αυτό υλοποιείται.

Η ενθυλάκωση υλοποιείται μέσω των προσδιοριστών πρόσβασης (Private, Protected ή Public).

Τυπικά, όλα τα μέλη μεταβλητές ενός Τύπου ορίζονται private, κρύβοντας τις λεπτομέρειες υλοποίησης, και οι περισσότερες ρουτίνες μέλη ορίζονται public, παρέχοντας μία διεπαφή στον χρήστη.

## Παράδειγμα - encapsulated.bas



```
1. Type car
2. Public:
3.     Declare Sub set_speed(ByVal new_speed As
Integer)
4.     Declare Sub get_speed(ByRef curr_speed As
Integer)
5.     Declare Sub Accelerate()
6. Private:
7.     Dim speed As Integer
8. End Type
9.
10. Sub car.set_speed(ByVal new_speed As Integer)
11.     This.speed = new_speed
12. End Sub
13.
14. Sub car.get_speed(ByRef curr_speed As Integer)
15.     curr_speed = This.speed
16. End Sub
17.
18. Sub car.Accelerate ()
19.     This.speed += 20
20. End Sub
21.
22. Dim bmw As car
23. Dim car_speed As Integer
```

```

24. | bmw.set_speed(100)
25. | bmw.get_speed(car_speed)
26. | Print car_speed
27. |
28. | bmw.Accelerate()
29. | bmw.get_speed(car_speed)
30. | Print car_speed
31. |
32. | Sleep
33. | End

```

## Έξοδος:



```

100
120

```

## Ανάλυση:

Στις γραμμές 1-8 έχουμε δηλωμένο το τύπο `car`.

Στις γραμμές 3,4 δηλώνουμε τις υπορουτίνες `getter/setter` ως `Public`.

Η υπορουτίνα `getter` περνάει μια παράμετρο κατά αναφορά ώστε να πάρει την τιμή της προστατευόμενης μεταβλητής μέλους `speed` και να την εκχωρήσει στην `curr_speed`.

Στην γραμμή 7 έχουμε ως `protected` το μέλος μεταβλητή `speed`. Εδώ γίνεται η ενθυλάκωση. Το μέλος μεταβλητή αποκρύβεται από τον τρόπο λειτουργίας του τύπου `car`.

Και οι υπορουτίνες `getter/setter` δίνουν μια διεπαφή χρήσης του τύπου `car`.

Από την γραμμή 10 μέχρι την 20 έχουμε τους ορισμούς των υπορουτινών μελών του τύπου `car`.

Στην γραμμή 22 δηλώνουμε ένα αντικείμενο `bmw` τύπου `car`.

Στην γραμμή 23 δηλώνουμε μια μεταβλητή ακεραίου για να λαμβάνουμε την ταχύτητα του αυτοκινητού/αντικειμένου.

Στην γραμμή 24 θέτουμε την ταχύτητα στο αντικείμενο.

Στην γραμμή 25 λαμβάνουμε την ταχύτητα με αναφορά στην μεταβλητή `car_speed`.

Στην γραμμή 26 τυπώνουμε στην οθόνη την ταχύτητα.

Στην γραμμή 28 καλούμε την υπορουτίνα `bmw.Accelerate()`.  
Στην γραμμή 29 λαμβάνουμε την ταχύτητα με αναφορά στην μεταβλητή `car_speed`.

Στην γραμμή 30 τυπώνουμε στην οθόνη την ταχύτητα.

Τέλος έχουμε την αναμονή του χρήστη και το τέλος του προγράμματος.

## **Κληρονομικότητα και Πολυμορφισμός (Inheritance Polymorphism)**

Μέχρι στιγμής είδαμε τι είναι ένα UDT και ότι μπορούμε να δημιουργήσουμε και εμείς τα δικά μας UDT.

Αν για παράδειγμα δημιουργήσουμε έναν τύπο `Car` τότε μπορούμε να δημιουργούμε αντικείμενα του τύπου δεδομένων `Car`.

Τι γίνεται όμως αν θέλουμε να δημιουργήσουμε πολλές εκδόσεις μοντέλων αυτοκινήτων ή αν κάθε νέο μοντέλο θέλουμε να συμπεριλαμβάνει χαρακτηριστικά από τα προηγούμενα μοντέλα;

Αυτό το πρόβλημα, δημιουργίας νέων παράγωγων τύπων UDT που παίρνουν τα χαρακτηριστικά άλλων τύπων έρχεται να λύσει η κληρονομικότητα.

Η κληρονομικότητα των UDT είναι ένας τρόπος για ένα UDT να γίνει η ειδική έκδοση ενός άλλου UDT.

Όταν ένα UDT κληρονομεί ένα άλλο, το νέο UDT λέμε ότι απορρέει από το πρώτο και ότι το πρώτο UDT είναι ο τύπος βάσης του νέου UDT.

Το νέο UDT θα έχει όλα τα μέλη του UDT βάσης όπως τις ιδιότητες, μεθόδους, μεταβλητές, σταθερές και ροτίνες και ότι επιπλέον εμείς προσθέσουμε.

Για παράδειγμα αν έχουμε ένα UDT `Car` με τις ιδιότητες `Color`, `Doors`, `Seats` και θέλουμε να δημιουργήσουμε ένα νέο

μοντέλο Car, το StarCar τότε μπορούμε να χρησιμοποιήσουμε την κληρονομικότητα ώστε το StarCar να λάβει όλα τα μέλη του Car και εμείς να προσθέσουμε ό,τι νέο μέλος θέλουμε π.χ. την ιδιότητα Brakes.

Η κληρονομικότητα επιτυγχάνεται με την λέξη Extends.

### Σύνταξη:



```
Type|Union typename Extends base_typename  
...  
End Type|Union
```

### Παράδειγμα - extends.bas



```
1. Type car  
2.     speed As Integer  
3.     colour As String  
4. End Type  
5.  
6. Type StarCar Extends car  
7.     brake As String  
8. End Type  
9.  
10. Dim bmw As car  
11. bmw.colour = "Red"  
12. bmw.speed = 100  
13.  
14.  
15. Dim starBMW As StarCar  
16. starBMW.colour = "White"  
17. starBMW.speed = 120  
18. starBMW.brake = "ABS"  
19.  
20. Print starBMW.brake  
21.  
22. Sleep  
23. End
```

## Έξοδος:



ABS

## Η λέξη-κλειδί **Base (Member Access)**

Παρέχει πρόσβαση σε μέλη τύπου βάσης που δεν είναι δηλωμένα ως Static.

Πολλές φορές από ένα αντικείμενο που έχει κληρονομήσει ένα άλλο θέλουμε να έχουμε πρόσβαση σε κάποιο μέλος του τύπου βάσης. Αυτό γίνεται με την λέξη-κλειδί Base.

## Σύνταξη:



Base.member

Base [ .Base ... ] .member

Σε περίπτωση πολλαπλής κληρονομικότητας χρησιμοποιούμε την λέξη-κλειδί base διαδοχικά, δηλαδή base.base.member κτλ.

## Παράδειγμα - base.bas



```
1. Type parent
2.     i As Integer
3.     Declare Sub show()
4. End Type
5.
6. Sub parent.show ()
7.     This.i = 10
8.     Print This.i
9. End Sub
10.
11. Type child Extends parent
12.     i As Integer
13.     Declare Sub show()
14. End Type
15.
16. Sub child.show ()
17.     This.i = 20
18.     Base.show()
19.     Print This.i
20. End Sub
21.
22. Dim p As parent
23. p.show
24.
25. Dim c As child
26. c.show
27.
28.
29. Sleep
30. End
```

### Έξοδος:



```
10
10
20
```

### Ανάλυση:

Στις γραμμές 1-4 δηλώνεται ένας τύπος χρήστη με όνομα parent, με μια μεταβλητή μέλος και μια υπορουτίνα μέλος. Στις γραμμές 6-9 έχουμε τον ορισμό της υπορουτίνας μέλος του τύπου χρήστη parent.



Στην γραμμή 7 εκχωρείται η τιμή 10 στην μεταβλητή μέλος i. Στην γραμμή 8 τυπώνεται στην οθόνη η τιμή της μεταβλητής μέλους i.

Στην γραμμή 11 με 14 δηλώνεται ο τύπος χρήστη child ο οποίος κληρονομεί τον τύπο χρήστη parent.

Στην γραμμή 12 δηλώνεται η μεταβλητή μέλος i του τύπου child.

Στην γραμμή 13 δηλώνεται η υπορουτίνα μέλος show() του τύπου child.

Στις γραμμές 16-20 υπάρχει ο ορισμός της υπορουτίνας μέλους show() του τύπου child.

Στην γραμμή 17 εκχωρείται η τιμή 20 στην μεταβλητή μέλος i του τύπου child.

Στην γραμμή 18 καλείται η υπορουτίνα Show() από τον τύπο βάση parent.

Στην γραμμή 19 τυπώνεται στην οθόνη η μεταβλητή μέλος i του τύπου child.

Στην γραμμή 22 δηλώνεται ένα αντικείμενο τύπου parent με το όνομα p.

Στην γραμμή 23 καλείται η υπορουτίνα μέλος show του τύπου parent. Και έτσι τυπώνεται στην οθόνη η τιμή 10.

Στην γραμμή 25 δηλώνεται ένα αντικείμενο τύπου child με όνομα c.

Στην γραμμή 26 καλείται η υπορουτίνα μέλος show του αντικειμένου child. Στην υπορουτίνα αυτή πρώτα καλούμε την υπορουτίνα της βάσης και τυπώνεται στην οθόνη πάλι η τιμή 10 και μετά τυπώνουμε την τιμή της μεταβλητής μέλους i που έχει την τιμή 20.

Τέλος στις γραμμές 28,29 έχουμε την αναμονή του χρήστη και το τέλος του προγράμματος.

## Ο τελεστής Is (Run-Time Type Information)

Ο τελεστής Is, ελέγχει εάν ένα αντικείμενο έχει συμβατό τύπο που παράγεται από έναν άλλο τύπο την ώρα της μεταγλώττισης.

### Σύνταξη:



result = expression Is typename

### expression

Η έκφραση που ελέγχεται, είναι ένα αντικείμενο το οποίο παράγεται άμεσα ή έμμεσα από το Object χρησιμοποιώντας το Extends.

### result

Επιστρέφει (-1) εάν η έκφραση είναι αντικείμενο ενός τύπου typename ή είναι ένα αντικείμενο βάσης παραγόμενο από τον τύπο της έκφρασης. Επιστρέφει (0) εάν είναι ένα αντικείμενο ασύμβατου τύπου.

## Παράδειγμα - is.bas



```
1. Type Vehicle Extends Object
2.     As String Name
3. End Type
4.
5. Type Car Extends Vehicle
6. End Type
7.
8. Type Cabriolet Extends Car
9. End Type
10.
11. Type Bike Extends Vehicle
12. End Type
13.
14.
15. Sub identify(ByVal p As Object Ptr)
16.     Print "Identifying:"
17.     ' Not a Vehicle object?
18.     If Not (*p Is Vehicle) Then
19.         Print , "unknown object"
20.         Return
21.     End If
22.     ' The cast is safe, because we know it's a
23. Vehicle object
24.     Print , "name: " & CPtr(Vehicle Ptr, p)->Name
25.     If *p Is Car Then
26.         Print , "It's a car"
27.     End If
28.     If *p Is Cabriolet Then
29.         Print , "It's a cabriolet"
30.     End If
31.     If *p Is Bike Then
32.         Print , "It's a bike"
33.     End If
34. End Sub
35.
36. Dim As Car ford
37. ford.name = "Ford"
38. identify(@ford)
39.
40. Dim As Cabriolet porsche
41. porsche.name = "Porsche"
42. identify(@porsche)
43.
44.
45. Dim As Bike mountainbike
46. mountainbike.name = "Mountain Bike"
47. identify(@mountainbike)
```

```

48.
49. Dim As Vehicle v
50. v.name = "some unknown vehicle"
51. identify(@v)
52.
53. Dim As Object o
54. identify(@o)
55.
56. Sleep
57. End

```

## Έξοδος:



```

Identifying:
    name: Ford
    It's a car
Identifying:
    name: Porsche
    It's a car
    It's a cabriolet
Identifying:
    name: Mountain Bike
    It's a bike
Identifying:
    name: some unknown vehicle
Identifying:
    unknown object

```

## Ανάλυση:

Στις γραμμές 1-12 έχουμε τις δηλώσεις των UDT με διαφορετική κληρονομικότητα το κάθε ένα.

Στις γραμμές 14 με 32 έχουμε την υπορουτίνα identify που δέχεται κατα αναφορά έναν δείκτη τύπου Object και εκτελεί λογικούς ελέγχους για το τι τύπος είναι.

Στις γραμμές 34-51 έχουμε τις δηλώσεις αντικειμένων διάφορων τύπων UDT που ελέγχονται με την συνάρτηση identify.

# Virtual και Abstract μέθοδοι

Οι virtual μέθοδοι είναι μέθοδοι οι οποίες μπορούν να υπερκαλυφθούν από άλλες μεθόδους στα κληρονομούμενα αντικείμενα επιτρέποντας έτσι τον δυναμικό πολυμορφισμό. Σε αντίθεση με τις Abstract μεθόδους οι virtual μέθοδοι πρέπει να έχουν μια υλοποίηση η οποία θα χρησιμοποιηθεί όταν η μέθοδος δεν θα υπερκαλυφθεί.

## Σύνταξη:



```
Type typename Extends base_typename  
  Declare Virtual Sub|Function|Property|Operator|  
  Destructor ...  
End Type
```

## Παράδειγμα - virtual.bas



```
1. Type Hello Extends Object  
2.     Declare Virtual Sub hi( )  
3. End Type  
4.  
5. Type HelloEnglish Extends Hello  
6.     Declare Sub hi( ) ' ' overriding subroutine  
7. End Type  
8.  
9. Type HelloFrench Extends Hello  
10.    Declare Sub hi( ) ' ' overriding subroutine  
11. End Type  
12.  
13. Type HelloGerman Extends Hello  
14.    Declare Sub hi( ) ' ' overriding subroutine  
15. End Type  
16.  
17. Sub Hello.hi( )  
18.     Print "hi!"  
19. End Sub  
20.  
21. Sub HelloEnglish.hi( ) ' ' overriding subroutine  
22.     Print "Hello!"  
23. End Sub  
24.  
25. Sub HelloFrench.hi( ) ' ' overriding subroutine  
26.     Print "Salut!"  
27. End Sub  
28.
```

```

29.
30. Sub HelloGerman.hi( ) ' ' overriding subroutine
31.     Print "Hallo!"
32. End Sub
33.
34. Randomize( Timer( ) )
35.
36.
37. Dim As Hello Ptr h
38.
39. For i As Integer = 0 To 9
40.     Select Case( Int( Rnd( ) * 4 ) + 1 )
41.     Case 1
42.         h = New HelloEnglish
43.     Case 2
44.         h = New HelloFrench
45.     Case 3
46.         h = New HelloGerman
47.     Case Else
48.         h = New Hello
49.     End Select
50.     h->hi( )
51.     Delete h
52. Next
53.
54. Sleep
55. End

```

## Έξοδος:



```

hi!
hi!
Salut!
Hello!
Salut!
Salut!
Hallo!
Hallo!
Hallo!
Hello!

```

## Ανάλυση:

Στις γραμμές 1-15 έχουμε τις δηλώσεις των UDT.

Στην γραμμή 2 δηλώνουμε ως Virtual την υπορουτίνα hi().

Στις γραμμές 17-31 έχουμε τους ορισμούς των υπορουτινών hi() για κάθε τύπο.

Στην γραμμή 33 αρχικοποιούμε την γεννήτρια τυχαίων αριθμών.

Στην γραμμή 35 δηλώνουμε έναν δείκτη τύπου Hello.

Στην γραμμή 37 έχουμε έναν βρόγχο επανάληψης For...Next με 10 επαναλήψεις.

Στην γραμμή 38 η συνάρτηση Rnd επιστρέφει έναν ακέραιο αριθμό από το 1 έως το 4. Επίσης έχουμε ένα μπλοκ Select για τις 4 περιπτώσεις.

Σε κάθε Case δημιουργείται ένα νέο αντικείμενο από τα UDT που ορίσαμε παραπάνω. Αν ο τυχαίος ακέραιος αριθμός είναι 4 τότε δημιουργείται ένα αντικείμενο Hello και καλείται η Virtual μέθοδος.

Στην γραμμή 48 καλείται η υπορουτίνα hi() του αντικειμένου h. Ανάλογα ποιο αντικείμενο έχει οριστεί καλείται η υπορουτίνα hi().

Στην γραμμή 49 διαγράφεται το αντικείμενο h που προηγουμένως δημιουργήθηκε με την λέξη New.

Τέλος έχουμε την αναμονή του χρήστη και το τέλος του προγράμματος.

# Abstract μέθοδοι.

Δήλωση μεθόδων Abstract.

## Σύνταξη:



```
Type typename Extends base_typename  
  Declare Abstract Sub|Function|Property|  
  Operator ...  
End Type
```

Οι μέθοδοι Abstract είναι ειδικές Virtual μέθοδοι με την διαφορά όμως ότι έχουν μόνο δήλωση και όχι υλοποίηση. Η υλοποίηση τους γίνεται σε παράγωγα UDT που κληρονομούν άλλους τύπους.

## Παράδειγμα - abstract.bas



```
1. Type Hello Extends Object  
2.   Declare Abstract Sub hi( )  
3. End Type  
4.  
5. Type HelloEnglish Extends Hello  
6.   Declare Sub hi( )  
7. End Type  
8.  
9. Type HelloFrench Extends Hello  
10.  Declare Sub hi( )  
11. End Type  
12.  
13. Type HelloGerman Extends Hello  
14.   Declare Sub hi( )  
15. End Type  
16.  
17. Sub HelloEnglish.hi( )  
18.   Print "hello!"  
19. End Sub  
20.  
21. Sub HelloFrench.hi( )  
22.   Print "Salut!"  
23. End Sub  
24.  
25. Sub HelloGerman.hi( )  
26.   Print "Hallo!"  
27.
```



```

28. End Sub
29.
30. Randomize( Timer( ) )
31. Dim As Hello Ptr h
32. For i As Integer = 0 To 9
33.     Select Case( Int( Rnd( ) * 3 ) + 1 )
34.         Case 1
35.             h = New HelloFrench
36.         Case 2
37.             h = New HelloGerman
38.         Case Else
39.             h = New HelloEnglish
40.         End Select
41.         h->hi( )
42.         Delete h
43.     Next
44. Sleep
45. End

```

## Έξοδος:



```

Hallo!
Salut!
Hallo!
Salut!
Salut!
Hallo!
Hallo!
Salut!
hello!
hello!

```

Το παράδειγμα είναι το ίδιο με το προηγούμενο, το virtual.bas με την διαφορά ότι είναι προσαρμοσμένο στην abstract μέθοδο.

# Κεφάλαιο 11ο - Run Time Functions

Στο κεφάλαιο αυτό θα δούμε τις ενσωματωμένες συναρτήσεις που παρέχει η γλώσσα FreeBASIC κατά τον χρόνο εκτέλεσης της.

Οι συναρτήσεις αυτές είναι στις εξής κατηγορίες:

Συναρτήσεις πινάκων - Array Functions

Διαχείριση Bit - Bit Manipulation

Συναρτήσεις κονσόλας - Console Functions

Συναρτήσεις ημερομηνίας και ώρας- Date and Time Functions

Συναρτήσεις διαχείρισης λαθών - Error Handling Functions

Συναρτήσεις εισόδου/εξόδου αρχείων - File IO Functions

Μαθηματικές συναρτήσεις - Mathematical Functions

Συναρτήσεις μνήμης - Memory Functions

Συναρτήσεις λειτουργικού συστήματος - Operating System Functions

Συναρτήσεις συμβολοσειρών - String Functions

Συναρτήσεις υποστήριξης νήματος - Threading Support Functions

Συναρτήσεις εισόδου του χρήστη - User Input Functions

## Συναρτήσεις πινάκων - Array Functions

### ReDim

Δηλώνει ή αλλάζει το μέγεθος ενός δυναμικού πίνακα.

#### Σύνταξη:



```
ReDim [ Shared ] symbolname([subscript [, ...]]) As  
datatype [, ...]
```

```
ReDim [ Shared ] As datatype  
symbolname([subscript [, ...]]) [, ...]
```

```
ReDim [ Preserve ] symbolname([subscript [, ...]])  
[, ...]
```

ή:  
ReDim [ Preserve ] [ ( ) expression [ ) ] ([subscript [, ...]]) [, ...]

Η λέξη-κλειδί ReDim μπορεί να χρησιμοποιηθεί για να δηλώσει ένα νέο δυναμικό πίνακα ή να αλλάξει το μέγεθος ενός υπάρχοντος δυναμικού πίνακα με το να διατηρήσει τις ίδιες διαστάσεις του πίνακα.

Όταν δηλώνεται ένας νέος δυναμικός πίνακας τα στοιχεία του εξ' ορισμού δημιουργούνται. Για απλούς τύπους δεδομένων όπως Integer ή Double τα στοιχεία του πίνακα αρχικοποιούνται με την τιμή μηδέν. Για UDT με εξ' ορισμού συνάρτηση δόμησης καλείται η συνάρτηση δόμησης.

## Παράδειγμα - redim.bas



```
1.  '' Define a variable-length array with 5 elements
2.  ReDim array(0 To 4) As Integer
3.
4.  For index As Integer = LBound(array) To
5.  UBound(array)
6.      array(index) = index
7.  Next
8.
9.  '' Resize a variable-length array with 10
10. elements
11. '' (the lower bound should be kept the same)
12. ReDim Preserve array(0 To 9)
13.
14. Print "index", "value"
15. For index As Integer = LBound(array) To
16. UBound(array)
17.     Print index, array(index)
18. Next
19.
20. Sleep
21. End
```

## Έξοδος:



```
index    value
0         0
1         1
2         2
3         3
4         4
5         0
6         0
7         0
8         0
9         0
```

## Ανάλυση:

Στην γραμμή 2 δηλώνεται ένας πίνακας με 5 στοιχεία.

Στις γραμμές 4-6 υπάρχει ένας βρόγχος επανάληψης For...Next. Για τιμές από το κατώτερο όριο του πίνακα μέχρι το ανώτερο όριο του πίνακα στην γραμμή 5 αρχικοποιούμε τον πίνακα για κάθε του στοιχείο.

Στην γραμμή 10 αναμεγεθύνουμε τον πίνακα στις διαστάσεις 0 με 9 δηλαδή για 10 στοιχεία, διατηρώντας τις προηγούμενες τιμές του.

Στην γραμμή 13-15 υπάρχει ένας βρόγχος επανάληψης For...Next ο οποίος τυπώνει στην οθόνη τις τιμές των στοιχείων του πίνακα.

## Preserve

Χρησιμοποιείται με την λέξη-κλειδί ReDim για να διατηρήσει τα περιεχόμενα ενός δυναμικού πίνακα κατά την αναμεγέθυνση του.

## Σύνταξη:



ReDim **Preserve** array(...) [As datatype]

## Παράδειγμα - preserve.bas



```
1. ReDim array(1 To 3) As Integer
2. array(1) = 10
3. array(2) = 5
4. array(3) = 8
5.
6. ReDim Preserve array(1 To 10)
7.
8. Dim i As Integer
9. For i = 1 To 10
10.     Print "array("; i; ") = "; array(i)
11. Next
12.
13. Sleep
14. End
```

## Έξοδος:



```
array( 1) = 10
array( 2) = 5
array( 3) = 8
array( 4) = 0
array( 5) = 0
array( 6) = 0
array( 7) = 0
array( 8) = 0
array( 9) = 0
array(10) = 0
```

## Ανάλυση:

Στην γραμμή 1 δηλώνεται ένας πίνακας με 3 στοιχεία.  
Στις γραμμές 2-4 ορίζονται τιμές στα στοιχεία του πίνακα.  
Στην γραμμή 6 αναμεγεθύνεται ο πίνακας διατηρώντας τα περιεχόμενα του.

Στην γραμμή 9-11 υπάρχει ένας βρόγχος επανάληψης For...Next όπου τυπώνει στην οθόνη τις τιμές των στοιχείων του πίνακα. Σημείωση ότι οι νέες τιμές μετά την αναμεγέθυνση είναι μηδέν εξ' ορισμού.

## Erase

Υπορουτίνα που διαγράφει πίνακες.

### Σύνταξη:



`Erase( array0 [, array1 ... arrayN ] )`

Όταν χρησιμοποιείται σε πίνακες ορισμένου μήκους κάνει reset σε όλα τα στοιχεία χωρίς να απελευθερώνει την δεσμευμένη μνήμη.

Στην περίπτωση αντικειμένων γίνεται καταστροφή και επαναδόμηση.

Όταν χρησιμοποιείται σε δυναμικό πίνακα απελευθερώνει την μνήμη που δεσμεύτηκε για τα στοιχεία του πίνακα, αλλά ο πίνακας παραμένει δηλωμένος στο ίδιο επίπεδο εμβέλειας με το ίδιο τύπο δεδομένων και αριθμό διαστάσεων παρά μόνο οι ανώτερες/κατώτερες τιμές των διαστάσεων επαναφέρονται στις τιμές -1/0.

Στην περίπτωση που έχουμε αντικείμενα γίνεται καταστροφή πριν απελευθερωθεί η μνήμη.

## Παράδειγμα - erase.bas



```
1. Dim MyArray1(1 To 10) As Integer
2. ReDim MyArray2(1 To 10) As Integer
3.
4. Print "MyArray1", LBound( MyArray1 ),
   UBound( MyArray1 )
5. Print "MyArray2", LBound( MyArray2 ),
   UBound( MyArray2 )
6.
7. Erase MyArray1, MyArray2
8.
9. Print "MyArray1", LBound( MyArray1 ),
   UBound( MyArray1 )
10. Print "MyArray2", LBound( MyArray2 ),
    UBound( MyArray2 )
11.
12. Sleep
13. End
```

### Έξοδος:



```
MyArray1    1      10
MyArray2    1      10
MyArray1    1      10
MyArray2    0      -1
```

### Ανάλυση:

Στις γραμμές 1-2 δηλώνονται δύο πίνακες.

Στην γραμμή 4 τυπώνονται στην οθόνη τα όρια του MyArray1.

Στην γραμμή 5 τυπώνονται στην οθόνη τα όρια του MyArray2.

Στην γραμμή 7 διαγράφονται οι δύο πίνακες.

Στις γραμμές 9,10 ξανατυπώνονται τα όρια των δύο πινάκων μετά την διαγραφή.

## LBound και UBound

Οι συναρτήσεις αυτές επιστρέφουν το κατώτερο και το ανώτερο όριο ενός πίνακα.

### Σύνταξη:



```
result = LBound( array [, dimension ] )  
result = UBound( array [, dimension ] )
```

### Παράδειγμα - LUBound.bas



```
1. Dim array(1 To 5) As Integer  
2.  
3. Print LBound(array)  
4. Print UBound(array)  
5.  
6. Sleep  
7. End
```

### Έξοδος:



```
1  
5
```

### Ανάλυση:

Στην γραμμή 1 δηλώνεται ένας πίνακας τύπου ακεραίου.

Στην γραμμή 3 τυπώνεται στην οθόνη το κατώτερο όριο του πίνακα.

Στην γραμμή 4 τυπώνεται στην οθόνη το ανώτερο όριο του πίνακα.

Τέλος έχουμε την αναμονή του χρήστη και το τέλος του προγράμματος.



# Διαχείριση bit - Bit Manipulation

## LoByte

Παίρνει το χαμηλότερο byte του τελεστή.

### Σύνταξη:



result = LoByte( expr )

## HiByte

Παίρνει το δεύτερο byte του τελεστή.

### Σύνταξη:



result = HiByte( expr )

## Παράδειγμα - LHByte.bas



```
1. Dim N As UInteger
2.
3. 'Note there are 16 bits
4. N = &b1010101110000001
5. Print "N is "; N
6. Print "The binary representation of N is
   "; Bin(N)
7. Print "The most significant byte (MSB) of N is
   "; HiByte(N)
8. Print "The least significant byte (LSB) of N is
   "; LoByte(N)
9. Print "The binary representation of the MSB is
   "; Bin(HiByte(N))
10. Print "The binary representation of the LSB is
    "; Bin(LoByte(N))
11. Sleep
12. End
```

## Έξοδος:



```
N is 43905
The binary representation of N is
1010101110000001
The most significant byte (MSB) of N is 171
The least significant byte (LSB) of N is 129
The binary representation of the MSB is
10101011
The binary representation of the LSB is
10000001
```

## LoWord και HiWord

### Σύνταξη:



#### **LoWord**

result = LoWord( expr )

#### **HiWord**

result = HiWord( expr )

LoWord: Παίρνει τη χαμηλότερη λέξη 16bit του τελεστή.

HiWord: Παίρνει τη δεύτερη λέξη 16bit του τελεστή

## Παράδειγμα - LoHiWord.bas



```
1. Dim N As UInteger
2.
3. 'Note there are 32 bits
4. N = &b10000000000000001111111111111111
5.
6.
7. Print "N is "; N
8. Print "The binary representation of N is
   "; Bin(N)
9. Print "The most significant word (MSW) of N is
   "; HiWord(N)
10. Print "The least significant word (LSW) of N is
    "; LoWord(N)
11. Print "The binary representation of the MSW is
    "; Bin(HiWord(N))
    Print "The binary representation of the LSW is
    "; Bin(LoWord(N))
```

```
12. Sleep
13. End
14.
```

## Έξοδος:



N is 2147614719

The binary representation of N is  
1000000000000001111111111111111

The most significant word (MSW) of N is 32769  
The least significant word (LSW) of N is 65535

The binary representation of the MSW is  
1000000000000001

The binary representation of the LSW is  
111111111111111

## Bit

Παίρνει την κατάσταση ενός μεμονωμένου bit σε ακέραιη τιμή.

## Σύνταξη:



result = Bit( value, bit\_number )

## Παράδειγμα - bit.bas



```
1. Print Bit(&B1000, 3)
2. Print Bit(4,2)
3. Print Bit(5,1)
4. Print Bit(&H8000000000000000ULL,63)
5. Sleep
6. End
```

Έξοδος:



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

## BitReset

Παίρνει την τιμή με ένα καθορισμένο bit εκκαθαρισμένο, από έναν αντιγραμμένο ακέραιο.

Σύνταξη:



```
result = BitReset( value, bit_number )
```

## Παράδειγμα - bitreset.bas



```
1. Print Bin(BitReset(&b10101, 2))  
2. Print BitReset(5,0)  
3. Print Hex(BitReset(&h8000000000000001,63))  
4.  
5. Sleep  
6. End
```

Έξοδος:



```
10001  
4  
1
```

## BitSet

Λαμβάνει την τιμή με ένα καθορισμένο σύνολο δυαδικών ψηφίων, από έναν ακέραιο αριθμό αντιγραφής.

### Σύνταξη:



```
result = BitSet( value, bit_number )
```

### Παράδειγμα - `bitset.bas`



```
1. Print Bin(BitSet(&b10001,2))
2. Print BitSet(4, 0)
3. Print Hex(BitSet(1ull, 63))
4.
5. Sleep
6. End
```

### Έξοδος:



```
10101
5
80000000000000001
```

# Συναρτήσεις κοσνόλας - Console Functions

## Cls

Καθαρίζει την οθόνη τόσο σε λειτουργία κειμένου όσο και σε λειτουργία γραφικών.

### Σύνταξη:



Cls mode

Η λέξη mode παίρνει τιμές από το 0 έως το 2.

Εάν είναι 0, καθαρίζει όλη την οθόνη

Εάν είναι 1, καθαρίζει τα γραφικά αν έχουν οριστεί, αλλιώς καθαρίζει το κείμενο.

Εάν είναι 2, καθαρίζει το κείμενο από την οθόνη.

Εάν παραλείπεται καθαρίζει και τα γραφικά και το κείμενο.

### Παράδειγμα - cls.bas



```
1.  ' set the color to light grey text on a blue
    background
    Color 7, 1
2.
3.  ' print text in the center of the screen
    Locate 12, 33
    Print "Hello Universe!"
4.
5.
6.
7.  Sleep
8.
9.
10. ' clear the screen to the background color
    Cls
11.
12. Sleep
13. End
14.
```

## Ανάλυση:

Στην γραμμή 2 ορίζεται το χρώμα του φόντου και του παρασκηνίου.

Στην γραμμή 5 ορίζουμε τις κολώνες και γραμμές που θα εμφανιστεί το κείμενο που θα τυπώσουμε στην οθόνη.

Στην γραμμή 6 τυπώνουμε το κείμενο στην οθόνη.

Στην γραμμή 8 βάζουμε τον χρήστη στην αναμονή πριν καθαρίσουμε την οθόνη.

Στην γραμμή 11 καθαρίζουμε την οθόνη στο χρώμα του παρασκηνίου.

Τέλος έχουμε την αναμονή του χρήστη και το τέλος του προγράμματος.

## Width

Ορίζει ή λαμβάνει τον αριθμό των γραμμών και στηλών της οθόνης

### Σύνταξη:



Width [columns] [, rows]

Επιστρέφει έναν 32 bit Long, όπου το High Word είναι ο αριθμός των γραμμών και το Low Word είναι ο αριθμός των στηλών που έχουν οριστεί αυτήν τη στιγμή.

### Παράδειγμα - width.bas



```
1. Dim As Integer w
2. w = Width
3. Print "rows: " & HiWord(w)
4. Print "cols: " & LoWord(w)
5.
6. Sleep
7. End
```

## Έξοδος:



```
rows: 30  
cols: 120
```

### Ανάλυση:

Στην γραμμή 1 δηλώνεται ένας ακέραιος.

Στην γραμμή εκχωρείται η τιμή από την συνάρτηση Width, στον ακέραιο.

Στις γραμμές 3,4 τυπώνεται στην οθόνη οι γραμμές και στήλες του μήκους της οθόνης.

Τέλος έχουμε την αναμονή του χρήστη και το τέλος του προγράμματος.

## View Print

Ορίζει την εκτυπώσιμη περιοχή κειμένου της οθόνης

### Σύνταξη:



```
View Print [ firstrow To lastrow ]
```

### Παράδειγμα:



```
1. Cls  
2. View Print 5 To 6  
3. Color , 1  
4. '' clear only View Print area  
5. Cls
```

### Ανάλυση:

Στην γραμμή 1 καθαρίζουμε όλη την οθόνη

Στην γραμμή 2 ορίζουμε την εκτυπώσιμη περιοχή κειμένου της οθόνης.



Στην γραμμή 3 θέτουμε τα χρώματα της περιοχής αυτής. Στην γραμμή 5 καθαρίζουμε την εκτυπώσιμη περιοχή της οθόνης.

## Color

Αλλάζει το χρώμα του προσκηνίου και του φόντου του κειμένου που πρόκειται να γραφτεί.

### Σύνταξη:



```
Color [foreground] [, background]
result = Color [( [foreground] [, background] )]
```

### Παράδειγμα - color.bas



```
1. ' Sets 320x240 in 32bpp color depth
2. Screen 14, 32
3.
4. ' Sets orange foreground and dark blue background
   color
5. Color RGB(255, 128, 0), RGB(0, 0, 64)
6.
7. ' Clears the screen to the background color
8. Cls
9.
10. ' Prints "Hello World!" in the middle of the
    screen
11. Locate 15, 14
12. Print "Hello World!"
13.
14. Sleep
15. End
```

## CsrLin

Επιστρέφει τη θέση γραμμής του δρομέα

### Σύνταξη:



```
result = CsrLin
```

### Παράδειγμα:



```
1. Print "The cursor is on row:"; CsrLin  
2.  
3. Sleep  
4. End
```

## Pos

Επιστρέφει την οριζόντια (από αριστερά προς τα δεξιά) θέση του δρομέα κειμένου

### Σύνταξη:



```
result = Pos ( )
```

## Locate

Ορίζει την τρέχουσα θέση δρομέα

### Σύνταξη:



```
Locate [row], [column], [state]
```

state: η κατάσταση του δρομέα μόνο στη λειτουργία κονσόλας: 0 είναι απενεργοποιημένο, 1 ενεργοποιημένο. ο δρομέας κειμένου δεν είναι ποτέ ορατός στη λειτουργία γραφικών.

## Screen

Λαμβάνει το χαρακτηριστικό χαρακτήρα ή χρώματος σε μια δεδομένη τοποθεσία της οθόνης.

### Σύνταξη:



result = Screen( row, column [, colorflag ] )

Η Screen επιστρέφει τον χαρακτήρα ή το χαρακτηριστικό χρώματος που βρίσκεται σε μια δεδομένη θέση εξόδου κονσόλας. Λειτουργεί σε λειτουργία κονσόλας και σε λειτουργία γραφικών.

Η μορφή του χαρακτηριστικού χρώματος εξαρτάται από το τρέχον βάθος χρώματος:

Οι τιμές χρωμάτων για την τυπική παλέτα χρωμάτων 16 είναι:

Value	Color	Value	Color
0	Black	8	Gray
1	Blue	9	Bright Blue
2	Green	10	Bright Green
3	Cyan	11	Bright Cyan
4	Red	12	Bright Red
5	Magenta	13	Pink

6	Brown	14	Yellow
7	White	15	Bright White

## Print | ?

Τυπώνει κείμενο στην οθόνη.

### Σύνταξη:



(Print | ?) [ expressionlist ] [ , | ; ]

### Παράδειγμα:



```

1.  '' print "Hello World!", and a new-line
2.  Print "Hello World!"
3.
4.  '' print several strings on one line, then print
    a new-line
5.  Print "Hello";
6.  Print "World"; "!";
7.  Print
8.
9.  '' column separator
10. ? "Hello!", "World!"

```

## Write

Εξάγει μια λίστα τιμών διαχωρισμένη με κόμμα στην οθόνη

### Σύνταξη:



Write [ expressionlist ]

## Παράδειγμα:



```
1. Dim i As Integer = 10
2. Dim d As Double = 123.456
3. Dim s As String = "text"
4.
5. Write 123, "text", -.45600
6. Write
7. Write i, d, s
```

## Έξοδος:



```
123,"text",-0.456
10,123.456,"text"
```

## Spc

Τυπώνει κενά στην οθόνη.

## Σύνταξη:



Print Spc( spaces ) [(, | ;)] ...

## Παράδειγμα - spc.bas



```
1. Print "foo"; Spc(5); "bar"
2. Print "hello"; Spc(4); "world"
3.
4. Sleep
5. End
```

## Έξοδος:



```
foo   bar
hello world
```

## Tab

Ορίζει τη στήλη κατά την εγγραφή σε οθόνη ή αρχείο

## Σύνταξη:



Print Tab( column ) [(, | ;)] ...

## Παράδειγμα - tab.bas



```
1.  ''Using Print with Tab to justify text in a table
2.
3.  Dim As String A1, B1, A2, B2
4.
5.  A1 = "Jane"
6.  B1 = "Doe"
7.  A2 = "Bob"
8.  B2 = "Smith"
9.
10.
11. Print "FIRST NAME"; Tab(35); "LAST NAME"
12. Print "-----"; Tab(35); "-----"
13. Print A1; Tab(35); B1
14. Print A2; Tab(35); B2
15.
16. Sleep
    End
```

## Έξοδος:



```
FIRST NAME                LAST NAME
-----
Jane                       Doe
Bob                         Smith
```

# Συναρτήσεις ημερομηνίας και ώρας - Date and Time Functions

Διαδικασίες που λειτουργούν με ημερομηνίες και ώρα.

Αυτές οι διαδικασίες παρέχουν τρόπους χρήσης των διαστημάτων ημερομηνίας και ώρας με συνεπή τρόπο. Παρέχονται πρόσθετες διαδικασίες για τον καθορισμό και τη λήψη της τρέχουσας ημερομηνίας και ώρας του συστήματος και την ανάκτηση μιας χρονικής σφραγίδας για ευαίσθητους αλγορίθμους χρονισμού.

## Now

Λαμβάνει την τρέχουσα ώρα συστήματος ως σειριακή ημερομηνία

### Σύνταξη:



```
#include "vbcompat.bi"  
result = Now
```

Επιστρέφει μια σειριακή ημερομηνία που περιέχει την ημερομηνία και την ώρα του συστήματος την ώρα εκτέλεσης.

Καθώς η ώρα είναι το δεκαδικό μέρος μιας σειριακής ημερομηνίας, εάν η τιμή του Now αποθηκευτεί σε έναν ακέραιο, η ώρα σε αυτήν θα επαναφερθεί σε 00:00:00

Ο μεταγλωττιστής δεν θα αναγνωρίσει αυτήν τη συνάρτηση εκτός εάν περιλαμβάνεται το vbcompat.bi.

## Παράδειγμα - now.bas



```
1. #include "vbcompat.bi"  
2.  
3. Dim a As Double = Now()  
4.  
5.  
6. Print a  
7. Print Format(a, "yyyy/mm/dd hh:mm:ss")  
8.  
9. Sleep  
10. End
```

## Έξοδος:



```
44410.68885416666  
2021-08-02 16:31:57
```

## Ανάλυση:

Στην γραμμή 1 εισάγουμε το αρχείο επικεφαλίδας vbcompat.bi.

Η έκφραση #include είναι μια εντολή προς τον προεπεξεργαστή να ενσωματώσει το αρχείο επικεφαλίδας που ακολουθεί. Περισσότερα σε επόμενο κεφάλαιο για τις εξωτερικές βιβλιοθήκες της FreeBASIC.

Στην γραμμή 3 δηλώνεται ένας Double, και εκχωρείται σε αυτόν η τιμή της σειριακής ημερομηνίας από την συνάρτηση Now().

Για να δούμε την τιμή της a, τυπώνουμε την τιμή της στην οθόνη, στην γραμμή 5.

Στην γραμμή 6 μορφοποιούμε την τιμή της μεταβλητής a σε ημερομηνία αναγνώσιμη από τον χρήστη και την τυπώνουμε στην οθόνη.

Τέλος έχουμε την αναμονή του χρήστη και το τέλος του προγράμματος.



## DateSerial

Δημιουργεί μια σειριακή ημερομηνία.

Ένας σειριακός αριθμός ημερομηνίας είναι ένας αριθμός που περιέχει τιμή ημερομηνίας και ώρας στην ίδια μορφή που χρησιμοποιείται από το PDS ή το VBDOS. Η τιμή είναι μια καταμέτρηση των ημερών από τις 0:00 π.μ. της 30 Δεκεμβρίου 1899. Χρησιμοποιείται κυρίως για εύκολο υπολογισμό του χρόνου μεταξύ δύο ημερομηνιών.

### Σύνταξη:



```
#include "vbcompat.bi"  
result = DateSerial( year, month, day )
```

Επιστρέφει μια σειριακή ημερομηνία που περιέχει την ημερομηνία που σχηματίζεται από τις τιμές στις παραμέτρους του έτους, του μήνα και της ημέρας. Η σειριακή ημερομηνία που επιστρέφεται δεν έχει δεκαδικό τμήμα.

Ο μεταγλωττιστής δεν θα αναγνωρίσει αυτήν τη συνάρτηση, εκτός εάν περιλαμβάνεται το vbcompat.bi ή το datetime.bi.

### Παράδειγμα - dateserial.bas



```
1. #include "vbcompat.bi"  
2.  
3. Dim a As Double = DateSerial(2005, 11, 28)  
4.  
5. Print Format(a, "yyyy/mm/dd hh:mm:ss")  
6.  
7. Sleep  
8. End  
9.
```

## Έξοδος:



2005-11-28 00:00:00

## TimeSerial

Λαμβάνει μια σειριακή ημερομηνία για τις καθορισμένες ώρες, λεπτά και δευτερόλεπτα

### Σύνταξη:



```
#include "vbcompat.bi"  
result = TimeSerial( hours, minutes, seconds )
```

Επιστρέφει μια σειριακή ημερομηνία που περιέχει το χρόνο που σχηματίζεται από τις τιμές στις παραμέτρους ωρών, λεπτών και δευτερολέπτων.

Η σειριακή ημερομηνία που επιστρέφεται δεν έχει ακέραιο μέρος.

Οι ώρες πρέπει να καθορίζονται στο εύρος 0-23

Ο μεταγλωττιστής δεν θα αναγνωρίσει αυτήν τη συνάρτηση, εκτός εάν περιλαμβάνεται το vbcompat.bi ή το datetime.bi.

## Παράδειγμα - timeserial.bas



```
1. #include "vbcompat.bi"  
2.  
3. Dim ds As Double = DateSerial(2005, 11, 28) +  
   TimeSerial(7, 30, 50)  
4.  
5. Print Format(ds, "yyyy/mm/dd hh:mm:ss")  
6.  
7. Sleep  
8. End
```

## Έξοδος:



```
2005-11-28 07:30:50
```

## DateValue

Επιστρέφει μια σειριακή ημερομηνία από μια συμβολοσειρά

### Σύνταξη:



```
#include "vbcompat.bi"  
result = DateValue( date_string )
```

Η συμβολοσειρά ημερομηνίας πρέπει να έχει τη μορφή που έχει οριστεί στις τοπικές ρυθμίσεις του λειτουργικού συστήματος.

Η DateValue (Date ()) θα λειτουργήσει σωστά μόνο εάν οι τοπικές ρυθμίσεις καθορίζουν την ίδια μορφή σύντομης ημερομηνίας της QB που χρησιμοποιείται (mm-dd-yyyy).

Εξετάστε το ενδεχόμενο να χρησιμοποιήσετε τη συνάρτηση Now στην έκφραση Fix (Now ()) για να λάβετε την τρέχουσα ημερομηνία ως σειρά ημερομηνίας.

Ο μεταγλωττιστής δεν θα αναγνωρίσει αυτήν τη συνάρτηση, εκτός εάν περιλαμβάνεται το vbcompat.bi ή το datetime.bi.

## Παράδειγμα - datevalue.bas



```
1. #include "vbcompat.bi"
2.
3. Dim As Integer v1, v2
4. Dim As String s1, s2
5.
6. Print "Enter first date: ";
7. Line Input s1
8.
9.
10. If IsDate( s1 ) = 0 Then
11.     Print "not a date"
12.     End
13. End If
14.
15. Print "Enter second date: ";
16. Line Input s2
17.
18. If IsDate( s2 ) = 0 Then
19.     Print "not a date"
20.     End
21. End If
22.
23. ' convert the strings to date serials
24. v1 = DateValue( s1 )
25. v2 = DateValue( s2 )
26.
27. Print "Number of days between dates is " &
28. Abs( v2 - v1 )
29.
30. Sleep
31. End
```

### Έξοδος:



```
Enter first date: 03-12-1977
Enter second date: 03-12-2001
Number of days between dates is 8766
```

## TimeValue

Επιστρέφει μια σειριακή ημερομηνία από μια συμβολοσειρά χρόνου.

### Σύνταξη:



```
#include "vbcompat.bi"  
result = TimeValue( timestring )
```

Η χρονική συμβολοσειρά πρέπει να έχει τη μορφή "23:59:59" ή "23:59:59PM"

Ο μεταγλωττιστής δεν θα αναγνωρίσει αυτήν τη συνάρτηση, εκτός εάν περιλαμβάνεται το vbcompat.bi ή το datetime.bi.

### Παράδειγμα - timevalue.bas



```
1. #include "vbcompat.bi"  
2.  
3. Dim ds As Double = TimeValue("07:12:28AM")  
4.  
5. Print Format(ds, "hh:mm:ss")  
6.  
7. Sleep  
8. End
```

### Έξοδος:



```
07:12:28
```

## Second

Επιστρέφει τα δευτερόλεπτα από μια σειριακή ημερομηνία.

### Σύνταξη:



```
#include "vbcompat.bi"  
result = Second( date_serial )
```

Ο μεταγλωττιστής δεν θα αναγνωρίσει αυτήν τη συνάρτηση, εκτός εάν περιλαμβάνεται το vbcompat.bi

### Παράδειγμα - second.bas



```
1.  #include "vbcompat.bi"  
2.  
3.  Dim ds As Double = DateSerial(2005, 11, 28) +  
4.  TimeSerial(7, 30, 50)  
5.  
6.  Print Format(ds, "yyyy/mm/dd hh:mm:ss ");  
   Second(ds)  
7.  
8.  Sleep  
9.  End  
10.
```

### Έξοδος:



```
2005-11-28 07:30:50 50
```

## Minute

Επιστρέφει τα λεπτά από μια σειριακή ημερομηνία.

### Σύνταξη:



```
#include "vbcompat.bi"  
result = Minute( date_serial )
```

Ο μεταγλωττιστής δεν θα αναγνωρίσει αυτήν τη συνάρτηση, εκτός εάν περιλαμβάνεται το vbcompat.bi

### Παράδειγμα - minute.bas



```
1.  #include "vbcompat.bi"  
2.  
3.  Dim ds As Double = DateSerial(2005, 11, 28) +  
4.  TimeSerial(7, 30, 50)  
5.  
6.  Print Format(ds, "yyyy/mm/dd hh:mm:ss ");  
   Minute(ds)  
7.  
8.  Sleep  
9.  End  
10.
```

### Έξοδος:



```
2005-11-28 07:30:50 30
```

## Hour

Επιστρέφει την ώρα από μια σειριακή ημερομηνία.

### Σύνταξη:



```
#include "vbcompat.bi"  
result = Hour( date_serial )
```

Ο μεταγλωττιστής δεν θα αναγνωρίσει αυτήν τη συνάρτηση, εκτός εάν περιλαμβάνεται το vbcompat.bi

### Παράδειγμα - hour.bas



```
1.  #include "vbcompat.bi"  
2.  
3.  Dim ds As Double = DateSerial(2005, 11, 28) +  
4.  TimeSerial(7, 30, 50)  
5.  
6.  Print Format(ds, "yyyy/mm/dd hh:mm:ss ");  
   Hour(ds)  
7.  
8.  Sleep  
9.  End  
10.
```

### Έξοδος:



```
2005-11-28 07:30:50 7
```



## Day

Επιστρέφει την ημέρα του μήνα από μια σειριακή ημερομηνία.

### Σύνταξη:



```
#include "vbcompat.bi"  
result = Day( date_serial )
```

Ο μεταγλωττιστής δεν θα αναγνωρίσει αυτήν τη συνάρτηση, εκτός εάν περιλαμβάνεται το vbcompat.bi

### Παράδειγμα - day.bas



```
1. #include "vbcompat.bi"  
2.  
3. Dim ds As Double = DateSerial(2005, 11, 28) +  
4. TimeSerial(7, 30, 50)  
5.  
6. Print Format(ds, "yyyy/mm/dd hh:mm:ss "); Day(ds)  
7.  
8. Sleep  
9. End  
10.
```

### Έξοδος:



```
2005-11-28 07:30:50 28
```

## Weekday

Επιστρέφει τον αριθμό των ημερών της εβδομάδας από μια σειριακή ημερομηνία.

### Σύνταξη:



```
#include "vbcompat.bi"  
result = Weekday( date_serial [, firstdayofweek ] )
```

Οι τιμές της ημέρας της εβδομάδας πρέπει να είναι στο εύρος 1-7, η σημασία της εξαρτάται από την παράμετρο `firstdayofweek`

Ο μεταγλωττιστής δεν θα αναγνωρίσει αυτήν τη συνάρτηση, εκτός εάν περιλαμβάνεται το `vbcompat.bi`

Το `firstdayofweek` είναι προαιρετικό, οι τιμές του είναι οι ακόλουθες:

<b>value</b>	<b>first day of week</b>	<b>constant</b>
omitted	sunday	
0	local settings	fbUseSystem
1	sunday	fbSunday
2	monday	fbMonday
3	tuesday	fbTuesday
4	wednesday	fbWednesday
5	thursday	fbThursday
6	friday	fbFriday
7	saturday	fbSaturday

## Παράδειγμα - weekday.bas



```
1. #include "vbcompat.bi"
2.
3. Dim a As Double = DateSerial (2005, 11, 28) +
   TimeSerial(7, 30, 50)
4.
5. Print Format(a, "yyyy/mm/dd hh:mm:ss ");
6. Weekday(a)
7.
8. Sleep
9. End
```

## Έξοδος:



```
2005-11-28 07:30:50 2
```

## Month

Επιστρέφει τον μήνα του έτους από μία σειριακή ημερομηνία.

## Σύνταξη:



```
#include "vbcompat.bi"
result = Month( date_serial )
```

Ο μεταγλωττιστής δεν θα αναγνωρίσει αυτήν τη συνάρτηση, εκτός εάν περιλαμβάνεται το vbcompat.bi

## Παράδειγμα - month.bas



```
1. #include "vbcompat.bi"
2.
3. Dim a As Double = DateSerial(2005,11,28) +
   TimeSerial(7,30,50)
4.
5. Print Format(a, "yyyy/mm/dd hh:mm:ss "); Month(a)
6.
7. Sleep
8. End
9.
```

### Έξοδος:



```
2005-11-28 07:30:50 11
```

## Year

Επιστρέφει το έτος από μια σειριακή ημερομηνία.

### Σύνταξη:



```
#include "vbcompat.bi"
result = Year( date_serial )
```

Ο μεταγλωττιστής δεν θα αναγνωρίσει αυτήν τη συνάρτηση, εκτός εάν περιλαμβάνεται το vbcompat.bi

## Παράδειγμα - year.bas



```
1. #include "vbcompat.bi"
2.
3. Dim a As Double = DateSerial (2005, 11, 28) +
   TimeSerial(7, 30, 50)
4.
5. Print Format(a, "yyyy/mm/dd hh:mm:ss "); Year(a)
6.
7. Sleep
8. End
```

## Έξοδος:



```
2005-11-28 07:30:50 2005
```

## DatePart

Επιστρέφει ένα μέρος μιας σειριακής ημερομηνίας.

## Σύνταξη:



```
#include "vbcompat.bi"
result = DatePart( interval, date_serial,
first_dayofWeek [, first_week_of_year ] )
```

Ο μεταγλωττιστής δεν θα αναγνωρίσει αυτήν τη συνάρτηση, εκτός εάν περιλαμβάνεται το vbcompat.bi

Συμβολοσειρά interval που υποδεικνύει ποιο μέρος της ημερομηνίας απαιτείται καθορίζεται ως εξής:

<b>value</b>	<b>interval</b>
yyyy	years
q	quarter(three months)
m	months
w	weekday
ww	week of the year
y	day of the year
d	day of the month
h	hours
n	minutes
s	seconds

first\_dayofweek

<b>value</b>	<b>first day of week</b>	<b>constant</b>
<b>omitted</b>	sunday	
0	local settings	fbUseSystem
1	sunday	fbSunday
2	monday	fbMonday
3	tuesday	fbTuesday
4	wednesday	fbWednesday
5	thursday	fbThursday
6	friday	fbFriday
7	saturday	fbSaturday

first\_weekofyear

value	first week of year	constant
0	local settings	fbUseSystem
1	January 1's week	fbFirstJan1
2	first weeks having 4 days in the year	fbFirstFourDays
3	first full week of year	fbFirstFullWeek

## Παράδειγμα - datepart.bas



```
1. #include "vbcompat.bi"
2.
3. Dim d As Double
4.
5. d = Now()
6.
7.
8. Print "Today is day " & DatePart( "y", d );
9. Print " in week " & DatePart( "ww", d );
10. Print " of the year " & DatePart( "yyyy", d )
11.
12. Sleep
    End
```

Έξοδος:



```
Today is day 214 in week 32 of the year 2021
```

## DateAdd

Επιστρέφει μια σειριακή ημερομηνία που αντιστοιχεί στη ληφθείσα ημερομηνία συν τον αριθμό διαστημάτων.

### Σύνταξη:



```
#include "vbcompat.bi"  
result = DateAdd( interval, number, date_serial )
```

Ο μεταγλωττιστής δεν θα αναγνωρίσει αυτήν τη συνάρτηση, εκτός εάν περιλαμβάνεται το vbcompat.bi

Το διάστημα (interval) καθορίζεται ως εξής:

#### **value interval**

yyyy	years
q	quarter(three months)
m	months
ww	weeks
d,w,y	days
h	hours
n	minutes
s	seconds



## Παράδειγμα - dateadd.bas



```
1. #include "vbcompat.bi"
2.
3. Const fmt = "dddd tttt"
4. Dim d As Double
5. d = Now()
6.
7. Print "1 hour from now is ";
8. Print Format( DateAdd( "h", 1, d ), fmt )
9.
10.
11. Print "1 day from now is ";
12. Print Format( DateAdd( "d", 1, d ), fmt )
13.
14. Print "1 week from now is ";
15. Print Format( DateAdd( "ww", 1, d ), fmt )
16.
17. Print "1 month from now is ";
18. Print Format( DateAdd( "m", 1, d ), fmt )
19.
20. Sleep
    End
```

### Έξοδος:



```
1 hour from now is 02-Aug-21 9:51:42 PM
1 day from now is 03-Aug-21 8:51:42 PM
1 week from now is 09-Aug-21 8:51:42 PM
1 month from now is 02-Sep-21 8:51:42 PM
```

## DateDiff

Επιστρέφει την διαφορά μεταξύ δύο σειριακών ημερομηνιών

### Σύνταξη:



```
#include "vbcompat.bi"
result = DateDiff( interval, date_serial1,
date_serial2 [,
                    firstdayofweek [, firstweekofyear
]] )
```

Ο μεταγλωττιστής δεν θα αναγνωρίσει αυτήν τη συνάρτηση, εκτός εάν περιλαμβάνεται το vbcompat.bi

Το διάστημα (interval) καθορίζεται ως εξής:

**value interval**

yyyy years  
q quarter(three  
months)  
m months  
ww weeks  
d,w,y days  
h hours  
n minutes  
s seconds

firstdayofweek Επηρεάζει την καταμέτρηση όταν χρησιμοποιείται το διάστημα "ww".

<b>value</b>	<b>first day of week</b>	<b>constant</b>
omitted	sunday	
0	local settings	fbUseSystem
1	sunday	fbSunday
2	monday	fbMonday
3	tuesday	fbTuesday
4	wednesday	fbWednesday
5	thursday	fbThursday
6	friday	fbFriday
7	saturday	fbSaturday

Το first\_weekofyear καθορίζει ποιο έτος (προηγούμενο ή επόμενο) θα πρέπει να περιλαμβάνει την εβδομάδα που καλύπτει το τέλος ενός έτους και την αρχή του επόμενου.

value	first week of year	constant
0	local settings	fbUseSystem
1	January 1's week	fbFirstJan1
2	first weeks having 4 days in the year	fbFirstFourDays
3	first full week of year	fbFirstFullWeek

## Παράδειγμα - datediff.bas



```

1.  #include "vbcompat.bi"
2.
3.  Dim s As String, d1 As Double, d2 As Double
4.
5.  Line Input "Enter your birthday: ", s
6.
7.
8.  If IsDate( s ) Then
9.      d1 = DateValue( s )
10.     d2 = Now()
11.
12.     Print "You are " & DateDiff( "yyy", d1, d2 ) &
13.     " years old."
14.     Print "You are " & DateDiff( "d", d1, d2 ) & "
15.     days old."
16.     Print "You are " & DateDiff( "s", d1, d2 ) & "
17.     seconds old."
18.
19. Else
20.     Print "Invalid date"
21.
22. End If
23.
24. Sleep
25. End

```

## Έξοδος:



```

Enter your birthday: 04-12-1977
You are 44 years old.
You are 15947 days old.
You are 1377896475 seconds old.

```

## IsDate

Ελέγχει αν μια συμβολοσειρά μπορεί να μετατραπεί σε σειριακή ημερομηνία.

### Σύνταξη:



```
#include "vbcompat.bi"  
result = IsDate( stringdate )
```

Ο μεταγλωττιστής δεν θα αναγνωρίσει αυτήν τη συνάρτηση, εκτός εάν περιλαμβάνεται το vbcompat.bi ή το datetime.bi. Επιστρέφει μη μηδέν (-1) εάν η συμβολοσειρά ημερομηνίας μπορεί να μετατραπεί σε σειριακή ημερομηνία, διαφορετικά επιστρέφει μηδέν (0).

### Παράδειγμα - isdate.bas



```
1. #include "vbcompat.bi"  
2.  
3. Dim s As String, d As Integer  
4.  
5. Do  
6.   Print  
7.   Print "Enter a date: "  
8.  
9.  
10.  Line Input s  
11.  
12.  If s = "" Then Exit Do  
13.  
14.  If IsDate( s ) = 0 Then  
15.    Print "'"; s; "' is not a valid date"  
16.  Else  
17.    d = DateValue( s )  
18.    Print "year = "; Year( d )  
19.    Print "month = "; Month( d )  
20.    Print "day = "; Day( d )  
21.  End If  
22.  
23. Loop  
24. End
```

## Έξοδος:



```
Enter a date:  
12-04-1977  
year = 1977  
month = 4  
day = 12
```

## MonthName

Επιστρέφει το όνομα ενός μήνα από την ολοκληρωμένη αναπαράστασή του.

## Σύνταξη:



```
#include "vbcompat.bi"  
result = MonthName( month_number [, abbreviate ] )
```

`month_number`

Τον αριθμό του μήνα του έτους - 1: Ιανουάριος έως 12: Δεκέμβριος

`abbreviate`

Σημαία για να υποδείξει ότι το όνομα πρέπει να είναι συντομευμένο.

Εάν η σημαία είναι αληθής, επιστρέφεται η συντομογραφία του μήνα. Εάν παραλειφθεί ή είναι ψευδής, επιστρέφεται ολόκληρο το όνομα.

Ο μεταγλωττιστής δεν θα αναγνωρίσει αυτήν τη συνάρτηση, εκτός εάν περιλαμβάνεται το `vbcompat.bi` ή το `datetime.bi`

## Παράδειγμα - monthname.bas



```
1. #include "vbcompat.bi"
2.
3. Dim ds As Double = DateSerial(2005, 11, 28) +
   TimeSerial(7, 30, 50)
4.
5. Print Format(ds, "yyyy/mm/dd hh:mm:ss ");
6.   MonthName(Month(ds))
7.
8. Sleep
9. End
```

## Έξοδος:



```
2005-11-28 07:30:50 November
```

## WeekdayName

Επιστρέφει το όνομα μιας ημέρας εβδομάδας από την ολοκληρωμένη αναπαράστασή του.

## Σύνταξη:



```
#include "vbcompat.bi"
result = WeekdayName( weekday [, abbreviate [,
firstdayofweek ] ] )
```

Ο μεταγλωττιστής δεν θα αναγνωρίσει αυτήν τη συνάρτηση, εκτός εάν περιλαμβάνεται το vbcompat.bi

Επιστρέφει το όνομα της ημέρας της γλώσσας του τοπικού λειτουργικού συστήματος από την τιμή της εβδομάδας 1 έως 7

Ο τρόπος ερμηνείας της ημέρας της εβδομάδας εξαρτάται από την παράμετρο firstdayofweek.

Εάν η σημαία είναι αληθής, επιστρέφεται μια συντομογραφία 3 γραμμάτων, εάν είναι ψευδής ή παραλειφθεί, επιστρέφεται ολόκληρο το όνομα.

Το `firstdayofweek` είναι μια προαιρετική παράμετρος που καθορίζεται ως εξής:

<b>value</b>	<b>first day of week</b>	<b>constant</b>
omitted	sunday	
0	local settings	<code>fbUseSystem</code>
1	sunday	<code>fbSunday</code>
2	monday	<code>fbMonday</code>
3	tuesday	<code>fbTuesday</code>
4	wednesday	<code>fbWednesday</code>
5	thursday	<code>fbThursday</code>
6	friday	<code>fbFriday</code>
7	saturday	<code>fbSaturday</code>

## Παράδειγμα - `weekdayname.bas`



```
1. #include "vbcompat.bi"
2.
3. Dim a As Double = DateSerial(2005, 11, 28) +
   TimeSerial(7, 30, 50)
4.
5. Print Format(a, "yyyy/mm/dd hh:mm:ss ");
6. WeekdayName(Weekday(a))
7.
8. Sleep
9. End
```

Έξοδος:



```
2005-11-28 07:30:50 Monday
```

## Date

Επιστρέφει την ημερομηνία ως συμβολοσειρά.

Σύνταξη:



result = Date

## Παράδειγμα - date.bas



```
1. | Print Date ' prints the current date  
2.  
3. | Sleep  
4. | End
```

Έξοδος:



```
08-03-2021
```

Η ημερομηνία είναι στην μορφή, mm-dd-yyyy



## Time

Επιστρέφει την ώρα ως συμβολοσειρά.

### Σύνταξη:



result = Time

### Παράδειγμα - time.bas



```
1. Print "the current time is: "; Time
2.
3. Sleep
4. End
```

### Έξοδος:



```
the current time is: 14:04:31
```

## SetDate

Ορίζει την ημερομηνία του συστήματος

### Σύνταξη:



result = SetDate( newdate )

### Παράδειγμα:



```
1. Dim m As String, d As String, y As String
2. m = "03" 'march
3. d = "13" 'the 13th
4. y = "1994" 'good ol' days
5. SetDate m + "/" + d + "/" + y
```

Η ημερομηνία μπορεί να είναι μία από τις ακόλουθες μορφές: "mm-dd-yy", "mm-dd-yyyy", "mm/dd/yy", ή "mm/dd/yyyy" όπου m ένα ψηφίο του μήνα, d ψηφίο της ημέρας και y ψηφίο του έτους.

## SetTime

Ορίζει την ώρα του συστήματος.

### Σύνταξη:



```
result = SetTime( newtime )
```

### Παράδειγμα:



```
1. SetTime "1:20:30"  
2.  
3. Sleep  
4. End
```

Η νέα ώρα μπορεί να είναι της μορφής "hh:mm:ss", "hh:mm", ή "hh", όπου h το ψηφίο της ώρας, m ψηφίο λεπτών και s ψηφίο δευτερολέπτων.

## Timer

Επιστρέφει το χρονικό διάστημα που έχει περάσει από ένα στατικό σημείο αναφοράς

### Σύνταξη:



```
result = Timer
```

## Παράδειγμα:



```
1. '' Example of using TIMER function
2. '' Note: see text about correct waiting
3. '' strategies
4. Dim Start As Double
5. Print "Wait 2.5 seconds."
6. Start = Timer
7. Do
8.     Sleep 1, 1
9. Loop Until (Timer - Start) > 2.5
10. Print "Done."
```

# Συναρτήσεις διαχείρισης λαθών - Error Handling Functions

## Erl

Λειτουργία χειρισμού σφάλματος για επιστροφή της γραμμής όπου παρουσιάστηκε το σφάλμα.

### Σύνταξη:



result = Erl

### Παράδειγμα - erl.bas



```
1. ' compile with -lang fblite or qb
2.
3. #lang "fblite"
4.
5. ' note: compilation with '-ex' option is required
6.
7.
8. On Error Goto ErrorHandler
9.
10. ' Generate an explicit error
11. Error 100
12.
13. End
14.
15. ErrorHandler:
16. Dim num As Integer = Err
17. Print "Error "; num; " on line "; Erl
18. Sleep
    Resume Next
```

### Έξοδος:



```
Error 100 on line 10
```

## Erfn

Μια συνάρτηση αναφοράς σφάλματος που επιστρέφει έναν δείκτη στο όνομα της συνάρτησης.

### Σύνταξη:



```
result = Erfn ( )
```

Επιστρέφει έναν δείκτη στη συμβολοσειρά που προσδιορίζει τη συνάρτηση όπου παρουσιάστηκε το σφάλμα.

Επιστρέφει NULL εάν η πηγή δεν είναι μεταγλωττισμένη με την επιλογή μεταγλωττιστή -exx.

### Παράδειγμα - `erfn.bas`



```
1. ' test.bas
2. ' compile with fbc -exx -lang fblite test.bas
3.
4. #lang "fblite"
5.
6. Sub Generate_Error
7.   On Error Goto Handler
8.   Error 1000
9.   Exit Sub
10. Handler:
11.   Print "Error Function: "; *Erfn()
12.   Resume Next
13. End Sub
14.
15. Generate_Error
```

## Ernm

Επιστρέφει έναν δείκτη στη συμβολοσειρά που προσδιορίζει τη μονάδα όπου παρουσιάστηκε το σφάλμα.

Επιστρέφει NULL εάν η πηγή δεν είναι μεταγλωττισμένη με την επιλογή μεταγλωττιστή -exx.

### Σύνταξη:



result = Ernm ( )

### Παράδειγμα - ermnbas



```
1. ' test.bas
2. ' compile with fbc -exx -lang fblite test.bas
3.
4. #lang "fblite"
5.
6. Sub Generate_Error
7.   On Error Goto Handler
8.   Error 1000
9.   Exit Sub
10. Handler:
11.   Print "Error Module : "; *Ernm()
12.   Resume Next
13. End Sub
14.
15. Generate_Error
```

### Έξοδος:



```
Error Module : C:\Users\User\FreeBASIC\Code\
ermnbas
```

## Err

Επιστρέφει ή ορίζει τον αριθμό σφάλματος χρόνου εκτέλεσης

### Σύνταξη:



result = Err( )

ή

Err = number

### Παράδειγμα - err.bas



```
1.  ' Compile with -lang fblite or qb
2.
3.  #lang "fblite"
4.
5.  On Error Goto Error_Handler
6.  Error 150
7.  End
8.
9.
10. Error_Handler:
11.     n = Err()
12.     Print "Error #"; n
13.     Resume Next
```

### Έξοδος:



```
Error # 150
```

## Error

Δημιουργεί ένα σφάλμα χρησιμοποιώντας έναν αριθμό σφάλματος.

### Σύνταξη:



Error number

Οι αριθμοί σφάλματος της FreeBASIC είναι:

- 0 No error
- 1 Illegal function call
- 2 File not found signal
- 3 File I/O error
- 4 Out of memory
- 5 Illegal resume
- 6 Out of bounds array access
- 7 Null Pointer Access
- 8 No privileges
- 9 interrupted signal
- 10 illegal instruction signal
- 11 floating point error signal
- 12 segmentation violation signal
- 13 Termination request signal
- 14 abnormal termination signal
- 15 quit request signal
- 16 return without gosub
- 17 end of file



## On Error

Το On Error ενεργοποιεί μια μετάβαση σε έναν χειριστή σφαλμάτων όταν εμφανιστεί ένα σφάλμα. Τέτοια σφάλματα μπορεί να προκληθούν από ενσωματωμένες προτάσεις όπως το Open ή όταν χρησιμοποιείται η δήλωση σφάλματος Error.

### Σύνταξη:



On [Local] Error Goto label

Το On Local Error μπορεί να χρησιμοποιηθεί για τον καθορισμό ενός τοπικού χειριστή σφαλμάτων μέσα σε μια διαδικασία.

### Παράδειγμα - onerror.bas



```
1. ' Compile with QB (-lang qb) dialect
2.
3. '$lang: "qb"
4.
5. On Error Goto errorhandler
6. Error 24 ' simulate an error
7. Print "this message will not be seen"
8.
9.
10. errorhandler:
11. n = Err
12. Print "Error #"; n; "!"
13. End
```

### Έξοδος:



```
Error # 24!
```

## Resume

Το Resume χρησιμοποιείται στον παραδοσιακό μηχανισμό διαχείρισης σφαλμάτων QB εντός ενός χειριστή σφαλμάτων (που ονομάζεται On Error) για να επιστρέψει την εκτέλεση στη γραμμή που προκάλεσε το σφάλμα. Συνήθως αυτό χρησιμοποιείται αφού χειριστεί το σφάλμα για να δοκιμάσει ξανά την προηγούμενη λανθασμένη λειτουργία με διορθωμένα δεδομένα.

### Σύνταξη:



Resume

### Παράδειγμα - resume.bas



```
1.  ' Compile with -lang fblite or qb
2.
3.  #lang "fblite"
4.
5.  Dim As Single i, j
6.
7.  On Error Goto ErrorHandler
8.
9.
10. i = 0
11. j = 1 / i ' this line causes a divide-by-zero
12.          ' error on the first try; execution
13.          ' jumps to
14.          ' ErrorHandler label
15.
16. Print j ' after the value of i is corrected,
17.          ' prints 0.5
18.
19. End ' end the program so that execution does not
20.     ' fall through to the error handler again
21.
22. ErrorHandler:
23.
24.
25. i = 2
26. Resume ' execution jumps back to 'j = 1 / i'
27.        'line, which does not cause an error this
28.        'time
```

## Resume Next

Το Resume Next χρησιμοποιείται στον παραδοσιακό μηχανισμό χειρισμού σφαλμάτων QB εντός ενός χειριστή σφαλμάτων (που ονομάζεται On Error) για να επιστρέψει την εκτέλεση στη γραμμή μετά από αυτήν που προκάλεσε το σφάλμα. Συνήθως αυτό χρησιμοποιείται για να αποφευχθεί η εκτέλεση της ίδιας γραμμής και η πρόκληση του σφάλματος ξανά.

### Σύνταξη:



Resume Next

### Παράδειγμα - resumenext.bas



```
1.  '' Compile with -lang fblite or qb
2.
3.  #lang "fblite"
4.
5.  Dim As Single i, j
6.
7.
8.  On Error Goto ErrorHandler
9.
10. i = 0
11. j = 5
12. j = 1 / i ' this line causes a divide-by-zero
13.           'error; execution jumps to ErrorHandler
14.           'label
15.
16. Print "ending..."
17.
18. End ' end the program so that execution does not
19.     'fall through to the error handler again
20.
21.
22. ErrorHandler:
23.
24. Resume Next ' execution jumps to 'Print
25.             '"ending..."' line, but j is now in
                'an undefined state
```

# **Συναρτήσεις εισόδου/εξόδου αρχείων - File IO Functions**

Δηλώσεις και διαδικασίες για εργασία με αρχεία και συσκευές.

Αυτές οι δηλώσεις και διαδικασίες παρέχουν δυνατότητες εισόδου/εξόδου αρχείου και συσκευής. Οι λεγόμενοι αριθμοί αρχείων μπορούν να συνδεθούν με αρχεία ή συσκευές, τα οποία μπορούν να διαβαστούν ή να γραφτούν χρησιμοποιώντας δεδομένα μορφοποιημένα (λειτουργία κειμένου) ή μη μορφοποιημένα (δυναμική λειτουργία).

Σε δυναμική λειτουργία, τα αρχεία και οι συσκευές μπορούν να διαβαστούν από ή να γραφτούν σε αυθαίρετες τοποθεσίες. Για εφαρμογές πολλαπλών νημάτων, αρχεία και συσκευές μπορούν επίσης να κλειδωθούν.

Σε αυτό το κεφάλαιο θα δούμε:

## **Άνοιγμα αρχείων ή συσκευών**

Διαδικασίες και άλλες λέξεις -κλειδιά που παρέχουν πρόσβαση ανάγνωσης ή εγγραφής σε αρχείο ή συσκευή.

## **Ανάγνωση και εγγραφή σε αρχεία ή συσκευές**

Διαδικασίες που διαβάζουν και γράφουν δεδομένα σε ανοιχτό αρχείο ή συσκευή.

## **Θέση αρχείου και άλλες πληροφορίες**

Διαδικασίες που καθορίζουν πού θα πραγματοποιηθεί η ανάγνωση και η γραφή σε ένα ανοιχτό αρχείο.

# Άνοιγμα αρχείων ή συσκευών

## FreeFile

Επιστρέφει τον αριθμό του επόμενου δωρεάν αριθμού αρχείου με έγκυρες τιμές 1 σε 255, ή 0 αν έχουν ανοίξει ήδη 255 αρχεία. Αυτή η τιμή είναι ένα απαιτούμενο όρισμα για να ανοίξετε ένα αρχείο. Το FreeFile είναι χρήσιμο όταν ανοίγετε αρχεία σε πολύπλοκα προγράμματα όπου ο προγραμματιστής δεν μπορεί να παρακολουθεί τους χρησιμοποιούμενους αριθμούς αρχείων.

Βεβαιωθείτε ότι κλείνετε πάντα τα αρχεία όταν δεν είναι πλέον απαραίτητα, διαφορετικά θα λάβετε διαρροή αριθμού αρχείου και δεν θα μπορείτε να ανοίξετε άλλα αρχεία αφού εξαντληθούν 255 αριθμοί αρχείων ενώ εκτελείται το πρόγραμμά σας.

Το FreeFile θα επιστρέφει πάντα τον μικρότερο ελεύθερο αριθμό αρχείου. Ο αριθμός αρχείου που επιστρέφεται από το FreeFile δεν θα αλλάξει μέχρι να ανοίξει αυτός ο αριθμός αρχείου ή μέχρι να κλείσει ένας μικρότερος αριθμός αρχείου:

- Για το λόγο αυτό, είναι συνετό να χρησιμοποιήσετε το FreeFile αμέσως πριν από το αντίστοιχο Open, για να διασφαλίσετε ότι ο ίδιος αριθμός αρχείου δεν επιστρέφεται και ανοίγει αλλού πρώτα.

- Σε περίπτωση πιθανής σύγκρουσης με άλλα νήματα, αυτή η αδιάσπαστη ακολουθία «FreeFile ... Open» πρέπει επιπλέον να θεωρηθεί ως κρίσιμο τμήμα του κώδικα και ως εκ τούτου πρέπει να προστατεύεται, για παράδειγμα με αμοιβαίο αποκλεισμό (χρησιμοποιώντας κλείδωμα mutex).

## Σύνταξη:



result = FreeFile

## Παράδειγμα - freefile.bas



```
1. ' Create a string and fill it.
2. Dim buffer As String, f As Long
3. buffer = "Hello World within a file."
4.
5. ' Find the first free file number.
6. f = FreeFile
7.
8. ' Open the file "file.ext" for binary usage,
9. ' using the file number "f".
10. Open "file.ext" For Binary As #f
11.
12. ' Place our string inside the file,
13. ' using file number "f".
14. Put #f, , buffer
15.
16.
17. ' Close the file.
18. Close #f
19.
20. ' End the program. (Check the file "file.ext"
21. ' upon running to see the output.)
22. End
```

### Ανάλυση:

Το παραπάνω παράδειγμα γράφει σε δυαδική μορφή ένα κείμενο σε ένα αρχείο.

Στην γραμμή 6 παίρνουμε το πρώτο διαθέσιμο αριθμό ελεύθερου αρχείου. Αυτόν τον αριθμό τον χρησιμοποιούμε στην δήλωση `Open` της γραμμής 10.

Στην γραμμή 14 γράφουμε στο αρχείο και στην γραμμή 17 κλείνουμε το αρχείο.

Στον τρέχοντα φάκελο από τον οποίο τρέχουμε το πρόγραμμα δημιουργείται αν δεν υπάρχει το αρχείο `file.ext`

## Open

Ανοίγει ένα αρχείο δίσκου για ανάγνωση ή εγγραφή χρησιμοποιώντας λειτουργίες αρχείων

### Σύνταξη:



```
result = Open( filename[,] For {Input|Output|
Append}[,] As filenumber )
ή
result = Open( filename[,] For Binary[,] Access
{Read|Write}[,] As filenumber )
ή
result = Open( filename[,] For Random[,] Access
{Read|Write}[,] As filenumber [,] Len =
record_length )
ή
Open filename For {Input|Output|Append} As
filenumber
ή
Open filename For Binary Access {Read|Write} As
filenumber
ή
Open filename For Random Access {Read|Write} As
filenumber [Len = record_length]
```

### Παράμετροι:

#### filename

Μια τιμή συμβολοσειράς του ονόματος του αρχείου δίσκου για να ανοίξει. Οι σχετικές διαδρομές αρχείων είναι σχετικές με τον τρέχοντα κατάλογο (βλ. CurDir).

#### encoding\_type

Η κωδικοποίηση που θα χρησιμοποιείται κατά την ανάγνωση ή τη γραφή κειμένου, μπορεί να είναι μία από τις εξής:

- Encoding "ascii" (ASCII encoding χρησιμοποιείται, default)
- Encoding "utf8" (8-bit Unicode encoding χρησιμοποιείται)

- Encoding "utf16" (16-bit Unicode encoding χρησιμοποιείται)
- Encoding "utf32" (32-bit Unicode encoding χρησιμοποιείται)

#### access\_type

Ο τύπος πρόσβασης που ζητείται από τη διαδικασία κλήσης. Πρόσβαση [Read] [Write] (μπορεί να χρησιμοποιηθεί τόσο η ανάγνωση όσο και η εγγραφή, η οποία είναι η προεπιλογή)

#### lock\_type

Επιβάλλει περιορισμούς στην πρόσβαση αρχείων δίσκου από άλλες διεργασίες (νήματα ή προγράμματα), μπορεί να είναι είτε:

Shared (το αρχείο μπορεί να έχει ελεύθερη πρόσβαση από άλλες διαδικασίες)

Lock [Read] [Write] (η πρόσβαση τόσο για ανάγνωση όσο και για εγγραφή μπορεί να αποκλειστεί σε άλλες διαδικασίες)

#### filenumber

Ένας διαθέσιμος αριθμός αρχείου για σύνδεση στο αρχείο δίσκου, ο οποίος μπορεί να βρεθεί με το FreeFile.

#### record\_length

Το μέγεθος, σε byte, κάθε εγγραφής που διαβάζεται ή γράφεται στο αρχείο δίσκου. Η προεπιλογή είναι 128.

### **Return Value**

Στην πρώτη χρήση, το Open () επιστρέφει μια τιμή 32 bit Long: μηδέν (0) σε επιτυχία και έναν μη μηδενικό κωδικό σφάλματος διαφορετικά.

### **Περιγραφή:**

Ανοίγει ένα αρχείο δίσκου για ανάγνωση ή/και εγγραφή. Ο αριθμός αρχείου file\_num είναι δεσμευμένος στο αρχείο στο δίσκο, για χρήση σε επόμενες λειτουργίες αρχείων, όπως Input # και Lock. Ο επόμενος διαθέσιμος αριθμός αρχείου μπορεί να ανακτηθεί με το FreeFile.



Οι λειτουργίες αρχείων Input, Output και Append ανοίγουν αρχεία δίσκου για διαδοχικά εισερχόμενα/εξερχόμενα κείμενα, χρήσιμα για την ανάγνωση ή εγγραφή αρχείων απλού κειμένου:

- Όταν καθορίζεται η λειτουργία εισόδου, μπορούν να χρησιμοποιηθούν μόνο λειτουργίες ανάγνωσης αρχείων, όπως η Line Input # και η Get #. Εάν το αρχείο δίσκου δεν υπάρχει, θα εμφανιστεί σφάλμα χρόνου εκτέλεσης.
- Η λειτουργία Προσάρτησης (Append) καθορίζει ότι μπορούν να χρησιμοποιηθούν μόνο λειτουργίες γραφής, όπως Print # και Put #. Οι λειτουργίες εγγραφής θα πραγματοποιηθούν στο τέλος του αρχείου δίσκου εάν υπάρχει, διατηρώντας τα υπάρχοντα δεδομένα.
- Η λειτουργία εξόδου μοιάζει με τη λειτουργία προσάρτησης, με την εξαίρεση ότι εάν το αρχείο υπάρχει, τα περιεχόμενά του διαγράφονται και το μήκος του μηδενίζεται πριν από τη σύνταξη.

Οι λειτουργίες αρχείων εισόδου (Input), εξόδου (Output) και προσθήκης (Append) επιτρέπουν επίσης την επιλογή μιας κωδικοποίησης χαρακτήρων που θα χρησιμοποιείται κατά την ανάγνωση ή την εγγραφή κειμένου στο αρχείο δίσκου. Μπορεί να καθορισθεί ASCII ή κωδικοποίηση Unicode (δείτε την περιγραφή της παραμέτρου encoding\_type παραπάνω).

Οι λειτουργίες δυαδικού (Binary) και τυχαίου (Random) αρχείου ανοίγουν αρχεία δίσκου για ανάγνωση ή εγγραφή τυχαίων μεγέθους δυαδικών δεδομένων με τυχαία πρόσβαση:

Η λειτουργία δυαδικού (Binary) αρχείου επιτρέπει την ανάγνωση και εγγραφή απλών τιμών τύπου δεδομένων, όπως Byte ή LongInt, χρησιμοποιώντας δυαδικές πράξεις ανάγνωσης ή εγγραφής, όπως Get #. Το LOC και το Seek είναι μεταξύ των διαδικασιών που χρησιμοποιούνται για την ανάγνωση και εγγραφή σε αυθαίρετες θέσεις στο αρχείο δίσκου.

Η τυχαία λειτουργία αρχείου (Random) είναι παρόμοια με το δυαδικό (Binary), εκτός από το ότι οι λειτουργίες εισόδου/εξόδου αρχείου χρησιμοποιούν πάντα ένα σταθερό μέγεθος δεδομένων κατά την ανάγνωση ή τη γραφή.

Από προεπιλογή, οι λειτουργίες δυαδικού και τυχαίου αρχείου επιτρέπουν τόσο τις λειτουργίες ανάγνωσης όσο και εγγραφής στο αρχείο δίσκου που έχει ανοίξει, αλλά αυτό μπορεί να αλλάξει καθορίζοντας έναν τύπο πρόσβασης (ανατρέξτε στην περιγραφή της παραμέτρου `access_type` παραπάνω). Κατά το άνοιγμα ενός υπάρχοντος αρχείου σε λειτουργία δυαδικού ή τυχαίου αρχείου και με τον μόνο τύπο πρόσβασης εγγραφής καθορισμένο, όλα τα δεδομένα του αρχείου διαγράφονται προηγουμένως κατά το άνοιγμα, πριν από οποιαδήποτε άλλη λειτουργία.

Για οποιαδήποτε λειτουργία αρχείου, η πρόσβαση στο αρχείο δίσκου που έχει ανοίξει μπορεί να περιοριστεί ή να παραχωρηθεί σε άλλα νήματα ή προγράμματα καθορίζοντας έναν τύπο κλειδώματος (δείτε την περιγραφή για την παράμετρο `lock_type` παραπάνω). Εάν δεν έχει καθοριστεί τύπος κλειδώματος, άλλα νήματα του τρέχοντος προγράμματος μπορούν να ανοίξουν ελεύθερα το αρχείο δίσκου (Shared), ενώ άλλα προγράμματα δεν μπορούν (Lock Read Write). Το κλείδωμα και το ξεκλείδωμα μπορούν να χρησιμοποιηθούν για τον προσωρινό περιορισμό της πρόσβασης σε μέρη ενός αρχείου.

Ο κωδικός σφάλματος που επιστρέφεται από το `Open` μπορεί να ελεγχθεί χρησιμοποιώντας το `Err` στην επόμενη γραμμή. Η έκδοση λειτουργίας του `Open` επιστρέφει απευθείας τον κωδικό σφάλματος ως `Long 32 bit`.

## Παράδειγμα - open.bas



```
1. ' Create a string and fill it.
2. Dim buffer As String, f As Long
3. buffer = "Hello World within a file."
4.
5. ' Find the first free file number.
6. f = FreeFile
7.
8. ' Open the file "file.ext" for binary usage,
9. ' using the file number "f".
10. Open "file.ext" For Binary As #f
11.
12. ' Place our string inside the file,
13. ' using file number "f".
14. Put #f, , buffer
15.
16.
17. ' Close the file.
18. Close #f
19.
20. ' End the program. (Check the file "file.ext"
21. ' upon running to see the output.)
22. End
```

Το παράδειγμα είναι το ίδιο με το free.bas.

## Open Com

Ανοίγει μια σειριακή θύρα για είσοδο και έξοδο.

### Σύνταξη:



result = Open Com( options[,] As[#] filename )

## Παράμετροι:

### options

Μια συμβολοσειρά που περιέχει επιλογές που χρησιμοποιούνται για τον έλεγχο της θύρας.

### filenumber

Ο αριθμός αρχείου για σύνδεση με τη θύρα.

## Επιστρεφόμενη τιμή:

Το Open Com () επιστρέφει ένα 32 bit Long: ένα μηδέν (0) στην επιτυχία και έναν μη μηδενικό κωδικό σφάλματος διαφορετικά.

## Περιγραφή:

Αυτή η εντολή ανοίγει μια σειριακή θύρα του υπολογιστή, επιτρέποντας την αποστολή και λήψη δεδομένων χρησιμοποιώντας τις κανονικές εντολές αρχείων όπως Print #, Input #, Get #, ...

Η κύρια παράμετρος είναι μια συμβολοσειρά που περιγράφει, τουλάχιστον, ποια θύρα επικοινωνίας πρέπει να ανοίξει. Έχει τη μορφή:

```
"Comn: [ baudrate ][ , [ parity ][ , [ data_bits ][ , [ stop_bits ]  
[ , [ extended_options ]]]]"
```

όπου:

**n**, θύρα Com για άνοιγμα. "1", "2", "3", "4" κλπ. Ορισμένες πλατφόρμες θα υποστηρίζουν περισσότερες σειριακές θύρες ανάλογα με τον τρόπο διαμόρφωσης του λειτουργικού συστήματος. Όπου το n δεν δίνεται, το "COM:" θα χαρτογραφηθεί στο "COM1:", εκτός από το Linux όπου "COM:" αντιστοιχεί στο "/dev/modem"

**baudrate**, "300" (προκαθορισμένο), "1200", ..., κτλ.

**parity**, "N" (none), "E" (even, default), "O" (odd), "S" (space), "M" (mark), "PE" (QB-quirk: checked, even parity)

**data\_bits**, "5", "6", "7" (προκαθορισμένο) or "8".

**stop\_bits**, "1", "1.5" ή "2".

**extended\_options:**

<b>Option</b>	<b>Action</b>
'CSn'	Set the CTS duration (in ms) (n>=0), 0 = turn off, default = 1000
'DSn'	Set the DSR duration (in ms) (n>=0), 0 = turn off, default = 1000
'CDn'	Set the Carrier Detect duration (in ms) (n>=0), 0 = turn off
'OPn'	Set the 'Open Timeout' (in ms) (n>=0), 0 = turn off
'TBn'	Set the 'Transmit Buffer' size (n>=0), 0 = default, depends on platform
'RBn'	Set the 'Receive Buffer' size (n>=0), 0 = default, depends on platform
'RS'	Suppress RTS detection
'LF'	Communicate in ASCII mode (add LF to every CR) - Win32 doesn't support this one
'ASC'	same as 'LF'
'BIN'	The opposite of LF and it'll always work
'PE'	Enable 'Parity' check
'DT'	Keep DTR enabled after CLOSE
'FE'	Discard invalid character on error
'ME'	Ignore all errors
'IRn'	IRQ number for COM (only supported (?) on DOS)

Όλα τα στοιχεία εκτός από τη θύρα COM είναι προαιρετικά. Η σειρά baudrate, parity, data\_bits, stop\_bits είναι σταθερή. Κάθε σταθερό στοιχείο που παραλείπεται (baudrate, κλπ ...) πρέπει να είναι κενό.

Ο κωδικός σφάλματος που επιστρέφεται από το Open Com μπορεί να ελεγχθεί χρησιμοποιώντας το Err στην επόμενη γραμμή. Η έκδοση λειτουργίας του Open Com επιστρέφει απευθείας τον κωδικό σφάλματος ως Long 32 bit.

## Παράδειγμα:



1. | `Open Com "COM1:9600,N,,2" As 1`

Ανοίγει το COM1 με 9600 baud, no parity, 7 data bits και 2 stop bits.

## Παράδειγμα:



1. | `Open Com "COM1:115200" As 1`

Ανοίγει το COM1 με 115200 baud, "even" parity, 7 data bits και 1 stop bits.

## Διαφορές στις πλατφόρμες:

Στην πλατφόρμα των Windows "COM:" αντιστοιχεί στο "COM1:"

Στην πλατφόρμα Linux

"COM:" maps to "/dev/modem"

"COM1:" maps to "/dev/ttyS0"

"COM2:" maps to "/dev/ttyS1", κ.λπ

"/dev/xyz:" maps to "/dev/xyz", κ.λπ

Το σειριακό πρόγραμμα οδήγησης DOS είναι πειραματικό και μπορεί να έχει πρόσβαση στις θύρες COM 1 έως 4

Χρησιμοποιεί τα ακόλουθα βασικά io και IRQ's ως προεπιλογή:

COM1 - & h3f8 - IRQ4

COM2 - & h2f8 - IRQ3

COM3 - & h3e8 - IRQ4

COM4 - & h2e8 - IRQ3

Ένα εναλλακτικό IRQ μπορεί να καθοριστεί χρησιμοποιώντας την επιλογή πρωτοκόλλου "IRn" όπου το n είναι 3 έως 7.

Προς το παρόν δεν υποστηρίζονται: Τα IRQ σε PIC, η εναλλακτική διεύθυνση εισόδου/εξόδου βάσης, τα χρονικά όρια και τα περισσότερα σφάλματα όπως εντοπίζονται στο QB, έλεγχος ροής υλικού, FIFO.

"COM:" αντιστοιχεί στο "COM1:"

## Open Cons

Ανοίγει τις τυπικές ροές εισόδου (stdin) ή εξόδου (stdout) της κονσόλας για χρήση σε λειτουργίες αρχείων.

### Σύνταξη:



Open Cons As [#]filename

Open Cons For Input As [#]filename

Open Cons For Output As [#]filename

Το Open Cons ανοίγει τις ροές stdin ή stdout της κονσόλας για ανάγνωση ή γραφή. Ένας αριθμός αρχείου είναι δεσμευμένος στη ροή, η οποία χρησιμοποιείται σε επόμενες λειτουργίες αρχείων, όπως η Input #. Μπορείτε να ανακτήσετε έναν διαθέσιμο αριθμό αρχείου με το FreeFile.

Η λειτουργία αρχείου εισόδου ανοίγει τη ροή stdin για ανάγνωση λειτουργιών αρχείων, όπως η Line Input #, ενώ η λειτουργία αρχείου εξόδου ανοίγει τη ροή stdout για εγγραφή λειτουργιών αρχείων, όπως η Print #. Η λειτουργία αρχείου εξόδου είναι η προεπιλεγμένη, εάν δεν έχει καθοριστεί.

Τα ρεύματα stdin και stdout είναι αυτά που χρησιμοποιούνται όταν η είσοδος ή η έξοδος της διαδικασίας κλήσης ανακατευθύνεται (διοχετεύεται) με εντολές OS ή όταν ανοίγει με το Open Pipe.

Για να ανοίξετε και τις ροές stdin και stdout για λειτουργίες αρχείων, μια διαδικασία πρέπει να χρησιμοποιεί πολλούς αριθμούς αρχείων.

Ο κωδικός σφάλματος που επιστρέφεται από το Open Cons μπορεί να ελεγχθεί χρησιμοποιώντας το Err στην επόμενη γραμμή. Η έκδοση λειτουργίας του Open Cons επιστρέφει απευθείας τον κωδικό σφάλματος ως 32 bit Long.

## Παράδειγμα - opencons.bas



```
1. Dim a As String
2.
3. Open Cons For Input As #1
4. Open Cons For Output As #2
5.
6. Print #2,"Please write something and press ENTER"
7. Line Input #1,a
8. Print #2, "You wrote : ";a
9.
10.
11. Close
12. Sleep
13. End
```

## Έξοδος:



```
Please write something and press ENTER
hello world
You wrote : hello world
```



## Open Err

Ανοίγει τόσο τις τυπικές ροές εισόδου (stdin) όσο και τις τυπικές ροές σφαλμάτων (stderr) για λειτουργίες αρχείων.

### Σύνταξη:



Open Err [for mode] as [#]filename

ή

result = Open Err( [for mode[,]] as [#]filename )

Αυτή η εντολή ανοίγει το stdin για ανάγνωση και το stderr για εγγραφή στην κονσόλα επιτρέποντας λειτουργίες ανάγνωσης και εγγραφής με κανονικές εντολές αρχείων.

Το stderr είναι μια ροή εξόδου διαφορετική από το stdout που επιτρέπει την ανακατεύθυνση μηνυμάτων σφάλματος ξεχωριστά από την κύρια έξοδο της κονσόλας.

Οι κανονικές εντολές της κονσόλας, όπως το Color and Locate, δεν λειτουργούν σε αυτήν τη λειτουργία, επειδή δεν δέχονται έναν αριθμό αρχείου.

Η λειτουργία [For Input | Output] επιτρέπεται για συμβατότητα, αλλά αγνοείται.

Ο κωδικός σφάλματος που επιστρέφεται από το Open Err μπορεί να ελεγχθεί χρησιμοποιώντας το Err στην επόμενη γραμμή. Η έκδοση λειτουργίας του Open Err επιστρέφει απευθείας τον κωδικό σφάλματος ως Long 32 bit.

### Παράδειγμα - openerr.bas



```
1. Dim a As String
2. Open Err For Input As #1
3. Print #1, "Please write something and press ENTER"
4. Line Input #1, a
5. Print #1, "You wrote: "; a
6. Close
7. Sleep
8. End
```

## Έξοδος:



```
Please write something and press ENTER
this is an error
You wrote: this is an error
```

## Open Lpt

Ανοίγει μια συσκευή εκτύπωσης.

### Σύνταξη:



```
Open Lpt ["[LPT[x]:][Printer_Name]
[,TITLE=Doc_Title][,EMU=TTY]" [For Input|
Output] As #filename
```

### Παράμετροι:

x, Καθορίζει έναν αριθμό θύρας. Εάν παραλειφθεί, η έξοδος αποστέλλεται στο σύστημα εκτύπωσης συστήματος Printer\_Name, Όνομα εκτυπωτή για άνοιγμα. Αυτή η παράμετρος αγνοείται στο DOS.

TITLE = Doc\_Title, Τίτλος της εργασίας εκτύπωσης, όπως φαίνεται από το πρόγραμμα του εκτυπωτή. Αυτή η παράμετρος αγνοείται στο DOS.

EMU = TTY, Εξομοίωση της εξόδου TTY σε εκτυπωτή Windows GDI, χρησιμοποιώντας απεικόνιση κειμένου προγράμματος οδήγησης. Αυτή η παράμετρος αγνοείται σε DOS και Linux.

For Input | Output, επιτρέπεται η παράμετρος για συμβατότητα, αλλά αγνοείται.

filename, Αχρησιμοποίητος αριθμός αρχείου για εκχώρηση στη συσκευή.

### Επιστρεφόμενη τιμή:

Ένα 32 bit Long: 0 επιστρέφεται εάν το Open Lpt () ολοκληρωθεί με επιτυχία, διαφορετικά μια μη μηδενική τιμή επιστρέφει για να υποδείξει την αποτυχία.

## Περιγραφή:

Το `Open Lpt` ανοίγει μια σύνδεση με μια συσκευή εκτυπωτή. Η σύνδεση αντιμετωπίζεται σαν αρχείο, επομένως τα δεδομένα ενδέχεται να εγγραφούν στον εκτυπωτή χρησιμοποιώντας τις εντολές `Print` και `Put #`.

Οποιοσδήποτε εκτυπωτής συνδεδεμένος στο σύστημα μπορεί να ανοίξει με το `Open Lpt`

`Open Lpt "LPT:"` ... θα προσπαθήσει να ανοίξει τον προεπιλεγμένο εκτυπωτή σε Windows και Linux και `"LPT1:"` σε DOS.

Το `LPrint` θα προσπαθήσει αυτόματα να ανοίξει τον προεπιλεγμένο εκτυπωτή σε Windows και Linux και `"LPT1:"` σε DOS.

Ο κωδικός σφάλματος που επιστρέφεται από το `Open Lpt` μπορεί να ελεγχθεί χρησιμοποιώντας το `Err` στην επόμενη γραμμή. Η έκδοση λειτουργίας του `Open Lpt` επιστρέφει απευθείας τον κωδικό σφάλματος ως 32 bit Long.

## Παράδειγμα:



```
1. ' Send some text to the Windows printer on LPT1:,  
2. ' using driver text imaging.  
3.  
4. Open Lpt "LPT1:EMU=TTY" For Output As #1  
5.  
6. Print #1, "Testing!"  
7.  
8. Close
```

## Παράδειγμα:



```
1. 'This simple program will print a PostScript file
2. 'to a PostScript compatible printer.
3.
4. Dim As UByte FFI, PPO
5. Dim As String temp
6.
7.
8. FFI = FreeFile()
9. Open "sample.ps" For Input Access Read As #FFI
10.
11. PPO = FreeFile()
12. Open Lpt "LPT1:" For Output As #PPO
13.
14. While (EOF(FFI) = 0)
15.     Line Input #FFI, temp
16.     Print #PPO, temp
17. Wend
18.
19. Close #FFI
20. Close #PPO
21.
22. Print "Printing Completed!"
23.
```

## Open Pipe

Ανοίγει τη προκαθορισμένη ροή εισόδου (stdin) ή εξόδου (stdout) για λειτουργίες αρχείων.

### Σύνταξη:



```
result = Open Pipe( command[,] For {Input|Output}
[,] As filenumber )
```

ή

```
result = Open Pipe( command[,] For Binary[,]
access_type[,] As filenumber )
```

### **Παράμετροι:**

shell\_command, Η εξωτερική διαδικασία για εκτέλεση στο κέλυφος εντολών του λειτουργικού συστήματος. Οι σχετικές διαδρομές αρχείων είναι σχετικές με τον τρέχοντα κατάλογο (βλ. CurDir). Κατά το άνοιγμα ενός σωλήνα για μια διαδικασία που απαιτεί διπλά εισαγωγικά είτε στην εκτελέσιμη διαδρομή του είτε στα ορίσματά του, ολόκληρη η συμβολοσειρά σωλήνων θα πρέπει να φωλιάζει μέσα σε διπλά εισαγωγικά.

access\_type, Ο τύπος πρόσβασης ανάγνωσης ή εγγραφής που ζητείται από τη διαδικασία κλήσης.  
Access {Read | Write} (μπορεί να ανοίξει η ροή stdin ή stdout της εξωτερικής διαδικασίας)

flenumber, Ένας διαθέσιμος αριθμός αρχείου για σύνδεση με τη ροή stdin ή stdout της εξωτερικής διαδικασίας.

### **Επιστροφόμενη τιμή:**

Στην πρώτη χρήση, το Open Pipe () επιστρέφει ένα 32 bit Long: ένα μηδέν (0) στην επιτυχία και έναν μη μηδενικό κωδικό σφάλματος διαφορετικά.

### **Περιγραφή:**

Το Open Pipe εκτελεί μια άλλη διαδικασία στο κέλυφος εντολών και ανοίγει είτε τα stdin είτε τα stdout streams για ανάγνωση ή εγγραφή. Ένας αριθμός αρχείου είναι δεσμευμένος στη ροή, η οποία χρησιμοποιείται σε επόμενες λειτουργίες αρχείων, όπως το Input #. Μπορείτε να ανακτήσετε έναν διαθέσιμο αριθμό αρχείων με το FreeFile. Εάν η εξωτερική διαδικασία δεν υπάρχει, εμφανίζεται ένα σφάλμα χρόνου εκτέλεσης.

Οι λειτουργίες αρχείων εισόδου και εξόδου ανοίγουν τις ροές stdin και stdout της εξωτερικής διαδικασίας, αντίστοιχα, για διαδοχικά εισερχόμενα/εξερχόμενα κείμενα, χρήσιμα για την ανάγνωση ή τη σύνταξη απλού κειμένου. Οι χαρακτήρες, οι λέξεις ή ολόκληρες γραμμές μπορούν στη συνέχεια να διαβαστούν ή να γραφτούν χρησιμοποιώντας

λειτουργίες αρχείων σε κατάσταση κειμένου, όπως το Line Input # και το Print #.

Η λειτουργία δυαδικού (Binary) αρχείου ανοίγει τις ροές stdin ή stdout της εξωτερικής διαδικασίας - ανάλογα με τον τύπο πρόσβασης που καθορίζεται (βλ. Περιγραφή της παραμέτρου access\_type παραπάνω) - για ανάγνωση ή εγγραφή τυχαίων μεγεθών και ερμηνευμένων ακατέργαστων δεδομένων τυχαίας πρόσβασης. Απλές τιμές τύπων δεδομένων, όπως το Byte και το LongInt, και ολόκληρα κομμάτια μνήμης μπορούν να διαβαστούν ή να γραφτούν στις ροές με λειτουργίες αρχείου δυαδικής λειτουργίας, όπως Get # και Put #.

Οι σωλήνες διπλής κατεύθυνσης δεν υποστηρίζονται από τη FreeBASIC και πρέπει να υλοποιούνται χρησιμοποιώντας τις λειτουργίες API του λειτουργικού συστήματος.

Ο κωδικός σφάλματος που επιστρέφεται από το Open Pipe μπορεί να ελεγχθεί χρησιμοποιώντας το Err στην επόμενη γραμμή. Η έκδοση λειτουργίας του Open Pipe επιστρέφει απευθείας τον κωδικό σφάλματος ως Long 32 bit.

### **Σφάλματα χρόνου εκτέλεσης:**

Το Open Pipe εμφανίζει ένα από τα ακόλουθα σφάλματα χρόνου εκτέλεσης:

#### (1) Illegal function call

το αρχείο δεν ήταν ελεύθερο εκείνη τη στιγμή.  
χρησιμοποιήστε το FreeFile για να διασφαλίσετε ότι ο αριθμός αρχείου είναι ελεύθερος.

## Παράδειγμα - openpipe.bas



```
1.  '' This example uses Open Pipe to run a shell
2.  '' command and retrieve its output.
3.  #ifdef __FB_UNIX__
4.  Const TEST_COMMAND = "ls *"
5.  #else
6.  Const TEST_COMMAND = "dir *.*"
7.  #endif
8.
9.  Open Pipe TEST_COMMAND For Input As #1
10.
11. Dim As String ln
12. Do Until EOF(1)
13.     Line Input #1, ln
14.     Print ln
15. Loop
16.
17. Sleep
18. Close #1
19. End
```

## Έξοδος:



```
total 244
drwxrwxr-x  2 user user 4096 Aug  4 16:41 .
drwxrwxr-x 11 user user 4096 Aug  1 13:31 ..
-rw-rw-r--  1 user user  105 Aug  1 19:27 bit.bas
-rw-rw-r--  1 user user  112 Aug  1 19:33 bitreset.bas
-rw-rw-r--  1 user user   93 Aug  1 19:37 bitset.bas
-rw-rw-r--  1 user user  230 Aug  2 12:17 cls.bas
-rw-rw-r--  1 user user  329 Aug  2 12:54 color.bas
-rw-rw-r--  1 user user  396 Aug  2 20:51 dateadd.bas
-rw-rw-r--  1 user user   52 Aug  3 14:00 date.bas
...
```

## Open Scrn

Ανοίγει απευθείας την κονσόλα για είσοδο και έξοδο ως αρχείο.

### Σύνταξη:



Open Scrn [for mode] as [#]filename

ή

result = Open Scrn( [for mode[,]] as [#]filename )

### Παράμετροι:

mode, Είτε Είσοδος είτε Έξοδος. Εάν παραλειφθεί, θεωρείται η Έξοδος.

filename, Αχρησιμοποίητος αριθμός αρχείου.

### Επιστρεφόμενη τιμή:

Ένα 32 bit Long: ένα μηδέν (0) επιστρέφεται εάν το Open Scrn () ολοκληρωθεί με επιτυχία, διαφορετικά μια μη μηδενική τιμή επιστρέφει για να υποδείξει την αποτυχία.

### Περιγραφή:

Αυτή η εντολή ανοίγει την κονσόλα τόσο για είσοδο όσο και για έξοδο ως αρχείο, επιτρέποντας την ανάγνωση/εγγραφή από/προς αυτή με κανονικές εντολές αρχείων.

Αυτή η εντολή μπορεί να χρησιμοποιήσει άμεση πρόσβαση στην κονσόλα για ταχύτητα σε ορισμένες εφαρμογές, επομένως δεν πρέπει να χρησιμοποιείται όταν η είσοδος / έξοδος απαιτείται να ανακατευθυνθεί ή να διοχετευθεί με σωλήνες με εντολές λειτουργικού συστήματος.

Οι κανονικές εντολές της κονσόλας, όπως το Color and Locate, δεν λειτουργούν σε αυτήν τη λειτουργία, επειδή δεν δέχονται έναν αριθμό αρχείου.

Το filename είναι ένας αχρησιμοποίητος αριθμός αρχείου.

Μπορείτε να βρείτε έναν αχρησιμοποίητο αριθμό αρχείου χρησιμοποιώντας το FreeFile.



Ο κωδικός σφάλματος που επιστρέφεται από το Open Scrn μπορεί να ελεγχθεί χρησιμοποιώντας το Err στην επόμενη γραμμή. Η έκδοση λειτουργίας του Open Scrn επιστρέφει απευθείας τον κωδικό σφάλματος ως 32 bit Long.

### **Σφάλματα χρόνου εκτέλεσης:**

Το Open Scrn εμφανίζει ένα από τα ακόλουθα σφάλματα χρόνου εκτέλεσης:

#### (1) Illegal function call

Το αρχείο δεν ήταν ελεύθερο εκείνη τη στιγμή. χρησιμοποιήστε το FreeFile για να διασφαλίσετε ότι ο αριθμός αρχείου είναι ελεύθερος.

### **Παράδειγμα:**



```
1. Dim a As String
2. Open Scrn For Input As #1
3. Print #1,"Please write something and press ENTER"
4. Line Input #1,a
5. Print #1, "You wrote";a
6. Close
7. Sleep
8. End
```

### **Close**

Συνάρτηση ροής I/O για να τερματίσετε την πρόσβαση σε μια συσκευή.

### **Σύνταξη:**



```
Close [#]filenum ] [, [#]filenum ...]
ή
result = Close( [#filenum] )
```

**Παράμετροι:****flenum**

Μια λίστα από αριθμούς αρχείων που είναι για κλείσιμο.

**Επιστρεφόμενη τιμή:**

Το Close επιστρέφει ένα 32 bit Long: ένα μηδέν (0) στην επιτυχία και έναν μη μηδενικό κωδικό σφάλματος διαφορετικά.

**Περιγραφή:**

Κλείνει τα αρχεία των οποίων οι αριθμοί αρχείων έχουν περάσει ως ορίσματα. Εάν περάσει ένας αχρησιμοποίητος αριθμός αρχείου, το Close επιστρέφει ένα σφάλμα.

Το Close χωρίς ορίσματα κλείνει όλα τα αρχεία που είναι ανοιχτά.

Ο τερματισμός του προγράμματος χρησιμοποιώντας μια δήλωση Close θα κλείσει αυτόματα όλα τα αρχεία.

Ο κωδικός σφάλματος που επιστρέφεται από το Close μπορεί να ελεγχθεί χρησιμοποιώντας το Err στην επόμενη γραμμή. Η έκδοση λειτουργίας του Close επιστρέφει απευθείας τον κωδικό σφάλματος ως Long 32 bit.

## Παράδειγμα - close.bas



```
1. ' Create a string and fill it.
2. Dim buffer As String, f As Integer
3.
4. buffer = "Hello World within a file."
5.
6. ' Find the first free file number.
7. f = FreeFile
8.
9.
10. ' Open the file "file.ext" for binary usage,
11. ' using the number "f".
12. Open "file.ext" For Binary As #f
13.
14. ' Place our string inside the file,
15. ' using number "f".
16. Put #f, , buffer
17.
18. ' Close the file. We could also do 'Close #f',
19. ' but it's only necessary if more than one number
20. ' is open.
21. Close
22.
23. ' End of program. (Check the file "file.ext"
24. ' upon running to see the output.)
```

## Reset

Κλείνει όλα τα ανοιχτά αρχεία ή επαναφέρει τους χειριστές εισόδου/εξόδου.

### Σύνταξη:



Reset  
ή  
Reset( streamno )

### Παράμετροι:

streamno, Ο αριθμός ροής για επαναφορά, 0 για stdin ή 1 για stdout.

## Περιγραφή:

Η επαναφορά, όταν καλείται χωρίς ορίσματα, κλείνει όλα τα αρχεία δίσκου.

Η επαναφορά, όταν καλείται με το όρισμα `streamno`, θα επαναφέρει τις ανακατευθυνόμενες ή τις σωληνωτές ροές που σχετίζονται με το `stdin` (0) ή το `stdout` (1).

## Σφάλματα χρόνου εκτέλεσης:

Η επαναφορά (`streamno`) μπορεί να ορίσει ένα από τα ακόλουθα σφάλματα χρόνου εκτέλεσης:

### (1) Illegal function call

το `streamno` δεν ήταν ούτε 0 ούτε 1

### (3) File I/O error

Η επαναφορά του `stdin` ή του `stdout` απέτυχε

## Παράδειγμα:



1. `Open "test.txt" For Output As #1`
2. `Print #1, "testing 123"`
3. `Reset`

## Παράδειγμα:



```
1. Dim x As String
2.
3. ' Read from STDIN from piped input
4. Open Cons For Input As #1
5. While EOF(1) = 0
6.     Input #1, x
7.     Print "*****"; x; "*****"
8. Wend
9. Close #1
10.
11. ' Reset to read from the keyboard
12. Reset(0)
13.
14. Print "Enter some text:"
15. Input x
16.
17.
18. ' Read from STDIN (now from keyboard)
19. Open Cons For Input As #1
20. While EOF(1) = 0
21.     Input #1, x
22.     Print "*****"; x; "*****"
23. Wend
    Close #1
```

## Καταστάσεις I/O αρχείου

### INPUT (FILE MODE)

Ορίζει το άνοιγμα ενός αρχείου κειμένου να είναι για είσοδο. Δηλαδή να μπορούμε να διαβάσουμε από το αρχείο δεδομένα και να τα αποθηκεύσουμε σε μεταβλητές.

### Σύνταξη:



open filename for **Input** [encoding encoding\_type]  
[lock lock\_type] as [#]filenum

## **Παράμετροι:**

### filename

Το όνομα του αρχείου που ανοίγεται για είσοδο.

### encoding\_type

Υποδεικνύει τον τύπο κωδικοποίησης για το αρχείο

### lock\_type

Ο τύπος κλειδώματος που θα χρησιμοποιηθεί όσο το αρχείο θα είναι ανοιχτό.

### filenum

Αχρησιμοποίητος αριθμός αρχείου για συσχέτιση με το ανοιχτό αρχείο

## **Περιγραφή:**

Μια λειτουργία αρχείου που χρησιμοποιείται με το Open για να ανοίξει ένα αρχείο κειμένου για ανάγνωση.

Αυτή η λειτουργία επιτρέπει την ανάγνωση διαδοχικών γραμμών κειμένου με την line Input #ή την ανάγνωση τιμών διαχωρισμένων με κόμμα με την Input #.

Τα αρχεία κειμένου δεν μπορούν να διαβαστούν και να γραφτούν ταυτόχρονα στη FreeBASIC, οπότε εάν απαιτούνται και οι δύο λειτουργίες στο ίδιο αρχείο, πρέπει να ανοίξουν δύο φορές.

Το όνομα αρχείου (filename) πρέπει να είναι μια παράσταση συμβολοσειράς που έχει ως αποτέλεσμα ένα νόμιμο όνομα αρχείου στο λειτουργικό σύστημα προορισμού, χωρίς χαρακτήρες μπαλαντέρ.

Το αρχείο θα αναζητηθεί στον παρόντα κατάλογο, εκτός εάν το όνομα αρχείου περιέχει μια διαδρομή.

Εάν το αρχείο δεν υπάρχει, δημιουργείται ένα σφάλμα. Ο δείκτης ορίζεται στον πρώτο χαρακτήρα του αρχείου.

Το `Encoding_type` υποδεικνύει την κωδικοποίηση Unicode του αρχείου, έτσι ώστε οι χαρακτήρες να διαβάζονται σωστά. Εάν παραλειφθεί, η κωδικοποίηση "ascii" είναι προεπιλεγμένη. Μόνο little endian κωδικοποιήσεις χαρακτήρα υποστηρίζονται αυτήν τη στιγμή.

- "utf8"
- "utf16"
- "utf32"
- "ascii" (προκαθορισμένη επιλογή)

Ο τύπος κλειδώματος (`lock_type`) υποδεικνύει τον τρόπο κλειδώματος του αρχείου για άλλες διαδικασίες, είναι ένας από τους εξής:

- Ανάγνωση (Read) - το αρχείο μπορεί να ανοίξει ταυτόχρονα από άλλες διαδικασίες, αλλά όχι για ανάγνωση
- Εγγραφή (Write) - το αρχείο μπορεί να ανοίξει ταυτόχρονα από άλλες διαδικασίες, αλλά όχι για εγγραφή
- Ανάγνωση και εγγραφή (Read Write) - το αρχείο δεν μπορεί να ανοίξει ταυτόχρονα από άλλες διαδικασίες (η προεπιλογή)

Το `filenum` είναι έγκυρος αριθμός αρχείου FreeBASIC (στο εύρος 1..255) που δεν χρησιμοποιείται για κανένα άλλο αρχείο που είναι προς το παρόν ανοιχτό.

Ο αριθμός αρχείου προσδιορίζει το αρχείο για τις υπόλοιπες λειτουργίες αρχείων.

Μπορείτε να βρείτε έναν δωρεάν αριθμό αρχείου χρησιμοποιώντας τη λειτουργία `Freefile`.

## Παράδειγμα - input\_mode.bas



```
1. Dim ff As UByte
2. Dim randomvar As Integer
3. Dim name_str As String
4. Dim age As Integer
5.
6. ' collect the test data and output to file with
7. ' Write #
8. Input "What is your name? ", name_str
9. Input "What is your age? ", age
10. Randomize
11. Print
12.
13. ff = FreeFile
14. Open "testfile" For Output As #ff
15. Write #ff, Int(Rnd*42), name_str, age
16. Close #ff
17.
18. ' clear variables
19. randomvar = 0
20. name_str = ""
21. age = 0
22.
23.
24. ' input the variables, using Input #
25. ff = FreeFile
26. Open "testfile" For Input As #ff
27. Input #ff, randomvar, name_str, age
28. Close #ff
29.
30. Print "Random Number was: " & randomvar
31. Print "Your name is: " & name_str
32. Print "Your age is: " & age
33.
34. Sleep
End
```

### Έξοδος:



```
What is your name? Dim
What is your age? 44

Random Number was: 6
Your name is: Dim
Your age is: 44
```



## **Ανάλυση:**

Στις γραμμές 1-4 δηλώνουμε ορισμένες μεταβλητές απαραίτητες για το πρόγραμμα.

Στις γραμμές 8-9 έχουμε είσοδο πληροφοριών από τον χρήστη οι οποίες αποθηκεύονται στις αντίστοιχες μεταβλητές.

Στην γραμμή 10 καλούμε την συνάρτηση Randomize για να παράγουμε έναν τυχαίο αριθμό.

Στην γραμμή 13 λαμβάνουμε τον ελεύθερο αριθμό διαθέσιμου αρχείου και το αποθηκεύουμε στην μεταβλητή ff.

Στην γραμμή 14 ανοίγουμε ένα αρχείο για γράψιμο δεδομένων.

Στην γραμμή 15 γράφουμε τα δεδομένα στο αρχείο.

Στην γραμμή 16 κλείνουμε το αρχείο.

Στις γραμμές 19-21 αρχικοποιούμε τις μεταβλητές στις οποίες θα αποθηκεύσουμε τιμές από την είσοδο από το αρχείο.

Στις γραμμές 24-27 λαμβάνουμε έναν αριθμό διαθέσιμου αρχείου, το ανοίγουμε για είσοδο και διαβάζουμε από αυτό τις τιμές του οι οποίες αποθηκεύονται στις αντίστοιχες μεταβλητές. Τέλος κλείνουμε το αρχείο.

Στις γραμμές 29-31 τυπώνουμε στην οθόνη τις τιμές των μεταβλητών.

Τέλος έχουμε την αναμονή του χρήστη και το τέλος του προγράμματος.

## Output

Καθορίζει το αρχείο κειμένου που θα ανοίξει για τη λειτουργία εξόδου

### Σύνταξη:



Open filename for Output [Encoding encoding\_type]  
[Lock lock\_type] as [#]filenum

### Παράμετροι:

#### filename

Το όνομα του αρχείου που ανοίγεται για είσοδο.

#### encoding\_type

Υποδεικνύει τον τύπο κωδικοποίησης για το αρχείο

#### lock\_type

Ο τύπος κλειδώματος που θα χρησιμοποιηθεί όσο το αρχείο θα είναι ανοιχτό.

#### filenum

Αχρησιμοποίητος αριθμός αρχείου για συσχέτιση με το ανοιχτό αρχείο

### Περιγραφή:

Μια λειτουργία αρχείου που χρησιμοποιείται με το Open για να ανοίξει ένα αρχείο κειμένου για εγγραφή.

Αυτή η λειτουργία χρησιμοποιείται για τη σύνταξη κειμένου με το Print # ή τιμές διαχωρισμένες με κόμμα με το Write #.

Τα αρχεία κειμένου δεν μπορούν να διαβαστούν και να γραφτούν ταυτόχρονα στη FreeBASIC, οπότε εάν απαιτούνται και οι δύο λειτουργίες στο ίδιο αρχείο, πρέπει να ανοίξουν δύο φορές.

Το όνομα αρχείου (filename) πρέπει να είναι μια παράσταση συμβολοσειράς που έχει ως αποτέλεσμα ένα νόμιμο όνομα

αρχείου στο λειτουργικό σύστημα προορισμού, χωρίς χαρακτήρες μπαλαντέρ.

Το αρχείο θα αναζητηθεί στον παρόντα κατάλογο, εκτός εάν το όνομα αρχείου περιέχει μια διαδρομή.

Εάν το αρχείο δεν υπάρχει, δημιουργείται ένα σφάλμα. Ο δείκτης ορίζεται στον πρώτο χαρακτήρα του αρχείου.

Το `Encoding_type` υποδεικνύει την κωδικοποίηση Unicode του αρχείου, έτσι ώστε οι χαρακτήρες να διαβάζονται σωστά. Εάν παραλειφθεί, η κωδικοποίηση "ascii" είναι προεπιλεγμένη. Μόνο little endian κωδικοποιήσεις χαρακτήρα υποστηρίζονται αυτήν τη στιγμή.

- "utf8"
- "utf16"
- "utf32"
- "ascii" (προκαθορισμένη επιλογή)

Ο τύπος κλειδώματος (`lock_type`) υποδεικνύει τον τρόπο κλειδώματος του αρχείου για άλλες διαδικασίες, είναι ένας από τους εξής:

- Ανάγνωση (Read) - το αρχείο μπορεί να ανοίξει ταυτόχρονα από άλλες διαδικασίες, αλλά όχι για ανάγνωση
- Εγγραφή (Write) - το αρχείο μπορεί να ανοίξει ταυτόχρονα από άλλες διαδικασίες, αλλά όχι για εγγραφή
- Ανάγνωση και εγγραφή (Read Write) - το αρχείο δεν μπορεί να ανοίξει ταυτόχρονα από άλλες διαδικασίες (η προεπιλογή)

Το `filenum` είναι έγκυρος αριθμός αρχείου FreeBASIC (στο εύρος 1..255) που δεν χρησιμοποιείται για κανένα άλλο αρχείο που είναι προς το παρόν ανοιχτό.

Ο αριθμός αρχείου προσδιορίζει το αρχείο για τις υπόλοιπες λειτουργίες αρχείων.

Μπορείτε να βρείτε έναν δωρεάν αριθμό αρχείου χρησιμοποιώντας τη λειτουργία Freefile.

## Παράδειγμα - output\_mode.bas



```
1. Dim ff As UByte
2. Dim randomvar As Integer
3. Dim name_str As String
4. Dim age_ubyte As UByte
5.
6. ff = Freefile
7. Input "What is your name? ",name_str
8. Input "What is your age? ",age_ubyte
9.
10.
11. Open "testfile" For Output As #ff
12. Write #ff, Int(Rnd(0)*42),name_str,age_ubyte
13. Close #ff
14.
15. randomvar=0
16. name_str=""
17. age_ubyte=0
18.
19. Open "testfile" For Input As #ff
20. Input #ff, randomvar,name_str,age_ubyte
21. Close #ff
22.
23. Print "Random Number was: ", randomvar
24. Print "Your name is: " + name_str
25. Print "Your age is: " + Str(age_ubyte)
26.
27. Sleep
28. End
```

## Έξοδος:



```
What is your name? Dim
What is your age? 44
Random Number was: 0
Your name is: Dim
Your age is: 44
```

Ο κώδικας του παραδείγματος είναι παρόμοιος με τον κώδικα του προηγούμενου παραδείγματος.

## Append

Καθορίζει το αρχείο κειμένου που θα ανοίξει για τη λειτουργία προσθήκης

### Σύνταξη:



Open filename for Append [Encoding encoding\_type] [Lock lock\_type] as [#]filenum

### Παράμετροι:

#### filename

Το όνομα του αρχείου που ανοίγεται για είσοδο.

#### encoding\_type

Υποδεικνύει τον τύπο κωδικοποίησης για το αρχείο

#### lock\_type

Ο τύπος κλειδώματος που θα χρησιμοποιηθεί όσο το αρχείο θα είναι ανοιχτό.

#### filenum

Αχρησιμοποίητος αριθμός αρχείου για συσχέτιση με το ανοιχτό αρχείο

### Περιγραφή:

Μια λειτουργία αρχείου που χρησιμοποιείται με το Open για να ανοίξει ένα αρχείο κειμένου για προσθήκη.

Αυτή η λειτουργία χρησιμοποιείται για τη σύνταξη κειμένου με το Print # ή τιμές διαχωρισμένες με κόμμα με το Write #.

Τα αρχεία κειμένου δεν μπορούν να διαβαστούν και να γραφτούν ταυτόχρονα στη FreeBASIC, οπότε εάν απαιτούνται και οι δύο λειτουργίες στο ίδιο αρχείο, πρέπει να ανοίξουν δύο φορές.

Το όνομα αρχείου (filename) πρέπει να είναι μια παράσταση συμβολοσειράς που έχει ως αποτέλεσμα ένα νόμιμο όνομα

αρχείου στο λειτουργικό σύστημα προορισμού, χωρίς χαρακτήρες μπαλαντέρ.

Το αρχείο θα αναζητηθεί στον παρόντα κατάλογο, εκτός εάν το όνομα αρχείου περιέχει μια διαδρομή.

Εάν το αρχείο δεν υπάρχει, δημιουργείται ένα σφάλμα. Ο δείκτης ορίζεται στον πρώτο χαρακτήρα του αρχείου.

Το `Encoding_type` υποδεικνύει την κωδικοποίηση Unicode του αρχείου, έτσι ώστε οι χαρακτήρες να διαβάζονται σωστά. Εάν παραλειφθεί, η κωδικοποίηση "ascii" είναι προεπιλεγμένη. Μόνο little endian κωδικοποιήσεις χαρακτήρα υποστηρίζονται αυτήν τη στιγμή.

- "utf8"
- "utf16"
- "utf32"
- "ascii" (προκαθορισμένη επιλογή)

Ο τύπος κλειδώματος (`lock_type`) υποδεικνύει τον τρόπο κλειδώματος του αρχείου για άλλες διαδικασίες, είναι ένας από τους εξής:

- Ανάγνωση (Read) - το αρχείο μπορεί να ανοίξει ταυτόχρονα από άλλες διαδικασίες, αλλά όχι για ανάγνωση
- Εγγραφή (Write) - το αρχείο μπορεί να ανοίξει ταυτόχρονα από άλλες διαδικασίες, αλλά όχι για εγγραφή
- Ανάγνωση και εγγραφή (Read Write) - το αρχείο δεν μπορεί να ανοίξει ταυτόχρονα από άλλες διαδικασίες (η προεπιλογή)

Το `filenum` είναι έγκυρος αριθμός αρχείου FreeBASIC (στο εύρος 1..255) που δεν χρησιμοποιείται για κανένα άλλο αρχείο που είναι προς το παρόν ανοιχτό.

Ο αριθμός αρχείου προσδιορίζει το αρχείο για τις υπόλοιπες λειτουργίες αρχείων.

Μπορείτε να βρείτε έναν δωρεάν αριθμό αρχείου χρησιμοποιώντας τη λειτουργία Freefile.

## Παράδειγμα - append\_mode.bas



```
1. Dim i As Integer
2. For i = 1 To 10
3.     Open "test.txt" For Append As #1
4.     Print #1, i; "extending test.txt"
5.     Close #1
6. Next
```

### Έξοδος:

Το αρχείο test.txt περιέχει τις εξής γραμμές:



```
1 extending test.txt
2 extending test.txt
3 extending test.txt
4 extending test.txt
5 extending test.txt
6 extending test.txt
7 extending test.txt
8 extending test.txt
9 extending test.txt
10 extending test.txt
```

### Ανάλυση:

Στην γραμμή 1 δηλώνουμε μια μεταβλητή ακεραίου τύπου. Στις γραμμές 2-6 έχουμε έναν βρόγχο For...Next που επαναλαμβάνεται 10 φορές.

Στην γραμμή 3 ανοίγουμε ένα αρχείο για προσθήκη. Αυτό σημαίνει ότι τα περιεχόμενά του αρχείου θα διατηρηθούν και οι νέες εγγραφές θα προστεθούν στο τέλος του αρχείου.

Στην γραμμή 4 γράφουμε στο αρχείο.

Στην γραμμή 5 κλείνουμε το αρχείο.

## Binary

Καθορίζει το αρχείο ή τη συσκευή που θα ανοίξει σε δυαδική κατάσταση.

### Σύνταξη:



Open filename for **Binary** [Access access\_type]  
[Lock lock\_type] as [#]filenum

### Παράμετροι:

filename

όνομα αρχείου για άνοιγμα

access\_type

υποδεικνύει εάν το αρχείο μπορεί να διαβαστεί, να γραφτεί ή και στα δύο

lock\_type

κλείδωμα για χρήση ενώ το αρχείο είναι ανοιχτό

filenum

αχρησιμοποίητος αριθμός αρχείου για συσχέτιση με το ανοιχτό αρχείο

### Περιγραφή:

Ανοίγει ένα αρχείο ή μια συσκευή για ανάγνωση ή / και γράψιμο για δυαδικά δεδομένα στο αριθμό αρχείου, με την ελεύθερη μορφή.

Εάν το αρχείο δεν υπάρχει, θα δημιουργηθεί ένα νέο αρχείο. Ο δείκτης αρχείου αρχικοποιείται με το Open στο byte αρ. 1. Λειτουργίες Get # και Put # μετακινούν το δείκτη του αρχείου σύμφωνα με το μέγεθος των δεδομένων, ο δείκτης μπορεί να οριστεί σε οποιοδήποτε byte στο αρχείο.

Τα δεδομένα που υπάρχουν στο αρχείο διατηρούνται από το Open.

Αυτή η λειτουργία αρχείου μπορεί να χρησιμοποιήσει οποιαδήποτε μεταβλητή buffer για την ανάγνωση/εγγραφή δεδομένων στο αρχείο.



Τα δεδομένα αποθηκεύονται σε δυαδική λειτουργία, στην ίδια εσωτερική μορφή που χρησιμοποιεί το FreeBASIC, μέσω του Get # και Put #.

Το όνομα αρχείου (filename) πρέπει να είναι μια παράσταση συμβολοσειράς που έχει ως αποτέλεσμα ένα νόμιμο όνομα αρχείου στο λειτουργικό σύστημα, χωρίς χαρακτήρες μπαλαντέρ.

Το αρχείο θα αναζητηθεί στον παρόντα κατάλογο, εκτός εάν δοθεί διαδρομή.

Τύπος πρόσβασης (Access\_type), από προεπιλογή, η δυαδική λειτουργία επιτρέπει τόσο την ανάγνωση όσο και την εγγραφή του αρχείου, εκτός εάν καθοριστεί ένας τύπος πρόσβασης, πρέπει να είναι ένας από τους εξής:

Ανάγνωση (Read) - το αρχείο ανοίγει μόνο για εισαγωγή

Εγγραφή (Write)- το αρχείο ανοίγει μόνο για έξοδο

Ανάγνωση και εγγραφή (Read Write) - το αρχείο ανοίγει για είσοδο και έξοδο (προεπιλογή)

Ο τύπος κλειδώματος (Lock\_type) υποδεικνύει τον τρόπο κλειδώματος του αρχείου για άλλες διαδικασίες (χρήστες ή νήματα), είναι ένας από τους εξής:

Κοινόχρηστο (Shared)- Το αρχείο μπορεί να έχει ελεύθερη πρόσβαση από άλλες διαδικασίες

Κλείδωμα ανάγνωσης (Lock Read) - Το αρχείο δεν μπορεί να ανοίξει ταυτόχρονα για ανάγνωση

Κλείδωμα εγγραφής (Lock Write) - Το αρχείο δεν μπορεί να ανοίξει ταυτόχρονα για εγγραφή

Κλείδωμα ανάγνωσης εγγραφής (Lock Read Write) - Το αρχείο δεν μπορεί να ανοίξει ταυτόχρονα από άλλες διαδικασίες.

Εάν δεν αναφέρεται τύπος κλειδώματος, το αρχείο θα μοιραστεί για άλλα νήματα του προγράμματος και θα κλειδώσει την ανάγνωση εγγραφής για άλλα προγράμματα. Το κλείδωμα και το ξεκλείδωμα μπορούν να χρησιμοποιηθούν για τον περιορισμό της πρόσκαιρης πρόσβασης σε μέρη ενός αρχείου.

Το `filenum` είναι έγκυρος αριθμός αρχείου (στο εύρος 1..255) που δεν χρησιμοποιείται για κανένα άλλο αρχείο που είναι προς το παρόν ανοιχτό. Ο αριθμός αρχείου προσδιορίζει το αρχείο για τις υπόλοιπες λειτουργίες αρχείων. Μπορείτε να βρείτε έναν ελεύθερο αριθμό αρχείου χρησιμοποιώντας τη λειτουργία `FreeFile`.

### Παράδειγμα - `binary_mode.bas`



```
1.  ''Create a binary data file with one number in it
2.  Dim x As Single = 17.164
3.  Open "MyFile.Dat" For Binary As #1
4.  '' put without a position setting will
5.  '' put from the last known file position
6.  '' in this case, the very beginning of the file.
7.      Put #1, , x
8.  Close #1
9.
10.
11.
12.  '' Now read the number from the file
13.  Dim a As Single = 0
14.  Open "MyFile.Dat" For Binary As #1
15.      Get #1, , a
16.  Close #1
17.
18.  Print a
19.  Sleep
    End
```

### Έξοδος:



17.164

## Random

Καθορίζει το αρχείο ή τη συσκευή που θα ανοίξει για λειτουργία τυχαίας πρόσβασης

### Σύνταξη:



```
Open filename for Random [Access access_type]  
[Lock lock_type] as [#]filenum [Len =  
record_length]
```

### Παράμετροι:

#### filename

Το όνομα του αρχείου που θα ανοιχθεί.

#### access\_type

Ο τύπος πρόσβασης, αν θα είναι για διάβασμα (Read) ή γράψιμο (Write) ή και τα δύο.

#### lock\_type

Ο τύπος κλειδώματος του αρχείου που είναι ανοιχτό.

#### filenum

Ο αριθμός του αρχείου που συσχετίζεται με το ανοιχτό αρχείο.

#### record\_length

το μέγεθος της εγγραφής που χρησιμοποιείται για το αρχείο

### Περιγραφή:

Ανοίγει ένα αρχείο ή μια συσκευή για ανάγνωση και/ή εγγραφή δυαδικών δεδομένων στο δεδομένο αρχείο, με εγγραφές μεγέθους record\_length.

Εάν το αρχείο δεν υπάρχει, θα δημιουργηθεί ένα νέο αρχείο, διαφορετικά τα δεδομένα που υπάρχουν στο αρχείο διατηρούνται από το Open. Ο δείκτης αρχείου εκκινείται με το Open στην αρχή του αρχείου, στον αριθμό εγγραφής 1. Οι λειτουργίες αρχείων μετακινούν τη θέση του αρχείου σε βήματα των bytes record\_length.

Αυτή η λειτουργία αρχείου χρησιμοποιεί μια μεταβλητή buffer τύπου Type που ορίζεται από τον χρήστη για την ανάγνωση/εγγραφή πλήρων εγγραφών σε ένα αρχείο. Η μεταβλητή buffer χρησιμοποιείται για να περιλαμβάνει πολλά πεδία.

Τα δεδομένα αποθηκεύονται σε δυαδική λειτουργία, στην ίδια εσωτερική μορφή που χρησιμοποιεί η FreeBASIC, μέσω του Get # και του Put #.

Το όνομα αρχείου (filename) πρέπει να είναι μια παράσταση συμβολοσειράς που έχει ως αποτέλεσμα ένα νόμιμο όνομα αρχείου στο λειτουργικό σύστημα προορισμού, χωρίς χαρακτήρες μπαλαντέρ.

Το αρχείο θα αναζητηθεί στον παρόντα κατάλογο, εκτός εάν το όνομα αρχείου περιέχει μια διαδρομή.

Τύπος πρόσβασης (Access\_type), από προεπιλογή, η δυαδική λειτουργία επιτρέπει τόσο την ανάγνωση όσο και την εγγραφή του αρχείου, εκτός εάν καθοριστεί ένας τύπος πρόσβασης, πρέπει να είναι ένας από τους εξής:

Ανάγνωση (Read) - το αρχείο ανοίγει μόνο για εισαγωγή

Εγγραφή (Write)- το αρχείο ανοίγει μόνο για έξοδο

Ανάγνωση και εγγραφή (Read Write) - το αρχείο ανοίγει για είσοδο και έξοδο (προεπιλογή)

Ο τύπος κλειδώματος (Lock\_type) υποδεικνύει τον τρόπο κλειδώματος του αρχείου για άλλες διαδικασίες (χρήστες ή νήματα), είναι ένας από τους εξής:

Κοινόχρηστο (Shared)- Το αρχείο μπορεί να έχει ελεύθερη πρόσβαση από άλλες διαδικασίες

Κλείδωμα ανάγνωσης (Lock Read) - Το αρχείο δεν μπορεί να ανοίξει ταυτόχρονα για ανάγνωση

Κλείδωμα εγγραφής (Lock Write) - Το αρχείο δεν μπορεί να ανοίξει ταυτόχρονα για εγγραφή

Κλείδωμα ανάγνωσης εγγραφής (Lock Read Write) - Το αρχείο δεν μπορεί να ανοίξει ταυτόχρονα από άλλες διαδικασίες.

Εάν δεν αναφέρεται τύπος κλειδώματος, το αρχείο θα μοιραστεί για άλλα νήματα του προγράμματος και θα κλειδώσει την ανάγνωση εγγραφής για άλλα προγράμματα. Το κλείδωμα και το ξεκλείδωμα μπορούν να χρησιμοποιηθούν για τον περιορισμό της πρόσκαιρης πρόσβασης σε μέρη ενός αρχείου.

Το `filenum` είναι έγκυρος αριθμός αρχείου (στο εύρος 1..255) που δεν χρησιμοποιείται για κανένα άλλο αρχείο που είναι προς το παρόν ανοιχτό. Ο αριθμός αρχείου προσδιορίζει το αρχείο για τις υπόλοιπες λειτουργίες αρχείων. Μπορείτε να βρείτε έναν ελεύθερο αριθμό αρχείου χρησιμοποιώντας τη λειτουργία `FreeFile`.

Το `record_length` είναι ο αριθμός των `byte` που θα μετακινεί ο δείκτης αρχείου για κάθε μεμονωμένο `Get` and `Put`, πρέπει να ταιριάζει με το μέγεθος της μεταβλητής `buffer` που χρησιμοποιείται κατά τη λήψη και την τοποθέτηση δεδομένων. Εάν παραλειφθεί, έχει προεπιλογή σε 128 `bytes`.

## Παράδειγμα - random\_mode.bas



```
1. ' Defines a UDT
2. Type ScoreEntry Field = 1
3.     As String * 20 Name
4.     As Single score
5. End Type
6.
7. ' Defines an object of type ScoreEntry
8. Dim As ScoreEntry entry
9.
10. '' Generate a fake boring highscore file
11. Open "scores.dat" For Random Access Write As #1
12. Len = SizeOf(entry)
13. For i As Integer = 1 To 10
14.     entry.name = "Player " & i
15.     entry.score = i
16.     Put #1, i, entry
17. Next
18. Close #1
19.
20.
21. '' Read out and display the entries
22. Open "scores.dat" For Random Access Read As #1
23. Len = SizeOf(entry)
24. For i As Integer = 1 To 10
25.     Get #1, i, entry
26.     Print i & ":", entry.name, Str(entry.score),
27.     entry.score
28. Next
29. Close #1
30.
31. Sleep
End
```

## Έξοδος:



```
1:      Player 1      1      1
2:      Player 2      2      2
3:      Player 3      3      3
4:      Player 4      4      4
5:      Player 5      5      5
6:      Player 6      6      6
7:      Player 7      7      7
8:      Player 8      8      8
9:      Player 9      9      9
10:     Player 10     10     10
```

## Δικαιώματα πρόσβασης αρχείων

### Access

Ρήτρα της δήλωσης Open για να καθορίσετε τα δικαιώματα πρόσβασης.

### Σύνταξη:



Open filename for Binary Access {Read | Write | Read Write} as [#]filenum

### Παράμετροι:

Read

Ανοίγει το αρχείο με δικαιώματα διαβάσματος μόνο.

Write

Ανοίγει το αρχείο με δικαιώματα εγγραφής μόνο.

Read Write

Ανοίγει το αρχείο με δικαιώματα διαβάσματος και εγγραφής.

### Περιγραφή:

Η πρόσβαση χρησιμοποιείται με τη δήλωση Open για να ζητήσετε δικαιώματα ανάγνωσης, εγγραφής ή ανάγνωσης και εγγραφής. Εάν η ρήτρα πρόσβασης δεν έχει καθοριστεί, η ανάγνωση και εγγραφή θεωρείται το προκαθορισμένο δικαίωμα πρόσβασης.

### Read (File Access)

Προσδιοριστής πρόσβασης αρχείου.

#### Σύνταξη:



Open filename As String For Binary Access **Read** As #filenum As Integer

### Περιγραφή:

Προσδιοριστής για τη ρήτρα πρόσβασης στη δήλωση Open. Το Read καθορίζει ότι το αρχείο είναι προσβάσιμο για εισαγωγή.

### Write (File Access)

Προσδιοριστής πρόσβασης αρχείου.

#### Σύνταξη:



Open filename As String For Binary Access **Write** As #filenum As Integer

### Περιγραφή:

Προσδιοριστής για τη ρήτρα πρόσβασης στη δήλωση Open. Το Write καθορίζει ότι το αρχείο είναι προσβάσιμο για έξοδο.



## Read Write (File Access)

Προσδιοριστής πρόσβασης αρχείου.

### Σύνταξη:



Open filename As String For Binary Access **Read Write** As #filenum As Integer

### Περιγραφή:

Προσδιοριστής για τη ρήτρα πρόσβασης στη δήλωση Open. Το Read Write καθορίζει ότι το αρχείο είναι προσβάσιμο για είσοδο και έξοδο.

## Κωδικοποίηση χαρακτήρων

### Encoding

Καθορίζει τη μορφή χαρακτήρων ενός αρχείου κειμένου.

### Σύνταξη:



Open filename for {Input|Output|Append} **Encoding** "utf-8"|"utf-16"|"utf-32"|"ascii" as [#]filenum

### Παράμετροι:

filename

Το όνομα του αρχείου.

Encoding "utf-8"|"utf-16"|"utf-32"|"ascii"

Υποδεικνύει τον τύπο κωδικοποίησης για το αρχείο

filenum

Το filenum είναι έγκυρος αριθμός αρχείου (στο εύρος 1..255) που δεν χρησιμοποιείται για κανένα άλλο αρχείο που είναι προς το παρόν ανοιχτό. Ο αριθμός αρχείου προσδιορίζει το

αρχείο για τις υπόλοιπες λειτουργίες αρχείων. Μπορείτε να βρείτε έναν ελεύθερο αριθμό αρχείου χρησιμοποιώντας τη λειτουργία FreeFile.

### **Περιγραφή:**

Η κωδικοποίηση καθορίζει τη μορφή για ένα αρχείο κειμένου Unicode, οπότε τα Winput # και Print # χρησιμοποιούν τη σωστή κωδικοποίηση. Εάν παραλειφθεί από μια εντολή Open, η κωδικοποίηση "ascii" είναι η προεπιλογή.

Μόνο little endian κωδικοποιήσεις χαρακτήρα υποστηρίζονται αυτήν τη στιγμή.

- "utf8"
- "utf16"
- "utf32"
- "ascii" (προκαθορισμένη επιλογή)

## **Ανάγνωση και εγγραφή σε αρχεία ή συσκευές**

### **Input #**

Διαβάζει μια λίστα τιμών από ένα αρχείο κειμένου.

### **Σύνταξη:**



Input # filename, variable\_list

## **Παράμετροι:**

### flenum

Ένας αριθμός αρχείου ενός αρχείου ή μιας συσκευής που ανοίγεται για είσοδο

### variable\_list

Μια λίστα μεταβλητών που χρησιμοποιούνται για τη διατήρηση των τιμών που διαβάζονται

## **Περιγραφή:**

Διαβάζει από ένα αρχείο κειμένου μέσω ενός δεσμευμένου αριθμού αρχείου ένα σύνολο τιμών χωρισμένων από οριοθέτες και τις γράφει με σειρά ανάγνωσης στις μεταβλητές στη `variable_list`.

Εάν μια μεταβλητή είναι αριθμητική, η τιμή ανάγνωσης μετατρέπεται από την παράσταση συμβολοσειράς της στον αντίστοιχο τύπο.

Οι αριθμητικές τιμές μετατρέπονται με παρόμοιο τρόπο στις διαδικασίες `Val` και `ValLng`, χρησιμοποιώντας την καταλληλότερη συνάρτηση για τη μορφή αριθμών.

Τα όρια μπορεί να είναι κόμματα ή αλλαγές γραμμών. Το `Whitespace` αντιμετωπίζεται επίσης ως διαχωριστικό μετά από αριθμούς. Μια συμβολοσειρά που περιλαμβάνει ένα κόμμα ή ένα κενό διάστημα πρέπει να περιβάλλεται από διπλά εισαγωγικά.

Για να διαβάσετε μια ολόκληρη γραμμή σε μια συμβολοσειρά, χρησιμοποιήστε την `Line Input`.

Η `Write #` μπορεί να χρησιμοποιηθεί για τη δημιουργία ενός αρχείου αναγνώσιμου με την `Input #`.

## Παράδειγμα - input.bas



```
1. Dim a As Integer
2. Dim b As String
3. Dim c As Single
4.
5. Open "myfile.txt" For Output As #1
6. Write #1, 1, "Hello, World", 34.5
7. Close #1
8.
9. Open "myfile.txt" For Input As #1
10. Input #1, a, b, c
11. Close #1
12.
13. Print a, b, c
14. Sleep
15. End
```

### Έξοδος:



```
1 Hello, World 34.5
```

### Write #

Εξάγει μια λίστα τιμών διαχωρισμένη με κόμμα σε ένα αρχείο κειμένου ή μια συσκευή.

### Σύνταξη:



Write # filename , [ expressionlist ]

## **Παράμετροι:**

### filenum

Ένας αριθμός αρχείου ενός αρχείου ή μιας συσκευής που ανοίγεται για έξοδο (Output) ή προσθήκη (Append).

### expressionlist

Μια λίστα στοιχείων που θα εγγραφούν

## **Περιγραφή:**

Εξάγει τις τιμές στη λίστα έκφρασης (expressionlist) στο αρχείο κειμένου ή τη συσκευή που είναι συνδεδεμένη στο αρχείο.

Οι τιμές διαχωρίζονται με κόμματα και οι συμβολοσειρές περικλείονται σε διπλά εισαγωγικά.

Αριθμητικές τιμές μεγαλύτερες από μηδέν (0) και μικρότερες από ένα (1) προτίθενται με μηδέν (0) εάν δεν δίνεται καμία τιμή (π.χ., η τιμή -.123 θα προκύψει ως -0.123). Τα επιπλέον μηδενικά περικόπτονται.

Εάν δεν παρέχεται λίστα έκφρασης, η Write # εξάγει μια επιστροφή μεταφοράς (σημειώστε ότι το κόμμα μετά το αρχείο είναι ακόμα απαραίτητο, ακόμη και αν δεν δίνεται λίστα έκφρασης).

Ο σκοπός της Write # είναι να δημιουργήσει ένα αρχείο που μπορεί να διαβαστεί με τη χρήση της Input #.

## Παράδειγμα - write.bas



```
1. Const filename As String = "file.txt"
2.
3. Dim filenum As Integer = FreeFile()
4.
5. If 0 <> Open(filename, For Output, As filenum)
6. Then
7.     Print "error opening " & filename & " for
8.     output."
9.     End -1
10. End If
11.
12.
13. Dim i As Integer = 10
14. Dim d As Double = 123.456
15. Dim s As String = "text"
16.
17. Write #filenum, 123, "text", -.45600
18. Write #filenum,
19. Write #filenum, i, d, s
```

### Έξοδος:

Θα δημιουργηθεί το αρχείο file.txt με το εξής περιεχόμενο.

```
123,"text",-0.456
10,123.456,"text"
```

### Input()

Διαβάζει έναν αριθμό χαρακτήρων από την κονσόλα ή το αρχείο.

### Σύνταξη:



```
result = Input[$]( n [, [#]filenum ] )
```

## **Παράμετροι:**

n

Αριθμός byte για ανάγνωση.

filenum

Αριθμός αρχείου συνδεδεμένου αρχείου ή συσκευής.

Επιστρεφόμενη τιμή

Επιστρέφει μια συμβολοσειρά των χαρακτήρων που έχουν διαβαστεί.

## **Περιγραφή:**

Διαβάζει έναν αριθμό χαρακτήρων από την κονσόλα ή ένα δεσμευμένο αρχείο/συσκευή που καθορίζεται από το αρχείο. Η πρώτη έκδοση περιμένει και διαβάζει n χαρακτήρες από το buffer του πληκτρολογίου. Τα εκτεταμένα κλειδιά δεν διαβάζονται.

Οι χαρακτήρες δεν τυπώνονται στην οθόνη.

Η δεύτερη έκδοση περιμένει και διαβάζει n χαρακτήρες από ένα αρχείο ή μια συσκευή. Η θέση του αρχείου ενημερώνεται.

## **Winput()**

Διαβάζει έναν αριθμό ευρέων χαρακτήρων από την κονσόλα ή το αρχείο.

## **Σύνταξη:**



```
result = WInput( num [, [#]filenum } )
```

## **Παράμετροι:**

num

Αριθμός χαρακτήρων για ανάγνωση.

filenum

Αριθμός αρχείου συνδεδεμένου αρχείου ή συσκευής.

## Επιστρεφόμενη τιμή

Επιστρέφει ένα WString των αναγνωσμένων χαρακτήρων.

### **Περιγραφή:**

Διαβάζει έναν αριθμό ευρέων χαρακτήρων από την κονσόλα ή ένα δεσμευμένο αρχείο/συσκευή που καθορίζεται από το αρχείο.

Η πρώτη έκδοση περιμένει και διαβάζει η μεγάλους χαρακτήρες από το buffer του πληκτρολογίου. Τα εκτεταμένα κλειδιά δεν διαβάζονται. Οι χαρακτήρες δεν τυπώνονται στην οθόνη.

Η δεύτερη έκδοση περιμένει και διαβάζει η μεγάλους χαρακτήρες από ένα αρχείο ή μια συσκευή. Η θέση του αρχείου ενημερώνεται.

Σημείωση: Η FreeBASIC δεν υποστηρίζει προς το παρόν την ανάγνωση ευρέων χαρακτήρων από την κονσόλα.

## **Line Input #**

Διαβάζει μία γραμμή κειμένου από ένα αρχείο.

### **Σύνταξη:**



Line Input #file number, string\_variable

### **Παράμετροι:**

#### file number

αριθμός αρχείου ενός αρχείου που έχει ανοίξει για Input.

#### string\_variable

μεταβλητή για λήψη της γραμμής κειμένου

### **Περιγραφή:**

Διαβάζει μια γραμμή από ένα ανοιχτό αρχείο κειμένου (ανοίγει για Εισαγωγή μέσω ενός δεσμευμένου αριθμού αρχείου) και την αποθηκεύει σε μια μεταβλητή συμβολοσειράς.



Μια γραμμή κειμένου τελειώνει, αλλά δεν περιλαμβάνει το τέλος των χαρακτήρων γραμμής. Ένας χαρακτήρας τέλους γραμμής μπορεί να είναι ο χαρακτήρας LF (Chr (10)) ή το ζεύγος χαρακτήρων CRLF (Chr (13,10)).

### Παράδειγμα - line\_input.bas



```
1. Dim s As String
2.
3. Open "myfile.txt" For Output As #1
4. Print #1, "Hello, World"
5. Close #1
6.
7. Open "myfile.txt" For Input As #1
8. Line Input #1, s
9. Close #1
10.
11.
12. Print s
13.
14. Sleep
End
```

### Έξοδος:



```
Hello, World
```

### (Print | ?) #

Γράφει μια λίστα τιμών σε ένα αρχείο ή συσκευή.

### Σύνταξη:



(Print | ?) # filename, [ expressionlist ] [ , | ; ]

## Παράμετροι:

### filenum

Ο αριθμός αρχείου ενός αρχείου ή συσκευής που ανοίγει για Έξοδος ή Προσάρτηση.

### expressionlist

Λίστα τιμών για εγγραφή.

## Περιγραφή:

Η Print # εξάγει μια λίστα τιμών σε ένα αρχείο κειμένου ή μια συσκευή.

Οι αριθμητικές τιμές μετατρέπονται στην αναπαράσταση συμβολοσειρών τους, με αριστερή θέση για το πρόσημο.

Τα αντικείμενα των τύπων που ορίζονται από τον χρήστη πρέπει να υπερφορτώνουν το τελεστή Cast () ως συμβολοσειρά.

Οι διαδοχικές τιμές στη λίστα έκφρασης διαχωρίζονται είτε με κόμμα (,) είτε με ερωτηματικό (;). Ένα κόμμα υποδηλώνει ότι η εκτύπωση πρέπει να πραγματοποιηθεί στο επόμενο όριο 14 στηλών, ενώ ένα ερωτηματικό υποδεικνύει ότι οι τιμές εκτυπώνονται χωρίς κενό μεταξύ τους.

Ένας χαρακτήρας νέας γραμμής εκτυπώνεται μετά τις τιμές στη λίστα έκφρασης, εκτός εάν η λίστα έκφρασης ακολουθείται με κόμμα ή ερωτηματικό.

Σημειώστε ότι το κόμμα (,) αμέσως μετά τον αριθμό αρχείου είναι ακόμα απαραίτητο, ακόμη και η λίστα έκφρασης είναι κενή. Σε αυτήν την περίπτωση, εκτυπώνεται μια νέα γραμμή, όπως και με μια κανονική λίστα έκφρασης που δεν έχει κόμμα ή ερωτηματικό στο τέλος.

## Παράδειγμα:



```
1. Open "bleh.dat" For Output As #1
2.
3. Print #1, "abc def"
4. Print #1, 1234, 5678.901, "xyz zzz"
5.
6. Close #1
```

## Put (File I/O)

Γράφει δεδομένα από ένα buffer σε ένα αρχείο.

### Σύνταξη:



```
Put #filename, position, data [, amount]  
varres = Put (#filename, position, data [, amount])
```

### Παράμετροι:

#### filename

Η τιμή πέρασε στο Open όταν άνοιξε το αρχείο.

#### position

Είναι η θέση από την οποία πρέπει να ξεκινήσει το Put στο αρχείο. Εάν το αρχείο άνοιξε για τυχαία πρόσβαση, η θέση είναι σε εγγραφές, αλλιώς δίνεται σε byte. Εάν παραλειφθεί, η εγγραφή ξεκινά από την τρέχουσα θέση του δείκτη αρχείου. Η θέση βασίζεται σε 1: δηλαδή η πρώτη εγγραφή ή byte ενός αρχείου βρίσκεται στη θέση 1.

Εάν η θέση παραλείπεται ή μηδενίζεται (0), η εγγραφή του αρχείου θα ξεκινήσει από την τρέχουσα θέση αρχείου.

#### data

Είναι το buffer από το οποίο γράφονται τα δεδομένα.

Μπορεί να είναι μια αριθμητική μεταβλητή, μια συμβολοσειρά, ένας πίνακας ή ένας τύπος που ορίζεται από το χρήστη.

Η λειτουργία θα προσπαθήσει να μεταφέρει στο δίσκο ολόκληρη τη μεταβλητή, εκτός εάν δοθεί το amount.

Κατά την τοποθέτηση πινάκων, τα δεδομένα πρέπει να ακολουθούνται από ένα κενό ζεύγος παρενθέσεων: '()'. Το Put θα γράψει όλα τα δεδομένα στον πίνακα.

Το amount δεν επιτρέπεται.

Κατά την τοποθέτηση συμβολοσειρών, ο αριθμός των byte που γράφονται είναι ίδιος με τον αριθμό των byte στα δεδομένα συμβολοσειράς. το amount δεν επιτρέπεται.

Σημείωση: Εάν θέλετε να γράψετε τιμές από ένα buffer, ΔΕΝ πρέπει να περάσετε έναν δείκτη στο buffer.

Αντ' αυτού, πρέπει να περάσετε την πρώτη μεταβλητή στο buffer. (Αυτό μπορεί να γίνει με αποπαραπομπή του δείκτη με το Operator \* (Value Of).)

Εάν περάσετε απευθείας έναν δείκτη, τότε το Put θα βάλει τη τιμή μνήμης από τη μεταβλητή δείκτη, όχι τη μνήμη στην οποία δείχνει.

### amount

Κάνει την Put να γράψει στο αρχείο ποσότητα διαδοχικών μεταβλητών - δηλ. Γράφει (amount \* SizeOf (data)) byte δεδομένων, ξεκινώντας από τη θέση των δεδομένων στη μνήμη, στο αρχείο. Εάν παραλειφθεί το ποσό, είναι προεπιλεγμένο στο 1, πράγμα που σημαίνει ότι το Put γράφει μόνο μία μεταβλητή.

### Επιστρεφόμενη τιμή

Το Put () επιστρέφει ένα 32 bit Long: 0 στην επιτυχία. μη μηδενικό στο σφάλμα. Ο "δίσκος γεμάτος" θεωρείται σφάλμα και έχει ως αποτέλεσμα τον κωδικό επιστροφής 3. Δεν υπάρχει διαθέσιμος ένας "ακριβής" αριθμός δεδομένων που έχει γραφτεί και δεν θα ήταν πραγματικά χρήσιμος.

### **Περιγραφή**

Γράφει δυαδικά δεδομένα από μια μεταβλητή buffer σε ένα αρχείο που ανοίγεται σε δυαδική ή τυχαία πρόσβαση.

Το Put μπορεί να χρησιμοποιηθεί ως συνάρτηση και θα επιστρέψει 0 στην επιτυχία ή έναν κωδικό σφάλματος σε περίπτωση αποτυχίας.

Για αρχεία που ανοίγονται σε τυχαία πρόσβαση, το μέγεθος σε byte των δεδομένων για εγγραφή πρέπει να ταιριάζει με το καθορισμένο μέγεθος εγγραφής.

## Παράδειγμα - put1.bas



```
1. ' Create variables for the file number,
2. ' and the number to put
3. Dim As Integer f
4. Dim As Long value
5.
6. ' Find the first free file number
7. f = FreeFile()
8.
9. ' Open the file "file.ext" for binary usage,
10. ' using the file number "f"
11. Open "file.ext" For Binary As #f
12. value= 10
13. ' Write the bytes of the integer 'value'
14. ' into the file, using file number "f"
15. ' starting at the beginning
16. ' of the file (position 1)
17. Put #f, 1, value
18.
19.
20. ' Close the file
    Close #f
```

## Παράδειγμα - put2.bas



```
1. ' Create an integer array
2. Dim buffer(1 To 10) As Integer
3.
4. For i As Integer = 1 To 10
5.     buffer(i) = i
6. Next
7.
8. ' Find the first free file file number
9. Dim f As Integer
10. f = FreeFile()
11.
12.
13. ' Open the file "file.ext" for
14. ' binary usage, using the file number "f"
15. Open "file.ext" For Binary As #f
16.
17. ' Write the array into the file,
18. ' using file number "f"
19. ' starting at the beginning
20. ' of the file (position 1)
21. Put #f, 1, buffer()
22.
```

```
23. | ' Close the file
24. | Close #f
```

## Παράδειγμα - put3.bas



```
1. | Dim As Byte Ptr lpBuffer
2. | Dim As Integer hFile, Counter, Size
3. | Size = 256
4. | lpBuffer = Allocate(Size)
5. |
6. | For Counter = 0 To Size-1
7. |     lpBuffer[Counter] = (Counter And &HFF)
8. | Next
9. |
10. | ' Get free file file number
11. | hFile = FreeFile()
12. |
13. | ' Open the file "test.bin" in binary writing mode
14. | Open "test.bin" For Binary Access Write As #hFile
15. |     ' Write 256 bytes from
16. |     ' the memory pointed to by lpBuffer
17. |     Put #hFile, , lpBuffer[0], Size
18. |
19. |
20. | ' Close the file
21. | Close #hFile
22. |
23. | ' Free the allocated memory
24. | Deallocate lpBuffer
```

## Get (File I/O)

Διαβάζει δεδομένα από ένα αρχείο σε ένα buffer

### Σύνταξη:



```
Get #filename, position, data [, [amount] [, bytesread  
 ] ]
```

```
varres = Get (#filename, position, data [, [amount] [,  
 bytesread ] ] )
```

### Παράμετροι:

#### filename

Η τιμή που πέρασε στο Open όταν άνοιξε το αρχείο.

#### position

Η θέση από την οποία πρέπει να ξεκινήσει η ανάγνωση. Εάν το αρχείο άνοιξε για τυχαία πρόσβαση, η θέση είναι σε εγγραφές. Διαφορετικά, είναι σε byte. Εάν παραλειφθεί, η ανάγνωση ξεκινά από την τρέχουσα θέση του δείκτη αρχείου.

Η θέση βασίζεται σε 1-n: δηλαδή η πρώτη εγγραφή ή byte ενός αρχείου βρίσκεται στη θέση 1.

Εάν η θέση παραλείπεται ή μηδενίζεται (0), η ανάγνωση αρχείου θα ξεκινήσει από την τρέχουσα θέση αρχείου.

#### data

Το buffer όπου γράφονται τα δεδομένα.

Μπορεί να είναι μια αριθμητική μεταβλητή, μια συμβολοσειρά, ένας πίνακας, ένας τύπος που ορίζεται από το χρήστη ή ένας δείκτης που δεν αναφέρεται.

Η λειτουργία ανάγνωσης θα προσπαθήσει να γεμίσει πλήρως τη μεταβλητή, εκτός εάν επιτευχθεί το τέλος αρχείου, EOF.

Κατά τη λήψη πινάκων, τα δεδομένα θα πρέπει να ακολουθούνται από ένα κενό ζεύγος παρενθέσεων: "()".

Η Get θα διαβάσει δεδομένα για όλες τις τιμές στον πίνακα, το amount δεν επιτρέπεται.

Κατά τη λήψη συμβολοσειρών, ο αριθμός των byte που διαβάζονται είναι ο ίδιος με τον αριθμό των byte στα δεδομένα συμβολοσειράς, το amount δεν επιτρέπεται.

Σημείωση: Εάν θέλετε να διαβάσετε τιμές σε ένα buffer, ΔΕΝ πρέπει να περάσετε έναν δείκτη στο buffer.

Αντ' αυτού, πρέπει να περάσετε την πρώτη μεταβλητή στο buffer. (Αυτό μπορεί να γίνει με κατάργηση παραπομπής του δείκτη με τον τελεστή \* (Value Of).)

Εάν περάσετε απευθείας έναν δείκτη, τότε το Get θα αντικαταστήσει τη μεταβλητή δείκτη, όχι τη μνήμη στην οποία δείχνει.

### amount

Κάνει το Get να διαβάσει το ποσό (amount) διαδοχικών μεταβλητών από αρχείο σε μνήμη, δηλαδή διαβάζει (amount \* SizeOf (data)) byte δεδομένων από το αρχείο στη μνήμη ξεκινώντας από τη θέση μνήμης των δεδομένων.

Εάν παραλειφθεί το ποσό (amount), είναι προεπιλεγμένο στο 1, πράγμα που σημαίνει ότι το Get διαβάζει μόνο μία μεταβλητή.

### bytesread

Μια μη υπογεγραμμένη ακέραιη μεταβλητή για αποδοχή του αποτελέσματος του αριθμού των byte που διαβάστηκαν με επιτυχία από το αρχείο.

### **Επιστρεφόμενη τιμή**

Το Get () επιστρέφει ένα 32 bit Long: ένα μηδέν (0) στην επιτυχία, μη μηδενικό στο σφάλμα.

Σημείωση: εάν επιτευχθεί EOF (τέλος αρχείου) κατά την ανάγνωση, το Get θα επιστρέψει επιτυχία.

Ο όγκος των byte που διαβάζονται μπορεί να ελεγχθεί περνώντας μια μεταβλητή bytesread.



## Περιγραφή

Διαβάζει δυαδικά δεδομένα από ένα αρχείο σε μια μεταβλητή buffer.

Το Get μπορεί να χρησιμοποιηθεί ως συνάρτηση και θα επιστρέψει το 0 στην επιτυχία ή έναν κωδικό σφάλματος σε περίπτωση αποτυχίας.

Για αρχεία που ανοίγονται σε τυχαία λειτουργία, το μέγεθος σε byte των δεδομένων για ανάγνωση πρέπει να ταιριάζει με το καθορισμένο μέγεθος εγγραφής.

## Θέση αρχείου και άλλες πληροφορίες

### LOF

Επιστρέφει το μήκος ενός ανοιχτού αρχείου στο δίσκο.

### Σύνταξη:



result = LOF( filenum )

### Παράμετροι:

filenum

Ο αριθμός αρχείου ενός ανοιχτού αρχείου δίσκου.

### Επιστρεφόμενη τιμή

Το μήκος σε byte ενός ανοιχτού αρχείου δίσκου.

### Περιγραφή:

Επιστρέφει το μήκος, σε byte, ενός αρχείου που είχε ανοίξει προηγουμένως με το Open χρησιμοποιώντας το δεδομένο αρχείο.

Με το Open Com επιστρέφει το μήκος των δεδομένων που εκκρεμούν για ανάγνωση στο buffer λήψης.

## Παράδειγμα - lof.bas



```
1. Dim f As Integer
2. f = FreeFile
3.
4. Open "file.ext" For Binary As #f
5. Print LOF(f)
6. Close #f
7.
8. Sleep
9. End
```

## LOC

Επιστρέφει τη θέση του αρχείου όπου εκτελέστηκε η τελευταία ανάγνωση/εγγραφή.

### Σύνταξη:



```
result = LOC( filenum )
```

### Παράμετροι:

#### filenum

Ο αριθμός ενός ανοιχτού αρχείου δίσκου.

### Επιστρεφόμενη τιμή

Η θέση του αρχείου όπου πραγματοποιήθηκε η τελευταία ανάγνωση/εγγραφή.

### Περιγραφή:

Επιστρέφει τη θέση όπου εκτελέστηκε το τελευταίο αρχείο ανάγνωσης/εγγραφής.

Η θέση αναφέρεται στα αρχεία:

Σε αρχεία που ανοίγονται FOR RANDOM χρησιμοποιείται το μήκος εγγραφής που καθορίστηκε όταν άνοιξε το αρχείο

Σε αρχεία κειμένου (FOR INPUT | OUTPUT | APPEND,

υποτίθεται ότι έχει μήκος εγγραφής 128 byte.

Σε αρχεία που ανοίγονται ως BINARY χρησιμοποιείται μήκος εγγραφής 1 byte.

Στο FreeBASIC η θέση του αρχείου βασίζεται σε 1-n θέσεις, η πρώτη εγγραφή ενός αρχείου είναι η εγγραφή 1 (LOC = 1 μετά την ανάγνωση ή τη σύνταξη της πρώτης εγγραφής, LOC = 0 για τη θέση έναρξης στο αρχείο).

Όταν χρησιμοποιείται με σειριακή συσκευή, το LOC επιστρέφει τον αριθμό των byte που περιμένουν να διαβαστούν από το buffer εισόδου της σειριακής συσκευής

## EOF

Ελέγχει εάν έχει φτάσει στο τέλος ενός ανοιχτού αρχείου (End Of File)

### Σύνταξη:



```
result = EOF( filenum )
```

### Παράμετροι:

filenum

Ο αριθμός ενός ανοιχτού αρχείου δίσκου.

### Επιστρεφόμενη τιμή

Επιστρέφει την τιμή true (-1) εάν έχει επιτευχθεί το τέλος του αρχείου, διαφορετικά μηδέν (0).

### Περιγραφή:

Όταν διαβάσετε από αρχεία που ανοίγονται για Input (File Mode), είναι χρήσιμο να γνωρίζετε πότε έχει φτάσει το τέλος του αρχείου, αποφεύγοντας έτσι τα σφάλματα που προκαλούνται από την ανάγνωση των άκρων των αρχείων.

Χρησιμοποιήστε το EOF για να το προσδιορίσετε. Το EOF αναμένει έναν έγκυρο αριθμό αρχείου από ένα ήδη ανοιγμένο αρχείο.

Χρησιμοποιήστε το FreeFile για να ανακτήσετε έναν διαθέσιμο αριθμό αρχείου αρχείου.  
Για τους αριθμούς αρχείων που συνδέονται με τα αρχεία που ανοίγονται για Έξοδο, το EOF επιστρέφει πάντα το 0.

### Παράδειγμα:



```
1. ' This code finds a free file number
2. ' to use and attempts to open the file
3. ' "file.ext"
4. ' and if successful, binds our file number
5. ' to the opened file. It reads the file
6. ' line by line, outputting it to the screen.
7. ' We loop until eof() returns true,
8. ' in this case we ignore the loop if file is
9. ' empty.
10. Dim As String file_name
11. Dim As Integer file_num
12. file_name = "file.ext"
13.
14. ' retrieve an available file number
15. file_num = FreeFile( )
16.
17. ' open our file and bind
18. ' our file number to it, exit on error
19. If( Open( file_name For Input As #file_num ) )
20. Then
21.     Print "ERROR: opening file " ; file_name
22.     End -1
23. End If
24.
25.
26. ' loop until we have reached the end of the file
27. Do Until EOF( file_num )
28.     Dim As String text
29.     ' read a line of text ...
30.     Line Input #file_num, text
31.     ' ... and output it to the screen
32.     Print text
33. Loop
34.
35. ' close file via our file number
36. Close #file_num
37.
38. End 0
```

## Seek (Statement)

Ορίζει τη θέση της επόμενης λειτουργίας ανάγνωσης/εγγραφής σε ένα αρχείο.

### Σύνταξη:



Seek [#]filenum, position

### Παράμετροι:

filenum

αριθμός ενός ανοιχτού αρχείου

position

η νέα θέση για λειτουργίες εισόδου/εξόδου

### Περιγραφή:

Ορίζει τη θέση στην οποία θα πραγματοποιηθεί η επόμενη λειτουργία ανάγνωσης ή εγγραφής σε ένα αρχείο.

Η θέση δίνεται σε εγγραφές εάν το αρχείο άνοιξε σε λειτουργία τυχαίας πρόσβασης, σε bytes σε οποιαδήποτε άλλη περίπτωση.

Η θέση βασίζεται σε 1-n - η πρώτη εγγραφή ενός αρχείου βρίσκεται στη θέση 1.

Η λειτουργία αναζήτησης χρησιμοποιείται για να πάρει τη θέση της επόμενης λειτουργίας ανάγνωσης ή εγγραφής.

## Παράδειγμα:



```
1. ' e.g. if you want to skip to
2. ' the 100th byte in the file
3. ' for reading/writing:
4. Dim f As Integer
5.
6. f = Freefile
7.
8. Open "file.ext" For Binary As #f
9. Seek f, 100
10. Close #f
```

## Seek (Function)

Παίρνει τη θέση της επόμενης λειτουργίας ανάγνωσης/εγγραφής για ένα αρχείο ή μια συσκευή.

### Σύνταξη:



```
Declare Function Seek ( ByVal filenum As Long ) As LongInt
```

### Παράμετροι:

#### filenum

αριθμός ενός ανοιχτού αρχείου

### Επιστροφή τιμής:

Η θέση του αρχείου όπου θα πραγματοποιηθεί η επόμενη λειτουργία ανάγνωσης ή εγγραφής.

### Περιγραφή:

Η θέση δίνεται σε εγγραφές εάν το αρχείο άνοιξε σε λειτουργία τυχαίας πρόσβασης, σε bytes σε οποιαδήποτε άλλη περίπτωση. Η θέση αρχείου που επιστρέφεται βασίζεται σε 1-n, οπότε η πρώτη εγγραφή ενός αρχείου είναι 1. Η δήλωση αναζήτησης χρησιμοποιείται για να ορίσει τη θέση της επόμενης λειτουργίας ανάγνωσης ή εγγραφής.

## Παράδειγμα:



```
1. Dim f As Integer, position As Integer
2. f = FreeFile
3. Open "file.ext" For Binary As #f
4. position = Seek(f)
5. Close #f
```

## Lock και Unlock

Η Lock περιορίζει την πρόσβαση ανάγνωσης/εγγραφής σε ένα αρχείο ή τμήμα ενός αρχείου

Η Unlock καταργεί έναν προηγούμενο περιορισμό πρόσβασης (κλείδωμα) σε ένα αρχείο

### Σύνταξη:



```
Lock #filenum, record
Lock #filenum, start To end
```

```
Unlock #filenum, record
Unlock #filenum, start To end
```

### Παράμετροι:

#### filenum

αριθμός ενός ανοιχτού αρχείου

#### record

Η εγγραφή (Random αρχεία) για κλείδωμα.

#### start

Η πρώτη θέση byte (Binary αρχεία) για να κλειδώσετε.

#### end

Η τελευταία θέση byte (Binary αρχεία) για να κλειδώσετε.

### Περιγραφή:

Το κλείδωμα περιορίζει προσωρινά την πρόσβαση άλλων νημάτων ή προγραμμάτων σε ένα αρχείο ή μέρος ενός

αρχείου, συνήθως για να επιτρέψει την ασφαλή εγγραφή σε αυτό.

Μετά την τροποποίηση των δεδομένων, θα πρέπει να κληθεί η Unlock με τις ίδιες παραμέτρους με το Lock.

Σημείωση: Αυτή η εντολή δεν λειτουργεί πάντα, ούτε όπως τεκμηριώθηκε ούτε όπως αναμενόταν. Αυτή τη στιγμή φαίνεται να μην λειτουργεί.

### Παράδειγμα:



```
1. ' e.g. locking a file,  
2. ' reading 100 bytes,  
3. ' and unlocking it.  
4. ' To run, make sure there exists  
5. ' a file called 'file.ext'  
6. ' in the current directory  
7. ' that is at least 100 bytes.  
8. Dim array(1 To 100) As Integer  
9. Dim f As Integer, i As Integer  
10. f = FreeFile  
11.  
12. Open "file.ext" For Binary As #f  
13.     Lock #f, 1 To 100  
14.     For i = 1 To 100  
15.         Get #f, i, array(i)  
16.     Next  
17.     Unlock #f, 1 To 100  
18. Close #f
```



# Μαθηματικές συναρτήσεις - Mathematical Functions

## Αλγεβρικές Συναρτήσεις

Συνάρτηση	Σύνταξη	Περιγραφή
Abs	result = Abs( number )	Επιστρέφει την απόλυτη τιμή ενός αριθμού.
Exp	result = Exp( number )	Επιστρέφει το e υψωμένο στη δύναμη ενός δεδομένου αριθμού
Log	result = Log( number )	Επιστρέφει το φυσικό λογάριθμο ενός δεδομένου αριθμού
Sqr	result = Sqr( number )	Επιστρέφει μια τετραγωνική ρίζα ενός αριθμού
Fix	result = Fix( number )	Επιστρέφει το ακέραιο μέρος ενός αριθμού, στρογγυλοποιώντας το μηδέν
Frac	result = Frac( number )	Επιστρέφει το δεκαδικό μέρος ενός αριθμού
Int	result = Int( number )	Επιστρέφει τον μεγαλύτερο ακέραιο που είναι μικρότερος ή ίσος με αυτόν
Sgn	result = Sgn( number )	Επιστρέφει το πρόσημο ενός αριθμού

## Παράδειγμα - algebraical.bas



```
1. Print Abs( -1 )
2. Print Exp(2)
3. Print Log(10)
4. Print Sqr(4)
5. Print Fix(1.9)
6. Print Frac(-10.625)
7. Print Int(-1.9)
8. Print Sgn ( -1.87 )
9.
10. Sleep
11. End
```

## Έξοδος:



```
1
7.38905609893065
2.302585092994046
2
1
-0.625
-2
-1
```

# Τριγωνομετρικές Συναρτήσεις

Συνάρτηση	Σύνταξη	Περιγραφή
Sin	result = Sin( angle )	Επιστρέφει το ημίτονο μιας γωνίας
Asin	result = Asin( number )	Βρίσκει το ημίτονο τόξου ενός αριθμού
Cos	result = Cos( angle )	Επιστρέφει το συνημίτονο μιας γωνίας
Acos	result = Acos( number )	Βρίσκει το συνημίτονο τόξου μιας γωνίας
Tan	result = Tan( angle )	Επιστρέφει την εφαπτομένη μιας γωνίας
Atn	result = Atn( number )	Επιστρέφει το τόξο εφαπτομένης ενός αριθμού
Atan2	result = ATan2( y, x )	Επιστρέφει την εφαπτομένη τόξου ενός λόγου

## Παράδειγμα - trigonometrical.bas



```
1. Print Sin(30)
2. Print Asin(0.5)
3. Print Cos(30)
4. Print Acos(0.5)
5. Print Tan(30)
6. Print Atn(30)
7. Print Atan2 ( 4, 5 )
8.
9. Sleep
10. End
```

## Έξοδος:



```
-0.9880316240928618  
0.5235987755982989  
0.154251449887584  
1.047197551196598  
-6.405331196646276  
1.537475330916649  
0.6747409422235526
```

## Συναρτήσεις Τυχαίων Αριθμών

### Randomize

Τροφοδοτεί τη γεννήτρια τυχαίων αριθμών.

### Σύνταξη:



Randomize [ seed ][, algorithm ]

### Παράμετροι:

#### seed

Μια τιμή διπλού σπόρου για τη γεννήτρια τυχαίων αριθμών, αλλά το κλασματικό μέρος περικόπτεται για όλους τους αλγόριθμους εκτός από τον αλγόριθμο #4 (βλ. Παρακάτω). Εάν παραλειφθεί, θα χρησιμοποιηθεί αντ' αυτού μια τιμή με βάση το χρονόμετρο (Timer).

#### algorithm

Μια ακέραη τιμή για την επιλογή του αλγορίθμου (δείτε παρακάτω, ή από την έκδοση `fbcs = 1.08` η τυπική κεφαλίδα `"fbmath.bi"` για τους διαθέσιμους αλγόριθμους). Εάν παραλειφθεί, χρησιμοποιείται ο προεπιλεγμένος αλγόριθμος για την τρέχουσα γλωσσική διάλεκτο.

## Περιγραφή:

Ορίζει τον τυχαίο σπόρο που βοηθά το Rnd να δημιουργήσει τυχαίους αριθμούς και επιλέγει τον αλγόριθμο που θα χρησιμοποιήσει.

Οι σταθερές για τον αλγόριθμο ορίζονται στο fbmath.bi (από την έκδοση fbc 1.08). Στη διάλεκτο -lang fb, αυτές οι σταθερές αποτελούν μέρος του FB Namespace.

Οι έγκυρες τιμές για τον αλγόριθμο είναι:

**FB\_RND\_AUTO (0)** - Προεπιλογή για την τρέχουσα διάλεκτο γλώσσας. Αυτός είναι ο αλγόριθμος **FB\_RND\_MTWIST (3)** στη διάλεκτο -lang fb, **FB\_RND\_QB (4)** στη διάλεκτο -lang qb και **FB\_RND\_CRT (1)** στη διάλεκτο -langl fblite.

**FB\_RND\_CRT (1)** - Χρησιμοποιεί τη συνάρτηση rand () της βιβλιοθήκης χρόνου εκτέλεσης C. Αυτό θα δώσει διαφορετικά αποτελέσματα ανάλογα με την πλατφόρμα.

**FB\_RND\_FAST (2)** - Χρησιμοποιεί γρήγορη εφαρμογή. Αυτό θα πρέπει να είναι σταθερό σε όλες τις πλατφόρμες και παρέχει λεπτομέρεια 32-bit, λογικό βαθμό τυχαιότητας.

**FB\_RND\_MTWIST (3)** - Χρησιμοποιεί το Mersenne Twister. Αυτό θα πρέπει να είναι σταθερό σε όλες τις πλατφόρμες, να παρέχει λεπτομέρειες 32-bit και να δίνει υψηλό βαθμό τυχαιότητας.

**FB\_RND\_QB (4)** - Χρησιμοποιεί μια συνάρτηση που έχει σχεδιαστεί για να δίνει τις ίδιες ακολουθίες τυχαίων αριθμών με την QBASIC. Αυτό θα πρέπει να είναι σταθερό σε όλες τις πλατφόρμες και παρέχει ακρίβεια 24-bit, με χαμηλό βαθμό τυχαιότητας.

**FB\_RND\_REAL (5)** - Διατίθεται σε Win32 και Linux, χρησιμοποιώντας δυνατότητες συστήματος (Win32 Crypto API, Linux /dev /urandom) για την παροχή κρυπτογραφικά τυχαίων αριθμών. Εάν αυτά τα API του συστήματος δεν

είναι διαθέσιμα, θα χρησιμοποιηθεί αντ 'αυτού ο αλγόριθμος **FB\_RND\_MTWIST (3)**.

Για κάθε δεδομένο σπόρο, κάθε αλγόριθμος θα παράγει μια συγκεκριμένη, ντετερμινιστική ακολουθία αριθμών για αυτόν τον σπόρο.

Εάν θέλετε κάθε κλήση στο Randomize να παράγει διαφορετική ακολουθία αριθμών, θα πρέπει να χρησιμοποιείται ένας σπόρος που δεν είναι αρκετά προβλέψιμος - για παράδειγμα, η τιμή που επιστρέφεται από το χρονόμετρο (Timer).

Η παράλειψη της παραμέτρου σπόρου θα χρησιμοποιήσει μια τιμή που βασίζεται σε αυτήν.

**Σημείωση:** για όλους τους αλγόριθμους εκτός από τον αλγόριθμο #4, επειδή το κλασματικό τμήμα του σπόρου είναι κομμένο, η χρήση της τιμής Χρονομέτρου απευθείας ως παραμέτρου θα παράγει τον ίδιο σπόρο εάν χρησιμοποιηθεί περισσότερες από μία φορές στο ίδιο δευτερόλεπτο.

Ωστόσο, γενικά δεν αξίζει να καλέσετε το Randomize δύο φορές με απρόβλεπτους σπόρους ούτως ή άλλως, επειδή η δεύτερη ακολουθία δεν θα είναι πιο τυχαία από την πρώτη, ή ακόμη πιθανώς χειρότερη προκαλώντας επικάλυψη αλληλουχίας.

Στις περισσότερες περιπτώσεις, το Misteren twister πρέπει να παρέχει μια αρκετά τυχαία ακολουθία αριθμών, χωρίς να απαιτείται επανασπορά μεταξύ κλήσεων Rnd.

Όταν καλείτε το Randomize με QB συμβατό αλγόριθμο, μέρος του παλιού σπόρου διατηρείται. Αυτό σημαίνει ότι αν καλέσετε Randomize πολλές φορές με τον ίδιο σπόρο, δεν θα έχετε την ίδια ακολουθία κάθε φορά. Για να λάβετε μια συγκεκριμένη ακολουθία σε λειτουργία συμβατή με QB, ορίστε τον σπόρο καλώντας το Rnd με αρνητική παράμετρο.

## Παράδειγμα - randomize.bas



```
1. ' Seed the RNG to the method using C's rand()
2. Randomize , 1
3. ' Print a sequence of random numbers
4. For i As Integer = 1 To 10
5.     Print Rnd
6. Next
7.
8. Sleep
9. End
```

### Έξοδος:



```
0.1859699254855514
0.5591087741777301
0.1771501135081053
0.9884689343161881
0.7918899091891944
0.5436976258642972
0.6241532918065786
0.2195916082710028
0.6672135572880507
0.6414299877360463
```

## Rnd

Επιστρέφει έναν τυχαίο αριθμό διπλής ακρίβειας στο εύρος [0, 1]

### Σύνταξη:



```
result = Rnd( seed )
```

## Παράμετροι:

### seed

Προαιρετικό όρισμα. Εάν ο σπόρος έχει τιμή μηδέν (0,0), επαναλαμβάνεται ο τελευταίος τυχαίος αριθμός που δημιουργήθηκε. Για οποιονδήποτε άλλο αριθμό επιστρέφεται ένας νέος τυχαίος αριθμός. Με τον αλγόριθμο συμβατό με QB, ένας αρνητικός αριθμός επανασυνδέει πλήρως τη γεννήτρια. Η προεπιλογή για κανένα όρισμα είναι η επιστροφή ενός νέου τυχαίου αριθμού.

## Επιστρεφόμενη τιμή:

Επιστρέφει τον τυχαίο αριθμό που δημιουργήθηκε.

## Περιγραφή:

Επιστρέφει έναν αριθμό τύπου Double στο εύρος [0, 1) (δηλ.  $0 \leq \text{Rnd} < 1$ ), με βάση έναν τυχαίο σπόρο (βλ. Randomize). Το Rnd μπορεί να χρησιμοποιήσει μια ποικιλία διαφορετικών αλγορίθμων - ανατρέξτε στην ενότητα Randomize για λεπτομέρειες των προεπιλεγμένων και επιλεγόμενων αλγορίθμων.

Το Rnd επιστρέφει την ίδια ακολουθία αριθμών κάθε φορά που εκτελείται ένα πρόγραμμα. Αυτή η ακολουθία μπορεί να αλλάξει με εκ νέου σπορά της γεννήτριας.

## Παράδειγμα - rnd.bas



```
1. Dim last As Integer
2. Dim first As Integer
3. Dim number As Integer
4.
5. first = 1
6. last = 10
7.
8. Randomize
9. For i As Integer = 1 To 10
10.     number = Rnd * (last - first) + first
11.     Print number
12. Next
13.
14. Sleep
15. End
```



## Έξοδος:



```
9  
2  
3  
7  
3  
6  
8  
6  
8  
2
```

## Συναρτήσεις μνήμης - Memory Functions

### Allocate

Διαθέτει ένα μπλοκ μνήμης από την ελεύθερη μνήμη.

### Σύνταξη:



```
result = Allocate( count )
```

### Παράμετροι:

#### count

Το μέγεθος, σε byte, του μπλοκ μνήμης που πρέπει να διατεθεί.

### Επιστρεφόμενη τιμή:

Εάν είναι επιτυχής, επιστρέφεται η διεύθυνση έναρξης της εκχωρημένης μνήμης. Διαφορετικά, εάν δεν ήταν δυνατή η κατανομή του ζητούμενου μεγέθους μπλοκ ή εάν το count είναι <0, τότε επιστρέφει ο μηδενικός δείκτης (0).

## **Περιγραφή:**

Προσπάθειες κατανομής ή κράτησης αριθμού byte από τον ελεύθερο σωρό. Η πρόσφατα κατανεμημένη μνήμη δεν αρχικοποιείται.

Δεδομένου ότι η αρχική τιμή της πρόσφατα κατανεμημένης μνήμης είναι απροσδιόριστη, το Allocate δεν πρέπει να χρησιμοποιείται απευθείας με συμβολοσειρά που περιέχει δεδομένα ή Udt, επειδή ο descriptor της συμβολοσειράς δεν διαγράφεται (περιέχει τυχαία δεδομένα), που μπορεί να προκαλέσει κατεστραμμένη συμβολοσειρά ή περισσότερο (προσπαθώντας να γράψετε σε τυχαία θέση στη μνήμη ή προσπάθεια ανακατανομής τυχαίου δείκτη).

Σε αυτήν την περίπτωση (με συμβολοσειρά ή συμβολοσειρά που περιέχεται σε UDT), είναι υποχρεωτική η χρήση της CAllocate (εκκαθάριση μνήμης) ή χρήση του New (κλήση συνάρτησης δόμησης - constructor) σε περίπτωση UDT ή, στη χειρότερη περίπτωση, μετά την εκχώρηση, για τη σαφή εκκαθάριση του descriptor (ρύθμιση στο 0) πριν τη πρώτη χρήση συμβολοσειράς.

Ο δείκτης που επιστρέφεται είναι Any Ptr και δείχνει την έναρξη της εκχωρημένης μνήμης. Αυτός ο δείκτης είναι εγγυημένος ότι είναι μοναδικός, ακόμη και αν ο αριθμός είναι μηδενικός.

Η εκχωρημένη μνήμη πρέπει να επιστραφεί στον ελεύθερο σωρό, με την χρήση της Deallocate όταν δεν χρειάζεται πλέον.

## Παράδειγμα - allocate.bas



```
1.  '' This program uses the ALLOCATE(...) function
2.  '' to create a buffer of 15 integers that is
3.  '' then filled with the first 15 numbers
4.  '' of the Fibonacci Sequence, then output to the
5.  '' screen.
6.  '' Note the call to DEALLOCATE(...)
7.  '' at the end of the program.
8.
9.  Const integerCount As Integer = 15
10.
11. '' Try allocating memory for a number of
12. '' integers.
13. ''
14. Dim buffer As Integer Ptr
15. buffer = Allocate(integerCount * SizeOf(Integer))
16. If (0 = buffer) Then
17.     Print "Error: unable to allocate memory,
18.         quitting."
19.     End -1
20. End If
21.
22. '' Prime and fill the memory with the fibonacci
23. '' sequence.
24. ''
25. buffer[0] = 0
26. buffer[1] = 1
27. For i As Integer = 2 To integerCount - 1
28.     buffer[i] = buffer[i - 1] + buffer[i - 2]
29. Next
30.
31. '' Display the sequence.
32. ''
33. For i As Integer = 0 To integerCount - 1
34.     Print buffer[i] ;
35. Next
36. Deallocate(buffer)
37. Sleep
End 0
```

### Έξοδος:



```
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

## CAllocate

Διαθέτει μνήμη για ορισμένο αριθμό στοιχείων από το ελεύθερο σωρό και διαγράφει το περιεχόμενο.

### Σύνταξη:



```
result = CAllocate( num_elements [, size ] )
```

### Παράμετροι:

num\_elements

Ο αριθμός των στοιχείων για τα οποία δεσμεύεται μνήμη.

size

Το μέγεθος, σε byte, κάθε στοιχείου.

### Επιστρεφόμενη τιμή:

Εάν είναι επιτυχής, επιστρέφεται η διεύθυνση έναρξης της εκχωρημένης μνήμης. Διαφορετικά, επιστρέφει ο μηδενικός δείκτης (0).

### Περιγραφή:

Το CAllocate αρχικοποιεί την εκχωρημένη μνήμη με μηδενικά.

Κατά συνέπεια, το CAllocate μπορεί επίσης να χρησιμοποιηθεί απευθείας με συμβολοσειρά ή με Udt που περιέχει συμβολοσειρά, επειδή ο περιγραφέας συμβολοσειράς διαγράφεται (ορίζεται στο 0) πρώτα.

Ομοίως, το CAllocate μπορεί επίσης να χρησιμοποιηθεί απευθείας με το ZString ή το WString επειδή τα δεδομένα συμβολοσειράς δημιουργούνται με μηδενικούς χαρακτήρες.

## Παράδειγμα - calloc.bas



```
1. ' Allocate and initialize
2. ' space for 10 integer elements.
3. Dim p As Integer Ptr = CAllocate(10,
   SizeOf(Integer))
4.
5. ' Fill the memory with integer values.
6. For index As Integer = 0 To 9
7.     p[index] = (index + 1) * 10
8. Next
9.
10. ' Display the integer values.
11. For index As Integer = 0 To 9
12.     Print p[index] ;
13. Next
14.
15. ' Free the memory.
16. Deallocate(p)
17.
18. Sleep
19. End
```

### Έξοδος:



```
10 20 30 40 50 60 70 80 90 100
```

## Reallocate

Ανακατανέμει τον αποθηκευτικό χώρο για ένα υπάρχον δεσμευμένο μπλοκ μνήμης.

### Σύνταξη:



```
result = Reallocate( pointer, count )
```

## **Παράμετροι:**

pointer

Η διεύθυνση της εκχωρημένης μνήμης που πρόκειται να ανακατανεμηθεί.

count

Ο αριθμός των byte, συνολικά, προς ανακατανομή.

## **Επιστρεφόμενη τιμή:**

Η διεύθυνση της ανακατανεμημένης μνήμης. Ένας μηδενικός (0) δείκτης επιστρέφεται εάν η ανακατανομή δεν ήταν επιτυχής και η αρχική μνήμη που δείχνει ο δείκτης παραμένει αμετάβλητη.

## **Περιγραφή:**

Προσπαθεί να ανακατανεμίσει ή να αλλάξει το μέγεθος, μνήμης που είχε εκχωρηθεί προηγουμένως με το Allocate ή το CAllocate. Τα περιεχόμενα του buffer διατηρούνται, αν και αν ο αριθμός είναι μικρότερος από το αρχικό μέγεθος του μπλοκ μνήμης, το buffer θα περικοπεί. Εάν το μέγεθος είναι αυξημένο, το πρόσθετο εύρος μνήμης δεν αρχικοποιείται σε τίποτα.

Όταν χρησιμοποιείτε το Reallocate, ο δείκτης αποτελέσματος πρέπει να αποθηκευτεί για να αποφευχθεί πιθανή διαρροή μνήμης, επειδή ο αρχικός δείκτης ενδέχεται να μην είναι πλέον έγκυρος μετά την ανακατανομή. Η τιμή του νέου δείκτη θα πρέπει να ελεγχθεί - εάν είναι 0, η ανακατανομή απέτυχε - ο αρχικός δείκτης παραμένει έγκυρος και η ποσότητα μνήμης που του αποδίδεται δεν έχει αλλάξει.

Η ανακατανεμημένη μνήμη πρέπει να ελευθερωθεί με το Deallocate όταν δεν χρειάζεται πλέον.

Εάν ο δείκτης είναι μηδενικός (0), τότε το ReAllocate συμπεριφέρεται πανομοιότυπα με το Allocate. Εάν ο δείκτης είναι έγκυρος και ο αριθμός είναι μηδενικός (0), τότε το ReAllocate συμπεριφέρεται παρόμοια με το Deallocate και επιστρέφει ένας μηδενικός δείκτης (0).

Εάν η μνήμη έχει προηγουμένως αφαιρεθεί από μια κλήση για Deallocate ή ReAllocate, η συμπεριφορά είναι απροσδιόριστη.

Κατά τη μη αυτόματη κατανομή μνήμης για περιγραφείς συμβολοσειρών (ή Udts που περιέχουν ένα), εάν ο αριθμός είναι μεγαλύτερος από το αρχικό μέγεθος του μπλοκ μνήμης, το νέο πρόσθετο εύρος μνήμης πρέπει να εκκαθαριστεί ρητά σε μηδενικά πριν από την πρώτη χρήση συμβολοσειράς (για παράδειγμα, χρησιμοποιώντας Clear ). Διαφορετικά, η πρόσβαση στη συμβολοσειρά θα προκαλέσει απροσδιόριστα αποτελέσματα (προσπάθεια εγγραφής ή ανάγνωσης σε τυχαία θέση στη μνήμη ή προσπάθεια αποεπιλογής τυχαίου δείκτη).

**ΣΗΜΕΙΩΣΗ:** Η ανακατανομή ενός δείκτη μέσα σε μια λειτουργία αντικειμένου, όταν αυτός ο δείκτης περιέχει το γονικό αντικείμενο της συνάρτησης, είναι απροσδιόριστη και πιθανότατα θα οδηγήσει σε φρικτά σφάλματα.

## Παράδειγμα - reallocate.bas



```
1. Dim a As Integer Ptr, b As Integer Ptr, i As Integer
2.
3. ' Allocate memory for 5 integers
4. a = Allocate( 5 * SizeOf(Integer) )
5.
6. If a = 0 Then Print "Error Allocating a": End
7.
8. For i = 0 To 4
9. ' Assign integers to the buffer
10. a[i] = (i + 1) * 2
11. Next i
12.
13. ' Reallocate memory for 5 additional integers
14. b = Reallocate( a, 10 * SizeOf(Integer) )
15.
16.
17. If b <> 0 Then
18. ' Discard the old pointer and use the new one
19. a = b
20.
21. For i = 5 To 9
22. ' Assign more integers to the buffer
23.
```

```

24.         a[i] = (i + 1) * 2
25.     Next i
26.
27.     For i = 0 To 9
28.         ' Print the integers
29.         Print i, a[i]
30.     Next i
31.     Print
32. Else
33.     '' Reallocate failed, memory unchanged
34.     Print "Error Reallocating a"
35.
36.     For i = 0 To 4
37.         ' Print the integers
38.         Print i, a[i]
39.     Next i
40.
41.     Print
42. End If
43. ' Clean up
44. Deallocate a
45.
46. Sleep
47. End

```

## Deallocate

Ελευθερώνει μνήμη που έχει δεσμευθεί προηγουμένως.

### Σύνταξη:



Deallocate( pointer )

### Παράμετροι:

pointer

η διεύθυνση του προηγουμένως δεσμευμένου buffer



## Περιγραφή:

Αυτή η διαδικασία απελευθερώνει τη μνήμη που είχε εκχωρηθεί προηγουμένως με το Allocate. Ο δείκτης πρέπει να είναι έγκυρος δείκτης.

Αφού επιστρέψει η διαδικασία, ο δείκτης θα καταστεί άκυρος (δείχνοντας μια μη έγκυρη διεύθυνση μνήμης) και η χρήση του (κατάργηση αναφοράς ή κλήση εκ νέου Deallocate) θα οδηγήσει σε απροσδιόριστη συμπεριφορά.

Όταν εκχωρήθηκε μνήμη για τη συγκράτηση ενός περιγραφέα συμβολοσειράς, η συμβολοσειρά πρέπει πάντα να καταστρέφεται (ρυθμίζεται σε " ") πριν από την εκχώρηση της περιγραφής συμβολοσειράς (επιτρέποντας την ανακατανομή της μνήμης που καταλαμβάνεται από τα δεδομένα συμβολοσειράς), διαφορετικά, δεν είναι δυνατή η ανακατανομή της αργότερα, και μπορεί να προκαλέσει διαρροή μνήμης στη συνέχεια του προγράμματος.

Η κλήση Deallocate σε μηδενικό δείκτη δεν προκαλεί καμία ενέργεια.

## Παράδειγμα - deallocate.bas



```
1.  ' initialize pointer
2.  ' to new memory address
3.  Dim As Integer Ptr integerPtr =
   Allocate( Len( Integer ) )
4.  ' use pointer
5.  *integerPtr = 420
6.  Print *integerPtr
7.  ' free memory back to system
8.  Deallocate( integerPtr )
9.  ' and zero the pointer
10. integerPtr = 0
11. End
```

## Peek

Λαμβάνει την τιμή ενός τύπου από μια διεύθυνση στη μνήμη

### Σύνταξη:



Peek( [ datatype, ] address )

### Παράμετροι:

#### datatype

Ο τύπος της τιμής που θα πάρει. Εάν παραλειφθεί, θεωρείται το UByte.

#### address

Η διεύθυνση στη μνήμη από την οποία λαμβάνετε την τιμή

### Περιγραφή:

Αυτή η διαδικασία επιστρέφει μια αναφορά στην τιμή της μνήμης που δίνεται από μια διεύθυνση μνήμης και είναι ισοδύναμη με:

\*cast (ubyte ptr, address)

ή

\*cast (data\_type ptr, address)

Έτσι, αυτή η λέξη -κλειδί μπορεί επίσης να χρησιμοποιηθεί για να εκχωρήσει μια τιμή σε μια θέση μνήμης, παρόμοια με το Poke.

**Σημείωση:** Όταν χρησιμοποιείτε το Peek, η επιλογή μεταγλωττιστή -exx δεν προσθέτει κώδικα για έλεγχο μηδενικού δείκτη (χωρίς έλεγχο ακυρότητας στην τιμή της διεύθυνσης).

## Παράδειγμα - poke\_peek.bas



```
1. ' define var and pointer
2. Dim i As Integer, p As Integer Ptr
3.
4. ' setup pointer
5. p = @i
6.
7. ' write a value to pointer
8. Poke Integer, p, 420
9.
10.
11. ' get a value from pointer
12. Print Peek(Integer, p)
13.
14. Sleep
15. End
```

## Έξοδος:



420

## Poke

Εκχωρεί μια τιμή σε μια θέση στη μνήμη.

## Σύνταξη:



Poke [ datatype, ] address, value

## Παράμετροι:

### datatype

Ο τύπος δεδομένων στην καθορισμένη διεύθυνση. Εάν παραλειφθεί, θεωρείται το UByte.

### address

Η θέση στη μνήμη για εκχώρηση.

### value

Η τιμή που θα εκχωρηθεί.

## Περιγραφή:

Το Poke εκχωρεί μια τιμή σε μια θέση στη μνήμη. Είναι ισοδύναμο με:

```
*cast(ubyte ptr, address) = value
```

ή

```
*cast(datatype ptr, address) = value
```

Όταν ο τύπος δεδομένων είναι ένας τύπος που ορίζεται από το χρήστη, το Poke εκχωρεί τιμή χρησιμοποιώντας τον τελεστή Let του τύπου.

Σημείωση: Όταν χρησιμοποιείτε το Poke, η επιλογή μεταγλωττιστή -exx δεν προσθέτει κώδικα για έλεγχο μηδενικού δείκτη.

## Παράδειγμα - poke\_peek.bas



```
1. ' define var and pointer
2. Dim i As Integer, p As Integer Ptr
3.
4. ' setup pointer
5. p = @i
6.
7. ' write a value to pointer
8. Poke Integer, p, 420
9.
10.
11. ' get a value from pointer
12. Print Peek(Integer, p)
13.
14. Sleep
15. End
```

## Έξοδος:



```
420
```

## Clear

Διαγράφει ή αρχικοποιεί κάποια μνήμη.

### Σύνταξη:



Clear( dst, [value], bytes )

### Παράμετροι:

dst

διεύθυνση εκκίνησης κάποιας μνήμης.

value

την τιμή για να ορίσετε όλα τα byte ίσα με αυτή.

bytes

αριθμός byte για διαγραφή.

### Περιγραφή:

Η Clear ορίζει ένα ή περισσότερα byte στη μνήμη σε μια συγκεκριμένη τιμή (η προεπιλεγμένη τιμή είναι μηδέν (0) εάν δεν έχει καθοριστεί).

Η διεύθυνση εκκίνησης λαμβάνεται από μια αναφορά σε μια μεταβλητή ή στοιχείο πίνακα.

**ΣΗΜΕΙΩΣΗ:** Για να καθαρίσετε τη μνήμη που αναφέρεται από έναν δείκτη, πρέπει πρώτα να καταργηθεί η παραπομπή (ή αλλιώς να ορίσετε τη λέξη -κλειδί ByVal μπροστά από το όνομα του δείκτη). Διαφορετικά, το Clear θα προσπαθήσει να διαγράψει τα byte στη θέση μνήμης της μεταβλητής δείκτη.

## Παράδειγμα - clear.bas



```
1. 'create an array with 100 elements
2. Dim array(0 To 99) As Integer
3.
4. 'clear the contents of the array to 0,
5. 'starting with the first element
6. Clear array(0), , 100 * SizeOf(Integer)
7.
8. 'allocate 20 bytes of memory
9. Dim As Byte Ptr p = Allocate(20)
10.
11.
12. 'set each of the first ten bytes to 0
13. Clear *p, 0, 10
14.
15. 'set each of the next ten bytes to 42
16. Clear p[10], 42, 10
17.
18. 'check the values of the allocated bytes
19. For i As Integer = 0 To 19
20.     Print i, p[i]
21. Next
22.
23. 'deallocate memory
24. Deallocate p
25. End
```

## **fb\_memcpy**

Αντιγράφει ένα μπλοκ μνήμης από μια τοποθεσία σε άλλη.

### **Σύνταξη:**



[result =] fb\_memcpy( dst, src, bytes )

### **Παράμετροι:**

dst

αρχική διεύθυνση της μνήμης προορισμού

src

αρχική διεύθυνση της αρχικής μνήμης

bytes

αριθμός byte για αντιγραφή

### **Επιστρεφόμενη τιμή:**

Επιστρέφει την αρχική διεύθυνση της μνήμης προορισμού.

### **Περιγραφή:**

Η fb\_memcpy αντιγράφει έναν δεδομένο αριθμό byte από τη θέση μνήμης src στη θέση μνήμης dst.

Κάθε διεύθυνση εκκίνησης λαμβάνεται από μια αναφορά σε μια μεταβλητή ή στοιχείο πίνακα.

Οι περιοχές μνήμης δεν πρέπει να επικαλύπτονται (διαφορετικά, η αντιγραφή δεν είναι εγγυημένη για τη σωστή λειτουργία, ειδικά ανάλογα με την πλατφόρμα).

Χρησιμοποιήστε το fb\_memmove κατά προτίμηση όταν οι περιοχές μνήμης επικαλύπτονται (ασφαλέστερη προσέγγιση).

Για την αποφυγή υπερχειλίσης, οι έγκυρες περιοχές μνήμης που επισημαίνονται τόσο από το src όσο και από το dst πρέπει να είναι τουλάχιστον ίσες σε μέγεθος με τον αριθμό των byte που πρέπει να αντιγραφούν.

Η συνάρτηση δεν ελέγχει για μηδενικό τερματικό χαρακτήρα στην περιοχή προέλευσης. Αντιγράφει πάντα ακριβώς τον δεδομένο αριθμό byte.

Το αποτέλεσμα είναι ένα δυαδικό αντίγραφο των δεδομένων.

**Σημείωση:** Για να αντιγράψετε από/στη μνήμη που αναφέρεται από έναν δείκτη, πρέπει πρώτα να γίνει κατάργησή του (ή αλλιώς να ορίσετε τη λέξη -κλειδί ByVal μπροστά από το όνομα του δείκτη). Διαφορετικά, το fb\_memcpy θα προσπαθήσει να αντιγράψει τα byte από/στη θέση μνήμης της μεταβλητής δείκτη.

## Παράδειγμα - fb\_memcpy.bas



```
1. ' Define a UDT
2. Type Person
3.     Dim As ZString * 40 Name
4.     Dim As Integer age
5. End Type
6.
7. ' Declare a pointer and UDT objects
8. Dim As ZString Ptr mynameptr = @"Pierre de
9.   Fermat"
10.
11. Dim As Person person1, person2
12.
13. ' using fb_memcpy to copy string
14. fb_memcpy(person1.name, *mynameptr,
15.   Len(*mynameptr) + 1)
16.
17. person1.age = 46
18.
19. ' using fb_memcpy to copy structure
20. fb_memcpy(person2, person1, SizeOf(Person))
21.
22. Print person2.name, person2.age
23. Sleep
24. End
```



## Έξοδος:



Pierre de Fermat

46

## fb\_MemCopyClear

Αντιγράφει το πρώτο μέρος ενός μπλοκ μνήμης από μια τοποθεσία σε άλλη και διαγράφει το υπόλοιπο.

## Σύνταξη:



`fb_memcopyclear( dst, dstlen, src, srclen )`

## Παράμετροι:

dst

αρχική διεύθυνση της μνήμης προορισμού

dstlen

αριθμός byte για εγγραφή

src

αρχική διεύθυνση της αρχικής μνήμης

srclen

αριθμός πρώτων byte για αντιγραφή (τα άλλα διαγράφονται)

## Περιγραφή:

Η `fb_memcopyclear` αντιγράφει έναν δεδομένο αριθμό byte (`dstlen`) από τη θέση μνήμης `src` στη θέση μνήμης `dst`, αλλά μόνο τα πρώτα `srclen` bytes αντιγράφονται πραγματικά και τα υπόλοιπα διαγράφονται (`(dstlen - srclen)` bytes).

Κάθε διεύθυνση εκκίνησης λαμβάνεται από μια αναφορά σε μια μεταβλητή ή στοιχείο πίνακα.

Οι περιοχές μνήμης δεν πρέπει να επικαλύπτονται (διαφορετικά, η αντιγραφή δεν είναι εγγυημένη για τη σωστή λειτουργία, ειδικά ανάλογα με την πλατφόρμα).

Για την αποφυγή υπερχειλίσης, οι έγκυρες περιοχές μνήμης που επισημαίνονται τόσο από το src όσο και από το dst πρέπει να είναι τουλάχιστον ίσες σε μέγεθος με τον αριθμό των byte που πρέπει να αντιγραφούν (συμπεριλαμβανομένων των διαγραμμένων byte).

Η συνάρτηση δεν ελέγχει για τυχόν μηδενικό χαρακτήρα τερματισμού στην περιοχή προέλευσης. Αντιγράφει πάντα ακριβώς τον δεδομένο αριθμό byte.

Το αποτέλεσμα είναι ένα δυαδικό αντίγραφο των δεδομένων για τα πρώτα srclen bytes και μηδενισμός για τα υπόλοιπα ((dstlen - srclen) byte).

## Παράδειγμα - fb\_memcopyclear.bas



```
1. Dim src As ZString * 10 = "FreeBASIC"
2. Dim dst As ZString * 10 = "012345678"
3.
4. Print "before:"
5. Print "src = " & src
6. Print "dst = " & dst
7. Print
8.
9.
10. ' copy first 4 bytes and clear the rest
11. fb_MemCopyClear(dst, SizeOf(dst), src, 4)
12.
13. Print "after:"
14. Print "src = " & src
15. Print "dst = " & dst
16. Sleep
17. End
```

## Έξοδος:



```
before:  
src = FreeBASIC  
dst = 012345678
```

```
after:  
src = FreeBASIC  
dst = Free
```

## fb\_memmove

Αντιγράφει ένα μπλοκ μνήμης από μια τοποθεσία σε άλλη.

### Σύνταξη:



```
[result =] fb_memmove( dst, src, bytes )
```

### Παράμετροι:

dst

αρχική διεύθυνση της μνήμης προορισμού

src

αρχική διεύθυνση της μνήμης πηγής

bytes

αριθμός byte για αντιγραφή

### Επιστρεφόμενη τιμή:

Επιστρέφει την αρχική διεύθυνση της μνήμης προορισμού.

### Περιγραφή:

fb\_memmove αντιγράφει έναν δεδομένο αριθμό byte από τη θέση μνήμης src στη θέση μνήμης dst. Κάθε διεύθυνση εκκίνησης λαμβάνεται από μια αναφορά σε μια μεταβλητή ή στοιχείο πίνακα.

Η αντιγραφή πραγματοποιείται σαν να χρησιμοποιήθηκε ένα ενδιάμεσο buffer, επιτρέποντας την επικάλυψη των περιοχών προορισμού και πηγής με οποιονδήποτε τρόπο (ασφαλέστερη προσέγγιση και για οποιαδήποτε πλατφόρμα).

Για την αποφυγή υπερχειλίσης, οι έγκυρες περιοχές μνήμης που επισημαίνονται τόσο από το src όσο και από το dst πρέπει να είναι τουλάχιστον ίσες σε μέγεθος με τον αριθμό των byte που πρέπει να αντιγραφούν.

Η συνάρτηση δεν ελέγχει για τυχόν μηδενικό χαρακτήρα τερματισμού στην περιοχή προέλευσης. Αντιγράφει πάντα ακριβώς τον δεδομένο αριθμό byte. Το αποτέλεσμα είναι ένα δυαδικό αντίγραφο των δεδομένων.

Σημείωση: Για να αντιγράψετε από/στη μνήμη που αναφέρεται από έναν δείκτη, πρέπει πρώτα να γίνει κατάργησή του (ή αλλιώς να ορίσετε τη λέξη -κλειδί ByVal μπροστά από το όνομα του δείκτη). Διαφορετικά, το fb\_memmove θα προσπαθήσει να αντιγράψει τα byte από/στη θέση μνήμης της μεταβλητής δείκτη.

## Παράδειγμα - fb\_memmove.bas



```
1. Dim As ZString * 33 z = "memmove can be very  
   useful....."  
2.  
3. Print z  
4.  
5. fb_memmove(z[20], z[15], 11)  
6.  
7. Print z  
8.  
9.  
10. Sleep  
11. End
```

## Swap

Ανταλλάσσει τις τιμές δύο μεταβλητών.

### Σύνταξη:



Swap a, b

### Παράμετροι:

a

Η τιμή της πρώτης μεταβλητής

b

Η τιμή της δεύτερης μεταβλητής.

### Περιγραφή:

Αντικαθιστά την τιμή δύο μεταβλητών, συμπεριλαμβανομένων των παρουσίων UDT (αλλάζει όλα τα μέλη δεδομένων).

**Σημείωση:** Όταν τα δεδομένα αναφέρονται από έναν δείκτη, μόνο ή μέσα σε μια περιγραφική δομή (ένα UDT, για παράδειγμα), το Swap ανταλλάσσει μόνο τις τιμές των δεικτών ή το περιεχόμενο των περιγραφικών δομών χωρίς πρόσβαση στα ίδια τα δεδομένα.

Για διαφορετικές συμβολοσειρές, το Swap ανταλλάσσει μόνο τους περιγραφείς των συμβολοσειρών αντί να ανακατανέμει τη μνήμη για την ανταλλαγή όλων των χαρακτήρων δεδομένων συμβολοσειρών.

Για τα UDT, το Swap απλώς ανταλλάσσει το περιεχόμενο των δομών, χωρίς να καλούνται τελεστές ή μέθοδοι.

## Παράδειγμα - swap.bas



```
1. ' using swap to swap 2 numbers:
2. Dim a As Integer, b As Integer
3.
4. Input "input a number: "; a
5. Input "input another number: "; b
6. Print "Before swap"
7. Print a, b
8.
9. Swap a, b
10.
11.
12. Print "After swap"
13. Print a, b
14.
15. Sleep
16. End
```

## Έξοδος:



```
input a number: ? 4
input another number: ? 5
Before swap
4      5
After swap
5      4
```

## SAdd

Επιστρέφει έναν δείκτη στα δεδομένα μιας μεταβλητής συμβολοσειράς.

## Σύνταξη:



result = SAdd( str )

## Παράμετροι:

str

η έκφραση συμβολοσειράς ή η μεταβλητή για να λάβετε τη διεύθυνση

## Επιστρεφόμενη τιμή:

Δείκτης των δεδομένων που σχετίζονται με το str.

## Περιγραφή:

Επιστρέφει τη μετατόπιση μνήμης των δεδομένων συμβολοσειράς στη μεταβλητή συμβολοσειράς.

## Παράδειγμα - sadd.bas



```
1. Dim s As String
2. Print SAdd(s)
3.
4. s = "hello"
5. Print "s=";s
6. Print SAdd(s)
7.
8. s = "abcdefg, 1234567, 54321"
9. Print "s=";s
10. Print SAdd(s)
11.
12.
13. Sleep
14. End
```

## Έξοδος:



```
0
s=hello
15318704
s=abcdefg, 1234567, 54321
15318704
```

# Συναρτήσεις λειτουργικού συστήματος - Operating System Functions

## **Exec**

Μεταφέρει προσωρινά την εκτέλεση σε εξωτερικό πρόγραμμα.

### **Σύνταξη:**



result = Exec( program, arguments )

### **Παράμετροι:**

#### program

Το όνομα αρχείου (συμπεριλαμβανομένης της διαδρομής αρχείου) του προγράμματος (εκτελέσιμο) για μεταφορά του ελέγχου σε αυτό.

#### arguments

Τα ορίσματα της γραμμής εντολών που πρέπει να περάσουν στο πρόγραμμα.

### **Επιστρεφόμενη τιμή:**

Η κατάσταση εξόδου του προγράμματος, είτε μείον ένα(-1), αν δεν μπορεί να εκτελεστεί το πρόγραμμα.

### **Περιγραφή:**

Μεταφέρει τον έλεγχο σε εξωτερικό πρόγραμμα. Όταν τελειώσει το πρόγραμμα, η εκτέλεση συνεχίζεται αμέσως μετά την κλήση στο Exec.



## Παράδειγμα - exec.bas



```
1. 'A Windows based example but
2. 'the same idea applies to Linux
3. Const exename = "NoSuchProgram.exe"
4. Const cmdline = "arg1 arg2 arg3"
5.
6. Dim result As Integer
7. result = Exec( exename, cmdline )
8.
9. If result = -1 Then
10.     Print "Error running "; exename
11. Else
12.     Print "Exit code:"; result
13. End If
```

## Chain

Μεταφέρει προσωρινά τον έλεγχο σε εξωτερικό πρόγραμμα.

### Σύνταξη:



```
result = Chain( program )
```

### Παράμετροι:

program

Το όνομα αρχείου (συμπεριλαμβανομένης της διαδρομής αρχείου) του προγράμματος (εκτελέσιμο) για μεταφορά του ελέγχου σε αυτό.

### Επιστρεφόμενη τιμή:

Επιστρέφει τον κωδικό εξόδου του εξωτερικού προγράμματος εάν εκτελεστεί με επιτυχία ή αρνητικό έναν (-1) διαφορετικά.

### Περιγραφή:

Μεταφέρει τον έλεγχο σε εξωτερικό πρόγραμμα. Όταν τελειώσει το πρόγραμμα, η εκτέλεση συνεχίζεται αμέσως μετά την κλήση στην Chain.

## Παράδειγμα - chain.bas



```
1. #ifdef __FB_LINUX__
2.     Dim As String program = "./program"
3. #else
4.     Dim As String program = "program.exe"
5. #endif
6.
7. Print "Running " & program & "... "
8. If (Chain(program) <> 0) Then
9.     Print program & " not found!"
10. End If
11.
12. Sleep
13. End
```

Ο λογικός έλεγχος #ifdef...#else...#endif γίνεται από τον προεπεξεργαστή που θα αναλύσουμε σε επόμενο κεφάλαιο.

## Run

Μεταφέρει την εκτέλεση σε εξωτερικό πρόγραμμα.

### Σύνταξη:



```
result = Run( program [, arguments ] )
```

### Παράμετροι:

#### program

Το όνομα αρχείου (συμπεριλαμβανομένης της διαδρομής αρχείου) του προγράμματος (εκτελέσιμο) για εκτέλεση αυτού.

#### arguments

Τα ορίσματα της γραμμής εντολών που πρέπει να περάσουν στο πρόγραμμα.

### Επιστρεφόμενη τιμή:

Επιστρέφει μείον ένα(-1), αν δεν μπορεί να εκτελεστεί το πρόγραμμα.

### Περιγραφή:

Μεταφέρει τον έλεγχο σε εξωτερικό πρόγραμμα. Όταν τελειώσει το πρόγραμμα, η εκτέλεση θα επιστρέψει στο σύστημα.

### Παράδειγμα - run.bas



```
1. ' Attempt to transfer control to
2. ' "program.exe" in the current directory.
3. Dim result As Integer = Run("program.exe")
4. ' at this point, "program.exe"
5. ' has failed to execute, and
6. ' result will be set to -1.
7.
8. Print result
9.
10. Sleep
11. End
```

### Kill

Διαγράφει ένα αρχείο από τον δίσκο / μέσο αποθήκευσης.

### Σύνταξη:



result = Kill( filename )

### Παράμετροι:

#### filename

Το όνομα αρχείου είναι το όνομα του αρχείου δίσκου για διαγραφή. Εάν το αρχείο δεν βρίσκεται στον τρέχοντα κατάλογο, η διαδρομή πρέπει επίσης να δοθεί ως διαδρομή/αρχείο.

## Επιστρεφόμενη τιμή:

Επιστρέφει μηδέν (0) σε επιτυχία ή μη μηδέν σε σφάλμα.

## Περιγραφή:

Το Kill διαγράφει ένα αρχείο από δίσκο / μέσο αποθήκευσης. Ο κωδικός σφάλματος που επιστρέφεται από το Kill μπορεί να ελεγχθεί χρησιμοποιώντας το Err στην επόμενη γραμμή. Η έκδοση λειτουργίας του Kill επιστρέφει απευθείας τον κωδικό σφάλματος ως Long 32 bit.

## Παράδειγμα - kill.bas



```
1. Dim filename As String = "file.ext"
2. Dim result As Integer = Kill( filename )
3. If result <> 0 Then Print "error trying to kill "
   ; filename ; " !"
4.
5. Sleep
6. End
```

## Name

Μετονομάζει ένα αρχείο στο δίσκο.

## Σύνταξη:



```
result = Name( oldname, newname )
```

## Παράμετροι:

oldname

Όνομα υπάρχοντος αρχείου.

newname

Νέο όνομα του αρχείου.

## Επιστρεφόμενη τιμή:

Επιστρέφει μηδέν (0) στην επιτυχία και μη μηδέν στην αποτυχία.

## Περιγραφή:

Μετονομάζει ένα αρχείο ή φάκελο που αρχικά ονομαζόταν παλιό όνομα σε νέο όνομα.

## Παράδειγμα - name.bas



```
1. Dim OldName As String
2. Dim NewName As String
3. Dim result As Integer
4.
5. OldName = "dsc001.jpg"
6. NewName = "landscape.jpg"
7.
8.
9. result = Name( OldName, NewName )
10.
11. If 0 <> result Then
12.     Print "error renaming " & oldname & " to " &
        newname & "."
13. End If
14.
15. Sleep
    End
```

## FileAttr

Επιστρέφει πληροφορίες σχετικά με έναν ανοιχτό αριθμό αρχείου

## Σύνταξη:



```
#include "file.bi"
result = FileAttr( filename, [ returntype ] )
```

ή

```
#include "vbcompat.bi"
result = FileAttr( filename, [ returntype ] )
```

## Παράμετροι:

### filenum

Ο αριθμός αρχείου ενός αρχείου ή συσκευής που άνοιξε με το Open.

### returntype

Μια ακέραιη τιμή που υποδεικνύει τον τύπο των πληροφοριών που πρέπει να επιστρέψουν.

## Επιστρεφόμενη τιμή:

Μια τιμή που σχετίζεται με τον τύπο επιστροφής, διαφορετικά 0 στο σφάλμα.

## Περιγραφή:

Οι πληροφορίες σχετικά με τον αριθμό αρχείου επιστρέφονται με βάση τον παρεχόμενο returntype.

Value	Description	constant
1	File Mode	fbFileAttrMode
2	File Handle	fbFileAttrHandle
3	Encoding	fbFileAttrEncoding

Για File Mode, returntype = 1 (fbFileAttrMode) η τιμή επιστροφής είναι το άθροισμα μιας ή περισσότερες από τις ακόλουθες τιμές:

Value	File Mode	Constant
1	Input	fbFileModeInput
2	Output	fbFileModeOutput
4	Random	fbFileModeRandom
8	Append	fbFileModeAppend
32	Binary	fbFileModeBinary

Για το File Handle, returntype = 2 (fbFileAttrHandle), η τιμή επιστροφής είναι το file handle όπως παρέχεται από την C Runtime για συσκευές τύπου αρχείου.

Μόνο στα Windows: Για το File Handle, returntype = 2 (fbFileAttrHandle), η τιμή που επιστρέφεται για συσκευές COM είναι το handle που επιστρέφει το CreateFile () κατά το πρώτο άνοιγμα της συσκευής.

Η τιμή που επιστρέφεται για συσκευές LPT είναι το handle που επιστρέφεται από το OpenPrinter () κατά το πρώτο άνοιγμα της συσκευής. Αυτή η τιμή handle μπορεί να περάσει σε άλλες λειτουργίες API των Windows.

Μόνο για Linux: Για το File Handle, returntype = 2 (fbFileAttrHandle), η τιμή που επιστρέφεται για συσκευές COM είναι ο περιγραφέας αρχείων που επιστρέφεται με ανοιχτό () όταν άνοιξε για πρώτη φορά η συσκευή.

Για το Encoding, returntype = 3 (fbFileAttrEncoding), η τιμή επιστροφής είναι μία από τις ακόλουθες τιμές:

Value	Endoding	Constant
0	Ascii	fbFileEncodASCII
1	UTF-8	fbFileEncodUTF8
2	UTF-16	fbFileEncodUTF16
3	UTF-32	fbFileEncodUTF32

## Παράδειγμα - fileattr.bas



```
1. #include "vbcompat.bi"
2. #include "crt.bi"
3.
4. Dim f As FILE Ptr, i As Integer
5. ' Open a file and write some text to it
6. Open "test.txt" For Output As #1
   f = Cast( FILE Ptr, FileAttr( 1, fbFileAttrHandle
   ))
7. For i = 1 To 10
8.     fprintf( f, !"Line %i\n", i )
9. Next i
10. Close #1
11. ' re-open the file and read the text back
12. Open "test.txt" For Input As #1
   f = Cast( FILE Ptr, FileAttr( 1, fbFileAttrHandle
   ))
```

```

13.      While feof(f) = 0
14.          i = fgetc(f)
15.          Print Chr(i);
16.      Wend
17.      Close #1
18.      Sleep
19.      End

```

## FileCopy

Αντιγράφει ένα αρχείο.

### Σύνταξη:



```

#include "file.bi"
FileCopy source, destination

```

ή

```

#include "file.bi"
result = FileCopy( source, destination )

```

### Παράμετροι:

#### source

Ένα όρισμα συμβολοσειράς που καθορίζει το όνομα αρχείου του αρχείου από το οποίο θα αντιγραφεί. Αυτό το αρχείο πρέπει να υπάρχει.

#### destination

Ένα όρισμα συμβολοσειράς που καθορίζει το όνομα αρχείου στο οποίο θα αντιγραφεί. Αυτό το αρχείο θα αντικατασταθεί εάν υπάρχει. Αυτό το αρχείο δεν πρέπει να αναφέρεται επί του παρόντος από οποιαδήποτε ανοιχτή λαβή αρχείων.

### Επιστρεφόμενη τιμή:

Επιστρέφει το 0 στην επιτυχία ή 1 εάν παρουσιάστηκε σφάλμα.



## Περιγραφή:

Αντιγράφει το περιεχόμενο του αρχείου προέλευσης στο αρχείο προορισμού, αντικαθιστώντας το αρχείο προορισμού εάν υπάρχει ήδη.

Είναι απαραίτητο να συμπεριλάβετε είτε το "file.bi" είτε το "vbcompat.bi" για να αποκτήσετε πρόσβαση σε αυτήν τη λειτουργία.

Ο κωδικός σφάλματος που επιστρέφεται από το FileCopy μπορεί να ελεγχθεί χρησιμοποιώντας το Err στην επόμενη γραμμή.

Η έκδοση συνάρτησης του FileCopy επιστρέφει απευθείας τον κωδικό σφάλματος ως Long 32 bit.

## Παράδειγμα - filecopy.bas



```
1. | #include "file.bi"  
2. | FileCopy "source.txt", "destination.txt"  
3. | End
```

## FileDateTime

Επιστρέφει την τελευταία τροποποιημένη ημερομηνία και ώρα ενός αρχείου ως σειριακή ημερομηνία.

## Σύνταξη:



```
#include "file.bi"  
result = FileDateTime( filename )
```

ή

```
#include "vbcompat.bi"  
result = FileDateTime( filename )
```

## Παράμετροι:

filename

Όνομα αρχείου για ανάκτηση ημερομηνίας και ώρας.

## Επιστρεφόμενη τιμή:

Επιστρέφει μια σειριακή ημερομηνία.

## Περιγραφή:

Επιστρέφει την τελευταία τροποποιημένη ημερομηνία και ώρα του αρχείου ως σειριακή ημερομηνία.

## Παράδειγμα - filedatetime.bas



```
4. #include "vbcompat.bi"
5.
6. Dim filename As String, d As Double
7.
8. Print "Enter a filename: "
9. Line Input filename
10.
11.
12. If FileExists( filename ) Then
13.     Print "File last modified: ";
14.     d = FileDateTime( filename )
15.     Print Format( d, "yyyy-mm-dd hh:mm AM/PM" )
16. Else
17.     Print "File not found"
18. End If
19.
20. Sleep
21. End
```

## Έξοδος:



```
Enter a filename:
file.ext
File last modified: 2021-08-07 04:25 PM
```

## FileExists

Ελέγχει την ύπαρξη αρχείου.

### Σύνταξη:



```
#include "file.bi"
result = FileExists( filename )
ή
#include "vbcompat.bi"
result = FileExists( filename )
```

### Παράμετροι:

filename

Το όνομα του αρχείου, το οποίο θα ελεγχθεί αν υπάρχει.

### Επιστρεφόμενη τιμή:

Επιστρέφει μη μηδέν (-1) εάν υπάρχει το αρχείο, διαφορετικά επιστρέφει μηδέν (0).

### Περιγραφή:

Το FileExists ελέγχει την ύπαρξη ενός αρχείου.

Εσωτερικά, μπορεί να εκδώσει μια λειτουργία Open () και ένα Close (), η οποία μπορεί να έχει συνέπειες - π.χ., οποιαδήποτε υπάρχοντα κλειδώματα στο αρχείο μπορεί να απελευθερωθούν.

Ανάλογα με τις ακριβείς απαιτήσεις, εναλλακτικές μέθοδοι ελέγχου ύπαρξης αρχείου μπορεί να είναι η χρήση της συνάρτησης Dir () (προσοχή στις ιδιότητες και η διασφάλιση ότι η διαδρομή δεν περιέχει μπαλαντέρ), ή το άνοιγμα του αρχείου και ο έλεγχος της τιμής επιστροφής για επιτυχία.

## Παράδειγμα - fileexists.bas



```
1. #include "vbcompat.bi"
2.
3. Dim filename As String
4. Print "Enter a filename: "
5. Line Input filename
6.
7. If FileExists( filename ) Then
8.     Print "File found: " & filename
9. Else
10.    Print "File not found: " & filename
11. End If
12.
13. Sleep
14. End
```

### Έξοδος:



```
Enter a filename:
file.ext
File found: file.ext
```

## FileLen

Βρίσκει το μήκος ενός αρχείου με το όνομα του αρχείου.

### Σύνταξη:



```
#include "file.bi"
result = FileLen(filename)
ή
#include "vbcompat.bi"
result = FileLen(filename)
```

### Παράμετροι:

filename

Το όνομα του αρχείου για το οποίο θα βρεθεί το μήκος του.

### Επιστρεφόμενη τιμή:

Επιστρέφει το μέγεθος σε byte του αρχείου που καθορίζεται από το όνομα αρχείου.

### Παράδειγμα - filelen.bas



```
1. #include "file.bi"  
2. Dim length As Integer  
3. length = FileLen("file.ext")  
4.  
5. Print "file.ext has length: " ; length  
6.  
7. Sleep  
8. End
```

### Έξοδος:



```
file.ext has length: 6
```

### FileSetEof

Ορίζει το μήκος ενός ανοιχτού αρχείου που είναι συνδεδεμένο με έναν αριθμό αρχείου.

### Σύνταξη:



```
#include "file.bi"  
result = FileSetEof(fnum)
```

### Παράμετροι:

fnum

Αριθμός αρχείου συνδεδεμένου αρχείου ή συσκευής.

### Επιστρεφόμενη τιμή:

Επιστρέφει το μηδέν (0) για επιτυχία ή έναν κωδικό σφάλματος εάν δεν ήταν δυνατή η ρύθμιση του τέλους του αρχείου (μέγεθος αρχείου).

## Περιγραφή:

Το FileSetEof ορίζει το τέλος του αρχείου με βάση την τρέχουσα θέση του αρχείου. Η θέση του αρχείου όπως στην Αναζήτηση είναι one based.

Όταν η τρέχουσα θέση αρχείου είναι πριν από το τέλος του αρχείου, το αρχείο περικόπτεται.

Τα περιεχόμενα του αρχείου πριν από την τρέχουσα θέση αρχείου διατηρούνται και τα περιεχόμενα του αρχείου πάνω ή μετά από την τρέχουσα θέση του αρχείου διαγράφονται.

Όταν η τρέχουσα θέση είναι πέρα από το τέλος του αρχείου, το αρχείο επεκτείνεται με byte μηδενικής τιμής. Αφού ολοκληρωθεί το FileSetEof, η τρέχουσα θέση αρχείου βρίσκεται στο τέλος του αρχείου.

Για να ορίσετε ένα αρχείο με μήκος N-byte όπου το αρχείο ανοίγει για δυαδική, έξοδο ή προσάρτηση, είναι απαραίτητο να αναζητήσετε τη θέση N-bytes + 1. Για να ορίσετε ένα αρχείο με μήκος N-εγγραφών όπου το αρχείο ανοίγει τυχαία, είναι απαραίτητο να αναζητήσετε τη θέση N-records + 1.

## Παράδειγμα:



```
1. #include "file.bi"
2.
3. ' create a zero length file
4. Open "file.dat" For Binary As #1
5. FileSetEof 1
6. Close #1
7. ' open same file and extend to 10000 bytes size
8. Open "file.dat" For Binary As #1
9. Seek #1, (10000 + 1)
10. FileSetEof 1
11. Close #1
12. ' open same file and truncate to 5000 bytes size
13. Open "file.dat" For Binary As #1
14. Seek #1, (5000 + 1)
15. FileSetEof 1
16. Close #1
17. ' clean-up
Kill "file.dat"
```

## FileFlush

Αδειάζει τα buffer ροής εφαρμογής στο σύστημα ή τα buffer συστήματος στο αρχείο.

### Σύνταξη:



```
#include "file.bi"  
result = FileFlush()  
result = FileFlush( filenum )  
result = FileFlush( filenum, systembuffers )
```

### Παράμετροι:

#### filenum

Αριθμός συνδεδεμένου αρχείου ή συσκευής. Εάν δεν δίνεται ή δίνεται -1, τότε αδειάζονται όλα τα ανοιχτά αρχεία.

#### systembuffers

Εάν δεν είναι μηδενική, αδειάζει τα buffer συστήματος στη φυσική συσκευή. Η προεπιλογή είναι μηδέν (0).

### Επιστρεφόμενη τιμή:

Επιστρέφει το μηδέν (0) για επιτυχία ή έναν κωδικό σφάλματος εάν δεν ήταν δυνατή το άδειασμα των προσωρινών προσωρινών αρχείων.

### Περιγραφή:

Το FileFlush εγγράφει την έξοδο της προσωρινής μνήμης της εφαρμογής στην υποκείμενη ροή και εάν τα σφάλματα συστήματος δεν είναι μηδενικά, στην υποκείμενη φυσική συσκευή επίσης.

## Παράδειγμα:



```
1. #include "file.bi"
2.
3. Dim As Long f1, f2
4. Dim As String s
5.
6.
7. Print "File length", "File string"
8.
9. f1 = Freefile
10. Open "fileflushtest.txt" For Output As #f1
11. Print #f1, "successful file flush"
12.
13. f2 = Freefile
14. Open "fileflushtest.txt" For Input As #f2
15. Line Input #f2, s
16. Print FileLen("fileflushtest.txt"), "" & s & ""
17. ' the string is not yet physically written to
   the file
18.
19. FileFlush(f1)
20.
21. Line Input #f2, s
22. Print FileLen("fileflushtest.txt"), "" & s & ""
23. ' the string is now physically written to the
   file
24.
25. Close #f2
26. Close #f1
27.
28. Sleep
29. End
```

## Έξοδος: (Windows)



```
File length  File string
0           "
23          'successful file flush'
```



# Συναρτήσεις συμβολοσειρών - String Functions

Αυτές οι δηλώσεις και οι διαδικασίες παρέχουν πολλούς τρόπους για τη δημιουργία και τον χειρισμό συμβολοσειρών και μέρη συμβολοσειρών. Οι αριθμοί μπορούν να μετατραπούν σε συμβολοσειρές και αντίστροφα. Παρέχονται επίσης διαδικασίες για να βοηθήσουν στη σειριοποίηση αριθμητικών δεδομένων, ίσως για συνεχή αποθήκευση.

## String (Function)

Δημιουργεί και συμπληρώνει μια συμβολοσειρά συγκεκριμένου μήκους με συγκεκριμένο χαρακτήρα.

### Σύνταξη:



```
result = String[$]( count, ch_code )  
ή  
result = String[$]( count, ch )
```

### Παράμετροι:

#### count

Ένας ακέραιος αριθμός που καθορίζει το μήκος της συμβολοσειράς που πρόκειται να δημιουργηθεί.

#### ch\_code

Ένας Long που προσδιορίζει τον κωδικό χαρακτήρα ASCII που θα χρησιμοποιηθεί για να συμπληρώσει τη συμβολοσειρά.

#### ch

Μια συμβολοσειρά της οποίας ο πρώτος χαρακτήρας πρόκειται να χρησιμοποιηθεί για να γεμίσει τη συμβολοσειρά.

### Επιστρεφόμενη τιμή:

Η συμβολοσειρά που δημιουργήθηκε. Μια κενή συμβολοσειρά θα επιστραφεί εάν είτε το `ch` είναι μια κενή συμβολοσειρά είτε το `count <= 0`.

### Περιγραφή:

Η συνάρτηση `String` επιστρέφει μια συμβολοσειρά με μήκος όσο το `count`, γεμάτη με έναν δεδομένο κωδικό χαρακτήρα ASCII ή ένα χαρακτήρα από συμβολοσειρά.

### Παράδειγμα:



```
1. Print String( 4, 69 )      '' prints "EEEE"
2. Print String( 5, "Indeed" )  '' prints "IIIII"
3. End 0
```

### Έξοδος:



```
EEEE
IIII
```

## Wstring (Function)

Δημιουργεί και συμπληρώνει μια συμβολοσειρά συγκεκριμένου μήκους με συγκεκριμένο χαρακτήρα UNICODE.

### Σύνταξη:



```
result = WString( count, ch_code )
ή
result = WString( count, ch )
```

## Παράμετροι:

### count

Ένας ακέραιος αριθμός που καθορίζει το μήκος της συμβολοσειράς που πρόκειται να δημιουργηθεί.

### ch\_code

Ένας Long που προσδιορίζει τον κωδικό χαρακτήρα UNICODE που θα χρησιμοποιηθεί για να συμπληρώσει τη συμβολοσειρά.

### ch

Μια συμβολοσειρά Wstring της οποίας ο πρώτος χαρακτήρας πρόκειται να χρησιμοποιηθεί για να γεμίσει τη συμβολοσειρά.

## Επιστρεφόμενη τιμή:

Η συμβολοσειρά Wstring που δημιουργήθηκε. Μια κενή συμβολοσειρά θα επιστραφεί εάν είτε το ch είναι μια κενή συμβολοσειρά είτε το count <= 0.

## Περιγραφή - wstring.bas

Η συνάρτηση WString επιστρέφει μια συμβολοσειρά με μήκος όσο το count, γεμάτη με έναν δεδομένο κωδικό χαρακτήρα UNICODE ή ένα χαρακτήρα από συμβολοσειρά WString.

## Παράδειγμα:



```
1. Print WString( 4, 934 )  
2. Print WString( 5, WStr("Indeed") )  
3. End 0
```

## Έξοδος:



```
ΦΦΦΦ  
IIII
```

## Space

Δημιουργεί μια συμβολοσειρά ενός δεδομένου μήκους γεμάτη με κενά (" ").

### Σύνταξη:



```
result = Space[$]( count )
```

### Παράμετροι:

count

Ένας ακέραιος που καθορίζει το μήκος της συμβολοσειράς που πρόκειται να δημιουργηθεί.

### Επιστρεφόμενη τιμή:

Η συμβολοσειρά που δημιουργήθηκε. Μια κενή συμβολοσειρά θα επιστραφεί εάν το `count <= 0`.

### Περιγραφή:

Το `Space` δημιουργεί μια συμβολοσειρά με τον καθορισμένο αριθμό κενών.

### Παράδειγμα - space.bas



```
1. Dim a As String
2. a = "x" + Space(3) + "x"
3. Print a
4.
5. Sleep
6. End
```

### Έξοδος:



```
x x
```

## WSpace

Δημιουργεί ένα WString δεδομένου μήκους γεμάτη με κενά (" ").

### Σύνταξη:



```
result = WSpace( count )
```

### Παράμετροι:

count

Ένας ακέραιος που καθορίζει το μήκος της συμβολοσειράς WString που πρόκειται να δημιουργηθεί.

### Επιστρεφόμενη τιμή:

Η συμβολοσειρά WString που δημιουργήθηκε. Μια κενή συμβολοσειρά WString θα επιστραφεί εάν το count <= 0.

### Περιγραφή:

Το WSpace δημιουργεί μια συμβολοσειρά WString με τον καθορισμένο αριθμό κενών.

### Παράδειγμα - Wspace.bas



```
1. Dim a As WString * 10
2. a = "x" + WSpace(3) + "x"
3. Print a
4.
5. Sleep
6. End
```

### Έξοδος:



```
x x
```

## Len

Επιστρέφει το μήκος μιας έκφρασης ή ενός τύπου δεδομένων.

### Σύνταξη:



result = Len( expression )  
ή  
result = Len( DataType )

### Παράμετροι:

expression

Μια έκφραση οποιουδήποτε τύπου.

DataType

Ένας Τύπος Δεδομένων.

### Επιστρεφόμενη τιμή:

Επιστρέφει το μέγεθος μιας έκφρασης ή DataType (και τα πεδία δεδομένων ενός UDT για έκδοση fbc> = 1.08) σε byte.

### Περιγραφή:

Το Len επιστρέφει το μήκος μιας έκφρασης ή το μέγεθος ενός DataType, σε byte.

Στην πρώτη μορφή, εάν η έκφραση είναι τύπου String, WString ή ZString, το μήκος της συμβολοσειράς σε χαρακτήρες επιστρέφεται.

Εάν η έκφραση είναι τύπου που ορίζεται από το χρήστη, καλείται ο τελεστής Len συμβατός με αυτόν τον τύπο δεδομένων. Διαφορετικά, επιστρέφεται το μέγεθος του τύπου δεδομένων της έκφρασης σε byte.

Στη δεύτερη μορφή, εάν η έκφραση είναι ZString ή WString, επιστρέφεται το μέγεθος σε byte ενός χαρακτήρα ASCII ή Unicode, αντίστοιχα.

Εάν ο τύπος δεδομένων είναι String, επιστρέφεται το μέγεθος του τύπου περιγραφής συμβολοσειράς.

Για έκδοση fbc <1.08: Όταν μια μεταβλητή από έναν δεδομένο χώρο ονομάτων (namespace) έχει πρόσβαση με το πρόθεμα ονόματος του χώρου ονομάτων, περικλείει το όρισμα στο Len με παρενθέσεις για να το αναγκάσει να θεωρηθεί ως έκφραση.

Για παράδειγμα Len ((namespace\_name.variable)).

Εάν υπάρχει τόσο ένας τύπος που ορίζεται από το χρήστη όσο και μια μεταβλητή ορατή με το ίδιο όνομα στο τρέχον πεδίο, ο τύπος που ορίζεται από τον χρήστη έχει προτεραιότητα έναντι της μεταβλητής.

Για να διασφαλίσετε ότι το Len παίρνει τη μεταβλητή αντί για τον τύπο που ορίζεται από το χρήστη, περικλείετε το όρισμα στο Len με παρενθέσεις για να το αναγκάσετε να θεωρηθεί ως έκφραση.

Για παράδειγμα Len ((μεταβλητή)).

Ο Len unary τελεστής μπορεί να υπερφορτωθεί με τύπους που ορίζονται από το χρήστη.

## Παράδειγμα- len.bas



```
1. Print Len("hello world") 'returns "11"  
2.  
3. Print Len(Integer) ' returns 4  
4.  
5.   
6. Type xyz  
7.     a As Integer  
8.     b As Integer  
9. End Type  
10. Print Len(xyz) ' returns 8  
11.  
12. Sleep  
13. End
```

## Έξοδος:



```
11  
8  
16
```

## Asc

Επιστρέφει την αντίστοιχη ακέραιη αναπαράσταση ASCII ή Unicode ενός χαρακτήρα.

### Σύνταξη:



```
result = Asc( str [, position ] )
```

### Παράμετροι:

str

Η πηγαία συμβολοσειρά.

position

Η θέση στη συμβολοσειρά ενός χαρακτήρα.

### Επιστρεφόμενη τιμή:

Η ακατέργαστη τιμή χαρακτήρων που αποθηκεύεται στη θέση στο str.

Εάν και το str και η θέση μπορούν να αξιολογηθούν κατά τη μεταγλώττιση (όπως Asc ("a") ή Asc (chr (97)) ή Asc ("abc", 2) ...), η τιμή επιστρέφεται σε αποτέλεσμα UInteger, αλλιώς σε ULong αποτέλεσμα.

### Περιγραφή:

Εάν το str είναι String ή ZString, επιστρέφεται η τιμή UByte σε αυτήν τη θέση. Αυτός θα είναι ένας κωδικός ASCII 7-bit, ή ακόμη και μια τιμή χαρακτήρα 8-bit από κάποια code-page, ανάλογα με τα δεδομένα συμβολοσειράς που είναι αποθηκευμένα στο str.

Εάν το str είναι WString, επιστρέφεται η τιμή UShort (Windows) ή ULong (Linux) σε αυτήν τη θέση. Αυτή θα είναι μια τιμή 16bit στα Windows (τα WStrings χρησιμοποιούν UTF16 εκεί) ή μια τιμή 32bit στο Linux (τα WStrings χρησιμοποιούν UTF32 εκεί).

Η συνάρτηση επιστρέφει μηδέν (0) εάν η συμβολοσειρά είναι συμβολοσειρά μηδενικού μήκους, η θέση είναι μικρότερη από



μία (1) ή η θέση είναι μεγαλύτερη από τον αριθμό των χαρακτήρων στο str.

Το Chr εκτελεί την αντίθετη συνάρτηση για τις συμβολοσειρές ASCII, ενώ το WChr είναι το αντίθετο για τις συμβολοσειρές Unicode, επιστρέφοντας μια συμβολοσειρά που περιέχει τον χαρακτήρα που παριστάνεται από τον κώδικα που έχει περάσει ως όρισμα.

### Παράδειγμα: - asc.bas



```
1. Print "the ascii code of 'a' is: "; Asc("a")
2. Print "the ascii code of 'b' is: "; Asc("abc", 2)
3.
4. Sleep
5. End
```

### Έξοδος:



```
the ascii code of 'a' is:97
the ascii code of 'b' is:98
```

## Chr

Επιστρέφει μια συμβολοσειρά χαρακτήρων από μία ή περισσότερες ακέραιες τιμές ASCII.

### Σύνταξη:



```
result = Chr[$]( ch0 [, ch1 ... chN ] )
```

### Παράμετροι:

ch

Η ακέραιη τιμή ενός χαρακτήρα ASCII.

## Επιστρεφόμενη τιμή:

Επιστρέφει μια συμβολοσειρά που περιέχει τους χαρακτήρες.

## Περιγραφή:

Το Chr επιστρέφει μια συμβολοσειρά που περιέχει τους χαρακτήρες που αντιπροσωπεύονται από τις τιμές ASCII που μεταβιβάζονται σε αυτήν.

Όταν το Chr χρησιμοποιείται με αριθμητικές σταθερές, το αποτέλεσμα αξιολογείται κατά τον χρόνο μεταγλώττισης, ώστε να μπορεί να χρησιμοποιηθεί σε μεταβλητές για αρχικοποιήσεις.

Το Asc εκτελεί την αντίθετη συνάρτηση, επιστρέφοντας τον κωδικό ASCII ενός χαρακτήρα που αντιπροσωπεύεται από μια συμβολοσειρά.

## Παράδειγμα - chr.bas



```
1. Print "the character represented by";
2. Print " the ASCII code of 97 is: "; Chr(97)
3. Print Chr(97, 98, 99) ' prints abc
4.
5. ' s initially has the value "abc"
6. Dim s As String = Chr(97, 98, 99)
7. Print s
8.
9. Sleep
10. End
```

## Έξοδος:



```
the character represented by the ASCII code of
97 is: a
abc
abc
```

## WChr

Επιστρέφει μια συμβολοσειρά χαρακτήρων από μία ή περισσότερες ακέραιες τιμές UNICODE

### Σύνταξη:



```
result = WChr[$]( ch0 [, ch1 ... chN ] )
```

### Παράμετροι:

ch

Η ακέραιη τιμή ενός χαρακτήρα UNICODE.

### Επιστρεφόμενη τιμή:

Επιστρέφει μια συμβολοσειρά που περιέχει τους χαρακτήρες.

### Περιγραφή:

Το WChr επιστρέφει μια συμβολοσειρά που περιέχει τους χαρακτήρες που αντιπροσωπεύονται από τις τιμές UNICODE που μεταβιβάζονται σε αυτήν.

Όταν το Chr χρησιμοποιείται με αριθμητικές σταθερές, το αποτέλεσμα αξιολογείται κατά τον χρόνο μεταγλώττισης, ώστε να μπορεί να χρησιμοποιηθεί σε μεταβλητές για αρχικοποιήσεις.

### Παράδειγμα - wchr.bas



```
1. Print "The UNICODE code of 934 is: "; WChr(934)
2. Print "Multiple UNICODE characters: "; WChr(933,
   934, 935)
3.
4. Sleep
5. End
```

## Έξοδος:



```
The UNICODE code of 934 is: Φ  
Multiple UNICODE characters: ΥΦΧ
```

## Bin

Επιστρέφει μια δυαδική (βάση 2) αναπαράσταση συμβολοσειράς ενός ακέραιου.

## Σύνταξη:



```
result = Bin[$]( number [, digits ] )
```

## Παράμετροι:

### number

Ένας αριθμός ή μια έκφραση που αξιολογείται σε έναν αριθμό. Ένας αριθμός κυμαινόμενης υποδιαστολής θα μετατραπεί σε LongInt.

### digits

Επιθυμητός αριθμός ψηφίων στην συμβολοσειρά που επιστρέφεται.

## Επιστρεφόμενη τιμή:

Μια συμβολοσειρά που περιέχει την μη προσημασμένη δυαδική αναπαράσταση αριθμού.

## Περιγραφή:

Επιστρέφει μια συμβολοσειρά που αντιπροσωπεύει την μη προσημασμένη δυαδική τιμή του ακέραιου αριθμού. Τα δυαδικά ψηφία κυμαίνονται από 0 έως 1.

Εάν καθορίσετε `digits > 0`, η συμβολοσειρά αποτελεσμάτων θα έχει ακριβώς αυτό το μήκος. Θα είναι περικομμένο ή γεμισμένο με μηδενικά στα αριστερά, εάν είναι απαραίτητο. Το μήκος της συμβολοσειράς δεν θα υπερβαίνει τον μέγιστο αριθμό ψηφίων που απαιτούνται για τον τύπο του αριθμού (32 για Long, 64 για LongInt).

Εάν θέλετε να κάνετε το αντίθετο, δηλαδή να μετατρέψετε ξανά μια δυαδική συμβολοσειρά σε έναν αριθμό, ο ευκολότερος τρόπος για να το κάνετε είναι να προσθέσετε τη συμβολοσειρά με "& B" και να τη μετατρέψετε σε έναν ακέραιο τύπο, χρησιμοποιώντας μια συνάρτηση όπως το CInt, παρόμοια με μια κανονική αριθμητική συμβολοσειρά. Π.χ. CInt("& B101")

### Παράδειγμα - bin



```
1. Print Bin(54321)
2. Print Bin(54321, 5)
3. Print Bin(54321, 20)
4.
5. Sleep
6. End
```

### Έξοδος:



```
1101010000110001
10001
00001101010000110001
```

## WBin

Επιστρέφει την δυαδική αναπαράσταση WString (Unicode) ενός αριθμού.

### Σύνταξη:



```
result = WBin( number [, digits] )
```

### Παράμετροι:

#### number

Ένας ακέραιος αριθμός ή μια έκφραση που εκτιμάται σε έναν ακέραιο αριθμό.

#### digits

Προαιρετικός αριθμός ψηφίων για επιστροφή.

### Επιστρεφόμενη τιμή:

Επιστρέφει μια δυαδική αναπαράσταση αριθμού WString, περικομμένη ή γεμισμένη με μηδενικά ("0") για να ταιριάζει στον αριθμό των ψηφίων, εάν καθορίζεται.

### Περιγραφή:

Επιστρέφει ένα WString (Unicode) που αντιπροσωπεύει τη δυαδική τιμή του ακέραιου αριθμού. Τα δυαδικά ψηφία κυμαίνονται από 0 έως 1.

Εάν καθορίσετε `digits > 0`, το αποτέλεσμα `wstring` θα είναι ακριβώς αυτό το μήκος. Θα είναι περικομμένο ή γεμισμένο με μηδενικά στα αριστερά, εάν είναι απαραίτητο.

Το μήκος της συμβολοσειράς που επιστρέφεται δεν θα είναι μεγαλύτερο από τον μέγιστο αριθμό ψηφίων που απαιτούνται για τον τύπο της έκφρασης (32 για Long, 64 για floating point ή LongInt)

## Παράδειγμα - wbin.bas



```
1. Print WBin(54321)
2. Print WBin(54321, 5)
3. Print WBin(54321, 20)
4.
5. Sleep
6. End
```

## Έξοδος:



```
1101010000110001
10001
00001101010000110001
```

## Hex

Επιστρέφει την δεκαεξαδική μορφή ενός αριθμού.

## Σύνταξη:



```
result = Hex[$]( number [, digits ] )
```

## Παράμετροι:

### number

Ένας αριθμός ή μια έκφραση που αξιολογείται σε έναν αριθμό. Ένας αριθμός κυμαινόμενης υποδιαστολής θα μετατραπεί σε LongInt.

### digits

Προαιρετικός αριθμός ψηφίων για επιστροφή.

## Επιστρεφόμενη τιμή:

Μια συμβολοσειρά που περιέχει την μη προσημασμένη δεκαεξαδική αναπαράσταση του αριθμού.

## Περιγραφή:

Επιστρέφει την μη προσημασμένη δεκαεξαδική συμβολοσειρά του ακέραιου αριθμού. Τα δεκαεξαδικά ψηφία κυμαίνονται από 0-9 ή A-F.

Εάν καθορίσετε `digits > 0`, η συμβολοσειρά αποτελεσμάτων θα είναι ακριβώς αυτό το μήκος. Θα είναι περικομμένο ή γεμισμένο με μηδενικά στα αριστερά, εάν είναι απαραίτητο. Το μήκος της συμβολοσειράς δεν θα υπερβαίνει τον μέγιστο αριθμό ψηφίων που απαιτούνται για τον τύπο του αριθμού (8 για Long, 16 για LongInt).

Εάν θέλετε να κάνετε το αντίθετο, δηλαδή να μετατρέψετε μια δεκαεξαδική συμβολοσειρά ξανά σε έναν αριθμό, ο ευκολότερος τρόπος για να το κάνετε είναι να προσθέσετε στη συμβολοσειρά το "&H" και να τη μετατρέψετε σε έναν ακέραιο τύπο, χρησιμοποιώντας μια συνάρτηση όπως CInt, παρόμοια με μια κανονική αριθμητική συμβολοσειρά. Π.χ. CInt("& HFF")

## Παράδειγμα - hex.bas



```
1. '54321 is D431 in hex
2. Print Hex(54321)
3. Print Hex(54321, 2)
4. Print Hex(54321, 5)
5.
6. Sleep
7. End
```

## Έξοδος:



```
D431
31
0D431
```



## WHex

Επιστρέφει την δεκαεξαδική αναπαράσταση WString (Unicode) ενός αριθμού.

### Σύνταξη:



```
result = WHex( number [, digits ] )
```

### Παράμετροι:

#### number

Ένας ακέραιος αριθμός ή μια έκφραση που εκτιμάται σε έναν ακέραιο αριθμό.

#### digits

Προαιρετικός αριθμός ψηφίων για επιστροφή.

### Επιστρεφόμενη τιμή:

Επιστρέφει μια δεκαεξαδική αναπαράσταση αριθμού WString, περικομμένη ή γεμισμένη με μηδενικά ("0") για να ταιριάζει στον αριθμό των ψηφίων, εάν καθορίζεται.

### Περιγραφή:

Τα δεκαεξαδικά ψηφία κυμαίνονται από 0-9 ή A-F.

Εάν καθορίσετε `digits > 0`, το προκύπτων WString θα έχει ακριβώς αυτό το μήκος. Θα είναι περικομμένο ή γεμισμένο με μηδενικά στα αριστερά, εάν είναι απαραίτητο.

Το μήκος του wstring δεν θα υπερβαίνει τον μέγιστο αριθμό ψηφίων που απαιτούνται για τον τύπο της έκφρασης (8 για Long, 16 για floating point ή LongInt).

### Παράδειγμα - whex.bas



```
1. '54321 is D431 in hex
2. Print WHex(54321)
3. Print WHex(54321, 2)
4. Print WHex(54321, 5)
5. Sleep
6. End
```

## .Εξοδος:



```
D431
31
0D431
```

## Oct

Επιστρέφει την οκταδική αναπαράστασή ενός αριθμού.

### Σύνταξη:



```
result = Oct[$]( number [, digits ] )
```

### Παράμετροι:

#### number

Ένας αριθμός ή μια έκφραση που αξιολογείται σε έναν αριθμό. Ένας αριθμός κυμαινόμενης υποδιαστολής θα μετατραπεί σε LongInt.

#### digits

Προαιρετικός αριθμός ψηφίων για επιστροφή.

### Επιστρεφόμενη τιμή:

Επιστρέφει μια συμβολοσειρά που αναπαριστά την μη προσημασμένη οκταδική μορφή ενός αριθμού.

### Περιγραφή:

Επιστρέφει την μη προσημασμένη οκταδική συμβολοσειρά αριθμού. Τα οκτώ ψηφία κυμαίνονται από 0 έως 7.

Εάν καθορίσετε `digits > 0`, η συμβολοσειρά αποτελεσμάτων θα είναι ακριβώς αυτό το μήκος. Θα είναι περικομμένο ή γεμισμένο με μηδενικά στα αριστερά, εάν είναι απαραίτητο.

Το μήκος της συμβολοσειράς που επιστρέφεται δεν θα είναι μεγαλύτερο από τον μέγιστο αριθμό ψηφίων που απαιτούνται για τον τύπο του αριθμού (3 χαρακτήρες για Byte, 6 για Short, 11 για Long και 22 για LongInt)  
Εάν θέλετε να κάνετε το αντίθετο, δηλαδή να μετατρέψετε μια οκταδική συμβολοσειρά ξανά σε έναν αριθμό, ο ευκολότερος τρόπος για να το κάνετε είναι να προσθέσετε τη συμβολοσειρά με "&O" και να τη μετατρέψετε σε έναν ακέραιο τύπο, χρησιμοποιώντας μια συνάρτηση όπως το CInt, παρόμοια με μια κανονική αριθμητική συμβολοσειρά. Π.χ. CInt("&O77")

### Παράδειγμα - oct.bas



```
1. Print Oct(54321)
2. Print Oct(54321, 4)
3. Print Oct(54321, 8)
4.
5. Sleep
6. End
```

### Έξοδος:



```
152061
2061
00152061
```

### WOct

Μετατρέπει έναν αριθμό σε οκταδική παράσταση Unicode.

### Σύνταξη:



```
result = WOct( number [, digits ] )
```

## Παράμετροι:

### number

Ένας ακέραιος αριθμός ή μια έκφραση που εκτιμάται σε έναν ακέραιο αριθμό.

### digits

Προαιρετικός αριθμός ψηφίων για επιστροφή.

## Επιστρεφόμενη τιμή:

Επιστρέφει μια οκταδική αναπαράσταση αριθμού WString, περικομμένη ή γεμισμένη με μηδενικά ("0") για να ταιριάζει στον αριθμό των ψηφίων, εάν καθορίζεται.

## Περιγραφή:

Τα οκτώ ψηφία κυμαίνονται από 0 έως 7.

Εάν καθορίσετε `digits > 0`, το προκύπτουν WString θα έχει ακριβώς αυτό το μήκος. Θα είναι περικομμένο ή γεμισμένο με μηδενικά στα αριστερά, εάν είναι απαραίτητο.

Το μήκος της συμβολοσειράς που επιστρέφεται δεν θα είναι μεγαλύτερο από τον μέγιστο αριθμό ψηφίων που απαιτούνται για τον τύπο του αριθμού (3 χαρακτήρες για Byte, 6 για Short, 11 για Long και 22 για LongInt)

## Παράδειγμα - woct.bas



```
1. Print WOct(54321)
2. Print WOct(54321, 4)
3. Print WOct(54321, 8)
4.
5. Sleep
6. End
```

## Έξοδος:



```
152061
2061
00152061
```

## Str

Επιστρέφει μια αναπαράσταση συμβολοσειράς ενός αριθμού, boolean ή Unicode συμβολοσειράς.

### Σύνταξη:



```
result = Str[$]( number )  
ή  
result = Str( string )
```

### Παράμετροι:

number

Αριθμητική έκφραση για μετατροπή σε συμβολοσειρά.

string

Έκφραση συμβολοσειράς για μετατροπή σε συμβολοσειρά.

### Περιγραφή:

Το Str μετατρέπει τις αριθμητικές μεταβλητές στην αναπαράσταση συμβολοσειρών τους. Όταν χρησιμοποιείται με αυτόν τον τρόπο το String είναι ισοδύναμο με το WStr που εφαρμόζεται σε αριθμητικές μεταβλητές και το αντίθετο της συνάρτησης Val, που μετατρέπει μια συμβολοσειρά σε αριθμό.

Το Str μετατρέπει τις boolean μεταβλητές στην αναπαράσταση συμβολοσειρών τους "false" / "true".

Το Str μετατρέπει επίσης τις συμβολοσειρές χαρακτήρων Unicode σε συμβολοσειρές χαρακτήρων ASCII. Όταν χρησιμοποιείται με αυτόν τον τρόπο κάνει το αντίθετο από το WStr. Εάν δοθεί μια συμβολοσειρά χαρακτήρων ASCII, αυτή η συμβολοσειρά επιστρέφεται χωρίς τροποποίηση.

## Παράδειγμα - str.bas



```
1. Dim a As Integer
2. Dim b As String
3.
4. a = 8421
5. b = Str(a)
6.
7. Print a, b
8.
9.
10. Sleep
11. End
```

## Έξοδος:



```
8421      8421
```

## WStr

Επιστρέφει μια αναπαράσταση συμβολοσειράς ευρέως χαρακτήρα ενός αριθμού ή συμβολοσειράς χαρακτήρων ASCII.

## Σύνταξη:



```
result = WStr( number )
ή
result = WStr( string )
```

## Παράμετροι:

### number

Αριθμητική έκφραση για μετατροπή σε συμβολοσειρά ευρέως χαρακτήρα.

### string

Έκφραση συμβολοσειράς για μετατροπή σε συμβολοσειρά ευρέως χαρακτήρα.

## Επιστρεφόμενη τιμή:

Επιστρέφει την αναπαράσταση ευρέος χαρακτήρα της αριθμητικής αναπαράστασης ή της αναπαράστασης συμβολοσειράς.

## Περιγραφή:

Το WStr μετατρέπει αριθμητικές μεταβλητές στην αναπαράσταση συμβολοσειρών ευρέος χαρακτήρα.

Το WStr μετατρέπει επίσης τις συμβολοσειρές χαρακτήρων ASCII σε συμβολοσειρές χαρακτήρων Unicode. Εάν δοθεί μια συμβολοσειρά ευρέως χαρακτήρα, αυτή η συμβολοσειρά επιστρέφεται χωρίς τροποποίηση.

## Παράδειγμα:



```
1. Dim zs As ZString * 20
2. Dim ws As WString * 20
3. zs = "Hello World"
4. ws = WStr(zs)
5.
6. Print ws
7. Print WStr("Unicode 'Hello World'")
8.
9. Sleep
10. End
```

## Έξοδος:



```
Hello World
Unicode 'Hello World'
```

## Format

Διαμορφώνει έναν αριθμό σε μια καθορισμένη μορφή.

### Σύνταξη:



```
#include "string.bi"  
result = Format[$]( numerical_expression,  
formatting_expression )
```

### Παράμετροι:

numerical\_expression

Ο αριθμός προς μορφοποίηση.

formatting\_expression

Μοτίβο μορφοποίησης

### Επιστρεφόμενη τιμή:

Το Format επιστρέφει μια συμβολοσειρά με το αποτέλεσμα της αριθμητικής έκφρασης που έχει μορφοποιηθεί όπως υποδεικνύεται στην έκφραση μορφοποίησης.

Η έκφραση μορφοποίησης είναι μια συμβολοσειρά που μπορεί να δώσει αριθμητικές τιμές ή τιμές ημερομηνίας-ώρας.

### Περιγραφή:

Για να ανακτήσετε σημαντικές τιμές ημερομηνίας-ώρας, η αριθμητική έκφραση πρέπει να είναι μια σειριακή ημερομηνία που λαμβάνεται από τις κατάλληλες συναρτήσεις.

Αυτή η συνάρτηση είναι μέρος της FreeBASIC, ωστόσο δεν αναγνωρίζεται από τον μεταγλωττιστή εκτός εάν περιλαμβάνεται το vbcompat.bi.



## Αριθμητικά Μοτίβα

Σύμβολο	Περιγραφή
Άδεια (Null) συμβολοσειρά	Γενική μορφή (χωρίς μορφοποίηση)
0	Ψηφιακό σύμβολο κράτησης θέσης: Εάν ο αριθμός έχει λιγότερα ψηφία από ό, τι υπάρχουν μηδενικά (και στις δύο πλευρές του δεκαδικού) στην έκφραση μορφής, εμφανίζονται μηδενικά που οδηγούν ή ακολουθούν. Εάν υπάρχουν περισσότερα ψηφία στα δεξιά του δεκαδικού από τα μηδενικά στη μορφή, ο αριθμός στρογγυλοποιείται. Εάν υπάρχουν περισσότερα ψηφία στα αριστερά του δεκαδικού από τα μηδενικά στη μορφή, τα ψηφία εμφανίζονται όλα
#	Θέση ψηφίου: ακολουθεί τους ίδιους κανόνες όπως και για το 0 ψηφίο, εκτός από του ότι τα αρχικά ή τελικά μηδενικά δεν εμφανίζονται
.	Σύμβολο κράτησης θέσης για δεκαδικό σημείο. Εάν η μορφή περιέχει μόνο # στα αριστερά του τότε αριθμοί μικρότεροι από 1 ξεκινούν με δεκαδικό σημείο.
%	Ποσοστό: Η έκφραση πολλαπλασιάζεται με 100 και εισάγεται ο χαρακτήρας %.
,	Χιλιάδες διαχωριστικό. Δύο παρακείμενα κόμματα, ή ένα κόμμα αμέσως στα αριστερά της θέσης της υποδιαστολής (αν υπάρχει δεκαδικό καθορισμένο ή όχι) σημαίνει «Παραλείψτε τα τρία ψηφία που εμπίπτουν μεταξύ αυτών των κόμματος ή μεταξύ του κόμματος και της υποδιαστολής, στρογγυλοποιώντας όπως απαιτείται .. '
E- E+ e- e+	Επιστημονική μορφή: Εάν μια μορφή περιέχει μονοψήφιο σύμβολο κράτησης

	<p>θέσης (0 ή #) στα δεξιά ενός E-, E+, e- ή e+, ο αριθμός εμφανίζεται σε επιστημονική μορφή και ένα E ή e παρεμβάλλεται μεταξύ του αριθμού και του εκθέτη. Ο αριθμός των 0 ή # στα δεξιά καθορίζει τον αριθμό των ψηφίων στον εκθέτη. Χρησιμοποιήστε E- ή e- για να τοποθετήσετε ένα σύμβολο μείον δίπλα στους αρνητικούς εκθέτες. Χρησιμοποιήστε ένα E+ ή e+ για να τοποθετήσετε ένα σύμβολο μείον δίπλα στους αρνητικούς εκθέτες και ένα σύμβολο συν δίπλα στους θετικούς εκθέτες.</p>
: ? + \$ () κενό	<p>Εμφάνιση κυριολεκτικού χαρακτήρα Για να εμφανίσετε έναν χαρακτήρα διαφορετικό από έναν από αυτούς, προηγηθείτε του χαρακτήρα με ανάστροφο (\) ή περικλείστε τους χαρακτήρες σε διπλά εισαγωγικά</p>
\	<p>Εμφάνιση του επόμενου χαρακτήρα στη συμβολοσειρά μορφής ως έχει</p>
Κείμενο εντός διπλών εισαγωγικών	<p>Εμφανίζει το κείμενο μέσα στα διπλά εισαγωγικά.</p>
:	<p>Ο διαχωριστής χρόνου χρησιμοποιείται για τον διαχωρισμό ωρών, λεπτών και δευτερολέπτων όταν διαμορφώνονται οι τιμές ώρας.</p>
/	<p>Ο διαχωριστής ημερομηνίας χρησιμοποιείται για τον διαχωρισμό ημέρας, μήνα και έτους κατά τη διαμόρφωση των τιμών ημερομηνίας.</p>

## Μοτίβα ημερομηνίας και ώρας

Σύμβολο	Περιγραφή
d, dd	Εμφανίζει την ημέρα με ένα μονοψήφιο / διψήφιο αριθμό (1-31 / 01-31)
ddd	Εμφάνιση της ημέρας ως συντομογραφία (Κυρ-Σαβ)
dddd	Εμφάνιση της ημέρας ως πλήρους ονόματος (Κυριακή-Σάββατο)
dddddd	Εμφάνιση σειριακού αριθμού ημερομηνίας ως πλήρους ημερομηνίας (συμπεριλαμβανομένης της ημέρας, του μήνα και του έτους)
m, mm	Εμφανίζει το μήνα ως ένα μονοψήφιο / διψήφιο αριθμό (1-12 / 01-12). Εάν ακολουθεί αμέσως η ώρα (h, hh), εμφανίζεται το λεπτό και όχι ο μήνας
M, MM	Εμφάνιση του μήνα ως μονοψήφιος/διψήφιος αριθμός (1-12/01-12), ακόμη και αν αμέσως μετά ακολουθεί η ώρα h ή hh
mmm	Εμφανίζει το μήνα ως συντομογραφία (Ιαν-Δεκ)
mmmm	Εμφάνιση του μήνα ως πλήρους ονόματος (Ιανουάριος-Δεκέμβριος)
y, yy	Εμφάνιση του έτους ως διψήφιο αριθμό (00-99)
yyyy	Εμφάνιση του έτους ως τετραψήφιο αριθμό (1900-2040)
h, hh	Εμφάνιση της ώρας ως μονοψήφιου/διψήφιου αριθμού (0-23/00-23)
m, mm	Εμφανίζει τα λεπτά ως μονοψήφιο / διψήφιο αριθμό (0-59 / 00-59). Εάν δεν ακολουθεί αμέσως η ώρα (h) ή (hh),

	εμφανίζεται ο μήνας και όχι το λεπτό
n, nn	Εμφάνιση του λεπτού ως μονοψήφιου/διψήφιου αριθμού (0-59/00-59), ακόμη και αν δεν ακολουθεί αμέσως h ή hh
s, ss	Εμφανίζει το δευτερόλεπτο ως ένα μονοψήφιο / διψήφιο αριθμό (0-59 / 00-59)
ttttt	Εμφανίζει έναν σειριακό αριθμό ως ένα πλήρες χρόνο, συμπεριλαμβανομένων ώρα, λεπτό και το δεύτερο
AM/PM (προκαθορισμένη τιμή), am/pm	Χρησιμοποιήστε το 12ωρο ρολόι που εμφανίζει ΠΜ ή πμ με οποιαδήποτε ώρα πριν το μεσημέρι, ΜΜ ή μ.μ. με οποιαδήποτε ώρα μεταξύ μεσημεριού και 11:59
A/P, a/p	Χρησιμοποιήστε το ρολόι 12 ωρών με ένδειξη Α ή α με οποιαδήποτε ώρα πριν το μεσημέρι, Ρ ή ρ με οποιαδήποτε ώρα μεταξύ μεσημεριού και 11:59

## Παράδειγμα:



```

1. #include "string.bi"
2. Print Format(5, "0")
3. Print Format(5, "0.00")
4. Print Format(5, "#,##0")
5. Print Format(5, "#,##0.00")
6. Print Format(5, "%")
7. Print Format(5, "0.00E+00")
8.
9. Sleep
10. End

```

## Έξοδος:



```
5
5.00
5
5.00
500%
5.00E+00
```

## Val

Μετατρέπει μια συμβολοσειρά σε αριθμό κινητής υποδιαστολής.

### Σύνταξη:



```
result = Val( strnum )
```

### Παράμετροι:

strnum

η συμβολοσειρά που περιέχει έναν αριθμό προς μετατροπή

### Επιστρεφόμενη τιμή:

Επιστρέφει έναν μετατρεπόμενο αριθμό διπλής ακρίβειας. Εάν ο πρώτος χαρακτήρας της συμβολοσειράς δεν είναι έγκυρος, το Val θα επιστρέψει 0.

### Περιγραφή:

Η Val ("10") θα επιστρέψει 10.0 και η Val ("10.10") θα επιστρέψει 10.1. Η συνάρτηση αναλύει τη συμβολοσειρά από τα αριστερά, παραλείποντας οποιοδήποτε λευκό διάστημα και επιστρέφει τον μεγαλύτερο αριθμό που μπορεί να διαβάσει, σταματώντας στον πρώτο μη κατάλληλο χαρακτήρα που βρίσκει. Η επιστημονική σημειογραφία αναγνωρίζεται, με το "D" ή το "E" να χρησιμοποιείται για τον καθορισμό του εκθέτη.

Η Val μπορεί να χρησιμοποιηθεί για τη μετατροπή ακέραιων αριθμών σε δυαδική / οκταδική / δεκαεξαδική μορφή, εάν έχουν το αντίστοιχο αναγνωριστικό ("&B" / "&O" / "&H"), για παράδειγμα: Val("&HFF") επιστρέφει 255.

### Σημείωση:

Εάν θέλετε να λάβετε μια ακέραιη τιμή από μια συμβολοσειρά, σκεφτείτε να χρησιμοποιήσετε την ValInt ή την ValLng αντ' αυτού. Είναι πιο γρήγορα, δεδομένου ότι δεν χρησιμοποιούν τους αριθμούς κινητής υποδιαστολής, και μόνο η ValLng παρέχει πλήρη ακρίβεια 64-bit για τους τύπους LongInt.

Εάν θέλετε να μετατρέψετε έναν αριθμό σε μορφή συμβολοσειράς, χρησιμοποιήστε τη συνάρτηση Str.

### Παράδειγμα -val.bas



```
1. Dim a As String, b As Double
2. a = "2.1E+30xa211"
3. b = Val(a)
4. Print a, b
5.
6. Sleep
7. End
```

### Έξοδος:



```
2.1E+30xa211 2.1e+30
```

## ValInt

Μετατρέπει μια συμβολοσειρά σε ακέραιο 32bit.

### Σύνταξη:



```
result = ValInt ( strnum )
```

### Παράμετροι:

strnum

Η συμβολοσειρά που πρόκειται να μετατραπεί.

### Επιστρεφόμενη τιμή:

Επιστρέφει μια τιμή Long της συμβολοσειράς που έχει μετατραπεί.

Εάν ο πρώτος χαρακτήρας της συμβολοσειράς δεν είναι έγκυρος, η ValInt θα επιστρέψει 0.

### Περιγραφή:

Για παράδειγμα, η ValInt ("10") θα επιστρέψει 10 και η ValInt ("10.60") θα επιστρέψει επίσης 10. Η συνάρτηση αναλύει τη συμβολοσειρά από τα αριστερά, παραλείποντας οποιοδήποτε λευκό διάστημα και επιστρέφει τον μεγαλύτερο αριθμό που μπορεί να διαβάσει, σταματώντας στον πρώτο μη κατάλληλο χαρακτήρα που βρίσκει. Τυχόν μη αριθμητικοί χαρακτήρες, συμπεριλαμβανομένων των δεκαδικών και των προσδιοριστών εκθέτη, θεωρούνται μη κατάλληλοι, για παράδειγμα, το ValInt ("23.1E+6") θα επιστρέψει μόλις 23.

Το ValInt μπορεί να χρησιμοποιηθεί για τη μετατροπή ακέραιων αριθμών σε δυαδική / οκταδική / δεκαεξαδική μορφή, εάν έχουν το σχετικό αναγνωριστικό ("&B" / "&O" / "&H"), για παράδειγμα: ValInt ("&HFF") επιστρέφει 255.

Εάν θέλετε να μετατρέψετε έναν αριθμό σε μορφή συμβολοσειράς, χρησιμοποιήστε τη συνάρτηση Str.

## Παράδειγμα - valint.bas



```
1. Dim a As String, b As Integer
2. a = "20xa211"
3. b = ValInt(a)
4. Print a, b
5.
6. Sleep
7. End
```

## Έξοδος:



```
20xa211    20
```

## ValLng

Μετατρέπει μια συμβολοσειρά σε ακέραιο 64bit.

## Σύνταξη:



```
result = ValLng ( strnum )
```

## Παράμετροι:

### strnum

Η συμβολοσειρά που πρόκειται να μετατραπεί.

## Επιστρεφόμενη τιμή:

Επιστρέφει ένα LongInt της συμβολοσειράς που έχει μετατραπεί

Εάν ο πρώτος χαρακτήρας της συμβολοσειράς δεν είναι έγκυρος, το ValLng θα επιστρέψει 0.



## Περιγραφή:

Για παράδειγμα, η ValLng ("10") θα επιστρέψει 10 και η ValLng ("10.60") θα επιστρέψει επίσης 10. Η συνάρτηση αναλύει τη συμβολοσειρά από τα αριστερά, παραλείποντας οποιοδήποτε λευκό διάστημα και επιστρέφει τον μεγαλύτερο αριθμό που μπορεί να διαβάσει, σταματώντας στον πρώτο μη κατάλληλο χαρακτήρα που βρίσκει. Τυχόν μη αριθμητικοί χαρακτήρες, συμπεριλαμβανομένων των δεκαδικών και των προσδιοριστών εκθέτη, θεωρούνται μη κατάλληλοι, για παράδειγμα, η ValLng ("23.1E+6") θα επιστρέψει μόλις 23. Η ValLng μπορεί να χρησιμοποιηθεί για τη μετατροπή ακέραιων αριθμών σε δυαδική / οκταδική / δεκαεξαδική μορφή, εάν έχουν το σχετικό αναγνωριστικό ("&B" / "&O" / "&H"), για παράδειγμα: ValLng("&HFF") επιστρέφει 255. Εάν θέλετε να μετατρέψετε έναν αριθμό σε μορφή συμβολοσειράς, χρησιμοποιήστε τη συνάρτηση Str.

## Παράδειγμα - vallng.bas



```
1. Dim a As String, b As LongInt
2. a = "20xa211"
3. b = ValLng(a)
4. Print a, b
5.
6. Sleep
7. End
```

## Έξοδος:



```
20xa211    20
```

## ValUInt

Μετατρέπει μια συμβολοσειρά σε ένα μη προσημασμένο 32bit ακέραιο.

### Σύνταξη:



```
result = ValUInt ( strnum )
```

### Παράμετροι:

strnum

Η συμβολοσειρά προς μετατροπή.

### Επιστρεφόμενη τιμή:

Επιστρέφει μια τιμή ULong της συμβολοσειράς που έχει μετατραπεί

Εάν ο πρώτος χαρακτήρας της συμβολοσειράς δεν είναι έγκυρος, η ValUInt θα επιστρέψει 0.

### Περιγραφή:

Για παράδειγμα, η ValUInt ("10") θα επιστρέψει 10 και η ValUInt ("10.60") θα επιστρέψει επίσης 10. Η συνάρτηση αναλύει τη συμβολοσειρά από τα αριστερά, παραλείποντας οποιοδήποτε λευκό διάστημα και επιστρέφει τον μεγαλύτερο αριθμό που μπορεί να διαβάσει, σταματώντας στον πρώτο μη κατάλληλο χαρακτήρα που βρίσκει. Τυχόν μη αριθμητικοί χαρακτήρες, συμπεριλαμβανομένων των δεκαδικών και των προσδιοριστών εκθέτη, θεωρούνται μη κατάλληλοι, για παράδειγμα, η ValUInt ("23.1E+6") θα επιστρέψει μόλις 23.

Η ValUInt μπορεί να χρησιμοποιηθεί για τη μετατροπή ακέραιων αριθμών σε δυαδική / οκταδική / δεκαεξαδική μορφή, εάν έχουν το σχετικό αναγνωριστικό ("&B" / "&O" / "&H"), για παράδειγμα: ValUInt ("&HFF") επιστρέφει 255. Εάν θέλετε να μετατρέψετε έναν αριθμό σε μορφή συμβολοσειράς, χρησιμοποιήστε τη συνάρτηση Str.

## Παράδειγμα - valuint.bas



```
1. Dim a As String, b As UInteger
2. a = "20xa211"
3. b = ValUInt(a)
4. Print a, b
5.
6. Sleep
7. End
```

## Έξοδος:



```
20xa211 20
```

## ValULng

Μετατρέπει μια συμβολοσειρά σε ένα μη προσημασμένο ακέραιο 64 bit.

## Σύνταξη:



result = ValULng ( strnum )

## Παράμετροι:

strnum

Η συμβολοσειρά προς μετατροπή.

## Επιστρεφόμενη τιμή:

Επιστρέφει ένα ULongInt της συμβολοσειράς που έχει μετατραπεί.

Εάν ο πρώτος χαρακτήρας της συμβολοσειράς δεν είναι έγκυρος, το ValULng θα επιστρέψει 0.

## Περιγραφή:

Για παράδειγμα, η ValULng ("10") θα επιστρέψει 10 και η ValULng ("10.60") θα επιστρέψει επίσης 10. Η συνάρτηση αναλύει τη συμβολοσειρά από τα αριστερά, παραλείποντας οποιοδήποτε λευκό διάστημα και επιστρέφει τον μεγαλύτερο αριθμό που μπορεί να διαβάσει, σταματώντας στον πρώτο μη κατάλληλο χαρακτήρα που βρίσκει. Τυχόν μη αριθμητικοί χαρακτήρες, συμπεριλαμβανομένων των δεκαδικών και των προσδιοριστών εκθέτη, θεωρούνται μη κατάλληλοι, για παράδειγμα, το ValULng ("23.1E+6") θα επιστρέψει μόλις 23.

Η ValULng μπορεί να χρησιμοποιηθεί για τη μετατροπή ακέραιων αριθμών σε δυαδική / οκταδική / δεκαεξαδική μορφή, εάν έχουν το αντίστοιχο αναγνωριστικό ("&B" / "&O" / "&H"), για παράδειγμα: ValULng("&HFF") επιστρέφει 255.

Εάν θέλετε να μετατρέψετε έναν αριθμό σε μορφή συμβολοσειράς, χρησιμοποιήστε τη συνάρτηση Str.

## Παράδειγμα - valulng.bas



```
1. Dim a As String, b As ULongInt
2. a = "20xa211"
3. b = ValULng(a)
4. Print a, b
5.
6. Sleep
7. End
```

## Έξοδος:



```
20xa211 20
```

## MKD

Κάνει ένα δυαδικό αντίγραφο από ένα Double σε μια συμβολοσειρά String, ορίζοντας το μήκος της σε 8 byte.

### Σύνταξη:



```
result = MKD[$]( number )
```

### Παράμετροι:

number

Ένας Double αριθμός.

### Επιστρεφόμενη τιμή:

Επιστρέφει ένα String με την δυαδική αντιγραφή του Double.

### Περιγραφή:

Κάνει ένα δυαδικό αντίγραφο από μια μεταβλητή Double σε μια συμβολοσειρά String, ορίζοντας το μήκος της σε 8 byte. Η συμβολοσειρά που προκύπτει μπορεί να διαβαστεί ξανά σε Double με την CVD.

Αυτή η συνάρτηση είναι χρήσιμη για την εγγραφή αριθμητικών τιμών σε buffer χωρίς τη χρήση ορισμού Type.

### Παράδειγμα - mkd.bas



```
1. Dim n As Double, e As String
2. n = 1.2345
3. e = MKD(n)
4. Print n, CVD(e)
5.
6. Sleep
7. End
```

## Έξοδος:



```
1.2345 1.2345
```

## MKI

Μήπως ένα δυαδικό αντίγραφο από μια ακέραια μεταβλητή σε μια συμβολοσειρά του ίδιου μήκους με το μέγεθος της μεταβλητής εισόδου.

## Σύνταξη:



```
result = MKI[$]( number )  
result = MKI[$]<bits>( number )
```

## Παράμετροι:

### number

Μια ακέραια Integer, ή ακέραια Integer<bits> μεταβλητή σε δυαδικό αντίγραφο σε μια συμβολοσειρά.

## Επιστρεφόμενη τιμή:

Επιστρέφει μια συμβολοσειρά που περιέχει ένα δυαδικό αντίγραφο αριθμού.

## Περιγραφή:

Δημιουργεί ένα δυαδικό αντίγραφο από μια μεταβλητή Integer ή Integer<bits> σε μια συμβολοσειρά, ορίζοντας το μήκος της στον αριθμό των byte στον τύπο. Η συμβολοσειρά που προκύπτει μπορεί να διαβαστεί σε έναν ακέραιο τύπο χρησιμοποιώντας CVI ή CVI <bits>.

Αυτή η συνάρτηση είναι χρήσιμη για την εγγραφή αριθμητικών τιμών σε buffer χωρίς τη χρήση ορισμού τύπου.

Το MKI υποστηρίζει μια προαιρετική παράμετρο <bits> πριν από το όρισμα. Εάν τα bits είναι 16, θα καλείται MKShort. αν τα bit είναι 32, θα κληθεί MKL. αν τα bit είναι 64, θα κληθεί η MKLongInt. Το μήκος της τιμής επιστροφής και ο απαιτούμενος τύπος ορίσματος αριθμού εξαρτώνται από τη συνάρτηση που καλείται.

## Παράδειγμα - mki.bas



```
1. Dim a As Integer, b As String
2. a=4534
3. b=MKI(a)
4. Print a, CVI(b)
5.
6. Sleep
7. End
```

## Έξοδος:



```
4534      4534
```

## MKL

Κάνει ένα δυαδικό αντίγραφο από μια Long μεταβλητή σε μια συμβολοσειρά, ορίζοντας το μήκος της σε 4 byte.

## Σύνταξη:



```
result = MKL( number )
```

## Παράμετροι:

### number

Μια Long μεταβλητή σε δυαδικό αντίγραφο σε μια συμβολοσειρά.

### Επιστρεφόμενη τιμή:

Επιστρέφει μια συμβολοσειρά με ένα δυαδικό αντίγραφο του Long.

### Περιγραφή:

Κάνει ένα δυαδικό αντίγραφο από μια Long μεταβλητή σε μια συμβολοσειρά, ορίζοντας το μήκος της σε 4 byte.

Η προκύπτουσα συμβολοσειρά μπορεί να διαβαστεί ξανά σε Long με CVL.

Αυτή η συνάρτηση είναι χρήσιμη για την εγγραφή αριθμητικών τιμών σε buffer χωρίς τη χρήση του Type.

### Παράδειγμα - mkl.bas



```
1. Dim a As Long, b As String
2. a = 4534
3. b = MKL(a)
4. Print a, CVL(b)
5.
6. Sleep
7. End
```

### Έξοδος:



```
4534      4534
```

### MKLongInt

Κάνει ένα δυαδικό αντίγραφο από μια μεταβλητή LongInt σε μια συμβολοσειρά, ορίζοντας το μήκος της σε 8 byte.

### Σύνταξη:



```
result = MKLongInt[$]( number )
```



## Παράμετροι:

number

Μια μεταβλητή LongInt σε δυαδικό αντίγραφο σε μια συμβολοσειρά.

## Επιστρεφόμενη τιμή:

Επιστρέφει μια συμβολοσειρά με ένα δυαδικό αντίγραφο του LongInt.

## Περιγραφή:

Κάνει ένα δυαδικό αντίγραφο από μια μεταβλητή LongInt σε μια συμβολοσειρά, ορίζοντας το μήκος της σε 8 bytes. Η συμβολοσειρά που προκύπτει μπορεί να διαβαστεί ξανά σε ένα longint από το CVLongInt

Αυτή η συνάρτηση είναι χρήσιμη για την εγγραφή αριθμητικών τιμών σε buffer χωρίς τη χρήση του Type.

## Παράδειγμα:



```
1. Dim a As LongInt, b As String
2. a = 4534
3. b = MKLongInt(a)
4. Print a, CVLongInt(b)
5.
6. Sleep
7. End
```

## Έξοδος:



```
4534      4534
```

## MKS

Κάνει ένα δυαδικό αντίγραφο από μια Single μεταβλητή σε μια συμβολοσειρά, ορίζοντας το μήκος της σε 4 bytes.

### Σύνταξη:



```
result = MKS[$]( number )
```

### Παράμετροι:

number

Μια Single μεταβλητή σε δυαδικό αντίγραφο σε μια συμβολοσειρά.

### Επιστρεφόμενη τιμή:

Επιστρέφει μια συμβολοσειρά με ένα δυαδικό αντίγραφο του Single.

### Περιγραφή:

Δημιουργεί ένα δυαδικό αντίγραφο από μια Single μεταβλητή σε μια συμβολοσειρά, ορίζοντας το μήκος της σε 4 byte.

Η συμβολοσειρά που προκύπτει μπορεί να διαβαστεί ξανά σε Single με την CVS.

Αυτή η συνάρτηση είναι χρήσιμη για την εγγραφή αριθμητικών τιμών σε buffer χωρίς τη χρήση του Type.

### Παράδειγμα - mks.bas



```
1. Dim n As Single, e As String
2. n = 1.2345
3. e = MKS(n)
4. Print n, CVS(e)
5.
6. Sleep
7. End
```

## Έξοδος:



```
1.2345 1.2345
```

## MKShort

Κάνει ένα δυαδικό αντίγραφο από μια Short μεταβλητή σε μια συμβολοσειρά, ορίζοντας το μήκος της σε 2 bytes.

## Σύνταξη:



```
result = MKShort[$](number)
```

## Παράμετροι:

### number

Μια Short μεταβλητή σε δυαδικό αντίγραφο σε μια συμβολοσειρά.

## Επιστρεφόμενη τιμή:

Επιστρέφει μια συμβολοσειρά με ένα δυαδικό αντίγραφο του Short.

## Περιγραφή:

Δημιουργεί ένα δυαδικό αντίγραφο από μια μεταβλητή SHORT σε μια συμβολοσειρά, ορίζοντας το μήκος της σε 2 bytes. Η συμβολοσειρά που προκύπτει μπορεί να διαβαστεί ξανά σε ένα Short με την CVShort

Αυτή η συνάρτηση είναι χρήσιμη για την εγγραφή αριθμητικών τιμών σε buffer χωρίς τη χρήση του Type.

## Παράδειγμα - mkshort.bas



```
1. Dim a As Short, b As String
2. a = 4534
3. b = MKShort(a)
4. Print a, CVShort(b)
5.
6. Sleep
7. End
```

## Έξοδος:



```
4534      4534
```

## CVD

Μετατρέπει έναν ακέραιο αριθμό 64 bit ή μια συμβολοσειρά 8 byte σε μια τιμή διπλής ακρίβειας.

## Σύνταξη:



```
result = CVD( l )
result = CVD( str )
```

## Παράμετροι:

l

Ένας 64-bit LongInt με ένα δυαδικό αντίγραφο μιας μεταβλητής διπλής ακρίβειας που είναι αποθηκευμένο σε αυτόν.

str

Μια συμβολοσειρά τουλάχιστον 8 bytes σε μήκος με ένα δυαδικό αντίγραφο μιας μεταβλητής διπλής ακρίβειας που είναι αποθηκευμένη σε αυτήν.

### Επιστρεφόμενη τιμή:

Επιστρέφει έναν Double που περιέχει ένα δυαδικό αντίγραφο της τιμής εισόδου.

### Περιγραφή:

Δημιουργεί δυαδικό αντίγραφο από μια συμβολοσειρά LongInt 64-bit ή 8-byte σε μια μεταβλητή Double.

Επιστρέφεται μια τιμή μηδέν (0,0) εάν η συμβολοσειρά είναι μικρότερη από 8 bytes σε μήκος. Το αποτέλεσμα θα έχει νόημα μόνο εάν η παράμετρος περιείχε μια τιμή διπλής ακρίβειας μορφοποιημένη σε IEEE-754, όπως αυτή που παράγεται από CVLongInt ή MKD.

Αυτή η συνάρτηση είναι χρήσιμη για την ανάγνωση αριθμητικών τιμών από τα buffer χωρίς χρήση του Type.

### Παράδειγμα - cvd.bas



```
1. Dim d As Double, l As LongInt
2. d = 1.125
3. l = CVLongInt(d)
4. Print Using "l = _&H&"; Hex(l)
5. Print Using "cvd(i) = &"; CVD(l)
6.
7. Sleep
8. End
```

### Έξοδος:



```
l = &H3FF2000000000000
cvd(i) = 1.125
```

## CVI

Μετατρέπει έναν αριθμό ή συμβολοσειρά κυμαινόμενης υποδιαστολής σε ακέραιη μεταβλητή χρησιμοποιώντας ένα δυαδικό αντίγραφο.

### Σύνταξη:



```
result = CVI( f )  
result = CVI( str )  
result = CVI<bits>( expr )
```

### Παράμετροι:

f

Ένας αριθμός κυμαινόμενης υποδιαστολής με δυαδικό αντίγραφο μιας ακέραιας μεταβλητής αποθηκευμένο σε αυτόν. Η ακρίβειά του (Single ή Double) εξαρτάται από το μέγεθος του Integer στην τρέχουσα πλατφόρμα

str

Μια συμβολοσειρά με ένα δυαδικό αντίγραφο μιας ακέραιας μεταβλητής που είναι αποθηκευμένη σε αυτήν. κομμάτια

bits

Καθορίζει ένα μέγεθος ακέραιου τύπου για επιστροφή. Οι τύποι και τα μεγέθη του expr που γίνονται δεκτά θα εξαρτηθούν από την αντίστοιχη συνάρτηση που καλείται.

expr

Μια έκφραση που θα αντιγραφεί σε Integer<bits>.

### Επιστρεφόμενη τιμή:

Μια μεταβλητή Integer ή Integer<bits> που περιέχει ένα δυαδικό αντίγραφο της έκφρασης εισόδου.

## Περιγραφή:

Επιστρέφει μια ακέραιη τιμή χρησιμοποιώντας τα δυαδικά δεδομένα που περιέχονται σε μια τιμή κυμαινόμενης υποδιαστολής ή μια συμβολοσειρά.

Επιστρέφεται μια τιμή μηδέν (0) εάν η συμβολοσειρά περιέχει λιγότερους χαρακτήρες από το μέγεθος του τύπου επιστροφής.

Το CVI μπορεί να χρησιμοποιηθεί για τη μετατροπή συμβολοσειρών που έχουν δημιουργηθεί με MKI.

Αυτή η συνάρτηση μπορεί επίσης να χρησιμοποιηθεί για τη μετατροπή τιμών μεγέθους Integer από μνήμη ή αρχείο προσωρινής μνήμης χωρίς να απαιτείται δομή τύπου.

Ωστόσο, όπως και με τη δομή τύπου, πρέπει να δοθεί ιδιαίτερη προσοχή κατά τη χρήση του CVI για τη μετατροπή συμβολοσειρών που έχουν διαβαστεί από ένα buffer.

Το CVI υποστηρίζει μια προαιρετική παράμετρο <bits> πριν από το όρισμα. Εάν τα bits είναι 16, θα κληθεί η CVShort. αν τα bit είναι 32, θα κληθεί η CVL. αν τα bit είναι 64, θα κληθεί η CVLongInt.

Ο τύπος επιστροφής και οι αποδεκτοί τύποι ορισμάτων εξαρτώνται από τη συνάρτηση που καλείται.

Η συμπεριφορά του CVI αλλάζει ανάλογα με το μέγεθος του τύπου δεδομένων Integer στην τρέχουσα πλατφόρμα.

- Για τον ακέραιο 16-bit (-lang qb), επιστρέφεται μια τιμή 16-bit και δεν γίνονται αποδεκτοί τύποι κυμαινόμενης υποδιαστολής.
- Για ακέραιο 32-bit, επιστρέφεται μια τιμή 32-bit και τα αριθμητικά ορίσματα ερμηνεύονται ως μεμονωμένες τιμές ακριβείας.
- Για τον ακέραιο 64-bit, επιστρέφεται μια τιμή 64-bit και τα αριθμητικά ορίσματα ερμηνεύονται ως τιμές διπλής ακρίβειας.

## Παράδειγμα - cvi.bas



```
1. Dim i As Integer, s As String
2. s = "ABCD"
3. i = CVI(s)
4. Print Using "s = "&""; s
5. Print Using "i = _&H&"; Hex(i)
6.
7. Sleep
8. End
```

## Έξοδος:



```
s = "ABCD"
i = &H0
```

## CVL

Μετατρέπει έναν αριθμό κυμαινόμενης υποδιαστολής μίας ακριβείας ή μια συμβολοσειρά τεσσάρων bytes σε μια ακέραιη (Long) μεταβλητή.

## Σύνταξη:



```
result = CVL( sng )
result = CVL( str )
```

## Παράμετροι:

### sng

Ένας αριθμός Single κυμαινόμενης υποδιαστολής με δυαδικό αντίγραφο μιας ακέραιας μεταβλητής αποθηκευμένο σε αυτόν.

### str

Μια συμβολοσειρά μήκους τουλάχιστον τεσσάρων bytes με ένα δυαδικό αντίγραφο μιας ακέραιας μεταβλητής αποθηκευμένης σε αυτήν.



## Επιστρεφόμενη τιμή:

Μια Long μεταβλητή για αντιγραφή του δυαδικού αντιγράφου ενός ακέραιου σε αυτή.

## Περιγραφή:

Επιστρέφει μια ακέραιη τιμή 32-bit με χρήση των δυαδικών δεδομένων που περιέχονται σε ένα Single ή μιας συμβολοσειράς μήκους τουλάχιστον τεσσάρων byte.

Επιστρέφεται μια τιμή μηδέν (0) εάν η συμβολοσειρά είναι μικρότερη από τέσσερα bytes σε μήκος.

Το CVL χρησιμοποιείται για τη μετατροπή συμβολοσειρών 4 bytes που δημιουργήθηκαν με MKL.

Αυτή η συνάρτηση μπορεί επίσης να χρησιμοποιηθεί για τη μετατροπή ακέραιων τιμών 32-bit από μνήμη ή αρχείο προσωρινής μνήμης χωρίς να απαιτείται δομή Type. Ωστόσο, όπως και με τη δομή Type, πρέπει να δοθεί ιδιαίτερη προσοχή κατά τη χρήση του CVL για τη μετατροπή συμβολοσειρών που έχουν διαβαστεί από ένα buffer

## Παράδειγμα - cvl.bas



```
1. Dim l As Long, s As String
2. s = "ABCD"
3. l = CVL(s)
4. Print Using "s = "&""; s
5. Print Using "l = &"; l
6.
7. Sleep
8. End
```

## Έξοδος:



```
s = "ABCD"
l = 1145258561
```

## CVLongInt

Μετατρέπει έναν αριθμό κυμαινόμενης υποδιαστολής διπλής ακριβείας ή συμβολοσειρά οκτώ byte σε μεταβλητή LongInt.

### Σύνταξη:



```
result = CVLongInt( dbl )  
result = CVLongInt( str )
```

### Παράμετροι:

#### dbl

Ένας Double αριθμός κυμαινόμενης υποδιαστολής με ένα δυαδικό αντίγραφο μιας μεταβλητής LongInt αποθηκευμένο σε αυτό.

#### str

Μια συμβολοσειρά τουλάχιστον οκτώ bytes σε μήκος με ένα δυαδικό αντίγραφο μιας μεταβλητής LongInt αποθηκευμένο σε αυτήν.

### Επιστρεφόμενη τιμή:

Μια μεταβλητή LongInt που περιέχει ένα δυαδικό αντίγραφο της μεταβλητής εισόδου.

### Περιγραφή:

Επιστρέφει μια τιμή LongInt 64-bit χρησιμοποιώντας τα δυαδικά δεδομένα που περιέχονται σε ένα διπλό ή μια συμβολοσειρά μήκους τουλάχιστον οκτώ bytes.

Επιστρέφεται μια τιμή μηδέν (0) εάν η συμβολοσειρά είναι μικρότερη από οκτώ bytes σε μήκος.

Η CVLongInt χρησιμοποιείται για τη μετατροπή συμβολοσειρών 8 byte που δημιουργήθηκαν με τη MKLongInt.

Αυτή η συνάρτηση μπορεί επίσης να χρησιμοποιηθεί για τη μετατροπή ακέραιων τιμών 64-bit από ένα buffer μνήμης ή αρχείου χωρίς να χρειάζεται δομή Type. Ωστόσο, όπως και με τη δομή Type, πρέπει να δοθεί ιδιαίτερη προσοχή κατά τη

χρήση του CVLongInt για τη μετατροπή συμβολοσειρών που έχουν διαβαστεί από ένα buffer.

## Παράδειγμα - cvlongint.bas



```
1. Dim ll As LongInt, s As String
2. s = "ABCDEFGH"
3. ll = CVLongInt(s)
4. Print Using "s = "&""; s
5. Print Using "ll = _&"; Hex(ll)
6.
7. Sleep
8. End
```

## Έξοδος:



```
s = "ABCDEFGH"
ll = &H4847464544434241
```

## CVS

Μετατρέπει ακέραιο 32-bit ή συμβολοσειρά 4-bytes σε μεταβλητή μίας ακριβείας.

## Σύνταξη:



```
result = CVS( i )
result = CVS( str )
```

## Παράμετροι:

i  
Ένας ακέραιος 32-bit με ένα δυαδικό αντίγραφο μιας μεταβλητής μίας ακριβείας αποθηκευμένο σε αυτό.

`str`

Μια συμβολοσειρά τουλάχιστον 4 bytes σε μήκος με ένα δυαδικό αντίγραφο μιας μεταβλητής μίας ακριβείας αποθηκευμένη σε αυτήν.

### Επιστρεφόμενη τιμή:

Επιστρέφει μια μεμονωμένη τιμή που περιέχει ένα δυαδικό αντίγραφο της τιμής εισόδου.

### Περιγραφή:

Δημιουργεί δυαδικό αντίγραφο από ακέραιο 32-bit ή συμβολοσειρά 4-bytes σε μεμονωμένη μεταβλητή.

Επιστρέφεται μια τιμή μηδέν (0.0) εάν η συμβολοσειρά είναι μικρότερη από 4 bytes σε μήκος. Το αποτέλεσμα θα έχει νόημα μόνο εάν η παράμετρος περιείχε τιμή μονής ακρίβειας μορφοποιημένη σε IEEE-754, όπως αυτή που παράγεται από CVI ή MKS.

Αυτή η συνάρτηση είναι χρήσιμη για την ανάγνωση αριθμητικών τιμών από τα buffer χωρίς χρήση του Type.

### Παράδειγμα -cvs.bas



```
1. Dim f As Single, i As Integer
2. f = 1.125
3. i = CVI(f)
4. Print Using "i = _&H&"; Hex(i)
5. Print Using "cvs(i) = &"; CVS(i)
6.
7. Sleep
8. End
```

### Έξοδος:



```
i = &H3FF2000000000000
cvs(i) = 0
```

## CVShort

Μετατρέπει μια συμβολοσειρά δύο byte σε μια μεταβλητή Short integer.

### Σύνταξη:



```
result = CVShort( str )
```

### Παράμετροι:

str

Μια συμβολοσειρά τουλάχιστον δύο byte σε μήκος, με ένα δυαδικό αντίγραφο μιας μεταβλητής Short Integer αποθηκευμένου σε αυτήν.

### Επιστρεφόμενη τιμή:

Short μεταβλητή που κρατά το δυαδικό αντίγραφο ενός short.

### Περιγραφή:

Επιστρέφει μια ακέραιη τιμή 16-bit με χρήση των δυαδικών δεδομένων που περιέχονται σε μια συμβολοσειρά μήκους τουλάχιστον δύο bytes. Επιστρέφεται μια τιμή μηδέν (0) εάν η συμβολοσειρά είναι μικρότερη από δύο bytes σε μήκος. Το CVShort χρησιμοποιείται για τη μετατροπή συμβολοσειρών 2 bytes που δημιουργήθηκαν με την MKShort.

Αυτή η συνάρτηση μπορεί επίσης να χρησιμοποιηθεί για τη μετατροπή ακέραιων τιμών 16-bit από ένα buffer μνήμης ή αρχείου χωρίς να απαιτείται δομή Type. Ωστόσο, όπως και με τη δομή Type, πρέπει να δοθεί ιδιαίτερη προσοχή κατά τη χρήση του CVShort για τη μετατροπή συμβολοσειρών που έχουν διαβαστεί από ένα buffer.

## Παράδειγμα - cvshort.bas



```
1. Dim si As Short, s As String
2. s = "AB"
3. si = CVShort(s)
4. Print Using "s = "&""; s
5. Print Using "si = _&H&"; Hex(si)
6.
7. Sleep
8. End
```

## Έξοδος:



```
s = "AB"
si = &H4241
```

## Left

Επιστρέφει το αριστερό μέρος μιας συμβολοσειράς.

## Σύνταξη:



```
result = Left[$]( str, n )
```

## Παράμετροι:

str

Η πηγαία συμβολοσειρά.

n

Ο αριθμός των χαρακτήρων που θα επιστραφούν από την συμβολοσειρά.

## Επιστρεφόμενη τιμή:

Επιστρέφει το αριστερό μέρος μιας συμβολοσειράς.

## Περιγραφή:

Επιστρέφει τους πιο αριστερούς  $n$  χαρακτήρες ξεκινώντας από τα αριστερά (αρχή) του `str`. Εάν το `str` είναι κενό, τότε επιστρέφεται η μηδενική συμβολοσειρά (`""`). Εάν  $n \leq 0$  τότε επιστρέφεται η μηδενική συμβολοσειρά (`""`). Εάν  $n > \text{len}(\text{str})$  τότε επιστρέφεται ολόκληρη η συμβολοσειρά προέλευσης.

## Παράδειγμα - `left.bas`



```
1. Dim text As String = "hello world"  
2. Print Left(text, 5)  
3.  
4. Sleep  
5. End
```

## Έξοδος:



```
hello
```

## Right

Επιστρέφει το δεξιό μέρος μιας συμβολοσειράς.

## Σύνταξη:



```
result = Right[$]( str, n )
```

## Παράμετροι:

str

Η πηγαία συμβολοσειρά.

n

Ο αριθμός των χαρακτήρων που θα επιστραφούν από την συμβολοσειρά.

## Επιστρεφόμενη τιμή:

Επιστρέφει το δεξιό μέρος μιας συμβολοσειράς.

## Περιγραφή:

Επιστρέφει τους πιο δεξούς  $n$  χαρακτήρες ξεκινώντας από τα δεξιά (τέλος) του `str`. Εάν το `str` είναι κενό, τότε επιστρέφεται η μηδενική συμβολοσειρά (`""`). Εάν  $n \leq 0$  τότε επιστρέφεται η μηδενική συμβολοσειρά (`""`). Εάν  $n > \text{len}(\text{str})$  τότε επιστρέφεται ολόκληρη η συμβολοσειρά προέλευσης.

## Παράδειγμα - `right.bas`



```
1. Dim text As String = "hello world"  
2. Print Right(text, 5)  
3.  
4. Sleep  
5. End
```

## Έξοδος:



```
world
```

## Mid (Function)

Επιστρέφει ένα μέρος από μια συμβολοσειρά.

## Σύνταξη:



```
result = Mid[$]( str, start [, n ] )
```



## Παράμετροι:

str

Η πηγαία συμβολοσειρά.

start

Η θέση εκκίνησης στο str του μέρους της συμβολοσειράς.  
Ο πρώτος χαρακτήρας ξεκινά από τη θέση 1.

n

Το μήκος του μέρους της συμβολοσειράς.

## Περιγραφή:

Επιστρέφει μέρος συμβολοσειράς ξεκινώντας από την αρχή στο str. Εάν το str είναι κενό, τότε επιστρέφει η μηδενική συμβολοσειρά (""). Εάν το start  $\leq 0$  ή start  $> \text{len}(\text{str})$ , τότε η μηδενική συμβολοσειρά ("") επιστρέφεται.

Στην πρώτη μορφή του Mid, επιστρέφονται όλοι οι υπόλοιποι χαρακτήρες. Στη δεύτερη μορφή, αν  $n < 0$  ή  $n > \text{len}(\text{str})$  τότε επιστρέφονται όλοι οι υπόλοιποι χαρακτήρες.

## Παράδειγμα - mid1.bas



```
1. Print Mid("abcdefg", 3, 2)
2. Print Mid("abcdefg", 3)
3. Print Mid("abcdefg", 2, 1)
4.
5. Sleep
6. End
```

## Έξοδος:



```
cd
cdefg
b
```

## LCASE

Επιστρέφει μια συμβολοσειρά με μικρά γράμματα.

### Σύνταξη:



```
result = LCASE$( str [ , mode ] )
```

### Παράμετροι:

str

Η συμβολοσειρά προς μορφοποίηση.

mode

Λειτουργία μετατροπής: 0 = τρέχουσα τοπική ρύθμιση, 1 = μόνο ASCII.

### Επιστρεφόμενη τιμή:

Επιστρέφει την συμβολοσειρά με μικρά γράμματα.

### Περιγραφή:

Επιστρέφει ένα αντίγραφο του str με όλα τα γράμματα να μετατρέπονται σε πεζά.

Εάν το str είναι κενό, επιστρέφεται η μηδενική συμβολοσειρά ("").

### Παράδειγμα - lcase.bas



```
1. Print LCASE("Hello World")
2.
3. Sleep
4. End
```

**Έξοδος:**



```
hello world
```

## UCase

Επιστρέφει μια συμβολοσειρά με μεγάλα γράμματα.

**Σύνταξη:**



```
result = UCase[$]( str [ , mode ] )
```

**Παράμετροι:**

str

Η συμβολοσειρά προς μορφοποίηση.

mode

Λειτουργία μετατροπής: 0 = τρέχουσα τοπική ρύθμιση, 1 = μόνο ASCII.

**Επιστρεφόμενη τιμή:**

Επιστρέφει την συμβολοσειρά με μεγάλα γράμματα.

**Περιγραφή:**

Επιστρέφει ένα αντίγραφο του str με όλα τα γράμματα να μετατρέπονται σε κεφαλαία.

Εάν το str είναι κενό, επιστρέφεται η μηδενική συμβολοσειρά ("").

## Παράδειγμα - ucase.bas



```
1. | Print UCase("Hello World")
2. |
3. | Sleep
4. | End
```

## Έξοδος:



```
HELLO WORLD
```

## LTrim

Αφαιρεί τις γύρω συμβολοσειρές ή χαρακτήρες στην αριστερή πλευρά μιας συμβολοσειράς.

## Σύνταξη:



```
result = LTrim[$]( str [, [ Any ] trimset ] )
```

## Παράμετροι:

str

Η συμβολοσειρά προς μορφοποίηση.

trimset

Η συμβολοσειρά που θα αφαιρεθεί.

## Επιστρεφόμενη τιμή:

Επιστρέφει μια συμβολοσειρά που τις έχει αφαιρεθεί από τα αριστερά μέρος της.

## Περιγραφή:

Αυτή η διαδικασία περικόπτει τους γύρω χαρακτήρες από τα αριστερά (αρχή) μιας συμβολοσειράς πηγής. Οι υποσυμβολοσειρές που ταιριάζουν με το trimset θα περικοπούν εάν καθοριστεί, διαφορετικά τα κενά (κωδικός ASCII 32) κόβονται.

Εάν χρησιμοποιείται η λέξη-κλειδί Any, κάθε χαρακτήρας που ταιριάζει με έναν χαρακτήρα στο trimset θα περικοπεί. Όλες οι συγκρίσεις έχουν διάκριση πεζών-κεφαλαίων.

## Παράδειγμα - Ltrim.bas



```
1. Dim s As String = "Hello World"  
2. Print Ltrim(s, "Hello")  
3.  
4. Sleep  
5. End
```

## Έξοδος:



World

## RTrim

Αφαιρεί τις γύρω συμβολοσειρές ή χαρακτήρες στην δεξιά πλευρά μιας συμβολοσειράς.

## Σύνταξη:



```
result = RTrim[$]( str [, [ Any ] trimset ] )
```

## Παράμετροι:

str

Η συμβολοσειρά προς μορφοποίηση.

trimset

Η συμβολοσειρά που θα αφαιρεθεί.

## Επιστρεφόμενη τιμή:

Επιστρέφει μια συμβολοσειρά που τις έχει αφαιρεθεί από τα δεξιά μέρος της.

## Περιγραφή:

Αυτή η διαδικασία περικόπτει τους γύρω χαρακτήρες από τα δεξιά (τέλος) μιας συμβολοσειράς πηγής. Οι υποσυμβολοσειρές που ταιριάζουν με το trimset θα περικοπούν εάν καθοριστεί, διαφορετικά τα κενά (κωδικός ASCII 32) κόβονται.

Εάν χρησιμοποιείται η λέξη-κλειδί Any, κάθε χαρακτήρας που ταιριάζει με έναν χαρακτήρα στο trimset θα περικοπεί. Όλες οι συγκρίσεις έχουν διάκριση πεζών-κεφαλαίων.

## Παράδειγμα - rtrim.bas



```
1. Dim s As String = "Hello World"  
2. Print Rtrim(s, "World")  
3.  
4. Sleep  
5. End
```

## Έξοδος:



```
Hello
```

## Trim

Αφαιρεί τις γύρω συμβολοσειρές ή χαρακτήρες στην αριστερή και δεξιά πλευρά μιας συμβολοσειράς.

### Σύνταξη:



```
result = Trim[$]( str [, [ Any ] trimset ] )
```

### Παράμετροι:

str

Η συμβολοσειρά προς μορφοποίηση.

trimset

Η συμβολοσειρά που θα αφαιρεθεί.

### Επιστρεφόμενη τιμή:

Επιστρέφει μια συμβολοσειρά που τις έχει αφαιρεθεί από τα δεξιά και αριστερά μέρος της.

### Περιγραφή:

Αυτή η διαδικασία περικόπτει τους γύρω χαρακτήρες από τα αριστερά (αρχή) και δεξιά (τέλος) μιας συμβολοσειράς πηγής. Οι υποσυμβολοσειρές που ταιριάζουν με το trimset θα περικοπούν εάν καθοριστεί, διαφορετικά τα κενά (κωδικός ASCII 32) κόβονται.

Εάν χρησιμοποιείται η λέξη-κλειδί Any, κάθε χαρακτήρας που ταιριάζει με έναν χαρακτήρα στο trimset θα περικοπεί.

Όλες οι συγκρίσεις έχουν διάκριση πεζών-κεφαλαίων.

## Παράδειγμα - trim.bas



```
1. Dim s1 As String = " ... Stuck in the middle ...  
2. "  
3. Print "" + Trim(s1) + ""  
4. Print "" + Trim(s1, Any " .") + ""  
5.  
6. Sleep  
7. End
```

## Έξοδος:



```
'... Stuck in the middle ...'  
'Stuck in the middle'
```

## InStr

Εντοπίζει την πρώτη εμφάνιση μιας συμβολοσειράς ή ενός χαρακτήρα μέσα σε μια συμβολοσειρά.

## Σύνταξη:



```
first = InStr( [ start, ] str, [ Any ] substring )
```

## Παράμετροι:

### start

Η θέση στη συμβολοσειρά στην οποία θα ξεκινήσει η αναζήτηση. Ο πρώτος χαρακτήρας ξεκινά από τη θέση 1.

### str

Η συμβολοσειρά προς αναζήτηση.

### substring

Η συμβολοσειρά που πρόκειται να ευρεθεί.



## Επιστρεφόμενη τιμή:

Η θέση της πρώτης εμφάνισης του substring στο str.

## Περιγραφή:

Εντοπίζει τη θέση της πρώτης εμφάνισης μιας συμβολοσειράς ή ενός χαρακτήρα μέσα σε μια άλλη συμβολοσειρά. Στην πρώτη μορφή του InStr (χωρίς παράμετρο start), η αναζήτηση ξεκινά από τον πρώτο χαρακτήρα.

Το μηδέν (0) επιστρέφεται εάν: είτε η μη συμβολοσειρά δεν βρέθηκε, είτε η str είτε η substring είναι κενές συμβολοσειρές, ή ξεκινούν <1.

Εάν έχει οριστεί η λέξη-κλειδί Any, το InStr επιστρέφει την πρώτη εμφάνιση οποιουδήποτε χαρακτήρα σε υποσύμβολο.

## Παράδειγμα - instr.bas



```
1. ' It will return 4
2. Print InStr("abcdefg", "de")
3. ' It will return 0
4. Print InStr("abcdefg", "h")
5.
6. Sleep
7. End
```

## Έξοδος:



```
4
0
```

## InStrRev

Εντοπίζει την τελευταία εμφάνιση μιας συμβολοσειράς ή ενός χαρακτήρα μέσα σε μια συμβολοσειρά.

### Σύνταξη:



first = InStrRev( [ start, ] str, [ Any ] substring )

### Παράμετροι:

#### start

Η θέση στη συμβολοσειρά στην οποία θα ξεκινήσει η αναζήτηση. Ο πρώτος χαρακτήρας ξεκινά από τη θέση 1.

#### str

Η συμβολοσειρά προς αναζήτηση.

#### substring

Η συμβολοσειρά που πρόκειται να ευρεθεί.

### Επιστρεφόμενη τιμή:

Η θέση της τελευταίας εμφάνισης του substring στο str.

### Περιγραφή:

Εντοπίζει τη θέση της τελευταίας εμφάνισης μιας συμβολοσειράς ή ενός χαρακτήρα μέσα σε μια άλλη συμβολοσειρά. Εάν η παράμετρος έναρξης (start) δεν έχει δοθεί ή είναι μικρότερη από μηδέν, η αναζήτηση ξεκινά με τον τελευταίο χαρακτήρα.

Το μηδέν (0) επιστρέφεται εάν:

- δεν βρέθηκε η substring, ή
- είτε το str είτε το subring είναι μια κενή συμβολοσειρά, ή
- η start είναι μηδενική, ή
- η start είναι μεγαλύτερη από το μήκος του str.

Εάν έχει οριστεί η λέξη-κλειδί Any, το InStrRev επιστρέφει την τελευταία εμφάνιση οποιουδήποτε χαρακτήρα στο substring.

### Παράδειγμα - instrrev.bas



```
1. ' It will return 4
2. Print InStrRev("abcdefg", "de")
3. ' It will return 0
4. Print InStrRev("abcdefg", "h")
5.
6. Sleep
7. End
```

### Έξοδος:



```
4
0
```

### Mid (Statement)

Αντικαθιστά μέρος μιας συμβολοσειράς με μια άλλη συμβολοσειρά.

### Σύνταξη:



Mid( text, start ) = expression

ή

Mid( text, start, length ) = expression

### Παράμετροι:

text

Η συμβολοσειρά που θα μορφοποιηθεί.

start

Η αρχική θέση στο κείμενο της συμβολοσειράς για αντικατάσταση. Ο πρώτος χαρακτήρας ξεκινά από τη θέση 1.

## length

Ο αριθμός των χαρακτήρων για αντικατάσταση.

### **Περιγραφή:**

Αντιγράφει μέγιστο μήκος χαρακτήρων έκφρασης σε κείμενο, ξεκινώντας από την αρχή.

Εάν το μήκος δεν καθορίζεται, αντιγράφεται όλη η έκφραση.

Το μέγεθος του κειμένου συμβολοσειράς είναι αμετάβλητο.

Εάν η έκφραση είναι πολύ μεγάλη, το μεγαλύτερο μέρος της αντιγράφεται μέχρι το τέλος του κειμένου.

### **Παράδειγμα - mid2.bas**



```
1. Dim text As String
2. text = "abc 123"
3. Print text 'displays "abc 123"
4.
5. ' replace part of text with another string
6. Mid(text, 5, 3) = "456"
7.
8. Print text 'displays "abc 456"
9.
10. Sleep
11. End
```

### **Έξοδος:**



```
abc 123
abc 456
```

## LSet

Προσαρμόζει προς τα αριστερά μια συμβολοσειρά.

### Σύνταξη:



LSet dst, src

LSet dst\_udt, src\_udt

### Παράμετροι:

dst

Η συμβολοσειρά που θα δεχθεί τα δεδομένα.

src

Η συμβολοσειρά από την οποία θα παρθούν δεδομένα

dst\_udt

Το UDT που θα δεχθεί δεδομένα.

src\_udt

Το UDT από το οποίο θα παρθούν δεδομένα.

### Περιγραφή:

Το LSet προσαρμόζει προς τα αριστερά το κείμενο στο buffer συμβολοσειράς dst, γεμίζοντας το αριστερό μέρος της συμβολοσειράς με το src και το δεξί μέρος με κενά. Το μέγεθος του buffer συμβολοσειράς δεν τροποποιείται. Εάν το κείμενο είναι πολύ μεγάλο για το μέγεθος του buffer συμβολοσειράς, το LSet περικόπτει χαρακτήρες από τα δεξιά.

Για συμβατότητα με την QBasic, το LSet μπορεί επίσης να αντιγράψει ένα UDT που ορίζεται από το χρήστη σε ένα άλλο. Το αντίγραφο γίνεται byte προς byte, χωρίς καμία φροντίδα για ταπεδία ή ευθυγράμμιση. Εναπόκειται στον προγραμματιστή να φροντίσει για την εγκυρότητα του αποτελέσματος.

## Παράδειγμα - lset.bas



```
1. Dim buffer As String
2. buffer = Space(10)
3. LSet buffer, "91.5"
4. Print "-[" & buffer & "]"-
5.
6. Sleep
7. End
```

## Έξοδος:



```
-[91.5  ]-
```

## RSet

Προσαρμόζει προς τα δεξιά μια συμβολοσειρά σε ένα buffer συμβολοσειράς.

## Σύνταξη:



RSet dst, src

## Παράμετροι:

dst

Ένα buffer String ή WString για αντιγραφή του κειμένου.

src

Η πηγή String ή WString που είναι δεξιά προσαρμοσμένη.

## Περιγραφή:

Το RSet right προσαρμόζει το κείμενο στο buffer συμβολοσειράς dst, συμπληρώνοντας το δεξί μέρος της συμβολοσειράς με το src και το αριστερό μέρος με κενά. Το μέγεθος του buffer συμβολοσειράς δεν τροποποιείται. Εάν το κείμενο είναι πολύ μεγάλο για το μέγεθος του buffer συμβολοσειράς, το RSet περικόπτει χαρακτήρες από τα δεξιά.

## Παράδειγμα - rset.bas



```
1. Dim buffer As String
2. buffer = Space(10)
3. RSet buffer, "91.5"
4. Print "-[" & buffer & "]"-
5.
6. Sleep
7. End
```

## Έξοδος:



```
-[ 91.5]-
```

# Συναρτήσεις υποστήριξης νήματος - Threading Support Functions

Διαδικασίες για εργασία με εφαρμογές πολλαπλών νημάτων.

Αυτές οι διαδικασίες επιτρέπουν τον προγραμματισμό πολλών νημάτων. Τα νήματα και οι μεταβλητές υπό όρους μπορούν να δημιουργηθούν και να καταστραφούν και μπορούν να ληφθούν τα λεγόμενα mutexes για την προστασία δεδομένων ευαίσθητων στο νήμα.

Ένας αμοιβαίος αποκλεισμός (mutual exclusion = mutex) χρησιμοποιείται συνεργατικά μεταξύ των νημάτων για να διασφαλιστεί ότι μόνο ένα από τα συνεργαζόμενα νήματα επιτρέπεται να έχει πρόσβαση στα δεδομένα ή να εκτελεί συγκεκριμένο κωδικό εφαρμογής κάθε φορά. Το mutex εμποδίζει την πρόσβαση στα δεδομένα από ένα νήμα μόνο εάν το νήμα χρησιμοποιεί το mutex πριν από την πρόσβαση στα δεδομένα.

## ThreadCall

Ξεκινά μια διαδικασία καθορισμένη από τον χρήστη με παραμέτρους σε ξεχωριστό νήμα εκτέλεσης.

### Σύνταξη:



```
threadid = ThreadCall subname([paramlist])
```

### Παράμετροι:

subname

Το όνομα της υπορουτίνας.

paramlist

Μια λίστα παραμέτρων για μετάβαση στην υπορουτίνα, όπως συμβαίνει με μια κανονική υπορουτίνα.



### **Επιστρεφόμενη τιμή:**

Η Threadcall επιστρέφει ένα handle Any Ptr στο νήμα που δημιουργήθηκε ή τον μηδενικό δείκτη (0) σε περίπτωση αποτυχίας.

### **Περιγραφή:**

Όπως το ThreadCreate, έτσι και το Threadcall δημιουργεί ένα νήμα που τρέχει ταυτόχρονα με τον κώδικα που το καλεί. Τοποθετώντας το "Threadcall" πριν από σχεδόν κάθε κανονική κλήση στο sub, το sub καλείται μέσα σε ένα νέο νήμα και επιστρέφει έναν δείκτη σε αυτό το νήμα.

Η χρήση Threadcall είναι απλούστερη μέθοδος δημιουργίας νημάτων και επιτρέπει τη μετάδοση δεδομένων στο νήμα χωρίς καθολικές μεταβλητές ή δείκτες που δεν είναι ασφαλείς. Ωστόσο, το ThreadCreate είναι πιο αποτελεσματικό και πρέπει να χρησιμοποιείται για προγράμματα που δημιουργούν μεγάλο αριθμό νημάτων.

Ενώ υποστηρίζονται οι περισσότερες υπορουτίνες, ενδέχεται να μην καλούνται οι ακόλουθοι τύποι υπορουτίνων:

- Υπορουτίνες που χρησιμοποιούν μεταβλητά ορίσματα.
- Υπορουτίνες με ενώσεις που περνούνται ως ορίσματα.
- Υπορουτίνες με τύπους χρηστών που περιέχουν ενώσεις, πίνακες, συμβολοσειρές ή πεδία bit τα οποία περνούνται ως ορίσματα.

Όταν χρησιμοποιείτε την Threadcall, απαιτείται παρένθεση γύρω από τη λίστα παραμέτρων εκτός εάν η υπορουτίνα δεν έχει παραμέτρους.

## Παράδειγμα - threadcall.bas



```
1.  '' Threading using "ThreadCall"
2.  Sub thread( id As String, tlock As Any Ptr, count
    As Integer )
3.      For i As Integer = 1 To count
4.          MutexLock tlock
5.          Print "thread " & id;
6.          Locate , 20
7.          Print i & "/" & count
8.          MutexUnlock tlock
9.      Next
10. End Sub
11.
12. Dim tlock As Any Ptr = MutexCreate()
13. Dim a As Any Ptr = ThreadCall thread("A", tlock,
    6)
14. Dim b As Any Ptr = ThreadCall thread("B", tlock,
    4)
15. ThreadWait a
16. ThreadWait b
17. MutexDestroy tlock
18. Print "All done (and without Dim Shared!)"
19.
20. Sleep
    End
```

### Έξοδος:



```
thread 1/4
thread 2/4
thread 3/4
thread 4/4
thread 1/6
thread 2/6
thread 3/6
thread 4/6
thread 5/6
thread 6/6
All done (and without Dim Shared!)
```

Η χρήση του mutex αναγκάζει να τελειώσει πρώτα το πρώτο νήμα και μετά να ξεκινήσει το επόμενο.

## ThreadCreate

Ξεκινά μια διαδικασία καθορισμένη από το χρήστη σε ξεχωριστό νήμα εκτέλεσης.

### Σύνταξη:



```
result = ThreadCreate ( procptr [, [ param ] [,  
stack_size ] ] )
```

### Παράμετροι:

#### procptr

Ένας δείκτης στη Sub που προορίζεται να λειτουργήσει ως νήμα (βλ. Operator Procptr (Procedure Pointer) για να μεταβεί ένας δείκτης σε ένα sub). ΤΗ υπορουτίνα πρέπει να έχει την ακόλουθη υπογραφή (ίδιες παράμετροι, ίδια σύμβαση κλήσης) για να είναι συμβατό με το procptr:

```
Declare Sub myThread ( ByVal userdata As Any Ptr )
```

#### userdata

Η παράμετρος Any Ptr του Sub που προορίζεται να λειτουργήσει ως νήμα. Η FreeBASIC αναμένει ότι αυτή η παράμετρος θα είναι παρούσα, δεν πρέπει να παραλειφθεί!

#### param

Οποιοδήποτε όρισμα Ptr που θα περάσει στο νήμα Sub στο οποίο επισημαίνεται το procptr μέσω της παραμέτρου δεδομένων χρήστη. Για παράδειγμα, αυτό μπορεί να είναι ένας δείκτης σε μια δομή ή έναν πίνακα που περιέχει διάφορες πληροφορίες για το νήμα που λειτουργεί. Εάν η παράμετρος δεν δίνεται, το 0 (μηδέν) θα περάσει στην παράμετρο των δεδομένων χρήστη του νήματος.

#### stack\_size

Προαιρετικός αριθμός byte για κράτηση για τη στοίβα αυτού του νήματος.

### **Επιστρεφόμενη τιμή:**

Το ThreadCreate επιστρέφει ένα handle Any Ptr στο δημιουργούμενο νήμα ή έναν μηδενικό δείκτη (0) σε περίπτωση αποτυχίας.

### **Περιγραφή:**

Η υπορουτίνα στην οποία αναφέρεται το procptr ξεκινά ως νήμα. Θα περάσει το περιεχόμενο της παραμέτρου param, ή 0 (μηδέν) εάν δεν έχει καθοριστεί, στην παράμετρο των δεδομένων χρήστη userdata.

Το sub που ξεκίνησε ως νήμα θα εκτελεστεί παράλληλα με το κύριο μέρος του προγράμματος. Το λειτουργικό σύστημα το επιτυγχάνει αναθέτοντάς το σε διαφορετικό επεξεργαστή εάν υπάρχει ή εναλλάσσοντας μεταξύ των νημάτων εκτέλεσης σε έναν επεξεργαστή.

Δεν υπάρχει καμία εγγύηση σχετικά με τη σειρά με την οποία εκτελούνται διαφορετικά νήματα και δεν μπορούν να γίνουν υποθέσεις σχετικά με τη σειρά με την οποία αρχίζουν να εκτελούνται πολλαπλά νήματα δημιουργίας.

Στις ταχύτερες περιπτώσεις εκκίνησης, το σώμα νήματος μπορεί να ξεκινήσει να εκτελείται ακόμη και πριν επιστρέψει το ThreadCreate.

Κάθε τρέχον νήμα μπορεί να αναγνωριστεί από το handle, το οποίο είναι μοναδικό σε όλα τα τρέχοντα νήματα. Δείτε το ThreadSelf.

Πριν από το κλείσιμο, τα προγράμματα πρέπει να περιμένουν τον τερματισμό όλων των νημάτων που έχουν ξεκινήσει χρησιμοποιώντας το ThreadWait. Εναλλακτικά, εάν δεν είναι απαραίτητο να περιμένετε με ασφάλεια για να ολοκληρωθεί η εκτέλεση ενός νήματος, μπορείτε να χρησιμοποιήσετε το ThreadDetach. Ωστόσο, εάν ένα πρόγραμμα εξέλθει ενώ ορισμένα νήματα είναι ακόμα ενεργά, αυτά τα νήματα θα ακυρωθούν από το σύστημα. Για κάθε νήμα που δημιουργείται, τα προγράμματα πρέπει να καλούν είτε το ThreadWait είτε το ThreadDetach για να διασφαλίσουν ότι οι πόροι συστήματος που σχετίζονται με τα handles του νήματος απελευθερώνονται. Διαφορετικά, ενδέχεται να υπάρχουν διαρροές πόρων μνήμης ή συστήματος.

Λόγω της φύσης των νημάτων, δεν μπορούν να γίνουν υποθέσεις σχετικά με την εντολή εκτέλεσης. Για την ανταλλαγή δεδομένων μεταξύ πολλαπλών νημάτων, συμπεριλαμβανομένου ενός νήματος και του κύριου μέρους του προγράμματος, πρέπει να χρησιμοποιούνται τα mutexes. Αυτά είναι κλειδώματα αμοιβαίου αποκλεισμού και μπορούν να "ανήκουν" σε ένα μόνο νήμα ενώ κάνετε κρίσιμες εργασίες, προκαλώντας άλλα νήματα να περιμένουν τη σειρά τους. Δείτε `MutexCreate`, `MutexLock`, `MutexUnlock`, `MutexDestroy`.

Το `stack_size` μπορεί να χρησιμοποιηθεί για να αλλάξει το μέγεθος της στοίβας του νήματος από το προεπιλεγμένο σύστημα. Αυτό μπορεί να είναι χρήσιμο όταν το πρόγραμμα απαιτεί μεγάλη στοίβα, για παράδειγμα λόγω πολλών επαναλήψεων διαδικασιών ή όταν κατανέμεται τεράστιες συμβολοσειρές/πίνακες στη στοίβα. Σε ορισμένα συστήματα (Linux), η στοίβα αυξάνεται αυτόματα πέρα από το μέγεθος `stack_size` εάν απαιτείται περισσότερος χώρος. Σε άλλα συστήματα (Win32), αυτό είναι το σταθερό μέγιστο επιτρεπόμενο. Η συμπεριφορά είναι απροσδιόριστη όταν χρησιμοποιείται περισσότερη στοίβα από το δεσμευμένο μέγεθος σε συστήματα όπου οι στοίβες δεν είναι σε θέση να αναπτυχθούν.

- Η παράμετρος `userdata` μπορεί να μη χρησιμοποιηθεί στο σώμα της υπορουτίνας `myThread`, αλλά η δήλωση της ως παραμέτρου `Any Ptr` είναι πάντα υποχρεωτική στην κεφαλίδα. Σε αυτήν την περίπτωση, η αντίστοιχη παράμετρος παραμέτρου μπορεί στη συνέχεια να παραλειφθεί κατά την κλήση της `ThreadCreate`, αλλιώς μπορεί να περάσει ακόμα ένα περιττό όρισμα (το '0' χρησιμοποιείται συνήθως επειδή αυτή η τιμή είναι άμεσα συμβατή με οποιονδήποτε δείκτη).

- Στην περίπτωση που τα δεδομένα πρέπει να διαβιβαστούν στο `myThread`, η παράμετρος `Any Ptr` μπορεί να χρησιμοποιηθεί για την αναφορά τους, απαιτώντας συνήθως μετατροπή τύπου (σιωπηρή ή ρητή) σε `Any Ptr` πριν τη μεταβίβαση στο `ThreadCreate` και αντίστροφη μετατροπή τύπου από `Any Ptr` στο σώμα του `myThread` πριν το χρησιμοποιήσετε.

## Παράδειγμα:



```
1.  '' Threading synchronization using Mutexes
2.  '' If you comment out the lines containing
3.  '' "MutexLock" and "MutexUnlock",
4.  '' the threads will not be in sync and some of
5.  '' the data may be printed
6.  '' out of place.
7.  Const MAX_THREADS = 10
8.  Dim Shared As Any Ptr ttylock
9.  '' Teletype unfurls some text across the screen
10. '' at a given location
11. Sub teletype( ByRef text As String, ByVal x As
12.             Integer, ByVal y As Integer )
13.     ''
14.     '' This MutexLock makes simultaneously
15.     '' running threads wait for each
16.     '' other, so only one at a time can continue
17.     '' and print output.
18.     '' Otherwise, their Locates would interfere,
19.     '' since there is only one
20.     '' cursor.
21.     ''
22.     '' It's impossible to predict the order in
23.     '' which threads will arrive
24.     '' here and which one will be the first to
25.     '' acquire the lock thus
26.     '' causing the rest to wait.
27.     ''
28.     MutexLock ttylock
29.     For i As Integer = 0 To (Len(text) - 1)
30.         Locate x, y + i
31.         Print Chr(text[i])
32.         Sleep 25, 1
33.     Next
34.     '' MutexUnlock releases the lock and lets other
35.     '' threads acquire it.
36.     MutexUnlock ttylock
37. End Sub
38. Sub thread( ByVal userdata As Any Ptr )
39.     Dim As Integer id = CInt(userdata)
40.     teletype "Thread (" & id & ").....", 1 +
41.     id, 1
42. End Sub
43. '' Create a mutex to synchronize the threads
44. ttylock = MutexCreate()
45. '' Create child threads
```

```

46. Dim As Any Ptr handles(0 To MAX_THREADS-1)
47. For i As Integer = 0 To MAX_THREADS-1
48.     handles(i) = ThreadCreate(@thread, CPtr(Any
Ptr, i))
49.     If handles(i) = 0 Then
50.         Print "Error creating thread:"; i
51.         Exit For
52.     End If
53. Next
54. ' This is the main thread. Now wait until all
55. ' child threads have finished.
56. For i As Integer = 0 To MAX_THREADS-1
57.     If handles(i) <> 0 Then
58.         ThreadWait(handles(i))
59.     End If
60. Next
61. '' Clean up when finished
62. MutexDestroy(ttylock)
63.
64. Sub print_dots(ByRef char As String)
65.     For i As Integer = 0 To 29
66.         Print char;
67.         Sleep CInt(Rnd() * 100), 1
68.     Next
69. End Sub
70.
71. Sub mythread(param As Any Ptr)
72.     '' Work (other thread)
73.     print_dots("*")
74. End Sub
75.
76. Randomize(Timer())
77. Print " main thread: ."
78. Print "other thread: *"
79. '' Launch another thread
80. Dim As Any Ptr thread1 = ThreadCreate(@mythread,
81. 0)
82. '' Work (main thread)
83. print_dots(".")
84. '' Wait until other thread has finished, if
85. '' needed
86. ThreadWait(thread1)
87. Print
88. Sleep
89. End

```





Για να αφήσετε ένα handle νήματος χωρίς να περιμένετε να τελειώσει το νήμα, χρησιμοποιήστε το ThreadDetach. Το ThreadWait δεν αναγκάζει το νήμα να τελειώσει. εάν ένα νήμα απαιτεί σήμα για να εξαναγκάσει το τέλος του, πρέπει να χρησιμοποιηθεί ένας μηχανισμός όπως shared μεταβλητές και mutexes.

Προκειμένου να αποφευχθούν διαρροές μνήμης, ο ασφαλής τρόπος τερματισμού ενός νήματος είναι να του δίνουμε πάντα σήμα ότι πρέπει να τερματιστεί και, στη συνέχεια, να καλέσουμε το ThreadWait σε αυτό το νήμα, εκτός εάν το ThreadDetach έχει κληθεί προηγουμένως.

## ThreadDetach

Απελευθερώνει ένα handle νήματος χωρίς να περιμένει να τελειώσει το νήμα.

### Σύνταξη:



```
#include "fbthread.bi"  
ThreadDetach( id )
```

### Παράμετροι:

id

Any Ptr handle ενός νήματος που δημιουργήθηκε από το ThreadCreate ή το ThreadCall.

### Περιγραφή:

Το ThreadDetach απελευθερώνει πόρους που σχετίζονται με το handle ενός νήματος που επιστρέφονται από το ThreadCreate ή το ThreadCall. Το handle του νήματος θα καταστραφεί από το ThreadDetach και δεν μπορεί να χρησιμοποιηθεί πια.

Σε αντίθεση με το ThreadWait, το ThreadDetach δεν περιμένει να τελειώσει το νήμα και η εκτέλεση του νήματος συνεχίζεται ανεξάρτητα. Οποιοσδήποτε διαθέσιμος πόρος θα ελευθερωθεί μόλις εξέλθει το νήμα.

Προκειμένου να αποφευχθούν διαρροές μνήμης, ο ασφαλής τρόπος τερματισμού ενός νήματος είναι να του δίνουμε πάντα σήμα ότι πρέπει να τερματιστεί και, στη συνέχεια, να καλέσουμε το ThreadWait σε αυτό το νήμα, εκτός εάν το ThreadDetach έχει κληθεί προηγουμένως.

Σημείωση: Καθώς το ThreadDetach καταστρέφει το handle του νήματος, το ThreadWait δεν μπορεί πλέον να ελέγξει για το τέλος του νήματος και ακόμη και η χρήση του ThreadWait καθίσταται απρόβλεπτη (μπορεί να καταστρέψει το πρόγραμμα). Η χρήση μεταξύ ThreadWait και ThreadDetach πρέπει να είναι αποκλειστική.

Αλλά τα mutexes και οι μεταβλητές υπό όρους μπορούν επίσης να χρησιμοποιηθούν με αποσπασμένα νήματα.

### Παράδειγμα - threaddetach.bas



```
1. #include "fbthread.bi"
2.
3. Sub mythread( ByVal param As Any Ptr )
4.     Print "hi!"
5. End Sub
6.
7.
8. Var thread = ThreadCreate( @mythread )
9.
10. ThreadDetach( thread )
11. ThreadDetach( ThreadCreate( @mythread ) )
12.
13. Sleep
14. End
```

### Έξοδος:



```
hi!
hi!
```

## ThreadSelf

Επιστρέφει το handle του τρέχοντος νήματος.

### Σύνταξη:



```
#include "fbthread.bi"  
result = ThreadSelf
```

### Επιστρεφόμενη τιμή:

Επιστρέφει έναν Any Ptr handle του τρέχοντος νήματος.

### Περιγραφή:

Το ThreadSelf χρησιμοποιείται για να πάρει το handle του τρέχοντος νήματος.

Αυτή η λειτουργία μπορεί να προσδιορίσει μοναδικά τα υπάρχοντα νήματα:

- Εάν υπάρχουν πολλά νήματα και ένα νήμα έχει ολοκληρωθεί, τότε αυτό το handle μπορεί να επαναχρησιμοποιηθεί.

- Έτσι, για όλα τα μόνα νήματα που εξακολουθούν να λειτουργούν, τα handles είναι μοναδικά.

Όταν δημιουργείται ένα νέο νήμα, ένα handle στο νήμα επιστρέφεται από τη συνάρτηση δημιουργίας.

Όταν το νήμα εκτελεί κώδικα, το ThreadSelf επιτρέπει την επιστροφή του handle του νήματος.

Το ThreadSelf μπορεί να χρησιμοποιηθεί για την κωδικοποίηση κάποιου είδους TLS (Thread Local Storage) από το μοναδικό handle κάθε νήματος.

Επομένως, μπορεί να οριστεί το ίδιο όνομα καθολικής μεταβλητής, αλλά με αποθηκευμένη τιμή συγκεκριμένη για το νήμα που έχει πρόσβαση.

Αυτό επιτρέπει την κωδικοποίηση γενικών διαδικασιών, αλλά με παραμέτρους ανάλογα με το νήμα που τις εκτελεί.

## Παράδειγμα - threadself.bas



```
1. #include "fbthread.bi"
2.
3. Dim As Any Ptr phandle(1 To 10)
4.
5. Sub myThread (ByVal p As Any Ptr)
6.     Print "Thread handle: " & ThreadSelf()
7. End Sub
8.
9.
10. For I As Integer = 1 To 10
11.     phandle(I) = ThreadCreate(@myThread)
12. Next I
13.
14. For I As Integer = 1 To 10
15.     ThreadWait(phandle(I))
16. Next I
17.
18. Sleep
19. End
```

## Έξοδος:



```
Thread handle: 31353696
Thread handle: 31355104
Thread handle: 31354400
Thread handle: 31354752
Thread handle: 31355456
Thread handle: 31355808
Thread handle: 31354048
Thread handle: 31356160
Thread handle: 31356512
Thread handle: 31356864
```

## MutexCreate

Δημιουργεί ένα mutex που χρησιμοποιείται για συγχρονισμό της εκτέλεσης των νημάτων.

### Σύνταξη:



```
result = MutexCreate
```

### Επιστρεφόμενη τιμή:

Επιστρέφει έναν Any Ptr handle σε περίπτωση επιτυχίας και τον μηδενικό δείκτη (0) στην αποτυχία.

### Περιγραφή:

Τα Mutexes, συντομογραφία "Mutually Exclusive", είναι ένας τρόπος συγχρονισμού κοινών δεδομένων εντός νημάτων.

Εάν υπάρχει μια καθολική μεταβλητή που χρησιμοποιείται από πολλά νήματα (ή μια τοπική στατική μεταβλητή που χρησιμοποιείται από ένα μόνο νήμα που ονομάζεται πολλές φορές), θα πρέπει να "κλειδωθεί" κατά τη χρήση του με ένα mutex. Αυτό σταματά όλα τα νήματα χρησιμοποιώντας το MutexLock με αυτό το mutex (συμπεριλαμβανομένου του σιωπηρού κύριου νήματος που εκτελεί το κύριο πρόγραμμα), έως ότου ξεκλειδωθεί με το MutexUnlock.

Το Mutexcreate δημιουργεί ένα mutex, επιστρέφοντας ένα handle στο οποίο πρέπει να αναφέρεται κατά το κλείδωμα, το ξεκλείδωμα ή την καταστροφή του mutex. Τα Mutexes που δημιουργούνται με Mutexcreate θα πρέπει να καταστραφούν όταν δεν είναι πλέον απαραίτητα ή πριν από το τέλος του προγράμματος με το MutexDestroy.

Το mutex είναι ένα κλείδωμα που εγγυάται τρία πράγματα:

1. Ατομικότητα - Το κλείδωμα ενός mutex είναι μια ατομική λειτουργία, που σημαίνει ότι το λειτουργικό σύστημα (ή η βιβλιοθήκη νημάτων) σας διαβεβαιώνει ότι εάν κλειδώσατε ένα mutex, κανένα άλλο νήμα δεν θα καταφέρει να κλειδώσει αυτό το mutex ταυτόχρονα.
2. Μοναδικότητα - Εάν ένα νήμα κατάφερε να κλειδώσει ένα mutex, είναι σίγουρο ότι κανένα άλλο νήμα δεν θα μπορεί να

κλειδώσει το νήμα έως ότου το αρχικό νήμα αφήσει το κλείδωμα.

3. Μη απασχολημένη αναμονή - Εάν ένα νήμα επιχειρήσει να κλειδώσει ένα νήμα που ήταν κλειδωμένο από ένα δεύτερο νήμα, το πρώτο νήμα θα τεθεί σε αναστολή (και δεν θα καταναλώσει πόρους CPU) έως ότου το κλείδωμα απελευθερωθεί από το δεύτερο νήμα. Αυτή τη στιγμή, το πρώτο νήμα θα ξυπνήσει και θα συνεχίσει την εκτέλεση, έχοντας το mutex κλειδωμένο από αυτό.

### **Παράδειγμα:**

Δείτε το παράδειγμα της ThreadCreate.

## **MutexLock**

Κλειδώνει ένα mutex.

### **Σύνταξη:**



MutexLock( id )

### **Παράμετροι:**

id

To Any Ptr handle του mutex που θα κλειδωθεί.

### **Περιγραφή:**

Το Mutexlock σταματά οποιαδήποτε άλλα νήματα χρησιμοποιώντας ένα "handle" ενός mutex, που δημιουργήθηκε από το MutexCreate, μέχρι να ξεκλειδώσει το handle με το MutexUnlock.

Ένα τέτοιο σταματημένο νήμα αναστέλλει την εκτέλεσή του και δεν καταναλώνει χρόνο CPU μέχρι να ξεκλειδώσει το mutex.

Για παραδείγματα δείτε το ThreadCreate.

## MutexUnlock

Απελευθερώνει ένα κλείδωμα ενός mutex.

### Σύνταξη:



MutexUnlock( id )

### Παράμετροι:

id

To Any Ptr handle του mutex που θα ξεκλειδωθεί.

### Περιγραφή:

Το Mutexunlock απελευθερώνει ένα mutex handle που δημιουργήθηκε από το MutexCreate και κλειδώθηκε με το MutexLock. Αυτό επιτρέπει σε άλλα νήματα που μοιράζονται το mutex να συνεχίσουν την εκτέλεση.

Για παράδειγμα δείτε το ThreadCreate.

## MutexDestroy

Καταστρέφει ένα mutex.

### Σύνταξη:



MutexDestroy( id )

### Παράμετροι:

id

To Any Ptr handle του mutex που θα καταστραφεί.

### **Περιγραφή:**

Το `MutexDestroy` καταστρέφει ένα `mutex` που δημιουργήθηκε από το `MutexCreate`. Αυτή η κλήση πρέπει να εκτελεστεί αφού τα νήματα που χρησιμοποιούν το `mutex` δεν χρησιμοποιούνται πλέον.

Για παράδειγμα δείτε το `ThreadCreate`.

### **CondCreate**

Δημιουργεί μια μεταβλητή υπό όρους για χρήση στο συγχρονισμό νημάτων.

### **Σύνταξη:**



`result = CondCreate`

### **Επιστρεφόμενη τιμή:**

Επιστρέφει ένα `handle` σε μια νέα μεταβλητή υπό όρους ή του μηδενικό δείκτη (0) σε περίπτωση αποτυχίας.

### **Περιγραφή:**

Μόλις η μεταβλητή δημιουργηθεί με την `Condocreate` και τα νήματα ξεκινήσουν, ένα ή περισσότερα από αυτά (συμπεριλαμβανομένου του σιωπηρού κύριου προγράμματος εκτέλεσης του κύριου προγράμματος) μπορούν να ρυθμιστούν σε `CondWait`, θα σταματήσουν έως ότου κάποιο νήμα λάβει `CondSignals` που μπορεί να επανεκκινήσει το νήμα αναμονής.

Το `CondBroadcast` μπορεί να χρησιμοποιηθεί για επανεκκίνηση όλων των νημάτων που περιμένουν την μεταβλητή υπό όρους. Στο τέλος του προγράμματος, το `CondDestroy` πρέπει να χρησιμοποιηθεί για να αποφευχθεί η διαρροή πόρων στο λειτουργικό σύστημα.



## Παράδειγμα - condcreate.bas



```
1. ''
2. '' make newly-created threads
3. '' wait until all threads are ready,
4. '' then start them all at once
5. ''
6. Dim Shared hcondstart As Any Ptr
7. Dim Shared hmutexstart As Any Ptr
8. Dim Shared start As Integer = 0
9. Dim Shared threadcount As Integer
10. Dim Shared hmutexready As Any Ptr
11. Dim Shared hcondready As Any Ptr
12.
13. Sub mythread(ByVal id_ptr As Any Ptr)
14.     Dim id As Integer = Cast(Integer, id_ptr)
15.
16.     '' signal that this thread is ready
17.     MutexLock hmutexready
18.     threadcount += 1
19.     Print "Thread #" & id & " is waiting..."
20.     CondSignal hcondready
21.     MutexUnlock hmutexready
22.
23.
24.     '' wait for the start signal
25.     MutexLock hmutexstart
26.     Do While start = 0
27.         CondWait hcondstart, hmutexstart
28.     Loop
29.
30.     '' now this thread holds the lock on hmutexstart
31.
32.     MutexUnlock hmutexstart
33.     '' print out the number of this thread
34.     For i As Integer = 1 To 40
35.         Print id;
36.     Next i
37. End Sub
38.
39. Dim threads(1 To 9) As Any Ptr
40. hcondstart = CondCreate()
41. hmutexstart = MutexCreate()
42. hcondready = CondCreate()
43. hmutexready = MutexCreate()
44. threadcount = 0
45. MutexLock(hmutexready)
46.
47. For i As Integer = 1 To 9
```

```

48.     threads(i) = ThreadCreate(@mythread, Cast(Any
Ptr, i))
49.     If threads(i) = 0 Then
50.         Print "unable to create thread"
51.     End If
52. Next i
53.
54. Print "Waiting until all threads are ready..."
55.
56. Do Until threadcount = 9
57.     CondWait(hcondready, hmutexready)
58. Loop
59.
60. MutexUnlock(hmutexready)
61.
62. Print
63. Print "Go!"
64. MutexLock hmutexstart
65. start = 1
66. CondBroadcast hcondstart
67. MutexUnlock hmutexstart
68. '' wait for all threads to complete
69. For i As Integer = 1 To 9
70.     If threads(i) <> 0 Then
71.         ThreadWait threads(i)
72.     End If
73. Next i
74.
75.
76. MutexDestroy hmutexready
77. CondDestroy hcondready
78. MutexDestroy hmutexstart
79. CondDestroy hcondstart
80.
81. Sleep
82. End

```

## Έξοδος:



```

Waiting until all threads are ready...
Thread #1 is waiting...
Thread #2 is waiting...
Thread #3 is waiting...
Thread #4 is waiting...
Thread #5 is waiting...
Thread #6 is waiting...

```

```
Thread #7 is waiting...
Thread #8 is waiting...
Thread #9 is waiting...
```

Go!

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
8 8 8 8 8 8 8 8 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
```

## CondWait

Διακόπτει την εκτέλεση του τρέχοντος νήματος έως ότου κάποια συνθήκη γίνει αληθινή.

### Σύνταξη:



CondWait ( handle, mutex )

## **Παράμετροι:**

### handle

Το handle της υπό όρους μεταβλητής.

### mutex

Το mutex που σχετίζεται με αυτήν τη μεταβλητή υπό όρους, η οποία πρέπει να κλειδωθεί κατά τον έλεγχο της κατάστασης και την κλήση της CondWait.

## **Περιγραφή:**

Λειτουργία που σταματά το νήμα εκεί που καλείται μέχρι κάποιο άλλο νήμα στείλει σήμα με την CondSignal ή την CondBroadcast στο handle του νήματος.

Μόλις δημιουργηθεί η μεταβλητή υπό όρους με το CondCreate και ξεκινήσουν τα νήματα, ένα από αυτά (συμπεριλαμβανομένου του σιωπηρού κύριου προγράμματος που εκτελεί το κύριο πρόγραμμα) μπορεί να ρυθμιστεί σε CondWait. Τότε το νήμα θα σταματήσει μέχρι κάποιο άλλο νήμα στείλει σήμα με την CondSignal που μπορεί να κάνει επανεκκίνηση του νήματος αναμονής. Το CondBroadcast μπορεί να χρησιμοποιηθεί για επανεκκίνηση όλων των νημάτων που περιμένουν την μεταβλητή υπό όρους. Στο τέλος του προγράμματος, το CondDestroy πρέπει να χρησιμοποιηθεί για να αποφευχθεί η διαρροή πόρων στο λειτουργικό σύστημα.

Όταν καλείτε το CondWait, το mutex πρέπει να είναι ήδη κλειδωμένο (χρησιμοποιώντας το ίδιο mutex με αυτό που χρησιμοποιείται με το CondSignal ή το CondBroadcast). Θα προκύψει ατομικό ξεκλείδωμα του mutex και αναμονή για τη μεταβλητή υπό όρους. Η εκτέλεση του νήματος κλήσης αναστέλλεται και δεν καταναλώνει χρόνο CPU έως ότου σηματοδοτηθεί η μεταβλητή συνθήκης. Όταν η μεταβλητή συνθήκης σηματοδοτηθεί, το mutex θα κλειδωθεί ξανά και στη συνέχεια η εκτέλεση θα επιστρέψει στο νήμα μετά την κλήση CondWait, αλλά με το mutex που ανήκει στον καλούντα. Ο καλούντας είναι στη συνέχεια υπεύθυνος για το ξεκλείδωμα του mutex προκειμένου να ολοκληρωθεί η υπορουτίνα CondWait, έτσι ώστε να μπορεί να συνεχιστεί η εκτέλεση μετά την κλήση CondWait.

Σημείωση: Είναι καλή συνήθεια να χρησιμοποιείτε το CondWait με προστατευμένο τρόπο έναντι πιθανών παραπλανητικών αφυπνίσεων.

Για το λόγο αυτό, το CondWait τίθεται σε έναν βρόχο για να ελέγξει αν ένα Boolean όρισμα είναι όντως αληθές (το όρισμα ορίστηκε true από ένα άλλο νήμα λίγο πριν την εκτέλεση του CondSignal ή του CondBroadcast) όταν το νήμα έχει τελειώσει με την αναμονή:

```
While predicate <> true : Condwait(handle, mutex) : Wend  
[ : predicate = false ]
```

ο βρόχος μπορεί να τερματιστεί μόνο όταν το όρισμα είναι αληθές.

Από την άλλη πλευρά, εάν το όρισμα είναι ήδη αληθές πριν το νήμα φτάσει στον βρόχο, το CondWait παραλείπεται εντελώς (επιτρέποντας να ληφθεί υπόψη μια περίπτωση CondSignal ή CondBroadcast που θα είχε χαθεί διαφορετικά, επειδή εκτελέστηκε πρόωρα σε ένα δεύτερο νήμα πριν από την το πρώτο νήμα το περιμένει)

Για παράδειγμα δείτε το CondCreate ή το CondSignal.

## CondSignal

Επανεκκίνηση ενός νήματος που έχει ανασταλεί από μια κλήση στο CondWait.

### Σύνταξη:



CondSignal ( handle )

## **Παράμετροι:**

handle

Το handle μιας μεταβλητής υπό όρους.

## **Περιγραφή:**

Μόλις δημιουργηθεί η μεταβλητή υπό όρους με το CondCreate και ξεκινήσουν τα νήματα, ένα από αυτά (συμπεριλαμβανομένου του κύριου προγράμματος εκτέλεσης του κύριου νήματος) μπορεί να οριστεί σε CondWait, έτσι θα σταματήσει έως ότου κάποιο άλλο νήμα στείλει σήμα με το CondSignal ότι το νήμα αναμονής μπορεί να κάνει επανεκκίνηση. Το CondBroadcast μπορεί να χρησιμοποιηθεί για επανεκκίνηση όλων των νημάτων που περιμένουν την μεταβλητή υπό όρους. Στο τέλος του προγράμματος, το CondDestroy πρέπει να χρησιμοποιηθεί για να αποφευχθεί η διαρροή πόρων στο λειτουργικό σύστημα.

Το Condsignal κάνει επανεκκίνηση ενός νήματος σε αναμονή. Θα πρέπει να καλείται μετά το κλείδωμα του mutex (χρησιμοποιώντας το ίδιο mutex με αυτό που χρησιμοποιείται με το CondWait). Εάν δεν περιμένουν νήματα υπό όρους, τίποτα δεν συμβαίνει (το σήμα χάνεται για πάντα). αν περιμένουν αρκετά, μόνο ένα επανεκκινείται. Στη συνέχεια, ο καλών πρέπει να ξεκλειδώσει το mutex προκειμένου να ολοκληρωθεί η υπορουτίνα CondWait.

## Παράδειγμα - condsignal.bas



```
1. Dim Shared As Any Ptr mutex
2. Dim Shared As Any Ptr cond
3. Dim Shared As String txt
4. Dim As Any Ptr pt
5. Dim Shared As Integer ok = 0
6.
7. Sub thread (ByVal p As Any Ptr)
8.     Print "thread is complementing the string"
9.     MutexLock(mutex)
10.    Sleep 400, 1
11.    txt &= " complemented by thread"
12.    ok = 1
13.    CondSignal(cond)
14.    MutexUnlock(mutex)
15.    Print "thread signals the processing
    completed"
16. End Sub
17.
18. mutex = Mutexcreate
19. cond = CondCreate
20. txt = "example of text"
21.
22. Print "main() initializes a string = " & txt
23. Print "main creates one thread"
24. Print
25.
26.
27. pt = ThreadCreate(@thread)
28. MutexLock(mutex)
29.
30. While ok <> 1
31.     CondWait(cond, mutex)
32. Wend
33.
34. Print
35. Print "back in main(), the string = " & txt
36. ok = 0
37.
38. MutexUnlock(mutex)
39. ThreadWait(pt)
40. MutexDestroy(mutex)
41. CondDestroy(cond)
42.
43.
44. Sleep
45. End
```

## Έξοδος:



```
main() initializes a string = example of text  
main creates one thread
```

```
thread is complementing the string  
thread signals the processing completed
```

```
back in main(), the string = example of text  
complemented by thread
```

## CondBroadcast

Επανεκκίνηση όλων των νημάτων που είναι σε αναμονή από το CondWait.

### Σύνταξη:



CondBroadcast ( handle )

### Παράμετροι:

handle

Το handle της μεταβλητής υπό όρους.

### Περιγραφή:

Μόλις η συνθήκη της μεταβλητής υπό όρους είναι CondCreate και ξεκινήσουν τα νήματα, ένα από αυτά (συμπεριλαμβανομένου του κύριου προγράμματος εκτέλεσης του σιωπηρού κύριου νήματος) μπορεί να οριστεί σε CondWait, έτσι θα σταματήσει έως ότου κάποιο νήμα στείλει σήμα με το CondSignal που μπορεί να επανεκκινήσει το νήμα αναμονής. Το CondBroadcast μπορεί να χρησιμοποιηθεί για επανεκκίνηση όλων των νημάτων που περιμένουν την μεταβλητή υπό όρους. Στο τέλος του προγράμματος, το



CondDestroy πρέπει να χρησιμοποιηθεί για να αποφευχθεί η διαρροή πόρων στο λειτουργικό σύστημα.  
Το Condbroadcast πρέπει να χρησιμοποιείται αντί για CondSignal για επανεκκίνηση όλων των νημάτων που περιμένουν υπό όρους.

Για παράδειγμα δείτε το CondCreate.

## CondDestroy

Καταστρέφει μια μεταβλητή υπό όρους πολλαπλών νημάτων όταν δεν είναι πλέον απαραίτητη.

### Σύνταξη:



CondDestroy ( handle )

### Παράμετροι:

handle

Το handle της μεταβλητής υπό όρους που θα καταστραφεί.

### Περιγραφή:

Μόλις ο όρος CondCreated και τα νήματα ξεκινήσουν, ένα από αυτά (συμπεριλαμβανομένου του κύριου προγράμματος εκτέλεσης του κύριου νήματος) μπορεί να ρυθμιστεί σε CondWait για τον όρο, έτσι τα νήματα θα σταματήσουν έως ότου κάποιο νήμα στείλει σήμα με την CondSignal που μπορεί να επανεκκινήσει το νήμα αναμονής. Το CondBroadcast μπορεί να χρησιμοποιηθεί για επανεκκίνηση όλων των νημάτων που περιμένουν τον όρο υπό όρους. Στο τέλος του προγράμματος, το CondDestroy πρέπει να χρησιμοποιηθεί για να αποφευχθεί η διαρροή πόρων στο λειτουργικό σύστημα.

Το Conddestroy καταστρέφει μια μεταβλητή συνθήκης, απελευθερώνοντας τους πόρους που μπορεί να έχει. Δεν πρέπει να περιμένουν νήματα για τη μεταβλητή συνθήκης κατά την είσοδο στο Conddestroy.

Για παραδείγματα δείτε την CondCreate ή την CondSignal.

## Συναρτήσεις εισόδου χρήστη - User Input Functions

### Input

Διαβάζει μια λίστα τιμών από το πληκτρολόγιο.

### Σύνταξη:



```
Input [;] ["prompt" ,|; ] variable_list
```

### Παράμετροι:

#### prompt

μια προαιρετική συμβολοσειρά που γράφεται στην οθόνη ως προτροπή. Εάν ακολουθείται από semicolon (;), ένα ερωτηματικό ("?") θα προσαρτηθεί στη γραμμή προτροπής. Εάν ακολουθεί κόμμα, τίποτα δεν θα προσαρτηθεί.

#### variable\_list

μια λίστα μεταβλητών διαχωρισμένων με κόμμα που χρησιμοποιούνται για τη διατήρηση των τιμών που διαβάζονται από τον χρήστη.

### Περιγραφή:

Διαβάζει τιμές μιας λίστας από το πληκτρολόγιο μέχρι την πρώτη επιστροφή του φορέα. Οι αριθμητικές τιμές μετατρέπονται από την αναπαράσταση συμβολοσειρών τους στους αντίστοιχους τύπους στη λίστα μεταβλητών. Οι χαρακτήρες αντηχούνται στην οθόνη καθώς πληκτρολογούνται.

Εάν υπάρχουν περισσότερες από μία τιμές στη λίστα εισαγωγής, τότε η γραμμή εισαγωγής θα χωριστεί με σάρωση για οριοθέτες - κόμματα (,) μετά από συμβολοσειρές ή

κόμματα και κενό διάστημα μετά τους αριθμούς. Ο περιφερειακός κενός χώρος θα περικοπεί από τιμές συμβολοσειράς. Εάν μια συμβολοσειρά εισόδου έχει κόμμα, πρέπει να τυλιχτεί σε εισαγωγικά ("...") για να αποτραπεί η διάσπασή της.

Για εισαγωγή σε μια συμβολοσειρά χωρίς οριοθέτηση, θα πρέπει να χρησιμοποιηθεί αντ' αυτού η εισαγωγή γραμμής (Line Input).

Η προτροπή - εάν υπάρχει - γράφεται στην οθόνη στην τρέχουσα θέση του δρομέα και οι χαρακτήρες που διαβάζονται αντηχούνται στην οθόνη αμέσως μετά την προτροπή. Εάν δεν έχει καθοριστεί προτροπή, οι χαρακτήρες επαναλαμβάνονται στην τρέχουσα θέση του δρομέα.

Το προαιρετικό κύριο ερωτηματικό (;) μετά την εισαγωγή είναι παρόμοιο με το προαιρετικό τελικό ερωτηματικό σε μια δήλωση Εκτύπωσης: ο δρομέας θα παραμείνει στην ίδια γραμμή μετά την επανάληψη όλων των χαρακτήρων, διαφορετικά, ο δρομέας θα μετακινηθεί στην αρχή της επόμενης γραμμής.

Εάν διαβαστούν περισσότερες τιμές από αυτές που αναφέρονται στη λίστα μεταβλητών, οι επιπλέον τιμές θα αγνοηθούν. εάν διαβαστούν λιγότερες τιμές (δηλ. ο χρήστης πατά enter πριν εισαγάγει όλες τις τιμές), οι υπόλοιπες μεταβλητές θα αρχικοποιηθούν (αριθμητικές μεταβλητές στο μηδέν (0) και μεταβλητές συμβολοσειράς στην κενή συμβολοσειρά ("")).

Οι αριθμητικές τιμές μετατρέπονται χρησιμοποιώντας παρόμοιες μεθόδους με τις διαδικασίες Val και ValLng, χρησιμοποιώντας την πιο κατάλληλη συνάρτηση για τη μορφή αριθμών, μετατρέποντας όσο το δυνατόν περισσότερους αριθμητικούς χαρακτήρες.

Η εισαγωγή έχει περιορισμένη ικανότητα επεξεργασίας: επιτρέπει τη χρήση του αριστερού και του δεξιού πλήκτρου δρομέα για πλοήγηση στο κείμενο και διαγραφή ή εισαγωγή χαρακτήρων. Εάν απαιτείται καλύτερη διεπαφή χρήστη, θα πρέπει να χρησιμοποιηθεί μια προσαρμοσμένη ρουτίνα εισόδου.

## Παράδειγμα - input2.bas



```
1. Dim user_name As String, user_age As Integer
2.
3. Input "Enter your name and age, separated by a
   comma: ", user_name, user_age
4.
5. Print "Your name is " & user_name & ", and you
   are " & user_age & " years old."
6.
7. Sleep
8. End
```

## Έξοδος:



```
Enter your name and age, separated by a
comma: Dim, 44
Your name is Dim, and you are 44 years old.
```

## Line Input

Διαβάζει μία γραμμή εισόδου από το πληκτρολόγιο.

## Σύνταξη:



Line Input [;] [promptstring {;|,} ] stringvariable

## Παράμετροι:

promptstring

προτροπή πριν την αναμονή για εισαγωγή

stringvariable

μεταβλητή για λήψη της γραμμής κειμένου

## Περιγραφή:

Διαβάζει μια γραμμή κειμένου από το πληκτρολόγιο και την αποθηκεύει σε μια μεταβλητή συμβολοσειράς.

## Παράδειγμα - lineinput.bas



```
1. Dim x As String
2. Line Input "Enter a line:", x
3. Print "You entered '"; x; "'"
4.
5. Sleep
6. End
```

## Έξοδος:



```
Enter a line:this is a test!
You entered 'this is a test!'
```

## Input()

Διαβάζει έναν αριθμό χαρακτήρων από την κονσόλα ή το αρχείο.

## Σύνταξη:



```
result = Input[$]( n [, [#]filenum ] )
```

## Παράμετροι:

n

Αριθμός bytes για ανάγνωση.

filenum

Αριθμός αρχείου συνδεδεμένου αρχείου ή συσκευής.

## Επιστρεφόμενη τιμή:

Επιστρέφει μια συμβολοσειρά των χαρακτήρων που διαβάστηκαν.

## Περιγραφή:

Διαβάζει έναν αριθμό χαρακτήρων από την κονσόλα ή ένα δεσμευμένο αρχείο/συσκευή που καθορίζεται από τον αριθμό αρχείου.

Η πρώτη έκδοση περιμένει και διαβάζει η χαρακτήρες από το buffer του πληκτρολογίου. Οι εκτεταμένοι χαρακτήρες δεν διαβάζονται. Οι χαρακτήρες δεν αντηχούνται στην οθόνη. Η δεύτερη έκδοση περιμένει και διαβάζει η χαρακτήρες από ένα αρχείο ή μια συσκευή. Η θέση του αρχείου ενημερώνεται.

## Παράδειγμα:



```
1. Print "Select a color by number"  
2. Print "1. blue"  
3. Print "2. red"  
4. Print "3. green"  
5. Dim choice As String  
6. Do  
7.     choice = Input(1)  
8. Loop Until choice >= "1" And choice <= "3"  
9.  
10. Print "You choose: "; choice  
11.  
12. Sleep  
13. End
```

## Έξοδος:



```
Select a color by number  
1. blue  
2. red  
3. green  
You choose: 3
```

## Winput()

Διαβάζει έναν αριθμό ευρέων χαρακτήρων από την κονσόλα ή το αρχείο.

### Σύνταξη:



```
result = WInput( num [, [#]filename } )
```

### Παράμετροι:

num

Αριθμός χαρακτήρων για ανάγνωση.

filename

Αριθμός αρχείου συνδεδεμένου αρχείου ή συσκευής.

### Επιστρεφόμενη τιμή:

Επιστρέφει ένα WString των αναγνωσμένων χαρακτήρων.

### Περιγραφή:

Διαβάζει έναν αριθμό ευρέων χαρακτήρων από την κονσόλα ή ένα δεσμευμένο αρχείο/συσκευή που καθορίζεται από το αρχείο.

Η πρώτη έκδοση περιμένει και διαβάζει η χαρακτήρες από το buffer του πληκτρολογίου. Οι εκτεταμένοι χαρακτήρες δεν διαβάζονται. Οι χαρακτήρες δεν αντηχούνται στην οθόνη.

Η δεύτερη έκδοση περιμένει και διαβάζει η χαρακτήρες από ένα αρχείο ή μια συσκευή. Η θέση του αρχείου ενημερώνεται.

Σημείωση: Το FreeBASIC δεν υποστηρίζει προς το παρόν την ανάγνωση ευρέων χαρακτήρων από την κονσόλα.

## Παράδειγμα - winput.bas



```
1. Dim char As WString * 2
2. Dim filename As String, enc As String
3. Dim f As Integer
4. Line Input "Please enter a file name: ", filename
5. Line Input "Please enter an encoding type
   (optional): ", enc
6. If enc = "" Then enc = "ascii"
7. f = FreeFile
8. If Open(filename For Input Encoding enc As #f) =
   0 Then
9.
10.     Print "Press space to read a character from
   the file, or escape to exit."
11.
12.     Do
13.
14.         Select Case Input(1)
15.
16.             Case " " 'Space
17.
18.                 If EOF(f) Then
19.
20.                     Print "You have reached the end
21. of the file."
22.                     Exit Do
23.
24.                 End If
25.
26.                 char = WInput(1, f)
27.                 Print char & "
   (char no " & Asc(char) & ")"
28.
29.                 Case Chr(27) 'Escape
30.
31.                     Exit Do
32.
33.                 End Select
34.
35.             Loop
36.
37.             Close #f
38.
39.         Else
40.
41.
```



```
42.         Print "There was an error opening the file."  
43.  
44.     End If  
45.  
46.     Sleep  
47. End  
48.
```

## Inkey

Επιστρέφει μια συμβολοσειρά που αντιπροσωπεύει το πρώτο κλειδί που περιμένει στο buffer πληκτρολογίου.

### Σύνταξη:



result = Inkey[\$]

### Επιστρεφόμενη τιμή:

Ο πρώτος χαρακτήρας που βρέθηκε στο buffer πληκτρολογίου ή μια κενή συμβολοσειρά ("" ) αν δεν βρέθηκε.

### Περιγραφή:

Βλέπει το buffer του πληκτρολογίου και επιστρέφει μια αναπαράσταση συμβολοσειράς του πρώτου χαρακτήρα, αν υπάρχει, που βρέθηκε. Ο χαρακτήρας στη συνέχεια αφαιρείται από το buffer και δεν επαναλαμβάνεται στην οθόνη. Εάν το buffer πληκτρολογίου είναι κενό, επιστρέφει αμέσως μια κενή συμβολοσειρά ("" ) χωρίς να περιμένετε χαρακτήρες.

Εάν ο χαρακτήρας βρίσκεται στο σύνολο χαρακτήρων ASCII, επιστρέφεται μια συμβολοσειρά ενός χαρακτήρα που αποτελείται από αυτόν τον χαρακτήρα. Εάν το κλειδί είναι ένα "εκτεταμένο" (αριθμητικό πληκτρολόγιο, δρομείς) επιστρέφεται μια συμβολοσειρά δύο χαρακτήρων, η πρώτη από την οποία είναι ο εκτεταμένος χαρακτήρας.

Τα πλήκτρα Shift, Ctrl, Alt και AltGr δεν μπορούν να διαβαστούν ανεξάρτητα από αυτήν τη συνάρτηση, καθώς δεν

εισέρχονται ποτέ στο buffer του πληκτρολογίου (αν και, ίσως, προφανώς, το Shift-A θα αναφέρεται από το Inkey διαφορετικά από το Control-A κτλ

## Παράδειγμα - inkey.bas



```
1. Print "press q to quit"
2. Do
3.     Sleep 1, 1
4. Loop Until Inkey = "q"
5.
6.
7.
8. ' ' Compile with -lang fblite or qb
9. #lang "fblite"
10.
11. #if __FB_LANG__ = "qb"
12. #define EXTCHAR Chr$(0)
13. #else
14. #define EXTCHAR Chr(255)
15. #endif
16.
17. Dim k As String
18. Print "Press a key, or Escape to end"
19. Do
20.     k = Inkey$
21.     Select Case k
22.         Case "A" To "Z", "a" To "z": Print
23. "Letter: " & k
24.         Case "1" To "9": Print
25. "Number: " & k
26.         Case Chr$(32): Print "Space"
27.         Case Chr$(27): Print "Escape"
28.         Case Chr$(9): Print "Tab"
29.         Case Chr$(8): Print "Backspace"
30.         Case Chr$(32) To Chr$(127)
31.             Print "Printable character: " & k
32.         Case EXTCHAR & "G": Print "Up Left /
33. Home"
34.         Case EXTCHAR & "H": Print "Up"
35.         Case EXTCHAR & "I": Print "Up Right /
36. PgUp"
37.         Case EXTCHAR & "K": Print "Left"
38.         Case EXTCHAR & "L": Print "Center"
39.         Case EXTCHAR & "M": Print "Right"
40.         Case EXTCHAR & "O": Print "Down Left /
41. End"
42.         Case EXTCHAR & "P": Print "Down"
```

```

37.         Case EXTCHAR & "Q": Print "Down Right /
PgDn"
38.         Case EXTCHAR & "R": Print "Insert"
39.         Case EXTCHAR & "S": Print "Delete"
         Case EXTCHAR & "k": Print "Close window /
Alt-F4"
40.
         Case EXTCHAR & Chr$(59) To
41. EXTCHAR & Chr$(68)
42.         Print "Function key: F" & Asc(k, 2) -
58
43.         Case EXTCHAR & Chr$(133) To EXTCHAR &
44. Chr$(134)
45.         Print "Function key: F" & Asc(k, 2) -
122
46.         Case Else
47.             If Len(k) = 2 Then
48.                 Print Using "Extended character:
chr$(###, ###)"; Asc(k, 1); Asc(k, 2)
49.                 ElseIf Len(k) = 1 Then
50.                     Print Using "Character chr$(
(###)"; Asc(k)
51.                     End If
52.             End Select
53.             If k = Chr$(27) Then Exit Do
54.             Sleep 1, 1
55.         Loop
56.
57.     Sleep
58. End

```

## Έξοδος:



```

press q to quit
Press a key, or Escape to end
Letter: q
Letter: w
Letter: e
Letter: r
Letter: t
Letter: y

```

## GetKey

Επιστρέφει τον κωδικό ascii του πρώτου χαρακτήρα στο buffer πληκτρολογίου.

### Σύνταξη:



```
result = GetKey
```

### Επιστρεφόμενη τιμή:

Η τιμή του κωδικού ascii που επιστρέφεται.

### Περιγραφή:

Επιστρέφει τον κωδικό ascii του πρώτου χαρακτήρα στο buffer πληκτρολογίου. Ο χαρακτήρας αφαιρείται από το buffer. Εάν δεν υπάρχει χαρακτήρας, το GetKey το περιμένει. Για εκτεταμένα κλειδιά (επιστροφή δύο χαρακτήρων), ο εκτεταμένος κώδικας επιστρέφεται στο πρώτο byte και ο κανονικός κώδικας επιστρέφεται στο δεύτερο byte (το τρίτο και το τέταρτο byte είναι πάντα null τουλάχιστον στη λειτουργία κονσόλας).

**ΠΡΟΕΙΔΟΠΟΙΗΣΗ:** Στη λειτουργία γραφικών και ανάλογα με το πατημένο πλήκτρο, το Getkey ενδέχεται να μην επιστρέφει πάντα την ίδια ακριβώς τιμή όπως στη λειτουργία κονσόλας (για ένα μη εκτεταμένο χαρακτήρα, το πιο σημαντικό κομμάτι του byte κωδικού ascii μπορεί να διαδοθεί στα υψηλότερα 3 byte της τιμής επιστροφής, όπως ένα προσημασμένο bit).

Το πλήκτρο ανάγνωσης δεν αντηχεί στην οθόνη.

Για μια λέξη-κλειδί που δεν σταματά το πρόγραμμα εάν δεν υπάρχει χαρακτήρας στο buffer, δείτε Inkey ή MultiKey.

## Παράδειγμα - getKey.bas



```
1. Dim As Long foo
2. Do
3.     foo = GetKey
4.     Print "total return: " & foo
5.
6.     If( foo > 255 ) Then
7.         Print "extended code: " & (foo And &hff)
8.         Print "regular code: " & (foo Shr 8)
9.     Else
10.        Print "regular code: " & (foo And &hff)
11.    End If
12.    Print
13. ' loop until ESCAPE pressed
14. Loop Until foo = 27
15.
16. Sleep
17. End
```

Τρέχουμε το πρόγραμμά και πατάμε το πλήκτρο **a**.

**Έξοδος:**



```
total return: 97
regular code: 97
```

# Κεφάλαιο 12ο - Διάλεκτοι FreeBASIC

Η γλώσσα FreeBASIC μας δίνει την δυνατότητα να γράφουμε προγράμματα συμβατά με άλλες εκδόσεις γλωσσών basic όπως η GW-BASIC ή η QuickBASIC.

Αυτό γίνεται με την επιλογή διαλέκτου γλώσσας με τον διακόπτη `-lang` όταν τρέχουμε τον μεταγλωττιστή `fbc`.

Η έκδοση FreeBASIC 0.17b εισάγει μια επιλογή γραμμής εντολών `-lang`, που χρησιμοποιείται για την αλλαγή της λειτουργίας συμβατότητας γλώσσας για διαφορετικές διαλέκτους της βασικής γλώσσας.

-lang επιλογή	Περιγραφή
<code>fb</code>	Συμβατότητα FreeBASIC (προεπιλογή)
<code>qb</code>	<code>qbasic</code> συμβατότητα
<code>fb-lite</code>	Συμβατότητα γλώσσας FreeBASIC, με πιο συμβατό στυλ κωδικοποίησης με QBASIC
<code>deprecated</code>	συμβατότητα με FB 0.16

Για περισσότερες λεπτομέρειες για τις διαλέκτους της FreeBASIC δείτε στο

<https://www.freebasic.net/wiki/CompilerDialects>

## **Κεφάλαιο 13ο - Προεπεξεργαστής (Preprocessor)**

Ο Προεπεξεργαστής εκτελεί κάποια επεξεργασία στον πηγαίο κώδικα πριν από το επόμενο βήμα της μεταγλώττισης.

Ο προεπεξεργαστής είναι ένα πρόγραμμα που αναλύει ένα αρχείο κειμένου και το κάνει να υποβληθεί σε ορισμένους μετασχηματισμούς.

Αυτοί οι μετασχηματισμοί μπορεί να είναι η συμπερίληψη ενός αρχείου, η διαγραφή ενός μπλοκ κειμένου ή η αντικατάσταση ενός μπλοκ κειμένου.

Ο προεπεξεργαστής εκτελεί αυτές τις λειτουργίες μέσω συγκεκριμένων εντολών που διαβάζει στο αρχείο που σαρώνεται.

Καλείται αυτόματα από τον μεταγλωττιστή, πριν από τη μεταγλώττιση, για την επεξεργασία των αρχείων για μεταγλώττιση.

### **Εντολές προεπεξεργαστή**

Όλες οι εντολές προεπεξεργαστή ξεκινούν από την αρχή της γραμμής με το σύμβολο ( «#»).

Οι εντολές προεπεξεργαστή στέλνονται στον μεταγλωττιστή για να ελέγξουν τι μεταγλωττίζεται και πώς. Μπορούν να χρησιμοποιηθούν για να επιλέξουν τη μεταγλώττιση ενός μπλοκ κώδικα παρά ενός άλλου για συμβατότητα μεταξύ πλατφορμών, να περιλαμβάνουν κεφαλίδες ή άλλα αρχεία προέλευσης, να ορίσουν μικρές ενσωματωμένες λειτουργίες που ονομάζονται μακροεντολές ή να αλλάξουν τον τρόπο με τον οποίο ο μεταγλωττιστής χειρίζεται μεταβλητές.

Οι εντολές του προεπεξεργαστή διακρίνονται στις εξής κατηγορίες:

### **Υποθετική Μεταγλώττιση (Conditional Compilation)**

Εντολές που επιτρέπουν τη διαμόρφωση μερών κώδικα βάσει συνθηκών.

### **Αντικατάσταση κειμένου (Text Replacement)**

Εντολές που δημιουργούν μακροεντολές αντικατάστασης κειμένου.

### **Οδηγίες αρχείων (File Directives)**

Εντολές που υποδεικνύουν στον μεταγλωττιστή πώς σχετίζονται άλλα αρχεία με το αρχείο προέλευσης.

### **Οδηγίες ελέγχου (Control Directives)**

Εντολές που ορίζουν επιλογές μεταγλώττισης, ελέγχουν τη μεταγλώττιση και αναφέρουν πληροφορίες μεταγλώττισης χρόνου.

### **Μεταεντολές (Metacommands)**

Εντολές που διατηρούνται για συμβατότητα προς τα πίσω.

## **Υποθετική Μεταγλώττιση (Conditional Compilation)**

### **#if...#elseif...#else...#endif**

Σύνταξη:



#if (expression1)

' Conditionally included statements if expression1 is True

#elseif (expression2)

' Conditionally included statements if expression2 is True

#else

' Conditionally included statements if both

' expression1 and expression2 are False

#endif



## Παράδειγμα:



```
1. #define WORDSIZE 16
2. #if (WORDSIZE = 16)
3.     ' Do some some 16 bit stuff
4. #elseif (WORDSIZE = 32)
5.     ' Do some some 32 bit stuff
6. #else
7.     #error WORDSIZE must be set To 16 Or 32
8. #endif
```

## Παράδειγμα:



```
1. #define MODULE_VERSION 1
2. Dim a As String
3. #if (MODULE_VERSION > 0)
4.     a = "Release"
5. #else
6.     a = "Beta"
7. #endif
8. Print "Program is "; a
```

## defined

Λειτουργία προεπεξεργαστή για να ελέγξετε εάν έχει οριστεί ένα σύμβολο.

## Σύνταξη:



defined (symbol\_name)

## Παράδειγμα:



```
1. Const a = 300
2. #define b 12
3. Dim c As Single
4. #if defined(a)
5.     Print "a is defined"
6. #endif
7. #if defined(b)
8.     Print "b is defined"
9. #endif
10. #if defined(c)
11.     Print "c is defined"
12. #endif
13. #if defined(d)
14.     Print "d is defined"
15. #Endif
16.
17. Sleep
18. End
```

## Έξοδος:



```
a is defined
b is defined
c is defined
```

## #ifdef

### Σύνταξη:



```
#ifdef symbol
' Conditionally included statements
#endif
```

### Περιγραφή:

Περιλαμβάνει υπό όρους δηλώσεις κατά τη μεταγλώττιση. Οι δηλώσεις εντός του μπλοκ `#ifdef ... #endif` περιλαμβάνονται εάν το σύμβολο έχει οριστεί και εξαιρούνται (αγνοούνται) εάν το σύμβολο δεν έχει οριστεί. Το σύμβολο `#ifdef` ισοδυναμεί με `#if defined (symbol)`

## Παράδειγμα:



```
1. #define _DEBUG
2. #ifdef _DEBUG
3.     ' Special statements for debugging
4. #endif
```

## #ifndef

### Σύνταξη:



#ifndef symbol  
' Conditionally included statements  
#endif

### Περιγραφή:

Περιλαμβάνει υπό όρους δηλώσεις κατά τη μεταγλώττιση. Οι δηλώσεις εντός του μπλοκ `#ifndef ... #endif` περιλαμβάνονται εάν το σύμβολο δεν έχει οριστεί και εξαιρούνται (αγνοούνται) εάν το σύμβολο έχει οριστεί. Το σύμβολο `#ifndef` ισοδυναμεί με `#if Not defined (symbol)`

## Παράδειγμα:



```
1. #ifndef __MYFILE_BI__
2. #define __MYFILE_BI__
3.     ' Declarations
4. #endif
```

# Αντικατάσταση κειμένου (Text Replacement)

## #define

### Σύνταξη:



```
#define identifier body  
#define identifier( [ parameters ] ) body  
#define identifier( [ parameters, ]  
Variadic_Parameter... ) body
```

### Περιγραφή:

Το `#define` επιτρέπει τη δήλωση μακροεντολών προεπεξεργαστή βασισμένου σε κείμενο. Μόλις ο μεταγλωττιστής δει ένα `#define`, θα αρχίσει να αντικαθιστά περαιτέρω εμφανίσεις αναγνωριστικού με το σώμα. Το σώμα μπορεί να είναι άδειο. Η επέκταση γίνεται αναδρομικά, έως ότου δεν υπάρχει τίποτα περισσότερο για επέκταση και ο μεταγλωττιστής μπορεί να συνεχίσει την ανάλυση του κώδικα που προκύπτει.

Το `#undef` μπορεί να χρησιμοποιηθεί για να κάνει τον μεταγλωττιστή να ξεχάσει ένα `#define`.

Οι παράμετροι μετατρέπουν ένα ορισμό σε μια μακροεντολή που μοιάζει με συνάρτηση, επιτρέποντας να μεταβιβάστουν ορίσματα κειμένου στη μακροεντολή. Οποιαδήποτε εμφάνιση των ονομάτων παραμέτρων στο σώμα θα αντικατασταθεί από το δεδομένο κείμενο ορίσματος κατά την επέκταση. Εάν μια κατά γράμμα μετάδοση γίνεται σε μια μακροεντολή, το όνομα της αντίστοιχης παραμέτρου στο σώμα μακροεντολής δεν μπορεί να χρησιμοποιηθεί ως τοπική μεταβλητή όπως σε ένα σώμα διαδικασίας. Για να μιμηθεί την ίδια λειτουργία όπως για μια διαδικασία, ο χρήστης πρέπει στη συνέχεια να δηλώσει ρητά μια τοπική μεταβλητή (με άλλο όνομα) στο σώμα της μακροεντολής και να την αρχικοποιήσει με το όνομα της παραμέτρου που πέρασε (αντικαταστάθηκε κατά την προεπεξεργασία από την κυριολεκτική μετάδοση).

Ο τελεστής # Stringize μπορεί να χρησιμοποιηθεί σε παραμέτρους μακροεντολών για να τις μετατρέψει σε κυριολεκτικούς χαρακτήρες συμβολοσειράς και ο τελεστής ## Concatenate μπορεί να χρησιμοποιηθεί για τη συγχώνευση κειμένου μαζί.

Σημείωση: Στη δήλωση #define που μοιάζει με συνάρτηση, το αναγνωριστικό θα πρέπει να ακολουθείται από τις παρενθέσεις ανοίγματος (()) αμέσως χωρίς κενό κενό ενδιάμεσα, διαφορετικά ο μεταγλωττιστής θα το αντιμετωπίσει ως μέρος του σώματος.

Οι ορισμοί είναι οριοθετημένοι, είναι ορατοί μόνο στο πεδίο στο οποίο ορίστηκαν. Εάν οριστεί σε επίπεδο ενότητας, ο ορισμός είναι ορατός σε ολόκληρη τη μονάδα. Εάν το αναγνωριστικό ορίζεται μέσα σε μια σύνθετη δήλωση που έχει πεδίο (Sub, For..Next, while..Wend, Do..Loop, Scope..End Scope κ.λπ.), το αναγνωριστικό που ορίζεται είναι ορατό μόνο σε αυτό το πεδίο. Τα ονόματα χώρων (namespaces) από την άλλη δεν έχουν καμία επίδραση στην ορατότητα ενός ορισμού.

Τα αναγνωριστικά μπορούν να ελεγχθούν με το #ifdef και άλλα, τα οποία μπορούν να χρησιμοποιηθούν για την απόκρυψη τμημάτων κώδικα από τον μεταγλωττιστή (σύνταξη υπό όρους).

Το αποτέλεσμα της επέκτασης μακροεντολής μπορεί να ελεγχθεί χρησιμοποιώντας την επιλογή μεταγλωττιστή -pp. Οι ορισμοί χρησιμοποιούνται συχνά για τη δήλωση σταθερών.

Η δήλωση Const είναι μια εναλλακτική λύση ασφαλής για τον τύπο.

**ΠΡΟΕΙΔΟΠΟΙΗΣΗ:** Όταν το σώμα ορισμού περιέχει μια έκφραση με τουλάχιστον έναν τελεστή, ενδέχεται να είναι υποχρεωτικό να περιβάλλετε ορισμένους όρους (παραμέτρους, ολόκληρο το σώμα) με παρενθέσεις για να μην υποβληθείτε σε ανεπιθύμητη αλλαγή προτεραιότητας τελεστών (εάν περάσετε ως όρισμα έκφραση με επίσης τελεστές, ή εάν χρησιμοποιείτε το ορισμό σε μια κύρια έκφραση με επίσης τελεστές).

## Παράδειγμα:



```
1.  '' Definition and check
2.  #define DEBUGGING
3.  #ifdef DEBUGGING
4.      ' ... statements
5.  #endif
6.
7.  '' Simple definition/text replacement
8.  #define False 0
9.  #define True (Not False)
10.
11. '' Function-like definition
12. #define MyRGB(R,G,B) (((R)Shl 16) Or ((G)Shl 8)
13. Or (B))
14. Print Hex( MyRGB(&hff, &h00, &hff) )
15. '' Line continuation and statements in a
16. '' definition
17. #define printval(bar) _
18.     Print #bar; " ="; bar
19.
20.
21. '' #defines are visible only
22. '' in the scope where they are defined
23. Scope
24.     #define LOCALDEF 1
25. End Scope
26. #ifndef LOCALDEF
27. #     Print LOCALDEF Is Not defined
28. #endif
29.
30. '' namespaces have no effect
31. '' on the visibility of a define
32. Namespace foo
33. #     define NSDEF
34. End Namespace
35. #ifdef NSDEF
36. #     Print NSDEF Is defined
37. #endif
```

## #Macro...#Endmacro

Οδηγία προεπεξεργαστή για τον ορισμό μιας μακροεντολής πολλαπλών γραμμών.

### Σύνταξη:



```
#macro identifier [?] ( [ parameters ] )  
body  
#endmacro  
ή  
#macro identifier [?] ( [ parameters, ]  
Variadic_Parameter... )  
body  
#endmacro
```

### Περιγραφή:

Το #macro είναι η έκδοση πολλών γραμμών του #define. Εάν χρησιμοποιείτε το προαιρετικό ερωτηματικό (?) Μετά το αναγνωριστικό στη σύνταξη ορισμού, μπορείτε να επικαλεστείτε μακροεντολές με παραμέτρους χωρίς να χρησιμοποιήσετε παρενθέσεις γύρω από τα ορίσματα.

**Σημείωση:** Προσοχή στη δυνατότητα ενεργοποίησης σύγκρουσης με εκφράσεις που περιέχουν το όνομα της μακροεντολής ως έναν από τους όρους τους.

**Σημείωση:** Σε αντίθεση με τη δήλωση #define που μοιάζει με λειτουργία, μπορούν να τοποθετηθούν κενά μεταξύ του ονόματος της μακροεντολής και της αρχικής παρένθεσης για οποιαδήποτε σύνταξη δήλωσης μακροεντολής.

**ΠΡΟΕΙΔΟΠΟΙΗΣΗ:** Στο σώμα μακροεντολής, μπορεί να είναι υποχρεωτικό να περιβάλλετε από παρενθέσεις οποιαδήποτε χρησιμοποιούμενη παράμετρο εάν βρίσκεται μέσα σε μια έκφραση με έναν τελεστή τουλάχιστον, προκειμένου να μην υποβληθείτε σε ανεπιθύμητη αλλαγή προτεραιότητας τελεστών (εάν περάσετε ως όρισμα μια έκφραση με τελεστές).

## Παράδειγμα - macro.bas



```
1. ' macro as an expression value
2. #macro Print1( a, b )
3.     a + b
4. #endmacro
5.
6. Print Print1( "Hello ", "World!" )
7.
8. ' macro as multiple statements
9. #macro Print2( a, b )
10.     Print a;
11.     Print " ";
12.     Print b;
13.     Print "!"
14. #endmacro
15.
16. Print2( "Hello", "World" )
17.
18.
19. Sleep
20. End
```

### Έξοδος:



```
Hello World!
Hello World!
```



## #undef

Οδηγία προπεξεργαστή για τον αποορισμό μιας μακροεντολής.

### Σύνταξη:



#undef symbol

### Περιγραφή:

Καταργεί τον ορισμό από ένα σύμβολο που είχε οριστεί προηγουμένως με #define.

Μπορεί να χρησιμοποιηθεί για να διασφαλιστεί ότι μια μακροεντολή ή ένα σύμβολο έχει περιορισμένη διάρκεια ζωής και δεν έρχεται σε αντίθεση με έναν παρόμοιο ορισμό μακροεντολής που μπορεί να οριστεί αργότερα στον πηγαίο κώδικα.

**Σημείωση:** Το #undef δεν πρέπει να χρησιμοποιείται για τον καθορισμό ονομάτων μεταβλητών ή συναρτήσεων που χρησιμοποιούνται στο τρέχον πεδίο συνάρτησης. Τα ονόματα χρειάζονται εσωτερικά από τον μεταγλωττιστή και η κατάργησή τους μπορεί να προκαλέσει περίεργα και απροσδόκητα αποτελέσματα.

### Παράδειγμα:



```
1. #define ADD2(a_, b_) ((a_) + (b_))
2. Print ADD2(1, 2)
3. ' Macro no longer needed so get rid of it ...
4. #undef ADD2
```

## Τελεστής # (Preprocessor Stringize)

Τελεστής προεπεξεργαστή για τη μετατροπή ορισμάτων μακροεντολών σε συμβολοσειρές.

### Σύνταξη:



#macro\_argument

### Περιγραφή:

Αυτός ο τελεστής μετατρέπει το macro\_argument σε μια συμβολοσειρά της οποίας η τιμή είναι το όνομα του ορίσματος. Αυτή η αντικατάσταση πραγματοποιείται κατά τη διάρκεια της μακροεντολής επέκτασης, πριν από τη μεταγλώττιση.

**Σημείωση:** λόγω αυτής της δυνατότητας, θα πρέπει να δοθεί προσοχή κατά τη χρήση δηλώσεων χειρισμού αρχείων σε μακροεντολή. Λόγω πιθανής ασάφειας με τις δηλώσεις χειρισμού αρχείων που λαμβάνουν μια παράμετρο "#filename", εάν το αρχείο είναι μία από τις μακροεντολές παραμέτρους, μπορεί να χρειαστεί να τυλίξετε την παράσταση αρχείου σε παρένθεση (π.χ. "#(filename)"), για διαχωρισμό είναι από το σύμβολο #. Διαφορετικά, το αρχείο αρχειοθετείται στη μακροεντολή.

### Παράδειγμα - operator#.bas



```
1. #define SEE(x) Print #x ;" = "; x
2. Dim variable As Integer, another_one As Integer
3. variable=1
4. another_one=2
5. SEE(variable)
6. SEE(another_one)
7.
8. Sleep
9. End
```

## Έξοδος:



```
variable = 1  
another_one = 2
```

## Τελεστής ## (Preprocessor Concatenate)

### Σύνταξη:



text##text

### Περιγραφή:

Αυτός ο τελεστής δημιουργεί μια νέα έκφραση συνδυάζοντας τα κείμενα και στις δύο πλευρές του. Αυτό το κείμενο μπορεί να αναγνωρισθεί από άλλες μακροεντολές και να επεκταθεί περαιτέρω. Μια χρήση, είναι η δημιουργία μιας μακροεντολής που επεκτείνεται σε διαφορετικά ονόματα μακροεντολών, ονόματα μεταβλητών και ονόματα συναρτήσεων ανάλογα με τα ορίσματα που λαμβάνονται.

### Σημείωση: για έκδοση fbc> = 1.08:

Κάντε χρήση μακροεντολών/ορισμού '##\_ ' για να ξεφύγετε από τον χαρακτήρα συνέχισης γραμμής '\_' για να επιτρέψετε τον συνδυασμό πολλών γραμμών διευρυμένου κώδικα μακροεντολής σε μία μόνο δήλωση.

### Παράδειγμα:



```
1. #define Concat(t,n) t##n  
2.  
3. Print concat (12,34)  
4. Dim Concat (hello,world) As Integer  
5. Concat (hello,world)=99  
6. Print helloworld  
7.  
8. Sleep  
9. End
```

## Έξοδος:



```
1234  
99
```

## Τελεστής ! (Escaped String Literal)

Δηλώνει ρητά ότι μια συμβολοσειρά πρέπει να υποβληθεί σε επεξεργασία για ακολουθίες διαφυγής.

## Σύνταξη:



```
!"text"
```

## Περιγραφή:

Αυτός ο τελεστής δηλώνει ρητά ότι η συμβολοσειρά κατά γράμμα που ακολουθεί (τυλιγμένη σε διπλά εισαγωγικά) πρέπει να υποβληθεί σε επεξεργασία για ακολουθίες διαφυγής. Αυτός είναι ένας τελεστής προεπεξεργαστή και μπορεί να χρησιμοποιηθεί μόνο με γράμματα συμβολοσειράς κατά τη μεταγλώττιση.

Η προεπιλεγμένη συμπεριφορά για τα γράμματα συμβολοσειράς είναι ότι δεν υποβάλλονται σε επεξεργασία για ακολουθίες διαφυγής. Η επιλογή Option Escape μπορεί να χρησιμοποιηθεί στη διάλεκτο fbLite για να παρακάμψει αυτήν την προεπιλεγμένη συμπεριφορά προκαλώντας την επεξεργασία όλων των συμβολοσειρών για ακολουθίες διαφυγής.

Χρησιμοποιήστε τον τελεστή \$ Operator (Non-Escaped String Literal) για να δηλώσετε ρητά ότι μια συμβολοσειρά δεν πρέπει να υποβάλλεται σε επεξεργασία για ακολουθίες διαφυγής.

## Παράδειγμα - operator!.bas



```
1. Print "Some escape sequence examples:"
2. Print !"1.\t\tsingle quote (\\') : \'
3. Print !"2.\tdouble quote (\\") : \'"
4. Print !"3.\tbackslash (\\) : \\\
5. Print !"4.\tascii char (\\65): \65"
6.
7. Sleep
8. End
```

### Έξοδος:



Some escape sequence examples:

```
1. single quote (\') : '
2. double quote (\") : "
3. backslash (\\) : \
4. ascii char (\\65): A
```

## Τελεστής \$ (Non-Escaped String Literal)

Δηλώνει ρητά ότι μια συμβολοσειρά δεν πρέπει να υποβάλλεται σε επεξεργασία για ακολουθίες διαφυγής.

### Σύνταξη:



\$"text"

### Περιγραφή:

Αυτός ο τελεστής δηλώνει ρητά ότι η συμβολοσειρά κατά γράμμα που ακολουθεί (τυλιγμένη σε διπλά εισαγωγικά) δεν πρέπει να υποβάλλεται σε επεξεργασία για ακολουθίες διαφυγής. Αυτός είναι ένας τελεστής προεπεξεργαστή και μπορεί να χρησιμοποιηθεί μόνο με γράμματα συμβολοσειράς κατά τη μεταγλώττιση.

Η προεπιλεγμένη συμπεριφορά για τα γράμματα συμβολοσειράς είναι ότι δεν υποβάλλονται σε επεξεργασία για ακολουθίες διαφυγής. Ωστόσο, το Option Escape στη

διάλεκτο `-lang fblite` μπορεί να χρησιμοποιηθεί για να παρακάμψει αυτήν την προεπιλεγμένη συμπεριφορά προκαλώντας την επεξεργασία όλων των συμβολοσειρών για ακολουθίες διαφυγής.

Χρησιμοποιήστε τον τελεστή `!` (Escaped String Literal) για να δηλώσει ρητά ότι μια συμβολοσειρά πρέπει να υποβληθεί σε επεξεργασία για ακολουθίες διαφυγής.

## Παράδειγμα - `operator$.bas`



```
1. #lang "fblite"
2.
3. Print "Default"
4. Print "Backslash : \\"
5. Print !"Backslash !: \\"
6. Print $"Backslash $: \\"
7. Print
8. Option Escape
9. Print "Option Escape"
10. Print "Backslash : \\"
11. Print !"Backslash !: \\"
12. Print $"Backslash $: \\"
13. Print
14.
15. Sleep
16. End
```

## Έξοδος:



```
Default
Backslash : \
Backslash !: \
Backslash $: \

Option Escape
Backslash : \
Backslash !: \
Backslash $: \
```

# Οδηγίες αρχείων (File Directives)

## #include

Δήλωση προεπεξεργαστή που περιλαμβάνει περιεχόμενο άλλου πηγαίου αρχείου.

### Σύνταξη:



```
#include [once] "file"
```

### Περιγραφή:

Το #include εισάγει τον πηγαίο κώδικα από άλλο αρχείο στο σημείο όπου εμφανίζεται η οδηγία #include. Αυτό έχει ως αποτέλεσμα τη μεταγλώττιση του πηγαίου κώδικα από το αρχείο συμπερίληψης σαν να ήταν μέρος του αρχείου προέλευσης που το περιλαμβάνει. Μόλις ο μεταγλωττιστής φτάσει στο τέλος του αρχείου συμπερίληψης, το αρχικό αρχείο προέλευσης συνεχίζει να μεταγλωττίζεται.

Αυτό είναι χρήσιμο για να αφαιρέσετε τον κώδικα από ένα αρχείο και να τον διαχωρίσετε σε περισσότερα αρχεία. Είναι χρήσιμο να έχετε ένα μόνο αρχείο με δηλώσεις σε ένα πρόγραμμα που σχηματίζεται από πολλές ενότητες.

Μπορείτε να συμπεριλάβετε αρχεία σε ένα αρχείο συμπερίληψης, αν και αποφύγετε να συμπεριλάβετε το αρχικό αρχείο στο ίδιο, αυτό δεν θα παράγει έγκυρα αποτελέσματα. Συνήθως, τα αρχεία περιλαμβάνουν μια επέκταση .bi και χρησιμοποιούνται κυρίως για τη δήλωση υπο/λειτουργιών/μεταβλητών μιας βιβλιοθήκης, αλλά οποιοσδήποτε έγκυρος πηγαίος κώδικας μπορεί να υπάρξει σε ένα αρχείο συμπερίληψης.

Ο προσδιοριστής Once λέει στον μεταγλωττιστή να συμπεριλάβει το αρχείο μόνο μία φορά ακόμη και αν περιλαμβάνεται αρκετές φορές από τον πηγαίο κώδικα.

Το \$Include είναι μια εναλλακτική μορφή συμπερίληψης, που υπάρχει μόνο για συμβατότητα με το QuickBASIC.

Συνιστάται η χρήση του #include αντ' αυτού.

Ο μεταγλωττιστής θα μετατρέψει αυτόματα τους χαρακτήρες διαχωριστή διαδρομής ('/' και '\') όπως απαιτείται για τη σωστή φόρτωση του αρχείου. Το όνομα του αρχείου μπορεί να είναι απόλυτη ή σχετική διαδρομή.

Για σχετικές διαδρομές ή όπου δεν υπάρχει καθόλου διαδρομή, το αρχείο συμπερίληψης αναζητείται με την ακόλουθη σειρά:

- Σχετικά με τον κατάλογο του αρχείου προέλευσης
- Σχετικά με τον τρέχοντα κατάλογο εργασίας
- Σε σχέση με τους καταλόγους προσθήκης που καθορίζονται με την επιλογή -i γραμμής εντολών
- Ο φάκελος συμπερίληψης της εγκατάστασης του FreeBASIC (FreeBASIC \ inc, όπου το FreeBASIC είναι ο φάκελος όπου βρίσκεται το εκτελέσιμο fbc)

### Παράδειγμα: header.bi



```
1. ' header.bi file
2. Type FooType
3.     Bar As Byte
4.     Barbeque As Byte
5. End Type
```

### main.bas



```
1. ' main.bas file
2. #include "header.bi"
3.
4. Dim Foo As FooType
5. Foo.Bar = 1
6. Foo.Barbeque = 2
```



## #includ

### Σύνταξη:



#includ "libname"

### Περιγραφή:

Περιλαμβάνει μια βιβλιοθήκη στη διαδικασία σύνδεσης σαν να ορίζει ο χρήστης -l libname στη γραμμή εντολών.

### Παράδειγμα:



```
1. | '' incomplete code snippet, this will include  
2. | '' libmystuff.a in the link process  
3. | #includ "mystuff"
```

## #libpath

Δήλωση προεπεξεργαστή για προσθήκη διαδρομής αναζήτησης για βιβλιοθήκες.

### Σύνταξη:



#libpath "path"

### Περιγραφή:

Προσθέτει μια διαδρομή αναζήτησης βιβλιοθήκης στη λίστα των διαδρομών αναζήτησης του συνδέσμου σαν να είχε καθοριστεί στη γραμμή εντολών με την επιλογή '-p'.

Οι διαδρομές είναι σχετικές με τον κατάλογο εργασίας στον οποίο έγινε επίκληση του fbc και όχι σχετική με τον κατάλογο του αρχείου προέλευσης.

Δεν δημιουργείται σφάλμα εάν η διαδρομή δεν υπάρχει και η μεταγλώττιση και η σύνδεση θα συνεχιστούν.

## Παράδειγμα:



1. | ' search the lib directory for external libraries
2. | #libpath "lib"

## Οδηγίες ελέγχου (Control Directives)

### #pragma

#### Σύνταξη:



- #pragma option [ = value ]
- ή
- #pragma push ( option [, value ] )
- ή
- #pragma pop ( option )

#### Παράμετροι:

Πιθανές τιμές για το option και τις σχετικές τιμές values:

Option	Value	Περιγραφή
msbitfields	0	Χρησιμοποιεί bitfields συμβατά με gcc (προεπιλογή)
	-1	Χρησιμοποιεί bitfields συμβατά με αυτά που χρησιμοποιούνται στους μεταγλωττιστές Microsoft C.
once	n/a	προκαλεί το αρχείο προέλευσης στο οποίο το pragma φαίνεται να συμπεριφέρεται σαν να συμπεριλήφθηκε στο #include once ...
constness	0	προκαλεί το αρχείο προέλευσης στο οποίο το pragma φαίνεται να απενεργοποιεί την προειδοποίηση "CONST qualifier discarded"
	-1	προκαλεί το αρχείο προέλευσης στο

	οποίο το pragma φαίνεται να ενεργοποιεί την προειδοποίηση "CONST qualifier discarded"
--	---

Εάν η τιμή value δεν είναι δεδομένη, ο μεταγλωττιστής υποθέτει το -1 (TRUE).

### Περιγραφή:

Επιτρέπει τη ρύθμιση των επιλογών μεταγλωττιστή μέσα στον πηγαίο κώδικα.

Το Push αποθηκεύει την τρέχουσα τιμή της επιλογής σε μια στοίβα και στη συνέχεια εκχωρεί τη νέα τιμή (ή -1) σε αυτήν. Το Pop επαναφέρει την επιλογή στην προηγούμενη τιμή της και την αφαιρεί από τη στοίβα. Αυτός ο μηχανισμός επιτρέπει την αλλαγή των επιλογών για ένα συγκεκριμένο μέρος του πηγαίου κώδικα, ανεξάρτητα από τη ρύθμιση που χρησιμοποιείται από το περιβάλλον, η οποία είναι ιδιαίτερα χρήσιμη στα αρχεία κεφαλίδας #include.

Το constness pragma προστίθεται για τον έλεγχο του μεταγλωττιστή fbc. Θα αφαιρεθεί στο μέλλον, δεν πρέπει να βασιστείτε σε αυτό.

### Παράδειγμα:



```
1.  '' MSVC-compatible bitfields:
2.  '' save the current setting
3.  '' and then enable them
4.  #pragma push(msbitfields)
5.
6.  '' do something that requires
7.  '' MS-compatible bitfields here
8.  '' restore original setting
9.  #pragma pop(msbitfields)
```

## #lang

Δήλωση προεπεξεργαστή για να ορίσετε τη διάλεκτο του μεταγλωττιστή.

### Σύνταξη:



#lang "lang"

### Παράμετροι:

#### lang

Η διάλεκτος που πρέπει να οριστεί, περικλείεται σε διπλά εισαγωγικά και πρέπει να είναι μία από τις λέξεις **"fb"**, **"fb-lite"**, **"qb"** ή **"deprecated"**.

### Περιγραφή:

Εάν η επιλογή -forcelang δεν δόθηκε στη γραμμή εντολών, το #lang μπορεί να χρησιμοποιηθεί για να ορίσετε τη διάλεκτο για τη μονάδα προέλευσης στην οποία εμφανίζεται. Το πολύ δύο περάσματα θα γίνουν στην ενότητα προέλευσης.

Στο πρώτο πέρασμα, εάν η καθορισμένη διάλεκτος είναι κάτι διαφορετικό από την προεπιλεγμένη διάλεκτο (επιλεγμένη με -lang, ή "fb" από προεπιλογή), ο μεταγλωττιστής θα επαναφέρει τον αναλυτή για ένα άλλο πέρασμα και θα επανεκκινήσει τη μεταγλώττιση στην αρχή της μονάδας προέλευσης.

Εάν αυτή η οδηγία συναντηθεί ξανά στο δεύτερο πέρασμα και η καθορισμένη διάλεκτος δεν ταιριάζει με τη νέα τρέχουσα διάλεκτο, εκδίδεται μια προειδοποίηση και η σύνταξη συνεχίζεται. Εάν αντιμετωπίστηκαν σφάλματα στο πρώτο πέρασμα, ο μεταγλωττιστής δεν θα επιχειρήσει δεύτερο πέρασμα.

Το #lang δεν μπορεί να χρησιμοποιηθεί σε καμία σύνθετη δήλωση, πεδίο εφαρμογής ή υπορουτίνα. Ωστόσο, μπορεί να τοποθετηθεί σε δηλώσεις προεπεξεργαστή επιπέδου λειτουργικής μονάδας ή να χρησιμοποιηθεί σε αρχείο συμπερίληψης.

Προς το παρόν δεν υπάρχει περιορισμός για το πού μπορεί να τοποθετηθεί αυτή η οδηγία σε μια ενότητα πηγής. Στο μέλλον αυτό μπορεί να αλλάξει, επομένως η καλύτερη πρακτική θα ήταν η χρήση αυτής της οδηγίας πριν από την πρώτη δήλωση, ορισμό ή εκτελέσιμη δήλωση στην πηγή.

Αυτή η οδηγία παρακάμπτει την επιλογή `-lang` εάν δόθηκε στη γραμμή εντολών. Ωστόσο, εάν η επιλογή `-forcelang` δόθηκε στη γραμμή εντολών, αυτή η οδηγία δεν θα έχει αποτέλεσμα. Εκδίδεται μια προειδοποίηση, αγνοείται η οδηγία και η σύνταξη θα συνεχιστεί. Αυτό επιτρέπει στον χρήστη να παρακάμψει ρητά τις οδηγίες `#lang`.

### Παράδειγμα:



```
1. | #lang "fblite"
```

### #print

Οδηγία διάγνωσης προεπεξεργαστή.

### Σύνταξη:



```
#print text
```

### Περιγραφή:

Προκαλεί τον μεταγλωττιστή να εξάγει κείμενο στην οθόνη κατά τη μεταγλώττιση.

### Παράδειγμα:



```
1. | #print Now compiling module foo
```

## #error

Οδηγία διάγνωσης προεπεξεργαστή.

### Σύνταξη:



`#error error_text`

### Περιγραφή:

Το `#error` διακόπτει τη μεταγλώττιση για να εμφανίσει σφάλμα κείμενο όταν το εντοπίσει ο μεταγλωττιστής και, στη συνέχεια, η ανάλυση συνεχίζεται.

Αυτή η λέξη-κλειδί πρέπει να περιβάλλεται από ένα `#if <condition> ... #endif`, έτσι ώστε ο μεταγλωττιστής να μπορεί να φτάσει στο `#error` μόνο εάν πληρείται το `<condition>`.

Σε κάθε περίπτωση, η τελική κατάσταση θα είναι "Αποτυχία μεταγλώττισης".

### Παράδειγμα:



```
1. | #define c 1
2. |
3. | #if c = 1
4. |     #error Bad value of c
5. | #endif
```

## #assert

Οδηγία διάγνωσης προεπεξεργαστή.

### Σύνταξη:



`#assert condition`

## Περιγραφή:

Επιβεβαιώνει την αλήθεια μιας έκφρασης υπό όρους κατά τη μεταγλώττιση. Εάν η συνθήκη είναι ψευδής, η μεταγλώττιση θα σταματήσει με ένα σφάλμα.

Αυτή η δήλωση διαφέρει από τη μακροεντολή Assert στο ότι το `#assert` αξιολογείται κατά τον χρόνο μεταγλώττισης και το Assert αξιολογείται κατά την εκτέλεση.

## Παράδειγμα:



1. `Const MIN = 5, MAX = 10`
2. `#assert MAX > MIN`
3. `' cause a compile-time error if MAX <= MIN`

## #line

Οδηγία προεπεξεργαστή για να ορίσετε τον τρέχοντα αριθμό γραμμής και όνομα αρχείου.

## Σύνταξη:



```
#line number [ "name" ]
```

## Παράμετροι:

number

νέος αριθμό γραμμής

name

νέο όνομα αρχείου (προαιρετικό)

## Περιγραφή:

Ενημερώνει τον μεταγλωττιστή για μια αλλαγή στον αριθμό γραμμής και το όνομα αρχείου και ενημερώνει ανάλογα τις τιμές μακροεντολής `__FILE__` και `__LINE__`.

Τόσο τα μηνυμάτα μεταγλώττισηςόσο και τα μηνύματα χρόνου εκτέλεσης επηρεάζονται από αυτήν την οδηγία.

Αυτή η οδηγία επιτρέπει σε άλλα προγράμματα να παράγουν πηγαίο κώδικα για τον μεταγλωττιστή της FreeBASIC και να του επιστρέφουν προειδοποιητικά μηνύματα και/ή μηνύματα σφάλματος που αναφέρονται στην αρχική πηγή που χρησιμοποιείται από το άλλο πρόγραμμα.

## Μεταεντολές (Metacommands)

### \$Dynamic

Μετα-εντολή για να αλλάξετε τον τρόπο κατανομής των πινάκων.

#### Σύνταξη:



```
'$Dynamic  
ή  
Rem $Dynamic
```

#### Περιγραφή:

Το '\$Dynamic είναι μια μετα-εντολή που καθορίζει ότι οι επόμενες δηλώσεις πίνακα έχουν μεταβλητό μήκος, είτε δηλώνονται με σταθερό εύρος είτε όχι. Αυτό παραμένει σε ισχύ για το υπόλοιπο τμήμα που χρησιμοποιεί το "\$ Dynamic και μπορεί να παρακαμφθεί με το "\$ Static. Είναι ισοδύναμο με τη δήλωση Option Dynamic.

#### Παράδειγμα:



```
1. | ' compile with -lang fblite or qb  
2. | #lang "fblite"  
3. |  
4. | '$DYNAMIC  
5. | Dim a(100)  
6. | .....  
7. | ReDim a(200)
```



## \$Static

Μετα-εντολή για να αλλάξετε τον τρόπο κατανομής των πινάκων.

### Σύνταξη:



'\$Static  
ή  
Rem \$Static

### Περιγραφή:

Το '\$ Static είναι μια μετα-εντολή που παρακάμπτει τη συμπεριφορά του \$Dynamic, δηλαδή, οι πίνακες που δηλώνονται είναι σταθερού μήκους. Αυτό παραμένει σε ισχύ για το υπόλοιπο τμήμα που χρησιμοποιεί το "\$Static" και μπορεί να αντικατασταθεί με το \$Dynamic. Είναι ισοδύναμο με τη δήλωση Option Static.

### Παράδειγμα:



```
1. ' compile with -lang fblite or qb
2. #lang "fblite"
3.
4. '$dynamic
5. Dim a(100)'<<this array will be variable-length
   '$static
6. Dim b(100)'<<this array will be fixed-length
```

## \$Lang

Δήλωση μετα-εντολής για να ορίσετε τη διάλεκτο του μεταγλωττιστή.

### Σύνταξη:



'\$lang: "lang"  
ή  
Rem \$lang: "lang"

### Παράμετροι:

"lang"

Η διάλεκτος που πρέπει να οριστεί, περικλείεται σε διπλά εισαγωγικά και πρέπει να είναι μία από τις λέξεις "fb", "fb-lite", "qb" ή "deprecated".

### Παράδειγμα:



1. | '\$lang: "qb"

# **Κεφάλαιο 14ο - Εγγενείς Ορισμοί (Intrinsic Defines)**

Οι εγγενείς ορισμοί ορίζονται από τον μεταγλωττιστή και μπορούν να χρησιμοποιηθούν ως οποιοδήποτε άλλο καθορισμένο σύμβολο. Οι εγγενείς ορισμοί συχνά μεταφέρουν πληροφορίες σχετικά με την κατάσταση του μεταγλωττιστή, είτε γενικά είτε σε συγκεκριμένο σημείο της διαδικασίας μεταγλώττισης. Οι περισσότεροι εγγενείς ορισμοί σχετίζονται με μια τιμή.

Οι εγγενείς ορισμοί κατηγοριοποιούνται ως εξής:

## **Πληροφορίες πλατφόρμας**

Ορισμοί που παρέχουν πληροφορίες για το σύστημα.

## **Πληροφορίες έκδοσης**

Ορισμοί οι οποίοι παρέχουν πληροφορίες σχετικά με την έκδοση μεταγλωττιστή fbc που χρησιμοποιείται.

## **Διακόπτες γραμμής εντολών**

Ορισμοί οι οποίοι παρέχουν πληροφορίες με τους διακόπτες γραμμής εντολών που χρησιμοποιούνται με το fbc.

## **Πληροφορίες Περιβάλλοντος**

Ορισμοί οι οποίοι παρέχουν πληροφορίες σχετικά με το περιβάλλον του λειτουργικού συστήματος.

## **Πληροφορίες σχετικά με τη σύνταξη**

Ορισμοί που παρέχουν πληροφορίες περιβάλλοντος σχετικά με τη διαδικασία σύνταξης.

## **Βασικές μακροεντολές**

Ενσωματωμένες βασικές μακροεντολές.

## **Σταθερές**

Ενσωματωμένες σταθερές.

Για παράδειγμα, εγγενείς ορισμοί πληροφοριών πλατφόρμας:

**\_\_FB\_WIN32\_\_**

Ορίζεται αν γίνεται μεταγλώττιση για Windows.

**\_\_FB\_LINUX\_\_**

Ορίζεται αν γίνεται μεταγλώττιση για Linux.

**\_\_FB\_DOS\_\_**

Ορίζεται αν γίνεται μεταγλώττιση για DOS.

**\_\_FB\_CYGWIN\_\_**

Ορίζεται αν συντάσσεται για το Cygwin.

**\_\_FB\_FREEBSD\_\_**

Ορίζεται αν γίνεται μεταγλώττιση για το FreeBSD.

**\_\_FB\_NETBSD\_\_**

Ορίζεται αν γίνεται μεταγλώττιση για NetBSD.

**\_\_FB\_OPENBSD\_\_**

Ορίζεται αν γίνεται μεταγλώττιση για OpenBSD.

**\_\_FB\_DARWIN\_\_**

Ορίζεται αν συντάσσεται για το Darwin.

**\_\_FB\_XBOX\_\_**

Ορίζεται αν γίνεται μεταγλώττιση για το Xbox.

**\_\_FB\_BIGENDIAN\_\_**

Καθορίζεται εάν γίνεται μεταγλώττιση σε ένα σύστημα με τη χρήση θέσης byte big-endian.

**\_\_FB\_PCOS\_\_**

Ορίζεται εάν γίνεται μεταγλώττιση για ένα κοινό λειτουργικό σύστημα PC (π.χ. DOS, Windows, OS/2).

**\_\_FB\_UNIX\_\_**

Ορίζεται αν γίνεται μεταγλώττιση για λειτουργικό σύστημα παρόμοιο με το Unix.

## **\_\_FB\_64BIT\_\_**

Ορίζεται αν συντάσσεται για στόχο 64bit.

## **\_\_FB\_ARM\_\_**

Ορίζεται αν συντάσσεται για την αρχιτεκτονική ARM.

## **\_\_Fb\_X86\_\_**

Ορίζεται αν συντάσσεται για την αρχιτεκτονική X86 / X86\_64.

### **Παράδειγμα:**



```
1. #ifdef __FB_LINUX__
2.     ' ... instructions only for Linux ...
3.     ' ... #libpath "/usr/X11/lib"
4. #else
5.     ' ... instructions not for Linux ...
6. #endif
```

Αναλυτικά για τις υπόλοιπες κατηγορίες κάθε εγγενούς ορισμού στην σελίδα

<https://www.freebasic.net/wiki/CatPgDddefines>

# Κεφάλαιο 15ο Εξωτερικές βιβλιοθήκες (External Libraries)

Αυτή είναι η λίστα των εξωτερικών βιβλιοθηκών που περιλαμβάνονται αυτήν τη στιγμή στη FreeBASIC.

## Graphical/test-based user interfaces

- [CGUI](#) - Library for making GUIs in a simple way.
- [Curses](#) - Standardized console user interface library.
- [GTK+](#) - Cross-platform Graphical User Interface Library.
- [IUP](#) - Portable toolkit for building graphical user interfaces.
- [wxC](#) - Cross-platform Graphical User Interface Library.
- [Windows API](#) - Windows GUIs and more
- [X11](#) - Windowing system commonly used on Linux systems

## Graphics

- [Allegro](#) - Game programming library.
- [DUGL](#) - Game and graphics library for DOS.
- [caca](#) - A colour ASCII art library.
- [Cairo](#) - 2D graphics library with support for multiple output devices.
- [DISLIN](#) - Library of subroutines and functions that display data graphically.
- [freeglut](#) - A free alternative to [GLUT](#), an OpenGL library for window creation and callback-based input handling
- [FreeImage](#) - Open Source library to support popular graphics image formats.
- [Freetype2](#) - A Free, High-Quality, and Portable Font Engine.
- [GD](#) - Open source code library for the dynamic creation of images by programmers.
- [GIFLIB](#) - Portable tools and library routines for working with GIF images.
- [GLUT](#) - the original (but now inactive) OpenGL Utility Toolkit
- [GLFW](#) - An OpenGL library for creating an OpenGL window and handling input from the user's main loop
- [GRX](#) - 2D graphics library.

[IL \(DevIL\)](#) - A full featured cross-platform image library.  
[japi](#) - Open source free software GUI toolkit using Java's AWT Toolkit.

[jpeglib](#) - Cross-platform library for reading/writing jpeg images.

[JPGalleg](#) - A small add-on for Allegro that adds JPG images handling capabilities to the library.

[libpng](#) - Allows reading and writing PNG images.

[OpenGL](#) - Cross-platform 3D Graphics library.

[PDFlib](#) - Portable library for dynamically generating PDF documents.

[SDL](#) - Cross-platform multimedia library.

[TinyPTC](#) - A small and easy to use framebuffer graphics library.

### Music/Sound, Audio/Video

[BASS](#) - Audio library for use in Windows with a beta for Linux.

[BASSMOD](#) - BASSMOD is a MOD only (XM, IT, S3M, MOD, MTM, UMX) version of BASS

[Flite](#) - Run time speech synthesis engine

[FMOD](#) - Audio library supporting just about any format.

[MediaInfo](#) - Library to read out technical and tag information from many media file formats

[mpg123](#) - MPEG (including MP3) decoder library

[Ogg](#) - Ogg multimedia container format creation/decoder library

[OpenAL](#) - Cross-platform 3D audio API.

[PortAudio](#) - Cross-platform audio I/O library

[sfx](#) - Freebasic sfx library is cross-platform and comes for Windows, Linux, Dos

[sndfile](#) - Library to read/write/convert audio files in various formats

[VLC](#) - Audio/video playback

[Vorbis](#) - Ogg Vorbis audio compression library

### Database

[GDBM](#) - Database functions using extensible hashing, primarily for storing key/data pairs to data files

[MySQL](#) - High-Quality, widely used database engine.

[PostgreSQL](#) - Free software object-relational database management system

[SQLite](#) - Small C library that implements a self-contained, embeddable, zero-configuration SQL database engine.

### Development Helpers

[CUnit](#) - Lightweight system for writing, administering, and running unit tests in C.

[GDSDL](#) - The Generic Data Structures Library is a collection of routines for generic data structures.

[gettext \(includes libintl\)](#) - Internationalization mechanism

[GNU ASpell](#) - Free and Open Source spell checker.

[libbfd](#) - Allows applications to use the same routines to operate on object files whatever the object file format.

### Embeddable Languages

[JNI](#) - Standard programming interface for writing Java native methods and embedding the Java virtual machine into native applications.

[json-c](#) - A JSON implementation in C

[libffi](#) - Foreign function interface and closure library

[libjit](#) - Runtime (just in time) compilation library

[Lua](#) - Lightweight, embeddable scripting engine using the Lua language.

[SpiderMonkey](#) - Embeddable javascript engine.

### Cryptography

[cryptlib](#) - A powerful security toolkit which allows even inexperienced crypto programmers to easily add encryption and authentication services to their software.

[UUID](#) - Universally Unique Identifier generation and parsing library

### Mathematics

[big\\_int](#) - Library for using arbitrarily large integers.

[Chipmunk](#) - 2D rigid body physics library

[GMP](#) - Free library for arbitrary precision arithmetic, operating on signed integers, rational numbers, and floating point numbers.

[GSL](#) - Provides a wide range of mathematical routines such



as random number generators, special functions and least-squares fitting.

[Newton](#) - Integrated solution for real time simulation of physics environments.

[ODE](#) - Open source, high performance library for simulating rigid body dynamics.

## Networking

[cgi-util](#) - Small C library for creating CGI programs for Websites.

[curl](#) - Free and easy-to-use client-side URL transfer library supporting almost every protocol.

[FastCGI](#) - Open extension to CGI that provides high performance without the limitations of server specific APIs.

[ZeroMQ](#) - High-performance asynchronous messaging library

## eXtensible Markup Language (XML)

[Expat](#) - Stream oriented XML parser library with several useful features.

[libxml](#) - De-facto standard library for accessing xml files.

[libxslt](#) - XSLT itself is a an XML language to define transformation for XML.

[Mini-XML](#) - Small XML parsing library that you can use to read XML and XML-like data files in your application.

## Regular Expressions

[PCRE](#) - Regular expression pattern matching using the same syntax as Perl.

[TRE](#) - Lightweight, robust, and efficient POSIX compliant regexp matching library.

## Compression

[bzip2](#) - For reading/writing .bz2 files or in-memory (de)compression using the bzip2 algorithms

[libzip](#) - Easy-to-use library for creating and unpacking .zip files

[liblzma](#) - Strong LZMA-based compression library used for .lzma and .xz file formats

[LZO](#) - Offers fast compression and very fast decompression.

[QuickLZ](#) - Very fast Compression Library

[zlib](#) - Lossless data compression library unencumbered by patents.

### System APIs

[C Runtime Library](#)

[DOS API](#)

[disp helper](#) - Helper library to use COM objects from plain C

[GLib](#) - GNOME's universal cross-platform software utility library

[Windows API](#)

[X11](#) - Windowing system commonly used on Linux systems

# Κεφάλαιο 16ο Εφαρμογές της FreeBASIC

Σε αυτό το κεφάλαιο θα δούμε μερικά παραδείγματα - εφαρμογές που έχουν αναπτυχθεί με την γλώσσα FreeBASIC.

## Εφαρμογή 1: Μετατροπή Θερμοκρασίας σε Fahrenheit/Celsius.

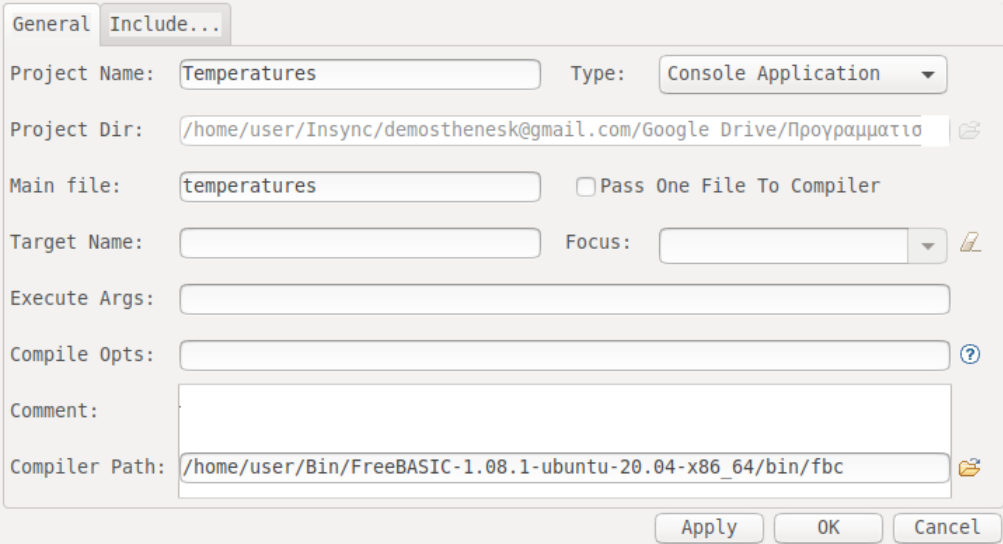
Κατά την εκκίνηση της εφαρμογής θα ζητείται από τον χρήστη να ορίσει ποια μετατροπή θέλει να κάνει.

Από Fahrenheit σε Celsius ή το αντίθετο.

Αφού κάνει την επιλογή ο χρήστης θα του ζητείται να εισάγει θερμοκρασία προς μετατροπή.

Τέλος θα του εμφανίζει το αποτέλεσμα της επιλογής του.

Στο IDE PoseidonFB δημιουργήστε ένα νέο Project και συμπληρώστε τις τιμές των ιδιοτήτων του έργου.



The image shows a dialog box for configuring a project in the PoseidonFB IDE. The dialog has two tabs: 'General' (selected) and 'Include...'. The fields are as follows:

- Project Name: Temperatures
- Type: Console Application
- Project Dir: /home/user/Insync/demosthenesk@gmail.com/Google Drive/Προγραμματισ
- Main file: temperatures
- Pass One File To Compiler:
- Target Name: (empty)
- Focus: (empty)
- Execute Args: (empty)
- Compile Opts: (empty)
- Comment: (empty)
- Compiler Path: /home/user/Bin/FreeBASIC-1.08.1-ubuntu-20.04-x86\_64/bin/fbc

Buttons at the bottom: Apply, OK, Cancel.

Εσείς εισαγάγετε τις δικές σας τιμές ανάλογα με την εγκατάσταση της FreeBASIC στο σύστημα σας και το φάκελο που θέλετε να υπάρχει το έργο.

Δημιουργήστε ένα αρχείο bas και κάνοντας δεξί κλικ πάνω του ορίστε το ως Main Module (Set As Main Module).

```

1  =Function FarToCel(F As Double) As Double
2      Return (5/9*(F-32))
3  End Function
4
5  =Function CelToFar (C As Double) As Double
6      Return (C*9/5+32)
7  End Function
8
9  Dim inputTemperature As Double = 0.0
10 Dim iChoice As Integer = 0
11
12
13 =Do
14     Print "Choose an option:"
15     Print "1 For Fahrenheit To Celsius"
16     Print "2 For Celsius To Fahrenheit"
17     Print "3 For exit..."
18     Input iChoice
19
20     Select Case iChoice
21     Case 1
22         Input "Input temperature in Fahrenheit: ", inputTemperature
23         Print "C= " ; FarToCel(inputTemperature)
24     Case 2
25         Input "Input temperature in Celsius: ", inputTemperature
26         Print "F= " ; CelToFar(inputTemperature)
27     Case 3
28         Exit Do
29     End Select
30 Loop

```

## Παράδειγμα - Project1



```

1.  Function FarToCel(F As Double) As Double
2.      Return (5/9*(F-32))
3.  End Function
4.
5.  Function CelToFar (C As Double) As Double
6.      Return (C*9/5+32)
7.  End Function
8.
9.  Dim inputTemperature As Double = 0.0
10. Dim iChoice As Integer = 0
11.
12.
13. Do
14.     Print "Choose an option:"
15.     Print "1 For Fahrenheit To Celsius"
16.     Print "2 For Celsius To Fahrenheit"
17.     Print "3 For exit..."
18.     Input iChoice
19.
20.     Select Case iChoice
21.     Case 1

```

```

22.     Input "Input temperature in Fahrenheit: ",
inputTemperature
23.     Print "C= " ; FarToCel(inputTemperature)
24.     Case 2
25.     Input "Input temperature in Celsius: ",
inputTemperature
26.     Print "F= " ; CelToFar(inputTemperature)
27.     Case 3
28.     Exit Do
29.     End Select
30. Loop

```

## Έξοδος:



```

Choose an option:
1 For Fahrenheit To Celsius
2 For Celsius To Fahrenheit
3 For exit...
? 1
Input temperature in Fahrenheit: 19
C= -7.222222222222223
Choose an option:
1 For Fahrenheit To Celsius
2 For Celsius To Fahrenheit
3 For exit...
? 2
Input temperature in Celsius: 10
F= 50
Choose an option:
1 For Fahrenheit To Celsius
2 For Celsius To Fahrenheit
3 For exit...
? 3

```

## Ανάλυση:

Στις γραμμές 1-7 έχουμε δύο συναρτήσεις μετατροπής θερμοκρασιών από Φαρενάιτ σε Κελσίου και το αντίστροφο. Στις γραμμές 9-10 δηλώνουμε δύο μεταβλητές. Η πρώτη η `inputTemperature` θα λαμβάνει την θερμοκρασία προς μετατροπή από τον χρήστη και η δεύτερη η `iChoice` θα

λαμβάνει την επιλογή ποια συνάρτηση θα τρέξει το πρόγραμμα για την μετατροπή της θερμοκρασίας. Στις γραμμές 13-17 βρίσκεται η διεπαφή με τον χρήστη εντός ενός βρόγχου Do...Loop. Προτρέπουμε τον χρήστη να διαλέξει συνάρτηση μετατροπής θερμοκρασίας και εκχωρεί την επιλογή του στην μεταβλητή iChoice. Στις γραμμές 19-28 υπάρχει ένα Select Case όπου ανάλογα την τιμή της μεταβλητής iChoice καλείτε και η ανάλογη συνάρτηση μετατροπής θερμοκρασίας. Τέλος υπάρχει και η επιλογή εξόδου από τον βρόγχο Do...Loop στην γραμμή 27.

## **Εφαρμογή 2 - Αλγόριθμος Ταξινόμησης Φυσαλίδας (Bubble Sort)**

Η ταξινόμηση φυσαλίδων λειτουργεί περνώντας διαδοχικά πάνω από μια λίστα, συγκρίνοντας κάθε τιμή με αυτήν αμέσως μετά. Εάν η πρώτη τιμή είναι μεγαλύτερη από τη δεύτερη, οι θέσεις τους αλλάζουν. Πάνω από μια σειρά από περάσματα, το πολύ ίσα με τον αριθμό των στοιχείων στη λίστα, όλες οι τιμές μετατοπίζονται στις σωστές θέσεις τους (οι μεγάλες τιμές «φουσκώνουν» γρήγορα προς το τέλος, σπρώχνοντας τις άλλες προς τα κάτω). Επειδή κάθε πέραςμα βρίσκει το μέγιστο στοιχείο και το βάζει στο τέλος, το τμήμα της λίστας που θα ταξινομηθεί μπορεί να μειωθεί σε κάθε πέραςμα. Μια μεταβλητή boolean χρησιμοποιείται για να παρακολουθεί αν έχουν γίνει αλλαγές στο τρέχον πέραςμα. όταν ένα πέραςμα ολοκληρωθεί χωρίς να αλλάξει τίποτα, ο αλγόριθμος βγαίνει στην έξοδο.

## Παράδειγμα - Project2



```
1. 'bubble sort
2. Sub BubbleSort(array1() As Integer)
3.     ' sort from lower bound to the higher bound
4.     Dim As Integer lb = LBound(array1)
5.     Dim As Integer ub = UBound(array1)
6.     Dim As Integer done, i
7.
8.     Do
9.         done = 0
10.        For i = lb To ub -1
11.            ' replace "<" with ">" for downwards sort
12.            If array1(i) > array1(i +1) Then
13.                Swap array1(i), array1(i +1)
14.                done = 1
15.            End If
16.        Next
17.    Loop Until done = 0
18. End Sub
19.
20. 'print array
21. Sub printArray(array2() As Integer)
22.     For i As Integer = 1 To 10
23.         Print ; array2(i)
24.     Next
25. End Sub
26.
27. 'declare an array
28. Dim array(1 To 10) As Integer
29.
30. 'Fill array with random integers
31. For i As Integer = 1 To 10
32.     Randomize
33.     'get a random integer from 1 to 10
34.     array(i) = Int(Rnd * 10) + 1
35. Next
36.
37.
38.
39. Print "Array before sort"
40. printArray(array())
41.
42. BubbleSort(array())
43.
44.
45. Print "Array after sort"
46. printArray(array())
```

Έξοδος:



```
Array before sort
10
7
5
5
4
3
4
5
3
5
Array after sort
3
3
4
4
5
5
5
5
7
10
```

## Εφαρμογή 3 - Αλγόριθμος Ταξινόμησης με Εισαγωγή (Insertion Sort)

Ένας αλγόριθμος ταξινόμησης που μετακινεί στοιχεία ένα προς ένα στη σωστή θέση. Ο αλγόριθμος αποτελείται από την εισαγωγή ενός στοιχείου κάθε φορά στο τμήμα που έχει προηγουμένως ταξινομηθεί, μετακινώντας τα στοιχεία υψηλότερης κατάταξης προς τα πάνω, όπως απαιτείται. Για να ξεκινήσετε, το πρώτο (ή το μικρότερο, ή οποιοδήποτε αυθαίρετο) στοιχείο του μη ταξινομημένου πίνακα θεωρείται το ταξινομημένο τμήμα.



## Παράδειγμα - Project3



```
1. ' insertion sort
2. Sub InsertionSort( arr() As Long )
3.
4.     ' sort from lower bound to the higher bound
5.     ' array's can have subscript range from -
6.     ' 2147483648 to +2147483647
7.
8.     Dim As Long lb = LBound(arr)
9.     Dim As Long i, j, value
10.
11.     For i = lb +1 To UBound(arr)
12.
13.         value = arr(i)
14.         j = i -1
15.         While j >= lb And arr(j) > value
16.             arr(j +1) = arr(j)
17.             j = j -1
18.         Wend
19.
20.         arr(j +1) = value
21.     Next
22. End Sub
23.
24.
25. 'print array
26. Sub printArray(array2() As Long)
27.     For i As Integer = 1 To 10
28.         Print ; array2(i)
29.     Next
30. End Sub
31.
32.
33. 'declare an array
34. Dim array(1 To 10) As Long
35.
36. 'Fill array with random integers
37. For i As Integer = 1 To 10
38.     Randomize
39.     'get a random integer from 1 to 10
40.     array(i) = Int(Rnd * 10) + 1
41. Next
42.
43.
44. Print "Array before sort"
45. printArray(array())
46.
47. InsertionSort(array())
```

```
48. |
49. | Print "Array after sort"
50. | printArray(array())
```

**Έξοδος:**



```
Array before sort
10
5
4
5
2
4
4
6
4
1
Array after sort
1
2
4
4
4
4
5
5
6
10
```

## **Εφαρμογή 4 - Αλγόριθμος Ταξινόμησης με Επιλογή (Selection sort)**

Βρίσκει πρώτα το μικρότερο στοιχείο στον πίνακα και το ανταλλάζει το με το στοιχείο στην πρώτη θέση, στη συνέχεια βρίσκει το δεύτερο μικρότερο στοιχείο και το ανταλλάζει το με το στοιχείο στη δεύτερη θέση και συνεχίζει με αυτόν τον τρόπο μέχρι να ταξινομηθεί ολόκληρος ο πίνακας.

## Παράδειγμα - Project4



```
1. 'selection sort
2. Sub SelectionSort(arr() As Long)
3.
4.     ' sort from lower bound to the higher bound
5.     ' array's can have subscript range from -
6.     ' 2147483648 to +2147483647
7.
8.     Dim As Long i, j, x
9.     Dim As Long lb = LBound(arr)
10.    Dim As Long ub = UBound(arr)
11.
12.
13.    For i = lb To ub -1
14.        x = i
15.        For j = i +1 To ub
16.            If arr(j) < arr(x) Then x = j
17.        Next
18.        If x <> i Then
19.            Swap arr(i), arr(x)
20.        End If
21.    Next
22. End Sub
23.
24. 'print array
25. Sub printArray(array2() As Long)
26.     For i As Integer = 1 To 10
27.         Print ; array2(i)
28.     Next
29. End Sub
30.
31. 'declare an array
32. Dim array(1 To 10) As Long
33.
34. 'Fill array with random integers
35. For i As Integer = 1 To 10
36.     Randomize
37.     'get a random integer from 1 to 10
38.     array(i) = Int(Rnd * 10) + 1
39. Next
40.
41.
42. Print "Array before sort"
43. printArray(array())
44.
45. SelectionSort(array())
46.
47. Print "Array after sort"
```

48. | `printArray(array())`

**Έξοδος:**



```
Array before sort
```

```
3
```

```
9
```

```
6
```

```
10
```

```
5
```

```
10
```

```
7
```

```
6
```

```
2
```

```
6
```

```
Array after sort
```

```
2
```

```
3
```

```
5
```

```
6
```

```
6
```

```
6
```

```
7
```

```
9
```

```
10
```

```
10
```

## Εφαρμογή 5 - Αλγόριθμος Γρήγορης Ταξινόμησης (Quick Sort).

Ταξινόμηση στοιχείων πίνακα (ή λίστας) χρησιμοποιώντας τον αλγόριθμο quicksort.

Τα στοιχεία πρέπει να έχουν αυστηρά ασθενή σειρά και ο δείκτης του πίνακα μπορεί να είναι οποιοδήποτε διακριτού τύπου.

Για γλώσσες όπου αυτό δεν είναι δυνατό, ταξινομεί μια σειρά από ακέραιους αριθμούς.

Το Quicksort, γνωστό και ως ταξινόμηση ανταλλαγής διαμερισμάτων, χρησιμοποιεί αυτά τα βήματα:

Επιλέξτε οποιοδήποτε στοιχείο του πίνακα για να είναι ο στοιχείο περιστροφής.

Χωρίζει όλα τα άλλα στοιχεία (εκτός από το στοιχείο περιστροφής) σε δύο διαμερίσματα.

Όλα τα στοιχεία μικρότερα από το στοιχείο περιστροφής πρέπει να βρίσκονται στο πρώτο διαμέρισμα.

Όλα τα στοιχεία μεγαλύτερα από το στοιχείο περιστροφής πρέπει να βρίσκονται στο δεύτερο διαμέρισμα.

Χρησιμοποιήστε την αναδρομή για να ταξινομήσετε και τα δύο διαμερίσματα.

Ενώνει το πρώτο ταξινομημένο διαμέρισμα, το στοιχείο περιστροφής και το δεύτερο ταξινομημένο διαμέρισμα.

Το καλύτερο στοιχείο περιστροφής δημιουργεί χωρίσματα ίσου μήκους (ή μήκη που διαφέρουν κατά 1).

Το χειρότερο στοιχείο περιστροφής δημιουργεί ένα κενό διαμέρισμα (για παράδειγμα, εάν το στοιχείο περιστροφής είναι το πρώτο ή το τελευταίο στοιχείο ενός ταξινομημένου πίνακα).

## Παράδειγμα - Project5



```
1. ' quick sort
2. Sub QuickSort(qs() As Long, l As Long, r As Long)
3.
4.     Dim As ULong size = r - l + 1
5.     If size < 2 Then Exit Sub
6.
7.     Dim As Long i = l, j = r
8.     Dim As Long pivot = qs(l + size \ 2)
9.
10.
11.     Do
12.         While qs(i) < pivot
13.             i += 1
14.         Wend
15.         While pivot < qs(j)
16.             j -= 1
17.         Wend
18.         If i <= j Then
19.             Swap qs(i), qs(j)
20.             i += 1
21.             j -= 1
22.         End If
23.     Loop Until i > j
24.
25.     If l < j Then quicksort(qs(), l, j)
26.     If i < r Then quicksort(qs(), i, r)
27. End Sub
28.
29. 'print array
30. Sub printArray(array2() As Long)
31.     For i As Integer = 1 To 10
32.         Print ; array2(i)
33.     Next
34. End Sub
35.
36. 'declare an array
37. Dim array(1 To 10) As Long
38.
39. 'Fill array with random integers
40. For i As Integer = 1 To 10
41.     Randomize
42.     'get a random integer from 1 to 10
43.     array(i) = Int(Rnd * 10) + 1
44. Next
45.
46. Print "Array before sort"
```

```
48. | printArray(array())
49. |
50. | QuickSort(array(), LBound(array), UBound(array))
51. |
52. | Print "Array after sort"
53. | printArray(array())
```

## Έξοδος:



Array before sort

```
6
9
7
3
1
9
3
2
8
6
```

Array after sort

```
1
2
3
3
6
6
7
8
9
9
```

## Εφαρμογή 6 - Αλγόριθμος Ταξινόμησης με Συγχώνευση (Merge Sort)

Η βασική ιδέα είναι να χωρίσει τη συλλογή σε μικρότερες ομάδες, μειώνοντάς τη στο μισό, έως ότου οι ομάδες έχουν μόνο ένα στοιχείο ή κανένα στοιχείο (οι οποίες είναι και οι δύο πλήρως ταξινομημένες ομάδες).

Στη συνέχεια, συγχωνεύει ξανά τις ομάδες έτσι ώστε τα στοιχεία τους να είναι σε τάξη.

Αυτός είναι ο τρόπος με τον οποίο ο αλγόριθμος αποκτά τη περιγραφή διαίρει και βασίλευε.

### Παράδειγμα - Project6



```
1. Sub copyArray(a() As Integer, iBegin As Integer,
2.   iEnd As Integer, b() As Integer)
3.     Redim b(iBegin To iEnd - 1) As Integer
4.     For k As Integer = iBegin To iEnd - 1
5.       b(k) = a(k)
6.     Next
7. End Sub
8.
9. ' Left source half is a(iBegin To iMiddle-1).
10. ' Right source half is a(iMiddle To iEnd-1).
11. ' Result is b(iBegin To iEnd-1).
12. Sub topDownMerge(a() As Integer, iBegin As
13.   Integer, iMiddle As Integer, iEnd As Integer, b()
14.   As Integer)
15.     Dim i As Integer = iBegin
16.     Dim j As Integer = iMiddle
17.
18. ' While there are elements in the left or right
19. ' runs...
20. For k As Integer = iBegin To iEnd - 1
21.   ' If left run head exists and is <= existing
22.   ' right run head.
23.   If i < iMiddle AndAlso (j >= iEnd OrElse a(i)
24.   <= a(j)) Then
25.     b(k) = a(i)
26.     i += 1
27.   Else
28.     b(k) = a(j)
29.     j += 1
30.   End If
```



```

26.     Next
27. End Sub
28.
29. ' Sort the given run of array a() using array b()
30. ' as a source.
31. ' iBegin is inclusive; iEnd
32. ' is exclusive (a(iEnd) is not in the set).
33. Sub topDownSplitMerge(b() As Integer, iBegin As
34. Integer, iEnd As Integer, a() As Integer)
35.     If (iEnd - iBegin) < 2 Then Return '' If run
36. size = 1, consider it sorted
37.     ' split the run longer than 1 item into halves
38.     Dim iMiddle As Integer = (iEnd + iBegin) \ 2
39.     '' iMiddle = mid point
40.     ' recursively sort both runs from array a()
41. into b()
42.     topDownSplitMerge(a(), iBegin, iMiddle, b())
43.     '' sort the left run
44.     topDownSplitMerge(a(), iMiddle, iEnd, b())
45.     '' sort the right run
46.     ' merge the resulting runs from array b() into
47. a()
48.     topDownMerge(b(), iBegin, iMiddle, iEnd, a())
49. End Sub
50.
51. ' Array a() has the items to sort;
52. ' array b() is a work array (empty initially).
53. Sub topDownMergeSort(a() As Integer, b() As
54. Integer, n As Integer)
55.     copyArray(a(), 0, n, b()) '' duplicate array
56. a() into b()
57.     topDownSplitMerge(b(), 0, n, a()) '' sort data
58. from b() into a()
59. End Sub
60.
61. Sub printArray(a() As Integer)
62.     For i As Integer = LBound(a) To UBound(a)
63.         Print Using "####"; a(i);
64.     Next
65.     Print
66. End Sub
67.
68. Dim a(0 To 9) As Integer = {4, 65, 2, -31, 0, 99,
69. 2, 83, 782, 1}
70.
71. Dim b() As Integer
72. Print "Unsorted : ";

```

```

66. | printArray(a())
67. | topDownMergeSort a(), b(), 10
68. | Print "Sorted : ";
69. | printArray(a())

```

## Έξοδος:



```

Unsorted : 4 65 2 -31 0 99 2 83 782 1
Sorted : -31 0 1 2 2 4 65 83 99 782

```

Για περισσότερους αλγόριθμους ταξινόμησης δείτε στο [https://rosettacode.org/wiki/Category:Sorting\\_Algorithms](https://rosettacode.org/wiki/Category:Sorting_Algorithms)

## Εφαρμογή 7 - Μάντεψε τον αριθμό

Το πρόγραμμα διαλέγει έναν αριθμό από το 1 έως το 10. Ο χρήστης καλείται να μαντέψει τον αριθμό.

### Παράδειγμα - Project7



```

1. | Randomize
2. | Dim n As Integer = Int(Rnd * 10) + 1
3. | Dim guess As Integer
4. |
5. | Print "Guess which number I've chosen in the
   | range 1 to 10"
6. | Print
7. |
8. | Do
9. |   Input " Your guess : "; guess
10. |
11. |   If n = guess Then
12. |     Print "Well guessed!"
13. |   End
14. | End If
15. |
16. |   If n < guess Then
17. |     Print "The number is smaller..."
18. |   End If
19. |
20. |

```

```
21. |   If n > guess Then
22. |       Print "The number is bigger..."
23. |   End If
24. | Loop
```

## Έξοδος:



Guess which number I've chosen in the range 1 to 10

```
Your guess : ? 5
The number is bigger...
Your guess : ? 7
The number is bigger...
Your guess : ? 9
The number is bigger...
Your guess : ? 10
Well guessed!
```

# Εφαρμογή 8 - Διαχείριση βάσης δεδομένων sqlite3

## Παράδειγμα - Project8



```
1. #include once "sqlite3.bi"
2. ' written by ike 2014 -
3. function callback cdecl (byval NotUsed as any
   ptr, byval argc as integer, byval argv as zstring
   ptr ptr, byval colName as zstring ptr ptr) as
   integer
4.   dim as integer i
5.   dim as string text
6.   ? "ARGC:", argc
7.     for i = 0 to argc - 1
8.       text = "NULL"
9.       if( argv[i] <> 0 ) then
10.        if *argv[i] <> 0 then
11.         text = *argv[i]
12.        end if
13.       end if
14.       print *colName[i], " = "; text; ""
15.     next i
16.     print
17.     function = 0
18.   end function
19.
20.   dim as sqlite3 ptr db
21.   ' If an error occurs, this pointer
22.   ' will point a the created error message.
23.   dim as zstring ptr errMsg
24.
25.   dim as integer rc = sqlite3_open("test.db", @db)
26.   ' The connection to the test.db database is
27.   ' created.
28.
29.   if (rc <> SQLITE_OK) then
30.     ? "Cannot open database:", sqlite3_errmsg(db)
31.     sqlite3_close(db)
32.     end 1
33.   end if
34.
35.   dim as string cmd
36.
37.   ' These SQL statements create a Cars table and
38.   ' fill it with data. The statements must be
39.
```

```

40. ' separated by semicolons.
41. cmd ="DROP TABLE IF EXISTS Cars; "
42. cmd &="CREATE TABLE Cars(Id INT, Name TEXT, Price
    INT); "
43. cmd &="INSERT INTO Cars VALUES(1, 'Zastava',
    7642); "
44. cmd &="INSERT INTO Cars VALUES(2, 'Yugo', 3127);
    "
45. cmd &="INSERT INTO Cars VALUES(3, 'Skoda', 4000);
    "
46. cmd &="INSERT INTO Cars VALUES(4, 'Volvo', 9000);
    "
47. cmd &="INSERT INTO Cars VALUES(5, 'Cadillac',
    5000); "
48. cmd &="INSERT INTO Cars VALUES(6, 'Jeep', 1000);
    "
49. cmd &="INSERT INTO Cars VALUES(7, 'GAZ', 1400); "
50. cmd &="INSERT INTO Cars VALUES(8, 'VAZ', 1600);"
51.
52. dim as zstring ptr sql1 = strptr(cmd)
53.
54. rc = sqlite3_exec(db, sql1, 0, 0, @errmsg)
55.
56. if (rc <> SQLITE_OK ) then
57.     ? "SQL error: ", errmsg
58.     sqlite3_free(errmsg)
59.     sqlite3_close(db)
60. end if
61.
62. ' read db
63. dim as zstring ptr sql2 = @"SELECT * FROM Cars;"
64.
65.
66. rc = sqlite3_exec (db, sql2, @callback, 0,
67. @errmsg)
68.
69. if (rc <> SQLITE_OK ) then
70.     ? "Failed to select data SQL error: ", errmsg
71.     sqlite3_free(errmsg)
72.     sqlite3_close(db)
73.     end 1
74. end if
75.
76. sqlite3_close(db)
77.
78. ? "done"
79. Sleep
80. End

```

## Έξοδος:



```
ARGC:      3
Id          = '1'
Name       = 'Zastava'
Price     = '7642'
```

```
ARGC:      3
Id          = '2'
Name       = 'Yugo'
Price     = '3127'
```

```
ARGC:      3
Id          = '3'
Name       = 'Skoda'
Price     = '4000'
```

```
ARGC:      3
Id          = '4'
Name       = 'Volvo'
Price     = '9000'
```

```
ARGC:      3
Id          = '5'
Name       = 'Cadillac'
Price     = '5000'
```

```
ARGC:      3
Id          = '6'
Name       = 'Jeep'
Price     = '1000'
```

```
ARGC:      3
Id          = '7'
Name       = 'GAZ'
Price     = '1400'
```

```
ARGC:      3
```

```
Id      = '8'  
Name    = 'VAZ'  
Price   = '1600'  
  
done
```

Για περισσότερα παραδείγματα δείτε τον φάκελο **examples** που βρίσκεται στον φάκελο της FreeBASIC που αποσυμπιέσατε.

## Βιβλιογραφία

1. <https://www.freebasic.net/forum/>
2. <https://www.freebasic.net/wiki/DocToc>
3. A Beginner's Guide to FreeBasic by Richard D. Clark and Ebben Feagan
4. [https://rosettacode.org/wiki/Rosetta\\_Code](https://rosettacode.org/wiki/Rosetta_Code)
5. <https://en.wikipedia.org/>
6. Mandeson Σύγχρονο Αγγλοελληνικό, Ελληνοαγγλικό λεξικό πληροφορικής

## Παράρτημα

### Creative Commons Αναφορά 4.0

Ασκώντας τα Αδειοδοτούμενα Δικαιώματα (τα οποία ορίζονται παρακάτω), αποδέχεσθε και συμφωνείτε να δεσμευτείτε από τους όρους και τις προϋποθέσεις αυτής της Διεθνούς Δημόσιας Άδειας (Δημόσια Άδεια) Creative Commons Αναφορά 4.0 (Creative Commons Attribution 4.0 International Public License). Στο βαθμό που αυτή η άδεια μπορεί να ερμηνευτεί ως σύμβαση, τα Αδειοδοτούμενα Δικαιώματα παραχωρούνται σε Εσάς εν είδη ανταλλάγματος ("consideration") λαμβάνοντας υπόψη την εκ μέρους Σας αποδοχή αυτών των όρων και των προϋποθέσεων και ο/η Χορηγών/-ούσα την Άδεια (Αδειοδότης/-ούσα) Σας παραχωρεί τα δικαιώματα εν είδη ανταλλάγματος ("consideration") λαμβάνοντας υπόψη τις ωφέλειες που ο/η Χορηγών/-ούσα την Άδεια (Αδειοδότης/-ούσα) αποκομίζει με το να διαθέσει το Αντικείμενο της Αδειοδότησης υπό αυτούς τους όρους και προϋποθέσεις.

#### Άρθρο 1. Ορισμοί.

**α. Υλικό που Προέρχεται από Προσαρμογή, Τροποποίηση ή Διασκευή (Τροποποίηση)** σημαίνει έργο ή άλλο αντικείμενο υπό την προστασία Δικαιωμάτων Πνευματικής Ιδιοκτησίας και Συγγενικών ή Παρομοίων Δικαιωμάτων το οποίο είτε προέρχεται από είτε είναι βασισμένο στο Αντικείμενο Αδειοδότησης και αποτελεί μετάφραση, διασκευή, τροποποίηση, μετατροπή ή με οποιαδήποτε άλλο τρόπο



τροποποίηση του Αντικειμένου Αδειοδότησης που απαιτεί συγκατάθεση του/της Χορηγούντα/-ούσας την Άδεια (Αδειοδότη/-ούσας) βάσει των Δικαιωμάτων Πνευματικής Ιδιοκτησίας και Συγγενικών ή Παρόμοιων Δικαιωμάτων που απολαμβάνει. Για τους σκοπούς της παρούσας Δημόσιας Άδειας, στις περιπτώσεις όπου το Αντικείμενο της Αδειοδότησης είναι μουσικό έργο, ερμηνεία ή εγγραφή ήχου (ηχογράφηση), ο συγχρονισμός του Αντικειμένου της Αδειοδότησης με μια κινούμενη εικόνα (συγχρονισμός) θα θεωρείται Υλικό που Προέρχεται από Προσαρμογή, Τροποποίηση ή Διασκευή (Τροποποίηση).

**β. Άδεια Διενεργούντος/-ούσας την Προσαρμογή** σημαίνει την άδεια που Εσείς εφαρμόζετε στα Δικαιώματα Πνευματικής Ιδιοκτησίας και σε Συγγενικά ή Παρόμοια Δικαιώματα που απολαμβάνετε επί των συμβολών Σας στο Υλικό που Προέρχεται από Προσαρμογή, Τροποποίηση ή Διασκευή (Τροποποίηση) σύμφωνα με τους όρους και τις προϋποθέσεις της παρούσας Δημόσιας Άδειας.

**γ. Δικαιώματα Πνευματικής Ιδιοκτησίας και Συγγενικά ή Παρόμοια Δικαιώματα** σημαίνει δικαιώματα πνευματικής ιδιοκτησίας και/ή δικαιώματα που προσομοιάζουν με τα δικαιώματα πνευματικής ιδιοκτησίας και τα οποία περιλαμβάνουν, μεταξύ άλλων, τα δικαιώματα των ερμηνευτών/-τριών ή εκτελεστών καλλιτεχνών, των ραδιοτηλεοπτικών οργανισμών, των παραγωγών ηχογραφήματων (των παραγωγών υλικών φορέων ήχου), των παραγωγών οπτικοακουστικών έργων (των παραγωγών υλικών φορέων ήχου και εικόνας), των εκδοτών/-τριών εντύπων καθώς και το Δικαίωμα Ειδικής Φύσης Κατασκευαστή/-τριας Βάσης Δεδομένων, ανεξαρτήτως του τρόπου με τον οποίο τα δικαιώματα αυτά ονοματίζονται ή κατηγοριοποιούνται στο νόμο. Για τους σκοπούς της παρούσας Δημόσιας Άδειας, τα δικαιώματα που αναφέρονται στο τμήμα [2\(b\)\(1\)-\(2\)](#) αυτής δεν αποτελούν Δικαιώματα Πνευματικής Ιδιοκτησίας και Συγγενικά ή Παρόμοια Δικαιώματα.

**δ. Αποτελεσματικά Τεχνολογικά Μέτρα** σημαίνει τα μέτρα αυτά, τα οποία εν απουσία κατάλληλης εξουσίας, δεν είναι δυνατόν να παραβιάζονται βάσει του νομικού πλαισίου που δημιουργήθηκε σε

συμμόρφωση με το Άρθρο 11 της Συνθήκης Πνευματικής Ιδιοκτησίας του ΠΟΔΙ, η οποία υιοθετήθηκε την 20η Δεκεμβρίου 1996, ή/και βάσει παρόμοιων διεθνών συνθηκών.

**ε. Εξαιρέσεις και Περιορισμοί** σημαίνει εξαιρέσεις εύλογης χρήσης (fair use/ fair dealing) ή/ και κάθε άλλη εξαίρεση ή περιορισμό Δικαιώματος Πνευματικής Ιδιοκτησίας και Συγγενικού ή Παρόμοιου Δικαιώματος που εφαρμόζονται στη χρήση που Εσείς κάνετε στο Αντικείμενο Αδειοδότησης.

**φ. Αντικείμενο Αδειοδότησης** σημαίνει το καλλιτεχνικό ή έργο λόγου (φιλολογικό), βάση δεδομένων ή οποιοδήποτε άλλο υλικό το οποίο ο/η Χορηγών/-ούσα την Άδεια (Αδειοδότης/-ούσα) διαθέτει με την παρούσα Δημόσια Άδεια.

**γ. Αδειοδοτούμενα Δικαιώματα** σημαίνει οι εξουσίες που παραχωρούνται σε Εσάς σύμφωνα με τους όρους και τις προϋποθέσεις της παρούσας Δημόσιας Άδειας, οι οποίες περιορίζονται μόνο στο Δικαίωμα Πνευματικής Ιδιοκτησίας και τα Συγγενικά ή Παρόμοια Δικαιώματα, τα οποία αφορούν στη χρήση του Αντικειμένου Αδειοδότησης που Εσείς κάνετε και τα οποία παραχωρούνται από το/τη Χορηγούντα/-ούσα την Άδεια (Αδειοδότη/-ούσα) ο/η οποίος/-α έχει σχετική εξουσία αδειοδότησης.

**η. Χορηγών/-ούσα την Άδεια (Αδειοδότης/-ούσα)** σημαίνει το ένα ή περισσότερα φυσικά ή νομικά πρόσωπα τα οποία παραχωρούν εξουσίες χρήσης υπό τους όρους της παρούσας Δημόσιας Άδειας.

**ι. Διαμοιρασμός** σημαίνει την παροχή υλικού στο κοινό με οποιονδήποτε τρόπο ή μέθοδο που προϋποθέτει προηγούμενη άδεια σύμφωνη με τα Αδειοδοτούμενα Δικαιώματα, όπως η αναπαραγωγή, η παρουσίαση στο κοινό, η δημόσια εκτέλεση, η διανομή, η διάδοση, η εισαγωγή προς διανομή και η διάθεση του στο κοινό με τρόπο στον οποίο συμπεριλαμβάνεται η δυνατότητα του κοινού να έχει πρόσβαση σε αυτό από τόπο και σε χρόνο της επιλογής του.

**j. Δικαίωμα Ειδικής Φύσης Κατασκευαστή/-τριας Βάσης Δεδομένων** σημαίνει δικαιώματα διαφορετικά από τα δικαιώματα

πνευματικής ιδιοκτησίας τα οποία προκύπτουν από την Οδηγία 96/9/ΕΚ της 11ης Μαρτίου 1996 του Ευρωπαϊκού Κοινοβουλίου και της Επιτροπής σχετικά με τη νομική προστασία των βάσεων δεδομένων, όπως έχει τροποποιηθεί ή/και ισχύει, καθώς επίσης και κάθε άλλο όμοιο στη φύση του δικαίωμα που υφίσταται οπουδήποτε στον κόσμο.

**κ. Εσείς** σημαίνει το φυσικό ή νομικό πρόσωπο που ασκεί τα Αδειοδοτούμενα Δικαιώματα που σας παραχωρούνται βάσει της παρούσας Δημόσιας Άδειας. **Δικό σας** έχει αντίστοιχο νόημα.

## **Άρθρο 2. Πεδίο εφαρμογής.**

### **α. Χορήγηση άδειας.**

1. Σύμφωνα με τους όρους και τις προϋποθέσεις της παρούσας Δημόσιας Άδειας, ο/η Χορηγών/-ούσα την Άδεια (Αδειοδότης/-ούσα) αποφασίζει να Σας παρέχει εφεξής μια παγκόσμια, άνευ αμοιβής, χωρίς δικαίωμα υποαδειοδότησης, μη αποκλειστική, μη ανακλητή άδεια για άσκηση των Αδειοδοτούμενων Δικαιωμάτων στο Αντικείμενο της Αδειοδότησης ώστε:

A. να αναπαράγετε και να πραγματοποιείτε Διαμοιρασμό του Αντικειμένου της Αδειοδότησης, εν όλω ή εν μέρει, και

B. να παράγετε, να αναπαράγετε, και να πραγματοποιείτε Διαμοιρασμό του Υλικού που Προέρχεται από Προσαρμογή, Τροποποίηση ή Διασκευή (Τροποποίηση).

2. Εξαιρέσεις και Περιορισμοί. Προς αποφυγή αμφιβολιών, στις περιπτώσεις όπου για τη δική Σας χρήση ισχύουν Εξαιρέσεις και Περιορισμοί, αυτή η Δημόσια Άδεια δεν εφαρμόζεται, και Έσείς δεν χρειάζεται να συμμορφώνεστε με τους όρους και τις προϋποθέσεις της.

3. Διάρκεια. Η διάρκεια της παρούσας Δημόσιας Άδειας καθορίζεται στο Άρθρο 6(a).

4. Μέσα και μορφότυπα · επιτρεπτές τεχνικές τροποποιήσεις. Ο/Η Χορηγών/-ούσα την Άδεια (Αδειοδότης/-ούσα) εξουσιοδοτεί Έσάς να ασκήσετε τα Αδειοδοτούμενα Δικαιώματα σε όλα τα μέσα και μορφότυπα, τα οποία είτε είναι τώρα γνωστά είτε θα δημιουργηθούν

στο μέλλον, και να κάνετε τις τεχνικές τροποποιήσεις οι οποίες είναι απαραίτητες να γίνουν. Ο/Η Χορηγών/-ούσα την Άδεια (Αδειοδότης/-ούσα) παραιτείται από και/ή συμφωνεί να μην διεκδικήσει οποιοδήποτε δικαίωμα ή εξουσία να Σας απαγορεύσει να προβείτε σε τεχνικές τροποποιήσεις απαραίτητες για την άσκηση των εξουσιών που σας παραχωρούνται βάσει της άδειας, συμπεριλαμβανομένων των τεχνικών τροποποιήσεων που είναι απαραίτητες για την παράκαμψη Αποτελεσματικών Τεχνολογικών Μέτρων. Για τους σκοπούς της παρούσας Δημόσιας Άδειας, προβαίνοντας στις τροποποιήσεις που επιτρέπονται βάσει αυτού του άρθρου 2(a)(4) δεν παράγεται σε καμία περίπτωση Υλικό που Προέρχεται από Προσαρμογή, Τροποποίηση ή Διασκευή (Τροποποίηση).

#### 5. Μεταγενέστεροι αποδέκτες.

A. Προσφορά από το/τη Χορηγούντα/-ούσα την Άδεια (Αδειοδότη/-ούσα) – Αντικείμενο Αδειοδότησης. Κάθε αποδέκτης του Αντικειμένου Αδειοδότησης λαμβάνει αυτόματα μια προσφορά από το/τη Χορηγούντα/-ούσα την Άδεια (Αδειοδότη/-ούσα) να ασκήσει τα Αδειοδοτούμενα Δικαιώματα σύμφωνα με τους όρους και τις προϋποθέσεις της παρούσας Δημόσιας Άδειας.

B. Όχι περιορισμοί μεταγενέστερης χρήσης. Δεν μπορείτε να προτείνετε ή να επιβάλετε πρόσθετους ή διαφορετικούς όρους ή προϋποθέσεις, ή να εφαρμόζετε Αποτελεσματικά Τεχνολογικά Μέτρα στο Αντικείμενο Αδειοδότησης εάν αυτό περιορίζει την άσκηση των Αδειοδοτούμενων Δικαιωμάτων σε οποιοδήποτε/οποιαδήποτε αποδέκτη του Αντικειμένου Αδειοδότησης.

6. Ανεξαρτησία Μερών. Καμία διάταξη της παρούσας Δημόσιας Άδειας δεν αποτελεί ή μπορεί να ερμηνευτεί ως άδεια για να επιβεβαιώσει ή να υπονοήσει ότι Εσείς είστε ή ότι η Δική Σας χρήση του Αντικειμένου Αδειοδότησης συνδέεται με, ή χρηματοδοτείται, ή αναγνωρίζεται, ή της παραχωρείται επίσημο καθεστώς από τον/την Χορηγούντα/-ούσα την Άδεια (Αδειοδότη/-ούσα) ή από άλλους που ορίζεται όπως λάβουν αναφορά δημιουργού, όπως προβλέπεται στο Άρθρο 3(a)(1)(A)(i).

#### **b. Άλλα δικαιώματα.**

1. Ηθικά δικαιώματα, όπως το δικαίωμα της ακεραιότητας, δεν αδειοδοτούνται υπό την παρούσα Δημόσια Άδεια, καθώς ούτε και το δικαίωμα στη δημοσιότητα ("right of publicity") στην ιδιωτικότητα ή άλλα παρόμοια δικαιώματα προσωπικότητας · ωστόσο, στο μέτρο του δυνατού, ο/η Χορηγών/-ούσα την Άδεια (Αδειοδότης/-ούσα) αποποιείται και/ή συμφωνεί να μην επικαλεστεί κανένα τέτοιο δικαίωμα, που κατέχει ο/η Χορηγών/-ούσα την Άδεια (Αδειοδότης/-ούσα), στο μέτρο που είναι απαραίτητο ώστε να επιτραπεί σε Εσάς να ασκήσετε τα Αδειοδοτούμενα Δικαιώματα που παραχωρούνται βάσει αυτής της άδειας, αλλά όχι με άλλον τρόπο.

2. Δικαιώματα ευρεσιτεχνιών και σημάτων δεν αδειοδοτούνται υπό την παρούσα Δημόσια Άδεια.

3. Στο μέτρο του δυνατού, ο/η Χορηγών/-ούσα την Άδεια (Αδειοδότης/-ούσα) αποποιείται έναντί Σας κάθε δικαιώματος είσπραξης τελών από τα Αδειοδοτούμενα Δικαιώματα, είτε άμεσα είτε μέσω οργανισμού συλλογικής διαχείρισης βάσει οποιωνδήποτε διατάξεων αναγκαστικού ή ενδοτικού δικαίου ή βάσει υποχρεωτικού συστήματος αδειοδότησης. Σε όλες τις άλλες περιπτώσεις ο/η Χορηγών/-ούσα την Άδεια (Αδειοδότης/-ούσα) ρητά διατηρεί το δικαίωμα είσπραξης των εν λόγω αμοιβών.

### **Άρθρο 3. Όροι της Άδειας.**

Η άσκηση των εξουσιών που σας παραχωρούνται βάσει της άδειας υπόκειται ρητά στις εξής προϋποθέσεις.

#### **α. Αναφορά.**

1. Αν πραγματοποιείτε Διαμοιρασμό του Αντικειμένου της Αδειοδότησης (συμπεριλαμβανομένης της τροποποιημένης μορφής), Εσείς είστε υποχρεωμένοι/-ες:

Α. να διατηρήσετε, εφόσον παρέχονται από το/τη Χορηγούντα/-ούσα την Άδεια (Αδειοδότη/-ούσα) με το Αντικείμενο Αδειοδότησης τα εξής:

i. αναγνώριση του/της/των δημιουργού/ δημιουργών του Αντικειμένου Αδειοδότησης και οποιωνδήποτε άλλων προσώπων που έχει οριστεί ότι δικαιούνται αναγνώρισης με οποιονδήποτε εύλογο τρόπο, ο

οποίος απαιτείται από το/τη Χορηγούντα/-ούσα την Άδεια (Αδειοδότη/-ούσα) (συμπεριλαμβανομένου του ψευδωνύμου, αν αυτό έχει οριστεί)

ii. γνωστοποίηση των Δικαιωμάτων Πνευματικής Ιδιοκτησίας

iii. γνωστοποίηση που αναφέρεται στην παρούσα Δημόσια Άδεια

iv. γνωστοποίηση που αναφέρεται στην αποποίηση εγγυήσεων

v. κανονιστικό Αναγνωριστικό Πόρου (URI) ή υπερσύνδεσμο στο Αντικείμενο Αδειοδότησης, στο βαθμό που αυτό είναι εύλογα δυνατό.

B. να αναφέρετε αν Εσείς τροποποιήσατε το Αντικείμενο Αδειοδότησης και να διατηρείτε αναφορά για οποιαδήποτε προηγούμενη τροποποίηση και

C. να αναφέρετε ότι το Αντικείμενο Αδειοδότησης αδειοδοτείται υπό τους όρους της παρούσας Δημόσιας Άδειας και να συμπεριλαμβάνετε το κείμενο ή το Κανονιστικό Αναγνωριστικό Πόρου (URI) ή τον υπερσύνδεσμο, της παρούσας Δημόσιας Άδειας.

2. Μπορείτε να ικανοποιήσετε τις προϋποθέσεις του Άρθρου 3(a)(1) με οποιονδήποτε εύλογο τρόπο βάσει του μέσου, του τρόπου και του πλαισίου με τα οποία Εσείς πραγματοποιήσατε Διαμοιρασμό του Αντικειμένου Αδειοδότησης. Για παράδειγμα, μπορεί να είναι εύλογο να ικανοποιήσετε τις προϋποθέσεις παρέχοντας ένα Κανονιστικό Αναγνωριστικό Πόρου (URI) ή έναν υπερσύνδεσμο σε μια πηγή που περιλαμβάνει τις απαιτούμενες πληροφορίες.

3. Αν ζητηθεί από το/τη Χορηγούντα/-ούσα την Άδεια (Αδειοδότη/-ούσα), είστε υποχρεωμένοι/-ες να αφαιρέσετε οποιαδήποτε πληροφορία που απαιτείται από το Άρθρο 3(a)(1)(A) στο βαθμό που αυτό είναι δυνατόν.

b. Αν πραγματοποιείτε Διαμοιρασμό του Υλικού που Προέρχεται από Προσαρμογή, Τροποποίηση ή Διασκευή (Τροποποίηση) που Εσείς έχετε πραγματοποιήσει, η Άδεια του Διενεργούντος/-ούσας την Προσαρμογή που εφαρμόζετε δεν πρέπει να εμποδίζει αποδέκτες του Υλικού που προέρχεται από Προσαρμογή, Τροποποίηση ή Διασκευή

(Τροποποίηση) από το να συμμορφώνονται με την παρούσα Δημόσια Άδεια.

#### **Άρθρο 4. Δικαιώματα Ειδικής Φύσης Κατασκευαστή/-τριας Βάσης Δεδομένων.**

Στις περιπτώσεις όπου τα Αδειοδοτούμενα Δικαιώματα περιλαμβάνουν Δικαιώματα Ειδικής Φύσης Κατασκευαστή/-τριας Βάσης Δεδομένων τα οποία εφαρμόζονται στη χρήση που Εσείς πραγματοποιείτε στο Αντικείμενο της Αδειοδότησης:

α. προς άρση κάθε αμφιβολίας, το Άρθρο 2(a)(1) παραχωρεί σε Εσάς το δικαίωμα να εξάγετε, να επαναχρησιμοποιείτε, να αναπαράγετε και να πραγματοποιείτε Διαμοιρασμό του συνόλου ή ουσιώδους μέρος των περιεχομένων της βάσης δεδομένων

β. εάν Εσείς περιλαμβάνετε όλο ή το ουσιώδες μέρος των περιεχομένων της βάσης δεδομένων σε μία βάση δεδομένων στην οποία Εσείς έχετε Δικαιώματα Ειδικής Φύσης Κατασκευαστή/-τριας Βάσης Δεδομένων, τότε η βάση δεδομένων στην οποία Εσείς έχετε Δικαιώματα Ειδικής Φύσης Κατασκευαστή/-τριας Βάσης Δεδομένων (αλλά όχι τα μεμονωμένα στοιχεία του περιεχομένου της) είναι Υλικό που Προέρχεται από Προσαρμογή, Τροποποίηση ή Διασκευή (Τροποποίηση), και

γ. πρέπει να συμμορφώνεστε με τους όρους του Άρθρου 3(a) εάν Εσείς πραγματοποιείτε Διαμοιρασμό του συνόλου ή ουσιώδους μέρους των περιεχομένων της βάσης δεδομένων.

Προς άρση κάθε αμφιβολίας, το εν λόγω Άρθρο 4 συμπληρώνει και δεν αντικαθιστά τις υποχρεώσεις Σας δυνάμει της παρούσας Δημόσιας Άδειας όπου τα Αδειοδοτούμενα Δικαιώματα περιλαμβάνουν άλλα Δικαιώματα Πνευματικής Ιδιοκτησίας και Συγγενικά ή Παρόμοια Δικαιώματα.

#### **Άρθρο 5. Αποποίηση Εγγυήσεων και Περιορισμός Ευθύνης.**

α. Εκτός εάν άλλως ξεχωριστά αναλάβει ο/η Χορηγών/-ούσα την Άδεια (Αδειοδότης/-ούσα), στο μέτρο του δυνατού, ο/η Χορηγών/-ούσα την Άδεια (Αδειοδότης/-ούσα) προσφέρει το Αντικείμενο Αδειοδότησης ως έχει και όπως διατίθεται, και δεν προβαίνει σε

δηλώσεις ή εγγυήσεις οποιουδήποτε είδους σχετικά με το Αντικείμενο Αδειοδότησης, είτε ρητές, είτε σιωπηρές, εκ του νόμου, ή άλλες. Αυτό περιλαμβάνει, χωρίς περιορισμό, εγγυήσεις τίτλου, εμπορευσιμότητας, καταλληλότητας για συγκεκριμένο σκοπό, μη-παράβασης, απουσία λανθάνοντων ή άλλων ελαττωμάτων, ακρίβειας, ή σχετικά με την παρουσία ή απουσία σφαλμάτων, είτε αγνώστων είτε εντοπίσιμων. Όπου αποποιήσεις των εγγυήσεων δεν επιτρέπονται εν όλω ή εν μέρει, αυτή η δήλωση αποποίησης μπορεί να μην ισχύει για Εσάς.

b. Στο μέτρο που είναι δυνατό, σε καμία περίπτωση δεν θα ευθύνεται ο/η Χορηγών/-ούσα την Άδεια (Αδειοδότης/-ούσα) έναντί Σας δυνάμει οποιασδήποτε νομικής αρχής ή βάσης (συμπεριλαμβανομένης, χωρίς περιορισμό, και της αμέλειας) ή άλλως για τυχόν άμεσες, ειδικές, έμμεσες, τυχαίες, επακόλουθες, τιμωρητικές, παραδειγματικές ή άλλες απώλειες, έξοδα, δαπάνες ή ζημίες που προκύπτουν από την παρούσα Δημόσια Άδεια ή τη χρήση του Αντικειμένου Αδειοδότησης, ακόμη και αν ο/η Χορηγών/-ούσα την Άδεια (Αδειοδότης/-ούσα) έχει ενημερωθεί για την πιθανότητα τέτοιων απωλειών, εξόδων, δαπανών ή ζημιών. Σε περίπτωση που ο περιορισμός ευθύνης δεν επιτρέπεται εν όλω ή εν μέρει, ο παρών περιορισμός δεν δύναται να ισχύει για Εσάς.

c. Η δήλωση αποποίησης εγγυήσεων και περιορισμού της ευθύνης που προβλέπεται ως ανωτέρω θα ερμηνεύεται κατά τρόπο που, στο μέτρο του δυνατού, προσεγγίζει περισσότερο την απόλυτη αποποίηση και απαλλαγή από κάθε ευθύνη.

#### **Άρθρο 6. Διάρκεια και Τερματισμός.**

a. Η παρούσα Δημόσια Άδεια έχει διάρκεια για όλο το χρόνο ισχύος των Δικαιωμάτων Πνευματικής Ιδιοκτησίας και Συγγενικών ή Παρόμοιων Δικαιωμάτων που παραχωρούνται εδώ. Ωστόσο, εάν Εσείς δεν συμμορφωθείτε με την παρούσα Δημόσια Άδεια, τότε τα δικαιώματά Σας δυνάμει της παρούσας Δημόσιας Άδειας τερματίζονται αυτόματα.



**b.** Όπου το δικαίωμά Σας να χρησιμοποιείτε το Αντικείμενο της Αδειοδότησης έχει τερματιστεί δυνάμει του Άρθρου **6(a)**, αυτό αποκαθίσταται:

1. αυτόματα κατά την ημερομηνία όπου η παράβαση έχει θεραπευτεί, υπό την προϋπόθεση ότι έχει θεραπευτεί μέσα σε 30 μέρες από τότε που λάβατε γνώση για την παράβαση ή

2. κατόπιν ρητής αποκατάστασης από το/τη Χορηγούντα/-ούσα την Άδεια (Αδειοδότη/-ούσα).

Προς άρση κάθε αμφιβολίας, το εν λόγω Άρθρο **6(b)** δεν επηρεάζει κανένα δικαίωμα του/της Χορηγούντος/-ούσας την Άδεια (Αδειοδότη/-ούσας) να αξιώσει θεραπείες λόγω παραβιάσεων Σας της παρούσας Δημόσιας Άδειας.

c. Προς άρση κάθε αμφιβολίας, ο/η Χορηγών/-ούσα την Άδεια (Αδειοδότης/-ούσα) μπορεί επίσης να παρέχει το Αντικείμενο της Αδειοδότησης δυνάμει ξεχωριστών όρων ή προϋποθέσεων ή να σταματήσει τη διανομή του Αντικειμένου της Αδειοδότησης σε οποιοδήποτε χρόνο: ωστόσο, αυτή του/της η ενέργεια δεν θα τερματίσει την παρούσα Δημόσια Άδεια.

**d.** Η ισχύς των οριζόμενων στα Άρθρα **1, 5, 6, 7** και **8** διατηρείται, παρά τον τερματισμό της παρούσας Δημόσιας Άδειας.

### **Άρθρο 7. Άλλοι Όροι και Προϋποθέσεις.**

a. Ο/Η Χορηγών/-ούσα την Άδεια (Αδειοδότης/-ούσα) δεν δεσμεύεται από τυχόν πρόσθετους ή διαφορετικούς όρους ή προϋποθέσεις που ανακοινώθηκαν από Εσάς, εκτός εάν αυτό έχει συμφωνηθεί ρητά.

b. Οποιοσδήποτε διευθετήσεις, μνημόνια ή συμφωνίες σχετικά με το Αντικείμενο Αδειοδότησης που δεν αναφέρονται στην παρούσα Άδεια είναι ξεχωριστές και ανεξάρτητες από τους όρους και τις προϋποθέσεις της παρούσας Δημόσιας Άδειας.

### **Άρθρο 8. Ερμηνεία.**

a. Για την αποφυγή αμφιβολιών, η παρούσα Δημόσια Άδεια δεν πρέπει να ερμηνευθεί ως να μειώνει, να περιορίζει, να απαγορεύει ή να επιβάλλει προϋποθέσεις για οποιαδήποτε χρήση του Αντικειμένου

Αδειοδότησης που θα μπορούσε νομίμως να γίνει χωρίς άδεια, βάσει της παρούσας Δημόσιας Άδειας.

β. Στο μέτρο του δυνατού, αν οποιαδήποτε διάταξη της παρούσας Δημόσιας Άδειας κριθεί μη εφαρμόσιμη, πρέπει να αναθεωρηθεί αυτόματα στο ελάχιστο αναγκαίο μέτρο ώστε να καταστεί εκτελεστή. Αν η διάταξη αυτή δεν μπορεί να αναθεωρηθεί, πρέπει να αποκοπεί από αυτή τη Δημόσια Άδεια χωρίς να επηρεάζεται η εκτελεστότητα των υπόλοιπων όρων και προϋποθέσεων.

γ. Κανένας όρος ή προϋπόθεση της παρούσας Δημόσιας Άδειας, δεν θα γίνει αντικείμενο παραίτησης και καμία παράλειψη συμμόρφωσης δε θα θεραπεύεται εκτός εάν συμφωνηθεί ρητώς από τον/τη Χορηγούντα/-ούσα την Άδεια (Αδειοδότη/-ούσα).

δ. Καμία πρόβλεψη της παρούσας Δημόσιας Άδειας δεν αποτελεί ή δύναται να ερμηνευθεί ως περιορισμός επί, ή παραίτηση από, όλα τα προνόμια και τις ασυλίες που απολαμβάνει ο/η Χορηγών/-ούσα την Άδεια (Αδειοδότη/-ούσα) ή Εσείς συμπεριλαμβανομένων αυτών που προκύπτουν από νομικές διαδικασίες οποιασδήποτε δικαιοδοσίας ή αρχής.

Το νομικό πρόσωπο Creative Commons δεν είναι συμβαλλόμενο μέρος στις δημόσιες άδειές του. Ωστόσο, το νομικό πρόσωπο Creative Commons μπορεί να επιλέξει να εφαρμόσει κάποια από τις άδειες δημόσιας χρήσης του στο υλικό που δημοσιεύει και σε αυτές τις περιπτώσεις θα θεωρείται ως "Ο/Η Χορηγών/-ούσα την Άδεια (Αδειοδότη/-ούσα)". Το κείμενο των δημόσιων αδειών Creative Commons είναι αφιερωμένο στο δημόσιο τομέα υπό το [CC0 Public Domain Dedication](#). Εκτός από τον περιορισμένο σκοπό διαμοιρασμού υλικού δυνάμει μιας δημόσιας άδειας Creative Commons ή όπως άλλως επιτρέπεται από την πολιτική του νομικού προσώπου Creative Commons όπως αυτή δημοσιεύεται στη διαδικτυακή διεύθυνση [creativecommons.org/policies](http://creativecommons.org/policies), το νομικό πρόσωπο Creative Commons δεν επιτρέπει τη χρήση του εμπορικού σήματος "Creative Commons" ή άλλου εμπορικού σήματος ή του λογοτύπου του νομικού προσώπου Creative Commons, χωρίς την προηγούμενη γραπτή συγκατάθεσή του

συμπεριλαμβανομένων, χωρίς περιορισμό, για τη χρήση αναφορικά με οποιοσδήποτε μη εξουσιοδοτημένες τροποποιήσεις οποιασδήποτε δημόσιας άδειάς του ή τυχόν άλλων διευθετήσεων, μνημονίων, ή συμφωνιών που αφορούν τη χρήση του αντικειμένου αδειοδότησης. Προς άρση κάθε αμφιβολίας, η εν λόγω παράγραφος δεν αποτελεί περιεχόμενο των δημόσιων αδειών.

Μπορείτε να επικοινωνήσετε με το νομικό πρόσωπο Creative Commons στη διαδικτυακή διεύθυνση [creativecommons.org](https://creativecommons.org).





