Το βιβλίο της Gambas3

Δημοσθένης Κόπτσης

- 144 παραδείγματα
- πλήρης οδηγός
- περιγραφή όλων των εντολών και συναρτήσεων
- visual programming για Linux σε BASIC
- χτίστε τις δικές σας εφαρμογές

CC BY SA

Το βιβλίο της Gambas3

Το βιβλίο της Gambas3

Copyright © 2023 Δημοσθένης Κόπτσης
Διατηρούνται ορισμένα δικαιώματα.

Το έργο αυτό αδειοδοτείται υπο την Άδεια Creative Commons Αναφορά προέλευσης - Παρόμοια Διανομή 3.0 (Atribution–Share Alike 3.0). Προκειμένου να δείτε ένα αντίγραφο αυτής της άδειας, επισκεφτείτε τη διεύθυνση http://creativecommons.org/licenses/by-sa/3.0/
legalcode ή στείλτε γράμμα στην Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.

Πίνακας Περιεχομένων

Εισαγωγή....19
Μια σύντομη ανασκόπηση στις γλώσσες προγραμματισμού....21
Προγράμματα....22
Διαδικαστικός, Δομημένος και Αντικειμενοστραφής προγραμματισμός....22
Εισαγωγή στην Gambas....25
Μέρος 1ο, Το περιβάλλον ανάπτυξης....27
Κεφάλαιο 1ο - Η πρώτη μου εφαρμογή “Hello World!”....27
Κεφάλαιο 2ο – Μενού περιβάλλοντος ανάπτυξης εφαρμογών....34
Κύρια Μενού....34
Αρχείο (File)....34
Άνοιγμα έργου (*Open project*)....35
Άνοιγμα πρόσφατου (*Open recent*)....36
Αποθήκευση έργου (*Save project*)....37
Αποθήκευση έργου ως (*Save project as*)....38
Έξοδος (*Quit*)....38
Έργο (Project)....38
Μεταγλώττισε (Compile)....39
Μεταγλώττισε όλα (Compile All)....39
Φτιάξε Εκτελέσιμο (Make Executable)....39
Ρύθμιση εκτελέσιμου (Configure executable)....40
Τοποθέτησε κάτω από Έλεγχο Έκδοσης (Put under version control) - Με το GIT (With Git)....41
Φτιάξε Ενημέρωση Κώδικα (Create Patch)....41
Εφάρμωσε Ενημέρωση Κώδικα (Apply Patch)....44
Φτιάξε Πακέτο πηγαίου κώδικα (Make Source archive)....45
Φτιάξε Πακέτο εγκατάστασης (Installation package)....46
Οδηγός δημιουργίας πακέτου εγκατάστασης....46
Βήμα 1ο....46
Βήμα2ο....47
Βήμα 3ο....48
Βήμα 4ο....49
Βήμα 5ο....50
Βήμα 6ο....51
Βήμα 7ο....52
Βήμα 8ο....53
Βήμα 9ο....54
Βήμα 10ο....55
Διαμοίρασε στην φάρμα λογισμικού (Share on software farm)....56
Καθάρισε (Clean up)....56
Ανανέωση (Refresh)....56
Ιδιότητες (Properties)....57
Καρτέλα “Περιγραφή (Description)”....57
Καρτέλα “Εξαρτήματα (Components)”....58
Καρτέλα “Βιβλιοθήκες (Libraries)”....59

Καρτέλα “Επιλογές (Options)”....59
Καρτέλα “Στατιστιά (Statistics)”....60
Αποσφαλμάτωση (Debug)....61
Ρύθμιση της αποσφαλμάτωσης (Configuration…)....62
Προβολή (View)....64
Εργαλεία (Tools)....65
Η μπάρα συντομεύσεων του κυρίως μενού....71
Βοήθεια (?)....74
Κεφάλαιο 3ο - Προβολή σχεδίασης....75
Σχεδίαση του έργου “Hello World”....76
Εικονίδια εντολών σχεδίασης....79
Ο συντάκτης μενού (Menu Editor)....82
Κεφάλαιο 4ο - Προβολή συγγραφής κώδικα....85
Μπάρα εντολών συγγραφής κώδικα....86
Το μενού του ποντικιού στην προβολή συγγραφής κώδικα....87
Επιστροφή στο παράδειγμα του Project1, “Hello World”....88
Ιδιότητες αντικειμένων και προσπέλαση τους....91
Μέρος 2ο, Η γλώσσα Basic της Gambas....93
Κεφάλαιο 1ο - Τα βασικά μέρη της γλώσσας....93
Αλφάβητο και Λεξιλόγιο της γλώσσας προγραμματισμού....93
Σύνταξη της γλώσσας....93
Σχόλια μέσα σε ένα πρόγραμμα....94
Κεφάλαιο 2ο - Τι είναι μια ρουτίνα....96
Κλήσεις ρουτινών....97
Ορισμοί ρουτινών....99
Υπορουτίνες....99
Εμβέλεια υπορουτινών....100
Συναρτήσεις....104
Διοχέτευση παραμέτρων ByRef....108
Κεφάλαιο 3ο - Μεταβλητές....111
Τύποι δεδομένων (Datatypes)....112
Μια ματιά στους τύπους δεδομένων....113
Boolean....113
Byte....113
Short....113
Integer....113
Η δεκαδική μορφή....114
Η δεκαεξαδική μορφή....114
Η δυαδική μορφή....114
Long....114
Single....114
Float....115
Date....115
String....115
Variant....115

Object....115
Pointer....116
Δηλώνοντας μεταβλητές....117
Εμβέλεια μεταβλητών....117
Σύνταξη ορισμού μεταβλητών....117
Σύνταξη καθολικών μεταβλητών....117
Εναλλακτικός τρόπος σύνταξης καθολικών μεταβλητών....118
Σύνταξη τοπικών μεταβλητών....118
Τι είναι ένας Προσδιορισμός, Identifier....119
Εναλλακτικός τρόπος δήλωσης τοπικών μεταβλητών....120
Δήλωση πολλών μεταβλητών....120
Παράδειγμα εμβέλειας μεταβλητών....120
Παράδειγμα δηλώσεων μεταβλητών διαφόρων τύπων δεδομένων....123
Στυλ γραφής μεταβλητών....123
Ονόματα μεταβλητών....124
Παραδείγματα....124
Ονόματα αντικειμένων φορμών....125
Ανακύκλωση μεταβλητών....126
Ειδικοί χαρακτήρες στοιχειοσειρών (strings)....129
Μετατροπή τιμών μεταβλητών....130
CBool....131
CByte....132
CDate....132
CFloat....134
CInt....135
CLong....136
Conv$....136
CShort....137
CSingle....138
CStr....139
Cvariant....140
DConv$....140
SConv$....140
Str$....141
Val....141
Κεφάλαιο 4ο - Σταθερές....143
Δηλώνοντας συμβολικές σταθερές....143
Σύνταξη σταθερών....144
Τύποι δεδομένων σταθερών....144
Παραδείγματα συμβολικών σταθερών....145
Σταθερές κυριολεκτικών τιμών ή σταθερές της γλώσσας προγραμματισμού....145
Προκαθορισμένες συμβολικές σταθερές....146
Τύποι δεδομένων προκαθορισμένων συμβολικών σταθερών....146
Τύποι αρχείων....146
Σταθερές αλφαριθμητικών (strings)....146
Τύποι ταξινόμησης....147

Ημέρες της εβδομάδος....147
Προκαθορισμένες μορφές....148
Μορφές αριθμών....148
Μορφές ημερομηνίας και χρόνου....148
Μορφές νομισματικών μονάδων....148
Διάφορες μορφές....149
Κατάταξη ψηφιολέξεων (Endianness ή Byte order)....149
Κεφάλαιο 5ο – Προτάσεις και εκφράσεις....150
Προτάσεις (statements)....150
Μπλοκ κώδικα....150
Εκφράσεις (expressions)....151
Τελεστές....151
Τελεστές εκχώρησης....152
Τελεστές αριθμητικοί....153
Γενικοί αριθμητικοί τελεστές....153
Αριθμητικοί τελεστές σύγκρισης....154
Προτεραιότητα αριθμητικών τελεστών....154
Λογικοί τελεστές....155
Τελεστές αλφαριθμητικών....157
Τελεστές αλφαριθμητικών γενικής χρήσεως....157
Τελεστές σύγκρισης αλφαριθμητικών....158
Προτεραιότητα τελεστών....159
Κεφάλαιο 6ο – Εμβάθυνση στις συναρτήσεις....160
Αναλυτικά τα μέρη μιας συνάρτησης....160
Εμβέλεια μεταβλητών....164
Τοπικές μεταβλητές....164
Ονόματα τοπικών μεταβλητών....167
Οι παράμετροι είναι τοπικές μεταβλητές....169
Καθολικές μεταβλητές....171
Τα ορίσματα των συναρτήσεων....172
Τιμές επιστροφής....173
Αναδρομή συναρτήσεων....175
Κεφάλαιο 7ο – Ροή προγράμματος....180
Εντολές λήψης λογικών αποφάσεων....181
Choose....181
IIf....181
IF ... THEN ... ELSE IF ...THEN ... ELSE ... ENDIF....182
SELECT CASE ... END SELECT....184
Εντολές βρόχων....186
FOR...NEXT....186
Λίγα λόγια για τις ένθετες προτάσεις....189
DO...LOOP....191
REPEAT...UNTIL....196
WHILE....WEND....197
FOR EACH...NEXT....197

Εντολή μετάβασης GOTO....199
Εντολές διαχείρισης λαθών....201
CATCH....201
DEBUG....203
Λίγα λόγια για τους “περιγραφείς αρχείων” (FD, file descriptors)....205
ERROR....206
FINALLY....208
TRY....209
Εντολές QUIT, END, RETURN, STOP, BREAK, SLEEP, WAIT....210
QUIT....210
END....211
RETURN....211
STOP....211
BREAK....212
SLEEP....212
WAIT....213
Λίγα λόγια για τον βρόχο συμβάντων (EVENT LOOP)....214
Κεφάλαιο 8ο – Διατάξεις ή Πίνακες....215
Δήλωση διατάξεων....217
Καθολικές μη δυναμικές διατάξεις....217
Δυναμικές διατάξεις....219
Δήλωση τοπικών δυναμικών διατάξεων....219
Δήλωση καθολικών δυναμικών διατάξεων....220
Αρχικοποίηση δυναμικών πινάκων....220
Ο βρόχος FOR EACH ... NEXT....222
Δυναμικές διατάξεις ως αντικείμενα....225
Οι κλάσεις Boolean[], Date[], Float[], Integer[], Long[], Short[], Pointer[], Single[]. 226
Η κλάση String[]....236
Η κλάση Object[]....237
Η κλάση Variant[]....237
Κεφάλαιο 9ο – Δομές (Structures)....238
Δήλωση δομών....238
Διαχείριση των πεδίων....240
Ενσωματωμένες δομές....241
Παραδείγματα δηλώσεων δομών....241
Διατάξεις δομών....242
Κεφάλαιο 10ο – Συναρτήσεις της Gambas....243
Αριθμητικές συναρτήσεις (Arithmetical Functions)....243
Abs....244
Ceil....244
DEC....245
Fix....245
Floor....246
Frac....246
INC....247

Int....247
Max....248
Min....248
Round....249
Sgn....249
Συναρτήσεις διαχείρισης Bits (Bits Manipulation Functions)....250
Bits και Bytes....250
Αναπαράσταση αρνητικών αριθμών στο δυαδικό σύστημα....250
BClr....252
BSet....253
BTst....253
BChg....254
Lsl....255
Lsr....255
Shl / Asl....257
Shr / Asr....258
Rol....259
Ror....260
Συναρτήσεις ελέγχου χαρακτήρα (Character Test Functions)....262
IsAscii....262
IsBlank....262
IsDigit....263
IsHexa....263
IsLCase....263
IsLetter....264
IsPunct....264
IsSpace....264
IsUCase....265
Συναρτήσεις μετατροπής τύπων δεδομένων (Conversion Functions)....266
Συναρτήσεις τύπων δεδομένων (Datatype Functions)....267
IsBoolean....267
IsDate....268
IsFloat....269
IsInteger....270
IsLong....271
IsNull....272
IsNumber....273
TypeOf....275
Συναρτήσεις ημερομηνίας και χρόνου (Date & Time Functions)....277
Date....277
DateAdd....278
DateDiff....279
Day....280
Hour....280
Minute....281

Month....282
Now....282
Second....283
Time....283
Timer....284
Week....284
WeekDay....285
Year....286
Συναρτήσεις αρχείων και καταλόγων (File & Directory Functions)....287
Access....287
COPY....288
DFree....289
Dir....289
Exist....290
IsDir....291
KILL....292
LINK....292
MKDIR....293
MOVE....294
RDir....294
RMDIR....295
Stat....295
Temp$....296
Συναρτήσεις μορφοποίησης (Formatting functions)....298
Bin$....298
Format$....299
Hex$....302
Συναρτήσεις μετάφρασης και τοπικής γλώσσας (Localization and Translation Functions) 304
Tr$....304
Περισσότερα για την μετάφραση ενός έργου....305
Οι μεταβλητές γλώσσας του περιβάλλοντος του λειτουργικού συστήματος....305
Μεταφράσεις String....307
Διαχείριση των String....307
Μετατροπές String....308
Τοπικές ρυθμίσεις για το GUI....308
Λογαριθμικές και εκθετικές συναρτήσεις (Logarithms & Exponentials Functions)....309
Cbr....309
Exp....310
Exp2....310
Exp10....311
Expm....311
Log....312
Log2....312
Log10....313
Logp....313

Sqr....314
Τριγωνομετρικές συναρτήσεις (Trigonometric Functions)....315
ACos....316
ACosh....316
Ang....317
ASin....317
ASinh....318
ATan....318
ATan2....319
ATanh....320
Cos....320
Cosh....321
Deg....321
Hyp....322
Mag....322
Pi....323
Sin....323
Sinh....324
Tan....324
Tanh....325
Rad....325
Συναρτήσεις τυχαίων αριθμών (Random Numbers Functions)....327
RANDOMIZE....327
Rnd....327
Συναρτήσεις διαχείρισης διεργασιών (Process Management)....329
EXEC....329
SHELL....332
SHELL$....334
Συναρτήσεις ροής και εισόδου/εξόδου (Stream & Input/Output functions)....335
Λίγα λόγια για την είσοδο και έξοδο....335
Οι συναρτήσεις ροής δεδομένων....337
OPEN....338
CLOSE....339
PRINT....339
Eof....342
Lof....342
ERROR....343
FLUSH....343
INPUT....343
LINE INPUT....345
READ....346
MEMORY....348
WRITE....348
OUTPUT TO....350
INPUT FROM....350

ERROR TO....351
PIPE....351
LOCK....352
UNLOCK....353
SEEK....353
Seek....354
Συναρτήσεις συμβολοσειρών (String Functions)....355
Asc....356
Chr$....357
Comp....358
Html$....359
InStr....359
LCase$....360
Left$....360
Len....361
LTrim$....362
Mid$....362
Quote$....363
Replace$....363
Right$....364
RInStr....365
RTrim$....366
Scan....366
Space$....369
Split....369
String$....370
Subst$....371
Trim$....372
UCase$....372
Unquote$....373
Διάφορες άλλες συναρτήσεις (Miscellaneous Functions)....374
Eval....374
Inline Arrays....375
Inline Collections....376
Διαχείριση εξωτερικών συναρτήσεων (External Function Management)....377
Η μνήμη και οι δείκτες στην γλώσσα C....378
Οι δείκτες στην Gambas3....380
Αναφορές σε αντικείμενα....386
Οι συναρτήσεις δεικτών....388
Alloc....388
Free....390
Realloc....391
StrPtr....391
VarPtr....393
Boolean@....393
Byte@....394

Float@....394
Long@....394
Integer@....395
Long@....395
Pointer@....395
Short@....396
Single@....396
String@....396
Προβλήματα με τους δείκτες....397
Διαρροές μνήμης (Memory Leak)....397
Αλλοίωση δεδομένων στην μνήμη (Memory Corruption)....397
Περιπλανώμενοι δείκτες (dangling pointers)....397
Η εκχώρηση νέας τιμής σε υπάρχων δείκτη δημιουργεί διαρροή μνήμης....398
Μη απελευθέρωση δείκτη οδηγεί σε διαρροή μνήμης....399
Εγγραφή δεδομένων με μεγαλύτερο μέγεθος μνήμης από αυτό που έχει δεσμευθεί. 400
Χρησιμοποίηση δείκτη που ήδη έχει απελευθερωθεί....401
Απελευθέρωση δείκτη που ήδη έχει απελευθερωθεί....402
Απελευθέρωση μνήμης που δεν έχει δεσμευτεί....403
Απόπειρα εγγραφής σε δείκτη που έχει ελευθερωθεί....404
Εγγραφή σε περιοχή μνήμης που ποτέ δεν δεσμεύτηκε....405
Ασφαλής χρήση δεικτών....406
Χρησιμοποιώντας δείκτες στις εξωτερικές συναρτήσεις της C....408
Εντολές διαχείρισης εξωτερικών συναρτήσεων....410
LIBRARY....410
EXTERN....410
Μέρος 3ο, Αντικειμενοστραφής προγραμματισμός (OOP)....419
Κεφάλαιο 1ο – Τι είναι οι κλάσεις και τα αντικείμενα....419
Δημιουργία αντικειμένων από κλάσεις....421
Εγγενείς κλάσεις (Native classes)....422
Κάθε αντικείμενο έχει ιδιότητες, μεθόδους, συμβάντα, μεταβλητές και σταθερές....426
Μεταβλητές (Variables)....426
Σταθερές (Constants)....427
Ιδιότητες (Properties)....429
Μέθοδοι (Methods)....430
Συμβάντα (Events)....431
Στατικές κλάσεις (Static classes)....432
Εικονικές κλάσεις (Virtual Classes)....432
Προσωπικές κλάσεις (Custom classes)....433
Κεφάλαιο 2ο – Gambas Object Model....436
Αντικείμενα (Objects) & Κλάσεις (Classes)....436
Κλάσεις (Classes)....437
Ιδιότητες (properties), μέθοδοι (methods) και μεταβλητές (variables)....438
Αναφορές (References)....438
Μη έγκυρα αντικείμενα (Invalid objects)....438

Ειδικές μέθοδοι (Special methods)....439
_init....440
_exit....440
_new....440
_free....441
_next....441
_get....442
_put....442
_call....443
_unknown....443
_compare....444
_attach....444
Κεφάλαιο 3ο Συμβάντα και Παρατηρητές (Events & Observers)....446
Συμβάντα (Events)....446
Κλειδώνοντας αντικείμενα (Locking Objects)....447
Παρατηρητές (Observers)....447
Κεφάλαιο 4ο Κληρονομικότητα (Inheritance)....449
Ποιες κλάσεις μπορούν να είναι η βάση για άλλες κλάσεις;....452
Εικονική διανομή (Virtual dispatching)....452
Κληρονομικότητα και μέθοδος δημιουργίας (Inheritance and constructor)....453
Η σειρά κλήσης των μεθόδων δημιουργίας....453
Η σειρά διοχέτευσης των παραμέτρων στην μέθοδο δημιουργίας....455
Κυκλική κληρονομικότητα (Circular Inheritance)....457
*Υπέρβαση συμβόλων (*Symbol Overriding)....458
Κεφάλαιο 5ο Πίνακες συμβόλων....461
Το προκαθορισμένο εσωτερικό εξάρτημα....461
Πίνακες συμβόλων....461
Ο πίνακας συμβόλων του έργου....461
Κεφάλαιο 6ο – Διαχείριση κλάσεων και αντικειμένων....462
CLASS....462
Class....463
CREATE PRIVATE....463
CREATE STATIC....464
EXPORT....464
Παράδειγμα:....464
INHERITS....465
IS....465
LAST....466
ME....467
NEW....468
Δυναμική υλοποίηση (Dynamic instanciation)....468
Object....470
SUPER....470
Κεφάλαιο 7ο – Διαχείριση ιδιοτήτων....472
Κεφάλαιο 8ο – Διαχείριση μεθόδων....475
Κεφάλαιο 9ο – Διαχείριση συμβάντων....477

EVENT....477
Διαχειριστής συμβάντος (Event Handler)....477
RAISE....480
Object.Attach....480
Object.Detach....480
STOP EVENT....481
WAIT....482
Μέρος 4ο, Δουλεύοντας με τις κλάσεις....484
Κεφάλαιο 1ο – Γραφική διεπαφή χρήστη (GUI)....484
gb.qt4....484
Δουλεύοντας με τις φόρμες, η κλάση Form....485
Ιδιότητες (Properties)....487
Action....487
Cursor....488
Font....489
Icon....490
Mask....491
Modal....492
ToolTip....493
Συμβάντα (Events)....494
KeyPress....494
Menu....495
Εφαρμογή 1....497
Δουλεύοντας με τα κουμπιά, η κλάση Button....509
Δουλεύοντας με κουτιά κειμένου, η κλάση TextBox....513
Δουλεύοντας με κουτιά επιλογής, η κλάση CheckBox....522
Δουλεύοντας με πλαίσια, η κλάση Frame....526
Δουλεύοντας με ραδιοπλήκτρα, η κλάση RadioButton....527
Δουλεύοντας με ετικέτες, η κλάση Label....529
Ιδιότητες (Properties)....530
Δουλεύοντας με περιοχές κειμένου, η κλάση TextArea....533
Δουλεύοντας με κουμπιά εναλλαγής, η κλάση ToggleButton....540
Δουλεύοντας με ετικέτες κειμένου, η κλάση TextLabel....544
Δουλεύοντας με κουτιά εικόνων, η κλάση PictureBox....546
Δουλεύοντας με εικόνες, η κλάση Picture....547
Δουλεύοντας με εικόνες, η κλάση Image....548
Δουλεύοντας με εικόνες, η κλάση DrawingArea....568
Δουλεύοντας με εικόνες, η κλάση Draw.....570
Δουλεύοντας με εικόνες, η κλάση SvgImage....581
Δουλεύοντας με λίστες, η κλάση ListView.....583
Δουλεύοντας με κουτιά από λίστες, η κλάση ListBox....588
Δουλεύοντας με αναδυόμενα κουτιά επιλογών, η κλάση ComboBox....593
Δουλεύοντας με λίστες πλέγματος, η κλάση GridView....597
Δουλεύοντας με λίστες δένδρου, η κλάση TreeView....603
Δουλεύοντας με λίστες σε στήλες, η κλάση ColumnView....609

Δουλεύοντας με γραμμές προόδου, η κλάση ProgressBar.....613
Δουλεύοντας με γραμμές κύλισης, η κλάση ScrollBar.....617
Δουλεύοντας με περιοχές κύλισης, η κλάση ScrollView.....620
Δουλεύοντας με καρτέλες, η κλάση TabStrip.....622
Δουλεύοντας με γραμμές εργαλείων, η κλάση ToolButton.....625
Δουλεύοντας με εικονίδια στην περιοχή ειδοποιήσεων, η κλάση TrayIcon.....628
Δουλεύοντας με την κλάση Application.....632
Η ιδιότητα Deamon.....637
Δουλεύοντας με τον εκτυπωτή, η κλάση Printer.....638
Το συμβάν Begin.....638
Το συμβάν Paginate.....638
Το συμβάν Draw.....638
The End event.....638
Πώς να σχεδιάσετε μια σελίδα.....639
Διαστάσεις της σελίδας.....639
Περιθώρια.....639
Απόλυτες συντεταγμένες.....639
Μέγεθος γραμμάτων.....639
Κεφάλαιο 2ο – Εμπλουτισμένα συστατικά διερμηνευτή (Interpreter enhancements).....648
gb.pcre.....648
Δουλεύοντας με τις Regular Expressions, η κλάση Regexp.....648
Δημιουργώντας patterns για τις Regular Expressions.....652
Εύρεση εκτυπώσιμων χαρακτήρων.....653
Επαναλήψεις.....653
Κλάσεις χαρακτήρων.....655
Σύνθετες κανονικές εκφράσεις.....656
gb.settings.....658
Αρχεία ρυθμίσεων.....658
Σύνταξη των αρχείων ρυθμίσεων.....658
Παράδειγμα σύνταξης αρχείου ρυθμίσεων:.....659
Κλάσεις.....659
Δουλεύοντας με τις ρυθμίσεις, η κλάση Settings.....659
gb.signal.....662
Κλάσεις.....663
Δουλεύοντας με τα σήματα του πυρήνα, η κλάση Signal.....663
Η εικονική κλάση .SignalHandler.....664
gb.eval.....665
Κλάση Expression.....666
Κλάση Highlight.....673
Κλάση System.....679
gb.args.....687
Κλάση Args.....687
Κεφάλαιο 3ο – Προχωρημένες τεχνικές GUI (Graphical User Interface).....689
gb.form.dialog.....689
Κλάση Dialog.....689
gb.form.mdi.....695

Κλάση Workspace....695
gb.form.terminal....697
Κλάση TerminalView....698
Κεφάλαιο 4ο – Βάσεις Δεδομένων (Databases)....701
gb.db....701
Πώς να ανοίξετε μια σύνδεση SQLite και να τη χρησιμοποιήσετε....702
Συνολική διαδικασία....702
Απαιτήσεις....702
Κεφάλαιο 5ο – Εξαρτήματα (Components)....707
Κεφάλαιο 6ο – Συμπεράσματα....707

Εισαγωγή

Το βιβλίο αυτό περιγράφει την γλώσσα προγραμματισμού Gambas.
Η Gambas είναι ένα πλήρες περιβάλλον ανάπτυξης εφαρμογών βασισμένη σε ένα διερμηνευτή (interpreter) της γλώσσας προγραμματισμού BASIC με επεκτάσεις αντικειμένων όμοια με την *Visual Basic*™ αλλά δεν είναι κλώνος της Visual Basic™.

Με την Gambas μπορείτε εύκολα και απλά να σχεδιάσετε το οπτικό μέρος της εφαρμογής, το οποίο θα αποτελείται από “φόρμες” οι οποίες θα έχουν κουμπιά που θα εκτελούν εντολές ή άλλα αντικείμενα που εσείς θα προγραμματίζετε την συμπεριφορά τους κατά την ροή του προγράμματος.

Η Γραφική Διεπαφή του Προγραμματιστή ή αλλιώς Graphic User Interface (GUI) αποτελείται από πλήθος αντικειμένων όπως παράθυρα, κουμπιά, ραδιοπλήκτρα, ετικέτες, εικόνες, μπάρες προόδου, κουτιά τσεκαρίσματος, ολισθαίνοντες ρυθμιστές και πολλά άλλα αντικείμενα (objects) τα οποία είναι πλήρως παραμετροποιήσιμα και μπορείτε να καθορίσετε την λειτουργία τους μέσα σε ένα πρόγραμμα.

Όλα αυτά τα αντικείμενα τα προσφέρουν οι βιβλιοθήκες της QT και GTK+ με τις οποίες η Gambas συνεργάζεται.

Άλλα χαρακτηριστικά της Gambas είναι η πρόσβαση σε βάσεις δεδομένων όπως η MySQL, PostgreSQL, SQLITE.

Αλλά οι δυνατότητες της Gambas δεν περιορίζεται μόνο σε αυτά τα στοιχεία που αναφέρθηκαν. Σε αυτό το βιβλίο θα περιγραφεί πλήρως η γλώσσα Gambas, οι εντολές της, οι συναρτήσεις, οι δομές ελέγχου κτλ. Το βιβλίο αυτό περιέχει παραδείγματα βασισμένα τόσο στις εντολές της γλώσσας όσο και στις κλάσεις που αυτή διαχειρίζεται. Οι κλάσεις της Gambas προσφέρουν τα γραφικά αντικείμενα ελέγχου για τις εφαρμογές όπως φόρμες, κουμπιά, πλαίσια κειμένου κτλ. Ωστόσο δεν θα γίνει αναλυτική παρουσίαση των κλάσεων σε αυτό το βιβλίο. Η μελέτη της γλώσσας Gambas θα πρέπει να γίνει σε συνδυασμό με την online wiki-εγκυκλοπαίδεια που αποτελεί την περιγραφή των κλάσεων με όλες τις ιδιότητες, συμβάντα, μεθόδους και σταθερές τους Σε αυτό το βιβλίο θα αναφερθούν σχεδόν όλα τα χαρακτηριστικά της γλώσσας Gambas εκτός από την περιγραφή των κλάσεων. Αυτό το βιβλίο φιλοδοξεί να αποτελέσει ένα πλήρη οδηγό για ένα μεγάλο μέρος της γλώσσας Gambas το οποίο όμως θα απευθύνεται στον αρχάριο προγραμματιστή ή στον προγραμματιστή που τώρα ξεκινά ή τον σπουδαστή που θέλει να μάθει εύκολα προγραμματισμό με μια γλώσσα BASIC στο λειτουργικό σύστημα LINUX.
Για την μελέτη αυτού του βιβλίου ο αναγνώστης θα πρέπει να γνωρίζει τις βασικές έννοιες της πληροφορικής σχετικά με την επιστήμη αυτή.

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.

Όταν θα παρουσιάζεται αυτό το εικονίδιο τότε σημαίνει ότι αυτό που ακολουθεί είναι κώδικας που γράφεται στον επεξεργαστή κειμένου του περιβάλλοντος ανάπτυξης εφαρμογών της Gambas.
Το κείμενο που είναι κώδικας επίσης θα έχει ιδιαίτερη μορφοποίηση με έγχρωμη γραφή και έντονη και πλάγια γραφή.

Παράδειγμα:

```
PUBLIC SUB btnDeleteDublicates_Click()

DIM iCounter AS Integer = 0
DIM sCommand AS String
DIM iTotalFiles AS Integer = 0

  IF lswDublicates.Count > 0 THEN
    'Delete files
    iTotalFiles = lswDublicates.Count
    ....
    lswDublicates.Clear

  ENDIF
END
```

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

Παράδειγμα:

```
user@ubuntu-desktop:~$ ls
Desktop    dwhelper        foremost  Pictures    StreamRipStar  Videos
Documents  examples.desktop  lmms     Public      Templates      vmware
Downloads  Folding         Music     rosegarden  testgrep
```

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 Gambas.

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 ίδιο 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 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.

Προγράμματα

Με τον όρο "Πρόγραμμα" ορίζουμε δύο πράγματα.

1. Τον πηγαίο κώδικα ο οποίος γράφεται από τον προγραμματιστή στον επεξεργαστή κειμένου του περιβάλλοντος ανάπτυξης εφαρμογών (IDE, Integrated design environtment)
2. Το τελικό εκτελέσιμο λογισμικό που χρησιμοποιεί ο χρήστης.

Διαδικαστικός, Δομημένος και Αντικειμενοστραφής προγραμματισμός.

Τα πρώτα προγράμματα ήταν μια σειρά εντολών που ενεργούσαν πάνω σε δεδομένα όπως μεταβλητές. Η σειρά εκτέλεσης ήταν από πάνω προς τα κάτω καθώς ο διερμηνευτής διάβαζε με την σειρά τις εντολές από τον πηγαίο κώδικα. Τα δεδομένα ήταν ξεχωριστά από τις συναρτήσεις και τις εντολές και ο προγραμματιστής για να εξετάσει το πρόγραμμα κατά την εκτέλεση του παρακολουθούσε την αλλαγή των δεδομένων και ποιες συναρτήσεις αλλάζουν ποια δεδομένα.

Αυτός είναι ο ***Διαδικαστικός προγραμματισμός***.

Στον Διαδικαστικό προγραμματισμό ο προγραμματιστής για να αλλάξει την ροή του προγράμματος είχε την εντολή **GOTO** (πήγαινε) η οποία τοποθετούσε την εκτέλεση του προγράμματος σε όποιο σημείο της ροής ήθελε ο προγραμματιστής.
Αυτού του τύπου όμως ο προγραμματισμός συχνά οδηγούσε σε λάθη και σε μπερδεμένη ροή της εκτέλεσης του προγράμματος. Ένας πολύπλοκος κώδικας με πολλές **GOTO** συχνά αποκαλούνταν "κώδικας σπαγκέτι" για να δείξει το όλο μπερδεμένο τρόπο γραφής του προγράμματος.

Έτσι στην συνέχεια ανακαλύφθηκε ένα νέο είδος προγραμματισμού ο ***Δομημένος προγραμματισμός***.

Στον δομημένο προγραμματισμό επικρατεί η φιλοσοφία του "διαίρει και βασίλευε". Όλο το πρόγραμμα διαιρείται σε μικρές ενέργειες που κάθε μία εκτελεί συγκεκριμένη εργασία η οποία είναι απλή και κατανοητή. Η κάθε εργασία όταν τελειώνει επιστρέφει την ροή του προγράμματος στο σημείο από όπου ξεκίνησε. Έτσι υπάρχει μία κύρια ροή προγράμματος (main) που καλεί όποτε χρειαστεί τις ανάλογες ενέργειες που είναι γραμμένες χωριστά από την κύρια ροή του προγράμματος.

Αργότερα η διεπαφή των υπολογιστών έγινε πιο διαλογική προς τον προγραμματιστή. Από εκεί που ο υπολογιστής ρωτούσε σε σειρά για το τι να κάνει, τώρα ο προγραμματιστής χρησιμοποιούσε κουμπιά, φόρμες και παράθυρα τα οποία πατούσε, άνοιγε, έκλεινε και κάθε

ένα από αυτά εκτελούσε μια ενέργεια.
Κάθε στοιχείο της διεπαφής ήταν ένα αντικείμενο (object) που είχε ιδιότητες (properties) όπως χρώμα, κείμενο, όνομα κτλ αλλά δεχόταν και συμβάντα (events) όπως να του κάνουν μονό ή διπλό κλικ, να δέχεται το πάτημα πλήκτρων από το πληκτρολόγιο κτλ.
Κάθε φορά που άλλαζε μια ιδιότητα ή πραγματοποιούνταν ένα συμβάν το αντικείμενο μπορούσε να εκτελέσει μια μέθοδο (method) η οποία ήταν μια σειρά εντολών.

Για παράδειγμα, "αν γίνει μονό κλικ στο κουμπί όρισε το κείμενο του αντικειμένου σε “Μου έκανες κλικ”

Το κουμπί είναι ένα αντικείμενο (object) π.χ. Button1
Το κλικ είναι ένα συμβάν (**event**) π.χ. Button1_Click
Το κείμενο του κουμπιού είναι μία ιδιότητα (**property**) π.χ. Button1.Text
Και το όρισε το κείμενο μια μέθοδος (method) π.χ. Button1.SetText ή
Button1.Text = “Μου έκανες κλικ”

Όπως βλέπετε για να προσπελαστούν οι ιδιότητες, τα συμβάντα και οι μέθοδοι του αντικειμένου ο προγραμματιστής πρέπει να γράψει το όνομα του αντικειμένου μετά μια τελεία (.) και το όνομα της ιδιότητας, μεθόδου ή συμβάντος.

Με αυτόν τον τρόπο ο προγραμματιστής διαχειρίζεται τα αντικείμενα και ό,τι αυτά προσφέρουν.

Αυτός ο προγραμματισμός είναι ο ***Αντικειμενοστραφής προγραμματισμός***.

Η Gambas είναι μια Αντικειμενοστραφής γλώσσα προγραμματισμού που όχι μόνο χρησιμοποιείται ήδη υπάρχοντα αντικείμενα αλλά σας δίνει την δυνατότητα να φτιάξετε και δικά σας.

Στον Αντικειμενοστραφή προγραμματισμό ο προγραμματιστής που χρησιμοποιεί τα αντικείμενα δεν τον νοιάζει πώς λειτουργεί το αντικείμενο, με ποιο τρόπο αλλάζει μια ιδιότητα του, ποιος κώδικας τρέχει πίσω από μια μέθοδο ή από ένα συμβάν. Το μόνο που τον νοιάζει είναι πώς να χρησιμοποιεί τα αντικείμενα για να φτιάχνει προγράμματα.

Αυτό το χαρακτηριστικό των αντικειμένων να είναι αυτόνομα ονομάζεται "εγκλεισμός" (encapsulation).

Με τον εγκλεισμό πετυχαίνουμε την απόκρυψη δεδομένων (Data hiding).
Δεν μας νοιάζει πώς λειτουργεί, για παράδειγμα ο συμπιεστής του ψυγείου, εμείς θέλουμε να χρησιμοποιήσουμε το ψυγείο. Το ίδιο όπως προαναφέρθηκε γίνεται και με τα αντικείμενα στον Αντικειμενοστραφή προγραμματισμό.

Μια άλλη έννοια που πρέπει να γνωρίζεται είναι η "κληρονομικότητα" (inheritance).

Όταν θα πάτε να κατασκευάσετε ένα δικό σας αντικείμενο μπορείτε είτε να το προγραμματίσετε από την αρχή ή να βασιστείτε σε ένα ήδη υπάρχον και να του δώσετε νέα

χαρακτηριστικά.
Στην δεύτερη αυτή περίπτωση το νέο αντικείμενο θα απορρέει από το πρώτο και θα κληρονομήσει κάποιες ιδιότητες, συμβάντα και μεθόδους από αυτό. Το χαρακτηριστικό αυτό λέγεται κληρονομικότητα.

Εισαγωγή στην Gambas

Η Gambas είναι μια γλώσσα προγραμματισμού βασισμένη στην BASIC δημιουργημένη από τον Γάλλο Benoît Minisini.

Το project στοχεύει στο να αναπτυχθεί μια γλώσσα προγραμματισμού που να δημιουργεί γραφικές εφαρμογές με παράθυρα, κουμπιά και άλλα αντικείμενα ώστε να έχουμε ένα πλήρες περιβάλλον ανάπτυξης εφαρμογών όπως στην *Visual Basic*™ αλλά για το LINUX.

Το μεγάλο πλήθος λαθών και η ασυνέπεια που χαρακτηρίζει την *Visual Basic* ήταν η κινητήρια δύναμη που ώθησε τον Minisini να αναπτύξει την γλώσσα αυτή.

Φαίνεται πως η Microsoft αντιλήφθηκε την φτωχή ποιότητα της γλώσσας καθώς η VB .Net δεν είναι συμβατή με τις προηγούμενες εκδόσεις της *Visual Basic*. Ο Minisini πιστεύει ότι η Microsoft εγκατέλειψε τον πηγαίο κώδικα της *Visual Basic*. Και ανέπτυξε το νέο project VB .Net που η σύνταξη του μοιάζει με την παλιά *Visual Basic*.

Η ***Gambas*** δεν προσπαθεί να είναι συμβατή με την *Visual Basic* και δεν θα γίνει ποτέ.

Ο Minisini πήρε από την *Visual Basic* ότι θεώρησε χρήσιμο και ανέπτυξε μια καλύτερη γλώσσα BASIC.

Η ***Gambas*** αποτελείται από τα εξής προγράμματα:

- Ένα μεταγλωττιστή (compiler).
- Ένα διερμηνευτή (interpreter).
- Έναν αρχειοθέτη (archiver).
- Ένα γραφικό περιβάλλον χρήστη. (GUI, graphical user interface).
- Ένα περιβάλλον ανάπτυξης (IDE, integrated development environment).

Το περιβάλλον ανάπτυξης είναι γραμμένο με την ίδια την Gambas έτσι ώστε να δειχθούν η δυνατότητες της γλώσσας.

Ποια είναι τα χαρακτηριστικά που η Gambas έχει πάρει από άλλες γλώσσες;

1. Ένα project της ***Gambas*** τοποθετείται σε ένα φάκελο. Ο αρχειοθέτης μετατρέπει το φάκελο του project σε ένα εκτελέσιμο αρχείο.
2. Μεταγλωττίζοντας ένα project απαιτεί μόνο την μεταγλώττιση των τροποποιημένων κλάσεων. Κάθε εξωτερική αναφορά σε μία κλάση καλείται δυναμικά στον χρόνο εκτέλεσης του προγράμματος.
3. Η ***Gambas*** έχει αρχιτεκτονική εξαρτημάτων λογισμικού (component architecture) η οποία επιτρέπει στην γλώσσα να επεκτείνεται με το να προστίθενται νέα εξαρτήματα λογισμικού (components). Ο κάθε ένας μπορεί να γράψει νέα εξαρτήματα λογισμικού ως κοινόχρηστες βιβλιοθήκες οι οποίες προσθέτουν νέες κλάσεις στον διερμηνευτή. Ένα εξάρτημα λογισμικού (Component) μπορεί να γραφτεί επίσης και στην ***Gambas***. Η αρχιτεκτονική των εξαρτημάτων λογισμικού έχουν τεκμηρίωση στην Wiki-Εγκυκλοπαίδεια (Wiki encyclopaedia) της Gambas.

4. Εξ ορισμού ο διερμηνευτής της ***Gambas*** είναι ένα πρόγραμμα τερματικού και όχι μια γραφική εφαρμογή. Η αρχιτεκτονική εξαρτημάτων χρησιμοποιείται για να εισάγει το γραφικό περιβάλλον σαν εξάρτημα λογισμικού και να το καταστήσει μέρος της γλώσσας.
5. Καθώς το γραφικό περιβάλλον του προγραμματιστή υλοποιείται ως ένα component, η ***Gambas*** θα είναι ανεξάρτητη από κάθε εργαλειοθήκη (toolkit) ! Έτσι θα είστε ικανοί να γράφετε προγράμματα και να διαλέγετε εργαλεία και αντικείμενα από την GTK+, Qt, κτλ.
6. Η υλοποίηση της γραφικής διεπαφής του προγραμματιστή έχει γίνει με την εργαλειοθήκη της Qt καθώς αυτή ήταν περισσότερο βολική στον Benoît Minisini. Συνιστάται ο προγραμματιστής να συμβουλευτεί και την τεκμηρίωση των εργαλειοθηκών της Qt και GTK+ για περισσότερη κατανόηση των contrlos που αυτές δίνουν.
7. Κάθε παράθυρο και πλαίσιο διαλόγου μπορεί να χρησιμοποιηθεί σαν αντικείμενο ελέγχου (control). Αυτό βέβαια στην Visual Basic δεν γινόταν χωρίς την χρήση ActiveX.
8. Τα projects της ***Gambas*** είναι εύκολα μεταφράσιμα σε κάθε γλώσσα.

Σε αυτό το βιβλίο χρησιμοποιήθηκε η έκδοση Gambas3 και το λειτουργικό σύστημα Debian 13 Trixie.

Για την εγκατάσταση της Gambas στο λειτουργικό σύστημα Linux δείτε την ιστοσελίδα της Gambas
http://gambas.sourceforge.net/en/main.html

Εικόνα 1: Το νέο λογότυπο της Gambas3

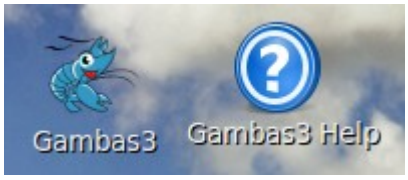
Μέρος 1ο, Το περιβάλλον ανάπτυξης

Κεφάλαιο 1ο - Η πρώτη μου εφαρμογή “Hello World!”

Αλλά ας δούμε τα πρώτα στάδια δημιουργίας ενός project.

Μπορείτε να ξεκινήσετε την Gambas γράφοντας στο τερματικό “gambas3” ή αν έχετε κάνει συντόμευση στην επιφάνεια εργασίας του εκτελέσιμου gambas3 να κάνετε διπλό κλικ σε αυτήν.

ή



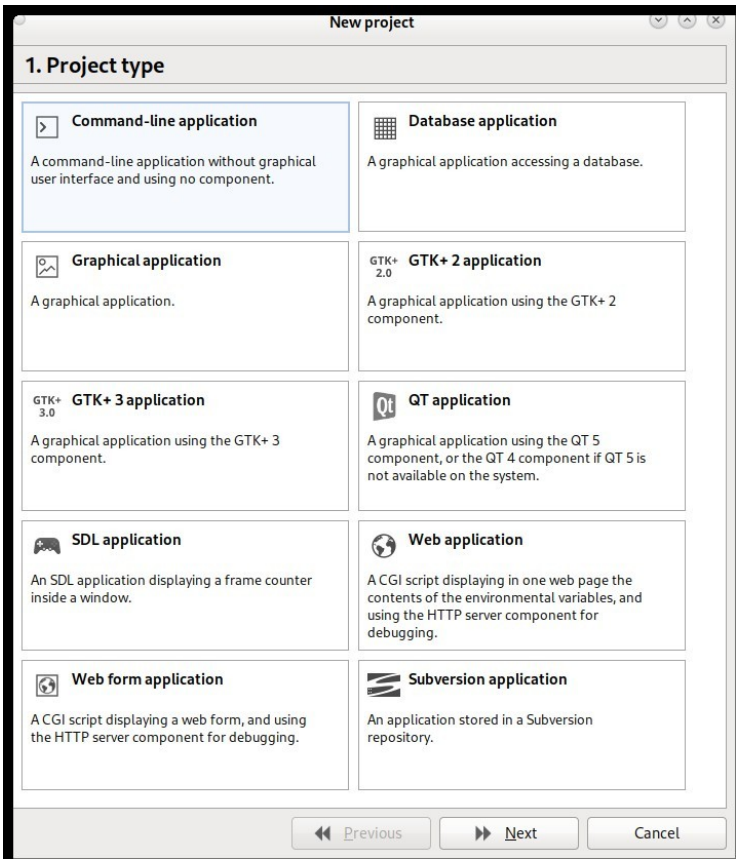
Εικόνα 2: Εικονίδια εκκίνησης της Gambas3 στην επιφάνεια εργασίας

Μετά την εκκίνηση εμφανίζεται το παράθυρο που σας καλωσορίζει και σας προτρέπει

1. να δημιουργήσετε ένα νέο project,
2. να ανοίξετε ένα υπάρχον project,
3. να ανοίξετε το τρέχον project
4. να ανοίξετε κάποιο παράδειγμα
5. να μεταβείτε στην συλλογή εφαρμογών της Gambas που ονομάζεται Software Farm
6. να ανοίξεται ένα αρχείο
7. να δείτε πληροφορίες σχετικά με την Gambas
8. και να κλείσετε την εφαρμογή

Εικόνα 3: Το παράθυρο καλωσορίσματος

Εμείς θα διαλέξουμε “**New** project” και το επόμενο παράθυρο μας δίνει επιλογές για τον τύπο του προγράμματος που θέλουμε να δημιουργήσουμε και τις ιδιότητες που θα έχει το project.



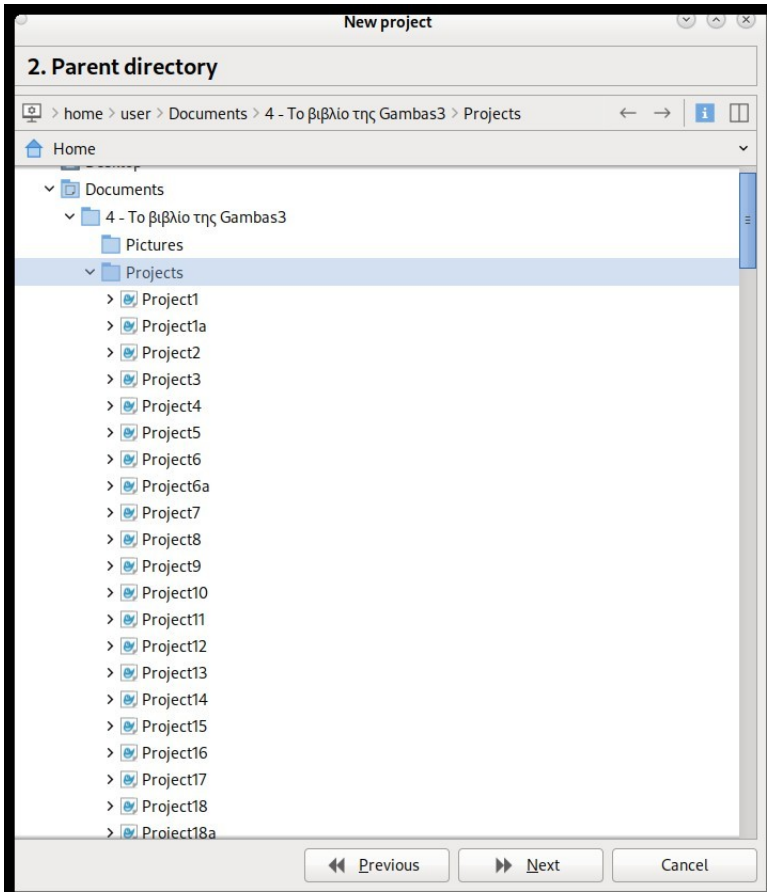
Εικόνα 4: Παράθυρο επιλογής project

Τα είδη των project που μπορούμε να δημιουργήσουμε είναι:

1. Μια εφαρμογή τερματικού
2. Μια εφαρμογή με βάσεις δεδομένων
3. Μια γραφική εφαρμογή γενικά
4. Μια γραφική εφαρμογή με την εργαλειοθήκη του GTK2+
5. Μια γραφική εφαρμογή με την εργαλειοθήκη του GTK3+
6. Μια γραφική εφαρμογή με την εργαλειοθήκη της Qt
7. Μια εφαρμογή SDL (Simple DirectMedia Layer)
8. Μια Web εφαρμογή
9. Μια Web form εφαρμογή
10. Μια εφαρμογή που θα αποθηκευτεί σε αποθετήριο Subversion

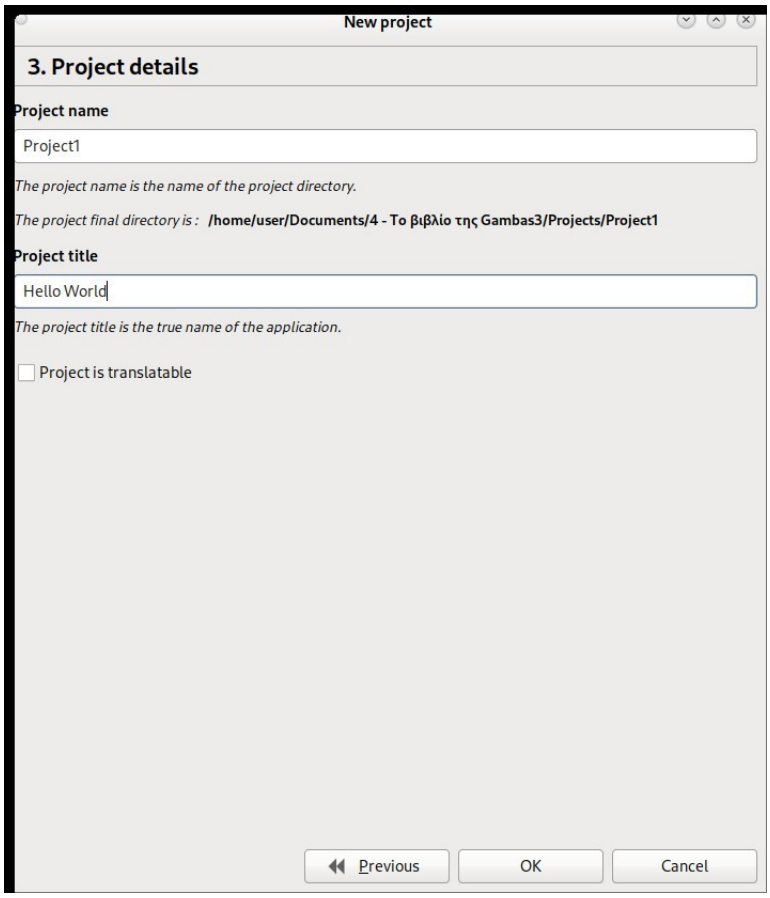
Εμείς επιλέγουμε “**Graphical Application**” και πατάμε επόμενο, “**Next**”.

Το επόμενο παράθυρο μας λέει να επιλέξουμε που θα βρίσκεται ο φάκελος του project.



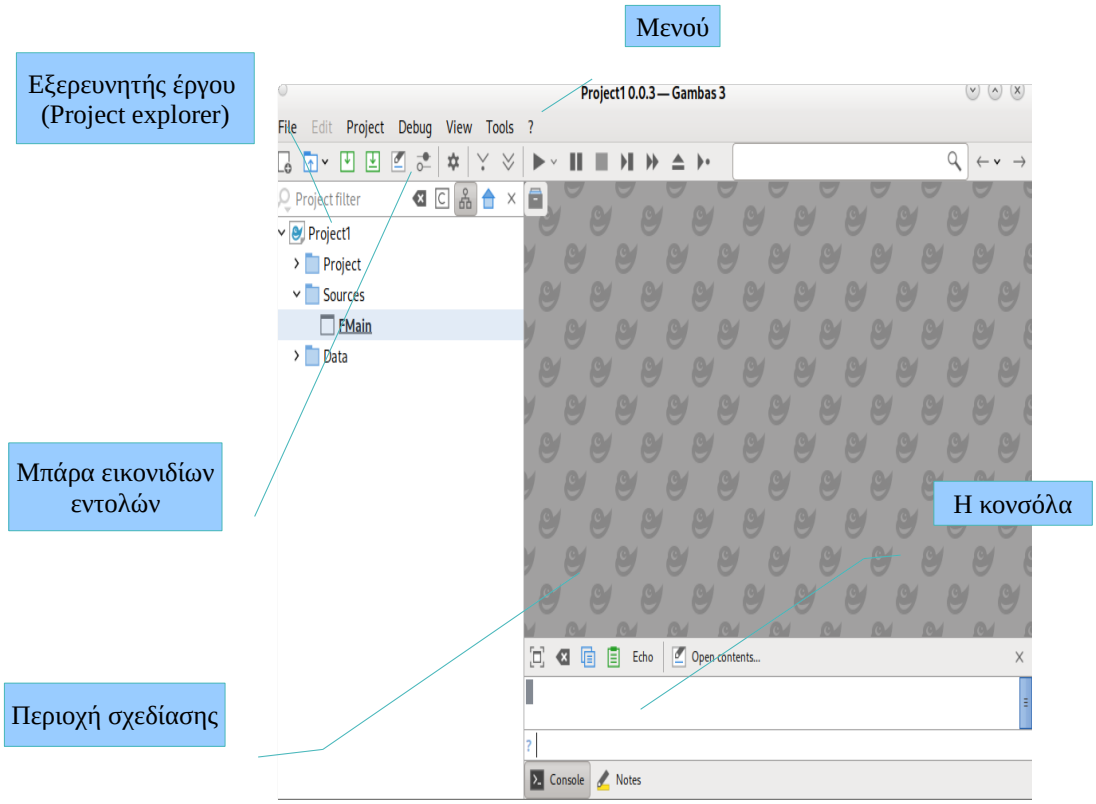
Εικόνα 5: Επιλογή φακέλου project

Το επόμενο παράθυρο συμπληρώνουμε το όνομα του Project και τον τίτλο του.
Σαν όνομα επιλέγουμε κείμενο που δεν έχει ειδικούς χαρακτήρες ή κενά, αλλιώς η Gambas θα μας βγάλει ένα μήνυμα λάθους.



Εικόνα 6: Τίτλος και όνομα προγράμματος

Πατώντας “**OK**” το νέο project δημιουργείται και μας εμφανίζεται το περιβάλλον ανάπτυξης εφαρμογών.



Εικόνα 7: Περιβάλλον ανάπτυξης εφαρμογών

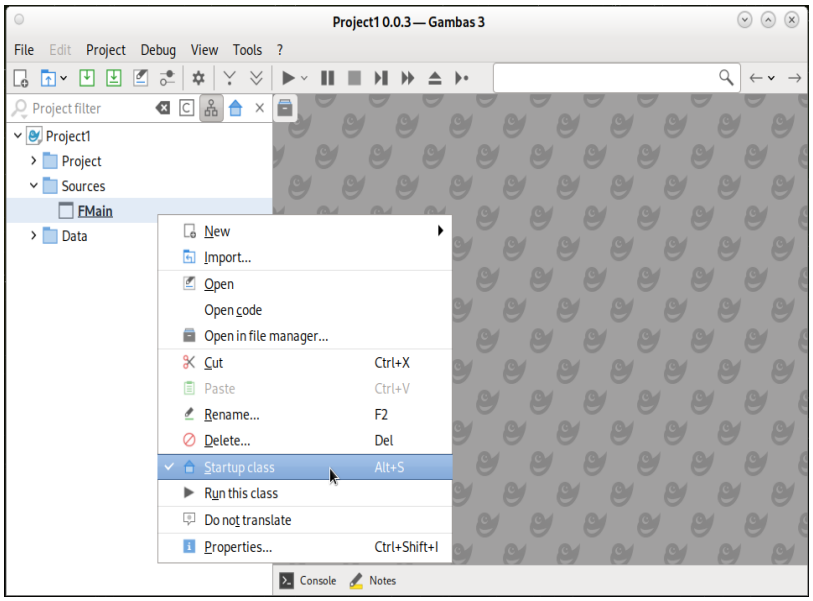
Στο πάνω μέρος υπάρχει το κλασικό μενού με τις επιλογές File (Αρχείο), Project (Έργο), Debug (Αποσφαλμάτωση), View (Προβολή), Tools (Εργαλεία), ? (Βοήθεια)

Αμέσως μετά υπάρχει μια μπάρα με εικονίδια εντολών από τις περισσότερο χρησιμοποιούμενες εντολές των μενού.

Μετά την περιοχή σχεδίασης στο αριστερό μέρος έχει τον εξερευνητή έργου όπου βλέπουμε σε δομή δένδρου τους φακέλους και τα αρχεία του project.
Όταν επιλέξουμε ένα αρχείο στο δεξί μέρος εμφανίζει το οπτικό του σχέδιο (design) ή τον πηγαίο κώδικα της κλάσης. Οι φάκελοι μπορεί να έχουν αρχεία Αρθρωμάτων (Modules), Κλάσεων (Classes), Φορμών (Forms), Δεδομένων (Data) όπως εικόνες, αρχείων ρυθμίσεων.

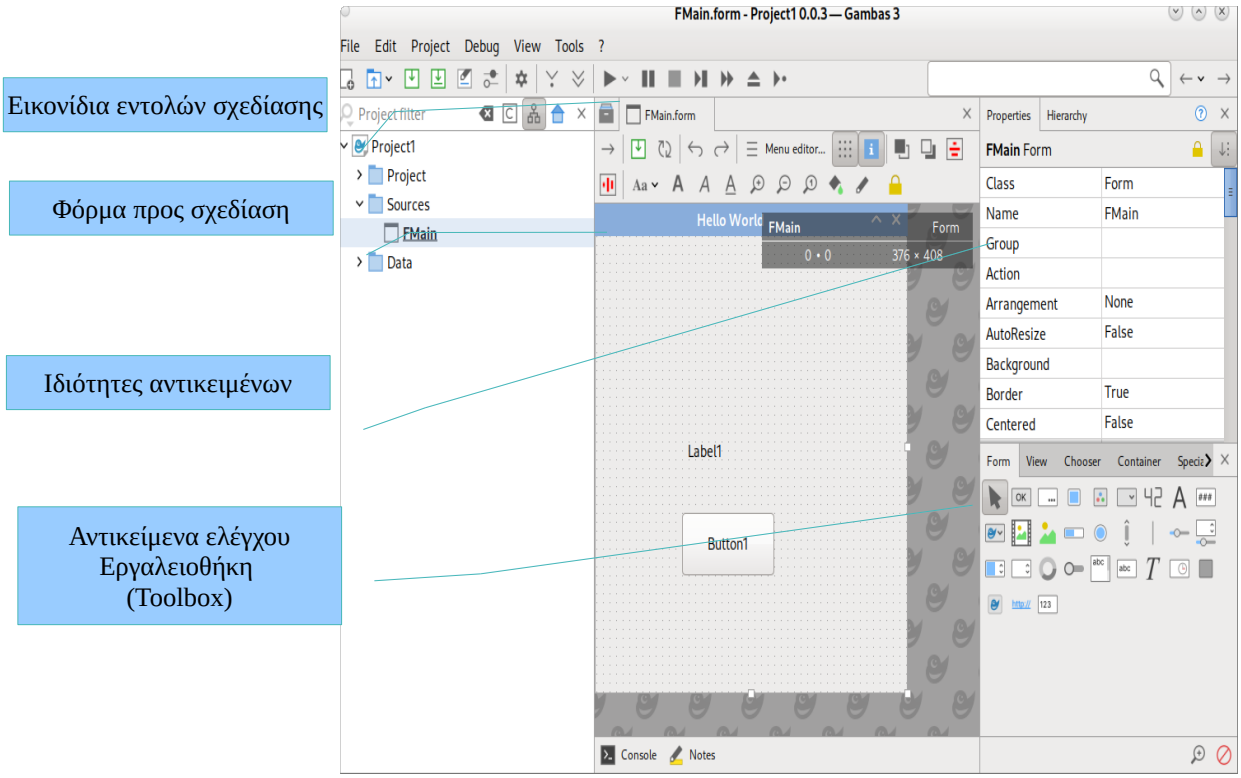
Στην εικόνα 7 βλέπουμε ότι με κάθε νέο project η Gambas έχει ήδη δημιουργήσει μια κύρια φόρμα από την οποία θα ξεκινήσει ο διερμηνευτής το πρόγραμμα.
Η φόρμα αυτή λέγεται FMain.

Η κλάση εκκίνησης εδώ είναι μια φόρμα, μπορεί όμως να είναι ακόμη και κλάση ή άρθρωμα (module). Για να ορίσουμε ποιο στοιχείο θα είναι η κλάση εκκίνησης κάνουμε δεξί κλικ στην κλάση που θέλουμε και επιλέγουμε “**Startup Class**”.



Εικόνα 8: Ορίζοντας την κλάση εκκίνησης

Με διπλό κλικ στην FMain εμφανίζεται η προβολή σχεδίασης της φόρμας.



Εικόνα 9: Η προβολή σχεδίασης της φόρμας

Η φόρμα αποτελείται από μία χρωματιστή επικεφαλίδα και από την περιοχή που θα τοποθετηθούν τα στοιχεία ελέγχου (contrlos) η οποία εδώ μας εμφανίζεται με τελείες οι οποίες παίζουν τον ρόλο στοίχισης των αντικειμένων ελέγχου στην φόρμα. Κάτω αριστερά ήδη φαίνονται μερικά αντικείμενα ελέγχου (controls), όπως κουμπιά, ετικέτες, κουτιά λίστας κτλ. Πάνω αριστερά υπάρχουν όλες οι ιδιότητες της φόρμας που μπορούμε να ορίσουμε. Οι ιδιότητες αυτές είναι για παράδειγμα, το κείμενο επικεφαλίδας, το χρώμα του κειμένου, το χρώμα του φόντου, το περίγραμμα της φόρμας κτλ

Ας εξηγήσουμε αναλυτικά λοιπόν το περιβάλλον εργασίας.

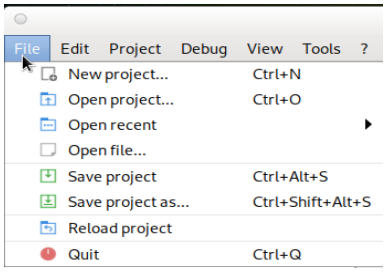
Κεφάλαιο 2ο – Μενού περιβάλλοντος ανάπτυξης εφαρμογών

Κύρια Μενού.

Τα κύρια μενού είναι τα εξής:

Αρχείο (File),
Έργο (Project),
Αποσφαλμάτωση (Debug),
Προβολή (View),
Εργαλεία (Tools),
Βοήθεια (?)

Αρχείο (File)



Εικόνα 10: Το μενού Αρχείο (File)

Στο μενού **Αρχείο (File)** έχουμε τις επιλογές:

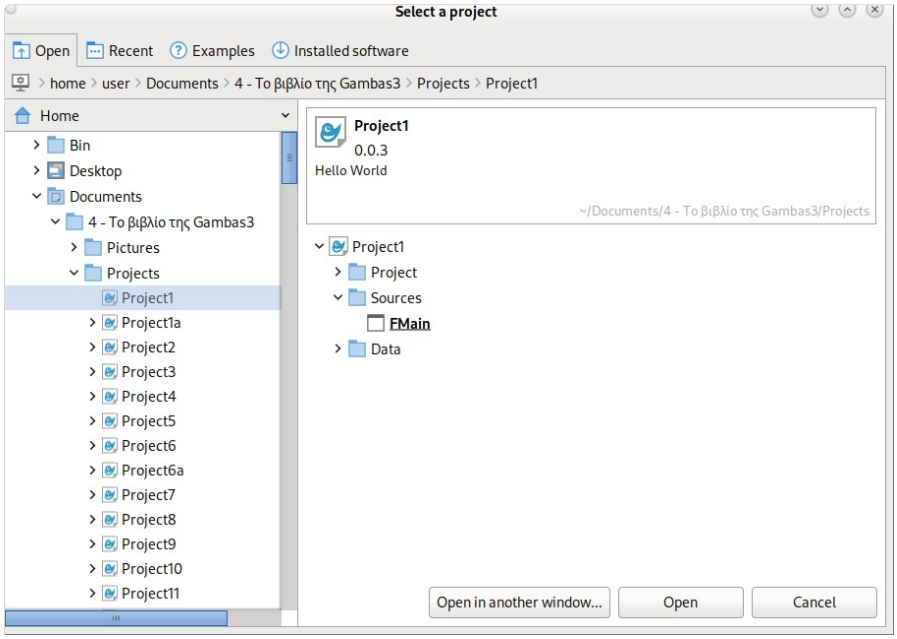
- Νέο έργο (***New project***), όπου δημιουργούμε ένα νέο έργο
- Άνοιγμα έργου (***Open project***), όπου ανοίγουμε ένα έργο προς εργασία
- Άνοιγμα πρόσφατου (***Open recent***), όπου μας εμφανίζονται τα πρόσφατα έργα και μπορούμε να τα ανοίξουμε
- Άνοιγμα αρχείου (***Open file***), όπου ανοίγουμε ένα μεμονωμένο αρχείο
- Αποθήκευση έργου (***Save project***), όπου αποθηκεύουμε όλες τις αλλαγές στην σχεδίαση και κώδικα που έχουμε κάνει στο έργο.
- Αποθήκευση έργου ως (***Save project as***), όπου αποθηκεύουμε ένα υπάρχων έργο σε νέο κατάλογο και με νέο όνομα. Έτσι έχουμε ένα διπλότυπο του αρχικού έργου.
- Επαναφόρτωση του έργου (**Reload project**)
- Έξοδος (***Quit***), όπου κλείνουμε το περιβάλλον ανάπτυξης εφαρμογών Gambas.

Τις φόρμες δημιουργίας νέου έργου τις έχουμε δει προηγουμένως οπότε δεν θα τις επαναλάβουμε εδώ.

Άνοιγμα έργου (*Open project*)

Όταν ο προγραμματιστής επιλέξει να ανοίξει ένα υπάρχων έργο τότε του εμφανίζεται ένα πλαίσιο διαλόγου επιλογής κάποιου έργου της Gambas.

Όταν ο προγραμματιστής περιηγηθεί στον κατάλογο όπου υπάρχει ένα έργο της Gambas τότε εμφανίζεται ένα εικονίδιο λογότυπου της Gambas όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 11: Το πλαίσιο διαλόγου ανοίγματος ενός έργου

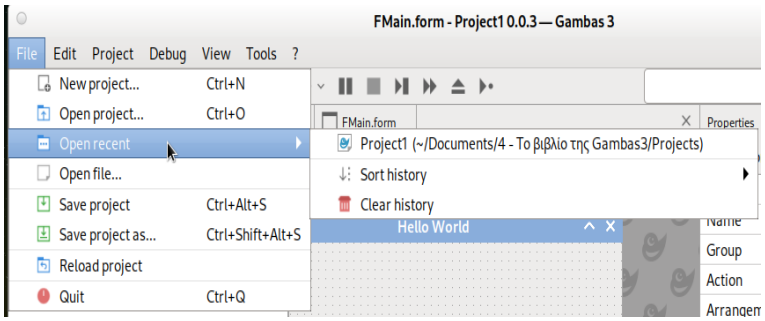
Το Project1 για παράδειγμα είναι ο φάκελος του έργου που όμως η Gambas το διαβάζει σαν ένα έργο (Project). Στο δεξί μέρος εμφανίζονται τα αρχεία που περιέχει το έργο.

Τα αρχεία ενός έργου είναι:

- Αρχεία .class, είναι τα αρχεία των κλάσεων που περιέχουν τον κώδικα σας.
- Αρχεία .form, είναι τα αρχεία του σχεδιασμού των φορμών. Τα αρχεία αυτά περιέχουν πληροφορίες για την σχεδίαση της φόρμας, τα αντικείμενα που έχει μια φόρμα κτλ.
- Αρχεία .module, είναι αρχεία μονάδων, είναι αρχεία κώδικα που ο οποίος μπορεί να αναπτυχθεί ξεχωριστά από τις κλάσεις και να συνδυαστεί με την εφαρμογή.
- Άλλα αρχεία όπως αρχεία εικόνων, εικονιδίων κτλ.

Άνοιγμα πρόσφατου (*Open recent*)

Το μενού ανοίγματος πρόσφατου έργου μας εμφανίζει τα πρόσφατα έργα και όταν επιλέξουμε κάποιο μας το ανοίγει απευθείας χωρίς να μεσολαβήσει πλαίσιο διαλόγου ανοίγματος.



Εικόνα 12: Άνοιγμα πρόσφατου έργου

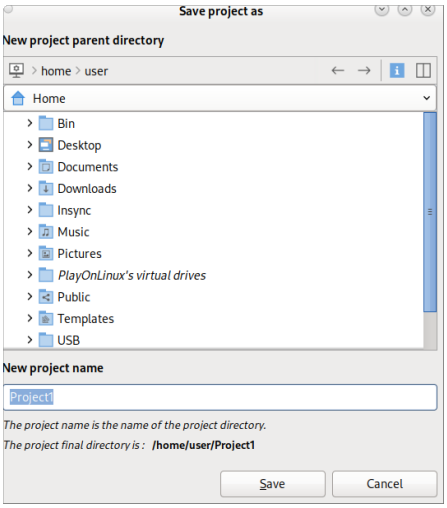
Υπάρχει η δυνατότητα να καθαρίσουμε το ιστορικό των πρόσφατων έργων, (***Clear history***) αλλά και να ταξινομήσουμε τα πρόσφατα έργα σύμφωνα με την διαδρομή τους στον σκληρό δίσκο (***Sort by path***).

Αποθήκευση έργου (*Save project*)

Αυτή η εντολή αποθηκεύει όλες τις αλλαγές που έχουν γίνει κατά την σχεδίαση και την συγγραφή κώδικα σε όλες τις φόρμες και όλες τις κλάσεις και μονάδες.

Αποθήκευση έργου ως (*Save project as*)

Η εντολή αυτή εμφανίζει ένα πλαίσιο διαλόγου αποθήκευσης του έργου σε μια νέα τοποθεσία. Το νέο έργο είναι ένα αντίγραφο του υπάρχοντος έργου.



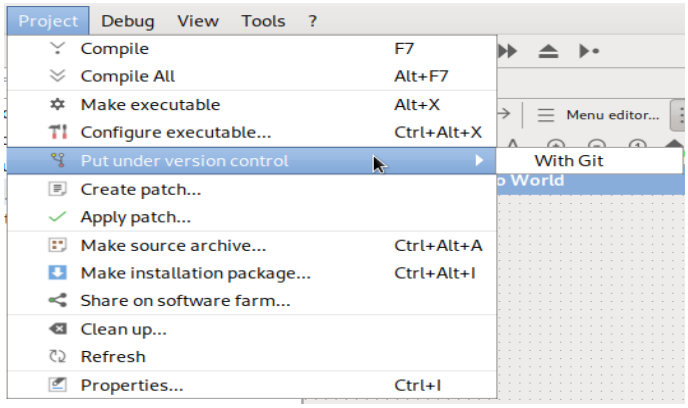
Εικόνα 13: Αποθήκευση ως...

Έξοδος (*Quit*)

Η εντολή αυτή τερματίζει το περιβάλλον ανάπτυξης εφαρμογών.

Έργο (Project)

Το μενού ***Έργο (Project)*** είναι το εξής:



Εικόνα 14: Μενού έργο

- Μεταγλώττιση (Compile)
- Μεταγλώττιση όλων (Compile All)
- Φτιάξε Εκτελέσιμο (Make Executable)
- Ρύθμιση εκτελέσιμου (Configure executable)

- Τοποθέτησε κάτω από Έλεγχο Έκδοοσης (Put under version control)
  - Με το GIT (With Git)
- Φτιάξε Ενημέρωση Κώδικα (Create Patch)
- Εφάρμωσε Ενημέρωση Κώδικα (Apply Patch)
- Πακέτο πηγαίου κώδικα (Make Source archive)
- Πακέτο εγκατάστασης (Make Installation package)
- Διαμοίρασε στην φάρμα λογισμικού (Share on software farm)
- Καθάρισε (Clean up)
- Ανανέωση (Refresh)
- Ιδιότητες (Properties)

Μεταγλώττισε (Compile)

Όταν επιλέγουμε να μεταγλωττίσουμε το έργο μας δίνεται η δυνατότητα να μεταγλωττίσουμε ένα αρχείο κλάσης με την εντολή “Μεταγλώττισε (Compile)”
Ο μεταγλωττιστής τοποθετεί τον μεταγλωττισμένο κώδικα στο κρυφό φάκελο .gambas που βρίσκεται στον φάκελο κάθε έργου. Τον μεταγλωττισμένο αυτό κώδικα θα τον χρησιμοποιήσει αργότερα ο διερμηνευτής.

Μεταγλώττισε όλα (Compile All)

Η εντολή “Μεταγλώττισε όλα (Compile All)” μεταγλωττίζει όλα τα αρχεία του έργου που περιέχουν κώδικα.

Φτιάξε Εκτελέσιμο (Make Executable)

Με την εντολή “Φτιάξε Εκτελέσιμο” δημιουργούμε το εκτελέσιμο αρχείο του έργου.
Ένα πλαίσιο διαλόγου μας εμφανίζεται για να επιλέξουμε την τοποθεσία που θα δημιουργηθεί το εκτελέσιμο.

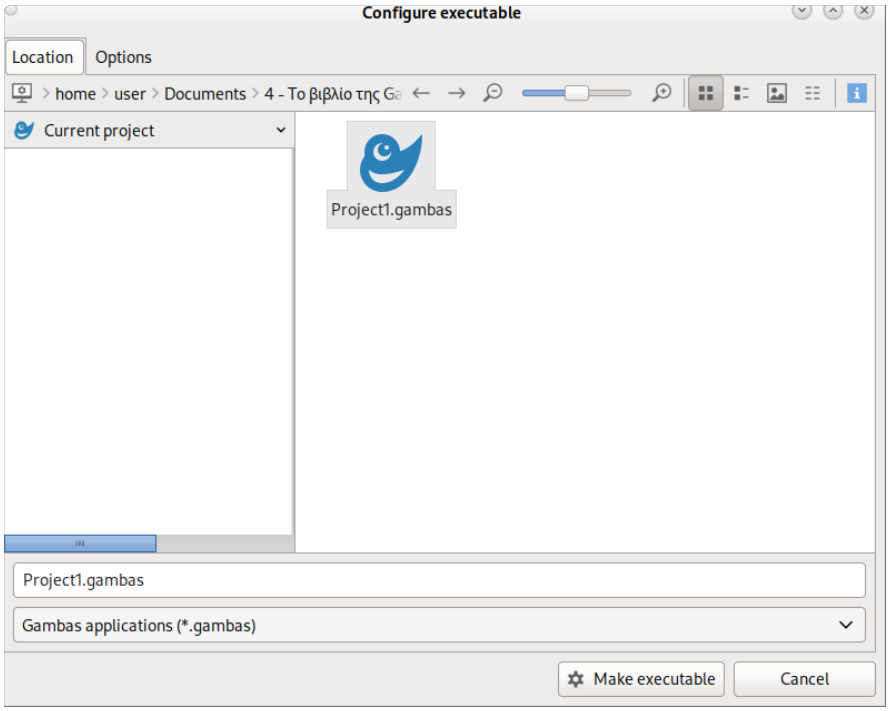
Το εκτελέσιμο αρχείο έχει επέκταση .gambas.

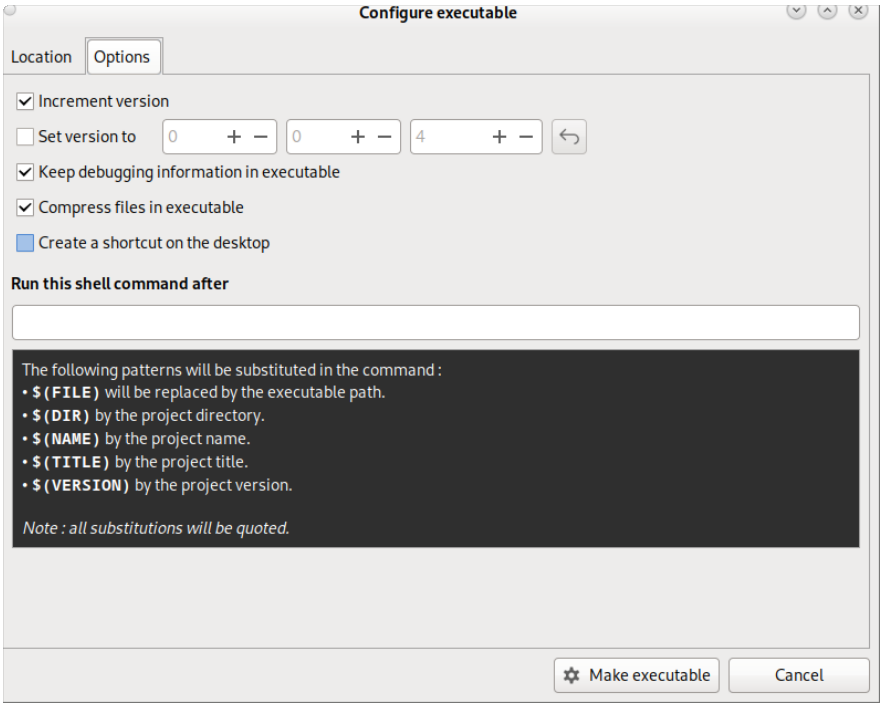
Οι παλιότερες εκδόσεις της Gambas δημιουργούσαν τεράστια εκτελέσιμα αν ο προγραμματιστής επέλεγε σαν τοποθεσία τον φάκελο του ίδιου του έργου. Το σφάλμα αυτό (huge executable bug) διορθώθηκε στις νεότερες εκδόσεις της Gambas.

Το εκτελέσιμο αρχείο θα πρέπει να τοποθετείται σε φάκελο εκτός του φακέλου του έργου.

Ρύθμιση εκτελέσιμου (Configure executable)

Με την επιλογή Ρύθμιση εκτελέσιμου εμφανίζεται ένα παράθυρο όπου μπορούμε να ορίσουμε την τοποθεσία που θα δημιουργηθεί το εκτελέσιμο, το όνομα του όπως και ρυθμίσεις για τον αριθμό έκδοσης του, όπως φαίνεται στις παρακάτω εικόνες.





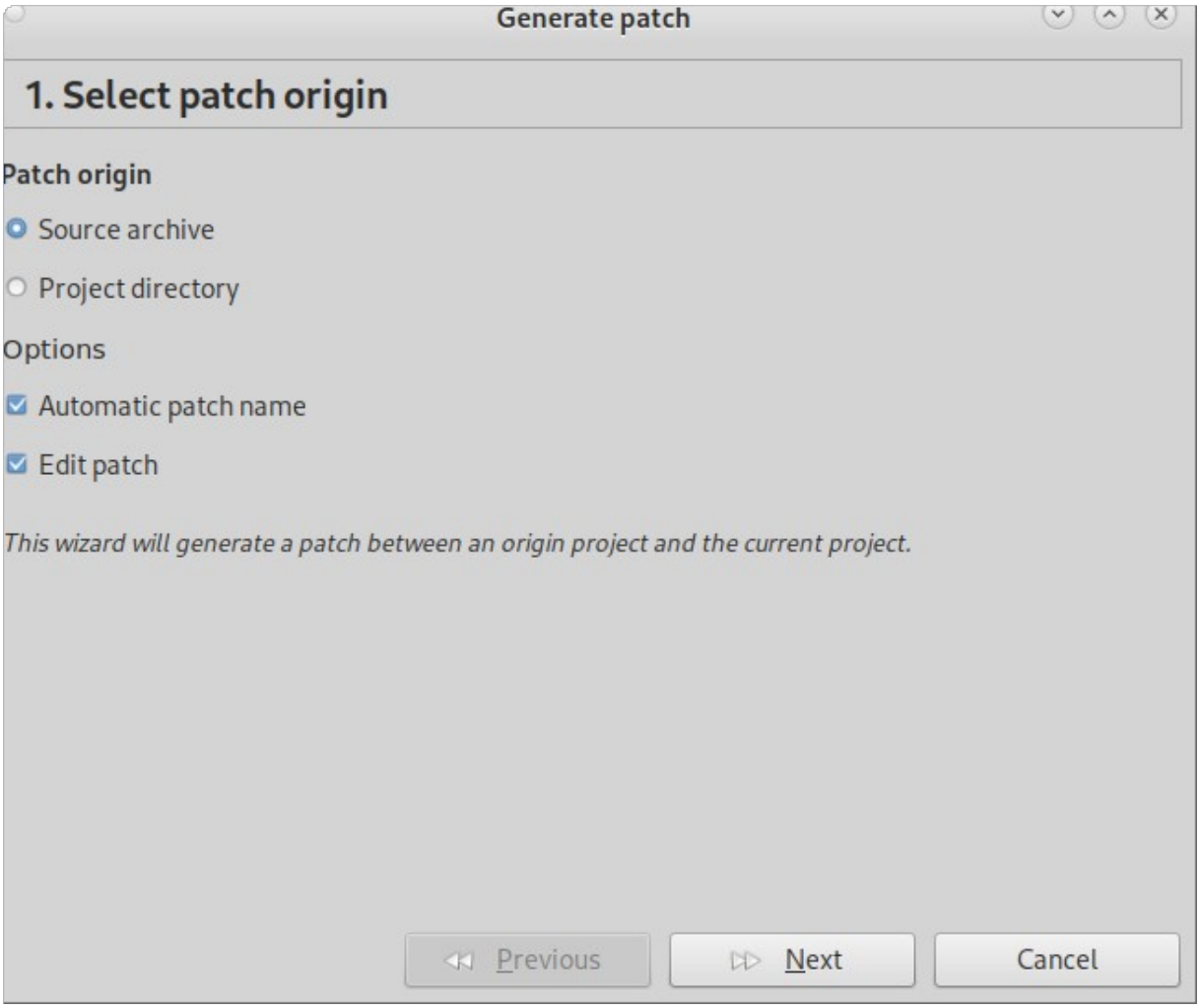
Τοποθέτησε κάτω από Έλεγχο Έκδοσης (Put under version control) - Με το GIT (With Git)

Με την επιλογή αυτή δημιουργείται ένας κρυφός φάκελος .git στην δομή του έργου με όλα τα αρχεία και ρυθμίσεις που τοποθετούν το έργο αυτό σε ένα αποθετήριο git (git repository).

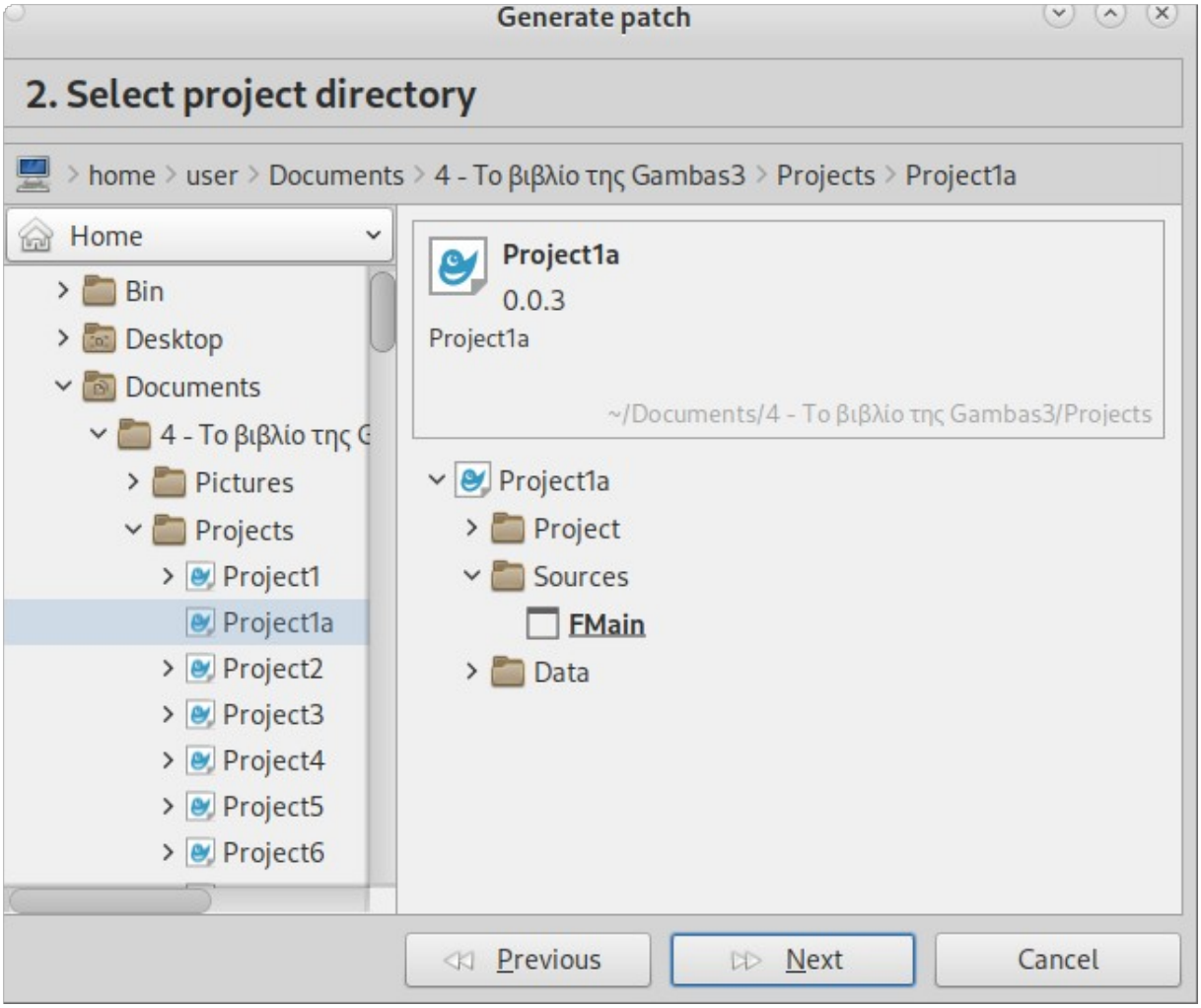
Φτιάξε Ενημέρωση Κώδικα (Create Patch)

Η επιλογή αυτή δημιουργεί μια Ενημέρωση Κώδικα (patch) μεταξύ ενός πηγαίου κώδικα προέλευσης (origin source) και του τρέχοντος έργου (current project).

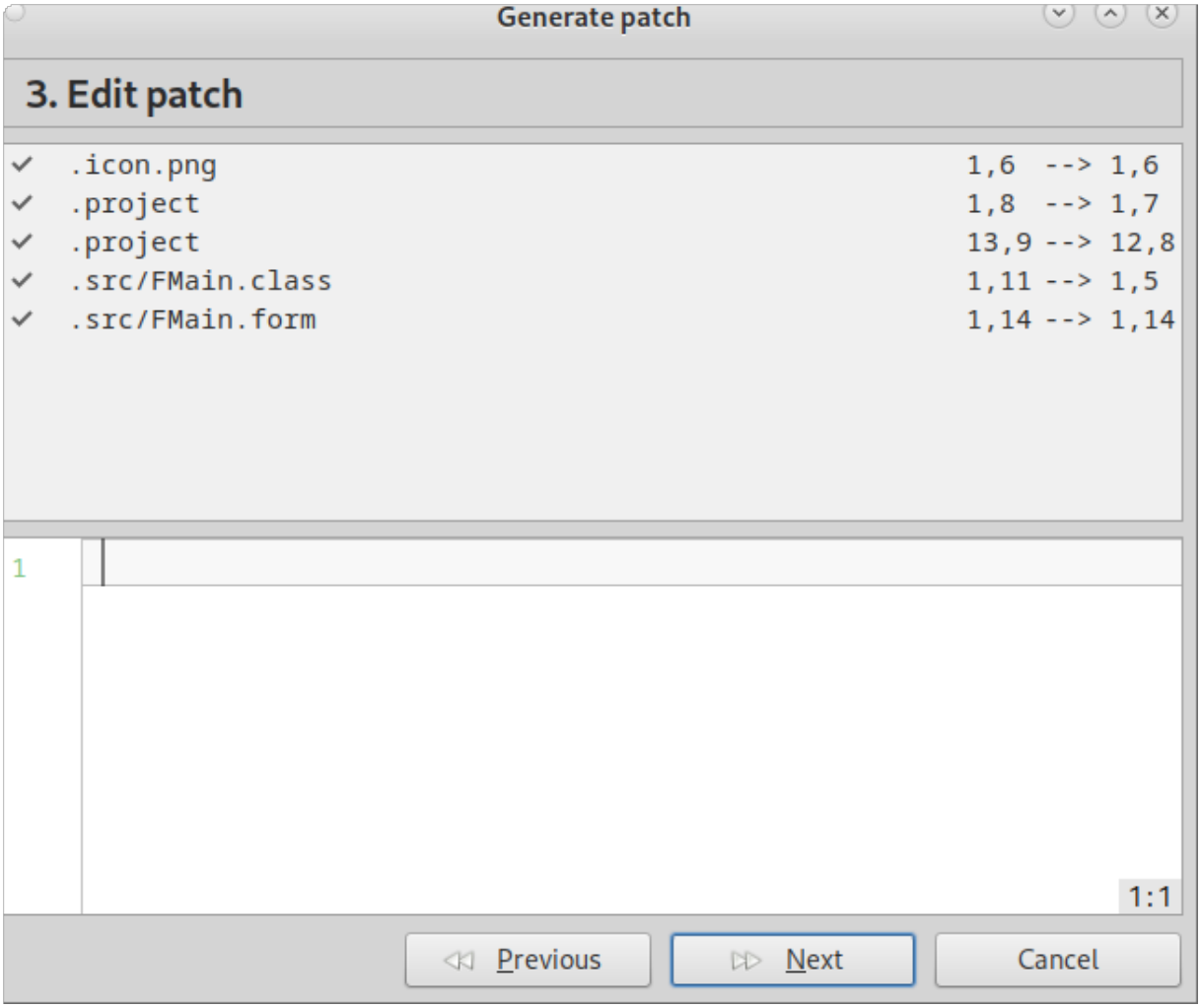
Αρχικά επιλέγουμε την προέλευση της Ενημέρωσης Κώδικα αν θα είναι από κάποιο συμπιεσμένο αρχείο πηγαίου κώδικα ή από την δομή κάποιου έργου.



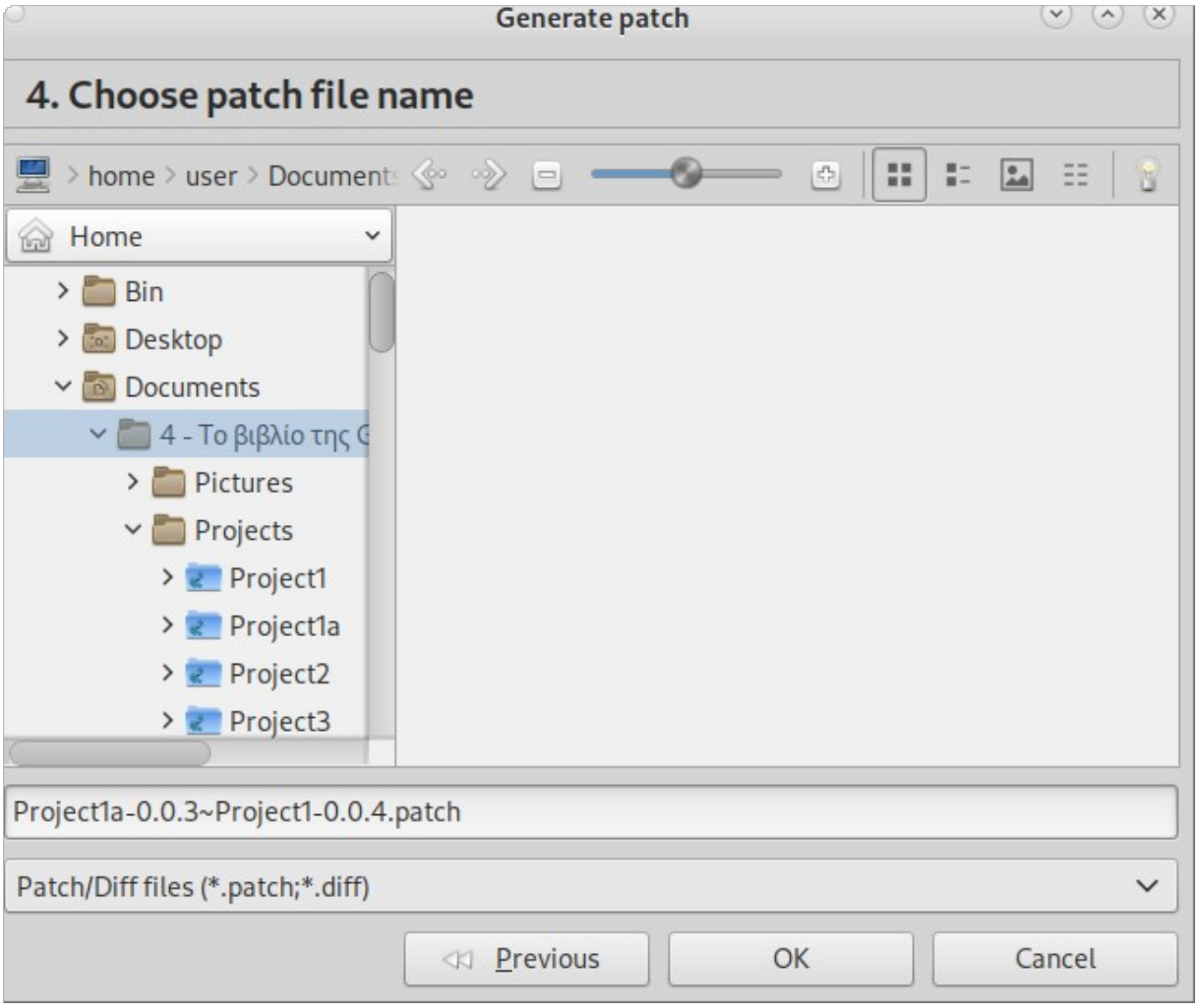
Στο συγκεκριμένο παράδειγμα επιλέξαμε η προέλευση να είναι κάποια δομή έργου με όνομα Project1a. Η πηγή της ενημέρωσης είναι που θα τροποποιηθεί και θα καταλήξει να μοιάζει με το τρέχον έργο.



Στο τρίτο βήμα επιλέγουμε ποια αρχεία θα τροποποιήσει το patch.



Στο 4ο βήμα επιλέγουμε την τοποθεσία που θα αποθηκευτεί η ενημέρωση κώδικα και το όνομα της.



Εφάρμωσε Ενημέρωση Κώδικα (Apply Patch)

Με την επιλογή αυτή διαλέγουμε την Ενημέρωση Κώδικα (patch) που θα εφαρμώσουμε στο τρέχον έργο.

Εάν δημιουργήσαμε μια Ενημέρωση Κώδικα (patch) μεταξύ 2 έργων για παράδειγμα μεταξύ του έργου Project1a και του έργου Project1 και θέλουμε να εφαρμόσουμε την ενημέρωση αυτή στον πηγαίο κώδικα του έργου Project1a τότε διαλέγουμε το patch, “Project1a-0.0.3~Project1-0.0.4.patch” καθώς έχουμε ανοιχτό το έργο Project1a.

Μετά την εφαρμογή της ενημέρωσης κώδικα το έργο Project1a θα έχει μετατραπεί ως όμοιο με το έργο Project1.

Φτιάξε Πακέτο πηγαίου κώδικα (Make Source archive)

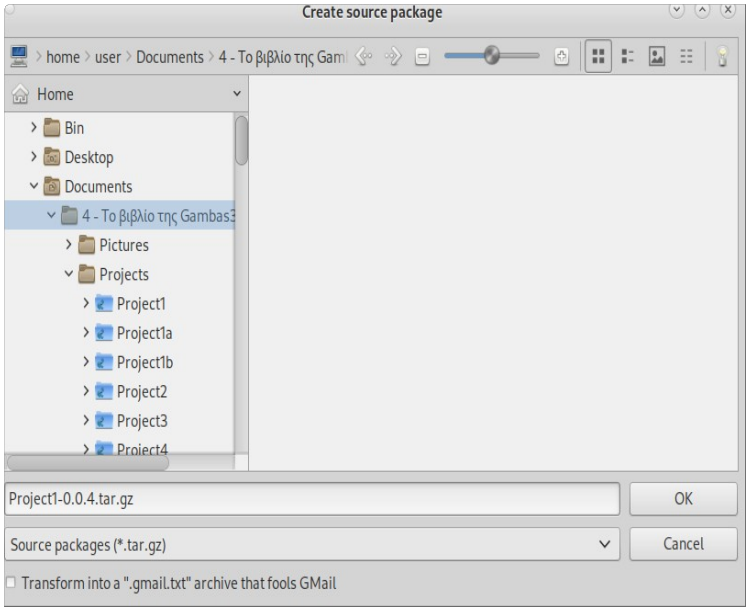
Με την εντολή “Φτιάξε Πακέτο πηγαίου κώδικα” δημιουργείτε ένα συμπιεσμένο αρχείο

*.tar.gz του έργου σας που περιέχει τον πηγαίο κώδικα.

Το όνομα του αρχείου είναι της μορφής **[Όνομα_έργου]-[Έκδοση]**.tar.gz

Για παράδειγμα:
Όνομα έργου = Project1
Έκδοση = 0.0.4
Πακέτο πηγαίου κώδικα = Project1-0.0.4.tar.gz

Η Gambas εμφανίζει ένα πλαίσιο διαλόγου για να ορίσετε που θα δημιουργηθεί το συμπιεσμένο αρχείο.



Εικόνα 15: Επιλογή της τοποθεσίας του πακέτου πηγαίου κώδικα

Εκτός από αρχεία *.tar.gz μπορείτε να δημιουργήσετε και αρχεία *.tar.bz2, *.tar.xz, *.zip.

Φτιάξε Πακέτο εγκατάστασης (Installation package)

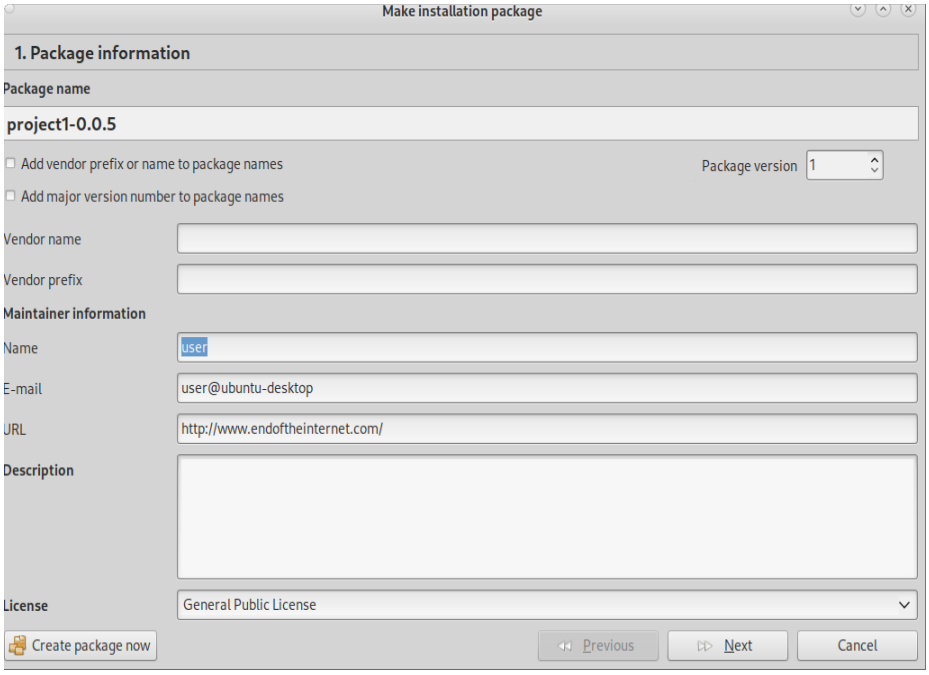
Η Gambas δίνει την δυνατότητα να δημιουργήσετε ένα πακέτο εγκατάστασης με την εντολή “Φτιάξε Πακέτο εγκατάστασης”.

Με την επιλογή της εντολής αυτής εμφανίζεται ένας οδηγός όπου σας καθοδηγεί στην δημιουργία του πακέτου εγκατάστασης.

Οδηγός δημιουργίας πακέτου εγκατάστασης.

Βήμα 1ο

Η πρώτη φόρμα που εμφανίζεται είναι η παρακάτω:



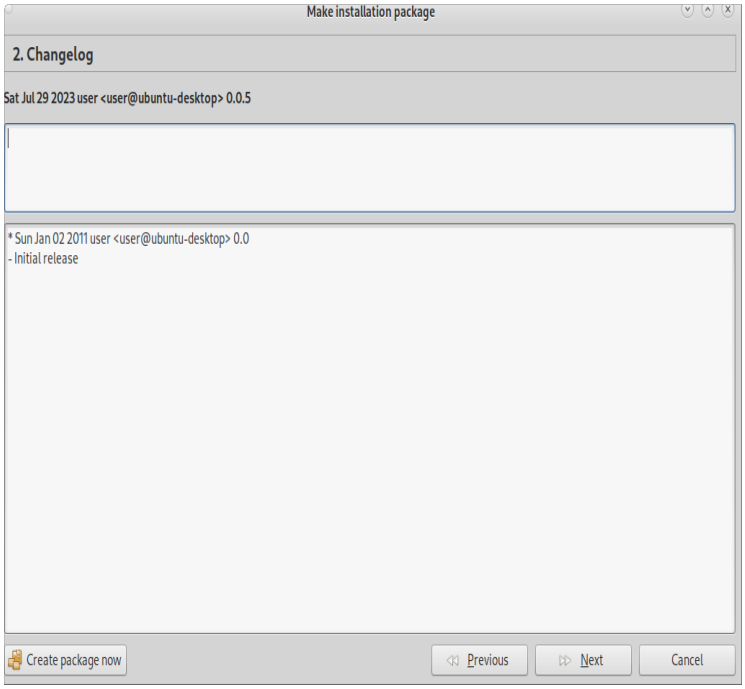
Εικόνα 16: Βήμα 1ο, δημιουργίας πακέτου εγκατάστασης

Όπως φαίνεται στην παραπάνω εικόνα συμπληρώνεται τα απαραίτητα στοιχεία όπως:

Πρόθεμα ονόματος προμηθευτή. (π.χ. gambas)
Επίθεμα αριθμού έκδοσης εφαρμογής (π.χ. 0.0.2)
Το όνομα σας
Την διεύθυνση της ηλεκτρονικής αλληλογραφίας
Το όνομα του προμηθευτή.
Ιστοσελίδα
Περιγραφή εφαρμογής
Άδεια που συνοδεύει την εφαρμογή

Βήμα2ο

Συμπλήρωση στοιχείων από το ημερολόγιο αλλαγών (changelog).
Το ημερολόγιο αλλαγών είναι ένα αρχείο κειμένου όπου ο κατασκευαστής της εφαρμογής καταγράφει τις αλλαγές που πραγματοποιήθηκαν από έκδοση σε έκδοση.

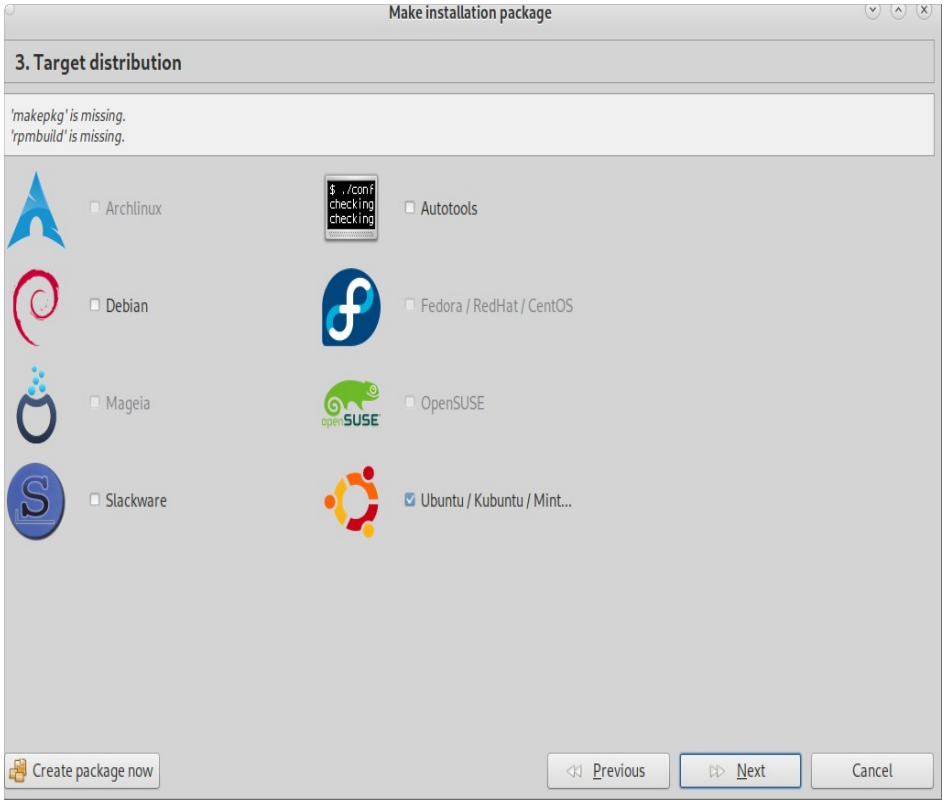


Εικόνα 17: Βήμα 2ο, Εισαγωγή ημερολόγιου αλλαγών

Στο πρώτο πεδίο καταγράφεται η παρούσα έκδοση.
Στο δεύτερο πεδίο καταγράφονται τα στοιχεία από το ημερολόγιο αλλαγών.

Βήμα 3ο

Στην συνέχεια επιλέγεται το είδος του πακέτου εγκατάστασης που θέλετε να δημιουργήσετε ανάλογα με την διανομή στην οποία θα εγκατασταθεί η εφαρμογή.
Ο οδηγός δημιουργίας πακέτων εγκατάστασης αναγνωρίζει αυτόματα ποια εργαλεία δημιουργίας πακέτων υπάρχουν στο σύστημα σας και σας εμφανίζει ενεργές τις επιλογές που μπορείτε να κάνετε. Όσες επιλογές δεν είναι διαθέσιμες από το σύστημα σας τις έχει απενεργοποιημένες και τις εμφανίζει γκριζαρισμένες.



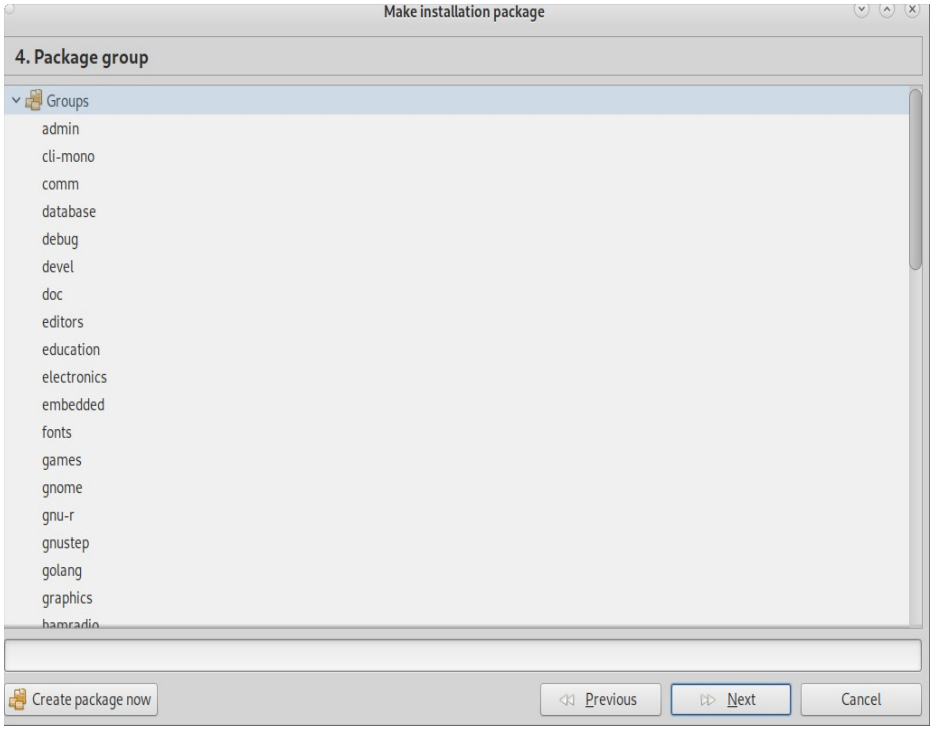
Εικόνα 18: Βήμα 3ο, επιλογές δημιουργίας πακέτων εγκατάστασης

Βήμα 4ο

Στην συνέχεια επιλέγουμε την ομάδα του πακέτου εγκατάστασης από μια λίστα προκαθορισμένων ομάδων για κάθε σύστημα προορισμού.

Στο πάνω μέρος του πλαισίου διαλόγου εμφανίζονται τα συστήματα που προορίζεται να εγκατασταθεί η εφαρμογή.

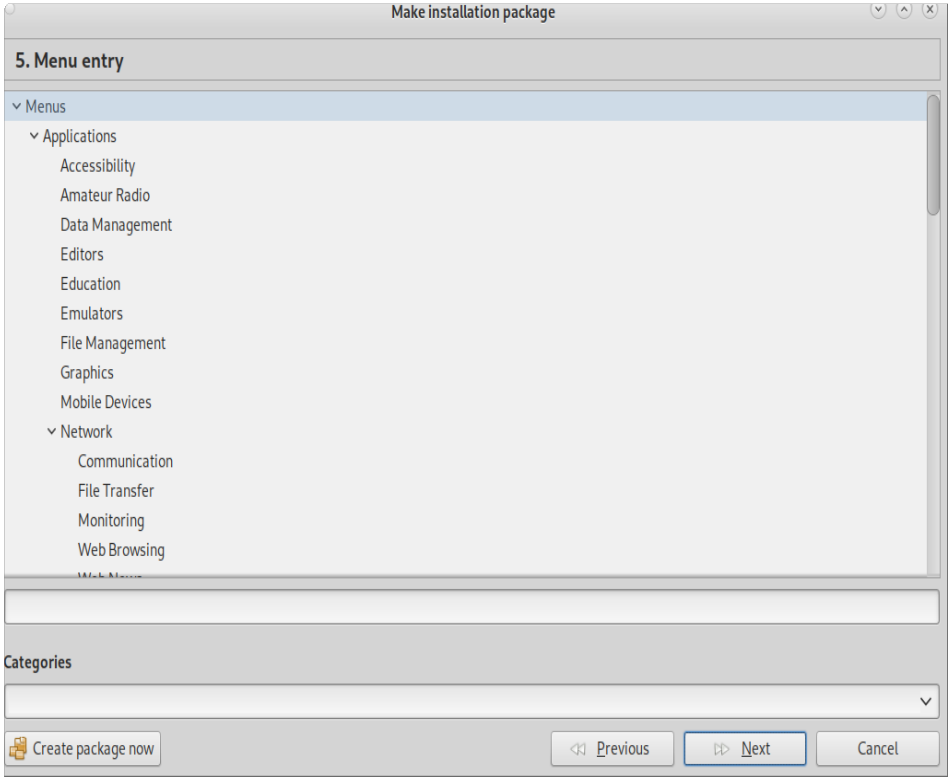
Ορίζουμε την ομάδα εγκατάστασης για κάθε σύστημα.



Εικόνα 19: Βήμα 4ο, επιλογή ομάδας πακέτου

Βήμα 5ο

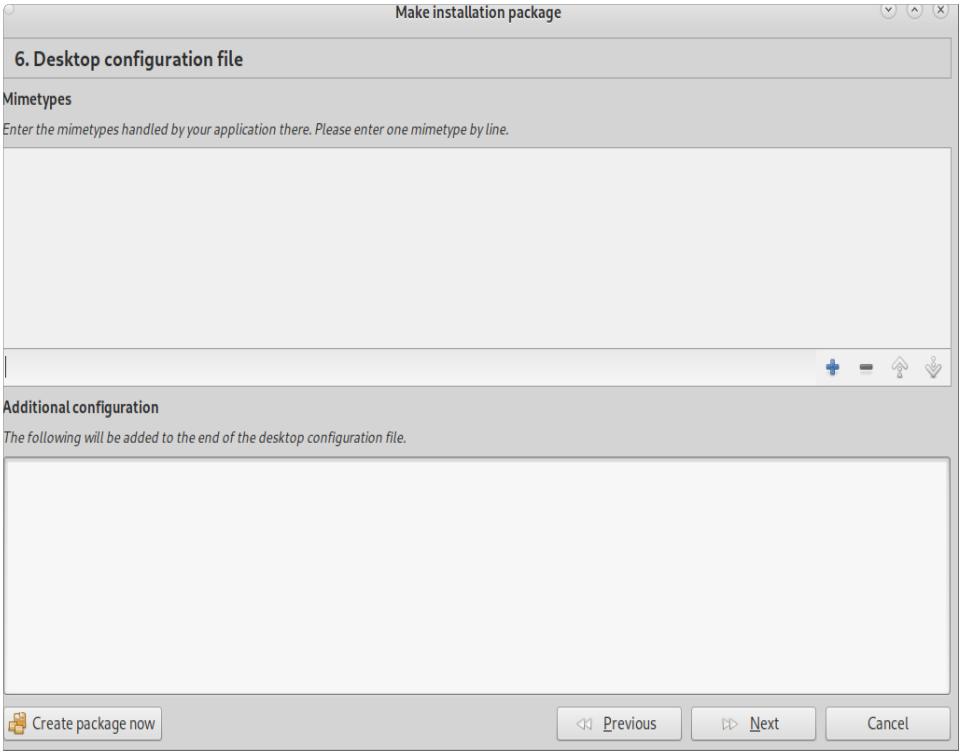
Επιλέγουμε για κάθε διανομή που θα εγκατασταθεί η εφαρμογή, την θέση της εφαρμογής στο μενού εφαρμογών.



Εικόνα 20: Βήμα 5ο, επιλογή εμφάνισης της εφαρμογής στο μενού του συστήματος

Βήμα 6ο

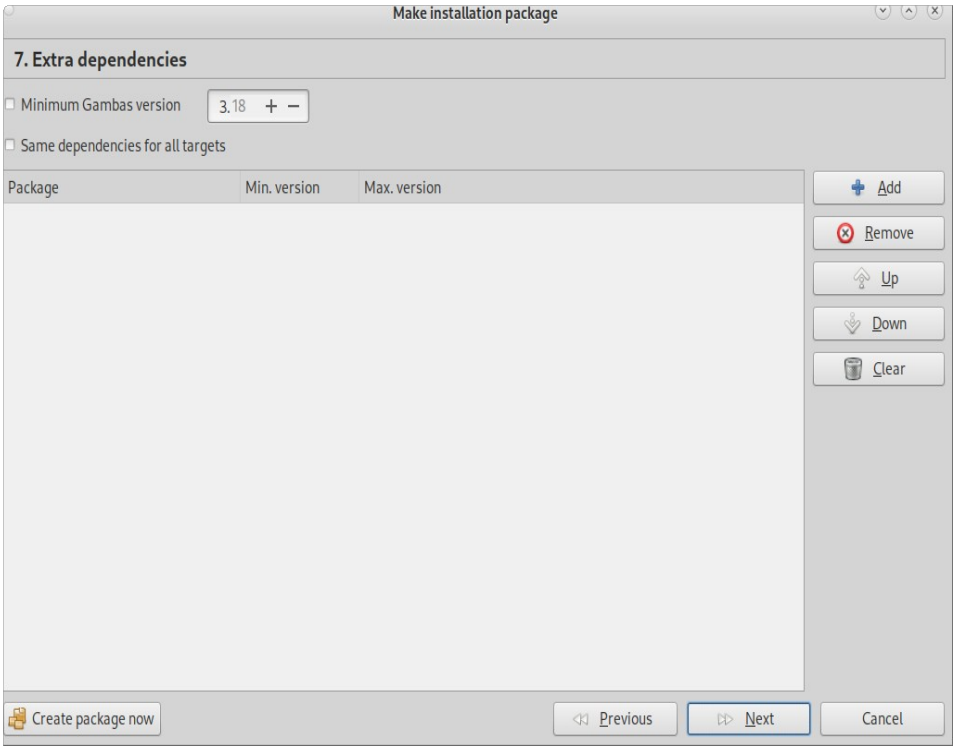
Στο επόμενο βήμα ορίζουμε τους τύπους mime που θα έχει η εφαρμογή μας.
Οι τύποι mime είναι όμοιοι με τις επεκτάσεις αρχείων αλλά πιο ευρέως αναγνωρισμένοι, δείχνουν τι πληροφορίες περιέχει ένα αρχείο και έτσι το λειτουργικό σύστημα γνωρίζει με ποια εφαρμογή να ανοίξει το κάθε αρχείο.
Σε αυτό το βήμα μπορούμε να δηλώσουμε τους τύπους mime που η εφαρμογή μας θα χειρίζεται έτσι ώστε κατά την εγκατάσταση το λειτουργικό σύστημα να αντιστοιχίσει τα αντίστοιχα αρχεία στην εφαρμογή μας.



Εικόνα 21: Βήμα 6ο, ορισμός τύπων mime που θα διαχειρίζεται η εφαρμογή μας.

Βήμα 7ο

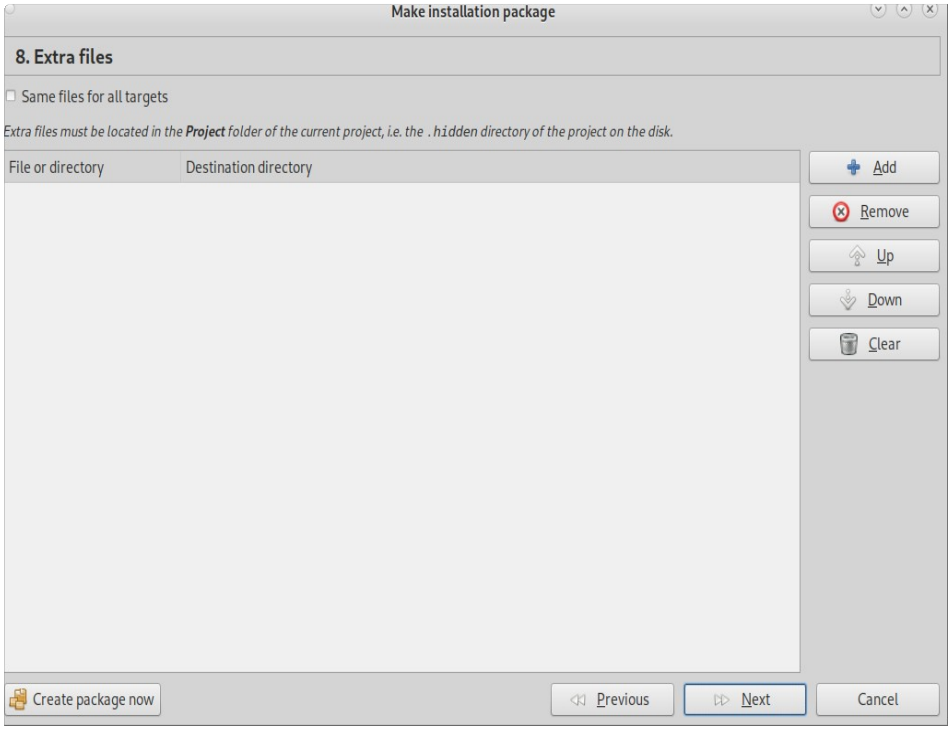
Στο επόμενο βήμα ορίζουμε αν η εφαρμογή μας θα απαιτεί άλλα εξαρτήματα εφαρμογών. Εδώ μπορούμε να κάνουμε μια λίστα με όλα τα απαραίτητα εξαρτήματα.



Εικόνα 22: Βήμα 7ο, η λίστα απαραίτητων εξαρτημάτων της εφαρμογής μας.

Βήμα 8ο

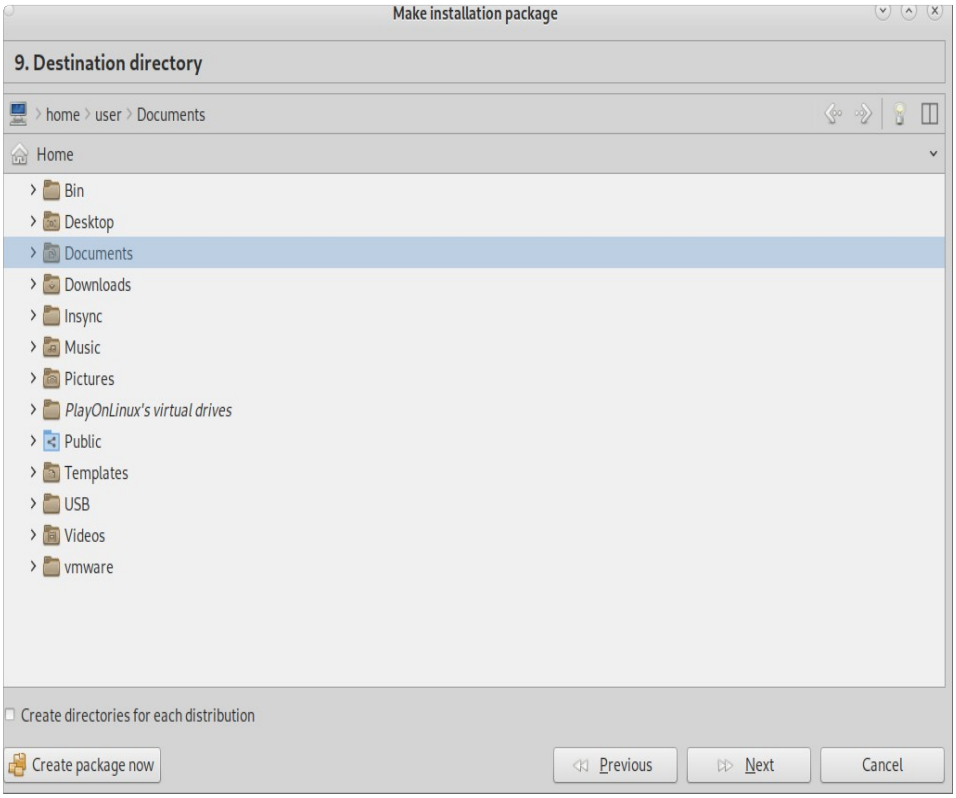
Στο επόμενο βήμα εισάγουμε άλλα αρχεία που θα έχει η εφαρμογή και τον φάκελο προορισμού τους.



Εικόνα 23: Βήμα 8ο, προσθήκη άλλων αρχείων

Βήμα 9ο

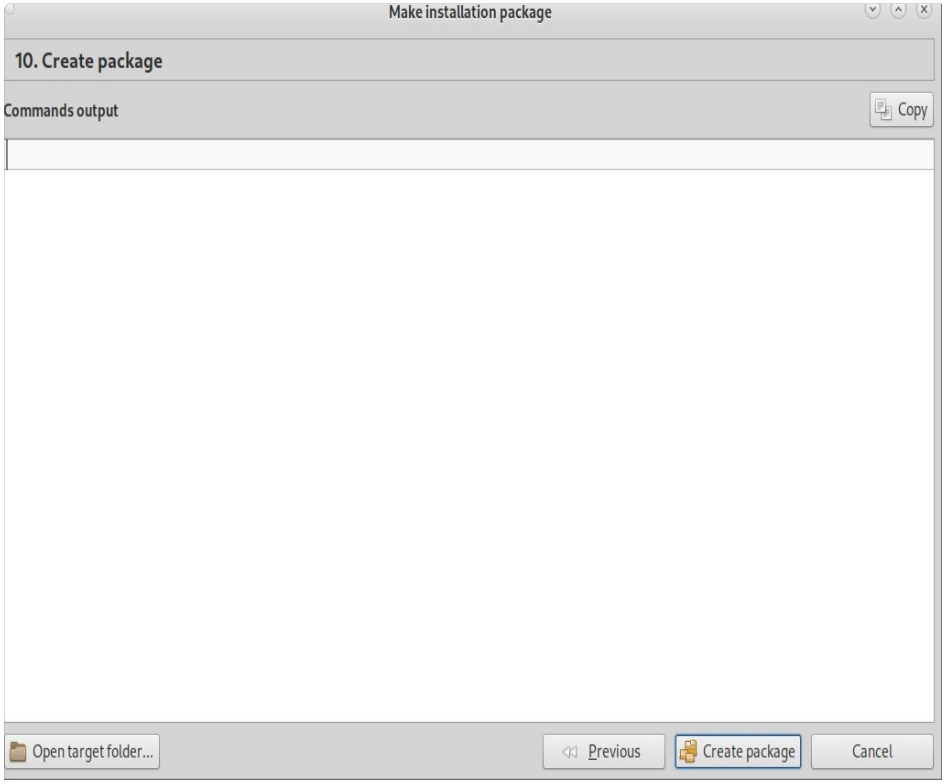
Στο επόμενο βήμα επιλέγουμε τον φάκελο όπου θα δημιουργηθούν τα πακέτα εγκατάστασης. Μπορούμε να επιλέξουμε με τικ, να δημιουργηθεί ξεχωριστός φάκελος για κάθε διανομή προορισμού.



Εικόνα 24: Βήμα 9ο, επιλογή τοποθεσίας δημιουργίας πακέτων εγκατάστασης

Βήμα 10ο

Όλες οι πληροφορίες έχουν συλλεχθεί και ο οδηγός είναι έτοιμος για την δημιουργία των πακέτων εγκατάστασης. Πατώντας το Create package, αρχίζει η δημιουργία των πακέτων.



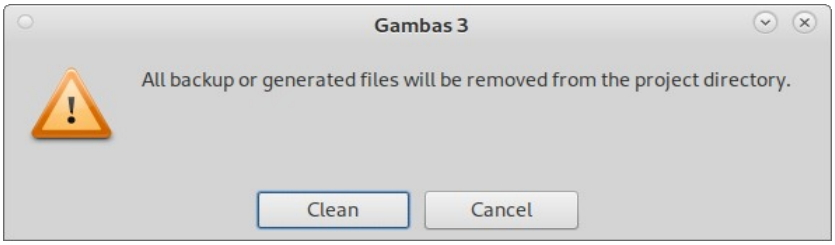
Εικόνα 25: Βήμα 10ο, δημιουργία των πακέτων εγκατάστασης

Διαμοίρασε στην φάρμα λογισμικού (Share on software farm)

Με την εντολή αυτή ανεβάζετε το έργο στην φάρμα λογισμικού της Gambas3.

Καθάρισε (Clean up)

Η εντολή “Καθάρισε” διαγράφει εφεδρικά αρχεία ή αρχεία που έχουν παραχθεί κατά την μεταγλώττιση.



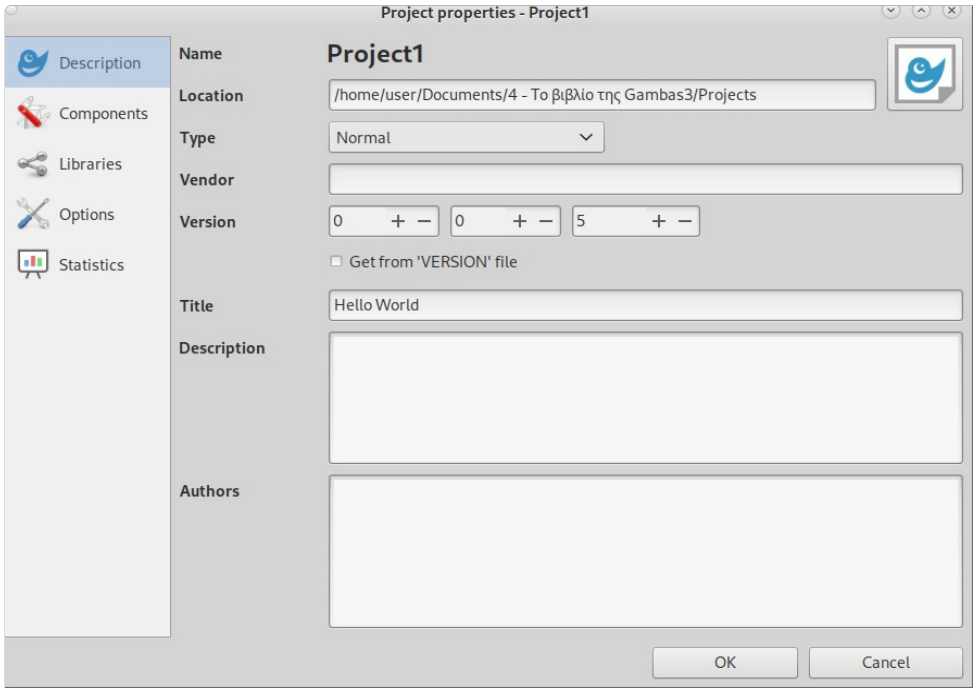
Εικόνα 26: Καθαρισμός αρχείων

Ανανέωση (Refresh)

Η εντολή ανανέωση ξαναδιαβάζει τα περιεχόμενα του φακέλου του έργου και ενημερώνει τον εξερευνητή έργου στο αριστερό μέρος του περιβάλλοντος εργασίας.

Ιδιότητες (Properties)

Η εντολή ιδιότητες εμφανίζει την παρακάτω φόρμα.



Εικόνα 27: Ιδιότητες έργου, καρτέλα "Περιγραφή (Description)"

Καρτέλα "Περιγραφή (Description)"

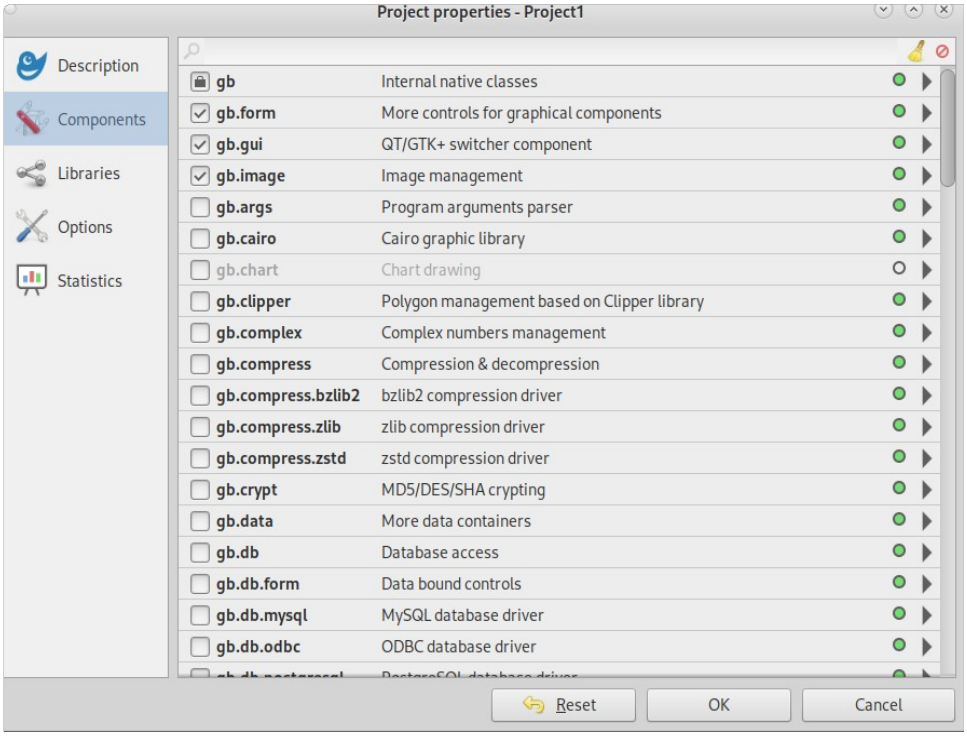
Στην καρτέλα Περιγραφή (Description) εμφανίζονται οι επιλογές:

- Αλλαγής εικονιδίου
- Τοποθεσία έργου
- Τύπος
- Εκδότης
- Αριθμός έκδοσης
- Τίτλος
- Περιγραφή
- Συγγραφείς έργου

Καρτέλα “Εξαρτήματα (Components)”

Στην καρτέλα εξαρτήματα εμφανίζεται η λίστα όλων των εξαρτημάτων του έργου ή όλων των εξαρτημάτων που μπορούμε να προσθέσουμε στο έργο.

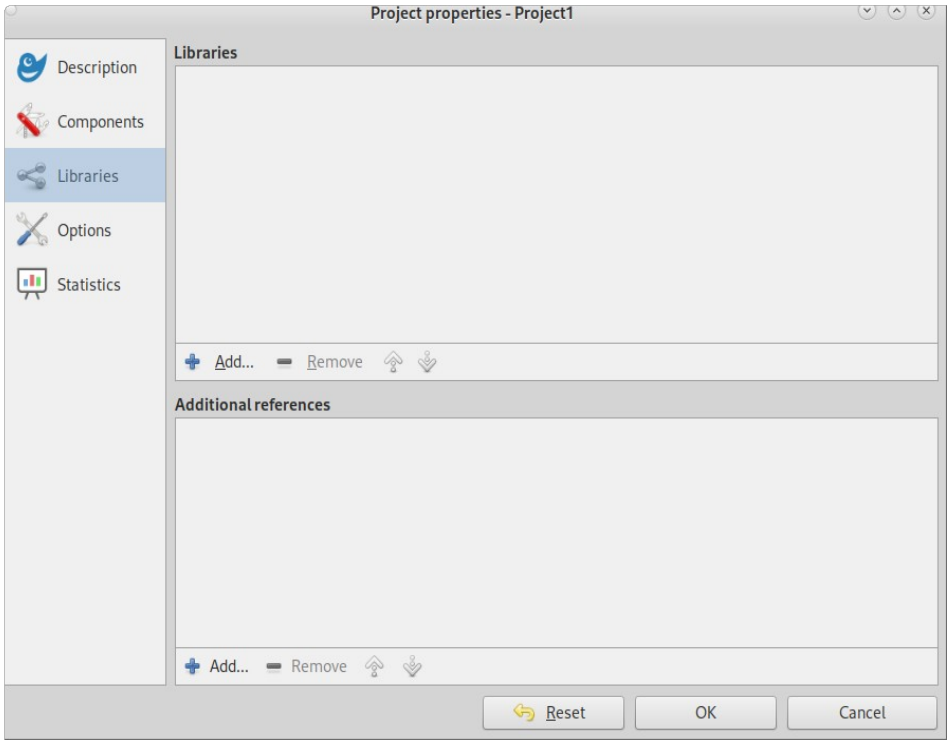
Τα εξαρτήματα είναι συστατικά μέρη ενός συστήματος, π.χ. μια ανεξάρτητη υπορουτίνα, ένα ανεξάρτητο στοιχείο ελέγχου, που μπορούν μετά τη μεταγλώττιση να συνδεθούν και να συνεργαστούν δυναμικά με διαφορετικά προγράμματα.



Εικόνα 28: Ιδιότητες έργου, καρτέλα "Εξαρτήματα (Components)"

Καρτέλα “Βιβλιοθήκες (Libraries)”

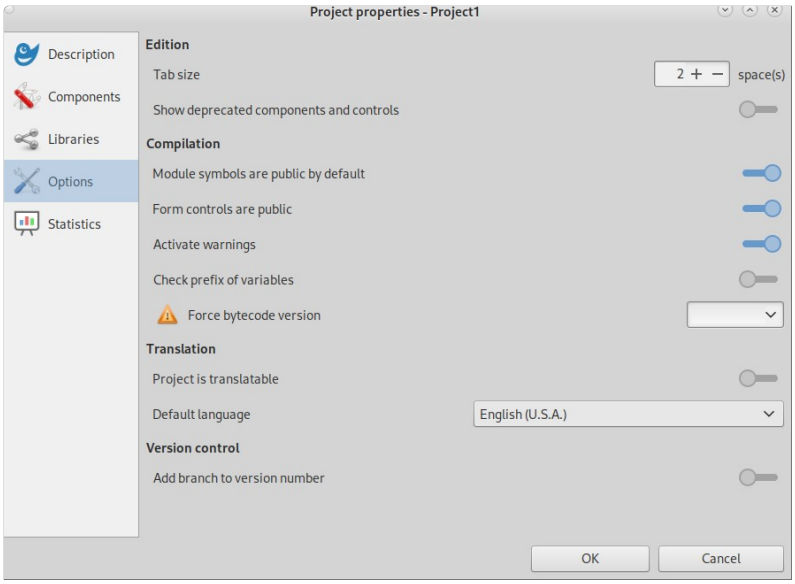
Στην καρτέλα βιβλιοθήκες μπορούμε να εισάγουμε άλλες βιβλιοθήκες στο έργο.



Εικόνα 29: Ιδιότητες έργου, καρτέλα "Βιβλιοθήκες (Libraries)"

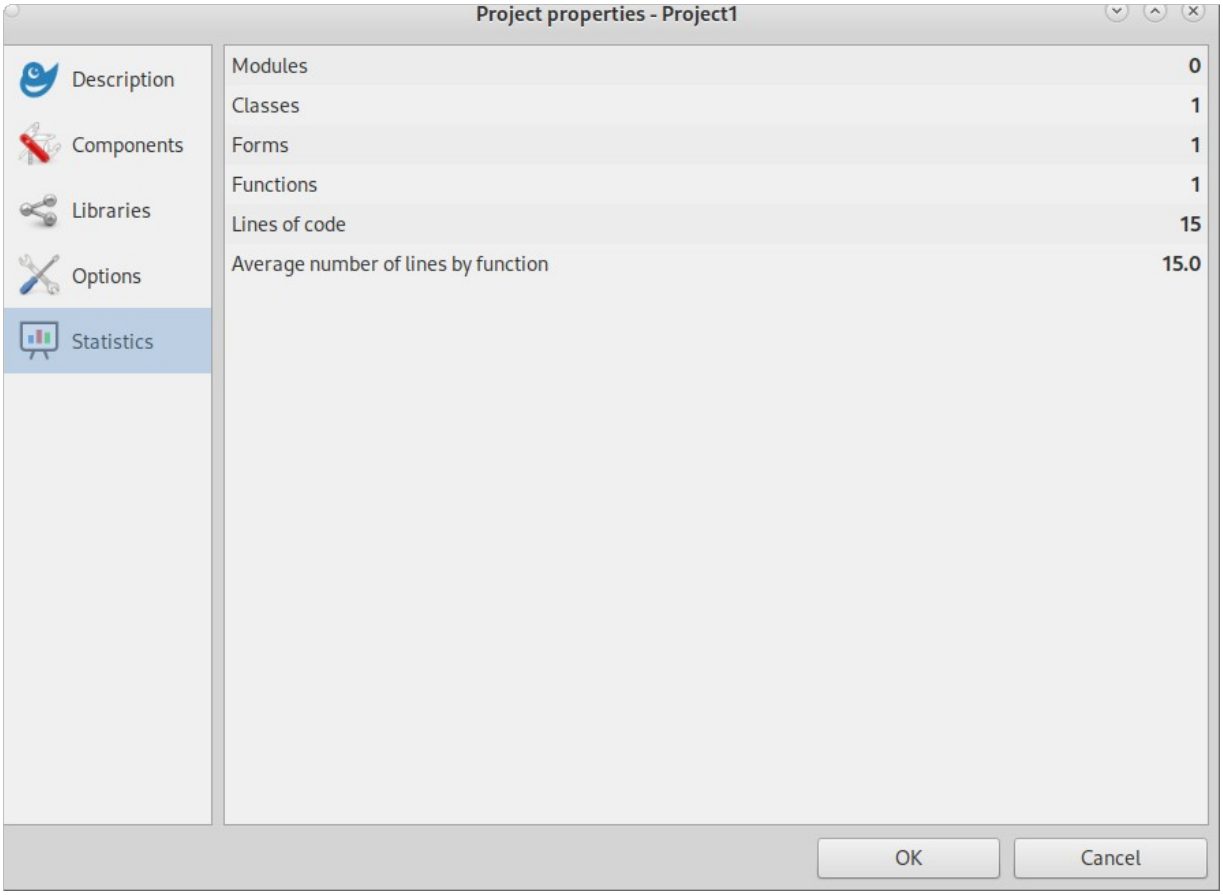
Καρτέλα “Επιλογές (Options)”

Στην καρτέλα Επιλογές μπορούμε να ρυθμίσουμε διάφορες ρυθμίσεις για το έργο όπως φαίνονται παρακάτω.



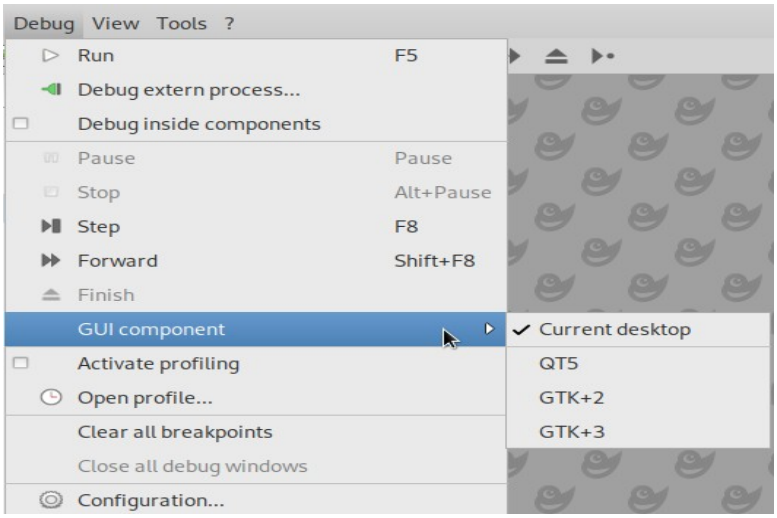
Καρτέλα “Στατιστιά (Statistics)”

Στην καρτέλα Στατιστικά μπορούμε να δούμε διάφορα στατιστικά για το έργο όπως φαίνονται παρακάτω.


Project properties - Project1
Description
Components
Libraries
Options
Statistics
Modules 0
Classes 1
Forms 1
Functions 1
Lines of code 15
Average number of lines by function 15.0
OK
Cancel

Αποσφαλμάτωση (Debug)

Το μενού “Αποσφαλμάτωση” είναι το παρακάτω.



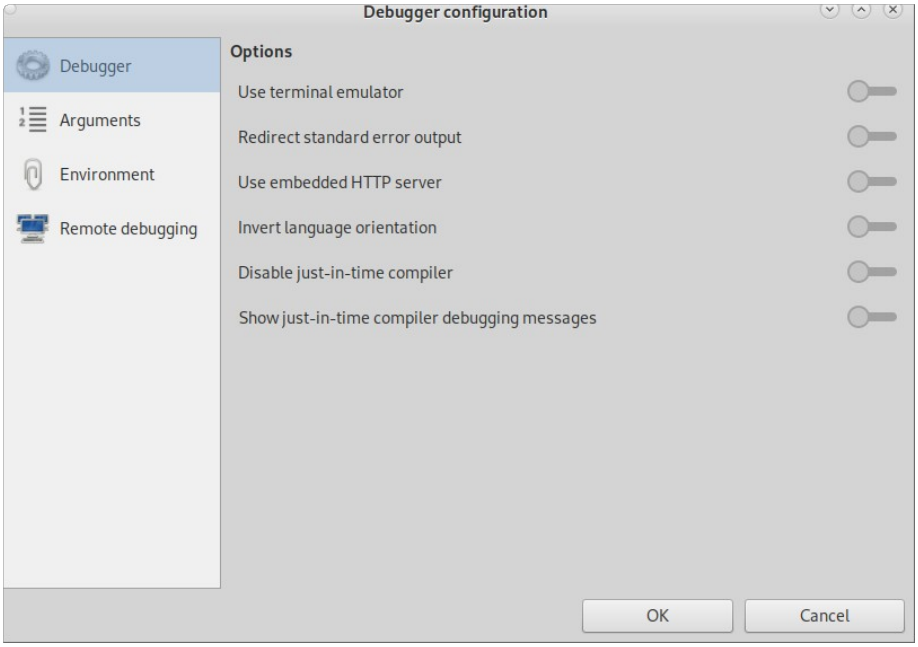
Εικόνα 30: Μενού αποσφαλμάτωσης

Εδώ δίνεται η επιλογή στον προγραμματιστή

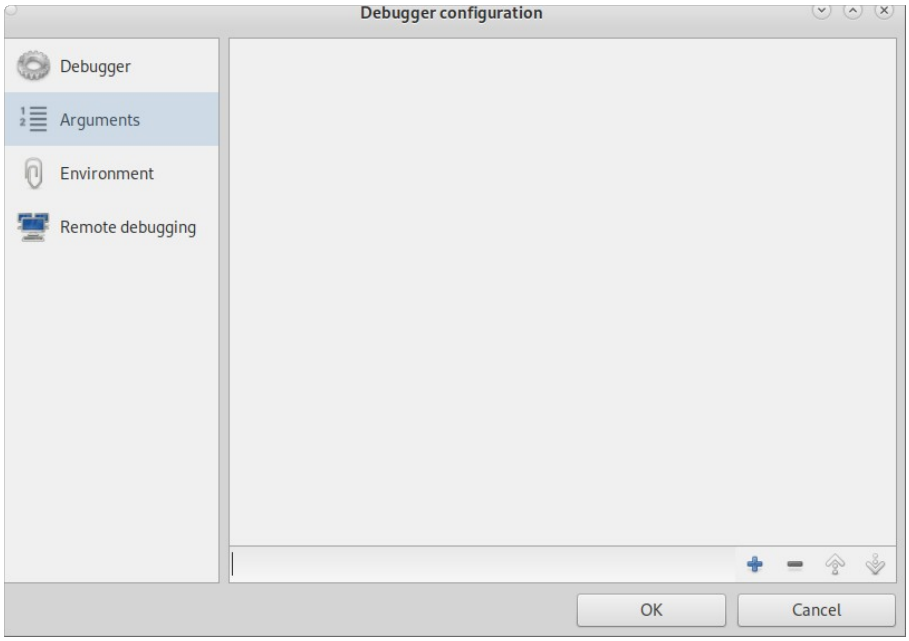
- Να τρέξει το πρόγραμμα του (Run, F5),
- Να αποσφαλματώσει μια εξωτερική διεργασία
- Να γίνει αποσφαλμάτωση σε εξαρτήματα (components)
- Να το σταματήσει σε κατάσταση παύσης (Pause),
- Να σταματήσει την εκτέλεση του προγράμματος (Stop)
- Όταν είναι σε κατάσταση παύσης να πάει ένα βήμα εμπρός (Step)
- Να συνεχίσει την εκτέλεση του προγράμματος (Forward)
- Να τερματίσει την ροή εκτέλεσης της υπορουτίνας (Finish)
- Να επιλέξει εξάρτημα γραφικής διεπαφής (Gui component)
- Να ενεργοποιήσει τα Προφίλ αποσφαλμάτωσης (Activate profiling)
- Να ανοίξει κάποιο προφίλ (Open profile)
- Να αφαιρέσει όλα τα σημεία διακοπής (Clear all breakpoints)
- Να κλείσει τα παράθυρα αποσφαλμάτωσης (Close all debug windows)
- Να ρυθμίσει την αποσφαλμάτωση (Configuration)

Ρύθμιση της αποσφαλμάτωσης (Configuration…)

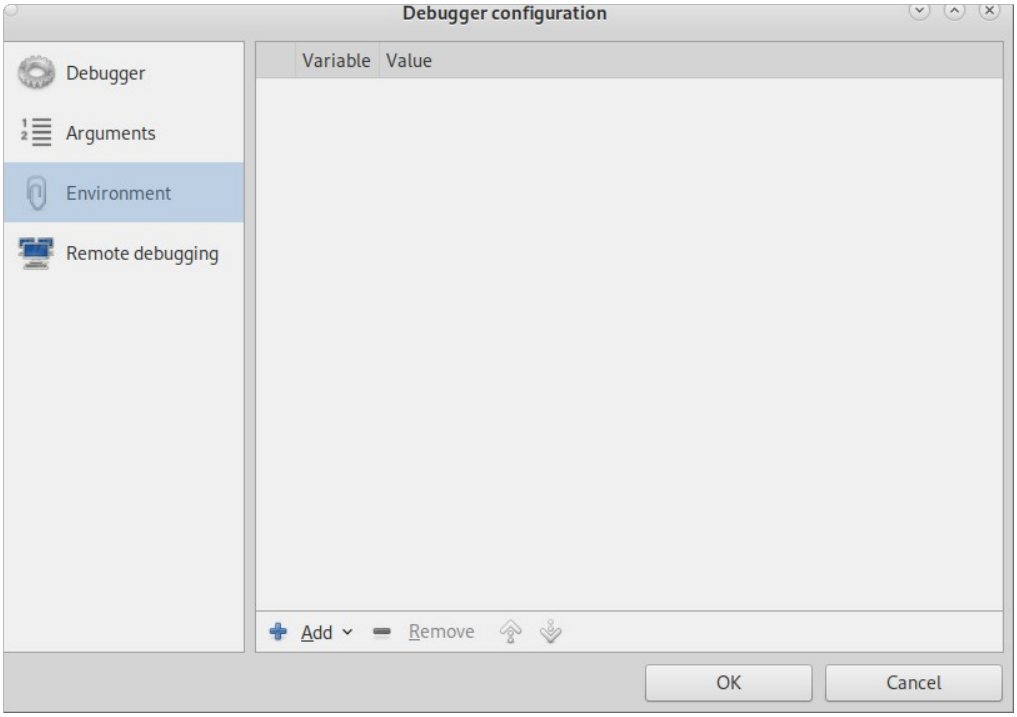
Με την εντολή αυτή εμφανίζεται το παράθυρο ρυθμίσεων της αποσφαλμάτωσης όπως φαίνεται παρακάτω, με διάφορες επιλογές για τον αποσφαλματωτή της Gambas3.



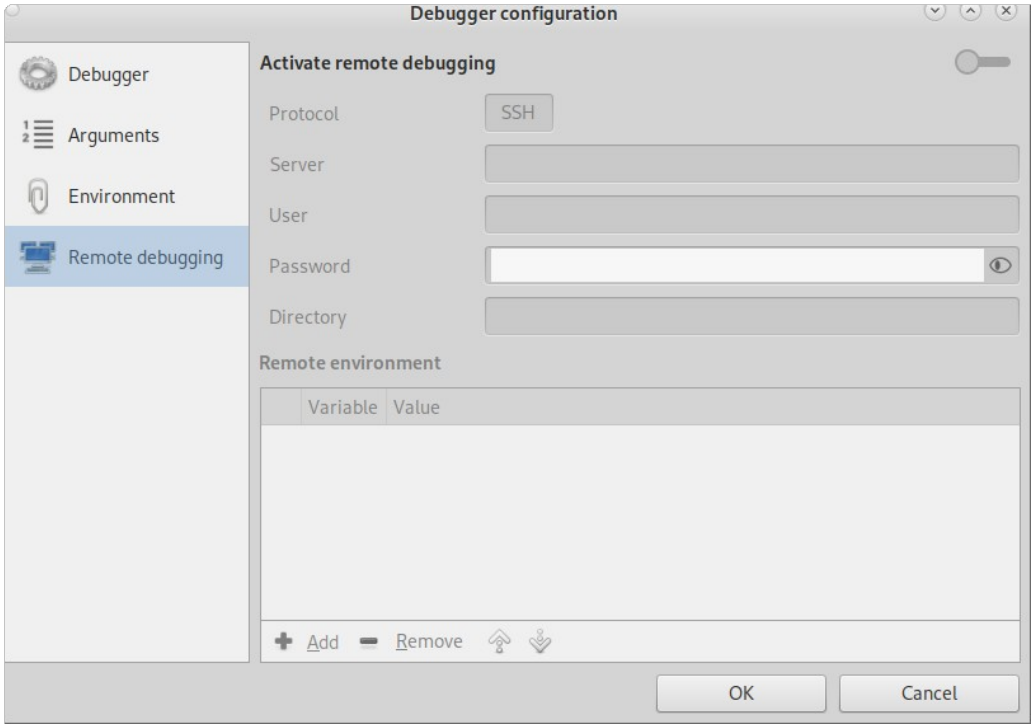
Επίσης μπορούμε να ορίσουμε Ορίσματα (Arguments) στον αποσφαλματωτή, όπως φαίνεται παρακάτω.



Να ορίσουμε ρυθμίσεις περιβάλλοντος (Enviroment)

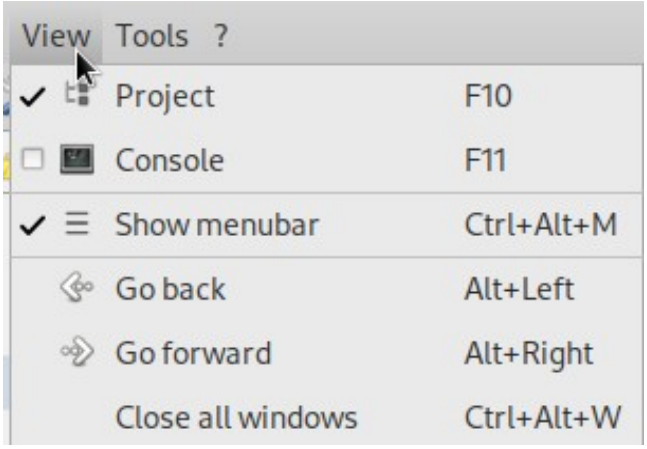


Ή ακόμη να εκτελέσουμε απομακρυσμένη αποσφαλμάτωση μέσω SSH.



Προβολή (View)

Το μενού “Προβολή (View)” είναι το παρακάτω.



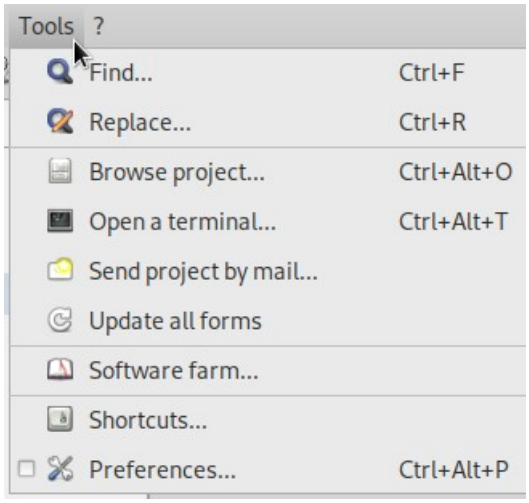
Εικόνα 31: Το μενού Προβολή (View)

Το μενού “Προβολή” δίνει τις επιλογές

- Προβολή της εξερεύνησης του έργου (Project explorer). Η εξερεύνηση του έργου είναι το αριστερό μέρος του περιβάλλοντος ανάπτυξης που εμφανίζει σε μορφή δένδρου τα αρχεία του έργου.
- Προβολή της κονσόλας της Gambas (Console).
- Να κρύψει ή να εμφανίσει την μπάρα μενού.
- Να πάει πίσω ένα παράθυρο (Go back)
- Να πάει εμπρός ένα παράθυρο (Go forward)
- Κλείσιμο όλων των παραθύρων (Close all windows).

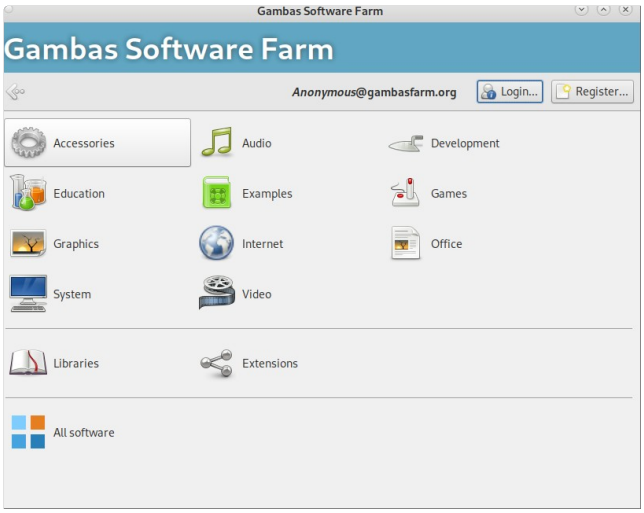
Εργαλεία (Tools)

Το μενού “Εργαλεία (Tools)” είναι το παρακάτω.

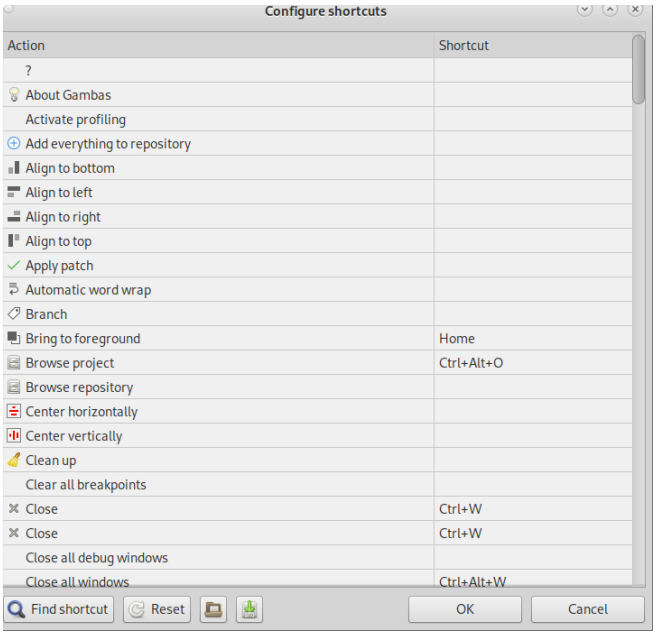


Από το Μενού Εργαλεία μπορούμε:

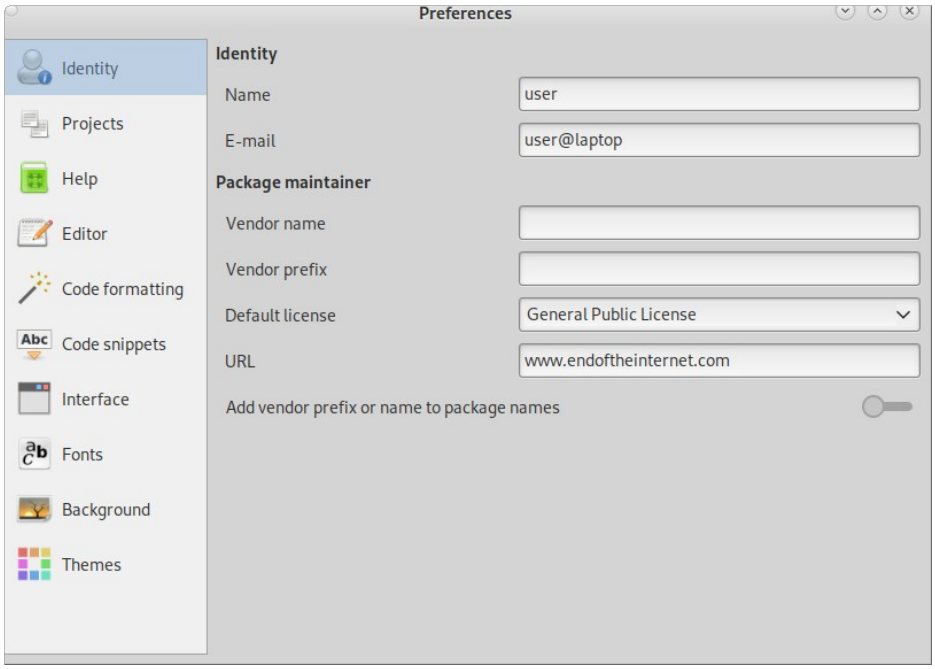
- Να κάνουμε αναζήτηση μιας λέξης ή έκφρασης στον πηγαίο κώδικα
- Να αντικαταστήσουμε μια έκφραση με μια άλλη στον πηγαίο κώδικα
- Να ανοίξουμε την εξερεύνηση αρχείων για το τρέχον έργο
- Να ανοίξουμε ένα τερματικό για το τρέχον έργο
- Να στείλουμε το έργο σε ένα email
- Να ανανεώσουμε όλες τις φόρμες
- Να ανοίξουμε ένα παράθυρο περιήγισης στην Φάρμα Λογισμικού



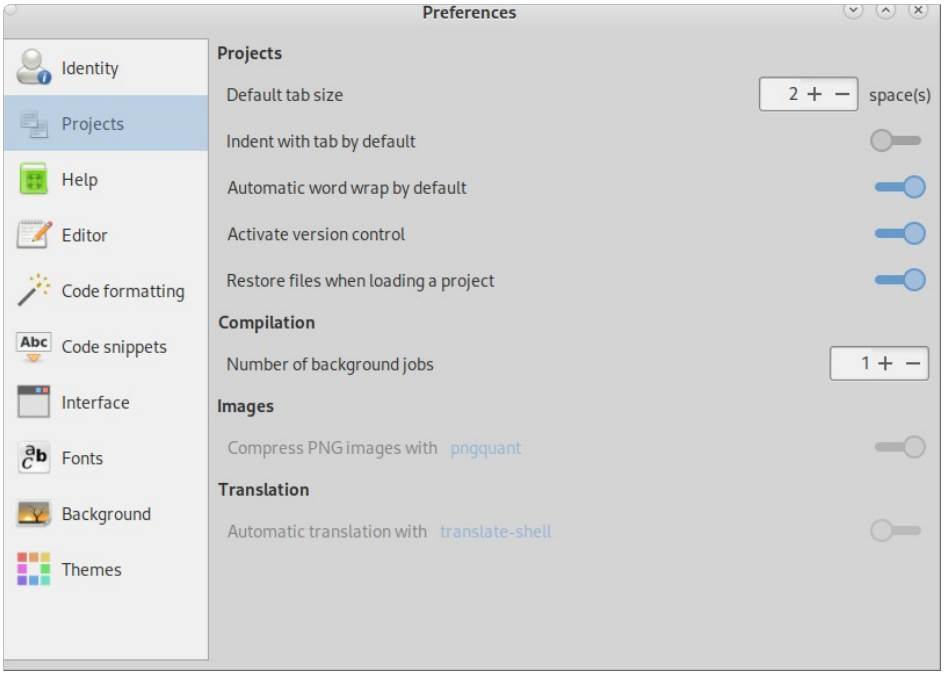
- Να ανοίξουμε ένα παράθυρο επεξεργασίας των συντομεύσεων του ide της Gambas3



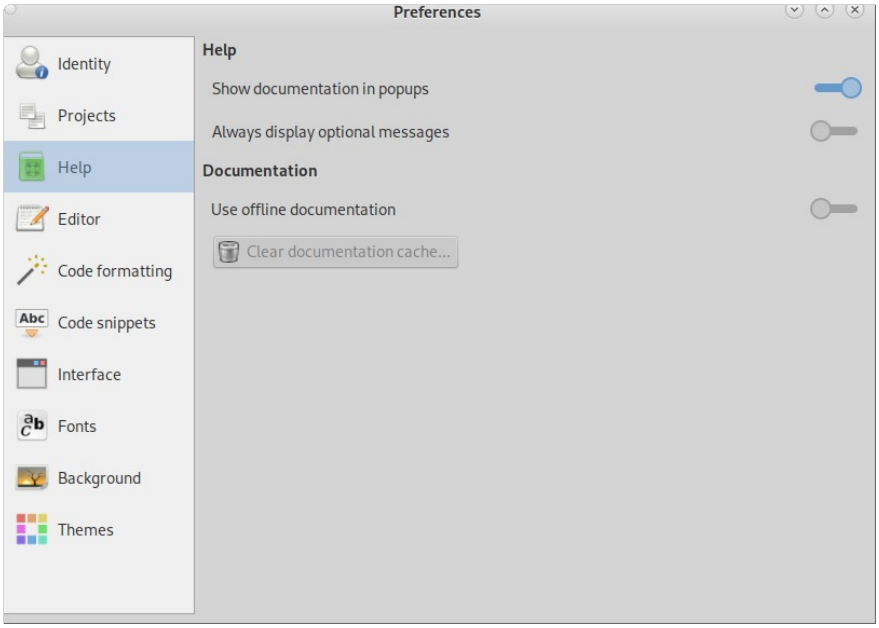
- Να ανοίξουμε ένα πράθυρο σχετικά με τις Προτιμήσεις του περιβάλλοντος ανάπτυξης εφαρμογών Gambas3. Η πρώτη καρτέλα είναι προτιμήσεις σχετικά με την ταυτότητα του προγραμματιστή.



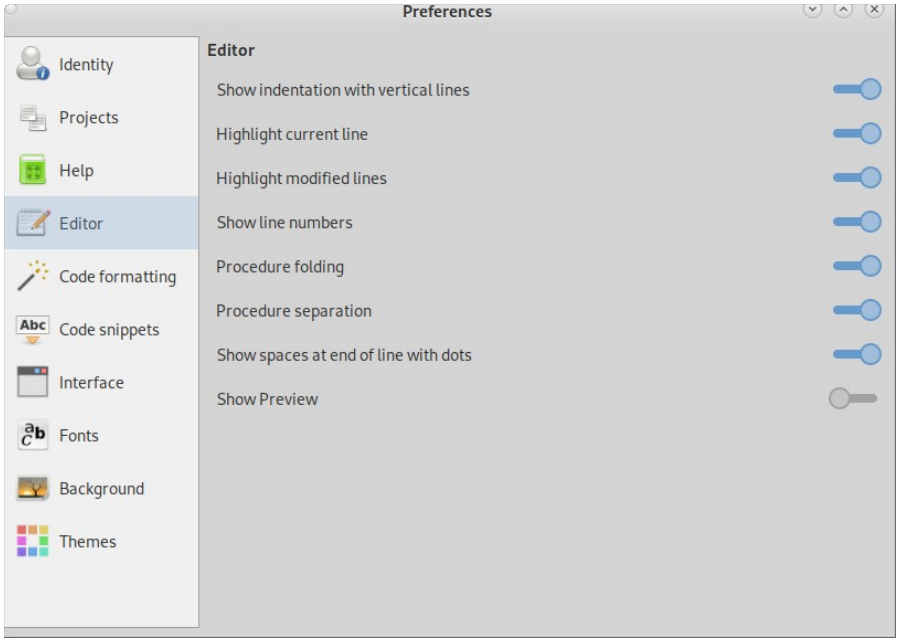
Η επόμενη καρτέλα έχει προτιμήσεις σχετικά με τα έργα.



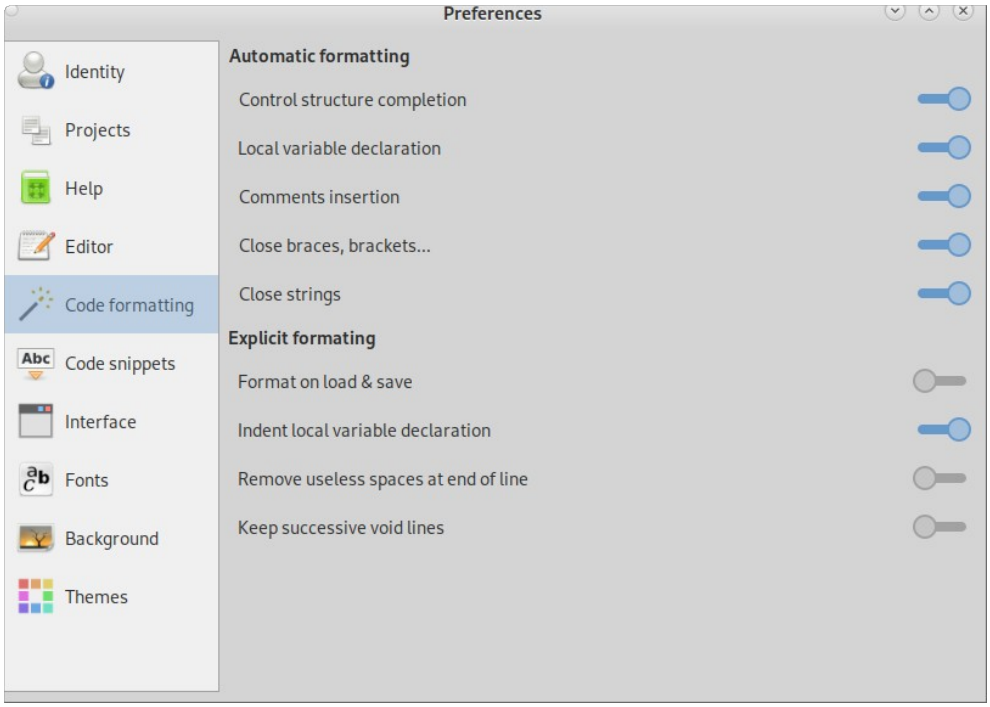
Η επόμενη καρτέλα έχει προτιμήσεις σχετικά με την Βοήθεια που παρέχεται για την γλώσσα προγραμματισμού.



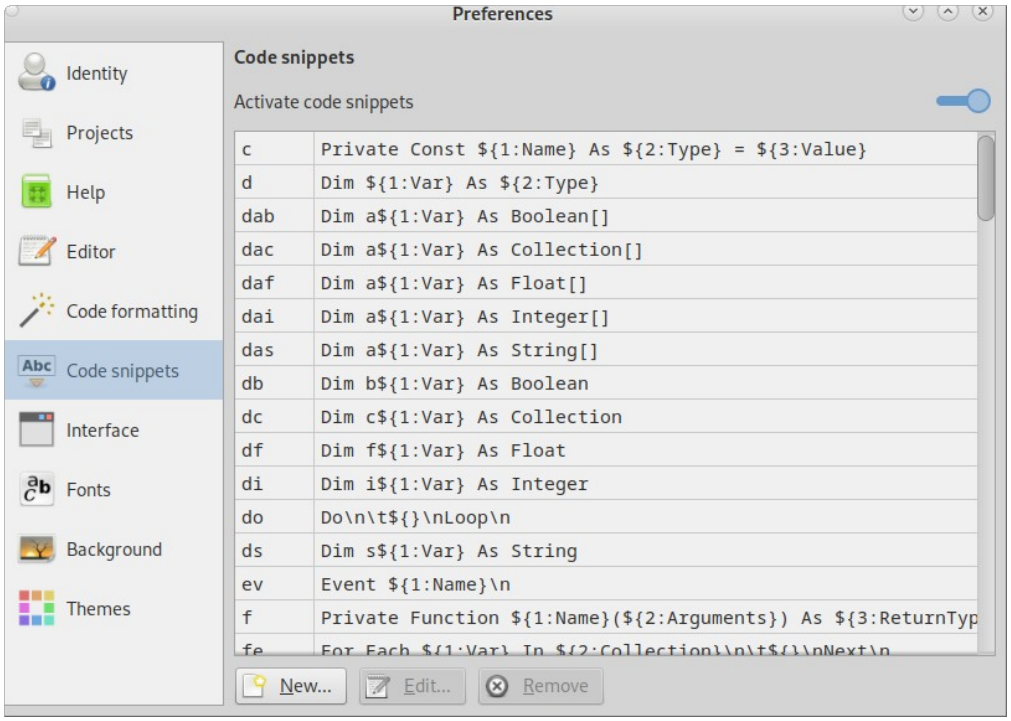
Η επόμενη καρτέλα έχει προτιμήσεις σχετικά με τον επεξεργαστή κώδικα.



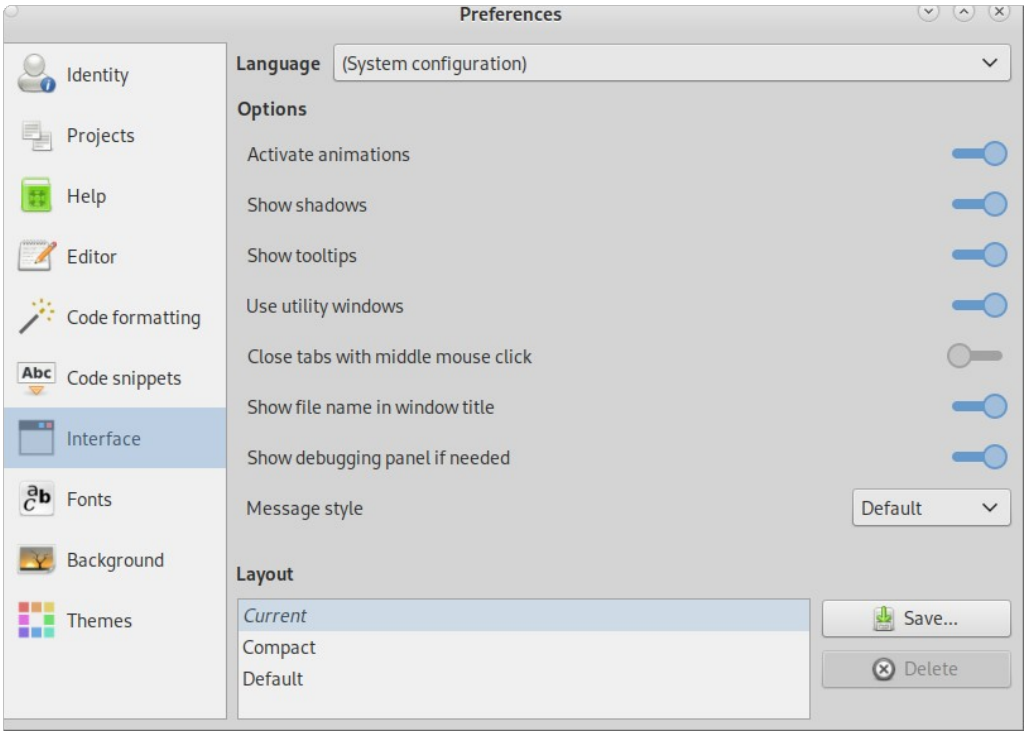
Η επόμενη καρτέλα έχει προτιμήσεις σχετικά με την μορφοποίηση κώδικα.



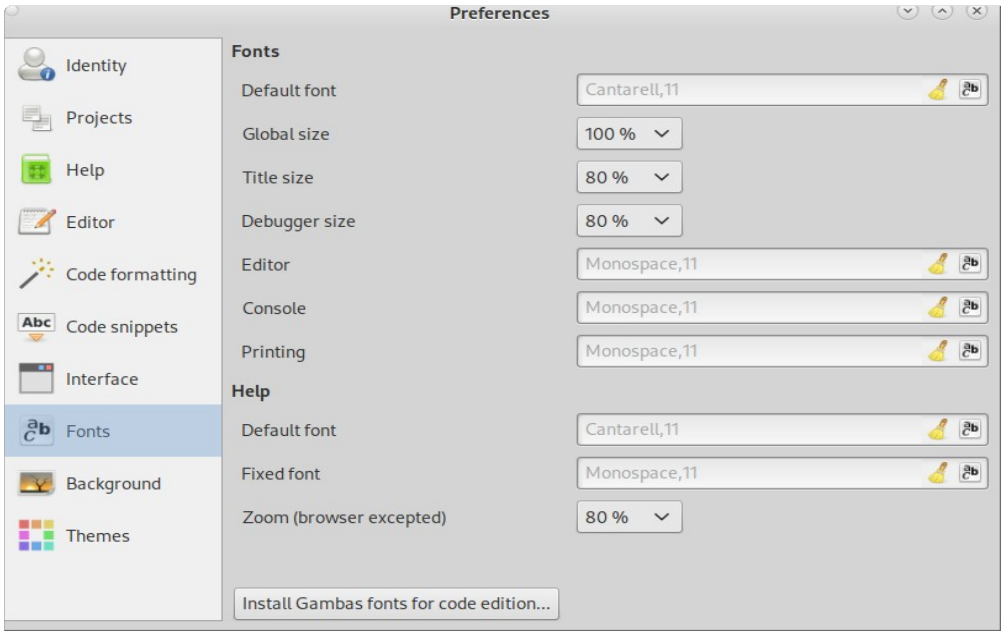
Η επόμενη καρτέλα έχει προτιμήσεις σχετικά με τις συντομεύσεις κώδικα.



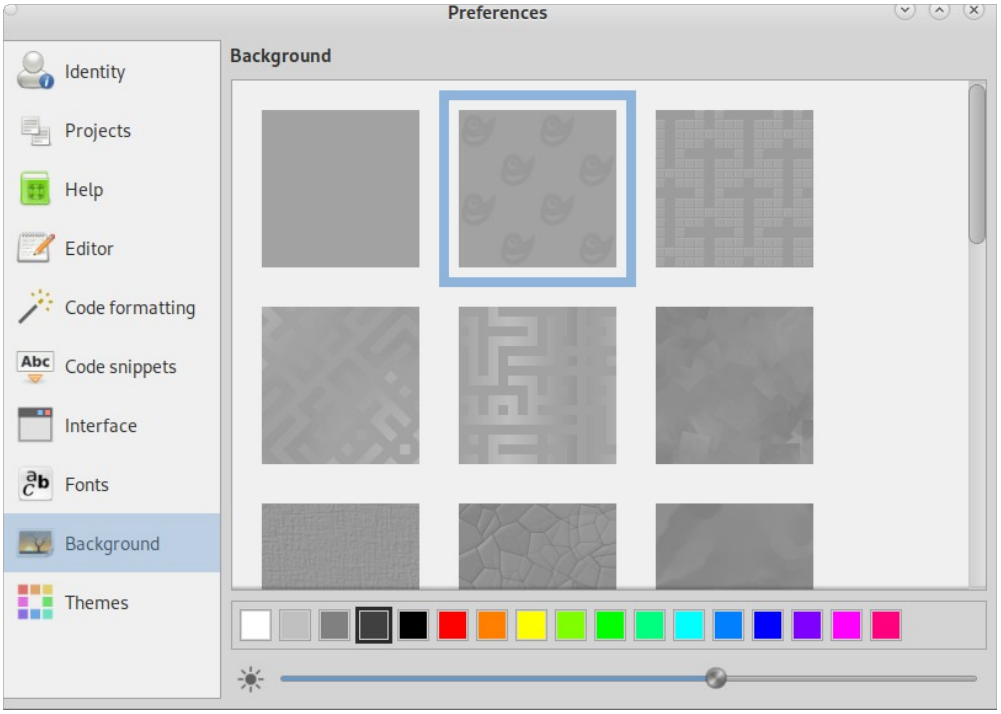
Η επόμενη καρτέλα έχει προτιμήσεις σχετικά με την διεπαφή του περιβάλλοντος ανάπτυξης.



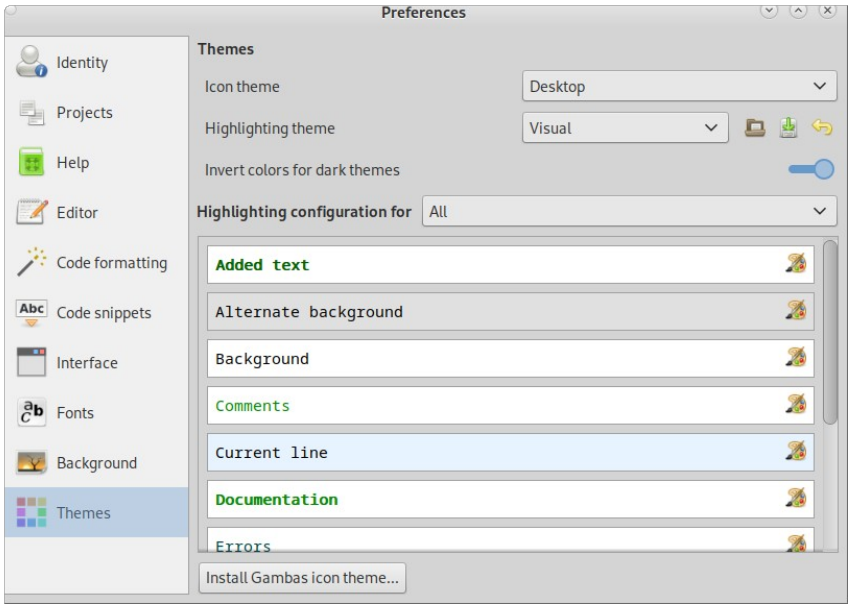
Η επόμενη καρτέλα έχει προτιμήσεις σχετικά με τις γραμματοσειρές της διεπαφής του περιβάλλοντος ανάπτυξης.



Η επόμενη καρτέλα έχει προτιμήσεις σχετικά με την εικόνα φόντου της διεπαφής ανάπτυξης εφαρμογών.



Η επόμενη καρτέλα έχει προτιμήσεις σχετικά με τα θέματα της διεπαφής ανάπτυξης εφαρμογών. Από εδώ μπορούμε να επιλέξουμε τα χρώμματα του επεξεργαστή κώδικα της διεπαφής.



Η μπάρα συντομεύσεων του κυρίως μενού

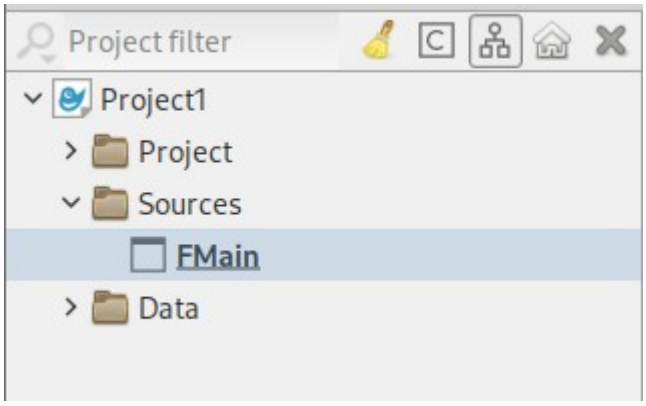
Κάτω από το Κυρίως Μενού υπάρχει η μπάρα συντομεύσεων του κυρίως μενού.
Εδώ υπάρχουν εντολές με την μορφή κουμπιού από το κυρίως μενού.

Οι εντολές αυτές είναι:

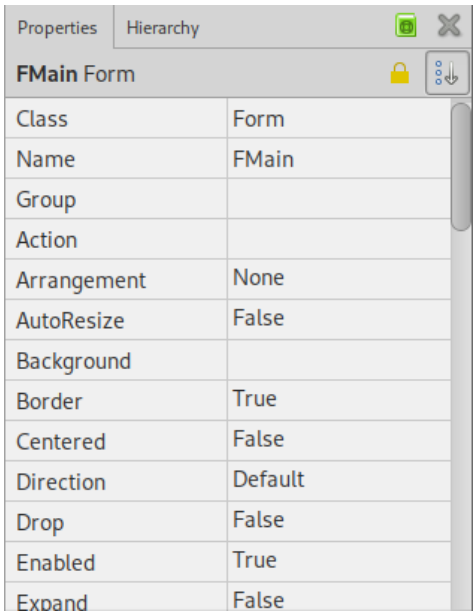
- Νέο έργο (New Project)
- Άνοιγμα έργου (Open Project)
- Αποθήκευση έργου (Save Project)
- Αποθήκευση έργου ως (Save as Project)
- Ιδιότητες έργου (Properties)
- Προτιμήσεις (Preferences)
- Δημιουργία εκτελέσιμου (Make executable)
- Μεταγλώτιση αρχείου (Compile)
- Μεταγλώτιση όλων (Compile All)
- Εκτέλεση έργου (Run)
- Παύση εκτέλεσης έργου (Pause)
- Τερματισμός εκτέλεσης έργου (Stop)
- Εκτέλεση έργου βήμα βήμα (Step)
- Εκτέλεση έργου ένα βήμα εμπρός (Forward)
- Εκτέλεση έργου έως το τέλος της συνάρτησης (Finish current function)
- Εκτέλεση έργου έως την τρέχουσα γραμμή (Run until current line)

Στην συνέχεια η διεπαφή χωρίζεται σε πέντε μέρη, αυτά είναι:

- Η εξερεύνηση των αρχείων έργου (Project Explorer)



- Οι ιδιότητες των αντικειμένων

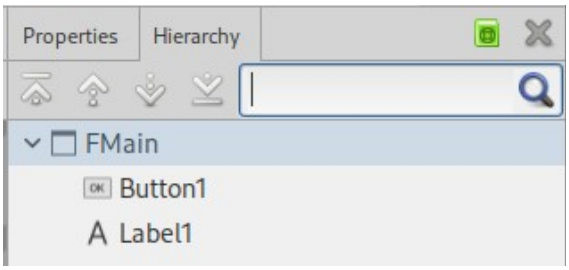


Όταν επιλέγουμε ένα αντικείμενο οι ιδιότητες του εμφανίζονται στην λίστα ιδιοτήτων.
Από την λίστα ιδιοτήτων μπορούμε να αλλάξουμε τις τιμές των ιδιοτήτων και να ορίσουμε τα χαρακτηριστικά των αντικειμένων.

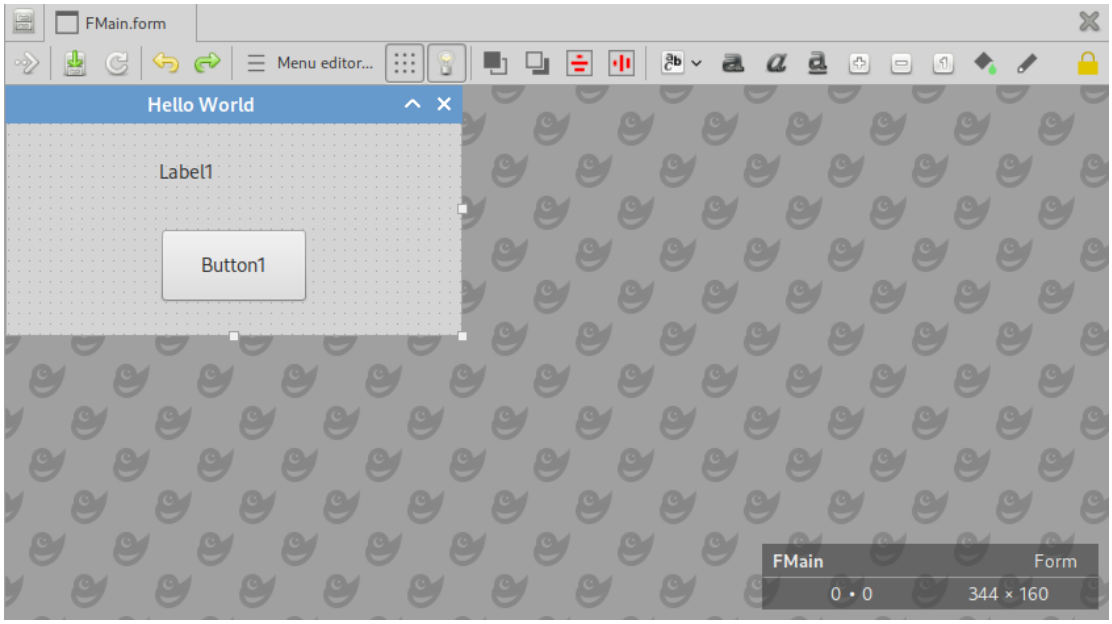
Κάποιο από όλα τα αντικείμενα δέχεται τα άλλα.
Μια φόρμα είναι ένας δοχέας αντικειμένων ελέγχου, δλδ κουμπιών, ετικετών κτλ.
Μια φόρμα μπορεί να δεχθεί και άλλα αντικείμενα δοχείς, όπως Πλαίσια (Frames), Καρτελοσειρές (TabStrips) κτλ που πάνω σε αυτά να τοποθετήσουμε αντικείμενα ελέγχου.

Δίπλα από την καρτέλα Ιδιότητες (Properties), υπάρχει η καρτέλα Ιεραρχία (Hierarchy). Σε αυτήν την καρτέλα μπορούμε να δούμε την ιεραρχία των αντικειμένων πάνω στον δοχέα αντικειμένων.

Η καρτέλα “Ιεραρχία” μας δίνει σε δομή δένδρου τους δοχείς και τα αντικείμενα τους.



- Η περιοχή σχεδίασης φορμών και αντικειμένων



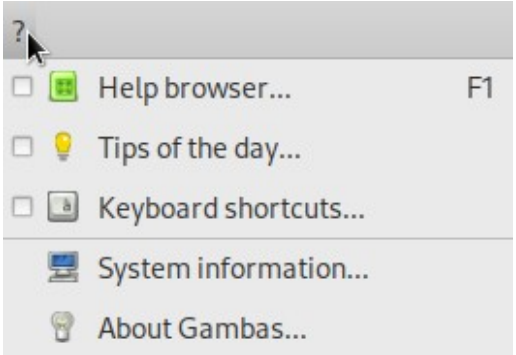
- Οι καρτέλες (Tabs) των αντικειμένων που μπορούν να σχεδιαστούν πάνω στις φόρμες

- Η περιοχή εμφάνισης της ενσωματομένης κονσόλας-τερματικό

Εδώ εμφανίζονται τα αποτελέσματα της εντολής PRINT.

Βοήθεια (?)

Το μενού βοήθειας είναι το παρακάτω.



Εικόνα 32: Το μενού βοήθειας

Σε αυτό υπάρχουν οι επιλογές:

- Βοήθεια της Gambas, εμφανίζει στον φυλλομετρητή την τεκμηρίωση της γλώσσας Gambas
- Συμβουλές της ημέρας
- Συντομεύσεις πληκτρολογίου
- Πληροφορίες συστήματος
- Σχετικά με την Gambas

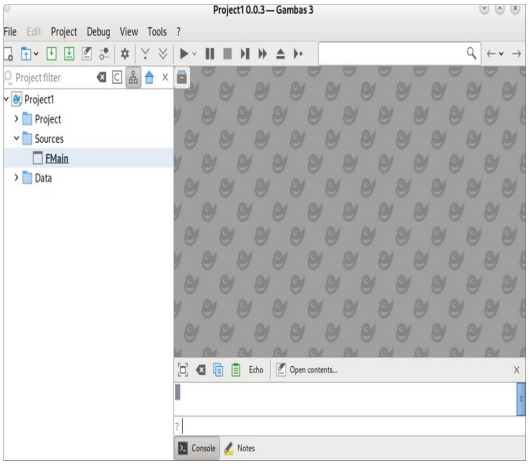
Κεφάλαιο 3ο - Προβολή σχεδίασης

Στον οπτικό αντικειμενοστραφή προγραμματισμό έχουμε να διαχειριστούμε οπτικά αντικείμενα όπως κουμπιά, ετικέτες, φόρμες κτλ.
Πριν να συγγράψουμε τον κώδικα σχεδιάζουμε την διεπαφή (interface) της εφαρμογής.
Μια διεπαφή αποτελείται από φόρμες και κάθε φόρμα έχει αντικείμενα ελέγχου όπου ο τελικός χρήστης θα τα χρησιμοποιεί για να λειτουργήσει την εφαρμογή.

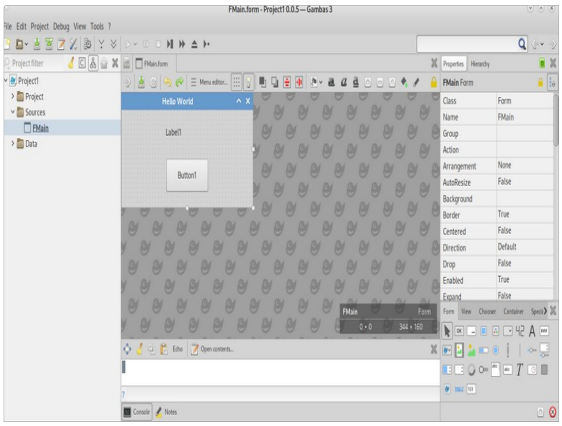
Τις φόρμες και τα αντικείμενα ελέγχου μας τα παρέχουν τα διάφορα εξαρτήματα (components) που επιλέγουμε για το έργο μας. Εμείς σε πρώτο στάδιο πρέπει να ορίσουμε τις φόρμες, τις διαστάσεις τους και άλλες ιδιότητες τους και να σχεδιάσουμε τα έτοιμα αντικείμενα ελέγχου πάνω στις φόρμες.

Όλα αυτά τα κάνουμε στην προβολή σχεδίασης του περιβάλλοντος ανάπτυξης εφαρμογών (IDE).

Παρακάτω φαίνεται η προβολή σχεδίασης του περιβάλλοντος ανάπτυξης εφαρμογών της Gambas.



Εικόνα 33: Η προβολή σχεδίασης (σκούρα γκρι περιοχή) χωρίς φόρμα



Εικόνα 34: Η προβολή σχεδίασης (σκούρα γκρι περιοχή) με φόρμα προς σχεδίαση

Σχεδίαση του έργου “Hello World”

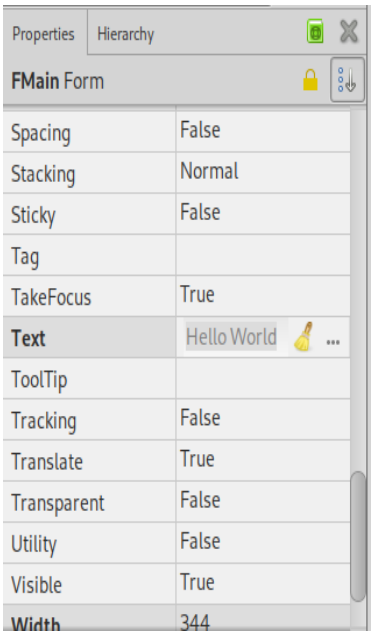
Στην συνέχεια θα δούμε κάποια βασικά στάδια της σχεδίασης της πρώτης μας εφαρμογής.

Σε αυτό το πρώτο μας παράδειγμα θα κάνουμε τα εξής:

1. Θα βάλουμε ένα κείμενο στην επικεφαλίδα της φόρμας.
2. Θα σχεδιάσουμε ένα κουμπί.
3. Θα σχεδιάσουμε μία ετικέτα

Βήμα 1ο:
Για να εμφανίσουμε την φόρμα FMain στην προβολή σχεδίασης μπορούμε να κάνουμε διπλό κλικ στην κλάση FMain του εξερευνητή έργου, ή με δεξί κλικ να εμφανίσουμε το μενού του εξερευνητή έργου και να επιλέξουμε Open.

Το αποτέλεσμα θα είναι να εμφανιστεί η φόρμα στην προβολή σχεδίασης.



Εικόνα 35: Η καρτέλα ιδιοτήτων και η περιοχή επεξήγησης ιδιοτήτων

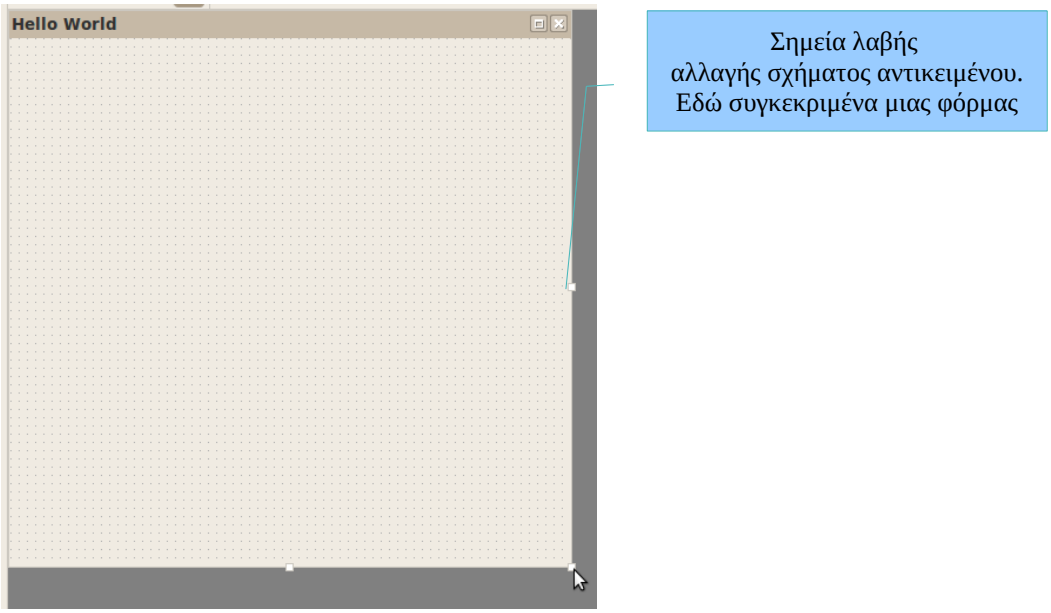
Μόλις ένα αντικείμενο έχει την εστίαση (focus) στη προβολή σχεδίασης η λίστα ιδιοτήτων στα δεξία μας δείχνει τις ιδιότητες του αντικειμένου αυτού.

Για να καθορίσουμε τα χαρακτηριστικά ενός αντικειμένου το μόνο που έχουμε να κάνουμε είναι να έχουμε διαβάσει την τεκμηρίωση της Gambas για το αντικείμενο αυτό και να ξέρουμε ποιες ιδιότητες του θα ορίσουμε με τις δικές μας τιμές.

Από το πάνω δεξί μέρος στην καρτέλα των ιδιοτήτων (Properties), βρίσκουμε την ιδιότητα (property) Text.

Κάνοντας κλικ στο **Text** μπορούμε να γράψουμε κείμενο. Εδώ γράφουμε “Hello World” Παρατηρούμε ότι καθώς γράφαμε το κείμενο, στην γαλάζια επικεφαλίδα της φόρμας το κείμενο αναγραφόταν.
Η ιδιότητα Text ορίζει την επικεφαλίδα της φόρμας.

Αν θέλουμε να αλλάξουμε το μέγεθος της φόρμας στην προβολή σχεδίασης στις άκρες της φόρμας και στην κάτω δεξιά γωνία υπάρχουν σημεία λαβής.
Κρατώντας το αριστερό πλήκτρο του ποντικιού πατημένο και σέρνοντας το ποντίκι ενός σημείο λαβής αλλάζουμε το σχήμα της φόρμας.



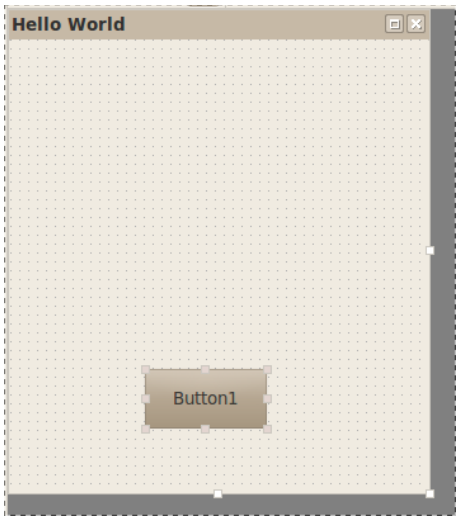
Εικόνα 36: Σημεία λαβής αλλαγής μεγέθους μιας φόρμας

Αυτό ισχύει και για κάθε αντικείμενο πάνω σε μια φόρμα.

Βήμα 2ο:
Το επόμενο στάδιο είναι να σχεδιάσουμε κάποια αντικείμενα ελέγχου (controls) στην φόρμα μας.

Από την εργαλειοθήκη επιλέγουμε το κουμπί.

Πηγαίνουμε το ποντίκι στην φόρμα στην γκρίζα περιοχή με τις τελείες.
Κρατάμε το αριστερό πλήκτρο του ποντικιού πατημένο και σέρνουμε το ποντίκι σχηματίζοντας ένα κουμπί.

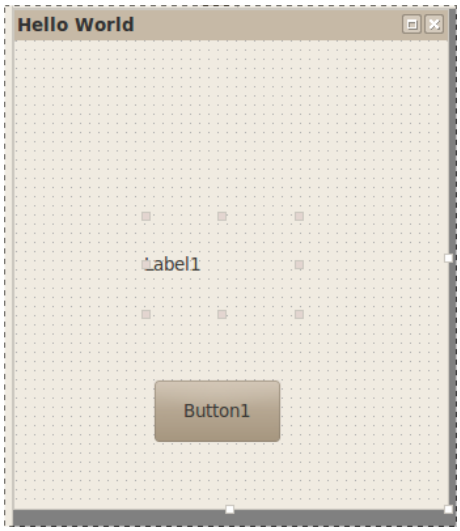


Εικόνα 37: Σχεδίαση ενός κουμπιού

Στην συνέχεια πηγαίνουμε στην λίστα ιδιοτήτων και θέτουμε στην ιδιότητα Text την τιμή Button1

Βήμα 3ο: Το ίδιο κάνουμε και για την ετικέτα.

Και έτσι έχουμε το αποτέλεσμα όπως φαίνεται παρακάτω.



Εικόνα 38: Το αποτέλεσμα της σχεδίασης

Στην συνέχεια πηγαίνουμε στην λίστα ιδιοτήτων και θέτουμε στην ιδιότητα Text την τιμή Label1.

Αυτός ήταν ο σχεδιασμός της φόρμας μας.

Εικονίδια εντολών σχεδίασης.

Όταν βρισκόμαστε στην προβολή σχεδίασης μιας φόρμας το περιβάλλον ανάπτυξης μας εμφανίζει μια μπάρα εικονιδίων σχετικά με ενέργειες της προβολής σχεδίασης. Η μπάρα αυτή είναι η παρακάτω.

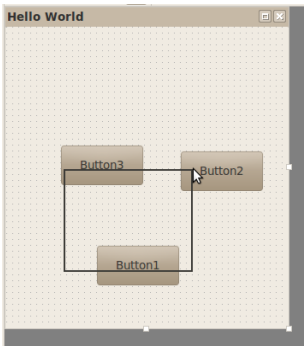
Εικόνα 39: Μπάρα εικονιδίων προβολής σχεδίασης

Οι εντολές που έχει η μπάρα της προβολής σχεδίασης είναι:

Αλλαγή προβολής μεταξύ σχεδίασης και κώδικα		Κεντράρισε Κάθετα	
Αποθήκευση		Επέλεξε γραμματοσειρά	
Ανανέωση προβολής		Έντονη γραφή	
Αναίρεση ενέργειας		Πλάγια γραφή	
Επανάληψη ενέργειας		Υπογραμισμένη γραφή	
Εμφάνιση συντάκτη μενού φόρμας	Menu editor...	Αύξηση μεγέθους γραμματοσειράς	
Εμφάνιση πλέγματος στην φόρμα		Μείωση μεγέθους γραμματοσειράς	
Εναλλαγή επεξήγησης εργαλείου πληροφοριών		Προκαθορισμένο μέγεθος γραμματοσειράς	

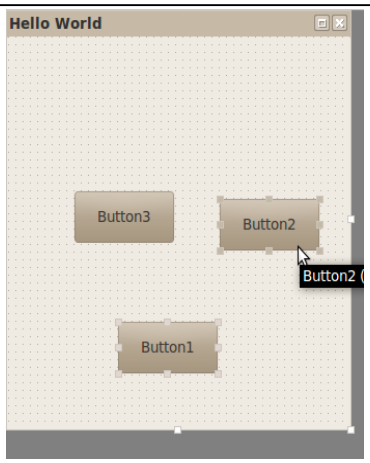
Φέρε στο προσκήνιο. Η εντολή αυτή φέρνει στο προσκήνιο ένα αντικείμενο που το έχει επικαλύψει άλλο.		Χρώμα φόντου	
Πάνε στο παρασκήνιο		Χρώμα προσκηνίου	
Κεντράρισε Οριζόντια		Κλείδωμα φόρμας	

Καθώς σχεδιάζουμε αντικείμενα μπορούμε να τα επιλέξουμε κρατώντας το αριστερό πλήκτρο του ποντικιού πατημένο και σέρνοντας το ποντίκι σχηματίζοντας ένα πλαίσιο επιλογής.



Εικόνα 40: Επιλογή αντικειμένων με σύρσιμο του ποντικιού

Η άλλη μέθοδος είναι να κρατήσουμε πατημένο το πλήτρο CTRL και να κάνουνε κλικ στα αντικείμενα που θέλουμε να επιλέξουμε.

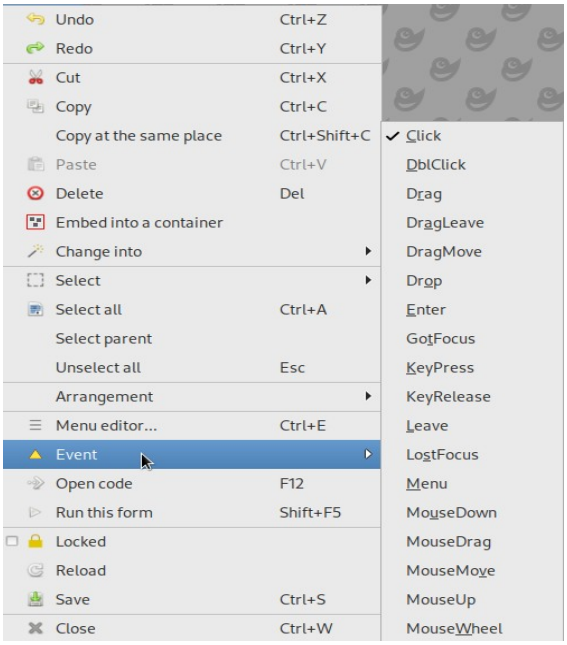


Εικόνα 41: Επιλογή αντικειμένων με πατημένο το πλήκτρο CTRL

Στην προβολή σχεδίασης έχουμε ακόμη ένα μενού. Το μενού που εμφανίζεται με δεξί κλικ σε επιλεγμένα αντικείμενα.

Αυτό είναι όπως φαίνεται δίπλα.

Στο μενού αυτό έχουμε τις εξής εντολές:



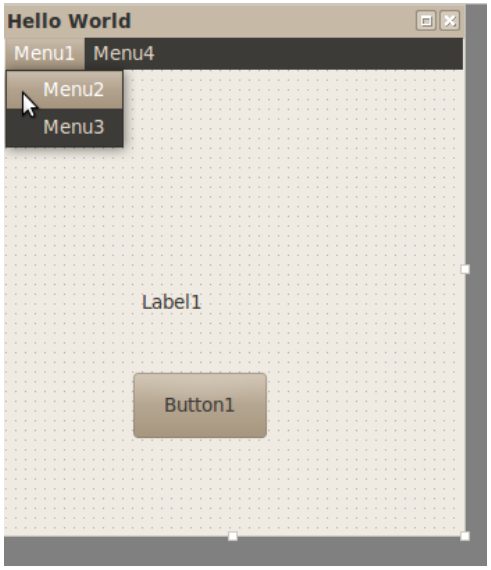
Εικόνα 42: Μενού με δεξί κλικ σε επιλεγμένα αντικείμενα

- Αναίρεση (Undo)
- Επανάληψη (Redo)
- Αποκοπή (Cut)
- Αντιγραφή (Copy)
- Αντιγραφή στο ίδιο μέρος (Copy at the same place)
- Επικόλληση (Paste)
- Διαγραφή (Delete)
- Ενσωμάτωση σε δοχέα (Embeded into container)
- Μετατροπή σε (Change Into)
- Επέλεξε (Select)
- Επέλεξε όλα (Select All)
- Επέλεξε τον γονέα (Select parent)
- Αποεπιλογή (Unselect all)
- Διαρύθμιση (Arrangement)
    - Φέρε στο προσκήνιο (Bring to foreground)
    - Πάνε στο παρασκήνιο (Send to background)
    - Οριζόντια διάταξη (Horizontal)
    - Διάταξη γραμμών (Rows)
    - Κάθετη διάταξη (Vertical)

    - Διάταξη κολώνων (Columns)
    - Κεντράρισε οριζόντια (Center horizontally)
    - Κεντράρισε κάθετα (Center vertically)
- Επεξεργαστής μενού (Menu editor)
- Συμβάν (Event)
- Άνοιγμα κώδικα (Open code)
- Τρέξε αυτήν την φόρμα (Run this form)
- Κλείδωσε (Locked)
- Επαναφόρτωσε (Reload)
- Αποθήκευσε (Save)
- Κλείσε (Close)

Ο συντάκτης μενού (Menu Editor)

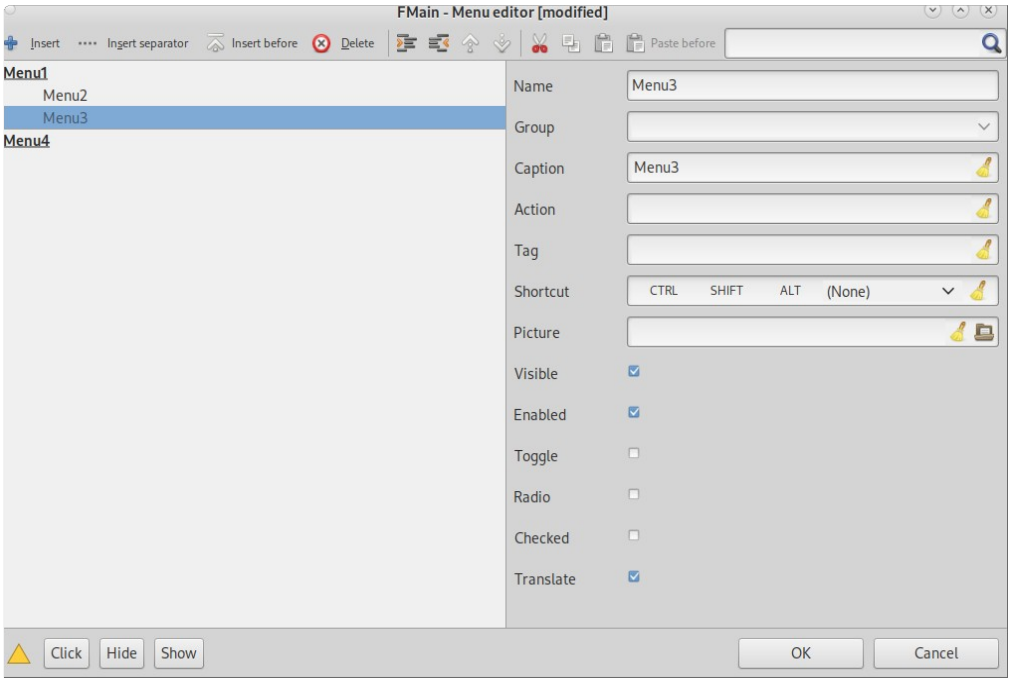
Η Gambas μας δίνει την δυνατότητα να κατασκευάσουμε μενού στην φόρμα μας.



Εικόνα 43: Το μενού μιας φόρμας

Αν από την μπάρα εντολών της προβολής σχεδίασης, επιλέξουμε το εικονίδιο του συντάκτη μενού

τότε εμφανίζεται η φόρμα σύνταξης μενού.



Εικόνα 44: Η φόρμα σύνταξης μενού

Για να δημιουργήσουμε ένα μενού πατάμε εισαγωγή (, Insert menu).

Με τα βελάκια πάνω - κάτω, ορίζουμε την σειρά των μενού. (, Move up, Move down).

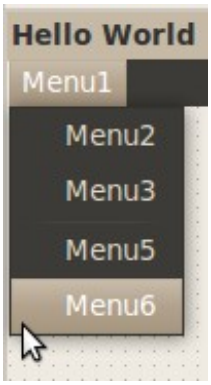
Με την εσοχή δεξιά – αριστερά, ορίζουμε πιο μενού θα είναι υπομενού ενός άλλου.
(, Indent, Unindent).

Όσα μενού εμφανίζονται με εσοχή (π.χ. ...Menu2) είναι υπομενού.

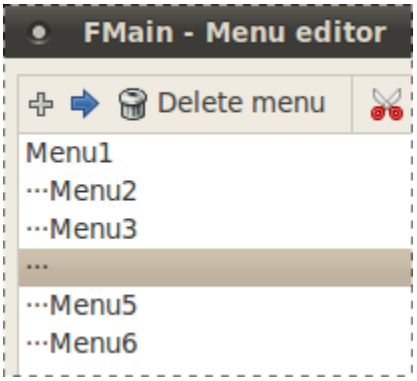
Ένα μενού έχει τις εξής ιδιότητες:

- Όνομα (Name)
- Ομάδα (Group)
- Ετικέτα (Caption)
- Ενέργεια (Action)
- Ετικέτα πληροφοριών (Tag)
- Πλήκτρο συντόμευσης (Shortcut)
- Εικόνα (Picture)
- Ορατό (Visible)
- Ενεργό (Enabled)
- Διακόπτης (Toggle)
- Ράδιοκουμπί (Radio)
- Μαρκαρισμένο (Checked)
- Μεταφραση (Translate)

Αν θέλουμε να χωρίσουμε οπτικά τα μενού μας με μια γραμμή τότε αφήνουμε την ετικέτα (Caption) κενή.



Εικόνα 45: η γραμμή διαχωρισμού όπως εμφανίζεται στα μενού



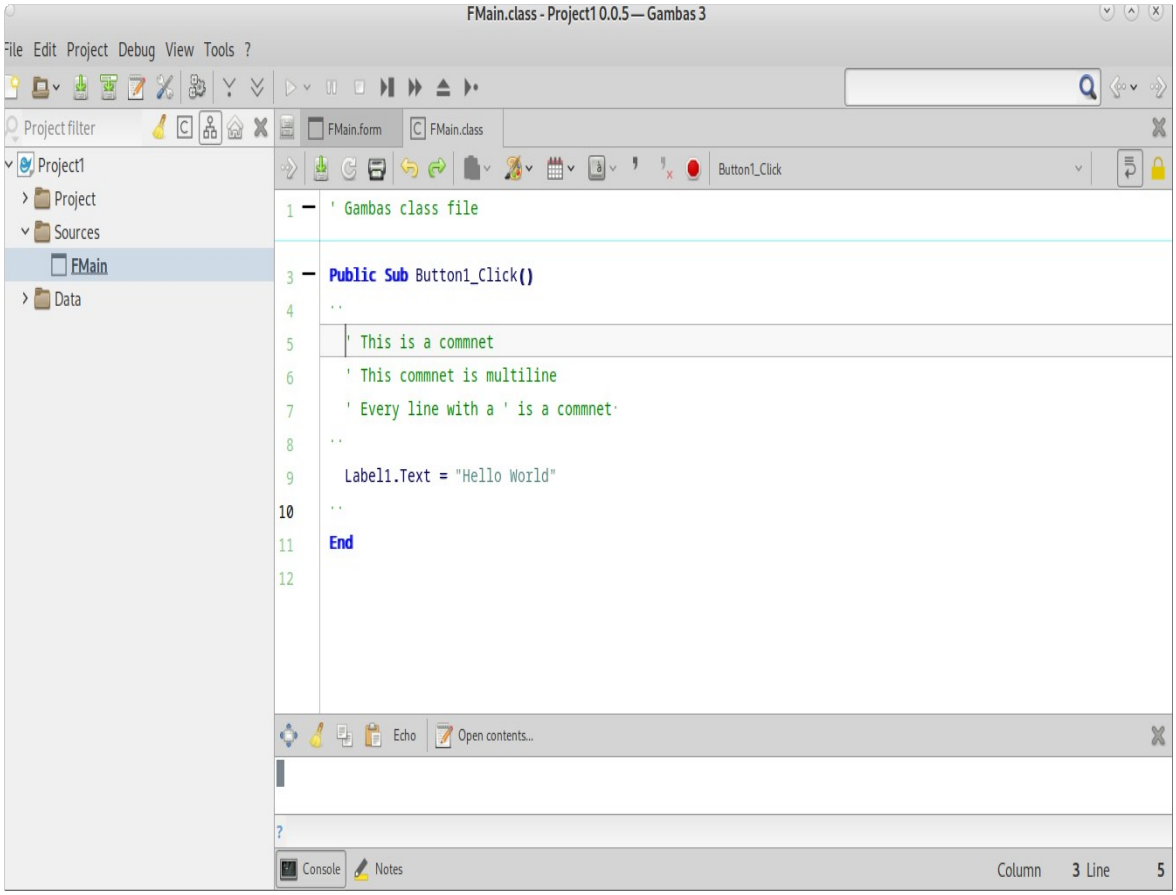
Εικόνα 46: Αφήνωντας το Caption κενό δημιουργούμε μια γραμμή διαχωρισμού στα μενού

Κεφάλαιο 4ο - Προβολή συγγραφής κώδικα

Αφού σχεδιάσουμε την φόρμα μπορούμε να αρχίσουμε να γράφουμε τον κώδικα της εφαρμογής.
Για να περάσουμε στην προβολή σύνταξης κώδικα έχουμε τους εξής τρόπους:

Κάνουμε διπλό κλικ στο αντικείμενο, αυτή η ενέργεια μας μεταφέρει σε ένα προκαθορισμένο συμβάν συνήθως στο Click().
Κάνουμε δεξί κλικ στο αντικείμενο και επιλέγουμε ένα συμβάν από την λίστα Event.
Κάνουμε δεξί κλικ στον εξερευνητή έργου σε μία κλάση και διαλέγουμε Edit.
Πατάμε το εικονίδιο εναλλαγής προβολών σχεδίασης – κώδικα .

Το αποτέλεσμα θα είναι να μεταφερθούμε στην προβολή συγγραφής κώδικα.

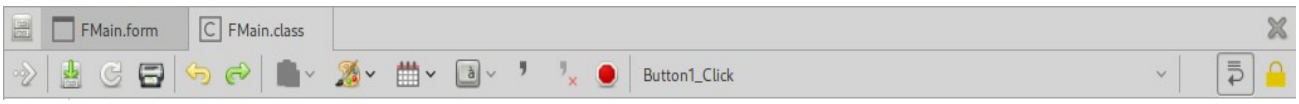


Εικόνα 47: Η προβολή συγγραφής κώδικα

Στο αριστερό μέρος του επεξεργαστή κειμένου κώδικα εμφανίζεται η αρίθμηση των γραμμών κώδικα.

Μπάρα εντολών συγγραφής κώδικα

Όταν είμαστε στην προβολή συγγραφής κώδικα εμφανίζεται η παρακάτω μπάρα εικονιδίων:



Εικόνα 48: Εικονίδια εντολών συγγραφής κώδικα.

Η μπάρα εικονιδίων συγγραφής κώδικα έχει τα εξής στοιχεία:

Εναλλαγή μεταξύ φόρμας / κώδικα

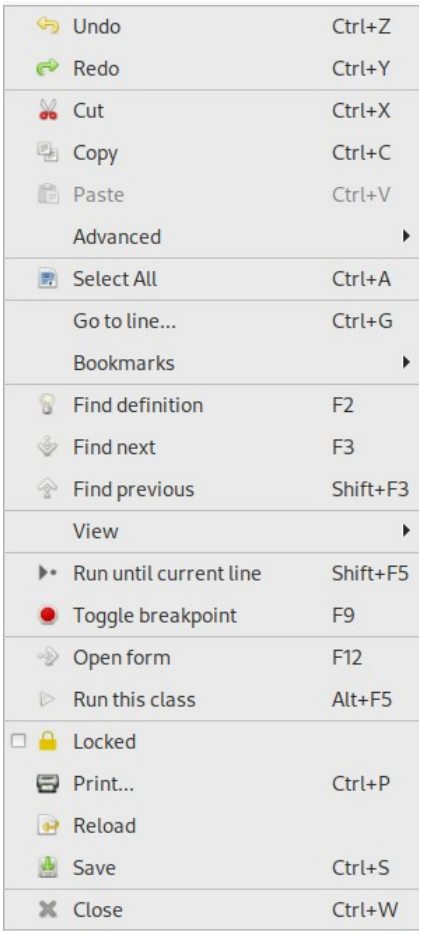
Εισαγωγή ημερομηνίας

Αποθήκευση		Εισαγωγή ειδικού χαρακτήρα	
Επαναφόρτωση		Σχολιασμός	
Εκτύπωση		Αφαίρεση σχολιασμού	
Αναίρεση		Εναλλαγή σημείου διακοπής	
Επανάληψη		Λίστα μεθόδων κλάσσης	Button1_Click
Επικόλληση		Αυτόματη αναδίπλωση λέξεων	
Εισαγωγή χρώμματος		Κλείδωμα	

Το μενού του ποντικιού στην προβολή συγγραφής κώδικα

Κατά την συγγραφή κώδικα μπορούμε να κάνουμε δεξί κλικ στην περιοχή συγγραφής του επεξεργαστή κειμένου και να επιλέξουμε από ένα αναδυόμενο μενού τις εξής εντολές:

- Αναίρεση (Undo)
- Επανάληψη (Redo)
- Αποκοπή (Cut)
- Αντιγραφή (Copy)
- Επικόλληση (Paste)
- Προχωρημένα (Advanced)
  - Κεφαλαία (Uppercase)
  - Μικρά (Lowercase)
  - Σχολιασμός (Comment)
  - Αναίρεση σχολιασμού (Uncomment)
  - Ειδική επικόλληση (Paste special)
  - Μορφοποίηση κώδικα (Format code)
  - Εισαγωγή χρώμματος (Insert color)
  - Εισαγωγή σημερινής ημερομηνίας (Insert current date)
- Επιλογή όλων (Select all)
- Πήγαινε στην γραμμή (Go to line)
- Σελιδοδείκτες (Bookmarks)
  - Ορισμός Σελιδοδείκτη (Define bookmark)
  - Καθαρισμός όλων των σελιδοδεικτών (Clear all bookmarks)
  - Προηγούμενος σελιδοδείκτης (Previous bookmark)
  - Επόμενος σελιδοδείκτης (Next bookmark)
- Εύρεση πρότασης (Find definition)
- Εύρεση επόμενου (Find next)
- Εύρεση προηγούμενου (Find previous)
- Πήγαινε πίσω (Go back)
- Προβολή (View)
  - Αυτόματη αναδίπλωση κειμένου (Automatic word wrap)
  - Χωρίς διαχωρισμό (No split)
  - Οριζόντιος διαχωρισμός (Horizontal split)
  - Κάθετος διαχωρισμός (Vertical split)
- Τρέξε μέχρι την τρέχουσα γραμμή (Run until current line)
- Εισαγωγή διακοπής (Toggle breakpoint)
  Εισάγει μία διακοπή εκτέλεσης στην τρέχουσα γραμμή του κώδικα
- Άνοιξε την φόρμα (Open form)
  Εναλλάσσει την προβολή στην προβολή σχεδίασης



Εικόνα 49: Το μενού του ποντικιού στην προβολή συγγραφής κώδικα

- Τρέξε αυτήν την κλάσση (Run this class)
- Κλειδωμα (Locked)
- Εκτύπωση (Print)
- Επαναφόρτωση (Reload)
- Αποθήκευση (Save)
- Κλείσιμο (Close)

Επιστροφή στο παράδειγμα του Project1, “Hello World”

Σε αυτό το πρώτο μας παράδειγμα κάναμε τα εξής:

1. Βάλαμε ένα κείμενο στην επικεφαλίδα της φόρμας.
2. Σχεδιάσαμε ένα κουμπί.
3. Σχεδιάσαμε μία ετικέτα

4. Τώρα θα γράψουμε κώδικα που όταν πατιέται το κουμπί η ετικέτα θα αναγράφει “Hello World”.

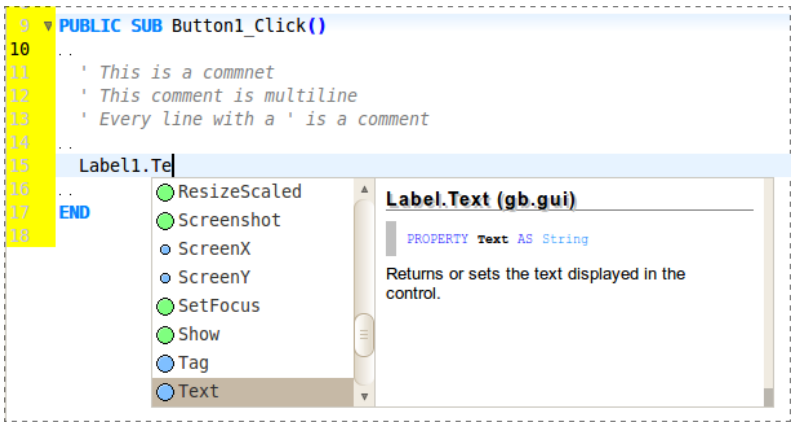
Βήμα 4ο:
Τώρα ας δούμε πώς θα γράψουμε κώδικα και να προγραμματίσουμε το κλικ του κουμπιού ώστε να αλλάζει το κείμενο της ετικέτας.

Κάθε αντικείμενο έχει ένα όνομα. Το όνομα μπορούμε να το ορίσουμε στην καρτέλα των ιδιοτήτων αριστερά πάνω.

Εμείς δεν θα αλλάξαμε τα ονόματα των αντικειμένων και εδώ χρησιμοποιούμε τα ονόματα που δίνει η γλώσσα εξ' ορισμού.
Έτσι έχουμε τα:

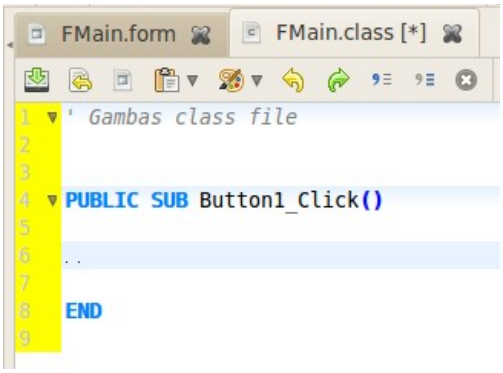
FMain, η φόρμα μας
Button1, το κουμπί
Label1, η ετικέτα.

Στην εισαγωγή είπαμε ότι κάθε αντικείμενο έχει ιδιότητες, μεθόδους και συμβάντα.
Για να προσπελάσουμε αυτά τα χαρακτηριστικά των αντικειμένων γράφουμε στον επεξεργαστή κειμένου, που θα δούμε στην συνέχεια, το όνομα του αντικειμένου αμέσως μετά μια τελεία και η Gambas μας δίνει μια λίστα σε tooltip των μεθόδων, ιδιοτήτων και γεγονότων του αντικειμένου. Επίσης εμφανίζει και μια μικρή βοήθεια σχετικά με την ιδιότητα, μέθοδο ή συμβάν που γράφουμε.



Εικόνα 50: Το χαρακτηριστικό αυτόματης ολοκλήρωσης γραφής με βοήθεια της Gambas3

Με διπλό κλικ στο κουμπί της φόρμας μεταφερόμαστε στον επεξεργαστή κειμένου με έτοιμη την ρουτίνα του συμβάντος Click() και τον δρομέα να μας περιμένει να γράψουμε κώδικα.



Εικόνα 51: Ο επεξεργαστής κειμένου κώδικα

Το συμβάν “Click()” του κουμπιού Button1 ορίζεται με μια ρουτίνα SUB η οποία είναι δημόσια PUBLIC και τελειώνει με την λέξη END.

Η ρουτίνα ενός συμβάντος έχει ως όνομα το όνομα του αντικειμένου συνοδευόμενο από μια κάτω παύλα και παρενθέσεις.

Έτσι το όνομα του συμβάντος κλικ του κουμπιού Button1 είναι Button1_Click().

Ό,τι κώδικα γράψουμε στην ρουτίνα Button1_Click() θα εκτελεστεί όταν ο χρήστης πατήσει το κουμπί Button1.

Αυτή ήταν μια σύντομη περιγραφή της σύνταξης μιας ρουτίνας συμβάντος ενός αντικειμένου.

Εμείς σε αυτό το πρώτο πρόγραμμα θέλουμε να ορίσουμε ένα κείμενο στην ετικέτα Label1.

Μια ετικέτα έχει το κείμενο της στην ιδιότητα **Text**.
Έτσι για να αλλάξουμε το κείμενο της ετικέτας αρκεί να αλλάξουμε το περιεχόμενο της ιδιότητας **Text.**

Για να αλλάξουμε το περιεχόμενο της ιδιότητας όμως πρέπει να την προσπελάσουμε και πώς είπαμε προσπελάζουμε μια ιδιότητα αντικειμένου; Με την σύνταξη Label1.Text

Για να θέσουμε το κείμενο στην ιδιότητα Text αρκεί να πούμε στην Gambas θέσε την τιμή της ιδιότητας Text στο κείμενο “Hello World”. Αυτό γίνεται με τον τελεστή εκχώρησης ορίσματος ο οποίος είναι το =

Για να ορίσουμε ότι μία τιμή είναι ένα κείμενο γράφουμε την τιμή μας μεταξύ διπλών εισαγωγικών.

Έτσι αν τοποθετήσουμε όλα μαζί αυτά που είπαμε για να ορίσουμε το κείμενο της ετικέτας Label1 θα πρέπει να γράψουμε.

```
Label1.Text = “Hello World”
```

Απλό.

Τον κώδικα αυτόν θα πρέπει να τον γράψουμε μέσα στην ρουτίνα του κουμπιού με το συμβάν κλικ.
Έτσι τελικά θα πρέπει να έχουμε:

```
PUBLIC SUB Button1_Click()

  Label1.Text = "Hello World"

END
```

Αυτός ο κώδικας λέει:
“Όταν ο χρήστης κάνει κλικ στο κουμπί Button1 θέσε το κείμενο του Label1 σε **Hello World**.

Μπορείτε να σώσετε το project και να πατήσετε **Run** από την μπάρα εικονιδίων εντολών για να το δοκιμάσετε.

Το αποτέλεσμα θα είναι αυτό:

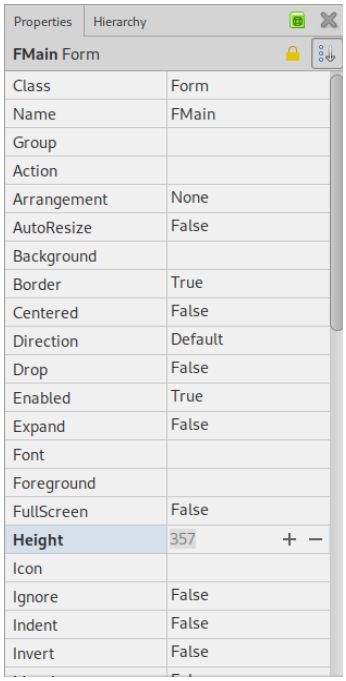
Εικόνα 52: Η εφαρμογή Hello World, πριν πατηθεί το κουμπί

Εικόνα 53: Η εφαρμογή Hello World, αφού πατηθεί το κουμπί

Ιδιότητες αντικειμένων και προσπέλαση τους

Στο παράδειγμα “Hello World” του Project1 έπρεπε να προσπελάσουμε την ιδιότητα κειμένου της ετικέτας Label1 ώστε να της δώσουμε τιμή.

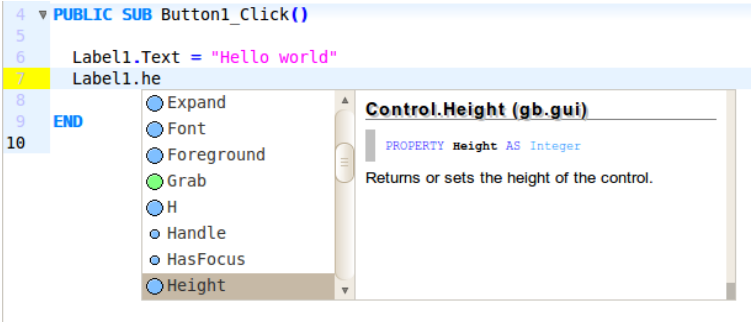
Όλα τα αντικείμενα έχουν ιδιότητες.
Σε κάποιες ιδιότητες θέτουμε τις τιμές τους κατά την σχεδίαση από την λίστα ιδιοτήτων του αντικειμένου στο πάνω δεξί μέρος του περιβάλλοντος ανάπτυξης εφαρμογών. Για να θέσουμε μια τιμή σε μια ιδιότητα, στην προβολή σχεδίασης, επιλέγουμε με αριστερό κλικ την ιδιότητα και πατάμε στο δεξί μέρος όπου φαίνεται η τιμή της. Για παράδειγμα αν θέλουμε να αλλάξουμε την τιμή του ύψους (Height) ενός αντικειμένου πατάμε εκεί που στην εικόνα γράφει **357** και θέτουμε την νέα τιμή.
Αν θέλουμε να προσπελάσουμε μία ιδιότητα, στην προβολή συγγραφής κώδικα, γράφουμε το όνομα του αντικειμένου μετά μία τελεία (**.**) και μας εμφανίζεται μία λίστα ιδιοτήτων και μεθόδων του αντικειμένου.



Εικόνα 54: Η λίστα Ιδιοτήτων (Properties)

Η λίστα αυτή εμφανίζεται αυτόματα κατά το γράψιμο του κώδικα και είναι ένα πολύ χρήσιμο χαρακτηριστικό “αυτόματης συμπλήρωσης” κώδικα και εμφάνισης βοήθειας.

Από την λίστα μπορούμε να επιλέξουμε την ιδιότητα που θέλουμε να διαβάσουμε ή να γράψουμε (read / write).



Εικόνα 55: Λίστα ιδιοτήτων αυτόματης συμπλήρωσης

Άλλες ιδιότητες επιτρέπεται να τις διαβάσουμε και να εκχωρήσουμε τις τιμές τους σε μεταβλητές ενώ σε άλλες μπορούμε να αλλάξουμε τις τιμές τους, δλδ να τις γράψουμε.
Για να θέσουμε μια τιμή σε μία ιδιότητα με κώδικα μετά την ιδιότητα βάζουμε τον τελεστή εκχώρησης (=) και την τιμή.

Για παράδειγμα:

```
Label1.Text = "Hello World"
Label1.Height = 30
```

Τέλος υπάρχουν και οι ιδιότητες που δεν μπορούμε να τις γράψουμε αλλά μόνο να τις διαβάσουμε. Αυτές είναι οι σταθερές (constants). Για περισσότερα στο κεφάλαιο των **Σταθερών, Μέρος 2ο - Κεφάλαιο 4ο**.

Συγχαρητήρια !

Μόλις σχεδιάσατε μια φόρμα με μερικά στοιχεία ελέγχου και τα προγραμματίσατε.
Αυτό είναι το πρώτο σας βήμα στον οπτικό αντικειμενοστραφή προγραμματισμό με την Gambas.

Μέρος 2ο, Η γλώσσα Basic της Gambas

Κεφάλαιο 1ο - 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 Gambas 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 η Gambas 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:

```
PUBLIC SUB Button1_Click()
```

και όχι

```
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 γράφατε 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 ' είναι ένα σχόλιο.

Παραδείγματα σχολίων:

```
PUBLIC SUB Button1_Click()

' This is a commnet
' This commnet is multiline
' Every line with a ' is a commnet

  Label1.Text = "Hello World"

END
```

Συνήθως οι προγραμματιστές χρησιμοποιούν τα σχόλια για να περιγράψουν τι κάνουν σε ένα project με εκατοντάδες σειρές ώστε όταν αυτό ανοιχτεί για επεξεργασία από άλλον προγραμματιστή ή μετά από καιρό να είναι δυνατόν εύκολα να κατανοηθεί.

Τα σχόλια τα χρησιμοποιούν επίσης οι προγραμματιστές για να δοκιμάσουν περιπτώσεις κώδικα όταν μία ενέργεια γίνεται με πολλούς τρόπους.

Για παράδειγμα:

```
DIM iCounter AS Integer = 0
DIM sCommand AS String
DIM iTotalFiles AS Integer = 0

  IF lswDublicates.Count > 0 THEN
    'Delete files
    iTotalFiles = lswDublicates.Count

    lswDublicates.MoveFirst
    FOR iCounter = 0 TO (iTotalFiles - 1)
       sCommand = "rm -f \"" &
lswDublicates[lswDublicates.Item.Key].Text & "\""
       SHELL sCommand WAIT
'      KILL lswDublicates[lswDublicates.Item.Key].Text
      lswDublicates.MoveNext
```

```
    NEXT

    lswDublicates.Clear

  ENDIF
```

Εδώ η γραμμή

' KILL lswDublicates[lswDublicates.Item.Key].Text

είναι ένα σχόλιο, αλλά αν δεν είχε την απόστροφο θα ήταν μια ενεργή πρόταση για την γλώσσα προγραμματισμού.

Η γραμμή αυτή τελεί την ίδια λειτουργία με τις γραμμές

sCommand = "rm -f \"" & lswDublicates[lswDublicates.Item.Key].Text & "\""
SHELL sCommand WAIT

απλά ο προγραμματιστής επέλεξε ένα τρόπο από τους δύο και τον άλλο τον κράτησε ως σχόλιο.

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

Κεφάλαιο 2ο - Τι είναι μια ρουτίνα.

Στο πρώτο μας πρόγραμμα αναφερθήκαμε στις ρουτίνες και είπαμε ότι αυτές μπορεί να είναι δημόσιες. Ας δούμε αυτό το θέμα.

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 Gambas έχουμε τα εξής είδη ρουτινών.

Ρουτίνες που δημιουργεί ο προγραμματιστής. Αυτές προσπελάζονται ως **SUB, FUNCTION, PROCEDURE**
Ρουτίνες των αντικειμένων όπου ονομάζονται μέθοδοι. Μια μέθοδος προσπελάζεται ως **SUB**.
Τα συμβάντα των αντικειμένων. Ένα event προσπελάζεται ως **SUB**.
Για παράδειγμα για ένα κουμπί (π.χ. Button1) έχουμε τα εξής συμβάντα.

Click DblClick Drag DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel

Όταν θέλουμε να γράψουμε κώδικα για να προγραμματίσουμε τι θα γίνει όταν ένα συμβάν λάβει μέρος, γράφουμε κώδικα στην ρουτίνα του συμβάντος.

Για παράδειγμα αν γίνει απλό κλικ στο κουμπί θα γράψουμε κώδικα στην ρουτίνα

```
PUBLIC SUB Button1_Click()
...
END
```

Αν ο προγραμματιστής πατήσει ένα πλήκτρο ενώ η εστίαση είναι στο κουμπί τότε θα γράψουμε κώδικα στη ρουτίνα

```
PUBLIC SUB Button1_KeyPress()
...
END
```

Όλα τα αντικείμενα έχουν συμβάντα και μεθόδους. Η τεκμηρίωση της Gambas περιγράφουν κάθε αντικείμενο και τα συμβάντα του και τις μεθόδους του. Καλό είναι να ανατρέχετε στην τεκμηρίωση όποτε σχεδιάζεται μια εφαρμογή ή θέλετε να γράψετε κώδικα για να δείτε τι σας προσφέρει κάθε αντικείμενο.

Κλήσεις ρουτινών

Πολλές φορές καθώς γράφουμε κώδικα πρέπει να πούμε στην Gambas τώρα να πατηθεί το κουμπί Button1.

Για να δώσουμε αυτήν την εντολή στην γλώσσα προγραμματισμού πρέπει να καλέσουμε την μέθοδο κλικ του Button1.

Αυτό γίνεται ως εξής:

```
...
Button1_Click()
...
```

Όταν ο διερμηνευτής διαβάσει την γραμμή αυτή θα οδηγήσει την ροή του προγράμματος στην ρουτίνα **PUBLIC SUB Button1_Click()** θα εκτελέσει τον κώδικα μέχρι να συναντήσει την λέξη **END** και μετά θα επιστρέψει πίσω στο σημείο από το οποίο ξεκίνησε.

Ας δούμε ένα παράδειγμα.

Σώστε το Project1 ως Project1a και γράψτε τον εξής κώδικα:

```
1   ' Gambas class file
2
3   PUBLIC SUB Form_Open()
4
5   Button1_Click()
6
7   END
8
9   PUBLIC SUB Button1_Click()
10
11  ' This is a commnet
12  ' This commnet is multiline
13  ' Every line with a ' is a commnet
14
15    Label1.Text = "Hello World"
16
17  END
```

Η αριστερή στήλη με τα νούμερα 1-16 είναι η αρίθμηση των γραμμών στον επεξεργαστή κειμένου. Για να την δείτε επιλέξτε από το μενού **Tools** → **Preferences** → **Editor** και θέστε την τιμή **Show Line Numbers** σε **Yes**. Κλείστε το παράθυρο **Preferences** και θα έχετε την αρίθμηση όπως φαίνεται στο παράδειγμα.

Ας δούμε το παράδειγμα αυτό.
Το πρόγραμμα ξεκινά στην γραμμή 1.
Η γραμμή αυτή είναι όμως ένα σχόλιο και ο διερμηνευτής αγνοεί την γραμμή αυτή.

Καθώς το πρόγραμμα εκτελείται η Gambas εμφανίζει την FMain.
Αυτή η ενέργεια είναι ένα συμβάν (event). Συγκεκριμένα κάθε φόρμα που εμφανίζεται καλεί την μέθοδο **Form_Open()**. Έτσι όταν ανοίγει και εμφανίζεται η FMain η Gambas καλεί την ρουτίνα PUBLIC SUB Form_Open().

Η ροή του προγράμματος τώρα περνάει στην γραμμή 3 όπου ξεκινά η μέθοδος Form_Open().
Η επόμενη γραμμή 5, **Button1_Click()** λέει στον διερμηνευτή να καλέσει την μέθοδο “κλικ” του Button1.

Έτσι η ροή του προγράμματος αλλάζει και ανακατευθύνετε στην γραμμή 9, PUBLIC SUB Button1_Click().

Οι γραμμές 11, 12, 13 είναι σχόλια και αγνοούνται από το διερμηνευτή.

Η γραμμή 15 ορίζει την τιμή της ιδιότητας Text του Label1.

Η ροή του προγράμματος φτάνει στην γραμμή 17 όπου η λέξη END ορίζει το τέλος της μεθόδου Button1_Click().

Η ροή του προγράμματος επιστρέφει στην γραμμή 6 αμέσως μετά από το σημείο από όπου κλήθηκε η ρουτίνα Button1_Click().
Συνεχίζει στην επόμενη γραμμή όπου συναντά την λέξη END και αυτή ορίζει το τέλος της ρουτίνας Form_Open() και έτσι το πρόγραμμα τελειώνει τις κλήσεις των ρουτινών-μεθόδων.

Αυτό ήταν ένα παράδειγμα πώς από μια ρουτίνα πηγαίνουμε σε μια άλλη και ποια είναι η ροή του προγράμματος.

Αν τρέξετε αυτό το πρόγραμμα θα δείτε κατευθείαν το κείμενο “Hello World” στην ετικέτα σαν να είχατε πατήσει το Button1.

Ορισμοί ρουτινών

Τώρα ας δούμε πώς ορίζουμε μια ρουτίνα.

Η Gambas αναγνωρίζει τρία είδη ρουτινών. Τις υπορουτίνες SUB ή PROCEDURE και τις συναρτήσεις FUNCTION.

Συνήθως οι υπορουτίνες είναι μέρη κώδικα αυτόνομα, όπως είπαμε, που δεν επιστρέφουν κάποια τιμή.
Οι συναρτήσεις από την άλλη είναι ρουτίνες που επιστρέφουν κάποια τιμή, για παράδειγμα έναν αριθμό ή ένα κείμενο.

Για παράδειγμα η μέθοδος Label1_Click() είναι μια ρουτίνα που όταν τελειώσει δεν επιστρέφει μία τιμή.

Αν έχουμε όμως μια συνάρτηση Sum() αυτή πριν την λέξη END θα έχει μια τιμή επιστροφής με την λέξη RETURN.

Η διαφορά όπως φαίνεται είναι ότι οι μέθοδοι δεν έχουν την λέξη RETURN να επιστρέφει τιμές.

Υπορουτίνες

Η σύνταξη για τον ορισμό των υπορουτινών είναι

```
[ STATIC ] { PUBLIC | PRIVATE } { PROCEDURE | SUB }
Identifier
(
[ [ BYREF ] Parameter AS Datatype [ , ... ] ] [ , ]
[ OPTIONAL [ BYREF ] Optional Parameter AS Datatype [ , ... ] ] [ , ] [ ... ]
)
...
END
```

Οι λέξεις μέσα σε [...] είναι προαιρετικές.
Οι λέξεις μέσα σε {...} είναι απαραίτητες.
Όσες λέξεις χωρίζονται με | σημαίνει ή την μία ή την άλλη λέξη.
Η υπογραμμισμένες λέξεις είναι κείμενο που γράφει ο προγραμματιστής.
Οι λέξεις με **έντονη** γραφή είναι λέξεις της γλώσσας Gambas.

Εμβέλεια υπορουτινών

Οι ρουτίνες είναι δημόσιες ή ιδιωτικές. Όταν είναι δημόσιες ορίζονται με την λέξη PUBLIC και όταν είναι ιδιωτικές με την λέξη PRIVATE.

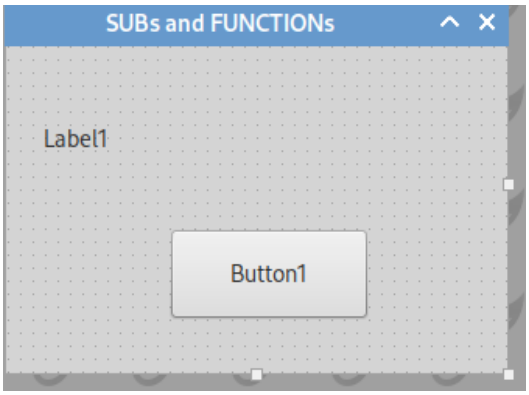
Όταν μια ρουτίνα είναι δημόσια μπορεί να κληθεί από μία άλλη κλάση όπως για παράδειγμα από μία άλλη φόρμα.

Έτσι αν θέλουμε να αλλάξουμε το κείμενο μιας ετικέτας της FMain από την φόρμα FForm2 θα πρέπει στην FMain να έχει οριστεί δημόσια η μέθοδος SetText() και στην φόρμα FForm2 να γράψουμε FMain.SetText().

Ας το δούμε με ένα παράδειγμα.

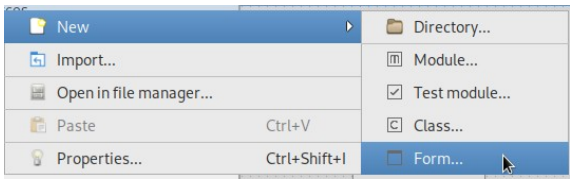
Δημιουργείστε όπως προηγουμένως ένα νέο project με όνομα Project2 και

- σχεδιάστε στην FMain μία ετικέτα Label με το όνομα Label1
- και ένα κουμπί με το όνομα Button1
- ορίστε την τιμή της ιδιότητας Text της FMain σε “SUBs and FUNCTIONs”



Εικόνα 56: Το αποτέλεσμα σχεδίασης

Δημιουργείστε μια νέα φόρμα την FForm2 κάνοντας δεξί κλικ στην δένδρο των αρχείων και επιλέγοντας New → Form.



Εικόνα 57: Δημιουργία μια νέας φόρμας

Συμπληρώστε στο παράθυρο που θα εμφανιστεί το όνομα της νέας φόρμας σαν FForm2.

Στην νέα φόρμα FForm2 απλά σχεδιάστε ένα κουμπί.

Στην FMain γράψτε τον παρακάτω κώδικα.

```
1  ' Gambas class file
2  PUBLIC SUB Button1_Click()
3
4    FForm2.Show
5
6  END
7
8  PUBLIC SUB SetText()
9
10   Label1.Text = "Hello again!"
11
12 END
```

Σε αυτόν τον κώδικα έχουμε δύο ρουτίνες.
Την μέθοδο **Button1_Click()** και μια δικιά μας υπορουτίνα με το όνομα **SetText**, την **SetText()**.

Η πρώτη μέθοδος εμφανίζει την FForm2 καλώντας την μέθοδο εμφάνισης της φόρμας FForm2. Η κλήση της μεθόδου εμφάνισης της φόρμας γίνεται με την **Fform2.Show** στην γραμμή 4.

Όταν λοιπόν πατηθεί το Button1 θα εμφανιστεί η δεύτερη φόρμα μας.

Η επόμενη υπορουτίνα, η **SetText()**, είναι απλά ένα αυτόνομο κομμάτι κώδικα που όταν καλείται αλλάζει το κείμενο στην ετικέτα **Label1**.

Αυτό γίνεται στην γραμμή 10 με την **Label1.Text = "Hello again!"**

Εδώ θα πρέπει να δώσουμε προσοχή!!! Για να μπορούμε να καλέσουμε την **SetText()** από άλλη κλάση, δλδ από μια άλλη φόρμα, η υπορουτίνα θα πρέπει να είναι δημόσια, δλδ να έχει οριστεί ως **PUBLIC**.

Στην FForm2 γράφουμε τον κώδικα.

```
1  ' Gambas class file
2  PUBLIC SUB Button1_Click()
3
4    FMain.SetText()
5
6  END
```

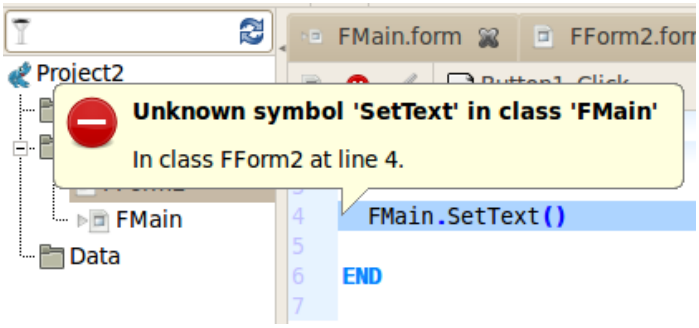
Σε αυτήν την υπορουτίνα στην γραμμή 4 καλούμε την SetText() της FMain.

Δοκιμάστε να τρέξετε το πρόγραμμα αυτό.
Θα εμφανιστεί πρώτα η FMain.
Θα πατήσετε τον Button1 και θα εμφανιστεί η FForm2.
Πατώντας το Button1 της FForm2 θα δείτε ότι αλλάζει το κείμενο της ετικέτας της FMain.

Αν στην FMain η υπορουτίνα SetText() ήταν ιδιωτική, δλδ είχε οριστεί ως

```
8  PRIVATE SUB SetText()
```

Και προσπαθούσατε να την καλέσετε από την άλλη φόρμα FForm2 θα παίρνατε ένα μήνυμα λάθους.



Εικόνα 58: Μήνυμα λάθους σύνταξης

Όταν μια υπορουτίνα είναι ιδιωτική έχει εμβέλεια μόνο μέσα στην κλάση που υπάρχει.
Όταν μια υπορουτίνα είναι δημόσια έχει εμβέλεια και στις άλλες κλάσεις και μπορεί να κληθεί από αυτές.

Για να δείτε πώς συμπεριφέρεται η ιδιωτική υπορουτίνα SetText() γράψτε τον παρακάτω κώδικα στην FMain.

```
1  ' Gambas class file
2  PUBLIC SUB Button1_Click()
3
4   SetText()
5
```

```
6   END
7
8   PRIVATE SUB SetText()
9
10    Label1.Text = "Hello again!"
11
12  END
```

Εδώ όταν πατήσετε το Button1 θα καλέσετε την SetText() και αυτή θα αλλάξει το κείμενο της ετικέτας **Label1**. Αυτό μπορείτε να το κάνετε διότι η SetText() είναι ιδιωτική και έχει εμβέλεια μέσα στην ίδια κλάση.
Δεν μπορείτε όμως από άλλη φόρμα να καλέσετε την SetText() όπως προηγουμένως.

Όταν μια υπορουτίνα έχει οριστεί ως **STATIC** έχει πρόσβαση μόνο στις μεταβλητές που έχουν οριστεί ως **STATIC**. Περισσότερα θα πούμε όταν αναφερθούμε στις μεταβλητές.

Κάθε υπορουτίνα θα παρατηρείσατε ότι μετά το όνομα της έχει παρενθέσεις (), δλδ SetText().

Εντός των παρενθέσεων μπορείτε να περνάτε μεταβλητές ώστε να τις χρησιμοποιήσετε μέσα στην υπορουτίνα.

Με άλλα λόγια οι παρενθέσεις είναι η είσοδος της υπορουτίνας.

Όταν υπάρχει η λέξη **OPTIONAL** η μεταβλητή είναι προαιρετική και μπορείτε να την παραλείψετε.

Η λέξη **AS** ορίζει τον τύπο των δεδομένων που ακολουθούν. Δλδ αν είναι κείμενο (string), αριθμός (integer, float), λογικός τύπος δεδομένου (boolean) κτλ

Συναρτήσεις

Η σύνταξη των συναρτήσεων είναι σχεδόν όμοια με την διαφορά ότι πρέπει να δηλώσουμε την λέξη FUNCTION και τον τύπο δεδομένων που επιστρέφει η συνάρτηση, δλδ στο τέλος του ορισμού της συνάρτησης γράφουμε ... **AS** Datatype

Για παράδειγμα αν έχουμε μια συνάρτηση που προσθέτει δύο αριθμούς ο τύπος επιστροφής είναι αριθμός.

Η σύνταξη των συναρτήσεων είναι

```
[ STATIC ] { PUBLIC | PRIVATE } { FUNCTION | PROCEDURE | SUB }
Identifier
(
[ [ BYREF ] Parameter AS Datatype [ , ... ] ] [ , ]
[ OPTIONAL [ BYREF ] Optional Parameter AS Datatype [ , ... ] ] [ , ] [ ... ]
)
AS Datatype
...
END
```

Η Gambas για λόγους ευελιξίας σας αφήνει να ορίσετε την συνάρτηση και με τις λέξεις **PROCEDURE** ή **SUB** εκτός από την λέξη **FUNCTION**.

Καλό όμως είναι να υιοθετήσετε ένα αυστηρό στυλ γραφής κώδικα και να χρησιμοποιείτε για συναρτήσεις την λέξη **FUNCTION**.
Αν όμως χρειαστείτε κάποια υπορουτίνα να λειτουργήσει σαν συνάρτηση η δυνατότητα αυτή υπάρχει.

Ας δούμε ένα παράδειγμα.

Δημιουργείστε ένα νέο project το Project3 με μια φόρμα και ένα κουμπί Button1.
Θέλουμε όταν πατάμε το κουμπί να καλείται μια συνάρτηση που θα προσθέτει δύο αριθμούς.

Στον κώδικα της φόρμας FMain γράψτε τα εξής:

```
1   ' Gambas class file
2   PUBLIC SUB Button1_Click()
3
4   DIM d AS Integer
5
6    d = Sum()
7    PRINT d
8
9   END
10
11  PUBLIC FUNCTION Sum() AS Integer
12
```

```
13  DIM a AS Integer
14  DIM b AS Integer
15  DIM c AS Integer
16
17  a = 1
18  b = 2
19  c = a + b
20  RETURN c
21
22  END
```

Στο παράδειγμα αυτό έχουμε δύο ρουτίνες την δημόσια μέθοδο **Button1_Click()** και την δημόσια συνάρτηση **Sum() AS Integer**

Η συνάρτηση **Sum()** επιστρέφει ένα ακέραιο αριθμό για αυτό στο τέλος της υπάρχει ο τύπος δεδομένων που επιστρέφει, δλδ το ...**AS Integer**

Εδώ ορίζουμε κάποιες μεταβλητές για να λειτουργήσει το παράδειγμα αυτό.

Ο πιο απλός τρόπος ορισμού μεταβλητής είναι με την λέξη **DIM** έπειτα το **όνομα** της μεταβλητής και ακολουθεί η λέξη **AS** και **ο τύπος** της μεταβλητής.

Εδώ θέλουμε ακέραιους και ο τύπος είναι integer.

Έτσι ορίζουμε τις μεταβλητές a, b, c, d.

Όταν ο προγραμματιστής πατήσει το Button1 τότε:

Στην γραμμή 4 ορίζεται μια μεταβλητή ακεραίου η d.

Στην γραμμή 6 καλείται η συνάρτηση Sum() που θα προσθέσει δύο αριθμούς.

Η ροή του προγράμματος πηγαίνει στην γραμμή 11 όπου καλείται η συνάρτηση Sum().

Στις γραμμές 13, 14, 15 ορίζονται οι μεταβλητές a, b, c ως ακέραιοι.

Στην γραμμή 17 η μεταβλητή a λαμβάνει την τιμή 1

Στην γραμμή 18 η μεταβλητή b λαμβάνει την τιμή 2

Στην γραμμή 19 η μεταβλητή c λαμβάνει την πρόσθεση των μεταβλητών a και b.

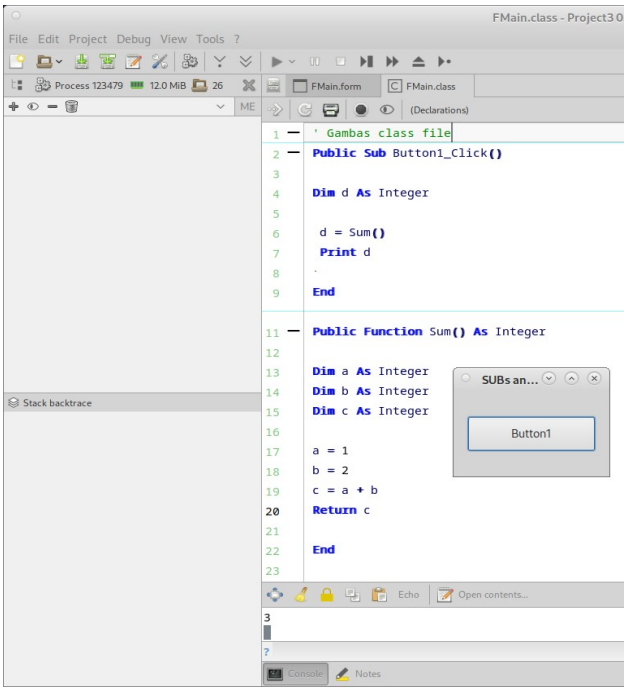
Στην γραμμή 20 η συνάρτηση επιστρέφει την τιμή της μεταβλητής c

Η ροή του προγράμματος επιστρέφει στην γραμμή 6 και η μεταβλητή d λαμβάνει την τιμή που της δίνει η συνάρτηση Sum().

Στην γραμμή 7 τυπώνεται στην έξοδο του προγράμματος η τιμή της μεταβλητής d. Και στην γραμμή 9 το πρόγραμμα τελειώνει.

Η έξοδος του προγράμματος φαίνεται στο κάτω μέρος του περιβάλλοντος ανάπτυξης (IDE) εκεί που λέει Console (τερματικό), διότι αυτή είναι η εξ ορισμού έξοδος.

Σε αυτό το παράδειγμα είδαμε πώς λειτουργεί μία συνάρτηση και πώς επιστρέφει μία τιμή.



Εικόνα 59: Η εξ' ορισμού έξοδος του προγράμματος

Αν ανάμεσα στις παρενθέσεις () της συνάρτησης Sum() θέταμε μεταβλητές τότε θα μπορούσαμε να έχουμε μία είσοδο στην συνάρτηση και οι μεταβλητές a, b, να ορίζονται με διαφορετικές τιμές και όχι με τις προκαθορισμένες που έχουν σε αυτό το παράδειγμα.
Για να δούμε αυτό το παράδειγμα δημιουργείστε το Project4 όπως προηγουμένως με μία φόρμα και ένα κουμπί.

Ο κώδικας της φόρμας εδώ είναι.

```
1   ' Gambas class file
2   PUBLIC SUB Button1_Click()
3
4   DIM d AS Integer
5
6     DIM q AS Integer
7     DIM w AS Integer
8
9     q = 1
10    w = 2
11
12  d = Sum(q, w)
13  PRINT d
14
15  END
16
17  PUBLIC FUNCTION Sum(a AS Integer, b AS Integer) AS Integer
```

```
18  DIM c AS Integer
19
20  c = a + b
21  RETURN c
22
23  END
24
```

Εδώ όπως βλέπετε στην μέθοδο Button1_Click ορίζουμε τις μεταβλητές q, w με τις τιμές 1, 2.

Στην γραμμή 12 καλούμε την συνάρτηση Sum(a AS Integer, b AS Integer) AS Integer.

Η συνάρτηση όμως τώρα δέχεται τιμές ακεραίων (integer) και τις τοποθετεί σε δικές της μεταβλητές a, b. Οι τιμές a, b που δέχεται μια συνάρτηση ονομάζονται ορίσματα της συνάρτησης.

Οι a, b είναι μεταβλητές ιδιωτικές της συνάρτησης Sum(). Αυτό σημαίνει ότι δεν έχουν εμβέλεια έξω από την συνάρτηση.

Στην γραμμή 21 γίνεται η πρόσθεση και η μεταβλητή c λαμβάνει την τιμή της προσθέσεως.

Στην γραμμή 12 η μεταβλητή d λαμβάνει την επιστροφή της συνάρτησης Sum() και τέλος στην γραμμή 13 η μεταβλητή d τυπώνεται στην έξοδο του τερματικού.

Σε αυτό το παράδειγμα η συνάρτηση Sum() είναι έτσι φτιαγμένη που δέχεται τιμές από τον προγραμματιστή.
Θα μπορούσε κάλλιστα με κάποιες τροποποιήσεις οι τιμές αυτές να δίνονται από κουτιά κειμένου (Textbox) στην φόρμα FMain.

Διοχέτευση παραμέτρων ByRef

Στις υπορουτίνες και τις συναρτήσεις μπορούμε όπως είδαμε να περνάμε παραμέτρους.
Αν έχουμε μια τιμή εκχωρημένη σε μια μεταβλητή και περάσουμε ως παράμετρο την μεταβλητή αυτή στην ρουτίνα περνάμε ένα αντίγραφο της τιμής της.
Αν στην ρουτίνα επεξεργαστούμε αυτήν την παράμετρο δεν θα αλλάξει η τιμή της αρχικής μεταβλητής.

Αυτή η μέδοθος διοχέτευσης τιμών ονομάζεται διοχέτευση κατά τιμή ή αλλιώς ByVal (By Value).

Η Visual Basic της Microsoft έδινε την δυνατότητα να περνάμε τις τιμές σε ρουτίνες όχι μόνο κατά τιμή αλλά κατά αναφορά στην μνήμη RAM.

Μια μεταβλητή έχει τοποθετήσει τα δεδομένα της στην μνήμη RAM αν ο προγραμματιστής έχει πρόσβαση στην διεύθυνση της μνήμης RAM, όπου φυλάσσονται τα δεδομένα, και αλλάξει αυτά, η μεταβλητή θα έχει μια νέα τιμή εκχωρημένη στην διεύθυνση της μνήμης RAM.

Η απευθείας διοχέτευση των τιμών μεταβλητών με αναφορά στην διεύθυνση μνήμης ονομάζεται διοχέτευση κατά αναφορά ή αλλιώς ByRef (By Reference).

Η Gambas3 για να δώσει την δυνατότητα σε προγραμματιστές της Visual Basic να κάνουν μεταφορά τον κώδικα τους στην Gambas κράτησε την λέξη BYREF αλλά όχι την λειτουργία της όπως ήταν στην VB.

Στην VB η BYREF αναφερόταν σε μια θέση μνήμης μιας μεταβλητής, στην Gambas3 η BYREF απλά τροποποιεί την μεταβλητή κατά το τέλος της ρουτίνας εκχωρώντας σε αυτήν νέα τιμή αν αυτή έχει αλλάξει.

Το αποτέλεσμα είναι το ίδιο. Μια μεταβλητή που έχει διοχετευτεί BYREF σε μια ρουτίνα αν η ρουτίνα της εκχωρήσει μια νέα τιμή η μεταβλητή θα έχει την νέα τιμή.

Για να κάνει αυτή την τροποποίηση η Gambas λειτουργεί ως εξής.

- Υπάρχει μια μεταβλητή με μια τιμή στην μνήμη π.χ. η sResult.
- Δηλώνουμε μια ρουτίνα και διοχετεύουμε BYREF την sResult, δλδ έχουμε την GetData(ByRef sResult).
- Η Gambas3 ωθεί στην μνήμη την sResult ως το πρώτο όρισμα
- Η ρουτίνα εκχωρεί νέα τιμή στην sResult
- Η Gambas3 μετά το τέλος της GetData() δεν απελευθερώνει την μνήμη που είναι η sResult.
- Μετά το τέλος της GetData() λαμβάνεται η τιμή από την μνήμη και εκχωρείται στην sResult.

Για να χρησιμοποιήσουμε μια τιμή BYREF πρέπει να δηλώσουμε την ρουτίνα με όρισμα BYREF αλλά και κατά την διοχέτευση της τιμής να δηλώσουμε ότι η τιμή είναι BYREF.

Μια δήλωση ρουτίνας με BYREF ορίσματα δηλώνει ότι **μπορούμε** να περάσουμε κατά αναφορά παραμέτρους.
Μια κλήση ρουτίνας με BYREF παραμέτρους δηλώνει ότι **θέλουμε** οι παράμετροι να διοχετευτούν κατά αναφορά.
Αν η ρουτίνα έχει δηλωμένα ορίσματα BYREF αλλά κάνουμε μια κλήση χωρίς να περάσουμε παραμέτρους BYREF τότε αυτές οι τιμές δεν θα περαστούν BYREF.

Ο διερμηνευτής της Gambas3 ελέγχει κατά την κατάσταση εκτέλεσης αν χρησιμοποιείτε την διοχέτευση BYREF και αν η ρουτίνα το επιτρέπει. Για αυτό τον λόγο θα πρέπει η χρήση κατά αναφορά να γίνεται τόσο κατά την δήλωση της ρουτίνας όσο και κατά την διοχέτευση παραμέτρων.
Για να δούμε το χαρακτηριστικό της “κατά αναφορά” διοχέτευσης παραμέτρων δημιουργήστε το Project5.

Το Project5 έχει ένα αρχείο μονάδας (Module) για κλάση εκκίνησης.
Όταν χρησιμοποιούμε αρχεία μονάδων ως κλάσεις εκκίνησης θα πρέπει να έχουμε μια ρουτίνα Main() από όπου θα ξεκινά το πρόγραμμα.

Ο κώδικας του Project5 είναι ο εξής:

```
1  ' Gambas module file
2
3  Public Sub Main()
4
5    Dim Number1 As Integer
6
7    Number1 = 10
8
9    IncrementVariable1(Number1)
10
11   Print Number1
12
13   IncrementVariable2(Number1)
14
15   Print Number1
16
17   IncrementVariable2(ByRef Number1)
18
19   Print Number1
20
21 End
22
23 Private Sub IncrementVariable1(Num1 As Integer)
24   'ByVal sub
25
26   Num1 = Num1 + 1
27
28 End
29
```

```
30  Private Sub IncrementVariable2(ByRef Num2 As Integer)
31    'ByRef sub
32
33    Num2 = Num2 + 1
34
35  End
```

Ανάλυση:
Έχουμε τρεις ρουτίνες. Τις Main(), IncrementVariable1(Num1 As Integer), IncrementVariable2(ByRef Num2 As Integer).

Στην IncrementVariable2(ByRef Num2 As Integer) έχουμε δηλώσει ότι μπορούμε να διοχετεύσουμε έναν ακέραιο (integer) κατά αναφορά (ByRef).

Στην IncrementVariable1(Num1 As Integer) έχουμε δηλώσει ότι μπορούμε να διοχετεύσουμε έναν ακέραιο αλλά μόνο κατά τιμή.

Το πρόγραμμα ξεκινώντας αναζητά την ρουτίνα Main() και ξεκινά την εκτέλεση από αυτήν.
Στην γραμμή 5 δηλώνουμε μια μεταβλητή με το όνομα Number1 και ότι σε αυτήν τη μεταβλητή μπορεί να εκχωρηθούν ακέραιοι (integer).

Στην γραμμή 7 εκχωρούμε τον ακέραιο 10 στην μεταβλητή.

Στην συνέχεια θα καλέσουμε μία μία τις ρουτίνες και θα παρατηρήσουμε ποιες από τις κλήσεις που θα κάνουμε αλλάζουν την τιμή της μεταβλητής.

Η πρώτη κλήση γίνεται στην γραμμή 9 όπου καλούμε την IncrementVariable1() διοχετεύοντας την την μεταβλητή Number1.

Στην συνέχεια η ροή του προγράμματος περνά στην γραμμή 23.
Η τιμή της Number1 εκχωρείται στην Num1.
Στην γραμμή 26 αυξάνουμε κατά μία μονάδα την Num1 και έτσι η τιμή της γίνεται 11.
Η τιμή όμως έχει περάσει ByVal και έτσι η Number1 δεν αλλάζει.
Η ρουτίνα τελειώνει στην γραμμή 28 και η ροή του προγράμματος επιστρέφει στην γραμμή 11.

Στην γραμμή 11 τυπώνουμε στην προκαθορισμένη έξοδο την τιμή της Number1 η οποία αφού δεν άλλαξε είναι 10.

Στην συνέχεια στην γραμμή 13 καλούμε την IncrementVariable2. Στην IncrementVariable2 έχουμε δηλώσει ότι μπορούμε να διοχετεύσουμε τιμή κατά αναφορά.
Ωστόσο στην γραμμή 13 δεν διοχετεύουμε την Number1 κατά αναφορά.
Η ροή του προγράμματος τώρα μεταφέρεται στην γραμμή 30.
Η τιμή της Number1 εκχωρείται στην Num2.
Στην γραμμή 33 αυξάνουμε κατά μία μονάδα την Num1 και έτσι η τιμή της γίνεται 11.
Η τιμή όμως έχει περάσει ByVal και έτσι η Number1 δεν αλλάζει.

Η ρουτίνα τελειώνει στην γραμμή 35 και η ροή του προγράμματος επιστρέφει στην γραμμή 15.

Στην γραμμή 15 τυπώνουμε στην προκαθορισμένη έξοδο την τιμή της Number1 η οποία αφού δεν άλλαξε είναι 10.

Στην συνέχεια στην γραμμή 17 καλούμε την IncrementVariable2. Αυτή την φορά διοχετεύουμε την τιμή κατά αναφορά.
Η ροή του προγράμματος τώρα μεταφέρεται στην γραμμή 30.
Η τιμή της Number1 εκχωρείται στην Num2.
Στην γραμμή 33 αυξάνουμε κατά μία μονάδα την Num1 και έτσι η τιμή της γίνεται 11.
Η τιμή όμως έχει περάσει ByRef και έτσι η Number1 θα αλλάξει.
Η ρουτίνα τελειώνει στην γραμμή 35.
Μετά το τέλος της ρουτίνας επειδή η Number1 έχει περάσει ByRef της εκχωρείται η τιμή της Num2 και έτσι τώρα η Number1 έχει την τιμή 11
Η ροή του προγράμματος επιστρέφει στην γραμμή 19.

Στην γραμμή 19 τυπώνουμε στην προκαθορισμένη έξοδο την τιμή της Number1 η οποία άλλαξε και είναι 11.

Έτσι στην κονσόλα της Gambas μπορούμε να δούμε τις τιμές 10, 10, 11 από τις κλήσεις των τριών ρουτινών.

Σε αυτό το σημείο καλύψαμε την σύνταξη των ρουτινών και την περιγραφή της.

Ας δούμε τώρα το θέμα των μεταβλητών.

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 Gambas.

Τύπος (Datatype)	**Περιγραφή τιμών**	**Τιμή εξ' ορισμού**	**Μέγεθος στην μνήμη**
`Boolean`	Αλήθεια (True) ή Ψέμα (False).	`FALSE`	1 byte
`Byte`	0...255	`0`	1 byte
`Short`	-32.768 ... +32.767	`0`	2 bytes
`Integer`	-2.147.483.648 ... +2.147.483.647	`0`	4 bytes
`Long`	-9.223.372.036.854.775.808 ... +9.223.372.036.854.775.807	`0`	8 bytes
`Single`	Όπως ένας *float στην* C. -1,7014118+38 ..... +1,7014118E+38	`0.0`	4 bytes
`Float`	*Όπως ένας double* στην C. -8.98846567431105E+307 ..... +8.98846567431105E+307	`0.0`	8 bytes
`Date`	Ημερομηνία και χρόνος που η κάθε μία τιμή αποθηκεύεται σε έναν ακέραιο (integer).	`NULL`	8 bytes
`String`	Ένα μήκος χαρακτήρων κειμένου.	`NULL`	4 bytes
`Variant`	Οποιοσδήποτε τύπος.	`NULL`	12 bytes
`Object`	Ανώνυμη αναφορά σε οποιοδήποτε αντικείμενο.	`NULL`	4 bytes
`Pointer`	Μία τιμή διεύθυνσης μνήμης. Ο τύπος αυτός είναι απόλυτα συνώνυμος με τον τύπο Integer των συστημάτων 32 bits.	`0`	4 bytes σε 32 bits συστήματα, 8 bytes σε 64 bits συστήματα.

Κάθε εγγενής τύπος δεδομένων έχει τον αντίστοιχο του τύπο δεδομένων πίνακα.
Ένας τύπος δεδομένων πίνακας αποτελείται από το όνομα του τύπου δεδομένων και ακολουθούν αγκύλες [].

Δλδ έχουμε τους τύπους δεδομένων πινάκων Boolean[], Byte[], Short[], Integer[], Single[], Float[], String[], Date[], Variant[], Pointer[], and Object[].
Επιπρόσθετα κάθε τύπος δεδομένων έχει τον αντίστοιχο του σε πίνακα.

Για παράδειγμα String[][] για έναν πίνακα πινάκων κειμένου, Collection[] για έναν πίνακα συλλογών κτλ.

Για περισσότερα για τους πίνακες στο κεφάλαιο 8.

Μια ματιά στους τύπους δεδομένων.

Boolean

Ο τύπος Boolean (λογικός τύπος δεδομένων) ορίζει τις τιμές TRUE (ΑΛΗΘΕΙΑ) και FALSE (ΨΕΜΑ).
Στις γλώσσες προγραμματισμού συνήθως η τιμή TRUE αντιστοιχεί στην αριθμητική τιμή 1 και η τιμή FALSE στην αριθμητική τιμή 0. Στην Gambas όμως η τιμή TRUE αντιστοιχεί στην αριθμητική τιμή -1 και η FALSE στην αριθμητική τιμή 0.

Έτσι αν θέλετε να μετατρέψετε μια αριθμητική τιμή σε λογικό τύπο δεδομένων, όπως θα δούμε αργότερα στο κεφάλαιο περί μετατροπών από τύπο σε τύπο, θα χρησιμοποιήσετε το -1 και 0 για το τύπο δεδομένων Boolean.

Byte

Ο τύπος byte (ψηφιολέξη) ορίζει τιμές ενός μη προσημασμένου ακεραίου (unsigned integer) με τιμές από το 0 έως το 255.
Μη προσημασμένος ακέραιος σημαίνει ότι η τιμή είναι ένας ακέραιος αριθμός χωρίς πρόσημο + ή - .
Οι μη προσημασμένοι (unsigned) αριθμοί, δλδ χωρίς πρόσημο είναι πάντα θετικοί. Ένα αριθμός προσημασμένος (signed), δλδ με πρόσημο μπορεί να είναι αρνητικός ή θετικός.

Ορισμένες γλώσσες προγραμματισμού δίνουν την δυνατότητα να ορίσει ο προγραμματιστής αν ο αριθμός θα είναι προσημασμένος ή όχι. Τέτοια γλώσσα είναι η C, και C++.

Στην Gambas δεν υπάρχει αυτή η δυνατότητα. Η Gambas ορίζει τους τύπους δεδομένων **Short, Integer, Long, Single, Float** ως προσημασμένους και το τύπο **Byte** ως μη προσημασμένο εξ' ορισμού. Με άλλα λόγια ενώ στην C++ μπορείτε να ορίσετε έναν μη προσημασμένο integer, στην Gamnas δεν μπορείτε διότι θεωρείτε προσημασμένος.

Short

Ο τύπος Short ορίζει τιμές ενός προσημασμένου ακεραίου (signed) με τιμές από το -32.768 έως το +32.767.

Integer

Ο τύπος Integer (ακέραιος) ορίζει τιμές ενός προσημασμένου ακεραίου (signed) με τιμές από το -2147483648 έως το +2147483647 ή αλλιώς από -2^31 και 2^31-1.

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

Η δεκαδική μορφή

Η δεκαδική μορφή είναι γνωστή σε όλους. Ένας προσημασμένος ακέραιος αριθμός σε δεκαδική μορφή αποτελείται από τα ψηφία 0 έως 9 και το πρόσημο – ή + .

Παράδειγμα:

```
X = 1
X = 26
X = 65290
```

Η 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 & ή &H ή &h

Παράδειγμα:

```
X = &001
X = &001A
X = &H001A
X = &h001A
X = &HFF0A
```

Η δυαδική μορφή

Η δυαδική μορφή ενός αριθμού αποτελείται από τα ψηφία 0, 1.
Ένας δυαδικός αριθμός αρχίζει με &X ή &x

Παράδειγμα:

```
X = &X00101
X = &x0011101
```

Long

Ο τύπος Long ορίζει τιμές ενός προσημασμένου ακεραίου (signed) με τιμές από το -9.223.372.036.854.775.808 έως το +9.223.372.036.854.775.807

Single

Ο τύπος Single ορίζει τιμές ενός προσημασμένου (signed) αριθμού κινητής υποδιαστολή με τιμές από το -1,7014118E+38 έως το +1,7014118E+38

Float

Ο τύπος Float ορίζει τιμές ενός προσημασμένου (signed) αριθμού κινητής υποδιαστολή με τιμές από το -8.98846567431105E+307 έως το +8.98846567431105E+307

Date

Ο τύπος Date (ημερομηνία) ορίζει την τιμή της ημερομηνίας και του χρόνου.
Ο τύπος αυτός περιέχει δύο ακέραιους αριθμούς.
Ο πρώτος ορίζει τις ημέρες από το 8000 π.Χ. μέχρι την τιμή που δίνεται.
Ο δεύτερος ορίζει τα μικροδευτερόλεπτα (microseconds) από τα μεσάνυχτα μέχρι την τιμή που δίνεται.

Η ημερομηνία αποθηκεύεται σε μονάδες μέτρησης UTC και μετατρέπεται στο τοπικό σύστημα μέτρησης όταν πρόκειται να εμφανιστεί στην οθόνη.

Αν μετατρέψετε την τιμή μιας ημερομηνίας σε αριθμό θα έχετε:
Παράδειγμα:

```
PRINT Now;; "-";; CFloat(Now)

07/13/2005 02:20:08 - 2485669.138981
```

String

Ο τύπος String (στοιχειοσειρά) ορίζει μια μεταβλητή μεταβλητού μήκους από χαρακτήρες.
Ο πρώτος χαρακτήρας στην στοιχειοσειρά απαριθμείται σαν 1 και όχι σαν 0.

Variant

Ο τύπος Variant ορίζει οποιαδήποτε τύπο μεταβλητής.
Εργασίες με μεταβλητές τύπου Variant είναι πιο αργές από αυτές με άλλους τύπους δεδομένων.

Object

Ο τύπος Object (αντικείμενο) ορίζει μια ανώνυμη αναφορά σε κάποιο αντικείμενο που ορίζει ο προγραμματιστής.

Εργασίες με ανώνυμα αντικείμενα είναι πιο αργές από αυτές που η κλάση των αντικειμένων είναι γνωστή.

Τα αντικείμενα στα οποία δεν υπάρχει πλέον αναφορά αυτόματα απελευθερώνονται από την μνήμη.

Στην Gambas δεν υπάρχουν αντικείμενα ανενεργά ώστε να χρειαστεί εύρεση τους και διαγραφή τους από την μνήμη. Δλδ δεν υπάρχει “garbage collector”.

Προσοχή αν δημιουργήσετε αλληλοεξαρτώμενες αναφορές σε αντικείμενα αυτά δεν θα αφαιρεθούν από την μνήμη ποτέ.
Για παράδειγμα αν ένα αντικείμενο A έχει αναφορά σε ένα αντικείμενο B και το B αναφορά στο A, τότε αυτά τα αντικείμενα δεν θα αφαιρεθούν από την μνήμη ποτέ.

`Pointer`

Ο τύπος Pointer (δείκτης διεύθυνσης μνήμης) ορίζει μια αριθμητική τιμή ακεραίου (integer) σε 32bits συστήματα ή μια αριθμητική τιμή Long σε 64bits συστήματα που αντιστοιχεί σε μια διεύθυνση μνήμης.

Δηλώνοντας μεταβλητές

Εμβέλεια μεταβλητών

Οι μεταβλητές μπορούν να οριστούν με εμβέλεια σε όλη την κλάση (καθολικές) ή με τοπική εμβέλεια μέσα σε μια υπορουτίνα (τοπικές), δλδ με εμβέλεια μέσα σε μια μέθοδο, συνάρτηση κτλ.

Όταν ορίζουμε μεταβλητές με εμβέλεια σε όλη την κλάση τις ορίζουμε στην αρχή της κλάσης.
Όταν ορίζουμε μεταβλητές με εμβέλεια σε μια υπορουτίνα τις ορίζουμε στην αρχή της υπορουτίνας.

Οι μεταβλητές με εμβέλεια σε όλη την κλάση είναι προσπελάσιμες από όλες τις υπορουτίνες της κλάσης.
Οι τοπικές μεταβλητές με εμβέλεια σε μια υπορουτίνα είναι προσπελάσιμες μόνο σε εντολές εντός της υπορουτίνας.

Σύνταξη ορισμού μεταβλητών

Σύνταξη καθολικών μεταβλητών.

Η σύνταξη γίνεται στην αρχή της κλάσης πριν από κάθε υπορουτίνα.

```
[ STATIC ] { PUBLIC | PRIVATE } Identifier AS Datatype [ = Expression ]
```

Όταν μια μεταβλητή οριστεί ως **STATIC** η μεταβλητή είναι προσπελάσιμη για κάθε αντικείμενο της κλάσης.

Όταν μια μεταβλητή οριστεί ως **PUBLIC** είναι προσπελάσιμη και από άλλες κλάσεις.

Όταν μια μεταβλητή οριστεί ως **PRIVATE** είναι προσπελάσιμη μόνο από κλάση που ορίζεται.

Ως Identifier είναι το όνομα που δίνει ο προγραμματιστής στην μεταβλητή.

Ο προγραμματιστής αφού ορίσει το όνομα που επιθυμεί θα πρέπει να ορίσει τον τύπο δεδομένων Datatype όπως αυτοί περιγράφτηκαν στην προηγούμενη ενότητα.

Ο προγραμματιστής μπορεί να θέσει μια τιμή στην μεταβλητή κατά τον ορισμό της.
Για να γίνει αυτό μετά τον ορισμό της ακολουθεί ο τελεστής εκχώρισης = και η τιμή.

Παράδειγμα:

```
'Ορισμοί καθολικών μεταβλητών
STATIC PUBLIC GridX AS Integer
STATIC PRIVATE bGrid AS Boolean
PUBLIC Name AS String
PUBLIC sMyArray AS String[]
PRIVATE Control AS Object

'Ορισμοί καθολικών μεταβλητών με αρχικές τιμές
PRIVATE Languages AS String[] = [ "fr", "it", "es", "de", "ja"
]
PUBLIC Name as String = “Peter”
```

Εναλλακτικός τρόπος σύνταξης καθολικών μεταβλητών

```
[ STATIC ] { PUBLIC | PRIVATE } Identifier AS NEW Class
(Arguments ...)

ή

[ STATIC ] { PUBLIC | PRIVATE } Identifier AS NEW Native_
Datatype [ Array_dimensions ... ]
```

Παράδειγμα:

```
STATIC PRIVATE Tasks AS NEW List
PRIVATE MyCollection AS NEW Collection(gb.Text)

PRIVATE Words AS NEW String[WORD_MAX * 2]
PUBLIC Matrix AS NEW Float[3, 3]
```

Σύνταξη τοπικών μεταβλητών.

```
DIM Identifier AS Datatype [ = Expression ]
```

Η σύνταξη τοπικών μεταβλητών αποτελείται από τα εξής μέρη:

Η σύνταξη αρχίζει με την λέξη **DIM**
Ακολουθεί το όνομα της μεταβλητής Identifier
Ακολουθεί η δήλωση του τύπου δεδομένων με το **AS** Datatype
Αν ο προγραμματιστής επιθυμεί να θέσει τιμή στην μεταβλητή ακολουθεί ο τελεστής εκχώρησης = και η τιμή.
Η εκχώρηση της τιμής είναι προαιρετική.

Τι είναι ένας Προσδιορισμός, Identifier

Ένας προσδιορισμός είναι μια ακολουθία χαρακτήρων που χρησιμοποιείται για να αναπαραστήσει ένα όνομα μιας μεταβλητής, μιας κλάσης, μιας μεθόδου, μιας ιδιότητας, μιας σταθεράς ή ενός συμβάντος.

- Ο πρώτος χαρακτήρας του προσδιορισμού πρέπει να είναι γράμμα του αλφαβήτου (a-z), είτε μία κάτω παύλα (_), είτε ένα σύμβολο του δολαρίου ($).
- Οι επόμενοι χαρακτήρες μπορεί να είναι από το αλφάβητο, είτε αριθμητικά ψηφία, είτε κάτω παύλα, είτε σύμβολο του δολαρίου, ή ένα σύμβολο ερωτήσεων (?).

Παρακάτω φαίνονται παραδείγματα αποδεκτών προσδιορισμών.

- `A`
- `i`
- `Xyz1972`
- `Null?`
- `$Global_Var`
- `_DoNotUse`
- `Event_Handler`

Μερικοί προσδιορισμοί στην Gambas είναι δεσμευμένες λέξεις κλειδιά, και έτσι δεν μπορούν να χρησιμοποιηθούν για ονόματα μεταβλητών.

Ωστόσο η Gambas δίνει την δυνατότητα να χρησιμοποιήσετε μια δεσμευμένη λέξη.

Για να εμποδίσετε τον διερμηνευτή να αναγνωρίσει έναν προσδιορισμό ως δεσμευμένη λέξη τον τοποθετείτε ανάμεσα από άγκιστρα {}.

Παράδειγμα:

```
DIM {Default} AS String
DIM {Dim} AS Collection
```

Παραδείγματα ορισμού μεταβλητών

```
'Ορισμοί τοπικών μεταβλητών
DIM iVal AS Integer
DIM sName AS String
DIM hObject AS Object
DIM hCollection AS Collection

'Ορισμοί τοπικών μεταβλητών και αρχικοποίηση τους
DIM bCancel AS Boolean = TRUE
DIM Languages AS String[] = [ "fr", "it", "es", "de", "ja" ]
DIM DefaultLanguage AS String = Languages[1]
```

Εναλλακτικός τρόπος δήλωσης τοπικών μεταβλητών

```
DIM Identifier AS NEW Class ( Arguments ... )
```

Παράδειγμα:

```
DIM aTask AS NEW String[]
DIM aCollection AS NEW Collection(gb.Text)
```

Δήλωση πολλών μεταβλητών.

Μπορείτε να δηλώσετε σε μία γραμμή πάνω από μία μεταβλητή ταυτόχρονα.

Παράδειγμα:

```
DIM Text AS String, Matrix AS NEW Float[3, 3]
DIM X, Y, W, H AS Integer
```

Ας δούμε λοιπόν με κάποια παραδείγματα όλα αυτά που περιγράψαμε.

Παράδειγμα εμβέλειας μεταβλητών.

Δημιουργήστε το Project6, το οποίο έχει μια φόρμα FMain, και ένα κουμπί Button1
Ο κώδικας στην φόρμα είναι ο εξής:

Παράδειγμα:

```
1  ' Gambas class file
2  PUBLIC sName AS String = "Kostas"
3  PRIVATE sLastName AS String = "Papadopoulos"
4
5  PUBLIC SUB Form_Open()
6
7  DIM sLocal AS String = "This is local var"
8
9  PRINT "I am in Form_Open"
10 PRINT sName
11 PRINT sLastName
12 PRINT sLocal
13
14 END
15
16 PUBLIC SUB Button1_Click()
17
18   PRINT "I am in Button1_Click"
19   PRINT sName
```

```
20    PRINT sLastName
21    PRINT sLocal
22
23  END
```

Στις γραμμές 2, 3 δηλώνουμε καθολικές μεταβλητές στοιχειοσειράς (String) και τις αρχικοποιούμε θέτοντας τες κάποια τιμή.
Στην γραμμή 7 δηλώνουμε τοπική μεταβλητή sLocal.

Στην γραμμή 5 ξεκινά η υπορουτίνα που εκτελείτε όταν ανοίγει η φόρμα.
Στην γραμμή 16 ξεκινά η υπορουτίνα που εκτελείται όταν πατιέται το κουμπί Button1.

Η εντολή PRINT τυπώνει στην εξ' ορισμού έξοδο (στο τερματικό) τα μηνύματα που ακολουθούν μετά την εντολή.

Όταν πατάμε να τρέξουμε το πρόγραμμα μας εμφανίζεται ένα μήνυμα λάθους που μας λέει ότι υπάρχει άγνωστη αναφορά μεταβλητής στην γραμμή 21.

Γιατί;
Διότι η μεταβλητή sLocal είναι δηλωμένη σαν τοπική μόνο για την ρουτίνα Form_Open().
Η Gambas δεν μπορεί να την διαβάσει εκτός ρουτίνας Form_Open().

Για να τρέξει το πρόγραμμα πρέπει να σβήσετε την γραμμή 21.

Αν διορθώσετε και τρέξετε το πρόγραμμα θα δείτε στο τερματικό τα εξής:

```
I am in Form_Open
Kostas
Papadopoulos
This is local var
```

Αν πατήσετε το κουμπί θα δείτε στην συνέχεια

```
I am in Button1_Click
Kostas
Papadopoulos
```

Όπως βλέπετε οι καθολικές μεταβλητές sName, sLastName που έχουν οριστεί στην αρχή της κλάσης είναι προσπελάσιμες από όλες τις υπορουτίνες της κλάσης, δλδ και από την Form_Open() και από την Button1_Click().

Η δε sName είναι δημόσια, PUBLIC και θα είναι προσπελάσιμη και από άλλες κλάσεις, όπως άλλες φόρμες.

Η δε sLastName είναι ιδιωτική, PRIVATE δεν θα είναι προσπελάσιμη από άλλες κλάσεις όπως άλλες φόρμες.

Για να δείτε αυτήν την διαφορά, αποθηκεύστε το πρόγραμμα ως Project6a και προσθέστε μία νέα φόρμα την FForm2.
Γράψτε τον κώδικα στην κλάση FForm2.

Παράδειγμα:

```
1  ' Gambas class file
2
3  PUBLIC SUB Form_Open()
4
5    PRINT "I am in FForm2"
6    PRINT FMain.sName
7    PRINT FMain.sLastName
8
9  END
```

Στον κώδικα της FMain προσθέστε την εντολή εμφάνισης της φόρμας FForm2 όταν θα πατηθεί το Button1

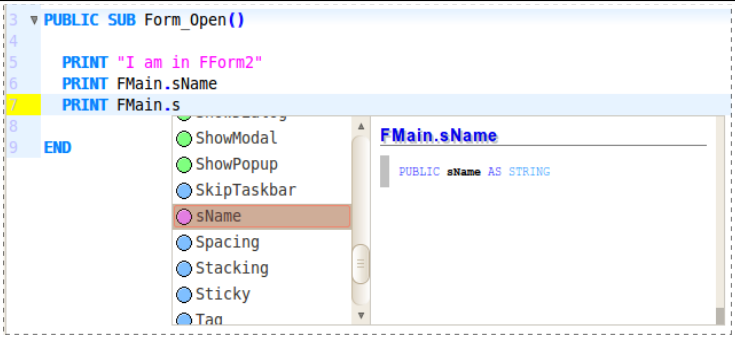
Παράδειγμα:

```
16 PUBLIC SUB Button1_Click()
17
18   PRINT "I am in Button1_Click"
19   PRINT sName
20   PRINT sLastName
21
22   FForm2.Show
23
24 END
```

Τρέξτε το πρόγραμμα και πατήστε το Button1 της FMain.
Θα γραφτεί ότι και προηγουμένως στο τερματικό
Μετά θα εμφανιστεί το μήνυμα των σειρών 5, 6 της FForm2
Και το πρόγραμμα θα σταματήσει από λάθος στην γραμμή 7 της FForm2.
Η Gambas αναφέρει ότι υπάρχει άγνωστη αναφορά σε μεταβλητή sLastName στην γραμμή 7 της FForm2.class .

Αυτό γίνεται διότι η sLastName είναι καθολική για την FMain αλλά ιδιωτική.
Ενώ η sName είναι καθολική αλλά δημόσια.

Αυτό θα έπρεπε να το αναγνωρίσετε επίσης από το συμβάν ότι καθώς γράφατε την γραμμή 7 της FForm2 η Gambas μετά το FMain. δεν θα σας εμφάνισε στην αυτόματη συμπλήρωση το όνομα της μεταβλητής sLastName.



Εικόνα 60: Η αυτόματη συμπλήρωση δεν εμφανίζει την sLastName

Για να τρέξει το πρόγραμμα θα πρέπει να αφαιρέσετε την γραμμή 7 από την κλάση Fform2.class

Παράδειγμα δηλώσεων μεταβλητών διαφόρων τύπων δεδομένων.

```
DIM bVar AS Boolean  = TRUE
DIM iNum AS Byte
DIM iA AS Short
DIM iRoads AS Integer
DIM iBig AS Long
DIM iArea AS Single
DIM fPi AS Float
DIM dDate AS Date
DIM sName AS String =”Kostas”
DIM vVar AS Variant
DIM hMyObject AS Object
DIM pRAM AS Pointer
```

Στυλ γραφής μεταβλητών

Όπως θα παρατηρείσατε τα ονόματα των μεταβλητών που χρησιμοποιήσαμε ήταν της μορφής bVar, iNum, fPi και όχι BVAR, INUM, FPI, ή bvar, inum, fpi.

Όποιο όνομα και να διαλέξετε είναι αποδεκτό.
Όμως στον προγραμματισμό και ειδικά όταν αυτός λαμβάνει χώρα σε ομαδικές συνεργασίες όπως στα πλαίσια μιας εταιρείας υιοθετούνται κάποια στυλ γραφής για να διευκολύνουν την ανάγνωση του κώδικα όταν αυτός εκτείνεται σε εκατοντάδες ή χιλιάδες γραμμές.

Ένα από τα στυλ γραφής μεταβλητών είναι η λεγόμενη “Ουγγρική σημειογραφία” (**Hungarian notation**). Το στυλ αυτό χρησιμοποιήθηκε για πρώτη φορά στην γλώσσα προγραμματισμού BCPL. Η BCPL δεν έχει τύπους δεδομένων αλλά έχει μόνο την λέξη μηχανής. Έτσι έπρεπε να ορισθεί ένας τρόπος ο προγραμματιστής να θυμάται τον τύπο δεδομένων της μεταβλητής από το όνομα της.
Ο τρόπος αυτός ήταν πριν το κύριο όνομα της μεταβλητής να γράφεται σε συντομογραφία ο τύπος της.

Για παράδειγμα αν έχουμε μια μεταβλητή για το μήκος, με όνομα Length, και είναι ένας ακέραιος integer τότε το όνομα της θα είναι intLength ή iLength ή ονομαζόμενη με άλλο παρόμοιο τρόπο.

Ο τρόπος αυτός θέλει το τύπο δεδομένων να αναγράφεται με μικρά και να ακολουθεί το όνομα της μεταβλητής με το πρώτο γράμμα κεφαλαίο (int + Length = intLength).

Αυτός ο τρόπος σημειογραφίας είναι ο **Hungarian notation.**

Έτσι σε ένα πρόγραμμα με πολλές γραμμές είναι βολικό να διαβάζουμε μια μεταβλητή και αμέσως να ξέρουμε τον τύπο της.

Το τρόπο αυτό ονομασίας μπορούμε να τον εφαρμόσουμε οπουδήποτε υπάρχει όνομα μέσα στην γλώσσα προγραμματισμού. Αυτό σημαίνει ότι και τα ονόματα των αντικειμένων μπορούν να έχουν αυτή την σημειογραφία.

Παρακάτω φαίνονται μερικά παραδείγματα αυτής της σημειογραφίας όπως την προτείνει ο Benoît Minisini.

Ονόματα μεταβλητών

Τα ονόματα μεταβλητών δεν μπορούν να έχουν κενά.

Όλες οι ιδιωτικές μεταβλητές μιας κλάσης θα πρέπει να αρχίζουν με $

Το πρώτο γράμμα είναι μικρό και περιγράφει τον τύπο δεδομένων της μεταβλητής:

Γράμμα	Τύπος
a	*Διάταξη (Array)*
b	Boolean
c	Collection
f	Float
h	Object (Handle, Αναφορά)
i	Integer, Long, Short, Byte.
n	Ένας ακέραιος που αποθηκεύει έναν αριθμό αντικειμένων.
s	String

Παραδείγματα

```
PRIVATE $iLast AS Integer
PRIVATE $sLast AS String
PRIVATE $hEditor AS Object
PRIVATE $sOldVal AS String
```

```
PRIVATE $bFreeze AS Boolean

PUBLIC SUB Form_Resize()

  DIM iWidth AS Integer
...
```

Ονόματα αντικειμένων φορμών

Όταν σχεδιάζετε ένα αντικείμενο σε μια φόρμα η Gambas δίνει ονόματα όπως "Label1" "Label2" ...

Πριν γράψετε οποιοδήποτε κώδικα αλλάξτε το όνομα του αντικειμένου σε "btnStart" ή "lstAddressSelect" κτλ.

Τα πρώτα δύο ή τρία γράμματα θα συνίσταται να γράφονται σύμφωνα με την ακόλουθη σημειογραφία:

Σημειογραφία	**Τύπος**
btn	Button, χρησιμοποιείστε btnOK btnCancel για τα OK, Cancel
chk	CheckBox
cvw	ColumnView
cmb	ComboBox
dwg	DrawingArea
dlg	FontChooser
edt	GambasEditor
frm	Frame
grd	GridView
spl	HSplit
iv	IconView
lbl	Label
lst	ListBox
lvw	ListView
pan	Panel
img	PictureBox
opt	RadioButton
svw	ScrollView
spb	SpinBox
tab	TabStrip

tim	Timer
txa	TextArea
txt	TextBox
txv	TextView
tbt	ToolButton
trv	TreeView
spl	VSplit

Ανακύκλωση μεταβλητών

Στις αριθμητικές μεταβλητές θα προσέξατε ότι ανάλογα τον τύπο 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 +32.76**8.**
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 Byte. 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 Byte αυξήσετε την μέγιστη τιμή της κατά ένα, δλδ από 225 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.

Παραδείγματα υπερχείλισης

Δημιουργείστε το Project7 και γράψτε τον εξής κώδικα.

Παράδειγμα:

```
1  ' Gambas class file
2
3  Public Sub Form_Open()
4
5  Dim iNum As Byte = 255
6  Dim iA As Short = 32767
7  Dim iRoads As Integer = 2147483647
8  Dim iBig As Long = 9223372036854775807
9  Dim iArea As Single = 1.7014118E+38
10 Dim fPi As Float = 8.98846567431105E+307
11
12 iNum += 1
13 iA += 1
14 iRoads += 1
15 iBig += 1
16 iArea += 1E+99
17 fPi += (1E+307) ^ 1E+304
18
19 Print iNum
20 Print iA
21 Print iRoads
22 Print iBig
23 Print iArea
24 Print fPi
25
26 End
27
28
29
30
```

Ο κώδικας εκτελείται όταν ανοίγει η φόρμα διότι έχει γραφτεί στο συμβάν Form_Open().

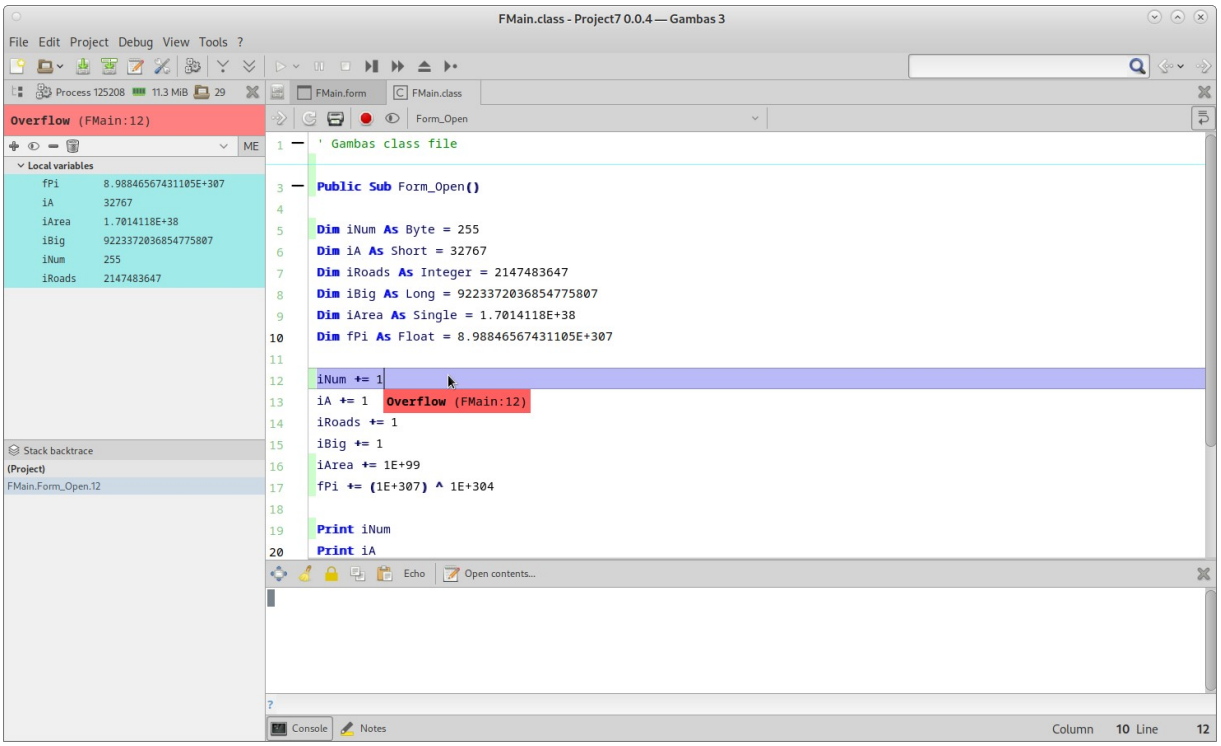
Στις γραμμές 4 έως 10 ορίζουμε μεταβλητές και τις θέτουμε τις ανώτερες τιμές που μπορούν να πάρουν.

Στις γραμμές 12 έως 16 αυξάνουμε τις τιμές των μεταβλητών κατά μία μονάδα.
Ο τελεστής += θα εξηγηθεί στο κεφάλαιο των τελεστών.

Στην γραμμή 17 αυξάνουμε την τιμή σε ένα πολύ μεγάλο αριθμό συγκεκριμένα προσθέτουμε τον 10^{99}
Το ίδιο κάνουμε και στην γραμμή 18
Στις γραμμές 20 έως 26 τυπώνουμε στην προκαθορισμένη έξοδο, δλδ στο τερματικό τις νέες τιμές των μεταβλητών.

Σε παλιότερες εκδόσεις της Gambas3 επιτρεπόταν η υπερχείλιση μεταβλητών και εμφάνιση των τιμών τους στο τερματικό.

Στην έκδοση 3.18.3 που εμείς δοκιμάζουμε όταν γίνεται υπερχείλιση οποιασδήποτε μεταβλητής εμφανίζεται ένα λάθος μεταγλώτισης.



Ειδικοί χαρακτήρες στοιχειοσειρών (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 στοιχειοσειρά.
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.

Χαρακτήρας διαφυγής	**Περιγραφή**	**Αντιστοιχία σε ASCII**
`\n`	Αλλαγή γραμμής	Chr$(13)
`\r`	Επαναφορά κεφαλής	Chr$(10)
`\t`	Οριζόντιο Tab	Chr$(9)
`\"`	Εισαγωγικά	Double quote
`\\`	Ανάποδη κάθετος	Backslash
`\xx`	Δεκαεξαδική μορφή	Chr$(&Hxx)

Παράδειγμα:

```
1  ' Gambas class file
2  PUBLIC SUB Form_Open()
3
4  DIM sVar AS String
5
6  sVar = " This is a text with \t \"escape\" characters \n "
7
8  PRINT sVar
9
10 END
11
```

Το αποτέλεσμα θα είναι:

```
This is a text with     "escape" characters
```

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 Gambas 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.
Οι συναρτήσεις αυτές ισχύουν και για μετατροπές τιμών σταθερών.

Αυτές είναι:

Συναρτήσεις μετατροπής.	
`CBool`	Μετατρέπει μια τιμή σε Boolean.
`CByte`	Μετατρέπει μια τιμή σε Byte.
`CDate`	Μετατρέπει μια τιμή σε Date.
`CFloat`	Μετατρέπει μια τιμή σε Float.
`CInt`	Μετατρέπει μια τιμή σε Integer.
`CLong`	Μετατρέπει μια τιμή σε Long.
`Conv$`	Μετατρέπει ένα string από ένα σετ χαρακτήρων σε άλλο σετ χαρακτήρων.
`CShort`	Μετατρέπει μια τιμή σε Short.
`CSng`	Μετατρέπει μια τιμή σε Single.
`CStr`	Μετατρέπει μια τιμή σε String.
`CVariant`	Μετατρέπει μια τιμή σε Variant
`DConv$`	Μετατρέπει ένα string από το σετ χαρακτήρων του συστήματος σε UTF-8 (desktop charset).
`SConv$`	Μετατρέπει ένα string από UTF-8 (desktop charset) σε σετ χαρακτήρων του συστήματος.
`Str$`	Μετατρέπει έναν αριθμό ή ημερομηνία σε string.
`Val`	Μετατρέπει ένα string σε αριθμό ή ημερομηνία.

CBool

Σύνταξη:

```
Result = CBool ( Expression ) AS Boolean
```

Μετατρέπει μια τιμή σε Boolean.
Η έκφραση είναι false αν η τιμή προς μετατροπή είναι:

- ένας boolean false
- η τιμή 0
- ένα string με μηδενικό μήκος
- μια κενή ημερομηνίας
- ένα κενό αντικείμενο.

Παράδειγμα:

```
1 PRINT Cbool(0)
2 PRINT CBool(1)
3 PRINT CBool("Peter");; CBool("")
4 PRINT CBool(NULL)
```

Έξοδος

```
False
True
True False
False
```

Σημείωση για την εντολή PRINT των παραδειγμάτων:
Αν μετά από μια έκφραση υπάρχει ; τότε ένα τέλος γραμμής τυπώνεται στην προκαθορισμένη έξοδο.
Αν μετά από μια έκφραση υπάρχει ;; τότε ένα κενό προστίθεται μεταξύ των εκφράσεων.
Αν μετά από μια έκφραση υπάρχει , τότε ένα tab προστίθεται μεταξύ των εκφράσεων.

Στην γραμμή 3 υπάρχει το ;; μεταξύ των εκφράσεων και έτσι αυτές τυπώνονται στην προκαθορισμένη έξοδο με ένα κενό διάστημα μεταξύ τους.

CByte

Σύνταξη:

```
Byte = CByte ( Expression AS Variant ) AS Byte
```

Μετατρέπει μια τιμή σε `Byte`

Η έκφραση Expression πρώτα μετατρέπεται σε integer. Μετά αν αυτός ο ακέραιος υπερχειλίζει το εύρος του τύπου δεδομένου byte συγκόπτεται.

Αν η έκφραση δεν μπορεί να μετατραπεί ένα μήνυμα λάθους προκύπτει.

Λάθη

Μύνημα	Περιγραφή
Type mismatch (#6)	Η έκφραση δεν μπορεί να μετατραπεί.

Παράδειγμα:

```
1 PRINT CByte("15")
2 PRINT CByte(10000)
3 PRINT CByte(TRUE)
```

Έξοδος

```
15
160
255
```

CDate

Σύνταξη:

```
Date = CDate ( Expression AS Variant ) AS Date
```

Μετατρέπει μια έκφραση (αριθμητική Float ή String) σε ημερομηνία Date ή string χρόνου.

Προσοχή! Οι τοπικές ρυθμίσεις δεν χρησιμοποιούνται από την συνάρτηση. Αυτό σημαίνει ότι η Gambas αναγνωρίζει για υποδιαστολή την τελεία (.) και όχι το κόμμα (,).

Λάθη.

Μήνυμα	Περιγραφή
Type mismatch (#6)	Η έκφραση δεν μπορεί να μετατραπεί.

Παράδειγμα:

```
1 PRINT CDate("12/30/1972 02:44:02")
2 PRINT Cdate(2484514)
```

Έξοδος

```
12/30/1972 02:44:02
05/14/2002 22:00:00
```

- Εάν η έκφραση είναι αριθμητική και κάτω από 1,0000 τότε μόνο τιμή χρόνου επιστρέφεται στην μορφή "hh:mm:ss" όπου h=ψηφίο ώρας, m=ψηφίο λεπτών, s=ψηφίο δευτερολέπτων.

  Παράδειγμα:

  ```
  PRINT Cdate(1.000)
  ```

  Έξοδος

  ```
  22:00:00
  ```

- Εάν η έκφραση είναι αριθμητική και πάνω από 1,0000 και το κλάσμα της είναι 0,00 τότε μόνο ημερομηνία επιστρέφεται στην Ευρωπαϊκή μορφή της "dd.mm.yy", όπου d=ψηφίο ημέρας, m=ψηφίο μηνός, y=ψηφίο έτους.
- Εάν η έκφραση είναι αριθμητική και πάνω από 1,0000 και το κλάσμα της δεν είναι 0,00 τότε ημερομηνία επιστρέφεται στην Ευρωπαϊκή μορφή της "dd.mm.yyyy hh:mm:ss", όπου d=ψηφίο ημέρας, m=ψηφίο μηνός, y=ψηφίο έτους, h=ψηφίο ώρας, m=ψηφίο λεπτών, s=ψηφίο δευτερολέπτων.

Παράδειγμα:

```
PRINT CDate(86397 / 86400)
PRINT CDate(1.3)
PRINT CDate(-0.25)
PRINT CDate(-1.45)
PRINT CDate(-700 + 2 / 76400)
PRINT CDate(2472645)
PRINT CDate(2472679)
PRINT CDate(2472679 + 1 / 86400)
PRINT CDate(2472679 + 1 / 1440)
PRINT Cdate(2472679.25)
```

Έξοδος

```
23:59:57
01/01/-4801 05:12:00
18:00:00
13:12:00
00:00:02
11/14/1969 22:00:00
12/18/1969 22:00:00
12/18/1969 22:00:01
12/18/1969 22:01:00
12/19/1969 04:00:00
```

CFloat

Σύνταξη:

```
Float = CFloat ( Expression AS Variant ) AS Float
Float = CFlt ( Expression AS Variant ) AS Float
```

Μετατρέπει μια έκφραση σε Float.

Προσοχή! Οι τοπικές ρυθμίσεις δεν χρησιμοποιούνται από την συνάρτηση. Αυτό σημαίνει ότι η Gambas αναγνωρίζει για υποδιαστολή την τελεία (.) και όχι το κόμμα (,).

Λάθη.

Μήνυμα	Περιγραφή
Type mismatch (#6)	Η έκφραση δεν μπορεί να μετατραπεί.

Παράδειγμα:

```
1 PRINT CFloat("+3.1414")
2 PRINT CFloat("1.0E+8")
```

Έξοδος

```
3.1414
1E+8
```

CInt

Σύνταξη:

```
iResult = CInt ( Expression AS Variant ) AS Integer
iResult = CInteger ( Expression AS Variant ) AS Integer
```

Μετατρέπει μια έκφραση σε Integer.

Λάθη.

Μήνυμα	Περιγραφή
Type mismatch (#6)	Η έκφραση δεν μπορεί να μετατραπεί.

Παράδειγμα:

```
1 PRINT CInt(Now)
2 PRINT CInt("175")
3 PRINT CInt(10000000)
4 PRINT CInt(TRUE)
5 PRINT CInt(3.68)
```

Έξοδος

```
2487613
175
10000000
-1
3
```

CLong

Σύνταξη:

```
lResult = CLong ( Expression AS Variant ) AS Long
lResult = CLng ( Expression AS Variant ) AS Long
```

Μετατρέπει μια έκφραση σε Long.

Λάθη.

Μήνυμα	Περιγραφή
Type mismatch (#6)	Η έκφραση δεν μπορεί να μετατραπεί.

Παράδειγμα:

```
1 PRINT CLong(PI)
```

Έξοδος

```
3
```

Conv$

Σύνταξη:

```
ConvertedString = Conv$ ( String AS String , SourceCharset AS
String , DestinationCharset AS String ) AS String

ConvertedString = Conv ( String AS String , SourceCharset AS
String , DestinationCharset AS String ) AS String
```

Μετατρέπει ένα string από ένα σετ χαρακτήρων σε άλλο. Ένα σετ χαρακτήρων αναπαριστάται όπως φαίνεται στον κώδικα "ASCII", "ISO-8859-1", ή "UTF-8".

Ο διερμηνευτής της Gambas χρησιμοποιεί εσωτερικά το σετ χαρακτήρων UTF-8.

Το σετ χαρακτήρων που χρησιμοποιείται από το σύστημα επιστρέφεται από την συνάρτηση

System.Charset. Και είναι UTF-8 σε ένα Ubuntu 10.04 LTS.

Το σετ χαρακτήρων που χρησιμοποιείται από το γραφικό περιβάλλον είναι UTF-8.

Η μετατροπή βασίζεται στην συνάρτηση *iconv()* των βιβλιοθηκών GNU.

Λάθη.

Μήνυμα	Περιγραφή
Bad string conversion (#32)	Το string προς μετατροπή περιέχει μη μεταφράσιμους χαρακτήρες.
Unsupported string conversion (#31)	Μη υποστηριζόμενη μετατροπή.

Παράδειγμα:

```
1 PRINT System.Charset
2 PRINT Desktop.Charset
3 PRINT Conv$("Gambas", "ASCII", "EBCDIC-US")
```

CShort

Σύνταξη:

```
Short = CShort ( Expression AS Variant ) AS Short
```

Μετατρέπει μια έκφραση σε Short.

Η έκφραση μετατρέπεται σε integer. Μετά, εάν ο integer υπερχειλίζει το εύρος τιμών του Short, συγκόπτεται.

Λάθη

Μήνυμα	Περιγραφή
Type mismatch (#6)	Η έκφραση δεν μπορεί να μετατραπεί.

Παράδειγμα:

```
1 PRINT CShort("15")
2 PRINT CShort(100000)
3 PRINT CShort(TRUE)
```

Έξοδος

```
15
-31072
-1
```

CSingle

Σύνταξη:

```
Result = CSingle ( Expression AS Variant ) AS Single
```

Μετατρέπει μια έκφραση σε Single.

Προσοχή! Οι τοπικές ρυθμίσεις δεν χρησιμοποιούνται από την συνάρτηση. Αυτό σημαίνει ότι η Gambas αναγνωρίζει για υποδιαστολή την τελεία (.) και όχι το κόμμα (,).

Λάθη

Μήνυμα	Περιγραφή
Type mismatch (#6)	Η έκφραση δεν μπορεί να μετατραπεί.

Παράδειγμα:

```
1 PRINT CSingle("+3.1616")
2 PRINT CSingle("1.0E+4")
3 PRINT CSingle(Pi);; CFloat(Pi)
4 PRINT CSingle(1E50)
```

Έξοδος

```
3.1616
10000
3.141592741013 3.14159265359
```

Overflow error

CStr

Σύνταξη:

```
sResult = CStr ( Expression AS Variant ) AS String
sResult = CString ( Expression AS Variant ) AS String
```

Μετατρέπει μια έκφραση σε string.

Προσοχή! Οι τοπικές ρυθμίσεις δεν χρησιμοποιούνται από την συνάρτηση.

Παράδειγμα:

```
1 PRINT CStr(-19776)
2 PRINT CStr(Now)
3 PRINT CStr(Pi)
```

Έξοδος

```
-19776
11/08/2010 10:38:23.365
3.14159265359
```

Προσοχή! Η Gambas έχει το χαρακτηριστικό της αυτόματης μετατροπής μεταβλητών όταν τελούνται εργασίες μεταξύ των μεταβλητών. Αυτό σημαίνει ότι αν προσθέσετε δύο string που περιέχουν αριθμητικές τιμές τότε η πράξη θα γίνει κανονικά. Η Gambas θα κάνει τις μετατροπές στο παρασκήνιο.

```
PRINT Cstr(19) + Cstr(1) ' εδώ το Cstr(19) και Cstr(1) είναι ίσο με
                         ' "19" και "1" δλδ έχουμε δύο strings
                         ' Η Gambas μετατρέπει τα strings σε αριθμούς
                         ' και προσπαθεί να τα προσθέσει.
```

Έξοδος

```
20
```

Αν όμως οι μεταβλητές δεν περιέχουν καθαρά αριθμούς τότε ένα λάθος προκύπτει.

```
PRINT CStr(19) + "1a"   'ένα λάθος στην γραμμή αυτή θα προκύψει.
```

Cvariant

Σύνταξη:

```
Result = CVar ( Expression )
Result = CVariant ( Expression )
```

Μετατρέπει μια έκφραση σε variant.

DConv$

Σύνταξη:

```
ConvertedString = DConv$ ( String AS String ) AS String

ConvertedString = DConv ( String AS String ) AS String
```

Μετατρέπει ένα string από το System.Charset σε Desktop.Charset, το οποίο πρέπει να είναι UTF-8.

Η μετατροπή βασίζεται στην συνάρτηση *iconv()* των βιβλιοθηκών GNU.

Λάθη

Μήνυμα	Περιγραφή
Bad string conversion (#32)	Το string προς μετατροπή περιέχει μη μεταφράσιμους χαρακτήρες.
Unsupported string conversion (#31)	Μη υποστηριζόμενη μετατροπή string.

SConv$

Σύνταξη:

```
ConvertedString = SConv$ ( String AS String ) AS String

ConvertedString = SConv( String AS String ) AS String
```

Μετατρέπει ένα string από το Desktop.Charset σε, System.Charset το οποίο πρέπει να είναι UTF-8 στο σετ χαρακτήρων του συστήματος.

Η μετατροπή βασίζεται στην συνάρτηση *iconv()* των βιβλιοθηκών GNU.

Λάθη

Μήνυμα	Περιγραφή
Bad string conversion (#32)	Το string προς μετατροπή περιέχει μη μεταφράσιμους χαρακτήρες.
Unsupported string conversion (#31)	Μη υποστηριζόμενη μετατροπή string.

Str$

Σύνταξη:

```
sResult = Str$ ( Expression AS Variant ) AS String

sResult = Str ( Expression AS Variant ) AS String
```

Μετατρέπει μια έκφραση σε string. Είναι η ακριβώς αντίστροφη συνάρτηση της Val.

Προσοχή! Οι τοπικές ρυθμίσεις χρησιμοποιούνται από την συνάρτηση.

Παράδειγμα:

```
1 PRINT Str(-19776)
2 PRINT Str(Now)
3 PRINT Str(Pi)
```

Έξοδος

```
-19776
11/08/2010 12:54:52
3.14159265359
```

Val

Σύνταξη:

```
sResult = Val ( String )
```

Μετατρέπει ένα string σε boolean, έναν αριθμό ή ημερομηνία ανάλογα με το περιεχόμενο του string.

Προσοχή! Οι τοπικές ρυθμίσεις χρησιμοποιούνται από την συνάρτηση.
Ο αλγόριθμος μετατροπής είναι ο εξής:

- Εάν το string μπορεί να διαβαστεί ως ημερομηνία ή τιμή χρόνου, τότε επιστρέφει ημερομηνία ή τιμή χρόνου
- Αλλιώς αν μπορεί να διαβαστεί ως αριθμός κινητής υποδιαστολής (float) τότε αυτός ο αριθμός επιστρέφεται
- Αλλιώς αν μπορεί να διαβαστεί ως ακέραιος (integer) τότε αυτός ο αριθμός επιστρέφεται.
- Αλλιώς αν το string είναι “TRUE” ή “FALSE” η τιμή του αντίστοιχου boolean επιστρέφεται.
- Αλλιώς αν τίποτα από αυτά δεν είναι επιστρέφεται το κενό (NULL).

Παράδειγμα:

```
1 PRINT Val("30/12/75 01:00")
2 PRINT Val("3.1415")
3 PRINT Val("-20")
4 PRINT Val("True")
5 PRINT IsNull(Val("Gambas"))
```

Έξοδος

```
3.1415
-20
True
True
```

Κεφάλαιο 4ο - Σταθερές

Όπως οι μεταβλητές και οι σταθερές είναι θέσεις αποθήκευσης δεδομένων οι οποίες όμως δεν αλλάζουν τιμή. Η τιμή που έχει μια σταθερά από την στιγμή που θα οριστεί μετά δεν είναι δυνατόν να αλλάξει.

Υπάρχουν δύο ειδών σταθερές.
Οι σταθερές **κυριολεκτικών τιμών** και οι **συμβολικές σταθερές**.

Οι σταθερές κυριολεκτικών τιμών ονομάζονται και σταθερές της γλώσσας και είναι η τιμή που πληκτρολογείτε άμεσα στο πρόγραμμα σας.

Έτσι όταν ορίζεται μια μεταβλητή π.χ.

```
DIM iArea AS Integer = 10
```

το 10 είναι είναι μια σταθερά κυριολεκτική τιμής.

Οι συμβολικές σταθερές είναι μια τιμή που αναπαρίσταται από ένα όνομα.
Οι συμβολικές σταθερές περιγράφονται αμέσως μετά.

Οι σταθερές δεν μπορούν να τροποποιηθούν ενώ το πρόγραμμα εκτελείται. Για να αλλάξετε την τιμή μιας σταθεράς θα πρέπει να αλλάξετε τον κώδικα.

Δηλώνοντας συμβολικές σταθερές

Πολλές φορές χρειαζόμαστε τιμές οι οποίες δεν αλλάζουν.
Για παράδειγμα ο αριθμός **π** ή η βάση των φυσικών λογαρίθμων **e** ή η σταθερά **Planck** ή η ταχύτητα του φωτός κτλ.

Αυτές τις τιμές καλό είναι να μην τις εκχωρούμε σε μεταβλητές διότι μέσα στο πρόγραμμα μας δεν θα μεταβληθούν αλλά θα παραμείνουν σε μια σταθερή τιμή.

Η Gambas δίνει την δυνατότητα να δημιουργήσει ο προγραμματιστής τις δικές του σταθερές.
Οι σταθερές που αναπαριστώνται από ένα όνομα ονομάζονται ***Συμβολικές σταθερές***.

Σύνταξη σταθερών

Η σύνταξη των σταθερών είναι η εξής:

```
{ PUBLIC | PRIVATE } CONST Identifier AS Datatype = Constant value
```

Η σύνταξη είναι παρόμοια με αυτή των μεταβλητών.
Μια σταθερά μπορεί να είναι δημόσια ή ιδιωτική **PUBLIC** ή **PRIVATE**.
Ορίζεται με την λέξη **CONST**
Ακολουθεί το όνομα της Identifier
Ακολουθεί ο τύπος δεδομένων της **AS** Datatype
Και τέλος εκχωρείται η τιμή της = Constant value

Κάθε σταθερά είναι καθολική, που σημαίνει ότι έχει εμβέλεια σε όλη την κλάση και δηλώνεται στην αρχή της.

Μπορούμε εναλλακτικά να δηλώσουμε μια λίστα σταθερών με απαρίθμηση. Αυτό γίνεται με την λέξη **ENUM**.
Η σύνταξη των απαριθμημένων σταθερών είναι η εξής:

```
{ PUBLIC | PRIVATE } ENUM Identifier [ = Value ] [ ,
Identifier [ = Value ] ... ]
```

Αν δεν τεθεί τιμή Value σε μια σταθερά η σταθερά λαμβάνει την προηγούμενη τιμή συν 1 ή μηδέν αν είναι η πρώτη σταθερά που απαριθμείται.

Τύποι δεδομένων σταθερών

Οι σταθερές έχουν τους εξής τύπους δεδομένων

Τύποι δεδομένων σταθερών
Boolean
Integer
Long
Float
String

Παραδείγματα συμβολικών σταθερών

```
PUBLIC CONST MAX_FILE AS Integer = 40

PRIVATE CONST MYBOOL AS Boolean = TRUE

PRIVATE CONST MAGIC_HEADER AS String = "# A line from form file"

PUBLIC ENUM X=1, Y=2, A=32, B=1467, C
```

Σταθερές κυριολεκτικών τιμών ή σταθερές της γλώσσας προγραμματισμού.

Αυτές είναι

Σταθερά	Παράδειγμα
Η τιμή ΑΛΗΘΕΙΑ.	TRUE
Η τιμή ΨΕΜΑ.	FALSE
Integer ή Long αριθμοί.	0, 123, -32769, 10000000000
Δεκαεξαδικοί προσημασμένοι ακέραιοι short.	&H1F5, &HFFFF, &H0000FFFF, &FFFF
Δεκαεξαδικοί προσημασμένοι ακέραιοι.	&H10BF332E, &10BF332E
Δεκαεξαδικοί μη προσημασμένοι ακέραιοι.	&H8000&, &HFFFF&
Δυαδικοί ακέραιοι.	&X1010010101, %101001011
Αριθμοί κινούμενης υποδιαστολής.	1.0, -5.345219E+45
Σταθερές συμβολοστοιχείων κειμένου ή αλλιώς αλφαρηθμιτικών.	"Hello World !"
Αλφαριθμητικόπρος μετάφραση.	("This software is cool")
Σταθερές κενού.	NULL

Προκαθορισμένες συμβολικές σταθερές.

Η Gambas έχει ορίσει κάποιες συμβολικές σταθερές για προγραμματιστική χρήση.

Αυτές είναι:

Τύποι δεδομένων προκαθορισμένων συμβολικών σταθερών.

Αυτές οι σταθερές επιστρέφονται από την συνάρτηση TypeOf.

Σταθερά	**Περιγραφή**
gb.Null	NULL
gb.Boolean	Boolean
gb.Byte	Byte
gb.Short	Short
gb.Integer	Integer
gb.Long	Long
gb.Single	Single
gb.Float	Float
gb.Date	Date
gb.String	String
gb.Variant	Variant
gb.Object	Αναφορά σε αντικείμενο

Τύποι αρχείων

Αυτές οι σταθερές χρησιμοποιούνται από την συνάρτηση Stat() .

Σταθερά	**Περιγραφή**
gb.File	Συνηθισμένο αρχείο
gb.Directory	Φάκελος
gb.Device	Αρχείο συσκευής
gb.Pipe	Διοχέτευση
gb.Socket	Υποδοχή
gb.Link	Συμβολικός σύνδεσμος

Σταθερές αλφαριθμητικών (strings)

Σταθερά	**Περιγραφή**
gb.NewLine	Χαρακτήρας νέας σειράς. Αντίστοιχο με το `Chr$(10)`.
gb.Tab	Χαρακτήρας Tab. Αντίστοιχο με `Chr$(9)`.

Τύποι ταξινόμησης

Αυτές οι σταθερές χρησιμοποιούνται από αντικείμενα ή πίνακες μεταβλητών στους οποίους μπορεί να γίνει ταξινόμηση των τιμών που περιέχουν ή από την συνάρτηση Comp(). Τέτοια αντικείμενα που περιέχουν πλήθος τιμών ονομάζονται “Συλλογές” (Collections). Η μέθοδος ταξινόμησης μπορεί να είναι δυαδική ή βάση κειμένου.

Σταθερά	**Περιγραφή**
gb.Binary	Δυαδική ταξινόμηση
gb.Text	Ταξινόμηση χωρίς ευαισθησία κεφαλαίων - μικρών
gb.Language	Ταξινόμηση κατά τις ρυθμίσεις της τοπικής γλώσσας.
gb.Language + gb.Text	Ταξινόμηση κατά τις ρυθμίσεις της τοπικής γλώσσας χωρίς ευαισθησία στα κεφαλαία - μικρά.

Ημέρες της εβδομάδος

Αυτές οι σταθερές ορίζουν τις ημέρες της εβδομάδος και επιστρέφονται από την συνάρτηση WeekDay().

Σταθερά	**Περιγραφή**
gb.Monday	Monday
gb.Tuesday	Tuesday
gb.Wednesday	Wednesday
gb.Thursday	Thursday
gb.Friday	Friday
gb.Saturday	Saturday
gb.Sunday	Sunday

Προκαθορισμένες μορφές

Μορφές αριθμών

Σταθερά	Περιγραφή
gb.GeneralNumber	Αριθμός με 12 δεκαδικά ψηφία. Χρησιμοποιείστε την επιστημονική μορφή αν η απόλυτη τιμή του αριθμού είναι μικρότερη από 10^{-4} ή μεγαλύτερη από 10^{7}.
gb.Fixed	Αντίστοιχο με το `"0.00"`
gb.Percent	Αντίστοιχο με το `"###%"`
gb.Scientific	Αναγραφή του αριθμού με εκθέτη και με 18 δεκαδικά ψηφία.

Μορφές ημερομηνίας και χρόνου

Σταθερά	Περιγραφή
gb.GeneralDate	Αναγραφή ημερομηνίας μόνο όταν η ημερομηνία και η τιμή του χρόνου έχει μέρος ημερομηνίας. Αναγραφή χρόνου εάν μέρος της τιμής είναι ημερομηνία.
gb.LongDate	Μορφή ημερομηνίας μακρού τύπου.
gb.MediumDate	Μορφή ημερομηνίας ενδιάμεσου τύπου.
gb.ShortDate	Μορφή ημερομηνίας σύντομου τύπου.
gb.LongTime	Μορφή χρόνου μακρού τύπου.
gb.MediumTime	Μορφή χρόνου ενδιάμεσου τύπου.
gb.ShortTime	Μορφή χρόνου σύντομου τύπου.

Μορφές νομισματικών μονάδων

Σταθερά	Περιγραφή
gb.Currency	Εθνική μορφή νομισματικής μονάδας.
gb.International	Διεθνής μορφή νομισματικής μονάδας.

Διάφορες μορφές

Σταθερά	Περιγραφή
gb.Standard	Χρησιμοποιείστε την σταθερά gb.GeneralNumber για μορφοποίηση αριθμών και την σταθερά gb.GeneralDate για μορφοποίηση ημερομηνιών και τιμών χρόνου.

Κατάταξη ψηφιολέξεων (Endianness ή Byte order)

Σταθερά	Περιγραφή
gb.BigEndian	Μεγάλη κατάταξη. Οι τιμές αποθηκεύονται στην μνήμη με προτεραιότητα των υψηλών bytes πρώτα. Για παράδειγμα ο δεκαεξαδικός αριθμός $4F52_{16}$ αποθηκεύεται ως 4F στην διεύθυνση 1000 και 52 στην διεύθυνση 1001
gb.LittleEndian	Μικρή κατάταξη. Οι τιμές αποθηκεύονται στην μνήμη με προτεραιότητα των χαμηλών bytes πρώτα. Για παράδειγμα ο δεκαεξαδικός αριθμός $4F52_{16}$ αποθηκεύεται ως 52 στην διεύθυνση 1000 και 4F στην διεύθυνση 1001

Κεφάλαιο 5ο – Προτάσεις και εκφράσεις

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

Στην Gambas 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 Gambas υπάρχουν επτά είδη μπλοκ κώδικα.

- Μπλοκ ρουτίνας
- Μπλοκ συναρτήσεων
- Μπλοκ επιλογής
- Μπλοκ υπόθεσης
- Μπλοκ “Με”
- Μπλοκ διαχείρισης λαθών
- 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	CATCH...
PROCEDURE ... END	FINALLY..
FUNCTION ... END	FOR ... NEXT
	FOR EACH ... NEXT
SELECT CASE ... END SELECT	DO .. LOOP
	REPEAT ... UNTIL
IF ... THEN ... ENDIF	WHILE ... WEND
WITH ... END WITH	

Τα μπλοκ CATCH και FINALLY έχουν αρχή αλλά δεν έχουν δικιά τους λέξη για το τέλος. Χρησιμοποιούνται πριν την λέξη END των ρουτινών.

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 float μεταβλητής Pi
MyName     'επιστρέφει κείμενο
a=b+c     'υπολογίζει το άθροισμα και επιστρέφει την τιμή στην μεταβλητή a.
```

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 εκχώρησης

Οι τελεστές εκχώρησης αποδίδουν μια τιμή σε μια μεταβλητή.
Εκτός από την απόδοση μπορούν να τελούν και κάποιες πράξεις.
Αυτοί είναι:

Τελεστής	**Σύνταξη**	**Περιγραφή**	**Παράδειγμα**
=	Variable = Expression	Απευθείας εκχώρηση.	a=10
+=	Variable **+=** Expression	Εκχώρηση με πρόσθεση. Αυτό είναι συνώνυμο με: Variable = Variable + Expression	a+=10
-=	Variable **-=** Expression	Εκχώρηση με αφαίρεση. Αυτό είναι συνώνυμο με: Variable = Variable - Expression	a-=10
***=**	Variable ***=** Expression	Εκχώρηση με πολλαπλασιασμό. Αυτό είναι συνώνυμο με: Variable = Variable * Expression	a*=10
/=	Variable **/=** Expression	Εκχώρηση με διαίρεση. Αυτό είναι συνώνυμο με: Variable = Variable / Expression	a/=10
\=	Variable **\=** Expression	Εκχώρηση με διαίρεση ακεραίου. Αυτό είναι συνώνυμο με: Variable = Variable \ Expression	a\=10
%=	Variable %= Expression	Εκχώρηση με υπολογισμό του υπόλοιπου μιας διαίρεσης ακεραίου. Αυτό είναι συνώνυμο με: Variable = Variable % Expression	a%= 10
&=	Variable **&=** Expression	Εκχώρηση με συνένωση. Αυτό είναι συνώνυμο με: Variable = Variable & Expression	a&=”text2”
&/=	Variable **&/=** Expression	Εκχώρηση με συνένωση διαδρομής. Αυτό είναι συνώνυμο με: Variable = Variable &/ Expression	a&/=”dirextory2”

Τελεστές αριθμητικοί

Οι αριθμητικοί τελεστές τελούν αριθμητικές πράξεις μεταξύ των εκφράσεων. Διακρίνονται σε δύο υποομάδες. Τους γενικούς αριθμητικούς τελεστές και τους αριθμητικούς τελεστές σύγκρισης.

Γενικοί αριθμητικοί τελεστές

Τελεστής	**Σύνταξη**	**Περιγραφή**	**Παράδειγμα**
+	Number + Number	Προσθέτει δύο αριθμούς.	5+2
-	- Number	Υπολογίζει τον αντίθετο ενός αριθμού.	-5
-	Number - Number	Αφαιρεί δύο αριθμούς.	5-2
*	Number * Number	Πολλαπλασιάζει δύο αριθμούς.	5*2
/	Number / Number	Διαιρεί δύο αριθμούς. Αν διαιρέσετε έναν αριθμό με το 0 ένα λάθος παράγεται Division By Zero (#26).	1/2
^	Number ^ Power	Υψώνει έναν αριθμό σε δύναμη.	5 ^ 2
\	Number \ Number Number DIV Number	Υπολογίζει το πηλίκο μιας διαίρεσης δύο αριθμών. Αν διαιρέσετε έναν αριθμό με το 0 ένα λάθος παράγεται Division By Zero (#26). Την ίδια πράξη κάνει και η συνάρτηση DIV.	5\2
%	Number MOD Number	Υπολογίζει το υπόλοιπο μιας διαίρεσης δύο αριθμών. Αν διαιρέσετε έναν αριθμό με το 0 ένα λάθος παράγεται Division By Zero (#26).	5 % 2

Αριθμητικοί τελεστές σύγκρισης

Τελεστής	Σύνταξη	Περιγραφή	Παράδειγμα
=	Number = Number	Επιστρέφει TRUE αν οι δύο αριθμοί είναι ίσοι.	5=5 *'True* 5=10 *'False*
<>	Number <> Number	Επιστρέφει TRUE αν οι δύο αριθμοί είναι διαφορετικοί.	5<>6 *'True*
<	Number1 < Number2	Επιστρέφει TRUE εάν ο Number1 είναι μικρότερος του Number2.	5<4 *'True*
>	Number1 > Number2	Επιστρέφει TRUE εάν ο Number1 είναι μεγαλύτερος του Number2.	5>6 *'True*
<=	Number1 <= Number2	Επιστρέφει TRUE εάν ο Number1 είναι μικρότερος ή ίσος με τον Number2.	5<=5 *'True*
>=	Number1 >= Number2	Επιστρέφει TRUE εάν ο Number1 είναι μεγαλύτερος ή ίσος με τον Number2.	5>=5 *'True*

Προτεραιότητα αριθμητικών τελεστών

Όταν σε μια πρόταση υπάρχουν παραπάνω από μια εκφράσεις με πολλούς αριθμητικούς τελεστές τότε υπάρχει μια σειρά προτεραιότητας στις πράξεις.

Αν γράψετε a = 5+2*9 τότε πρώτα τελείται ο πολλαπλασιασμός και μετά η πρόσθεση. Έτσι το a = 23

Στον παρακάτω πίνακα αναφέρεται η προτεραιότητα των αριθμητικών τελεστών.

Τελεστής	Παράδειγμα
- (αντίθετο)	`w = - a ^ 4` είναι το ίδιο με `( - a ) ^ 4`
^	`w = 3 ^ 2 * 4 ^ 3` είναι το ίδιο με `(3 ^ 2) * ( 4 ^ 3 )`
* / DIV MOD	`w = 3 * 2 + 4 * 3` είναι το ίδιο με `(3 * 2) + ( 4 * 3 )`
+ -	
= <> >= <= > <=	`w = 3 + 2 = 4 + 1` είναι το ίδιο με `( 3 + 2 ) = ( 4 + 1 )`
OR AND XOR	`w = i >50 AND i < 10` είναι το ίδιο με `( i > 50 ) AND ( i < 10 )`

Για να ορίσετε ομάδες πράξεων χρησιμοποιείστε παρενθέσεις ()

Λογικοί τελεστές

Οι λογικοί τελεστές τελούν λογικές μαθηματικές πράξεις σε αριθμούς.
Οι λογικοί τελεστές στην Gambas είναι AND, OR, NOT, XOR
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

A	**Αποτέλεσμα**
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

A	B	Αποτέλεσμα
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

A	B	Αποτέλεσμα

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
```

Τελεστές αλφαριθμητικών

Οι τελεστές αλφαριθμητικών τελούν πράξεις μεταξύ strings.
Αυτοί χωρίζονται σε δύο κατηγορίες, τους τελεστές αλφαριθμητικών γενικής χρήσεως και τους τελεστές αλφαριθμητικών συγκρίσεως.

Τελεστές αλφαριθμητικών γενικής χρήσεως.

Τελεστής	Σύνταξη	Περιγραφή	Παράδειγμα

&	String **&** String	Ενώνει δύο strings.	`sName = “Peter ”`
			`sLastname = “Papadopoulos”`
			`sFullName = sName & sLastName`

&/	String **&/** String	Ενώνει δύο strings που περιέχουν ονόματα αρχείων ή καταλόγων. Προσθέτει ένα χαρακτήρα διαδρομής (/) μεταξύ των δύο strings εάν χρειάζεται.	`PRINT "dir1" &/ "file1"` `'dir1/file1`

Τελεστές σύγκρισης αλφαριθμητικών

Τελεστής	Σύνταξη	Περιγραφή	Παράδειγμα
=	String **=** String	Επιστρέφει TRUE αν τα δύο strings είναι ίδια.	`PRINT "Peter" = "Peter" 'TRUE`
==	String **==** String	Επιστρέφει TRUE αν τα δύο strings είναι ίδια. Η σύγκριση δεν είναι ευαίσθητη στα κεφαλαία μικρά.	`PRINT "abc" == "abc" 'true` `PRINT "abc" == "ab" 'false`

LIKE	String **LIKE** String	Ελέγχει εάν ένα string ταιριάζει σε μία μορφή (pattern).	`PRINT "abc" LIKE "abc" 'true`
			`PRINT "abc" LIKE "aBc" 'true`
			`PRINT "abc" LIKE "abcd" 'false`

<>	String **<>** String	Επιστρέφει TRUE εάν δύο strings είναι διαφορετικά.	`PRINT "abc" <> "abc" 'false` `PRINT "abc" <> "aBc" 'true`

<	String1 **<** String2	Επιστρέφει TRUE εάν το String1 είναι αυστηρά χαμηλότερο του String2. Η σύγκριση γίνεται στις δυαδικές τιμές των χαρακτήρων του πίνακα ASCII.	`PRINT "abc" < "abc" 'false`
			`PRINT "abc" < "aBc" 'false`
			`PRINT "ABC" < "abc" 'true`
			`'ο κώδικας`
			`PRINT "A" < "a"`
			`'είναι παρόμοιος με`
			`PRINT Asc("A") < Asc("a")`

>	String1 **>** String2	Επιστρέφει TRUE εάν το String1 είναι αυστηρά υψηλότερο του String2. Η σύγκριση γίνεται στις δυαδικές τιμές των χαρακτήρων του πίνακα ASCII.	`PRINT "abc" > "abc" 'false`
			`PRINT "abc" > "aBc" 'true`
			`PRINT "ABC" > "abc" 'false`

<=	String1 **<=** String2	Επιστρέφει TRUE εάν το String1 είναι χαμηλότερο ή ίσο του String2. Η σύγκριση γίνεται στις δυαδικές τιμές των χαρακτήρων του πίνακα ASCII.	`PRINT "abc" <= "abc" 'true`
			`PRINT "abc" <= "aBc" 'false`
			`PRINT "ABC" <= "abc" 'true`

>=	String1 **>=** String2	Επιστρέφει TRUE εάν το String1 είναι υψηλότερο ή ίσο του String2. Η σύγκριση γίνεται στις δυαδικές τιμές των χαρακτήρων του πίνακα ASCII.	`PRINT "abc" >= "abc" 'true`
			`PRINT "abc" >= "aBc" 'true`
			`PRINT "ABC" >= "abc" 'false`

Προτεραιότητα τελεστών

Τελεστής	Δύναμη	Παράδειγμα
- (άρνηση) NOT	Μέγιστο.	a = - b ^ 4 είναι το ίδιο με (- b) ^ 4
IS	11	
&	9	
&/	8	
^	7	a = 3 ^ 2 * 4 ^ 3 είναι το ίδιο με (3 ^ 2) * (4 ^ 3)
* / \ DIV MOD	6	a = 5 * 3 + 4 * 4 είναι το ίδιο με (5 * 3) + (4 * 4)
+ -	5	
<> >= <= > < LIKE	4	a = 5 + 3 = 6 + 2 είναι το ίδιο με (5 + 3) = (6 + 2)
OR AND XOR	2	a = b > 20 AND b < 30 είναι το ίδιο με (b > 20) AND (b < 30)

Όσο πιο δυνατός είναι ένας τελεστής τόσο το αργότερο τελεί μια πράξη.

Μπορείτε να χρησιμοποιείτε παρενθέσεις για να ορίζεται μέρη που θα τελούνται ομαδικά.

Κεφάλαιο 6ο – Εμβάθυνση στις συναρτήσεις

Στο Κεφάλαιο 2, είδαμε την σύνταξη των ρουτινών και συναρτήσεων και μερικώς την χρήση τους για να εξηγήσουμε το παράδειγμα “Hello World”.
Σε αυτό το κεφάλαιο θα δούμε κάποια προχωρημένα θέματα για τις συναρτήσεις όπως:

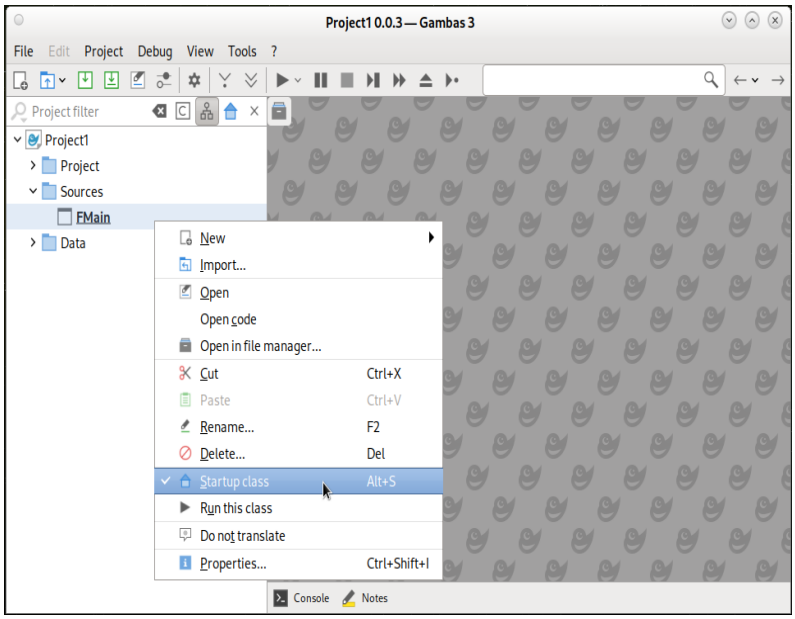
- αναλυτικά τα μέρη μιας συνάρτησης,
- εμβέλεια μεταβλητών,
- τα ορίσματα των συναρτήσεων,
- τιμές επιστροφής
- αναδρομή συναρτήσεων.

Αναλυτικά τα μέρη μιας συνάρτησης

Μια συνάρτηση είναι υποπρόγραμμα που περιέχει εντολές και ενεργεί στα δεδομένα.
Μια συνάρτηση επιστρέφει μια τιμή ενώ μια ρουτίνα δεν επιστρέφει.

Έτσι καλό είναι να χρησιμοποιείτε τις SUB για υπορουτίνες που δεν επιστρέφουν τιμές και τις FUNCTION, PROCEDURE για συναρτήσεις που επιστρέφουν τιμές.

Κάθε πρόγραμμα έχει μία κλάση εκκίνησης που ορίζεται με δεξί κλικ στο όνομα του αρχείου και επιλογή αυτού ως “Κλάση εκκίνησης”.

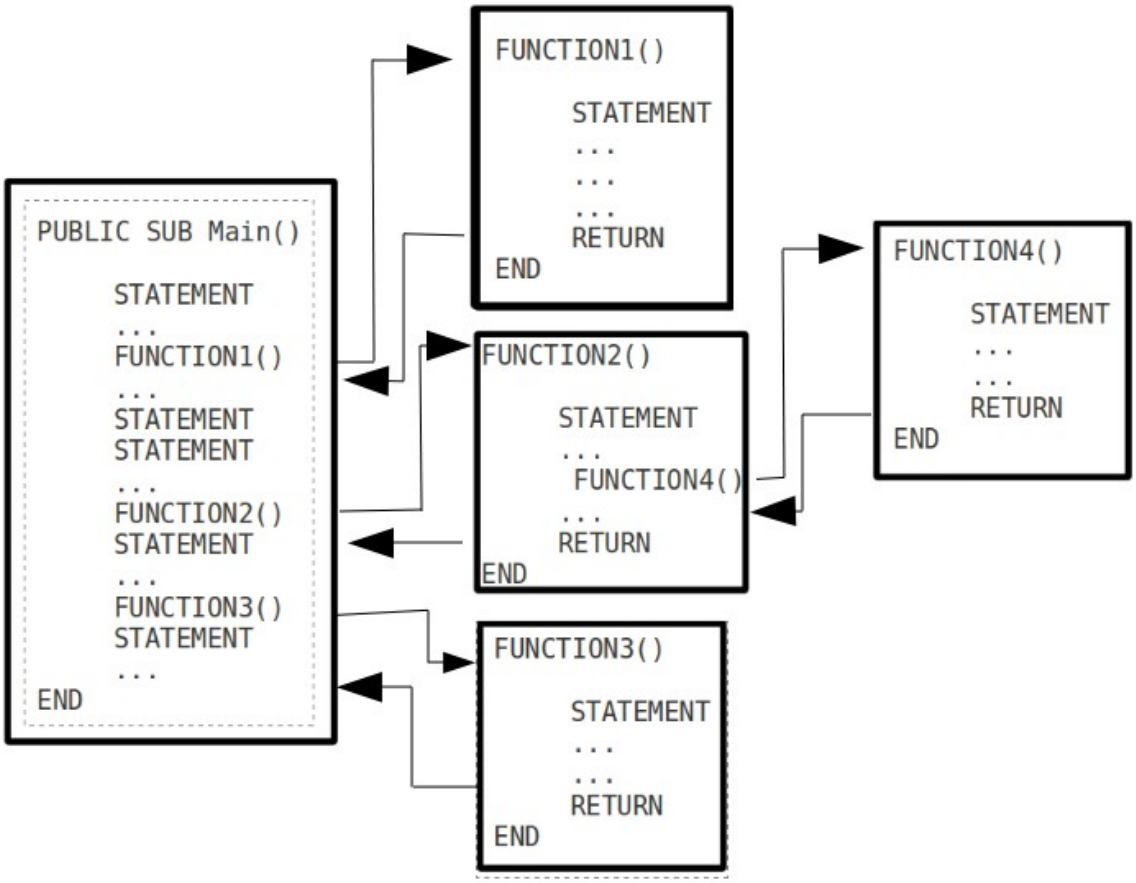


Εικόνα 61: Ορίζοντας την κλάση εκκίνησης

Αν η κλάση εκκίνης είναι ένα Module τότε πρέπει να ορίσουμε μία δημόσια υπορουτίνα με το όνομα Main(). Έτσι σε ένα Module ως κλάση εκκίνησης θα γράφαμε.

```
PUBLIC SUB Main()

...

END
```

Η Main() είναι η καρδιά του προγράμματος, θα καλεί άλλες συναρτήσεις και υπορουτίνες που με την σειρά τους αυτές μπορούν να καλούν άλλες και έτσι το πρόγραμμα θα ελέγχεται από ξεχωριστά μέρη αυτόνομου κώδικα που κάθε μέρος θα εκτελεί μία συγκεκριμένη εργασία και μόλις τελειώνει θα επιστρέφει την ροή ελέγχου πίσω στην Main().



Εικόνα 62: Αναπαράσταση κλήσεων ρουτινών και συναρτήσεων από την Main()

Από την Main() η ροή του προγράμματος συνεχίζεται στην επόμενη γραμμή της κληθείσας συνάρτησης.

Πολύπλοκες εργασίες θα πρέπει να διασπώνται σε πολλές συναρτήσεις και κάθε μία συνάρτηση να εκτελεί μία συγκεκριμένη εργασία.

Οι συναρτήσεις είναι δύο ειδών. Αυτές που ορίζει ο προγραμματιστής και αυτές που του προσφέρει η γλώσσα προγραμματισμού.

Για παράδειγμα η Gambas ήδη προσφέρει ένα πλήθος συναρτήσεων όπου μπορείτε να χρησιμοποιήσετε στα προγράμματα σας.

Για παράδειγμα, η Abs() επιστρέφει την απόλυτη τιμή ενός αριθμού, η Round() στρογγυλοποιεί έναν αριθμό, η Acos() επιστρέφει το συνημίτονο ενός αριθμού, ήδη είδαμε τις συναρτήσεις μετατροπής τιμών από τύπο δεδομένων σε άλλο τύπο Cbool(), Cbyte(), Cfloat κτλ.
Όλες αυτές οι συναρτήσεις και πολλές άλλες ακόμη είναι συναρτήσεις της γλώσσας Gambas.

Είδαμε σε προηγούμενο κεφάλαιο πώς ορίζουμε μία συνάρτηση.
Σε συντομία η δήλωση μιας συνάρτησης είναι η εξής:

```
STATIC PUBLIC FUNCTION Identifier (OPTIONAL Parameter AS Datatype, ...) AS Datatype
STATIC PRIVATE FUNCTION Identifier (OPTIONAL Parameter AS Datatype, ...) AS Datatype
```

Οι λέξεις που αποτελούν αυτήν την δήλωση άλλες είναι απαραίτητες και άλλες προαιρετικές.

Τα μέρη της δηλώσεως μιας συνάρτησης είναι:

- `STATIC`, προαιρετική λέξη με την οποία η συνάρτηση έχει πρόσβαση μόνο στις STATIC μεταβλητές.
- `PUBLIC` ή `PRIVATE`, απαραίτητες λέξεις που δηλώνουν αν η συνάρτηση θα είναι δημόσια ή ιδιωτική. Όταν είναι δημόσια είναι προσπελάσιμη από άλλες κλάσεις και όταν είναι ιδιωτική δεν είναι.
- `FUNCTION`, απαραίτητη λέξη που δηλώνει ότι αυτό που ακολουθεί είναι μια συνάρτηση.
- `Identifier`, απαραίτητος προσδιορισμός. Είναι το όνομα της συνάρτησης.
- `()`, απαραίτητα σύμβολα όπου εντός τους ορίζονται οι μεταβλητές που δέχεται η συνάρτηση σαν είσοδο.
- `OPTIONAL`, προαιρετική λέξη που δηλώνει ότι η μεταβλητή είναι προαιρετική και ο χρήστης μπορεί να την παραλείψει εάν θέλει. Όταν παραλείπουμε ένα όρισμα από μια συνάρτηση απλά γράφουμε ένα κόμμα (,). Για παράδειγμα αν έχουμε μια συνάρτηση που μπορεί να προσθέσει τρεις αριθμούς και ο πρώτος είναι προαιρετικός αυτή την συνάρτηση όταν θα την καλούσαμε θα γράφαμε:

```
SumThreeNumbers(, 10, 15)
```

Για να δείτε αυτό το χαρακτηριστικό δημιουργήστε ένα νέο Project8.
Εισάγεται ένα Module και διαγράψτε την FMain.
Ορίστε το Module σαν κλάση εκκίνησης και γράψτε τον παρακάτω κώδικα.

Παράδειγμα:

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  PRINT SumThreeNumbers(, 10, 15)
6
7  END
8
9  PUBLIC FUNCTION SumThreeNumbers(OPTIONAL a AS Integer, b AS
   Integer, c AS Integer) AS Integer
10
11   RETURN (a + b + c)
12
13 END
```

Ανάλυση:
Στην γραμμή 3 ορίζεται η Main(), η οποία δεν έχει μεταβλητές εισόδου και για αυτό έχει παρενθέσεις χωρίς να εσωκλείουν κάτι.

Στην γραμμή 5 καλείται η `SumThreeNumbers()` και λαμβάνει τρία ορίσματα. Το πρώτο επειδή είναι προαιρετικό παραλείπεται και γράφουμε απλά ένα κόμμα για να συνεχίσουμε να γράψουμε τα άλλα δύο δλδ το 10 και 15.
Αφού κληθεί η `SumThreeNumbers()` τυπώνεται το αποτέλεσμα που επιστρέφει στην προκαθορισμένη έξοδο. Το αποτέλεσμα είναι το 25.

Στην γραμμή 9 έχουμε την δήλωση της συνάρτησης `SumThreeNumbers()`. Όπως φαίνεται είναι μια δημόσια, PUBLIC, συνάρτηση που λαμβάνει μια προαιρετική τιμή, OPTIONAL, a και υποχρεωτικές τιμές τις b και c. Η συνάρτηση δέχεται ακεραίους αριθμούς και επιστρέφει έναν ακέραιο, Integer.

Στην γραμμή 11 γίνεται η πρόσθεση των τριών αριθμών και την τιμή αυτή την επιστρέφει η συνάρτηση.

Ας συνεχίσουμε την περιγραφή των μερών της δηλώσεως μιας συνάρτησης.

- `Parameter`, προαιρετική έκφραση που είναι το όνομα της μεταβλητής που θα χρησιμοποιήσει η συνάρτηση σαν είσοδο.
- `AS`, απαραίτητη λέξη όταν υπάρχει είσοδος στην συνάρτηση, που δηλώνει τον τύπο δεδομένων της εισόδου ή που δηλώνει στο τέλος απαραίτητα τον τύπο δεδομένων που επιστρέφει η συνάρτηση.
- `Datatype`, απαραίτητη λέξη όταν υπάρχει είσοδος στην συνάρτηση που δείχνει τον τύπο δεδομένων του ορίσματος ή τον τύπο δεδομένων που επιστρέφει η συνάρτηση.
- `,` προαιρετικό σύμβολο που χρησιμοποιείται όταν υπάρχουν ορίσματα πάνω από ένα ως είσοδος στην συνάρτηση.

Στην δήλωση υπορουτινών υπάρχει ακριβώς η ίδια λογική απλά οι υπορουτίνες δεν επιστρέφουν τιμές και έτσι στο τέλος δεν δηλώνουμε τύπο δεδομένων.
Η σύνταξη δήλωσης υπορουτίνας είναι η εξής:

```
STATIC PUBLIC SUB Identifier (OPTIONAL Parameter AS Datatype, ...)
STATIC PRIVATE SUB Identifier (OPTIONAL Parameter AS Datatype, ...)
STATIC PUBLIC PROCEDURE Identifier (OPTIONAL Parameter AS Datatype, ...)
STATIC PRIVATE PROCEDURE Identifier (OPTIONAL Parameter AS Datatype, ...)
```

Η Gambas δεν είναι αυστηρή στην σύνταξη ρουτινών και σας επιτρέπει να χρησιμοποιήσετε για υπορουτίνες εκτός από SUB, PROCEDURE και την λέξη FUNCTION.
Καλό θα είναι όμως να υιοθετήσετε ένα αυστηρό στυλ γραφής κώδικα και να χρησιμοποιείτε τις λέξεις SUB, PROCEDURE για δήλωση υπορουτινών και την λέξη FUNCTION για δήλωση συναρτήσεων.

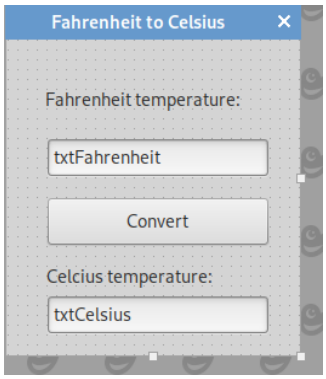
Εμβέλεια μεταβλητών

Τοπικές μεταβλητές

Εκτός από το να διοχετεύετε μεταβλητές σε μια συνάρτηση μπορείτε να ορίζετε μεταβλητές και μέσα στο σώμα της συνάρτησης. Αυτές οι μεταβλητές ονομάζονται τοπικές και ορίζονται με την λέξη DIM.
Όταν η συνάρτηση επιστρέφει οι τοπικές μεταβλητές δεν είναι διαθέσιμες. Οι παράμετροι που διοχετεύονται σε μια συνάρτηση είναι τοπικές μεταβλητές.

Ας δούμε ένα παράδειγμα. Το παράδειγμα θα μετατρέπει βαθμούς Fahrenheit σε βαθμούς Celsius.

Δημιουργείστε το Project9.
Σχεδιάστε δύο Label με το όνομα lLabel1 και lLabel2, ένα TextBox με το όνομα txtFahrenheit, ένα Button με το όνομα btnConvert και τέλος ένα άλλο TextBox με το όνομα txtCelsius, όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 63: Η διεπαφή του Project8

Ορίστε τις ιδιότητες:
Fmain.Text = Fahrenheit to Celsius είναι η επικεφαλίδα της φόρμας.
Fmain.Border = Fixed είναι η ιδιότητα του περιγράμματος της φόρμας ώστε να μην γίνεται resize η φόρμα.
lLabel1.Text = Fahrenheit temperature: είναι το κείμενο της ετικέτας.
lLabel2.Text = Celsius temperature: είναι το κείμενο της ετικέτας.
txtFahrenheit.Text=<τίποτα> αφήστε κενό το κείμενο.
btnConvert.Text=Concert
txtCelsius.Text=<τίποτα> αφήστε κενό το κείμενο.

Αυτός ήταν ο σχεδιασμός της διεπαφής, ας περάσουμε τώρα στον κώδικα.

Αρχικά θέλουμε όταν θα εμφανιστεί η φόρμα να είναι στο κέντρο της οθόνης, για αυτό κατά το άνοιγμα της φόρμας θα πρέπει να γράψουμε κώδικα που να θέτει την φόρμα στο κέντρο της οθόνης. Ο κώδικας αυτός είναι ο εξής:

```
PUBLIC SUB Form_Open()

  ME.Center

END
```

Έπειτα θα πρέπει να γράψουμε την συνάρτηση μετατροπής.
Η συνάρτηση αυτή θα επιστρέφει έναν πραγματικό αριθμό δλδ έναν float.
Επίσης θα δέχεται ως όρισμα έναν άλλο πραγματικό αριθμό, δλδ έναν άλλο float.

Οπότε η δήλωση της θα είναι η εξής:

```
PUBLIC FUNCTION Convert(fTemp AS Float) AS Float

END
```

Η μεταβλητή `fTemp` είναι μια τοπική μεταβλητή της συνάρτησης και βρίσκεται στην είσοδο της συνάρτησης.

Στο σώμα της συνάρτησης θα πρέπει να έχουμε μια μεταβλητή που θα κρατάει το αποτέλεσμα της πράξης της μετατροπής και την ίδια την πράξη μετατροπής των βαθμών Φαρενάιτ σε βαθμούς Κέλσιους.

Ο τύπος μετατροπής είναι **Κελσίους = ((Φαρενάιτ - 32) * 5) / 9**.

Οπότε στον κώδικα μας θα ορίσουμε μια άλλη τοπική μεταβλητή fResult η οποία θα είναι ένας float και θα γράψουμε την πράξη μετατροπής.

Ο κώδικας μας θα είναι:

```
PUBLIC FUNCTION Convert(fTemp AS Float) AS Float

  DIM fResult AS Float

  fResult = ((fTemp - 32) * 5) / 9
  RETURN fResult

END
```

Το επόμενο βήμα είναι να γράψουμε κώδικα που θα λέει ότι όταν ο χρήστης πατήσει το κουμπί btnConvert να γίνει η μετατροπή των βαθμών.

Εδώ θα πρέπει να κάνουμε τα εξής:
Να μετατρέψουμε το string txtFahrenheit.Text σε τύπο δεδομένων float.
Και να διοχετεύσουμε την τιμή του νέου τύπου δεδομένων float στην είσοδο της συνάρτησης `Convert(fTemp AS Float)`.
Για να γίνει αυτό θα χρειαστούμε μια τοπική μεταβλητή την fTempFahr.

Ο κώδικας μας θα είναι:

```
PUBLIC SUB btnConvert_Click()

  DIM fTempFahr AS Float

  fTempFahr = Val(txtFahrenheit.Text)

  txtCelsius.Text = Convert(fTempFahr)

END
```

Ορίσουμε την μεταβλητή `fTempFahr`
Η συνάρτηση Val() μετατρέπει τον τύπο δεδομένων string σε αριθμητική τιμή.
Τέλος καλούμε την συνάρτηση μετατροπής.

Όλος ο κώδικας μαζί του Project9 είναι:

Παράδειγμα:

```
1  ' Gambas class file
2
3  PUBLIC SUB Form_Open()
4
5    ME.Center
6
7  END
8
9
10 PUBLIC SUB btnConvert_Click()
11
12   DIM fTempFahr AS Float
13
14   fTempFahr = Val(txtFahrenheit.Text)
15
16   txtCelsius.Text = Convert(fTempFahr)
17
18 END
19
20 PUBLIC FUNCTION Convert(fTemp AS Float) AS Float
21
22   DIM fResult AS Float
23
24   fResult = ((fTemp - 32) * 5) / 9
25   RETURN fResult
26
27 END
```

Ονόματα τοπικών μεταβλητών

Ένα σημείο προσοχής είναι στο πώς η Gambas διαχειρίζεται τα ονόματα των μεταβλητών.

Οι μεταβλητές μπορούν να έχουν ίδια ονόματα και να βρίσκονται σε διαφορετικές συναρτήσεις.
Για παράδειγμα μια μεταβλητή x που βρίσκεται στην υπορουτίνα Form_Open() είναι διαφορετική από την μεταβλητή x που βρίσκεται στην υπορουτίνα MyFunction().

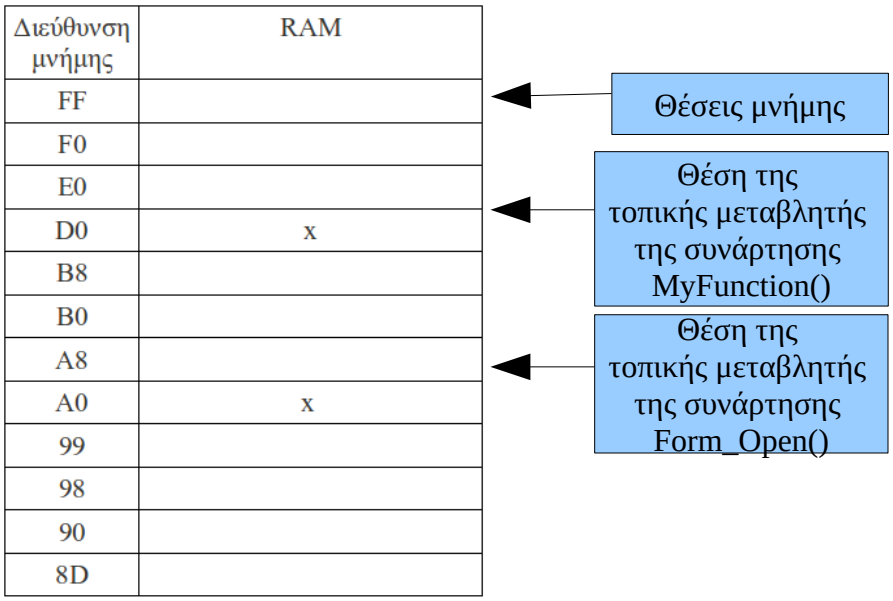
Το Project10 δείχνει το θέμα αυτό.

Παράδειγμα:

```
1  ' Gambas class file
2  PUBLIC SUB Form_Open()
3
4  DIM x AS Integer = 1
5
6  PRINT "I am in Form_Open"
```

```
7   PRINT x
8
9   MyFunction()
10
11  PRINT "I am in Form_Open"
12  PRINT x
13
14  END
15
16  PUBLIC SUB MyFunction()
17
18    DIM x AS Integer = 2
19
20    PRINT "I am in MyFunction"
21    PRINT x
22
23  END
```

Στην γραμμή 4 ορίζεται η τοπική μεταβλητή x στην υπορουτίνα Form_Open() και λαμβάνει την τιμή 1
Στην συνέχεια τυπώνεται στην προκαθορισμένη έξοδο η τιμή της.
Καλείται η συνάρτηση MyFunction().
Στην γραμμή 18 ορίζεται η τοπική μεταβλητή x της MyFunction(). Αν και η μεταβλητή έχει το ίδιο όνομα αντιστοιχεί σε διαφορετική διεύθυνση μνήμης και έτσι καταλαμβάνει διαφορετική θέση στην RAM.



Εικόνα 64: Η RAM και οι θέσεις μνήμης

Έτσι η Gambas ξεχωρίζει τις μεταβλητές ανεξαρτήτως αν έχουν το ίδιο όνομα. Αυτό είναι ένα

σημείο προσοχής που δεν θα πρέπει να μπερδεύει τον προγραμματιστή.

Οι παράμετροι είναι τοπικές μεταβλητές

Οι τιμές που διοχετεύονται σε μια συνάρτηση ονομάζονται ορίσματα. Τα ορίσματα τα δέχονται μεταβλητές στην είσοδο της συνάρτησης. Η ενέργεια αυτή λέγεται διοχέτευση.
Οι μεταβλητές των ορισμάτων εισόδου μιας συνάρτησης είναι τοπικές μεταβλητές στην συνάρτηση. Αν γίνει κάποια αλλαγή σε αυτές δεν επηρεάζονται οι τιμές στην κληθείσα συνάρτηση. Αυτό σημαίνει ότι αν διοχετεύσουμε την τιμή της μεταβλητής x σε μια συνάρτηση που έχει είσοδο μια επίσης μεταβλητή x, οι δύο μεταβλητές είναι διαφορετικές. Όποια αλλαγή γίνει στην δεύτερη μεταβλητή x δεν θα επηρεάσει την πρώτη μεταβλητή x.
Αυτό το θέμα φαίνεται στο Project11.

Παράδειγμα:

```
1  ' Gambas class file
2  PUBLIC SUB Form_Open()
3
4  DIM x AS Integer = 10
5  DIM y AS Integer
6
7  PRINT "i am in Form_Open"
8  PRINT "x="; x
9
10 y = MyFunction(x)
11
12 PRINT "i am in Form_Open"
13 PRINT "x="; x
14 PRINT "y="; y
15
16 END
17
18 PUBLIC FUNCTION MyFunction(x AS Integer) AS Integer
19
20   PRINT "i am in MyFunction"
21   PRINT "x="; x
22
23   PRINT "Increase x"
24   INC x
25   PRINT "x="; x
26   RETURN x
27
28 END
```

Στην γραμμή 4 και 5 ορίζονται δύο τοπικές μεταβλητές η x και η y. Η x μάλιστα λαμβάνει και την τιμή 10.

Στην γραμμή 10 καλείται η συνάρτηση MyFunction().
Η συνάρτηση αυτή λαμβάνει σαν όρισμα την τιμή της τοπικής μεταβλητής x δλδ το 10.
η συνάρτηση MyFunction() έχει μια δικιά της τοπική μεταβλητή x η οποία είναι διαφορετική μεταβλητή από την x της Form_Open(). Η μεταβλητή x της MyFunction είναι η μεταβλητή

εισόδου της συνάρτησης.

Έτσι όταν αυξάνουμε την τιμή της στην γραμμή 24 και η συνάρτηση επιστρέφει, στην γραμμή 13 τυπώνεται η μεταβλητή x της Form_Open(). Η τοπική μεταβλητή x της Form_Open() δεν έχει αλλάξει, αυτή που άλλαξε είναι η τοπική μεταβλητή x της MyFunction().

Σημείωση για την σύνταξη της εντολής PRINT.

Σύνταξη:

```
PRINT [ # hStream , ] Expression [ { ; | ;; | , } Expression ... ]
[ { ; | ;; | , }  ]
```

Η εντολή PRINT τυπώνει την έκφραση Expression στο ρεύμα # hStream, αν δεν οριστεί ρεύμα τότε χρησιμοποιείται η προκαθορισμένη έξοδος του τερματικού.

- Αν δεν υπάρχει Ελληνικό ερωτηματικό τότε ένας χαρακτήρας νέας γραμμής τυπώνεται (\n).
- Αν υπάρχει Ελληνικό ερωτηματικό η έκφραση που ακολουθεί τυπώνεται αμέσως μετά.
- Αν υπάρχει διπλό Ελληνικό ερωτηματικό τότε μεταξύ των εκφράσεων τυπώνεται το κενό διάστημα.
- Αν υπάρχει ένα κόμμα ανάμεσα στις εκφράσεις τότε τυπώνεται ένας χαρακτήρας TAB.

Παράδειγμα:

```
1  'Αν x=10
2
3  PRINT "x="; x
4
5  'τυπώνεται στην προκαθορισμένη έξοδο
6  'x=10
```

Καθολικές μεταβλητές

Όπως έχουμε αναφέρει οι καθολικές μεταβλητές δηλώνονται έξω από κάθε συνάρτηση και έχουν εμβέλεια σε όλες τις συναρτήσεις της κλάσης.

Όταν μια καθολική μεταβλητή έχει το ίδιο όνομα με μια τοπική μεταβλητή η συνάρτηση χρησιμοποιεί την τοπική μεταβλητή. Η τοπική μεταβλητή αποκρύπτει την καθολική με το ίδιο όνομα.
Αυτό φαίνεται στο παράδειγμα του Project12.

Παράδειγμα:

```
1  ' Gambas class file
2  PRIVATE x AS Integer = 10
3
4  PUBLIC SUB Form_Open()
5
6  DIM x AS Integer = 20
7
8  PRINT "x="; x
9
10 END
```

Στην γραμμή 2 δηλώνεται έξω από κάθε συνάρτηση μια ιδιωτική καθολική μεταβλητή η x και λαμβάνει την τιμή 10.
Στην γραμμή 6 δηλώνεται μια τοπική μεταβλητή x και λαμβάνει την τιμή 20.
Όταν τελικά τυπώνουμε στην προκαθορισμένη έξοδο την μεταβλητή x βλέπουμε ότι η τιμή της δεν είναι αυτής της καθολικής μεταβλητής αλλά αυτής της τοπικής. Η τοπική x απόκρυψε την καθολική x.
Αν βάλουμε ένα σχόλιο στην γραμμή 6 και τρέξουμε το πρόγραμμα θα δούμε ότι τώρα τυπώνεται η τιμή της καθολικής μεταβλητής.

```
'DIM x AS Integer = 20
```

Το ίδιο ακριβώς συμβαίνει και στις μεταβλητές εισόδου των συναρτήσεων. Μια μεταβλητή x στην είσοδο μιας συνάρτησης είναι μια τοπική μεταβλητή που αποκρύπτει την καθολική με το ίδιο όνομα.

Η χρήση των καθολικών μεταβλητών θα πρέπει να γίνεται με προσοχή.
Επειδή αυτές οι μεταβλητές μπορούν να αλλάξουν από οποιοδήποτε σημείο της κλάσης και ειδικά αν είναι δημόσιες από οποιοδήποτε σημείο του προγράμματος, δλδ και από άλλες κλάσεις, σε ένα κώδικα με πολλές γραμμές είναι δύσκολο να βρεθεί ένα λάθος που αλλάζει ανεπιθύμητα την καθολική μεταβλητή.
Για αυτόν τον λόγο χρησιμοποιούνται τοπικές μεταβλητές και μπλοκ κώδικα αυστηρά ορισμένα να τελούν μια συγκεκριμένη λειτουργία.

Τα ορίσματα των συναρτήσεων

Τα ορίσματα των συναρτήσεων όπως είδαμε είναι μεταβλητές. Δεν είναι αναγκαίο τα ορίσματα να είναι όλα του ίδιου τύπου δεδομένων. Μια συνάρτηση μπορεί να δεχθεί διαφορετικούς τύπους δεδομένων για ορίσματα αλλά θα επιστρέφει μόνο έναν τύπο δεδομένων.
Για παράδειγμα:

```
PUBLIC FUNCTION MyFunction(a AS Integer, b AS Float, c AS Byte) AS Integer
```

Ένα άλλο σημείο που αξίζει να προσέξετε είναι ότι μπορείτε αντί για μεταβλητές να διοχετεύετε τιμές επιστροφής από άλλες συναρτήσεις. Αυτό όμως δεν συνιστάται διότι κάνει δυσανάγνωστο τον κώδικα.
Για παράδειγμα: Αν έχουμε μια συνάρτηση που τελεί μια πρόσθεση και μια άλλη που τελεί έναν πολλαπλασιασμό μπορούμε να γράψουμε την πρώτη σαν όρισμα της δεύτερης.
Αυτό φαίνεται στο παράδειγμα Project13

Παράδειγμα:

```
1  ' Gambas class file
2  PUBLIC SUB Form_Open()
3
4  PRINT MyFunction2(MyFunction1(1, 5), 5)
5
6  END
7
8  PUBLIC FUNCTION MyFunction1(a AS Integer, b AS Float) AS
9  Integer
10
11   RETURN (a + b)
12
13 END
14
15 PUBLIC FUNCTION MyFunction2(a AS Integer, b AS Float) AS
16 Integer
17
18   RETURN (a * b)
19
20 END
```

Ανάλυση:
Έχουμε δύο συναρτήσεις. Η μία προσθέτει δύο αριθμούς και η δεύτερη πολλαπλασιάζει δύο αριθμούς.

Στην γραμμή 4, έχουμε την έκφραση:

```
PRINT MyFunction2(MyFunction1(1, 5), 5)
```

Στην έκφραση αυτή τυπώνεται στην προκαθορισμένη έξοδο το αποτέλεσμα της MyFunction2().

Το πρώτο όμως όρισμα αυτής της συνάρτησης είναι το MyFunction1(1,5) και το δεύτερο όρισμα το 5.
Εδώ βλέπουμε ότι το πρώτο όρισμα είναι η επιστρέφουσα τιμή της MyFunction1().

Η Gambas πρώτα υπολογίζει την τιμή της συνάρτησης MyFunction1(1,5) και μετά υπολογίζει την τιμή της συνάρτησης MyFunction2(6,5), όπου 6 είναι η τιμή της MyFunction1(1,5).

Έτσι οι πράξεις που γίνονται είναι:
1+5 =6
6*5=30

Η έκφραση αυτή ήταν σαν να γράφαμε PRINT (1+5) * 5, μόνο που το κάναμε με συναρτήσεις.

Με αυτόν τον τρόπο μπορείτε να γράφετε συναρτήσεις και ορίσματα ως συναρτήσεις, μόνο που ο κώδικας σας μπορεί να γίνει δύσκολα κατανοητός και δύσκολα συντηρήσιμος σε περίπτωση λάθους.

Τιμές επιστροφής

Με τον ίδιο τρόπο μπορείτε να γράφετε και τιμές επιστροφής.
Ο απλός τρόπος είναι να επιστρέφετε μία τιμή μιας μεταβλητής. Για παράδειγμα: RETURN x, όμως αντί για μεταβλητή μπορείτε να έχετε μια πιο πολύπλοκη έκφραση που να υπολογίζει πρώτα μια τιμή και να την επιστρέφει.

Το προηγούμενο παράδειγμα του Project13 τροποποιημένο για να βγάζει το ίδιο αποτέλεσμα είναι το Project14.
Ο κώδικας φαίνεται παρακάτω:

Παράδειγμα:

```
1   ' Gambas class file
2   PUBLIC SUB Form_Open()
3
4   PRINT MyFunction2(5)
5
6   END
7
8   PUBLIC FUNCTION MyFunction1(a AS Integer, b AS Float) AS
9   Integer
10
11    RETURN (a + b)
12
13  END
14
15  PUBLIC FUNCTION MyFunction2(a AS Integer) AS Integer
16
17    RETURN (MyFunction1(1, 5) * a)
18
19  END
```

Στην γραμμή 4 καλείται η συνάρτηση MyFunction2() και διοχετεύεται η τιμή 5.
Η ροή του προγράμματος πηγαίνει στην γραμμή 14.
Στην γραμμή 16 καλείται η MyFunction1() και της διοχετεύονται οι τιμές 1,5.
Η ροή του προγράμματος πηγαίνει στην γραμμή 8.
Στην γραμμή 10 επιστρέφεται η τιμή της πρόσθεσης και η ροή του προγράμματος επιστρέφει στην γραμμή 16.
Τελείται ο πολλαπλασιασμός και η τιμή αυτή επιστρέφεται.
Η ροή του προγράμματος επιστρέφει στην γραμμή 4 όπου και τυπώνεται το αποτέλεσμα στην προκαθορισμένη έξοδο.

Στην γραμμή 16 ήταν σαν να είχαμε την πράξη (1+5) * 5, μόνο που αυτή η έκφραση υπολόγισε την τιμή και μετά την επέστρεψε επειδή ήταν μαζί με την λέξη RETURN.

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 ν-1 + ν-2 εφόσον ν>2.

Το πρόγραμμα που θα γράψουμε θα βρίσκει τον νιοστό όρο της σειράς Fibonacci.
Για να τερματίσει το πρόγραμμα θα πρέπει ν <3.

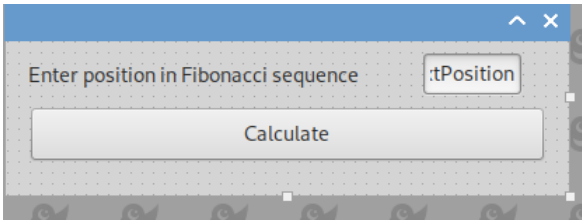
Ο αλγόριθμος του Project15 είναι:

- 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.

Το interface της εφαρμογής είναι όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 65: Η διεπαφή του Project15

Αποτελείται από μία ετικέτα με όνομα lblLabel1, ένα κουτί κειμένου με όνομα txtPosition και ένα κουμπί με όνομα btnCalc.

Ο χρήστης θα εισάγει την θέση του όρου της ακολουθίας στο txtPosition και θα πατά το κουμπί btnCalc για υπολογισμό.

Ας δούμε τον κώδικα.

Παράδειγμα:

```
1   ' Gambas class file
2
3   Public Sub Form_Open()
4
5     Me.Center
6
7   End
8
9   Public Sub Fibonacci(a As Integer) As Integer
10
11    Print "Calculate Fibonacci("; a; ")..."
12
13    If a < 2 Then
14      Print "Return "; a
15      Return a
16    Else
17      Print "Call Fibonacci("; (a - 1); ") and Fibonacci("; (a
    - 2); ").\n"
18      Return (Fibonacci(a - 1) + Fibonacci(a - 2))
19    Endif
20
21  End
22
23  Public Sub btnCalc_Click()
24
25    Dim x As Integer
26    Dim result As Integer
```

```
27    Dim sMessage As String
28
29    x = Val(txtPosition.Text)
30
31    result = Fibonacci(x)
32
33    sMessage = "The " & x & "th Fibonacci number is " & result
34
35    Message.Info(sMessage)
36
37  End
```

Η πρώτη συνάρτηση απλά με το που φορτωθεί η φόρμα την τοποθετεί στο κέντρο της οθόνης

```
3  PUBLIC SUB Form_Open()
4
5    ME.Center
6
7  END
```

Στις γραμμές 9-21 έχουμε τον κώδικα:

```
9   Public Sub Fibonacci(a As Integer) As Integer
10
11    Print "Calculate Fibonacci("; a; ")..."
12
13    If a < 2 Then
14      Print "Return "; a
15      Return a
16    Else
17      Print "Call Fibonacci("; (a - 1); ") and Fibonacci("; (a - 2); ").\
    n"
18      Return (Fibonacci(a - 1) + Fibonacci(a - 2))
19    Endif
20
21  End
```

Στην γραμμή 9 ορίζεται η συνάρτηση Fibonacci(). Η οποία λαμβάνει έναν ακέραιο a και επιστρέφει ένα ακέραιο.

Η γραμμή 11 απλά πληροφορεί ότι θα υπολογιστεί ο a-στος όρος. Πρώτα τυπώνεται το string "Calculate Fibonacci(" ακολουθεί η τιμή του a και ακολουθεί το string ")...". Οι τρεις εκφράσεις χωρίζονται με Ελληνικά ερωτηματικά όπως περιγράφεται στην σύνταξη της εντολής PRINT.

Στην γραμμή 13 έχουμε ένα μπλοκ κώδικα IF...ELSE...ENDIF.

Σε συντομία μετά το IF γράφουμε την έκφραση που θέλουμε να ελέγξουμε, δλδ αν a<2 και

ακολουθεί η λέξη THEN.
Αν η έκφραση είναι TRUE γράφουμε τον κώδικα στο μπλοκ αληθείας.
Στην συγκεκριμένη περίπτωση αν a<2 τυπώνουμε ένα μήνυμα και επιστρέφει η συνάρτηση την τιμή a.

Ακολουθεί το ELSE. Όπου γράφουμε τον κώδικα στο μπλοκ όταν δεν ισχύει το a<2.
Σε αυτό το μπλοκ κώδικα εμφανίζουμε ένα μήνυμα, ότι υπολογίζουμε τον a-1 και a-2 όρο.
Και ακολουθεί μια πολύπλοκη έκφραση RETURN.
Η RETURN πριν επιστρέψει την τιμή καλεί την Fibonacci(a – 1) και Fibonacci(a – 2) και τις προσθέτει.
Σε αυτό το σημείο γίνεται η αναδρομή!

Η RETURN επιστρέφει το άθροισμα από τον όρο a μέχρι τον όρο 1.
Έτσι αν a=10, επιστρέφει το άθροισμα αναδρομικά από το 10 έως το 1.

Στην συνέχεια έχουμε τον κώδικα του πατήματος του κουμπιού.

```
23  PUBLIC SUB btnCalc_Click()
24
25    DIM x AS Integer
26    DIM result AS Integer
27    DIM sMessage AS String
28
29    x = Val(txtPosition.Text)
30
31    result = Fibonacci(x)
32
33    sMessage = "The " & x & "th Fibonacci number is " & result
34
35    Message.Info(sMessage)
36
37  END
```

Δηλώνονται τρεις μεταβλητές. Οι x, result, sMessage που αντίστοιχα είναι δύο ακέραιοι και ένα αλφαριθμητικό.

Η x λαμβάνει την αριθμητική τιμή του string του txtPosition, αφού αυτή έχει μετατραπεί από την συνάρτηση Val(().

Η result δεχεται την τιμή που επιστρέφει η Fibonacci(x).

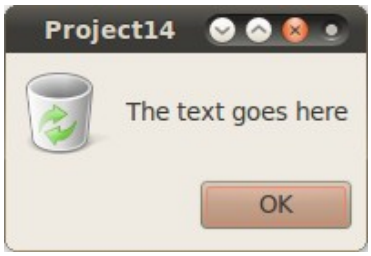
Η sMessage δέχεται ένα string το οποίο αποτελείται από κείμενο εντός διπλών εισαγωγικών που ενώνεται με τον τελεστή &. Αυτό το κάνουμε για να εισάγουμε στο string τις τιμές των μεταβλητών x, result.

Τέλος η γραμμή 35 εμφανίζει ένα μήνυμα πληροφορίας με το κείμενο της μεταβλητής sMessage.

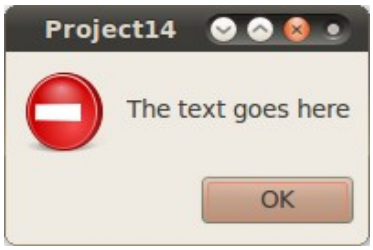
Για να εμφανίσετε κουτιά μηνυμάτων γράφετε **Message.** και η Gambas σας δίνει μια λίστα με τα προκαθορισμένα κουτιά μηνυμάτων.

Τα προκαθορισμένα κουτιά μηνυμάτων είναι:

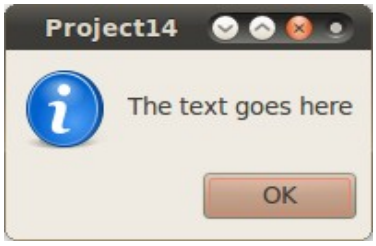
Κουτί μηνύματος διαγραφής



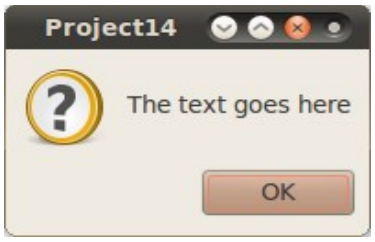
Κουτί μηνύματος λάθους



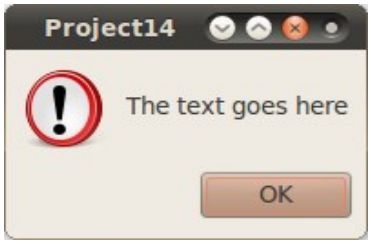
Κουτί μηνύματος πληροφορίας



Κουτί μηνύματος ερώτησης



Κουτί μηνύματος προειδοποίησης



Στα κουτιά μηνυμάτων μπορείτε να ορίσετε τρία κουμπιά όπως OK, CANCEL ή YES, NO και κείμενο προς πληροφόρηση.

Εμείς στο παράδειγμα αυτό χρησιμοποιήσαμε το κουτί πληροφόρησης.

```
Message.Info(sMessage)
```

Κεφάλαιο 7ο – Ροή προγράμματος

Στα κεφάλαια 2 και 6 είδαμε με ποιο τρόπο οι συναρτήσεις όταν τις καλούμε αλλάζουν την ροή του προγράμματος. Εκτός από τις κλήσεις συναρτήσεων υπάρχουν και άλλες εντολές που διακλαδώνουν την ροή του προγράμματος. Τέτοιες εντολές είναι οι εντολές βρόχων (FOR..NEXT, DO...WHILE κτλ), οι εντολές λήψης λογικών αποφάσεων (IF...ELSE...ENDIF), οι εντολές διαχείρισης λαθών (CATCH, FINALLY), η εντολή μετάβασης (GOTO), κτλ.

Εντολές λήψης λογικών αποφάσεων

Choose

Η εντολή Choose επιστρέφει μία τιμή από μία λίστα τιμών.

Σύνταξη:

```
Value = Choose ( Choice , Result #1 , Result #2 [ , ... ] )
```

- Value, είναι η τιμή που επιστρέφει η Choose.
- Choice, είναι ένας ακέραιος διαφορετικός του μηδενός όπου δείχνει ποια τιμή να επιλεχθεί προς επιστροφή από την λίστα τιμών. Η πρώτη τιμή αντιστοιχεί στον αριθμό 1, η δεύτερη στον αριθμό 2 κτλ.
  Αν δεν οριστεί η Choice ή είναι αριθμός αρνητικός ή το μηδέν τότε επιστρέφεται η τιμή του κενού, NULL.
- Result #1 **,** Result #2, ... είναι οι τιμές της λίστας.

Παράδειγμα:

```
1  DIM X AS Integer = 5
2  PRINT Choose(X, "one", "two", "three", "four", "five")
```

Έξοδος

```
five
```

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 ( bTest AS Boolean , vTrue , vFalse )

Value = If ( bTest AS Boolean , vTrue , vFalse )
```

- Value, είναι η τιμή που επιστρέφει η IIf ή η If. Η Value μπορεί να είναι οποιοσδήποτε τύπος δεδομένων και έτσι ορίζεται ως Variant.
- bTest, η έκφραση η οποία ελέγχεται αν είναι αληθής ή ψευδής. Επειδή η έκφραση επιστρέφει έναν Boolean ορίζεται ως τύπος δεδομένου Boolean.
- vTrue, η τιμή που επιστρέφει όταν η bTest είναι TRUE. Η vTrue μπορεί να είναι οποιοσδήποτε τύπος δεδομένων και έτσι ορίζεται ως Variant.
- vFalse, η τιμή που επιστρέφει όταν η bTest είναι FALSE. Η vFalse μπορεί να είναι οποιοσδήποτε τύπος δεδομένων και έτσι ορίζεται ως Variant.

Παράδειγμα:

```
1 DIM X AS Integer = 3
2 PRINT IIf(x > 10, "X is greater than 10", "x is lower than 10
  ")
```

Έξοδος

```
x is lower than 10
```

IF ... THEN ... ELSE IF ...THEN ... ELSE ... ENDIF

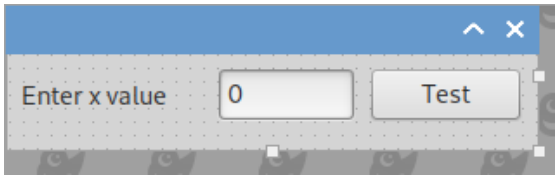
Το IF...THEN...ELSE IF...THEN...ENDIF ονομάζεται έκφραση συνθήκης.
Μετά το IF ελέγχεται μία πρόταση αν είναι αληθής.
Αν είναι εκτελείται ο κώδικας μετά το THEN.
Διαφορετικά μπορούμε να κάνουμε και άλλον λογικό έλεγχο αληθείας με την ELSE IF.
Μπορούμε να έχουμε όσους λογικούς ελέγχους θέλουμε με το ELSE 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 [ { AND IF | OR IF } Expression ... ] [ THEN ]
...
[ ELSE IF Expression [ { AND IF | OR IF } Expression ... ] [ THEN ]
... ]
[ ELSE
... ]
ENDIF
```

Expression, είναι η έκφραση που ελέγχεται αν είναι αληθής.
Όπως φαίνεται από την σύνταξη μπορούμε να έχουμε και σύνθετες IF συνδυάζοντας τες με AND ή OR.

Παράδειγμα:
Ας δούμε την IF με ένα παράδειγμα. Δημιουργείστε το Project16 με το εξής interface.



Εικόνα 66: Η διεπαφή (interface) του Project15

Δημιουργείστε ένα Label με όνομα lblLabel1, ένα Textbox με όνομα txtTextBox1, και ένα Button, με όνομα btnButton1.

Ο κώδικας του Project16 είναι ο εξής:
Παράδειγμα:

```
1  ' Gambas class file
2
3  PUBLIC SUB Form_Open()
4    ME.Center
5  END
6
7  PUBLIC SUB btnButton1_Click()
8
9  DIM x AS Integer
10
11 x = Val(txtTextBox1.Text)
12
13 IF x = 10 THEN
14   Message.Info("x is equal 10")
15
16 ELSE IF x > 10 THEN
17   Message.Info("x is greater than 10")
18
```

```
19 ELSE
20   Message.Info("x is lower than 10")
21
22 ENDIF
23
24 END
```

Όταν φορτώνεται η φόρμα την τοποθετούμε στο κέντρο της οθόνης

```
PUBLIC SUB Form_Open()
  ME.Center
END
```

μετατρέπουμε το κείμενο του txtTextBox1 σε αριθμητική τιμή και εκχωρούμε την τιμή σε μία μεταβλητή ακεραίου.

```
DIM x AS Integer

x = Val(txtTextBox1.Text)
```

Τέλος ελέγχουμε την τιμή της x και σε κάθε περίπτωση εμφανίζουμε ένα μήνυμα στον χρήστη με το Message.Info().

```
IF x = 10 THEN
  Message.Info("x is equal 10")

ELSE IF x > 10 THEN
  Message.Info("x is greater than 10")

ELSE
  Message.Info("x is lower than 10")

ENDIF
```

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 Expression [ TO Expression #2 ] [ , ... ]
... ]

[ CASE Expression [ TO Expression #2 ] [ , ... ]
... ]

[ { CASE ELSE | DEFAULT }
... ]

END SELECT
```

Expression, είναι οι εκφράσεις που θα συγκριθούν.
Αν καμία σύγκριση δεν επιστρέψει TRUE μπορούμε να δηλώσουμε προκαθορισμένη εκτέλεση κώδικα με το **CASE ELSE | DEFAULT**.
Το **END SELECT** τερματίζει το μπλοκ κώδικα της **SELECT**.

Παράδειγμα:
Ας δούμε ένα παράδειγμα.

Δημιουργείστε ένα Project17 με ακριβώς το ίδιο interface με το Project16.

Τώρα θα γράψουμε έναν κώδικα SELECT αντί ενός IF και θα φανούν οι διαφορές των δύο.

Παράδειγμα:

```
1   ' Gambas class file
2
3   PUBLIC SUB Form_Open()
4     ME.Center
5   END
6
7   PUBLIC SUB btnButton1_Click()
8
9   DIM x AS Integer
10
11  x = Val(txtTextBox1.Text)
12
13  SELECT CASE x
```

```
14  CASE 1
15    Message.Info("i am in case1: x is equal 1")
16
17  CASE 2
18    Message.Info("i am in case2: x is equal 2")
19
20  CASE 20
21    Message.Info("i am in case3: x is equal 20")
22
23  CASE 30
24    Message.Info("i am in case4: x is equal 30")
25
26  CASE ELSE
27    Message.Info("i am in case else: x is equal " & x)
28
29  END SELECT
30
31  END
```

Εδώ όπως φαίνεται η τιμή της μεταβλητής x συγκρίνεται με τις περιπτώσεις των τιμών 1, 2, 20, 30 και όποια τιμή μετά την σύγκριση είναι αληθής εκτελείται το αντίστοιχο μπλοκ κώδικα. Αν καμία δεν είναι αληθής εκτελείται ο κώδικας της CASE ELSE.

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 Variable = Expression TO Expression [ STEP Expression ]
 ...
NEXT
```

Variable, είναι το όνομα μιας μεταβλητής όπου θα την ελέγχει ο βρόχος FOR...NEXT.
Expression, είναι μια έκφραση η οποία εκχωρείται στην μεταβλητή ως αρχική τιμή.
TO Expression, είναι μια έκφραση η οποία ορίζει το τέλος του βρόχου.
STEP Expression, είναι μια προαιρετική έκφραση που δείχνει το βήμα μεταβολής εάν δεν οριστεί το προκαθορισμένο βήμα είναι 1.

Η μεταβλητή θα πρέπει να είναι τύπου δεδομένων Byte, ή Short, ή Integer, ή Long ή Float.
Η αρχική έκφραση θα πρέπει να είναι μικρότερη της τελικής για θετικές τιμές μεταβολής, ενώ για αρνητικές τιμές μεταβολής θα πρέπει να είναι μεγαλύτερη της τελικής.

Αν δεν συμβαίνει αυτό η ροή του προγράμματος δεν θα εισέλθει καθόλου στον βρόχο.

Παράδειγμα:
Δημιουργήστε το Project18.
Η φόρμα έχει ένα TextArea control με το όνομα txtTextArea1.
Ο κώδικας είναι ο εξής:

Παράδειγμα:

```
1   ' Gambas class file
2
3   PUBLIC SUB Form_Open()
4
5   DIM i AS Integer
6
7   ME.Center
8   txtTextArea1.Text = ""
9
10  FOR i = 32 TO 126
11    txtTextArea1.Text &= i & "\n"
12  NEXT
13
14  END
```

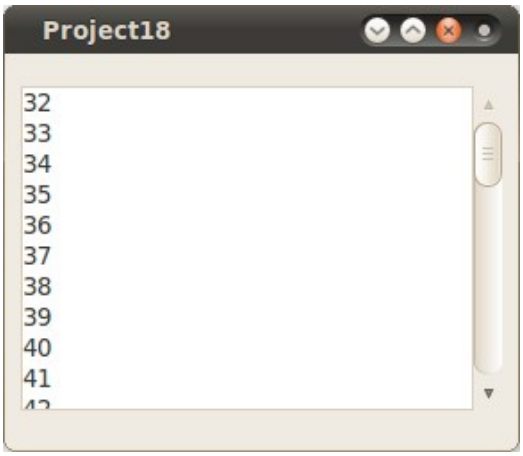
Στην γραμμή 5 ορίζεται μια μεταβλητή ακεραίου.
Στην γραμμή 7 η φόρμα κεντράρεται στην οθόνη.
Στην γραμμή 8 η τιμή του κειμένου του TextArea control ορίζεται κενή, έτσι διαγράφουμε κάποιο κείμενο από ένα control.

Στην γραμμή 10 ξεκινάει ο βρόχος FOR.
Εκχωρείται η τιμή 32 στην μεταβλητή i και η συνθήκη εξόδου είναι όταν εκχωρηθεί και η τιμή 126 στην μεταβλητή.

Στην γραμμή 11 επαναλαμβάνεται ο κώδικας μέσα στον βρόχο 126-32=94 φορές.
Η γραμμή αυτή λέει το κείμενο του txtTextArea1 θα συμπληρώνεται με την τιμή της μεταβλητής i και έναν χαρακτήρα νέας γραμμή.
Ο τελεστής & ενώνει δύο string, εδώ ενώνει την τιμή του i με το “\n”.
Ο τελεστής &= συμπληρώνει συνεχώς ενώνοντας στην προηγούμενη τιμή την νέα τιμή των string.
Με άλλα λόγια:
όταν η i είναι 32 το κείμενο θα είναι “32\n”.

όταν η i είναι 33 το κείμενο θα είναι “32\n33\n”.
όταν η i είναι 34 το κείμενο θα είναι “32\n33\n34\n”.
Ο χαρακτήρας \n όταν τυπώνεται δημιουργεί μια νέα γραμμή, έτσι το control txtTextArea1 θα έχει κείμενο:

32
33
34
35
36
....
κτλ από το 32 έως το 126 που ορίζει ο βρόχος.



Εικόνα 67: Το Project18 όταν τρέχει.

Ας τροποποιήσουμε το Project18 ώς εξής και ας το σώσουμε ως Project18a:
Παράδειγμα:

```
1   ' Gambas class file
2
3   PUBLIC SUB Form_Open()
4
5   DIM i AS Integer
6
7   ME.Center
8   txtTextArea1.Text = ""
9
10  FOR i = 32 TO 126
11  '  txtTextArea1.Text &= i & "\n"
12    txtTextArea1.Text &= Chr(i) & "\n"
13  NEXT
14
15  END
```

Η αλλαγή είναι στις γραμμές 11, 12. Η 11 απενεργοποιήθηκε με το να γίνει ένα σχόλιο.
Στην 12 τώρα το κείμενο του txtTextArea1 είναι ο χαρακτήρας του ASCII κώδικα με την τιμή i και ένας χαρακτήρας νέας γραμμής για να αλλάζουμε γραμμή.
Η συνάρτηση Chr() λαμβάνει μια τιμή ακεραίου που αντιστοιχεί σε κάποιο σύμβολο του πίνακα ASCII.
Για τον πίνακα ASCII δείτε στο παράρτημα.

Αν θέλουμε να εμφανίσουμε στο κείμενο του txtTextArea1 κάτι σαν και αυτό:
Ascii code (33) = !
θα πρέπει να διαμορφώσουμε ανάλογα την γραμμή 12. Σε αυτήν την περίπτωση η γραμμή θα είναι.

```
12   txtTextArea1.Text &= "Ascii code (" & i & ")=" & Chr(i) & "\n"
```

Ας εξηγήσουμε αυτήν την γραμμή.
Εμείς θέλουμε να διαμορφώσουμε ένα string το οποίο θα είναι για παράδειγμα το “Ascii code (33) = !”.
Αυτό το string αποτελείται από σταθερά μέρη και μεταβλητά.
Τα μεταβλητά μέρη είναι το 33 και το !. Τα μέρη αυτά είναι η τιμή της μεταβλητής i και η συνάρτηση Chr(i).
Επομένως αν είχαμε μια σύνθεση των σταθερών μερών και των μεταβλητών θα είχαμε κάτι σαν και αυτό:

Ascii code (i) = Chr(i)

Με πράσινο χρώμα είναι τα σταθερά μέρη και με κόκκινο τα μεταβλητά.
Ένα σταθερό μέρος είναι ένα string ανάμεσα σε διπλά εισαγωγικά. Οπότε η παραπάνω πρόταση θα ήταν.
“Ascii code (”i“) =” Chr(i)

Ο τρόπος για να ενώσουμε δύο string είναι με τον τελεστή ένωσης &. Οπότε τα σταθερά μέρη και τα μεταβλητά ενωμένα θα ήταν.

“Ascii code (” & i & “) =” & Chr(i)
Τέλος προσθέτουμε και τον χαρακτήρα νέας γραμμή και έχουμε:

```
"Ascii code (" & i & ")=" & Chr(i) & "\n"
```

ή όλο μαζί εκχωρημένο στην τιμή κειμένου του control txtTextArea1 αθροιστικά.

```
txtTextArea1.Text &= "Ascii code (" & i & ")=" & Chr(i) & "\n"
```

Αυτό ήταν και ένα παράδειγμα πολύπλοκης ένωσης string!

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 ή μιας υπορουτίνας σε άλλους βρόχους ή υπορουτίνες. 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.

Για παράδειγμα,

```
FOR i = 32 TO 126
  txtTextArea1.Text &= Chr(i) & "\n"
NEXT
```

η πρόταση **txtTextArea1.Text &= Chr(i) & "\n"** γράφεται εσωτερικά στο μπλοκ FOR...NEXT με μια εσοχή προς τα δεξιά.
Η πρόταση αυτή είναι μια φωλιασμένη πρόταση.
Φωλιασμένο μπορεί να είναι ακόμη και ένα μπλοκ κώδικα. Για παράδειγμα ένα FOR...NEXT μέσα σε ένα άλλο FOR...NEXT.
Για παράδειγμα αν θέλουμε να εμφανίσουμε την προπαίδεια θα έχουμε τον εξής κώδικα.
Δημιουργείστε το Project19.

Παράδειγμα:

```
1   ' Gambas class file
2
3   PUBLIC SUB Form_Open()
4
5   DIM i AS Integer
6   DIM q AS Integer
7   DIM r AS Integer
8
9   FOR i = 1 TO 10
10      PRINT "************** Προπαίδεια του "; i; "
    **************"
11      FOR q = 1 TO 10
12        r = i * q
13        PRINT i; "X"; q; "="; r
14      NEXT
15  NEXT
16
17  END
```

Έξοδος

```
************** Προπαίδεια του 1 **************
1X1=1
1X2=2
1X3=3
1X4=4
1X5=5
1X6=6
1X7=7
1X8=8
1X9=9
1X10=10
************** Προπαίδεια του 2 **************
2X1=2
2X2=4
2X3=6
2X4=8
2X5=10
```

κτλ ...

Ανάλυση κώδικα

Στις γραμμές 5,6,7 δηλώνουμε τρεις μεταβλητές ακεραίων.
Στην γραμμή 9 υπάρχει ο πρώτος βρόχος FOR.
Μέσα σε αυτόν είναι ένθετες οι προτάσεις από την γραμμή 10 έως την 14.
Η γραμμή 11 δηλώνει έναν άλο βρόχο FOR ο οποίος αποτελεί μέρος του φωλιάσματος του πρώτου βρόχου FOR.
Οι γραμμές 12,13 είναι προτάσεις φωλιασμένες στον δεύτερο βρόχο.
Οι γραμμές 10,11,12,13,14 είναι προτάσεις φωλιασμένες στον πρώτο βρόχο.

DO...LOOP

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

Σύνταξη:

```
DO [ WHILE Condition ]
.
.
.
[ BREAK | CONTINUE ]
.
.
.
LOOP [ UNTIL Condition ]
```

- **DO**, απαραίτητη λέξη, πάντα η πρώτη λέξη του βρόχου, δηλώνει την αρχή του βρόχου.
- **WHILE**, προαιρετική λέξη, δηλώνει την συνθήκη αληθείας για να εκτελείται ο βρόχος.
- **Condition**, μία συνθήκη αληθείας, είναι μια έκφραση που επιστρέφει πάντα έναν Boolean.
- **BREAK**, προαιρετική λέξη, δηλώνει την άμεση έξοδο από τον βρόχο και την ροή του προγράμματος στην επόμενη πρόταση εκτός βρόχου.
- **CONTINUE**, προαιρετική λέξη, δηλώνει την εγκατάλειψη των προτάσεων που ακολουθούν και ανακατευθύνει την ροή του προγράμματος στην αρχή του βρόχου.
- **LOOP**, απαραίτητη λέξη δηλώνει το τέλος του βρόχου.
- **UNTIL**, προαιρετική λέξη, δηλώνει την συνθήκη η οποία πρέπει να είναι αληθής για τον τερματισμό εκτέλεσης του βρόχου.

Παραδείγματα:
Δημιουργείστε το Project20. Το Project20 δεν έχει γραφικό περιβάλλον. Έτσι δεν έχει φόρμες αλλά μία μονάδα, ένα module σαν κλάση εκκίνησης, η οποία έχει την συνάρτηση Main().

Παράδειγμα:

```
1   ' Gambas module file
2
3   PUBLIC SUB Main()
4
5   DIM a AS Variant
6
7   DO WHILE TRUE
8
9     a = Val(InputBox("Enter an integer between 1-4, 0 for
10  exit"))
11
12    SELECT CASE a
13    CASE 0
14      PRINT "\nExit!!!"
15      QUIT
16    CASE 1
17      Example1()
18
19    CASE 2
20      Example2()
21
22    CASE 3
23      Example3()
24
25    CASE 4
26      Example4()
27
28    CASE ELSE
29      PRINT "\nNothing to do..."
30
31    END SELECT
32
33  LOOP
34
35  END
36
37
38  PUBLIC SUB Example1()
39    DIM x AS Integer = 1
40    PRINT "\nExample1:"
41    DO WHILE x < 10
42      PRINT x; ", ";
43      INC x
44    LOOP
```

```
45  END
46
47  PUBLIC SUB Example2()
48    DIM x AS Integer = 1
49    PRINT "\nExample2:"
50    DO WHILE x <= 10
51      PRINT x; ", ";
52      INC x
53    LOOP
54  END
55
56  PUBLIC SUB Example3()
57    DIM x AS Integer = 1
58    PRINT "\nExample3:"
59    DO WHILE x <= 10
60      PRINT x; ", ";
61      INC x
62      IF x = 5 THEN BREAK
63    LOOP
64  END
65
66  PUBLIC SUB Example4()
67    DIM x AS Integer = 1
68    PRINT "\nExample4:"
69    DO
70      PRINT x; ", ";
71      INC x
72    LOOP UNTIL x = 10
73  END
```

Ο αλγόριθμος του παραδείγματος αυτού είναι ο εξής:
Ο χρήστης εισάγει μία τιμή από το 1-4 ή 0 για έξοδο.
Κάθε τιμή οδηγεί την ροή του προγράμματος σε μία συνάρτηση παραδείγματος.
Κάθε συνάρτηση είναι ένα παράδειγμα της DO...WHILE.
Η διαδικασία επαναλαμβάνεται μέχρι ο χρήστης να επιλέξει το 0.

Ανάλυση
Στην γραμμή 3 έχουμε την δήλωση της Main(), είναι η ρουτίνα εκκίνησης.

Στην γραμμή 5 δηλώνουμε μια μεταβλητή Variant για να εκχωρούμε την τιμή από ένα κουτί εισόδου τιμών (InputBox). Ο τύπος δεδομένων που επιλέχτηκε είναι Variant διότι οι τιμές από το InputBox μπορεί να είναι αλφαρηθμιτικές. Εμείς την είσοδο από το InputBox την μετατρέπουμε σε έναν ακέραιο με την συνάρτηση Val() για να χρησιμοποιήσουμε αυτήν την τιμή σε μια δομή SELECT. Αν ο χρήστης εισάγει έναν χαρακτήρα η Val() επιστρέφει NULL και εμφανίζεται ένα σφάλμα αν η μεταβλητή ήταν integer. Ενώ με τον γενικό τύπο Variant το

εμπόδιο αυτό παρακάμπτεται.

Στην γραμμή 7 ξεκινά ένας βρόχος DO..LOOP με τέλος στην γραμμή 32.
Ο βρόχος αυτός είναι ατέρμων. Δλδ δεν τελειώνει ποτέ διότι δεν έχει συνθήκη τερματισμού.
Για να τερματίσει λοιπόν το πρόγραμμα χρησιμοποιούμε την πρόταση QUIT ως μια επιλογή του SELECT. Αν ο χρήστης εισάγει το 0 τότε τερματίζεται το πρόγραμμα.

Στην γραμμή 9 εκχωρούμε την τιμή του InputBox στην μεταβλητή a, αφού έχουμε μετατρέψει την τιμή σε αριθμητική τιμή από string με την συνάρτηση Val().

Στην γραμμή 11 αρχίζει η δομή SELECT η οποία τελειώνει στην γραμμή 30.
Η δομή SELECT ελέγχει την τιμή της a. Για κάθε τιμή καλείται μία υπορουτίνα. Αν η τιμή είναι 0 τότε εμφανίζεται ένα μήνυμα εξόδου και το πρόγραμμα τερματίζει.

Από την γραμμή 37 και κάτω έχουμε τις υπορουτίνες.
Κάθε υπορουτίνα ορίζει μια μεταβλητή x με την τιμή 1.
Εμφανίζει ένα μήνυμα για το ποια υπορουτίνα είναι, π.χ. Example1. Το \n σημαίνει ότι αυτό που ακολουθεί εμφάνισε το σε νέα γραμμή. Είναι όπως έχουμε δει ο χαρακτήρας new line.
Έπεται ένας βρόχος DO...WHILE που σε κάθε παράδειγμα δείχνει και από ένα χαρακτηριστικό του βρόχου.

Η πρόταση
PRINT x; ", ";
αναλύεται ως εξής,
τύπωσε το x,
τύπωσε στην ίδια γραμμή το “, ”
τύπωσε στην ίδια γραμμή αυτό που θα ακολουθήσει.
Ο χαρακτήρας ; είναι αυτός που κάνει την PRINT να τυπώνει στην ίδια γραμμή.

Στο Example1 η συνθήκη τερματισμού είναι x < 10. Έτσι το αποτέλεσμα είναι:

Έξοδος:

```
Example1:
1, 2, 3, 4, 5, 6, 7, 8, 9,
```

Το x τυπώνεται όσο είναι μικρότερο του 10. Η τιμή 10 δεν τυπώνεται.

Στο Example2 η συνθήκη τερματισμού είναι x<=10. Έτσι εδώ θα τυπωθεί και το 10.

Έξοδος:

```
Example2:
1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
```

Στο Example3 υπάρχει εντός του DO η πρόταση **IF x = 5 THEN BREAK**.
Όταν το x έχει την τιμή 5 τότε η ροή του προγράμματος θα βγει από τον βρόχο.

Έξοδος:

```
Example3:
1, 2, 3, 4,
```

Στο Example4 γίνεται επίδειξη του LOOP UNTIL. Οι τιμές εδώ που τυπώνονται είναι από 1 έως 9. Το παράδειγμα αυτό είναι αντίστοιχο του Example1. Εδώ το πρόγραμμα θα εισέλθει τουλάχιστον μία φορά στον βρόχο και στο τέλος θα γίνει ο έλεγχος εξόδου από τον βρόχο. Ενώ στο Example1 το πρόγραμμα μπορεί να μην εισέλθει στον βρόχο διότι η συνθήκη τερματισμού είναι στην αρχή.

Έξοδος:

```
Example4:
1, 2, 3, 4, 5, 6, 7, 8, 9,
```

Τέλος αν ο χρήστης δεν εισάγει τιμές από το 1 έως το 4 εμφανίζεται το μήνυμα **Nothing to do...**
Και αν επιλέξει το 0 το μήνυμα **Exit!!!**

Έξοδος:

```
Nothing to do...

Exit!!!
```

REPEAT...UNTIL

Ο βρόχος εκτελεί μία φορά τουλάχιστον το μπλοκ κώδικα και στο τέλος του βρόχου ορίζεται η συνθήκη τερματισμού.

Σύνταξη:

```
REPEAT
.
.
.
[ BREAK | CONTINUE ]
.
.
.
UNTIL Condition
```

Ο βρόχος αυτός είναι συνώνυμος του
DO...LOOP UNTIL.
Για περισσότερα δείτε τον βρόχο DO...LOOP.

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
.
.
.
[ BREAK | CONTINUE ]
.
.
.
WEND
```

Ο βρόχος αυτός είναι συνώνυμος του
DO WHILE...LOOP.
Για περισσότερα δείτε τον βρόχο DO...LOOP.

FOR EACH...NEXT

Ενεργεί ένα βρόχο απαρίθμησης ενός αντικειμένου, μιας συλλογής (Collection) ή ενός πίνακα (Array).

Τα απαριθμούμενα αντικείμενα είναι αντικείμενα που περιέχουν ένα ευρετήριο και μια διάταξη μεταβλητών. Ένα απαριθμούμενο αντικείμενο αντιστοιχεί κάθε τιμή του ευρετηρίου σε μια μεταβλητή και έτσι η προσπέλαση των μεταβλητών γίνεται βάσει της τιμής στο ευρετήριο.

Τέτοιες δομές δεδομένων για παράδειγμα είναι οι πίνακες (Arrays).
Η δομή ενός απλού πίνακα είναι:
Names[0]=”Kostas”
Names[1]=”Petros”
Names[2]=”Nikos”
Names[3]=”Maria”
...

Ο πίνακας Names[] είναι μια δομή δεδομένων που μπορεί να απαριθμείται (enumarated).
Ο απαριθμητός τύπος δεδομένων μπορεί να δεχθεί και να χειριστεί μια ακολουθία συγκεκριμένων τιμών με καθορισμένη σειρά. Για περισσότερα στο κεφάλαιο για τους “Πίνακες”.

Ο βρόχος FOR EACH...NEXT εκτελεί μια απαρίθμηση σε ένα απαριθμούμενο αντικείμενο ή τύπο δεδομένων.

Σύνταξη:

```
FOR EACH Variable IN Expression
...
NEXT
```

- **FOR EACH**, απαραίτητη πρόταση που δηλώνει την αρχή του βρόχου
- Variable, μια μεταβλητή που στην οποία θα εκχωρείται κάθε τιμή της απαρίθμησης
- **IN**, απαραίτητη λέξη που δηλώνει ότι ακολουθεί έκφραση απαρίθμησης
- Expression, απαραίτητη έκφραση η οποία επιστρέφει ένα απαριθμούμενο αντικείμενο ή τύπο δεδομένων
- **NEXT**, απαραίτητη λέξη όπου δηλώνει την επανάληψη του βρόχου.

Ας δούμε ένα παράδειγμα. Δημιουργείστε το Project21 το οποίο θα περιέχει μόνο μία μονάδα module με την συνάρτηση Main(). Ορίστε το module αυτό σαν κλάση εκκίνησης.

Παράδειγμα:

```
1   ' Gambas module file
2
3   PUBLIC SUB Main()
4
5   DIM Names AS NEW String[4]
6   DIM Element AS String
7
8   Names[0] = "Kostas"
9   Names[1] = "Petros"
10  Names[2] = "Nikos"
11  Names[3] = "Maria"
12
13  FOR EACH Element IN Names
14    PRINT Element
15  NEXT
16
17  END
```

Έξοδος:

```
Kostas
Petros
Nikos
Maria
```

Ανάλυση:
Στην γραμμή 3 ορίζεται η ρουτίνα Main().
Στην γραμμή 5 δηλώνεται ένας πίνακας με τύπο δεδομένων String και 4 θέσεις.
Ένας πίνακας διαφέρει από μια μεταβλητή στην δήλωση του διότι ο τύπος δεδομένων του ακολουθείται από [].
Εντός των αγκυλών ορίζονται οι θέσεις του πίνακα ή αλλιώς οι διαστάσεις του.
Στο παράδειγμα μας ο πίνακας Names έχει 4 θέσεις.
Στην γραμμή 6 ορίζεται μια μεταβλητή String.

Στις γραμμές 8 με 11 εκχωρούνται τιμές στις θέσεις του πίνακα Names.
Μια θέση του πίνακα προσπελάζεται με το να ορίσουμε την τιμή του ευρετηρίου του. Οι προκαθορισμένες τιμές ευρετηρίου είναι ακέραιοι με τιμή εκκίνησης το 0.

Έτσι η πρώτη θέση του πίνακα Names έχει τιμή ευρετηρίου (Index) 0,
Η δεύτερη θέση του πίνακα Names έχει τιμή ευρετηρίου, 1
κτλ

Περισσότερα δείτε στο κεφάλαιο για τους πίνακες.

Στην γραμμή 13 ξεκινάει ο βρόχος FOR EACH.
Ο οποίος τυπώνει κάθε τιμή κάθε στοιχείου του πίνακα.

Ο βρόχος FOR EACH ξεκινά να απαριθμεί τον πίνακα Names προσπελάζοντας κάθε θέση του και εκχωρώντας την τιμή της στην μεταβλητή Element.

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.

Δημιουργείστε το Project22 με μία μονάδα Module η οποία θα είναι κλάση εκκίνησης με την ρουτίνα Main() και τον παρακάτω κώδικα.

Παράδειγμα:

```
1   ' Gambas module file
2
3   PUBLIC SUB Main()
4
5     DIM i AS Integer
6
7     FOR i = 0 TO 10
8       PRINT i
9       IF i = 5 THEN GOTO OutOfFor
10    NEXT
11
12    OutOfFor:
13    PRINT "I am out of FOR...NEXT now."
14
15  PRINT "Next lines continue here."
16
17  END
```

Ανάλυση:
Στην γραμμή 5 δηλώνεται μια μεταβλητή ακεραίου.
Στην γραμμή 7 ξεκινά ένας βρόχος FOR.
Στην γραμμή 8 τυπώνεται η τιμή της μεταβλητής.
Στην γραμμή 9 γίνεται ένας έλεγχος αν το i έχει την τιμή 5. Εάν την έχει τότε εκτελείτε μια εντολή GOTO.
Η εντολή GOTO οδηγεί την ροή του προγράμματος σε ένα σημείο του κώδικα με την ετικέτα OutOfFor.
Η ροή του προγράμματος συνεχίζεται από την ετικέτα και κάτω.
Έτσι στην γραμμή 13 τυπώνεται το μήνυμα ότι τώρα το πρόγραμμα βρίσκεται εκτός FOR.
Αλλά συνεχίζει η εκτέλεση του προγράμματος και στην συνέχεια η γραμμή 15 τυπώνει ότι έπονται οι επόμενες εντολές στις επόμενες σειρές.

Μια GOTO αλλάζει την ροή του προγράμματος χωρίς η ροή του προγράμματος να επιστρέφει πίσω στην γραμμή της κλήσης της GOTO όπως γίνεται όταν καλούμε μια υπορουτίνα ή μια συνάρτηση.

Αυτό το χαρακτηριστικό οδήγησε στο παρελθόν σε συγγραφή πολύ μπερδεμένων κωδίκων που ειρωνικά ονομάζονταν “κώδικες σπαγκέτι”.

Η εντολή GOTO δεν μπορεί ωστόσο να χρησιμοποιηθεί για να εισέλθει το πρόγραμμα σε μια άλλη δομή, όπως ένα FOR..NEXT.
Το παρακάτω παράδειγμα είναι λάθος.

Παράδειγμα:

```
1   ' Gambas module file
2
3   PUBLIC SUB Main()
4
5     DIM i AS Integer
6
7     FOR i = 0 TO 10
8       PRINT i
9       IF i = 5 THEN GOTO OutOfFor
10    NEXT
11
12    FOR i = 0 TO 20
13      OutOfFor:       'This is forbidden
14      PRINT "I am in a new FOR...NEXT now."
15    NEXT
16
17  PRINT "Next lines continue here."
18
19  END
```

Εντολές διαχείρισης λαθών

Όταν συμβεί ένα λάθος μεταγλώττισης η Gambas διακόπτει την εκτέλεση του προγράμματος και εμφανίζει ένα μήνυμα λάθους.
Η Gambas δίνει την δυνατότητα να μην διακοπεί η εκτέλεση του προγράμματος και να διαχειριστούμε ένα λάθος μεταγλώττισης με τις εντολές διαχείρισης λαθών.

CATCH

Δηλώνει την αρχή του μπλοκ κώδικα διαχείρισης λαθών.

Σύνταξη:

```
SUB Function ( ... )
...
CATCH
...
END
```

Εάν συμβεί ένα λάθος στο μπλοκ κώδικα της CATCH εμφανίζεται ένα μήνυμα λάθους. Η CATCH δεν διαχειρίζεται τα δικά της λάθη.

Η διαχείριση των λαθών γίνεται με την βοήθεια της κλάσης Error.
Για οδηγίες χρήσεως της κλάσης Error δείτε το αρχείο βοήθειας της Gambas.
Εδώ θα αναφέρουμε την βασική χρήση της κλάσης Error.

Το μπλοκ της εντολής CATCH έχει αρχή αλλά όχι λέξη τέλους όπως για παράδειγμα άλλα μπλοκ όπως το IF...ENDIF.

Το τέλος του μπλοκ CATCH είναι το τέλος της ρουτίνας δλδ η λέξη END.

Αν υπάρχει κώδικας από την λέξη FINALLY το μπλοκ αυτό θα πρέπει να προηγείται του μπλοκ κώδικα της CATCH.

Ας δούμε ένα παράδειγμα με το Project23. Δημιουργήστε το Project23, το οποίο έχει μια κλάση εκκίνησης μία μονάδα module.
Το παρακάτω παράδειγμα δημιουργεί επίτηδες ένα λάθος, το οποίο το διαχειριζόμαστε στο μπλοκ της CATCH.

Παράδειγμα:

```
1   ' Gambas module file
2
3   PUBLIC SUB Main()
4
5     DIM i AS String
6
7     FOR i = 0 TO 10
8       PRINT i
9     NEXT
10
11    CATCH
12    PRINT Error.Text
13    PRINT Error.Where
14    PRINT Error.Code
15    Error.Clear
16
17  END
```

Έξοδος:

```
Type mismatch: wanted Number, got String instead
Module1.Main.7
6
```

Ανάλυση:
Στην γραμμή 5 δηλώνουμε μια αλφαριθμητική μεταβλητή.
Όμως στην γραμμή 7 εκχωρούμε στην μεταβλητή έναν ακέραιο.
Σε αυτό το σημείο εμφανίζεται ένα λάθος και η ροή του προγράμματος πηγαίνει στην CATCH και στον κώδικα που ακολουθεί.

Στην γραμμή 12 εμφανίζεται το μήνυμα λάθους.
Στην γραμμή 13 εμφανίζεται το που βρίσκεται το λάθος, σε ποιο αρχείο και ποια γραμμή κώδικα.
Στην γραμμή 14 εμφανίζεται ο αριθμός του λάθους όπως έχει οριστεί η αρίθμηση των λαθών από την Gambas.
Στην γραμμή 15 καθαρίζουμε το λάθος και επιτρέπουμε την συνέχιση της εκτέλεσης του προγράμματος.

DEBUG

Εμφανίζει εκφράσεις στην προκαθορισμένη έξοδο λαθών.

Σύνταξη:

```
DEBUG Expression [ { ; | ;; | , } Expression ... ] [ { ; | ;; | , } ]
```

- Αν δεν υπάρχει Ελληνικό ερωτηματικό τότε ένας χαρακτήρας νέας γραμμής τυπώνεται (\n).
- Αν υπάρχει Ελληνικό ερωτηματικό η έκφραση που ακολουθεί τυπώνεται αμέσως μετά.
- Αν υπάρχει διπλό Ελληνικό ερωτηματικό τότε μεταξύ των εκφράσεων τυπώνεται το κενό διάστημα.
- Αν υπάρχει ένα κόμμα ανάμεσα στις εκφράσεις τότε τυπώνεται ένας χαρακτήρας TAB.

Το Project23 σώστε το ως Project23a και προσθέστε την έκφραση

```
DEBUG "An error!"
```

μετά την CATCH.

Παράδειγμα:

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5    DIM i AS String
6
7    FOR i = 0 TO 10
8      PRINT i
9    NEXT
10
11   CATCH
12   DEBUG "An error!"
13   PRINT Error.Text
14   PRINT Error.Where
15   PRINT Error.Code
16   Error.Clear
17
18 END
```

Έξοδος:

```
Module1.Main.12: An error!
Type mismatch: wanted Number, got String instead
Module1.Main.7
6
```

Η DEBUG τυπώνει το όνομα του αρχείου που εμφανίζει το λάθος, την γραμμή στην οποία η DEBUG υπάρχει, και το μήνυμα του προγραμματιστή.

Η διαφορά των εντολών PRINT και DEBUG είναι ότι η πρώτη τυπώνει εκφράσεις στην προκαθορισμένη έξοδο, ενώ η δεύτερη στην προκαθορισμένη έξοδο λαθών.

Λίγα λόγια για τους “περιγραφείς αρχείων” (FD, file descriptors).

Στον προγραμματισμό ένας περιγραφέας (descriptor) είναι ένας δείκτης προς ένα αρχείο. Ένα αρχείο σε ένα σύστημα POSIX μπορεί να είναι

- ένα απλό file
- ένα folder
- μια συσκευή π.χ. /dev/sda
- ένα pipe
- ένα socket
- ένα link

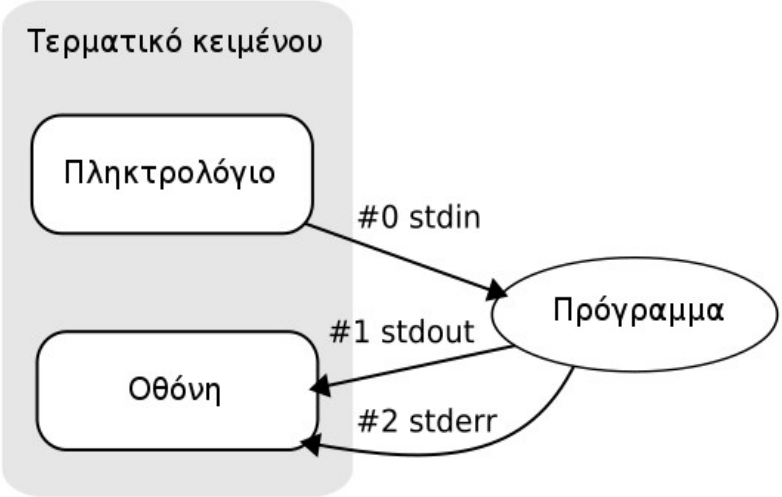
Στα συστήματα POSIX όπως το Linux, υπάρχουν τρεις βασικοί περιγραφείς που κάθε ένας τους ορίζεται με έναν ακέραιο.

Τιμή ακεραίου	**Όνομα**	**Μετάφραση**
0	Standard Input (stdin)	Προκαθορισμένη είσοδος.
1	Standard Output (stdout)	Προκαθορισμένη έξοδος.
2	Standard Error (stderr)	Προκαθορισμένη έξοδος λαθών.

- Η προκαθορισμένη είσοδος λαμβάνει δεδομένα από όλες τις συσκευές που εισάγουν δεδομένα στον η/υ όπως το ποντίκι, το πληκτρολόγιο, το scanner κτλ.
- Η προκαθορισμένη έξοδος είναι συνήθως το τερματικό, όπου γράφουμε εντολές και εμφανίζονται τα αποτελέσματα τους.
- Η προκαθορισμένη έξοδος λαθών, είναι συνήθως πάλι το τερματικό, αποτελεί ξεχωριστό περιγραφέα από την προκαθορισμένη έξοδο και εδώ εμφανίζονται τα λάθη των εφαρμογών εξ' ορισμού.

Όταν ο χρήστης πληκτρολογεί κάτι στο τερματικό προς ένα πρόγραμμα, το πρόγραμμα

διαβάζει την stdin και δέχεται είσοδο προς αυτό.
Το πρόγραμμα εκτελείται και τυπώνει μηνύματα στην stdout.
Αν το πρόγραμμα δημιουργήσει ένα λάθος τότε τυπώνονται μηνύματα λάθους στην stderr.
Συνήθως τόσο η stdout όσο και η stderr τυπώνουν μηνύματα στην οθόνη.



Εικόνα 68: Περιγραφή των προκαθορισμένων εισόδων, εξόδων

Η εντολή PRINT τυπώνει εκφράσεις στην stdout.
Η εντολή DEBUG τυπώνει εκφράσεις στην stderr.

ERROR

Η λέξη ERROR έχει διπλή λειτουργία στην Gambas.

Πρώτη χρήση:
Δηλώνει έναν Boolean όταν εμφανίζεται ένα λάθος.

Σύνταξη:

```
ERROR AS Boolean
```

Επιστρέφει TRUE αν έχει συμβεί ένα λάθος.
Χρησιμοποιείστε την εντολή αυτή μετά το TRY για να ελέγξετε αν η εντολή εκτελέσθηκε με επιτυχία.

Παράδειγμα:

```
1  ' Gambas module file
2  PUBLIC SUB Main()
3
4  DIM FileName AS String
5
6  TRY KILL FileName
7  IF ERROR THEN PRINT "Cannot remove file. "; Error.Text
8
9  END
```

Έξοδος:

```
Cannot remove file. Access forbidden
```

Ανάλυση:
Στην γραμμή 4 ορίζουμε μια μεταβλητή string χωρίς τιμή.
Στην γραμμή 6 προσπαθούμε να διαγράψουμε ένα αρχείο το οποίο θα είχε το path του string.
Επειδή όμως το string είναι χωρίς τιμή εμφανίζεται ένα λάθος.
Στην γραμμή 7 ελέγχουμε μήπως εμφανίστηκε ένα λάθος και τυπώνουμε ένα μήνυμα.

Δεύτερη χρήση:
Τυπώνει μηνύματα του προγραμματιστή στην προκαθορισμένη έξοδο λαθών όπως ακριβώς η εντολή PRINT.

Σύνταξη:

```
ERROR Expression [ { ; | ;; | , } Expression ... ] [ { ; | ;; | , }
]
```

- Αν δεν υπάρχει Ελληνικό ερωτηματικό τότε ένας χαρακτήρας νέας γραμμής τυπώνεται (\n).
- Αν υπάρχει Ελληνικό ερωτηματικό η έκφραση που ακολουθεί τυπώνεται αμέσως μετά.
- Αν υπάρχει διπλό Ελληνικό ερωτηματικό τότε μεταξύ των εκφράσεων τυπώνεται το κενό διάστημα.
- Αν υπάρχει ένα κόμμα ανάμεσα στις εκφράσεις τότε τυπώνεται ένας χαρακτήρας TAB.

Η διαφορά της ERROR με την DEBUG είναι ότι με την ERROR ο προγραμματιστής τυπώνει δικά του μηνύματα λάθους στην stderr.

Στο προηγούμενο παράδειγμα κανονικά δεν έπρεπε να τυπώσουμε το μήνυμα λάθους με την PRINT.
Το σωστό θα ήταν να χρησιμοποιήσουμε την ERROR για να τυπωθούν τα μηνύματα στην

stderr.

Παράδειγμα:

```
1  ' Gambas module file
2  PUBLIC SUB Main()
3
4  DIM FileName AS String
5
6  TRY KILL FileName
7  'IF ERROR THEN PRINT "Cannot remove file. "; Error.Text
8  IF ERROR THEN ERROR "Cannot remove file. "; Error.Text
9
10 END
```

Έξοδος:

```
Cannot remove file. Access forbidden
```

FINALLY

Σε αυτό το μπλοκ κώδικα τοποθετούνται προτάσεις που θα εκτελεστούν ακόμη και μετά την εμφάνιση ενός λάθους. Το μπλοκ κώδικα FINALLY θα πρέπει να τοποθετηθεί πριν το μπλοκ κώδικα της CATCH.

Σύνταξη:

```
SUB Function(...)
...
FINALLY
...
END
```

Όταν δημιουργηθεί ένα λάθος εκτελείται ο κώδικας του μπλοκ FINALLY χωρίς να εμφανιστεί το λάθος. Αυτή η λειτουργία φαίνεται στο Project24.

Παράδειγμα:

```
1   ' Gambas module file
2   PUBLIC SUB Main()
3
4   DIM FileName AS String
5
6   TRY KILL FileName
7
8   FINALLY
9     PRINT "I am in FINALLY block"
10
11  CATCH
12    ERROR Error.Text
13
14  END
```

Έξοδος:

```
I am in FINALLY block
```

Ανάλυση:
Το προηγούμενο παράδειγμα με ένα μπλοκ FINALLY.
Η γραμμή 12 της CATCH δεν εκτελείται διότι αμέσως μετά την εμφάνιση λάθους εκτελείται το μπλοκ κώδικα της FINALLY.

TRY

Η εντολή προσπαθεί να εκτελέσει μια πρόταση χωρίς να εμφανίσει ένα λάθος.

Σύνταξη:

```
TRY Statement
```

- TRY, απαραίτητη λέξη της εντολής TRY.
- Statement, η πρόταση που θα εκτελεστεί και δεν θα εμφανίσει το λάθος.

Χρησιμοποιείστε την εντολή ERROR για να ελέγξετε αν ένα λάθος εμφανίζεται μετά την εκτέλεση μιας εντολής TRY.

Δείτε για περισσότερα την εντολή ERROR.

Εντολές QUIT, END, RETURN, STOP, BREAK, SLEEP, WAIT

Υπάρχουν κάποιες εντολές της Gambas 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 **QUIT, END, RETURN, STOP, BREAK, SLEEP, WAIT**

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

QUIT

Η εντολή QUIT τερματίζει την εφαρμογή αμέσως. Όλα τα παράθυρα κλείνουν και η μνήμη αποδεσμεύεται αμέσως.

Για να δείτε ένα παράδειγμα δημιουργείστε το Project25 με μία φόρμα και ένα κουμπί.
Στο συμβάν κλικ του κουμπιού γράψτε τον εξής κώδικα.

Παράδειγμα:

```
1   ' Gambas class file
2
3
4   PUBLIC SUB Button1_Click()
5
6     DIM i, x AS Integer = 0
7
8     x = 10 / i
9
10    CATCH
11    DEBUG Error.Text
12    QUIT
13
14  END
```

Έξοδος:

```
FMain.Button1_Click.11: Division by zero
```

Ανάλυση:
Στην γραμμή 6 δηλώνουμε δύο μεταβλητές ακεραίων και τις εκχωρούμε την τιμή 0.
Στην γραμμή 8 διαιρούμε έναν αριθμό με το 0 το οποίο εμφανίζει ένα λάθος.
Η γραμμή 10 ξεκινά την διαχείριση λαθών.
Στην γραμμή 11 εμφανίζουμε το λάθος στην stderr.
Στην γραμμή 12 δηλώνουμε τον τερματισμό του προγράμματος.

END

Η εντολή δηλώνει το τέλος μιας ρουτίνας, υπορουτίνας ή συνάρτησης.
Είναι η τελευταία λέξη του μπλοκ που αρχίζει με SUB, PROCEDURE ή FUNCTION.

Παράδειγμα:

```
1  PUBLIC SUB Button1_Click()
2  ...
3  END
```

RETURN

Εγκαταλείπει μια συνάρτηση επιστρέφοντας μια τιμή.

Σύνταξη:

```
RETURN [ Expression ]
```

RETURN, απαραίτητη λέξη δήλωσης της εντολής επιστροφής μιας τιμής από μια συνάρτηση.
Expression, απαραίτητη έκφραση που αποδίδει μια τιμή.

Για περισσότερα δείτε στο Κεφάλαιο για τις συναρτήσεις.

STOP

Σταματά την εκτέλεση του προγράμματος και εμφανίζει τον αποσφαλματωτή (debuger) όπως ακριβώς σαν να είχε θέσει ο προγραμματιστής ένα σημείο διακοπής (breakpoint) στον επεξεργαστή κειμένου του κώδικα.

Σύνταξη:

```
STOP
```

Παράδειγμα:

```
1  PUBLIC SUB Button1_Click()
2    ' Stop if the following condition is true
3    IF Name = "Nikos" THEN STOP
4  END
```

BREAK

Εγκαταλείπει έναν βρόχο αμέσως.

Σύνταξη:

```
BREAK
```

Παράδειγμα:

```
PUBLIC SUB Button1_Click()

  DIM X AS Integer

  FOR X = 1 TO 1000
    IF X = 200 THEN BREAK
    PRINT X
  NEXT

END
```

SLEEP

Σταματάει την εκτέλεση του προγράμματος ορίζοντας μια χρονοκαθυστέρηση σε δευτερόλεπτα.

Σύνταξη:

```
SLEEP Delay
```

Η τιμή της χρονοκαθυστέρησης μπορεί να είναι ένας float. Έτσι αν θέλετε να ορίσετε microsecond μπορείτε να ορίσετε την τιμή 0.1 δλδ 100 ms.

Κατά την διάρκεια της χρονοκαθυστέρησης δεν καλείται ο βρόχος ελέγχου των συμβάντων και έτσι τα συμβάντα δεν θα ανταποκρίνονται στην εφαρμογή.

WAIT

Εκτελεί συνεχώς τον βρόχο των συμβάντων.
Η συνάρτηση δεν εκτελεί την ροή του προγράμματος στις επόμενες προτάσεις για την δοθείσα χρονοκαθυστέρηση.

Σύνταξη:

```
WAIT Delay
```

Η τιμή της χρονοκαθυστέρησης μπορεί να είναι ένας float. Έτσι αν θέλετε να ορίσετε microsecond μπορείτε να ορίσετε την τιμή 0.1 δλδ 100 ms.

Αν δεν δοθεί τιμή χρονοκαθυστέρησης οι συναρτήσεις επεξεργάζονται κάθε συμβάν και επιστρέφουν αμέσως. Σε αυτήν την περίπτωση τα συμβάντα του ποντικιού και του πληκτρολογίου αγνοούνται.

Αν η εντολή WAIT γραφτεί σε κάποια ρουτίνα ελέγχου συμβάντος συνεχώς θα καλεί τον βρόχο συμβάντων και επομένως την ρουτίνα που θα την περιέχει. Αυτό θα υπερχειλίσει την στοίβα της μνήμης.
Αν θέλετε να ορίσετε μια χρονοκαθυστέρηση σε μια ρουτίνα ελέγχου συμβάντος χρησιμοποιείστε την SLEEP.
Για παράδειγμα αν έχετε μια ρουτίνα ελέγχου του συμβάντος READ μιας σειριακής θύρας ο παρακάτω κώδικας θα υπερχειλίσει την στοίβα διότι η ρουτίνα READ είναι ρουτίνα που καλείται από τον βρόχο συμβάντων και η WAIT θα την καλεί συνεχώς και μαζί με αυτήν και τον εαυτό της κάθε φορά.

Παράδειγμα:

```
1   PUBLIC SUB MySerialPort_Read()
2
3     DIM sData, sTemp AS String
4     DIM iTries AS Integer
5
6     FOR iTries = 1 TO 5
7       ' wait a bit and read response.
8       WAIT 0.01
9       ' see if we got some data.
10      READ #LAST, sTemp, Lof(LAST)
11      sData &= sTemp
12      ...
13   NEXT
14
15  END
```

Λίγα λόγια για τον βρόχο συμβάντων (EVENT LOOP)

Όταν ένα πρόγραμμα της Gambas έχει τελειώσει την εκτέλεση της ρουτίνας εκκίνησης Main(), εισέρχεται στον λεγόμενο βρόχο συμβάντων.

Το πρόγραμμα θα τρέχει σε αυτόν τον βρόχο όσο:

- Υπάρχει μια διεργασία Process η οποία τρέχει.
- Υπάρχει ένας Timer ο οποίος είναι ενεργοποιημένος.
- Υπάρχει ένα αρχείο File το οποίο παρακολουθείται.
- Υπάρχει τουλάχιστον ένα παράθυρο υψηλού επιπέδου (toplevel Window) το οποίο έχει δημιουργηθεί ακόμη και αν δεν είναι ορατό.

Κατά τον βρόχο αυτό, κάθε συμβάν που εγείρεται από οποιοδήποτε αντικείμενο καλεί τον διαχειριστή συμβάντος (event handler).

Μπορείτε να καλέσετε τον βρόχο συμβάντων με την εντολή WAIT.

Όταν χρησιμοποιείτε την WAIT το γραφικό περιβάλλον ανανεώνεται και επανασχεδιάζεται, τα αντικείμενα μπορούν να λάβουν τιμές στις ιδιότητες τους κτλ. Αυτό είναι χρήσιμο όταν εκτελείται μεγάλες και χρονοβόρες διεργασίες και θέλετε να παρακολουθείτε ταυτόχρονα το αποτέλεσμα.

Κεφάλαιο 8ο – Διατάξεις ή Πίνακες

Μέχρι τώρα αναφέραμε τον τρόπο αποθήκευσης δεδομένων σε μεταβλητές.
Κάθε μεταβλητή είναι μια θέση μνήμης τέτοιου μεγέθους όσο έχει οριστεί από τον τύπο δεδομένων της.
Για παράδειγμα ένας ακέραιος (integer) καταλαμβάνει 4 bytes, μια μεταβλητή Byte καταλαμβάνει 1 byte κτλ.

Υπάρχουν περιπτώσεις που θέλουμε να ορίσουμε μια ομάδα μεταβλητών ίδιου τύπου και ίδιου ονόματος.
Αυτή η ομάδα μεταβλητών είναι μια συλλογή δεδομένων ίδιων τύπων.
Για να αποθηκευτεί μια τέτοια ομάδα δεδομένων στην μνήμη υπάρχουν οι “διατάξεις” (Arrays) ή αλλιώς όπως λέγονται “πίνακες”.

Μια διάταξη είναι ένα σύνολο θέσεων αποθήκευσης δεδομένων που είναι του ίδιου τύπου.
Μια απλή διάταξη στην Gambas εμφανίζεται με ένα όνομα, όπως μια μεταβλητή, και στην συνέχεια ακολουθούν αγκύλες που περιέχουν τον αριθμό των θέσεων αποθήκευσης.

Για παράδειγμα μια διάταξη που κρατά ονόματα είναι μια συλλογή strings και εμφανίζεται ως εξής.

```
Names[10]
```

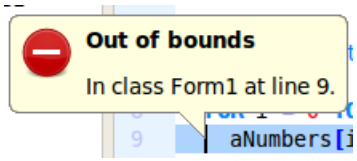
Το Names, είναι το όνομα της διάταξης, το 10 είναι οι θέσεις της διάταξης.

Κάθε θέση μιας διάταξης ονομάζεται “στοιχείο”, (element).
Ο αριθμός που δείχνει την θέση μιας τιμής ονομάζεται “δείκτης”, (index) ή κλειδί (key).

Στο παράδειγμα Names[10], το Names[1], Names[2], Names[3] κτλ είναι στοιχεία της διάταξης.
Το 1, 2, 3 κτλ είναι η τιμή του δείκτη που αλλάζει για να έχουμε αναφορά σε διαφορετική θέση μνήμης.
Ο δείκτης σε μία διάταξη περιβάλλεται από αγκύλες.

Το πρώτο στοιχείο μιας διάταξης έχει δείκτη το 0.
Έτσι στην διάταξη Names[10], το πρώτο στοιχείο είναι το Names[0], το δεύτερο το Names[1] το τελευταίο στοιχείο θα είναι το Names[9].
Συνολικά η διάταξη θα έχει 10 στοιχεία τα 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Αυτός ο τρόπος αρίθμησης πολλές φορές στην αρχή μπερδεύει και μπορεί να οδηγήσει σε λάθη τα οποία συνήθως είναι να μετράμε εκτός ορίων της διάταξης.
Σε αυτήν την περίπτωση θα εμφανιστεί ένα μήνυμα λάθους σαν το παρακάτω.



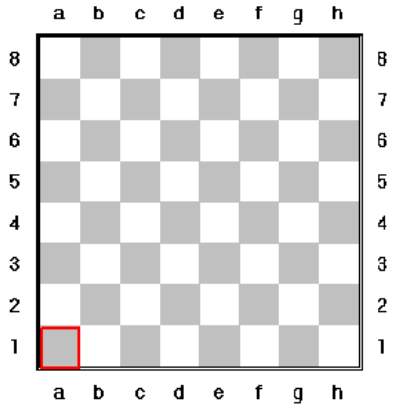
Εικόνα 69: Μήνυμα εκτός ορίων για προσπέλαση στοιχείων διάταξης

Το σύνολο των δεικτών αποτελεί ένα ευρετήριο.
Οι διατάξεις μπορούν να έχουν ταυτόχρονα πολλά ευρετήρια δεικτών.
Όταν μία διάταξη έχει πάνω από ένα ευρετήριο δεικτών έχει τότε πολλές διαστάσεις.
Κάθε ευρετήριο δεικτών είναι μία διάσταση της διάταξης.

Για παράδειγμα μία διάταξη με δύο διαστάσεις ορίζεται ως

```
Chess[8,8]
```

Η παραπάνω πρόταση σημαίνει ότι η διάταξη Chess έχει μία διάσταση 8 θέσεων και άλλη μία άλλων 8 θέσεων.
Κάθε συνδυασμός των δεικτών των διαστάσεων αντιστοιχεί σε μία θέση δεδομένων.
Στο παράδειγμα αυτό, η πρώτη διάσταση των 8 θέσεων αντιστοιχεί στις κάθετες γραμμές μιας σκακιέρας, δλδ στα a, b, c ...
Η δεύτερη διάσταση των 8 θέσεων αντιστοιχεί στις οριζόντιες γραμμές μιας σκακιέρας, δλδ στα 1, 2, 3...
Ο συνδυασμός για παράδειγμα Chess[0,0] αντιστοιχεί στην στην θέση a1.



Στην Gambas το πλήθος των διαστάσεων μιας διάταξης είναι μέχρι 8.
Δλδ μπορεί να δηλωθεί διάταξη του τύπου Array[3,3,1,5,6,4,5,9]
Οι διατάξεις συχνά ονομάζονται και απλά ως “πίνακες”.

Δήλωση διατάξεων

Μια διάταξη μπορεί να είναι όπως και μια μεταβλητή καθολικής εμβέλειας αλλά όχι τοπικής. Μια διάταξη μπορεί να έχει καθορισμένο και αμετάβλητο αριθμό στοιχείων, δλδ να είναι μια μη δυναμική διάταξη ή ο αριθμός των στοιχείων της να αυξάνει και να μειώνεται, δλδ να είναι μια δυναμική διάταξη.

Καθολικές μη δυναμικές διατάξεις

Μια διάταξη καθολικής εμβέλειας δηλώνεται ως εξής

Σύνταξη:

```
[ STATIC ] { PUBLIC | PRIVATE } Identifier [ Array dimensions ... ]
AS Native Datatype
```

Όταν μια διάταξη οριστεί ως **STATIC** η διάταξη είναι προσπελάσιμη για κάθε αντικείμενο της κλάσης.

Όταν μια διάταξη οριστεί ως **PUBLIC** είναι προσπελάσιμη και εκτός από την κλάση που ορίζεται.
Όταν μια διάταξη οριστεί ως **PRIVATE** είναι προσπελάσιμη μόνο από κλάση που ορίζεται.

Ως Identifier είναι το όνομα που δίνει ο προγραμματιστής στην διάταξη.
Εντός των αγκυλών ορίζονται αριθμητικά με ακεραίους οι διαστάσεις της διάταξης.

Ο προγραμματιστής αφού ορίσει το όνομα της διάταξης που επιθυμεί θα πρέπει να ορίσει τον τύπου δεδομένων Datatype της διάταξης. Αυτό γίνεται με την λέξη **AS** και ακολουθεί ο τύπος δεδομένων.
Στην δήλωση της μη δυναμικής καθολικής διάταξης δεν μπορούμε να ορίσουμε αρχικές τιμές.

Παράδειγμα:

```
1 STATIC aNumbers[10] AS Integer
2 PRIVATE aNumbers[10] AS NEW Integer
3 PRIVATE aChess[8,8] AS Integer
```

Η εκχώρηση τιμών είναι η ίδια για κάθε μη δυναμικό πίνακα.

Παράδειγμα:

```
1  aNumbers[0]=31
2  aNumbers[1]=24
3  aChess[0,0]=11
4  aChess[1,0]=21
```

Παράδειγμα:
Δημιουργείστε το Project26. Το έργο αυτό έχει μια μονάδα (module) με μια συνάρτηση Main().
Δηλώνουμε μια καθολική στατική διάταξη 10 στοιχείων ακεραίων την aNumbers[10].
Και στην συνέχεια εκχωρούμε τιμές σε κάθε στοιχείο και τυπώνουμε την τιμή κάθε στοιχείου στην stdout.

Ο κώδικας της μονάδος (Module) είναι ο εξής:

```
1   ' Gambas module file
2   Private aNumbers[10] As Integer
3
4   Public Sub Main()
5
6     Dim i As Integer
7
8     For i = 0 To 9
9       aNumbers[i] = i
10      Print aNumbers[i];;
11    Next
12
13  End
```

Έξοδος:

```
0 1 2 3 4 5 6 7 8 9
```

Ανάλυση:
Στην γραμμή 2 δηλώνουμε έναν στατικό πίνακα ακεραίων με 10 θέσεις.
Στην γραμμή 4 ξεκινάει η συνάρτηση Main().
Στην γραμμή 6 δηλώνουμε μια μεταβλητή ακεραίου.

Στην γραμμή 8 ξεκινάει ένας βρόχος που μετράει από το 0 μέχρι το 9.
Σε κάθε επανάληψη εκχωρούμε μια τιμή σε ένα στοιχείο του πίνακα, στην γραμμή 9 και τυπώνουμε την τιμή που έχει ένα στοιχείο στην stdout, στην γραμμή 10.

Δυναμικές διατάξεις

Μια διάταξη μέχρι τώρα είδαμε ότι έχει προκαθορισμένο αριθμό στοιχείων.
Αυτό όμως δεν είναι απαραίτητο. Μια διάταξη μπορεί να έχει δυναμικό αριθμό στοιχείων (dynamic array).
Οι δυναμικές διατάξεις δεν έχουν προκαθορισμένο αριθμό στοιχείων και μπορούν να αυξάνουν ή να μειώνουν τον αριθμό των στοιχείων τους. Οι δυναμικές διατάξεις είναι πραγματικά αντικείμενα στην Gambas, ενώ οι μη δυναμικές διατάξεις δεν είναι αντικείμενα.

Οι δυναμικές διατάξεις μπορεί να έχουν τοπική ή καθολική εμβέλεια.

Δήλωση τοπικών δυναμικών διατάξεων

Σύνταξη:

```
DIM Identifier AS [ NEW ] Native Datatype [Array dimensions ...
][ = [Expression] ]
```

- DIM, απαραίτητη λέξη στην δήλωση διατάξεων τοπικής εμβέλειας.
- Identifier, το όνομα της διάταξης
- AS, απαραίτητη λέξη που δηλώνει τον τύπο δεδομένων της διάταξης
- NEW, προαιρετική λέξη που ορίζει ένα νέο αντικείμενο
- Native Datatype, ο εγγενής τύπος δεδομένων της διάταξης
- [Array dimensions], οι διαστάσεις της διάταξης. Αν δεν οριστούν τότε προκαθορίζεται μία διάσταση, αν οριστούν με ακεραίους που διαχωρίζονται με κόμμα μπορεί να είναι μέχρι οκτώ στο πλήθος.
- = [Expression], μια λίστα εκφράσεων που διαχωρίζεται με κόμμα και περικλείεται από τον τελεστή διάταξης []. Ο τελεστής διάταξης δημιουργεί μια προσωρινή διάταξη στην μνήμη που εκχωρείται σε μία δυναμική διάταξη με τον τελεστή εκχώρησης = ώστε έτσι να οριστούν αρχικές τιμές στην διάταξη.

Παράδειγμα:

```
1 DIM aNumbers AS Integer[]
2 DIM aNumbers AS NEW Integer[]
3 DIM aNames AS String[1, 1, 1]
4 DIM aChess AS NEW Integer[8, 8]
```

Δήλωση καθολικών δυναμικών διατάξεων

Σύνταξη:

```
[ STATIC ] { PUBLIC | PRIVATE } Identifier AS [ NEW ] Native
Datatype [Array dimensions ... ][ = [Expression] ]
```

- STATIC, η διάταξη είναι προσπελάσιμη για κάθε αντικείμενο της κλάσης.
- PUBLIC, η διάταξη είναι προσπελάσιμη και από άλλες κλάσεις.
- PRIVATE, η διάταξη είναι προσπελάσιμη μόνο από κλάση που ορίζεται.
- Identifier, το όνομα της διάταξης
- AS, απαραίτητη λέξη που δηλώνει τον τύπο δεδομένων της διάταξης
- NEW, προαιρετική λέξη που ορίζει ένα νέο αντικείμενο
- Native Datatype, ο εγγενής τύπος δεδομένων της διάταξης
- Array dimensions, οι διαστάσεις της διάταξης. Αν δεν οριστούν τότε προκαθορίζεται μία διάσταση, αν οριστούν με ακεραίους που διαχωρίζονται με κόμμα μπορεί να είναι μέχρι οκτώ στο πλήθος.
- = [Expression], μια λίστα εκφράσεων που διαχωρίζεται με κόμμα και περικλείεται από τον τελεστή διάταξης []. Ο τελεστής διάταξης δημιουργεί μια προσωρινή διάταξη στην μνήμη που εκχωρείται σε μία δυναμική διάταξη με τον τελεστή εκχώρησης = ώστε έτσι να οριστούν αρχικές τιμές στην διάταξη.

Παράδειγμα:

```
STATIC PUBLIC aNumbers AS Integer[]
STATIC PRIVATE aNumbers AS Integer[]
PRIVATE aNumbers AS NEW Integer[]
PUBLIC aNames AS String[]
PRIVATE aChess AS Integer[8, 8]
```

Αρχικοποίηση δυναμικών πινάκων

Στις δυναμικές διατάξεις μπορούμε να ορίσουμε και αρχικές τιμές κατά την δήλωση τους.

Παράδειγμα:

```
PUBLIC aNames AS String[] = ["Nikos", "Petros", "Maria"]
DIM aNumbers AS Integer[] = [0, 1, 4, 5]
DIM aNames AS String[] = ["Nikos", "Petros", "Maria"]
```

Οι τιμές που εκχωρούνται τοποθετούνται ανάμεσα σε αγκύλες και διαχωρίζονται από κόμμα. Όταν εκχωρούμε αρχικές τιμές στην δήλωση δυναμικών διατάξεων δεν χρησιμοποιούμε ταυτόχρονα την λέξη NEW.

Παράδειγμα:

```
1  DIM aNumbers AS NEW Integer[]
2  'ή η μία δήλωση ή η άλλη
3  DIM aNumbers AS Integer[] = [0, 1, 4, 5]
```

Αν θέλαμε να γράψουμε το αντίστοιχο παράδειγμα του Project26 αλλά με δυναμικό πίνακα θα είχαμε ένα νέο αντίστοιχο Project27 με τον ακόλουθο κώδικα.

Παράδειγμα:

```
1   ' Gambas module file
2
3   Public Sub Main()
4
5     Dim aNumbers As New Integer[]
6     Dim i As Integer
7
8     For i = 0 To 9
9       aNumbers.Add(i)
10      Print aNumbers[i];;
11    Next
12
13  End
```

Έξοδος:

```
0 1 2 3 4 5 6 7 8 9
```

Ανάλυση:
Στην γραμμή 5 δηλώνεται ένας δυναμικός πίνακας ακεραίων.
Για να προσθέσουμε τιμές στα στοιχεία του πίνακα είναι απαραίτητο να τον δηλώσουμε με την λέξη NEW.
Έτσι δεν μπορούμε να χρησιμοποιήσουμε την έκφραση aNumbers[i] = i.
Θα πρέπει να προσθέσουμε στοιχεία με την μέθοδο που δίνουν οι πίνακες την Add(value).
Έτσι στον βρόχο FOR γράφουμε
aNumbers.Add(i).

Αν δεν χρησιμοποιούσαμε την λέξη NEW στην δήλωση του πίνακα αλλά απλά τον δηλώναμε με την πρόταση DIM aNumbers AS Integer[] τότε η Gambas θα εμφάνιζε ένα λάθος μεταγλώττισης ότι το αντικείμενο είναι κενό (Null).
Για να συνεχίσουμε την συγγραφή του κώδικα θα έπρεπε να εκχωρήσουμε αρχική τιμή στον πίνακα και μετά να το καθαρίσουμε από τιμές.
Θα έπρεπε να γράψουμε κάτι σαν και αυτό.

Παράδειγμα:

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5    DIM aNumbers AS Integer[] = [0]
6    DIM i AS Integer
7
8    aNumbers.Clear
9
10   FOR i = 0 TO 9
11     aNumbers.Add(i)
12     PRINT aNumbers[i]
13   NEXT
14
15 END
```

Αυτός όμως ο αλγόριθμος αν και δουλεύει δεν είναι επαγγελματικός αλλά πρόχειρος.
Για να αποφύγουμε τέτοια “κόλπα” θα πρέπει να δηλώνουμε τους δυναμικούς πίνακες με την λέξη NEW για άμεση χρήση τους.

Θα πρέπει να αρχικοποιείται ένας δυναμικός πίνακας όταν μόνο πρέπει ειδάλλως αν στην συνέχεια του εκχωρούμε τιμές καλό είναι να χρησιμοποιούμε την λέξη NEW.

Ο βρόχος FOR EACH ... NEXT

Οι πίνακες μπορούν να προσπελαστούν και να απαριθμηθούν με τον βρόχο FOR EACH...NEXT.

Σύνταξη:

```
FOR EACH Variable IN Expression
...
NEXT

FOR EACH Expression
...
NEXT
```

- FOR EACH, απαραίτητη πρόταση που ξεκινά τον βρόχο
- Variable, μια μεταβλητή που θα δέχεται τις τιμές κάθε στοιχείου του πίνακα
- IN, λέξη που δηλώνει ότι αυτό που ακολουθεί είναι ένας δοχέας
- Expression, έκφραση που επιστρέφει έναν πίνακα ή μια συλλογή.

Η έκφραση μπορεί να είναι:

- Ένας πίνακας
- Μια συλλογή
- Ένα απαριθμούμενο αντικείμενο που δεν είναι πραγματικός δοχέας, όπως για παράδειγμα το αποτέλεσμα ενός query μιας βάσης δεδομένων. Στην περίπτωση αυτή δεν χρησιμοποιούμε την λέξη IN.

Παράδειγμα:

```
1   ' Gambas module file
2
3   PUBLIC SUB Main()
4
5     DIM aNumbers AS NEW Integer[]
6     DIM i AS Integer
7     DIM a AS Integer
8
9     FOR i = 0 TO 9
10      aNumbers.Add(i)
11      PRINT aNumbers[i];;
12    NEXT
13
14    PRINT "\nNow we will enumerate the array"
15    FOR EACH a IN aNumbers
16      PRINT a;;
17    NEXT
18
19  END
```

Ανάλυση:
Το παράδειγμα αυτό είναι η συνέχεια του Project27.
Στην γραμμή 7 δηλώνουμε ακόμη μια μεταβλητή ακεραίου την a.
Αφού εκχωρήσουμε τιμές στον πίνακα με τον βρόχο FOR...NEXT στις γραμμές 9 με 11, στην γραμμή 14 εμφανίζουμε ένα μήνυμα ότι θα απαριθμήσουμε τον πίνακα με την FOR EACH.
Στην γραμμή 15 ξεκινάει ο βρόχος FOR EACH.
Κάθε στοιχείο του πίνακα εκχωρεί την τιμή του στην μεταβλητή a.
Στην γραμμή 16 τυπώνουμε την τιμή της a.
Το αποτέλεσμα είναι να έχουμε πάλι μια δεύτερη απαρίθμηση από το 0 έως το 9. Μόνο που σε αυτήν την δεύτερη φορά έχει γίνει με τον βρόχο FOR EACH.

Έξοδος:

```
0 1 2 3 4 5 6 7 8 9
Now we will enumerate the array
0 1 2 3 4 5 6 7 8 9
```

Στις περιπτώσεις που η απαριθμούμενη έκφραση δεν είναι ένας πίνακας αλλά μια λίστα τιμών όπως μια λίστα τιμών που επιστρέφει ένα ερώτημα σε μια βάση δεδομένων δεν χρησιμοποιούμε την λέξη IN, όπως φαίνεται στο παράδειγμα.

Παράδειγμα:

```
1 DIM Res AS Result
2
3 Res = DB.Exec("SELECT * FROM MyTable")
4
5 FOR EACH Res
6   PRINT Res!Code; " "; Res!Name
7 NEXT
```

Ανάλυση:
Στην γραμμή 1 δηλώνουμε μια μεταβλητή Res ως αποτέλεσμα ενός ερωτήματος σε βάση δεδομένων.
Στην γραμμή 3 εκχωρούμε στη Res την επιστροφή από το ερώτημα στην βάση. Η Res δεν είναι μια πραγματική διάταξη που έχει οριστεί από τον προγραμματιστή. Η Res είναι μια κλάση της Gambas που συμπεριφέρεται σαν μια διάταξη και μπορεί να απαριθμηθεί.
Στην γραμμή 5 ξεκινά ο βρόχος FOR EACH για την απαριθμούμενη Res. Σημειώστε ότι σε αυτήν την περίπτωση η σύνταξη του βρόχου FOR EACH είναι χωρίς την λέξη IN.
Στην γραμμή 6 τυπώνεται το πεδίο Code και Name του αποτελέσματος Res. Ένα αποτέλεσμα βάσης δεδομένων προσπελάζεται με ! Αντί για . που χρησιμοποιούν τα αντικείμενα. Για περισσότερα δείτε στο κεφάλαιο για τις βάσεις δεδομένων.

Δυναμικές διατάξεις ως αντικείμενα

Η Gambas έχει τις εξής δυναμικές διατάξεις, οι οποίες είναι ταυτόχρονα και αντικείμενα:
Boolean[]
Date[]
Float[]
Integer[]
Long[]
Object[]
Short[]
Pointer[]
Single[]
String[]
Variant[]

Οι κλάσεις αυτές κληρονομούν την διάταξη (Array).
Οι κλάσεις αυτές μπορούν να δημιουργηθούν, συμπεριφέρονται σαν μία διάταξη που μπορεί να διαβαστεί και να γραφτεί και είναι απαριθμήσιμες με τον βρόχο FOR ... EACH.

Κάθε κλάση έχει ιδιότητες (properties) και μεθόδους (methods).
Για να τις μελετήσουμε μπορούμε να τις χωρίσουμε σε δύο ομάδες.

1η ομάδα.
Οι κλάσεις Boolean[], Date[], Float[], Integer[], Long[], Short[], Pointer[], Single[] έχουν τις ίδιες ιδιότητες και μεθόδους.
Η κλάση String[] έχει τις ίδιες ιδιότητες και μεθόδους με τις προηγούμενες εκτός από την μέθοδο Write().

2η ομάδα.
Οι κλάσεις Variant[], Object[] έχουν κάποιες από τις ιδιότητες και μεθόδους των προηγούμενων αντικειμένων.

Αλλά ας δούμε αναλυτικά κάθε ομάδα αντικειμένων.

Οι κλάσεις Boolean[], Date[], Float[], Integer[], Long[], Short[], Pointer[], Single[].

Ιδιότητες:
Οι κλάσεις αυτές έχουν τις εξής ιδιότητες:

Ιδιότητα	**Περιγραφή**
Bounds	Επιστρέφει μια εικονική κλάση διάταξης που αντιπροσωπεύει τα όρια από κάθε διάσταση της διάταξης.
Count	Επιστρέφει έναν integer που δείχνει τον αριθμό των στοιχείων της διάταξης
Data	Επιστρέφει έναν δείκτη (pointer) για τα δεδομένα του πίνακα
Dim	Επιστρέφει τον αριθμό των διαστάσεων της διάταξης
Length	Επιστρέφει τον αριθμό των στοιχείων της διάταξης
Max	Επιστρέφει την μεγαλύτερη τιμή από τα κλειδιά της διάταξης. Είναι με άλλα λόγια ο αριθμός των στοιχείων της διάταξης μείον 1. (Array.Max = Array.Count - 1)
Type	Επιστρέφει τον τύπο δεδομένων που μπορούν να εκχωρηθούν στην διάταξη.

Μέθοδοι:
Οι κλάσεις αυτές έχουν τις εξής μεθόδους:

Μέθοδοι της κλάσεως Array
Clear Remove Resize Reverse
Μέθοδοι του αντικειμένου
Add Copy Delete Exist Extract Fill Find Insert Pop Push Read Sort Write

Clear
Σύνταξη:

```
SUB Clear ( )
```

Καθαρίζει την διάταξη με το να αφαιρεί τα στοιχεία της.
Μετά τον καθαρισμό η διάταξη έχει μέγεθος 0.

Remove
Σύνταξη:

```
SUB Remove ( Index AS Integer [ , Length AS Integer ] )
```

Αφαιρεί ένα ή περισσότερα στοιχεία από τον πίνακα.
Αφαιρεί το στοιχείο στην θέση Index.
Αν οριστεί και το Length αφαιρεί και τα επόμενα στοιχεία σε πλήθος όσα ορίζει ο ακέραιος Length.

Resize
Σύνταξη:

```
SUB Resize ( Size AS Integer )
```

Αλλάζει το μέγεθος της διάστασης ενός μονοδιάστατου πίνακα.
Αν η διάσταση μειωθεί στοιχεία αφαιρούνται από τον πίνακα.
Αν η διάσταση αυξηθεί στοιχεία προστίθονται από στον πίνακα με τις αρχικές τους τιμές.

Αν ο πίνακας είναι πολυδιάστατος δεν μπορεί να αλλάξει μέγεθος.

Reverse
Σύνταξη:

```
SUB Reverse ( )
```

Αντιστρέφει την σειρά των στοιχείων ενός πίνακα.

Add
Σύνταξη:

```
SUB Add ( Value AS Float [ , Index AS Integer ] )
SUB Add ( Value AS Integer [ , Index AS Integer ] )
SUB Add ( Value AS Long [ , Index AS Integer ] )
...
```

Προσθέτει ένα στοιχείο στην διάταξη στην θέση που ορίζεται από το Index.

Value, η τιμή που θα προστεθεί ως νέο στοιχείο. Η τιμή αυτή θα πρέπει να είναι του ίδιου τύπου δεδομένων με τον τύπο δεδομένων που έχει δηλωθεί η διάταξη.
Index, ένας integer αριθμός που δηλώνει την θέση που θα προστεθεί η νέα τιμή.

Τα στοιχεία που αποθηκεύονται μετά από αυτήν την τιμή μετατοπίζονται από αυτήν την θέση και μετά.
Αν δεν οριστεί θέση index, το νέο στοιχείο τοποθετείται στο τέλος της διάταξης.

Παράδειγμα:

```
1   ' Gambas module file
2
3   PUBLIC SUB Main()
4
5     DIM aBool AS NEW Boolean[]
6
7     aBool.Add(TRUE)
8     aBool.Add(FALSE)
9     aBool.Add(TRUE)
10    aBool.Add(TRUE)
11
12    aBool.Add(TRUE, 1)
13
14    PRINT aBool[0]
15    PRINT aBool[1]
16    PRINT aBool[2]
17    PRINT aBool[3]
18
19  END
```

Έξοδος:

```
True
True
False
True
```

Ανάλυση:
Στην γραμμή 5 δηλώνουμε μια δυναμική διάταξη Boolean[].
Στις γραμμές 7-11 προσθέτουμε τιμές.
Στην γραμμή 12 προσθέτουμε μια τιμή TRUE στην θέση 1. Η θέση 1 ήδη έχει την τιμή FALSE, όμως τώρα θα της εκχωρηθεί η τιμή TRUE και τα στοιχεία θα μετακινηθούν προς τα κάτω από την θέση αυτή.

Έτσι όταν τυπώσουμε στις γραμμές 14-17 τις τιμές βλέπουμε ότι η θέση 1 έχει την τιμή TRUE.

Copy
Σύνταξη:

```
FUNCTION Copy ( [ Start AS Integer, Length AS Integer ] ) AS Boolean[]
FUNCTION Copy ( [ Start AS Integer, Length AS Integer ] ) AS Date[]
FUNCTION Copy ( [ Start AS Integer, Length AS Integer ] ) AS Float[]
...
```

Επιστρέφει ένα αντίγραφο της διάταξης από την θέση ενός στοιχείου μέχρι το μήκος που θα οριστεί.

Start, ένας ακέραιος που δηλώνει το πρώτο στοιχείο από το οποίο θα ξεκινήσει η αντιγραφή.
Length, ένας ακέραιος που δηλώνει πόσα στοιχεία θα αντιγραφούν από το πρώτο στοιχείο.

Αν δεν οριστούν Start, Length, τότε ολόκληρη η διάταξη αντιγράφεται.

Delete
Συνώνυμο με το Extract().

Extract
Σύνταξη:

```
FUNCTION Extract ( Start AS Integer [ , Length AS Integer ] ) AS Boolean[]
FUNCTION Extract ( Start AS Integer [ , Length AS Integer ] ) AS Date[]
FUNCTION Extract ( Start AS Integer [ , Length AS Integer ] ) AS Float[]
...
```

Αφαιρεί στοιχεία από μια διάταξη από το στοιχείο με την θέση Start και τόσα στοιχεία όσο η τιμή Length και τα επιστρέφει.
Αν η τιμή Length είναι αρνητικός αριθμός επιστρέφονται όλα τα στοιχεία από την θέση Start και μετά.

Παράδειγμα:

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5    DIM aBool AS NEW Boolean[]
6    DIM bBool AS NEW Boolean[]
7
8    aBool.Add(TRUE)
9    aBool.Add(FALSE)
10   aBool.Add(TRUE)
11   aBool.Add(TRUE)
12   aBool.Add(TRUE)
13
14   bBool = aBool.Extract(2, -1)
15
16   PRINT aBool.Count
17   PRINT bBool.Count
18
19   PRINT bBool[0]
20   PRINT bBool[1]
21   PRINT bBool[2]
22
23 END
```

Έξοδος:

```
2
3
True
True
True
```

Ανάλυση:
Στις γραμμές 5, 6 ορίζουμε δύο διατάξεις Boolean[].
Στις γραμμές 8 με 12 προσθέτουμε στοιχεία στην διάταξη aBool.
Στην γραμμή 14 αφαιρούμε τα στοιχεία της πρώτης διάταξης από την θέση 2 μέχρι το τέλος και τα εκχωρούμε στην διάταξη bBool[].

Στις γραμμές 16 και 17 ελέγχουμε πόσα στοιχεία έχει η κάθε διάταξη.
Η aBool[] έχει 2 και η bBool[] έχει τρία.

Στις γραμμές 19-21 τυπώνουμε τις τιμές των στοιχείων της δεύτερης διάταξης και όντως βλέπουμε ότι της έχει εκχωρηθούν οι τρεις τελευταίες τιμές της aBool[].

Exist

Σύνταξη:

```
FUNCTION Exist ( Value AS Boolean ) AS Boolean
```

Επιστρέφει TRUE αν η συγκεκριμένη τιμή υπάρχει εκχωρημένη σε κάποιο στοιχείο της διάταξης.

Παράδειγμα:

```
1   ' Gambas module file
2
3   PUBLIC SUB Main()
4
5     DIM aInts AS NEW Integer[]
6
7     aInts.Add(1)
8     aInts.Add(2)
9     aInts.Add(3)
10    aInts.Add(4)
11    aInts.Add(5)
12
13    PRINT aInts.Exist(0)
14    PRINT aInts.Exist(3)
15
16  END
```

Έξοδος:

```
False
True
```

Ανάλυση:

Στην γραμμή 5 δηλώνουμε μια διάταξη Integer[].
Στην συνέχεια στις γραμμές 7 με 11 προσθέτουμε τιμές.

Στην γραμμή 13 ελέγχουμε αν υπάρχει η τιμή 0 και λαμβάνουμε FALSE.
Στην γραμμή 14 ελέγχουμε αν υπάρχει η τιμή 3 και λαμβάνουμε TRUE.

Fill

Σύνταξη:

```
SUB Fill ( Value AS Boolean [ , Start AS Integer, Length AS Integer ] )
SUB Fill ( Value AS Integer [ , Start AS Integer, Length AS Integer ] )
SUB Fill ( Value AS Float [ , Start AS Integer, Length AS Integer ] )
...
```

Εκχωρεί την τιμή Value στα στοιχεία της διάταξης από την θέση Start μέχρι και Length στοιχεία.
Αν για παράδειγμα η τιμή Start είναι 2 και η τιμή Length είναι 4 τότε τα στοιχεία 2, 3, 4, 5 θα έχουν την τιμή Value.

Find

Σύνταξη:

```
FUNCTION Find ( Value AS Boolean ) AS Integer
```

Ψάχνει την τιμή Value σε μια διάταξη και επιστρέφει την θέση στην οποία την βρίσκει πρώτα.

Παράδειγμα:

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5    DIM aInts AS NEW Integer[]
6
7    aInts.Add(1)
8    aInts.Add(4)
9    aInts.Add(4)
10   aInts.Add(4)
11   aInts.Add(5)
12   PRINT aInts.Find(4)
13
14 END
```

Έξοδος:

```
1
```

Ανάλυση:
Στην γραμμή 5 δηλώνουμε έναν πίνακα ακεραίων.
Στις γραμμές 7 με 11 προσθέτουμε τιμές.
Στην γραμμή 12 τυπώνουμε την πρώτη θέση που υπάρχει η τιμή 4.
Η συνάρτηση επιστρέφει την θέση 1.

Insert
Σύνταξη:

```
FUNCTION Insert ( Array AS Boolean[] [ , Pos AS Integer ] ) AS Boolean[]
FUNCTION Insert ( Array AS Date[] [ , Pos AS Integer ] ) AS Date[]
FUNCTION Insert ( Array AS Float[] [ , Pos AS Integer ] ) AS Float[]
...
```

Εισάγει τις τιμές ενός πίνακα στον υπάρχων στην θέση Pos.
Αν δεν οριστεί θέση τότε οι τιμές εισάγονται στο τέλος του πίνακα.

Παράδειγμα:

```
1   ' Gambas module file
2
3   PUBLIC SUB Main()
4
5     DIM aInts AS NEW Integer[]
6     DIM bInts AS NEW Integer[]
7     DIM element AS Integer
8
9     aInts.Add(1)
10    aInts.Add(4)
11    aInts.Add(4)
12    aInts.Add(4)
13    aInts.Add(5)
14
15    bInts.Add(6)
16    bInts.Add(7)
17    bInts.Add(8)
18    bInts.Add(9)
19    bInts.Add(10)
20
21    aInts.Insert(bInts, 3)
22
23    FOR EACH element IN aInts
24      PRINT element;;
25    NEXT
26
27  END
```

Έξοδος:

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

Ανάλυση:
Στις γραμμές 5 με 7 δηλώνουμε δύο πίνακες ακεραίων και μια μεταβλητή ακεραίου integer.
Στις γραμμές 9 με 13 εισάγουμε τιμές στον πρώτο πίνακα.
Στις γραμμές 15 με 19 εισάγουμε τιμές στον δεύτερο πίνακα.
Στην γραμμή 21 εισάγουμε στο στοιχείο 3 και μετά τις τιμές του δεύτερου πίνακα bInts[] στον πρώτο πίνακα aInts[].

Για να επαληθεύσουμε το αποτέλεσμα τυπώνουμε τις τιμές των στοιχείων του aInts[].

Βλέπουμε ότι τα στοιχεία 3, 4, 5, 6, 7 έχουν τις τιμές του πίνακα aInts[] και επίσης ότι ο πίνακας άλλαξε μέγεθος αποκτώντας ακόμη 5 στοιχεία.

Pop
Σύνταξη:

```
FUNCTION Pop ( ) AS Boolean
FUNCTION Pop ( ) AS Date
FUNCTION Pop ( ) AS Float
...
```

Αφαιρεί το τελευταίο στοιχείο του πίνακα και το επιστρέφει.
Αν ο πίνακας είναι κενός ένα λάθος επιστρέφεται.

Push
Σύνταξη:

```
SUB Push ( Value AS Boolean )
SUB Push ( Value AS Date )
SUB Push ( Value AS Float )
...
```

Προσθέτει ένα στοιχείο στο τέλος του πίνακα.

Read
Σύνταξη:

```
SUB Read ( Stream AS Stream [ , Start AS Integer, Length AS Integer
] )
```

Εκχωρεί τιμές στα στοιχεία ενός πίνακα διαβάζοντας ένα αντικείμενο stream.
Εξ' ορισμού οι τιμές εισάγονται στην αρχή του πίνακα και μέχρι το τέλος του.
Αν οριστεί το Start οι τιμές εισάγονται από την θέση αυτή.
Αν οριστεί το Length οι τιμές εισάγονται για Length στοιχεία.

Sort
Σύνταξη:

```
FUNCTION Sort ( [ Mode AS Integer ] ) AS Boolean[]
FUNCTION Sort ( [ Mode AS Integer ] ) AS Date[]
FUNCTION Sort ( [ Mode AS Integer ] ) AS Float[]
...
```

Ταξινομεί τα στοιχεία ενός πίνακα κατά αύξουσα ή φθίνουσα σειρά.
Η λέξη Mode δηλώνει τον τύπο ταξινόμησης και μπορεί να πάρει τις τιμές
gb.Ascent
gb.Descent

Write
Σύνταξη:

```
SUB Write ( Stream AS Stream [ , Start AS Integer, Length AS
Integer ] )
```

Γράφει τα δεδομένα ενός πίνακα σε ένα αντικείμενο stream.
Εξ' ορισμού οι τιμές που γράφονται είναι από την αρχή του πίνακα και μέχρι το τέλος του.
Αν οριστεί το Start οι τιμές γράφονται από την θέση αυτή.
Αν οριστεί το Length οι τιμές γράφονται για Length στοιχεία.

Η κλάση String[]

Η κλάση String[] έχει τις ίδιες ιδιότητες με τις προηγούμενες κλάσεις και τις περισσότερες μεθόδους από τις προηγούμενες εκτός των Read(), Write().
Επιπρόσθετα έχει την μέθοδο Join().

Join
Σύνταξη:

```
FUNCTION Join ( [ Separator AS String, Escape AS String ] ) AS String
```

Ενώνει όλα τα στοιχεία του πίνακα σε ένα string.
Τα στοιχεία διαχωρίζονται από τον χαρακτήρα Seperator και εσωκλείονται με τον χαρακτήρα Escape.
Αν υπάρχουν δύο χαρακτήρες Escape τότε ο πρώτος χαρακτήρας είναι ο πρώτος χαρακτήρας εγκλεισμού της λέξης και ο δεύτερος χαρακτήρας ο τελευταίος χαρακτήρας εγκλεισμού της λέξης.

Οι χαρακτήρες διαφυγής μέσα στα string διπλασιάζονται.

Παράδειγμα:

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  DIM aClaim AS String[]
6
7  aClaim = ["Gambas", "Almost", "Means", "Basic", "!"]
8  PRINT aClaim.Join("/")
9  PRINT aClaim.Join(",", "!")
10
11 END
```

Έξοδος:

```
Gambas/Almost/Means/Basic/!
!Gambas!,!Almost!,!Means!,!Basic!,!!!!
```

Η κλάση Object[]

Η κλάση Object[] έχει τις ίδιες ιδιότητες με τις προηγούμενες κλάσεις και τις εξής μεθόδους:
Add() Clear() Copy() Delete() Exist() Extract() Fill() Find() Insert() Pop() Push() Remove() Resize() Reverse() Sort()

Οι μέθοδοι αυτοί περιγράφτηκαν στις προηγούμενες κλάσεις.

Η κλάση Variant[]

Η κλάση Variant[] έχει τις ίδιες ιδιότητες με τις προηγούμενες κλάσεις και τις εξής μεθόδους:
Add() Clear() Copy() Delete() Extract() Fill() Insert() Pop() Push() Remove() Resize() Reverse()

Οι μέθοδοι αυτοί περιγράφτηκαν στις προηγούμενες κλάσεις.

Κεφάλαιο 9ο – Δομές (Structures)

Μια δομή είναι ένα σύνολο τύπων δεδομένων που προσπελάζονται κάτω από το ίδιο όνομα.
Οι τύποι δεδομένων δεν είναι απαραίτητο να είναι όλοι ίδιοι.
Μπορεί να έχουμε κάτω από το ίδιο όνομα έναν ακέραιο, ένα πραγματικό αριθμό, ένα byte, ένα boolean κτλ.

Η βασική διαφορά των δομών σε σχέση με τους πίνακες είναι ότι οι πίνακες είναι συλλογές δεδομένων του ίδιου τύπου κάτω από το ίδιο όνομα ενώ οι δομές συλλογές δεδομένων διαφορετικών τύπων κάτω από το ίδιο όνομα.

Δήλωση δομών

Μια δομή δηλώνεται με την παρακάτω σύνταξη.
Μια δομή είναι ακριβώς όπως μια κλάση με την διαφορά όμως ότι έχει δημόσιες μεταβλητές.
Μια δομή δηλώνεται στην περιοχή καθολικών μεταβλητών.

Σύνταξη:

```
PUBLIC STRUCT Identifier

  Field 1 [ Embedded array declaration ] AS [ Datatype ]
  Field 2 [ Embedded array declaration ] AS [ Datatype ]
    .
    .
    .
  Field n [ Embedded array declaration ] AS [ Datatype ]

END STRUCT
```

Αν δηλώσετε την ίδια δομή σε διαφορετικές κλάσεις αυτές θα πρέπει να έχουν τα ίδια πεδία, αλλιώς ο διερμηνευτής θα εμφανίσει ένα λάθος.

Μια δομή αποτελείται από το όνομα της, Identifier

και τα πεδία της όπου αυτά μπορεί να είναι διαφορετικού τύπου δεδομένων.

Παράδειγμα:

```
1  ' Gambas module file
2
3  Public Struct Product
4    Weight As Integer
5    Price As Float
6  End Struct
```

Στο συγκεκριμένο παράδειγμα έχουμε δηλώσει μια δομή με το όνομα Product η οποία έχει δύο πεδία, έναν ακέραιο και ένα πραγματικό αριθμό.

Στην συνέχεια μπορούμε να δηλώσουμε αντικείμενα αυτής της δομής τα οποία θα έχουν ως ιδιότητες τα πεδία της δομής.

Παράδειγμα:
Το παράδειγμα είναι το Project28

```
1  ' Gambas module file
2
3  Public Struct Product
4    Weight As Integer
5    Price As Float
6  End Struct
7
8  Public apple As New Product
9
10
11 Public Sub Main()
12
13   Dim banana As New Product
14
15   apple.Price = 12.34
16   apple.Weight = 23
17
18   banana.Price = 20.54
19   banana.Weight = 16
20
21   Print "The apple is weight:"; apple.Weight
22   Print "The apple costs:"; apple.Price
23   Print "The banana is weight"; banana.Weight
24   Print "The banana costs:"; banana.Price
25
26 End
```

Ανάλυση:
Στις γραμμές 3 με 6 δηλώνουμε την δομή με τα πεδία της.
Στην γραμμή 8 δηλώνουμε ένα αντικείμενο της δομής αυτής με το όνομα apple. Το αντικείμενο apple είναι δηλωμένο ως καθολικό δημόσιο αντικείμενο Product.
Στην γραμμή 11 ξεκινάει η ρουτίνα Main().
Στην γραμμή 13 δηλώνουμε ένα άλλο αντικείμενο Product το οποίο έχει τοπική εμβέλεια.
Στις γραμμές 16 – 19 εκχωρούμε τιμές στα πεδία των αντικειμένων apple, banana.
Κάθε αντικείμενο Product έχει τις ιδιότητες Price και Weight, στις οποίες μπορούμε να δώσουμε τιμές.

Στην συνέχεια στις γραμμές 21-24 τυπώνουμε στην stdout τις τιμές που έχουν τα πεδία των αντικειμένων Product.

Διαχείριση των πεδίων

Τα πεδία καταλαμβάνουν τόση μνήμη όση είναι πολλαπλάσια του τύπου δεδομένων της δομής.

- Ένας Boolean ή ένα Byte μπορεί να αποθηκευτεί σε οποιαδήποτε διεύθυνση μνήμης.
- Ένας Short αποθηκεύεται σε ζυγό αριθμό διεύθυνσης.
- Ένας Integer αποθηκεύεται σε διεύθυνση μνήμης που είναι πολλαπλάσιο του 4.
- κτλ...

Με αυτόν τον τρόπο δήλωσης θέσεων μνήμης μπορεί να υπάρχουν κενά μεταξύ των πεδίων μιας δομής. Για παράδειγμα, αν δηλώσετε ένα Byte πεδίο και μετά ένα Integer πεδίο, θα έχετε ένα κενό 3 bytes.

Σε αυτό το θέμα υπάρχει ένα πρόβλημα. Όταν η γλώσσα C μεταγλωττίζει ένα πηγαίο κώδικα, ο μεταγλωττιστής μπορεί και διευθετεί τα πεδία μιας δομής. Αυτή η εργασία δεν είναι τυποποιημένη ούτε περιγράφεται σε κάποια τεκμηρίωση.

Στο μέλλον μπορεί να υπάρξει κάποια λύση στο θέμα αυτό με την βοήθεια της βιβλιοθήκης libffi που χρησιμοποιεί η Gambas. Η βιβλιοθήκη αυτή μπορεί να στείλει μια δομή σε μια συνάρτηση της C διορθώνοντας το πρόβλημα αυτό.

Ενσωματωμένες δομές

Μια δομή μπορεί να είναι ενσωματωμένη μέσα σε μια κανονική κλάση ή σε μια άλλη δομή. Αυτό γίνεται με το να δηλωθεί μαι μεταβλητή με τον ακόλουθο τρόπο:

Σύνταξη:

	[**PRIVATE** | **PUBLIC**] Identifier **AS STRUCT** Structure name

Παραδείγματα δηλώσεων δομών

```
1  ' Gambas class file
2
3  PUBLIC STRUCT Arm
4    Length AS Float
5    NumberOfFingers AS Integer
6    HasGlove AS Boolean
7  END STRUCT
8
9  PUBLIC STRUCT Leg
10   Length AS Float
11   NumberOfFingers AS Integer
12   HasSock AS Boolean
13   HasShoe AS Boolean
14 END STRUCT
15
16 PUBLIC STRUCT Man
17   FirstName AS String
18   LastName AS String
19   Age AS Integer
20   Eyes AS String
21   LeftArm AS STRUCT Arm
22   RightArm AS STRUCT Arm
23   LeftLeg AS STRUCT Leg
24   RightLeg AS STRUCT Leg
25 END STRUCT
```

Διατάξεις δομών

Μπορείτε επίσης να δημιουργήσετε διατάξεις δομών.
Για να δημιουργήσετε μια τέτοια διάταξη θα πρέπει να δηλώσετε την διάταξη ως εξής:

Σύνταξη:

	`Identifier [ Dimensions ] AS STRUCT Structure name`

Οι διατάξεις αυτού του τύπου είναι αποκλειστικά στατικές (embedded array) και όχι δυναμικές διατάξεις.
Τα μέλη της δομής βρίσκονται ενσωματωμένα στην διάταξη.
Τέτοιες διατάξεις δεν είναι πραγματικοί πίνακες, έχουν ορισμένες μεθόδους από την κλάση array.

- Μπορούν να προσπελαστούν από ένα ευρετήριο (index)
- Μπορούν να απαριθμηθούν
- Μπορείτε να πάρετε πληροφορίες για το μήκος και τις διαστάσεις του πίνακα.

Κεφάλαιο 10ο – Συναρτήσεις της Gambas

Η Gambas μας δίνει ένα πλήθος έτοιμων συναρτήσεων για να διαχειριζόμαστε τα δεδομένα μας.
Ήδη είδαμε τις συναρτήσεις μετατροπής τύπων δεδομένων. Εκτός από αυτές τις συναρτήσεις υπάρχουν αριθμητικές συναρτήσεις, διαχείρισης bits, ελέγχου χαρακτήρων, συναρτήσεις ημερομηνίας και χρόνου, συναρτήσεις μορφοποίησης, τριγωνομετρικές, λογαριθμικές, διαχείρισης διεργασιών, διαχείρισης αρχείων και φακέλων και πολλές άλλες.

Σε αυτό το κεφάλαιο θα δούμε αναλυτικά όλες τις συναρτήσεις που προσφέρει η Gambas.
Οι έτοιμες αυτές συναρτήσεις που προσφέρει η Gambas ονομάζονται “εγγενείς” συναρτήσεις (Intrinsic Functions).

Αριθμητικές συναρτήσεις (Arithmetical Functions)

`Abs`	Επιστρέφει την απόλυτη τιμή ενός αριθμού.
`Ceil`	Επιστρέφει τον μικρότερο ακέραιο αριθμό από ένα δοσμένο αριθμό
`DEC`	Μειώνει μια αριθμητική μεταβλητή.
`Fix`	Επιστρέφει το ακέραιο μέρος ενός αριθμού
`Floor`	Επιστρέφει το ακέραιο μέρος ενός αριθμού που δεν είναι μεγαλύτερο από τον δοσμένο αριθμό
`Frac`	Επιστρέφει το κλασματικό μέρος ενός αριθμού.
`INC`	Αυξάνει μια αριθμητική μεταβλητή
`Int`	Επιστρέφει το ακέραιο μέρος ενός αριθμού.
`Max`	Επιστρέφει τον μέγιστο αριθμό.
`Min`	Επιστρέφει τον μικρότερο αριθμό.
`Round`	Στρογγυλοποιεί έναν αριθμό.
`Sgn`	Επιστρέφει το πρόσημο ενός αριθμού.

Abs

Επιστρέφει την απόλυτη τιμή ενός αριθμού.

Σύνταξη:

```
Value = Abs ( Number AS Float )
```

Παράδειγμα:

```
PRINT Abs(-2.5)
```

Έξοδος:

```
2.5
```

Ceil

Επιστρέφει τον μικρότερο ακέραιο αριθμό από ένα δοσμένο αριθμό

Σύνταξη:

```
Value = Ceil ( Number )
```

Παράδειγμα:

```
PRINT Ceil(Pi)
```

Έξοδος:

```
4
```

DEC

Μειώνει μια αριθμητική μεταβλητή.

Σύνταξη:

```
DEC Variable
```

Η εντολή αυτή είναι αντίστοιχη της

```
Variable = Variable - 1
```

ή

```
Variable -= 1
```

Παράδειγμα:

```
X = 7
DEC X
PRINT X
```

Έξοδος:

```
6
```

Fix

Επιστρέφει το ακέραιο μέρος ενός αριθμού.

Σύνταξη:

```
Value = Fix ( Number )
```

Παράδειγμα:

```
PRINT Fix(Pi)
```

Έξοδος:

```
3
```

Floor

Επιστρέφει το ακέραιο μέρος ενός αριθμού που δεν είναι μεγαλύτερο από τον δοσμένο αριθμό. Όταν ο αριθμός είναι ένας Float η συνάρτηση αυτή είναι συνώνυμη της Int()

Σύνταξη:

```
Value = Floor ( Number )
```

Παράδειγμα:

```
PRINT Floor(Pi)
PRINT Floor(-Pi)
```

Έξοδος:

```
3
-4
```

Frac

Επιστρέφει το κλασματικό μέρος ενός αριθμού.

Σύνταξη:

```
Value = Frac ( Number ) AS Float
```

Παράδειγμα:

```
PRINT Frac(Pi)
```

Έξοδος:

```
0.14159265359
```

INC

Αυξάνει μια αριθμητική μεταβλητή

Σύνταξη:

```
INC Variable
```

Η εντολή αυτή είναι αντίστοιχη της

```
Variable = Variable + 1
```

ή

```
Variable += 1
```

Παράδειγμα:

```
X = 7
DEC X
PRINT X
```

Έξοδος:

```
8
```

Int

Επιστρέφει το ακέραιο μέρος ενός αριθμού.

Σύνταξη:

```
Value = Int ( Number )
```

Παράδειγμα:

```
PRINT Int(Pi)
```

Έξοδος:

```
3
```

Η συμπεριφορά της Int και Fix είναι η ίδια για θετικούς αριθμούς και διαφορετική για αρνητικούς.

Οι διαφορές της Int() με την Cint() είναι:

- Η Int() μπορεί να επιστρέψει έναν Float, η CInt περιορίζεται στους ακεραίους 32 bit.
- Η Int() στρογγυλοποιεί στην επόμενη χαμηλή τιμή π.χ. -4.6 σε -5, ενώ η CInt στρογγυλοποιεί γύρω στο 0 π.χ. -4.6 σε -4

Max

Επιστρέφει τον μέγιστο αριθμό.

Σύνταξη:

```
Value = Max ( Expression , Expression )
```

Παράδειγμα:

```
PRINT Max(6, 4)
```

Έξοδος:

```
6
```

Min

Επιστρέφει τον μικρότερο αριθμό.

Σύνταξη:

```
Value = Min ( Expression , Expression )
```

Παράδειγμα:

```
PRINT Min(6, 4)
```

Έξοδος:

```
4
```

Round

Στρογγυλοποιεί έναν αριθμό.

Σύνταξη:

```
Value = Round ( Number [ , Digits ] )
```

Εάν ορίζεται η τιμή Digits η στρογγυλοποίηση γίνεται προς το `10 ^` Digits.

Παράδειγμα:

```
PRINT Round(Pi, -2)
```

Έξοδος:

```
3.14
```

Sgn

Επιστρέφει το πρόσημο ενός αριθμού.

Σύνταξη:

```
Sign = Sgn ( Number AS Float )
```

Εάν ο αριθμός είναι 0 τότε μηδέν επιστρέφεται.
Εάν είναι αρνητικός -1 επιστρέφεται.
Εάν είναι θετικός +1 επιστρέφεται.

Παράδειγμα:

```
PRINT Sgn(-3)
PRINT Sgn(3)
PRINT Sgn(0)
```

Έξοδος:

```
-1
1
0
```

Συναρτήσεις διαχείρισης Bits (Bits Manipulation Functions)

Bits και Bytes

Λίγα λόγια για τα bits και τα byte.
Ένα bit είναι το ψηφίο 0 ή 1. Ένας δυαδικός αριθμός αποτελείται από ένα σύνολο ψηφίων 0 και 1. Για παράδειγμα 01011101. Οκτώ bit αποτελούν ένα byte. Για παράδειγμα 10011101.
Η αρίθμηση των ψηφίων ενός δυαδικού αριθμού γίνεται από τα δεξιά προς τα αριστερά.

1	**0**	**0**	**1**	**1**	**1**	**0**	**1**
8ο ψηφίο	7ο ψηφίο	6ο ψηφίο	5ο ψηφίο	4ο ψηφίο	3ο ψηφίο	2ο ψηφίο	1ο ψηφίο

Ένας δυαδικός στα μαθηματικά αναπαριστάται με τα ψηφία του και βάση το 2 ως εξής:
10011101_2

Ένας δεκαδικός αντίστοιχα με βάση το 10 είναι για παράδειγμα 157_{10}

Για να μετατρέψουμε έναν δυαδικό αριθμό σε δεκαδικό προσθέτουμε το αποτέλεσμα των ψηφίων επί το 2 υψωμένο στην δύναμη που ορίζει η θέση του ψηφίου μείον 1.

Πιο απλά στο παραπάνω παράδειγμα
το πρώτο ψηφίο είναι το **1** στην θέση ***1*** οπότε παράγουμε το αποτέλεσμα $1X2^{0.}$
Το δεύτερο ψηφίο είναι το **0** στην θέση ***2*** οπότε παράγουμε το αποτέλεσμα $0X2^1$
κτλ

Έτσι έχουμε $\mathbf{1}X2^0 + \mathbf{0}X2^1 + \mathbf{1}X2^2 + \mathbf{1}X2^3 + \mathbf{1}X2^4 + \mathbf{0}X2^5 + \mathbf{0}X2^6 + \mathbf{1}X2^7 = 157$ δλδ $10011101_2 = 157_{10}$

<u>Αναπαράσταση αρνητικών αριθμών στο δυαδικό σύστημα</u>

Για να αναπαρασταθούν αρνητικοί αριθμοί με το δυαδικό σύστημα αρίθμησης χρησιμοποιούνται δύο τρόποι, το *συμπλήρωμα ως προς 1* και το *συμπλήρωμα ως προς 2*.

Συμπλήρωμα ως προς 1
Σε αυτήν την μέθοδο αντιστρέφονται τα ψηφία του δυαδικού αριθμού, όπου δηλαδή 0 γίνεται 1 και όπου 1 γίνεται 0, και ο αριθμός που προκύπτει θεωρείται ο αρνητικός του πρώτου.

Έτσι για παράδειγμα, ο (θετικός) αριθμός 7 σε μορφή ψηφιολέξης (byte) είναι ο ακόλουθος:
0000 0111
και ο αρνητικός -7 σε μορφή συμπληρώματος ως προς 1 γίνεται:
1111 1000

Το πρόβλημα με την συγκεκριμένη μέθοδο είναι πως υπάρχουν δύο αναπαραστάσεις για το μηδέν:
0000 0000 (για ένα "θετικό" μηδέν)
και
1111 1111 (για ένα "αρνητικό" μηδέν)

Για να αντιμετωπιστεί αυτό το πρόβλημα δημιουργήθηκε μια δεύτερη μέθοδος, το συμπλήρωμα ως προς 2

Συμπλήρωμα ως προς 2
Στο συμπλήρωμα ως προς 2, μετά την αντιστροφή των δυαδικών ψηφίων προστίθεται επιπλέον ο αριθμός 1. Έτσι, και πάλι με παράδειγμα τον αριθμό 7 σε μορφή ψηφιολέξης (byte):
0000 0111
αντιστρέφουμε όπως στο συμπλήρωμα ως προς 1:
1111 1000
και τελικά προσθέτουμε το 1:
1111 1001

Η Gambas για να αναπαραστήσει τους αρνητικούς αριθμούς χρησιμοποιεί την μέθοδο συμπληρώματος ως προς 2.

Πρόσημο και μέτρο
Υπάρχει και ένας τρίτος τρόπος αναπαράστασης των αρνητικών αριθμών, ο οποίος δε χρησιμοποιείται πολύ συχνά, αλλά είναι πιο προσιτός στον άνθρωπο , καθώς μοιάζει πολύ με τον τρόπο αναπαράστασης αρνητικών αριθμών στο δεκαδικό σύστημα.

Στο σύστημα αυτό, το πρώτο από αριστερά δυαδικό ψηφίο λαμβάνεται ως το πρόσημο του δυαδικού αριθμού, ενώ τα ψηφία που ακολουθούν χρησιμοποιούνται για τον υπολογισμό του μέτρου του. Αν το πρώτο ψηφίο από τα αριστερά είναι 0, ο αριθμός θεωρείται θετικός, ενώ αν το πρώτο ψηφίο από τα αριστερά είναι 1, ο αριθμός θεωρείται αρνητικός.

Έτσι, αν χρησιμοποιούμε 8 bits (δυαδικά ψηφία) για την αναπαράσταση του αριθμού, το 7 είναι

00000111

ενώ , το -7 θα είναι αντίστοιχα

10000111

Οι συναρτήσεις διαχείρισης bits είναι:

`BClr`	Θέτει μηδέν το ψηφίο bit ενός ακεραίου.
`BSet`	Θέτει σε ένα το ψηφίο bit ενός ακεραίου.
`BTst`	Ελέγχει λογικά, σε έναν ακέραιο, αν ένα ψηφίο bit είναι 1.
`BChg`	Αντιστρέφει ένα ψηφίο bit σε έναν ακέραιο.
`Lsl`	Μετατοπίζει έναν ακέραιο προς τα αριστερά.
`Lsr`	Μετατοπίζει έναν ακέραιο προς τα δεξιά.
`Shl` / `Asl`	Μετατοπίζει έναν ακέραιο προς τα αριστερά διατηρώντας το πρόσημο του.
`Shr` / `Asr`	Μετατοπίζει έναν ακέραιο προς τα δεξιά διατηρώντας το πρόσημο του.
`Rol`	Περιστρέφει έναν ακέραιο προς τα αριστερά.
`Ror`	Περιστρέφει έναν ακέραιο προς τα δεξιά.

BClr

Θέτει μηδέν το ψηφίο bit ενός ακεραίου.

Σύνταξη:

```
Value = BClr ( Number AS Integer , Bit AS Integer ) AS Integer
```

- Value, είναι η τιμή που επιστρέφει η συνάρτηση, η τιμή είναι αριθμητική
- Number, ο ακέραιος αριθμός
- Bit, η θέση του ψηφίου bit που θα τεθεί σε μηδέν. Η αρίθμηση ξεκινά από το μηδέν.

Παράδειγμα:

```
'15 in bit format is 1111
 PRINT BClr(15, 0) '1110 clear first digit
 PRINT BClr(15, 1) '1101 clear second digit
 PRINT BClr(15, 3) '0111 clear fourth digit
```

Έξοδος:

```
14
13
7
```

BSet

Θέτει σε ένα το ψηφίο bit ενός ακεραίου.

Σύνταξη:

```
Value = BSet ( Number AS Integer , Bit AS Integer ) AS Integer
```

- Value, είναι η τιμή που επιστρέφει η συνάρτηση, η τιμή είναι αριθμητική
- Number, ο ακέραιος αριθμός
- Bit, η θέση του ψηφίου bit που θα τεθεί σε ένα. Η αρίθμηση ξεκινά από το μηδέν.

Παράδειγμα:

```
'9 in bit format is 1001
PRINT BSet(9, 0) '1001 makes nothing first digit is already 1
PRINT BSet(9, 1) '1011 this is 11
PRINT BSet(9, 2) '1101 this is 13
PRINT BSet(9, 4) '11001 this is 25, added one digit 1 1+1001
```

Έξοδος:

```
9
11
13
25
```

BTst

Ελέγχει λογικά, σε έναν ακέραιο, αν ένα ψηφίο bit είναι 1.

Σύνταξη:

```
Result = BTst ( Number AS Integer , Bit AS Integer ) AS Boolean
```

- Resutl, είναι η τιμή που επιστρέφει η συνάρτηση, η τιμή είναι ένας Bolean
- Number, ο ακέραιος αριθμός
- Bit, η θέση του ψηφίου bit που θα ελεχθεί. Η αρίθμηση ξεκινά από το μηδέν.

Παράδειγμα:

```
'9 in bit format is 1001
PRINT BTst(9, 0)
PRINT BTst(9, 1)
PRINT BTst(9, 2)
PRINT BTst(9, 3)
PRINT BTst(9, 4)
```

Έξοδος:

```
True
False
False
True
False
```

BChg

Αντιστρέφει ένα ψηφίο bit σε έναν ακέραιο.

Σύνταξη:

```
Value = BChg ( Number AS Integer , Bit AS Integer ) AS Integer
```

- Value, είναι η τιμή που επιστρέφει η συνάρτηση, η τιμή είναι αριθμητική
- Number, ο ακέραιος αριθμός
- Bit, η θέση του ψηφίου bit που θα αντιστραφεί. Αν το ψηφίο είναι 0 γίνεται 1και αν είναι 1 γίνεται 0, αν δεν υπάρχει θεωρείται σαν να ήταν 0 και γίνεται 1. Η αρίθμηση ξεκινά από το μηδέν.

Παράδειγμα:

```
'9 in bit format is 1001
PRINT BChg(9, 0) '1000 this is 8
PRINT BChg(9, 1) '1011 this is 11
PRINT BChg(9, 2) '1101 this is 13
PRINT BChg(9, 4) '11001 this is 25
```

Έξοδος:

```
8
11
13
25
```

Lsl

Μετατοπίζει έναν ακέραιο προς τα αριστερά.

Σύνταξη:

```
Value = Lsl ( Number , Bit AS Integer )
```

Προσθέτει ψηφία 0 στα δεξιά του αριθμού μετατοπίζοντας τον προς τα αριστερά.
Το ψηφίο του πρόσημου αγνοείται.

- Value, είναι η τιμή που επιστρέφει η συνάρτηση, η τιμή είναι αριθμητική
- Number, ο ακέραιος αριθμός
- Bit, το πλήθος των bit που θα προστεθούν.

Παράδειγμα:

```
'9 in bit format is 1001
PRINT Lsl(9, 0) '1001 this is 9 no shift made
PRINT Lsl(9, 1) '10010 this is 18
PRINT Lsl(9, 2) '100100 this is 36
PRINT Lsl(9, 4) '10010000 this is 144
```

Έξοδος:

```
9
18
36
144
```

Η Lsl() δεν επεξεργάζεται τα bit του πρόσημου των αριθμών και έτσι δεν το διατηρεί κατά την μετατόπιση.

Lsr

Μετατοπίζει έναν ακέραιο προς τα δεξιά.

Σύνταξη:

```
Value = Lsl ( Number , Bit AS Integer )
```

Αφαιρεί ψηφία από τα δεξιά του αριθμού από την δοθείσα θέση και δεξιά.
Το ψηφίο του πρόσημου αγνοείται.

- Value, είναι η τιμή που επιστρέφει η συνάρτηση, η τιμή είναι αριθμητική
- Number, ο ακέραιος αριθμός
- Bit, το πλήθος των bit που θα αφαιρεθούν.

Παράδειγμα:

```
'9 in bit format is 1001
PRINT Lsr(9, 0) '1001 this is 9 no shift made
PRINT Lsr(9, 1) '100 this is 4, 100-1
PRINT Lsr(9, 2) '10 this is 2,  10-01
PRINT Lsr(9, 4) '0 this is 0,   0-1001
```

Έξοδος:

Η Lsr() θα επεξεργαστεί 32 bits από έναν αρνητικό short αριθμό και 64 bits από integer, long.

Έτσι για παράδειγμα:
Ένας αρνητικός short είναι ο -9_2 =
$11111111111111111111111111111111111111111111111111111111111110111_2$

Η Lsr() επειδή το -9 είναι short θα επεξεργαστεί τα bits από το 1ο έως το 32ο.
Δηλαδή τα 11111111111111111111111111110111.
Θα αφαιρέσει από αυτά το πρώτο και θα μετατοπίσει τα υπόλοιπα 31 δεξιά.
1111111111111111111111111111011-1
0-111111111111111111111111111011
Τα υπόλοιπα bits θα τα συμπληρώσει με 0.
Οπότε θα έχουμε τα 64 bits
0000000000000000000000000000000111111111111111111111111111011.

Έτσι

```
PRINT Bin(Lsr(-9,1), 64)
'0000000000000000000000000000000111111111111111111111111111011
```

Ανάλυση:
Εκτελούμε την συνάρτηση Lsr(), για τον αριθμό -9 και μετατοπίζουμε 1 bit.
Η συνάρτηση Bin() που δέχεται σαν όρισμα ότι επιστρέφει η Lsr(-9, 1) επιστρέφει με την σειρά της τον δυαδικό αριθμό με 64 ψηφία bit.

Στην πραγματικότητα όμως είδαμε ότι -9_2 =
$11111111111111111111111111111111111111111111111111111111111110111_2$
Για να μετατοπίσουμε το $11111111111111111111111111111111111111111111111111111111111110111_2$ κατά ένα bit θα πρέπει να μετατρέψουμε το -9 από short σε long.
Αυτό γίνεται με την συνάρτηση μετατροπής τύπων δεδομένων την Clng(-9).
Έτσι θα έπρεπε να γράψουμε:

```
PRINT Bin(Lsr(CLng(-9), 1), 64)
'0111111111111111111111111111111111111111111111111111111111111011
```

Η Lsr() με άλλα λόγια δεν επεξεργάζεται τα bit του πρόσημου των αριθμών και έτσι δεν το διατηρεί κατά την μετατόπιση.

Shl / Asl

Μετατοπίζει έναν Byte, Short, Integer, ή Long προς τα αριστερά διατηρώντας το πρόσημο του.
Σύνταξη:

```
Value = Shl ( Number , Bit AS Integer )

Value = Asl ( Number , Bit AS Integer )
```

Προσθέτει ψηφία 0 στα δεξιά του αριθμού μετατοπίζοντας τον προς τα αριστερά.

Τύπος	Εύρος Bit	Πλήθος Bit
Byte	0...7	8
Short	0...15	16
Integer	0...31	32
Long	0...63	64

- Value, είναι η τιμή που επιστρέφει η συνάρτηση, η τιμή είναι αριθμητική
- Number, ο ακέραιος αριθμός
- Bit, το πλήθος των bit που θα προστεθούν.

Παράδειγμα:

```
'9 in bit format is 1001
PRINT Shl(9, 0) '1001 this is 9 no shift made
PRINT Shl(9, 1) '10010 this is 18
PRINT Shl(9, 2) '100100 this is 36
PRINT Shl(9, 4) '10010000 this is 144
```

Έξοδος:

```
9
18
36
144
```

Shr / Asr

Μετατοπίζει έναν Byte, Short, Integer, ή Long προς τα δεξιά διατηρώντας το πρόσημο του.
Σύνταξη:

```
Value = Shr ( Number , Bit AS Integer )

Value = Asr ( Number , Bit AS Integer )
```

Μετατοπίζει έναν Byte, Short, Integer, ή Long προς τα δεξιά.
Αφαιρεί ψηφία από τα δεξιά του αριθμού μετατοπίζοντας τον προς τα δεξιά.

Τύπος	Εύρος Bit	Πλήθος Bit
Byte	0...7	8
Short	0...15	16
Integer	0...31	32
Long	0...63	64

- Value, είναι η τιμή που επιστρέφει η συνάρτηση, η τιμή είναι αριθμητική
- Number, ο ακέραιος αριθμός
- Bit, το πλήθος των bit που θα αφαιρεθούν.

Παράδειγμα:

```
'9 in bit format is 1001
PRINT Shr(9, 0) '1001 this is 9 no shift made
PRINT Shr(9, 1) '100-1 this is 4
PRINT Shr(9, 2) '10-01 this is 2
PRINT Shr(9, 4) '0-1001 this is 0
```

Έξοδος:

```
9
4
2
0
```

Προηγουμένως είδαμε ότι Lsr(CLng(-9), 1) έδινε τον
$0111111111111111111111111111111111111111111111111111111111111011_2$

Αν γράψουμε τώρα την

```
PRINT Bin(Shr(-9, 1), 64)
```

θα έχουμε τον εξής δυαδικό αριθμό

$1111111111111111111111111111111111111111111111111111111111111011_2$

Συγκρίνετε τους δύο αριθμούς.
$0111111111111111111111111111111111111111111111111111111111111011_2$
$1111111111111111111111111111111111111111111111111111111111111011_2$

Ο πρώτος έχει μετατοπιστεί προς τα δεξιά και δεν έχουν διατηρηθεί τα bits του πρόσημου.
Ο δεύτερος έχει μετατοπιστεί προς τα δεξιά και το πρόσημο δεν έχει αλλάξει, η Shr() το έχει διατηρήσει.

Rol

Περιστρέφει έναν ακέραιου προς τα αριστερά.
Σύνταξη:

```
Value = Rol ( Number , Bit )
```

- Value, είναι η τιμή που επιστρέφει η συνάρτηση, η τιμή είναι αριθμητική
- Number, ο ακέραιος αριθμός
- Bit, τα bits που θα περιστραφούν.

Αν Number = 5 και Bit=3 τότε ο αριθμός θα αυξηθεί κατά Value = Number x 2^{Bit} δλδ, Value = $5x2^3$= 40.

Έτσι

```
PRINT Rol(5, 3)
'40
```

Παράδειγμα:
Αν θέλαμε να δούμε την δυαδική αναπαράσταση θα είχαμε.

```
PRINT Bin(5, 64)
PRINT Bin(Rol(5, 3), 64)
PRINT Rol(5, 3)
```

Έξοδος:

```
0000000000000000000000000000000000000000000000000000000000000101
0000000000000000000000000000000000000000000000000000000000101000
40
```

Το $101_2=5_{10}$ και το $101000_2=40_{10}$ έτσι το 5 αυξήθηκε και έγινε 40.

Προσοχή στις υπερχειλίσεις! Όταν ο δυαδικός αριθμός είναι στα όρια που μπορεί να δεχθεί ένας ακέραιος και αυξήσετε την τιμή ο δυαδικός αριθμός θα υπερχειλίσει.

Έτσι αν έχετε τον ακέραιο 2147483647 που είναι η μεγαλύτερη τιμή που μπορεί να δεχθεί μια μεταβλητή ακεραίου και την στρέψετε αριστερά αυξάνοντας την κατά 1 bit θα έχετε το εξής αποτέλεσμα.

Παράδειγμα:

```
PRINT Bin(2147483647, 64)
PRINT Bin(Rol(2147483647, 1), 64)
PRINT Rol(2147483647, 1)
```

Έξοδος:

```
0000000000000000000000000000000011111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111110
-2
```

Ο αριθμός δεν θα είναι 2147483647 x 2^1 = 4294967294 αλλά -2.

Ror

Περιστρέφει έναν ακέραιου προς τα δεξιά.

Σύνταξη:

```
Value = Ror ( Number , Bit )
```

- Value, είναι η τιμή που επιστρέφει η συνάρτηση, η τιμή είναι αριθμητική
- Number, ο ακέραιος αριθμός
- Bit, τα bits που θα περιστραφούν.

Αν Number = 40 και Bit=3 τότε ο αριθμός θα μειωθεί κατά Value = Number / 2^{Bit} δλδ, Value = $40/2^3$= 5.

Έτσι

```
PRINT Ror(40, 3)
'5
```

Παράδειγμα:
Αν θέλαμε να δούμε την δυαδική αναπαράσταση θα είχαμε.

```
PRINT Bin(40, 64)
PRINT Bin(Ror(40, 3), 64)
PRINT Ror(40, 3)
```

Έξοδος:

```
0000000000000000000000000000000000000000000000000000000000101000
0000000000000000000000000000000000000000000000000000000000000101
5
```

Το $101000_2=40_{10}$ και το $101_2=5_{10}$ έτσι το 40 μειώθηκε και έγινε 5.

Προσοχή στις υπερχειλίσεις! Όταν ο δυαδικός αριθμός είναι στα όρια που μπορεί να δεχθεί ένας ακέραιος και μειώσετε την τιμή ο δυαδικός αριθμός θα υπερχειλίσει.

Συναρτήσεις ελέγχου χαρακτήρα (Character Test Functions)

Αυτές είναι

`IsAscii`	Ελέγχει αν ένα string περιέχει μόνο χαρακτήρες ASCII.
`IsBlank`	Ελέγχει αν ένα string περιέχει μόνο κενούς χαρακτήρες.
`IsDigit`	Ελέγχει αν ένα string περιέχει μόνο ψηφία.
`IsHexa`	Ελέγχει αν ένα string περιέχει δεκαεξαδικά ψηφία.
`IsLCase`	Ελέγχει αν ένα string περιέχει μικρά γράμματα.
`IsLetter`	Ελέγχει αν ένα string περιέχει μόνο γράμματα.
`IsPunct`	Ελέγχει αν ένα string περιέχει μόνο εκτυπώσιμους μη αλφαριθμητικούς χαρακτήρες.
`IsSpace`	Ελέγχει αν ένα string περιέχει μόνο space χαρακτήρες.
`IsUCase`	Ελέγχει αν ένα string περιέχει μόνο κεφαλαία.

IsAscii

Ελέγχει αν ένα string περιέχει μόνο χαρακτήρες ASCII.

Σύνταξη:

```
bResult = IsAscii ( sArgument AS String ) AS Boolean
bResult = Ascii? ( sArgument AS String ) AS Boolean
```

bResult, η τιμή που επιστρέφει η συνάρτηση είναι τιμή boolean.
sArgument, το όρισμα της συνάρτησης είναι ένας string ή μια έκφραση που η Gambas μετατρέπει σε string.

IsBlank

Ελέγχει αν ένα string περιέχει μόνο κενούς χαρακτήρες.

Σύνταξη:

```
Result = IsBlank ( sArgument AS String ) AS Boolean
Result = Blank? ( sArgument AS String ) AS Boolean
```

Result, η τιμή που επιστρέφει η συνάρτηση είναι τιμή boolean.
sArgument, το όρισμα της συνάρτησης είναι ένας string ή μια έκφραση που η Gambas μετατρέπει σε string.

IsDigit

Ελέγχει αν ένα string περιέχει μόνο ψηφία.

Σύνταξη:

```
Result = IsDigit ( sArg AS String ) AS Boolean
Result = Digit? ( sArg AS String ) AS Boolean
```

Result, η τιμή που επιστρέφει η συνάρτηση είναι τιμή boolean.
sArg, το όρισμα της συνάρτησης είναι ένας string ή μια έκφραση που η Gambas μετατρέπει σε string.

IsHexa

Ελέγχει αν ένα string περιέχει δεκαεξαδικά ψηφία.

Σύνταξη:

```
Result = IsHexa ( vExpr AS String ) AS Boolean
Result = Hexa? ( vExpr AS String ) AS Boolean
```

Result, η τιμή που επιστρέφει η συνάρτηση είναι τιμή boolean.
vExpr, το όρισμα της συνάρτησης είναι ένας string ή μια έκφραση που η Gambas μετατρέπει σε string.

IsLCase

Ελέγχει αν ένα string περιέχει μικρά γράμματα.

Σύνταξη:

```
Result = IsLCase ( sArg AS String ) AS Boolean
Result = LCase? ( sArg AS String ) AS Boolean
Result = IsLower ( sArg AS String ) AS Boolean
Result = Lower? ( sArg AS String ) AS Boolean
```

Result, η τιμή που επιστρέφει η συνάρτηση είναι τιμή boolean.
sArg, το όρισμα της συνάρτησης είναι ένας string ή μια έκφραση που η Gambas μετατρέπει σε string.

IsLetter

Ελέγχει αν ένα string περιέχει μόνο γράμματα.

Σύνταξη:

```
Result = IsLetter ( sArg AS String ) AS Boolean
Result = Letter? ( sArg AS String ) AS Boolean
```

Result, η τιμή που επιστρέφει η συνάρτηση είναι τιμή boolean.
sArg, το όρισμα της συνάρτησης είναι ένας string ή μια έκφραση που η Gambas μετατρέπει σε string.

IsPunct

Ελέγχει αν ένα string περιέχει μόνο εκτυπώσιμους μη αλφαριθμητικούς χαρακτήρες.

Σύνταξη:

```
bResult = IsPunct ( sExpr AS String ) AS Boolean
bResult = Punct? ( sExpr AS String ) AS Boolean
```

Result, η τιμή που επιστρέφει η συνάρτηση είναι τιμή boolean.
sExpr, το όρισμα της συνάρτησης είναι ένας string ή μια έκφραση που η Gambas μετατρέπει σε string.

Ένας εκτυπώσιμος χαρακτήρας μη αλφαριθμητικός είναι ένας χαρακτήρας από τους εξής
`!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~`

IsSpace

Ελέγχει αν ένα string περιέχει μόνο space χαρακτήρες.

Σύνταξη:

```
bResult = IsSpace ( sExpr AS String ) AS Boolean
bResult = Space? ( sExpr AS String ) AS Boolean
```

bResult, η τιμή που επιστρέφει η συνάρτηση είναι τιμή boolean.
sExpr, το όρισμα της συνάρτησης είναι ένας string ή μια έκφραση που η Gambas μετατρέπει σε string.

Ένας χαρακτήρας space είναι ένας από τους παρακάτω:

- Το κενό διάστημα " ",
- Το `Chr$(12)`,
- Το `"\n"`,
- Το `"\r"`,
- Το `"\t"`,
- Το `Chr$(11)`.

IsUCase

Ελέγχει αν ένα string περιέχει μόνο κεφαλαία.

Σύνταξη:

```
vResult = IsUCase ( vExpr AS String ) AS Boolean
vResult = UCase? ( vExpr AS String ) AS Boolean
vResult = IsUpper ( vExpr AS String ) AS Boolean
vResult = Upper? ( vExpr AS String ) AS Boolean
```

vResult, η τιμή που επιστρέφει η συνάρτηση είναι τιμή boolean.
vExpr, το όρισμα της συνάρτησης είναι ένας string ή μια έκφραση που η Gambas μετατρέπει σε string.

Συναρτήσεις μετατροπής τύπων δεδομένων (Conversion Functions)

Τις συναρτήσεις αυτές τις είδαμε σε προηγούμενο κεφάλαιο. Εδώ απλά θα τις αναφέρουμε.

Συναρτήσεις μετατροπής.	
`CBool`	Μετατρέπει μια τιμή σε Boolean.
`CByte`	Μετατρέπει μια τιμή σε Byte.
`CDate`	Μετατρέπει μια τιμή σε Date.
`CFloat`	Μετατρέπει μια τιμή σε Float.
`CInt`	Μετατρέπει μια τιμή σε Integer.
`CLong`	Μετατρέπει μια τιμή σε Long.
`Conv$`	Μετατρέπει ένα string από ένα σετ χαρακτήρων σε άλλο σετ χαρακτήρων.
`CShort`	Μετατρέπει μια τιμή σε Short.
`CSingle`	Μετατρέπει μια τιμή σε Single.
`CStr`	Μετατρέπει μια τιμή σε String.
`CVariant`	Μετατρέπει μια τιμή σε Variant
`DConv$`	Μετατρέπει ένα string από το σετ χαρακτήρων του συστήματος σε UTF-8 (desktop charset).
`SConv$`	Μετατρέπει ένα string από UTF-8 (desktop charset) σε σετ χαρακτήρων του συστήματος.
`Str$`	Μετατρέπει έναν αριθμό ή ημερομηνία σε string.
`Val`	Μετατρέπει ένα string σε αριθμό ή ημερομηνία.

Συναρτήσεις τύπων δεδομένων (Datatype Functions)

Οι συναρτήσεις ελέγχου τύπου δεδομένων είναι οι παρακάτω:

IsBoolean	Επιστρέφει TRUE αν μια έκφραση είναι τιμή Boolean.
IsDate	Επιστρέφει TRUE αν μια έκφραση είναι τιμή Date.
IsFloat	Επιστρέφει TRUE αν μια έκφραση είναι τιμή Float.
IsInteger	Επιστρέφει TRUE αν μια έκφραση είναι τιμή Integer.
IsLong	Επιστρέφει TRUE αν μια έκφραση είναι τιμή Long.
IsNull	Επιστρέφει TRUE αν μια έκφραση είναι τιμή NULL.
IsNumber	Επιστρέφει TRUE αν μια έκφραση είναι αριθμός.
SizeOf	Επιστρέφει το μέγεθος της μνήμης που χρησιμοποιεί ένας τύπος δεδομένων .
TypeOf	Επιστρέφει τον τύπο δεδομένων μιας έκφρασης.

IsBoolean

Επιστρέφει TRUE αν μια έκφραση είναι τιμή Boolean.

Σύνταξη:

```
Result = IsBoolean ( vExpression ) AS Boolean
Result = Boolean? ( vExpression ) AS Boolean
```

Result, η τιμή που επιστρέφει η συνάρτηση είναι τιμή boolean.
vExpression, το όρισμα της συνάρτησης είναι μια έκφραση.

Παράδειγμα:

```
PRINT IsBoolean(TRUE)
PRINT IsBoolean(FALSE)
PRINT IsBoolean("hello")
PRINT IsBoolean(1)
PRINT IsBoolean(-1)
```

Έξοδος:

```
True
True
False
False
False
```

IsDate

Επιστρέφει TRUE αν μια έκφραση είναι τιμή Date.

Σύνταξη:

```
Result = IsDate ( vExpression ) AS Boolean
Result = Date? ( vExpression ) AS Boolean
```

Result, η τιμή που επιστρέφει η συνάρτηση είναι τιμή boolean.
vExpression, το όρισμα της συνάρτησης είναι μια έκφραση.

Παράδειγμα:

```
PRINT IsDate(TRUE)
PRINT IsDate(FALSE)
PRINT IsDate("hello")
PRINT IsDate(1)
PRINT IsDate(-1)
PRINT IsDate(Now)
```

Έξοδος:

```
False
False
False
False
False
True
```

IsFloat

Επιστρέφει TRUE αν μια έκφραση είναι τιμή Float.

Σύνταξη:

```
Result = IsFloat ( vExpression ) AS Boolean
Result = Float? ( vExpression ) AS Boolean
```

Result, η τιμή που επιστρέφει η συνάρτηση είναι τιμή boolean.
vExpression, το όρισμα της συνάρτησης είναι μια έκφραση.

Παράδειγμα:

```
PRINT IsFloat(TRUE)
PRINT IsFloat(FALSE)
PRINT IsFloat("hello")
PRINT IsFloat(1)
PRINT IsFloat(-1)
PRINT IsFloat(Now)
PRINT IsFloat(Pi)
PRINT IsFloat(1.23)
```

Έξοδος:

```
False
False
False
False
False
False
True
True
```

IsInteger

Επιστρέφει TRUE αν μια έκφραση είναι τιμή Integer.

Σύνταξη:

```
Result = IsInteger ( vExpression ) AS Boolean
Result = Integer? ( vExpression ) AS Boolean
```

Result, η τιμή που επιστρέφει η συνάρτηση είναι τιμή boolean.
vExpression, το όρισμα της συνάρτησης είναι μια έκφραση.

Παράδειγμα:

```
PRINT IsInteger(TRUE)
PRINT IsInteger(FALSE)
PRINT IsInteger("hello")
PRINT IsInteger(1)
PRINT IsInteger(-1)
PRINT IsInteger(Now)
PRINT IsInteger(Pi)
PRINT IsInteger(1.23)
PRINT IsInteger(23)
```

Έξοδος:

```
False
False
False
True
True
False
False
False
True
```

IsLong

Επιστρέφει TRUE αν μια έκφραση είναι τιμή Long.

Σύνταξη:

```
Result = IsLong ( vExpression ) AS Boolean
Result = Long? ( vExpression ) AS Boolean
```

Result, η τιμή που επιστρέφει η συνάρτηση είναι τιμή boolean.
vExpression, το όρισμα της συνάρτησης είναι μια έκφραση.

Παράδειγμα:

```
PRINT IsLong(TRUE)
PRINT IsLong(FALSE)
PRINT IsLong("hello")
PRINT IsLong(1)
PRINT IsLong(-1)
PRINT IsLong(Now)
PRINT IsLong(Pi)
PRINT IsLong(1.23)
PRINT IsLong(-9223372036854775808)
```

Έξοδος:

```
False
False
False
False
False
False
False
False
True
```

IsNull

Επιστρέφει TRUE αν μια έκφραση είναι τιμή NULL.

Σύνταξη:

```
Result = IsNull ( vExpression ) AS Boolean
Result = Null? ( vExpression ) AS Boolean
```

Result, η τιμή που επιστρέφει η συνάρτηση είναι τιμή boolean.
vExpression, το όρισμα της συνάρτησης είναι μια έκφραση.

Παράδειγμα:

```
PRINT IsNull(TRUE)
PRINT IsNull(FALSE)
PRINT IsNull("hello")
PRINT IsNull(1)
PRINT IsNull(-1)
PRINT IsNull(Now)
PRINT IsNull(Pi)
PRINT IsNull(1.23)
PRINT IsNull(-9223372036854775808)
PRINT IsNull(NULL)
```

Έξοδος:

```
False
False
False
False
False
False
False
False
False
True
```

IsNumber

Επιστρέφει TRUE αν μια έκφραση είναι αριθμός.

Σύνταξη:

```
Result = IsNumber ( vExpression ) AS Boolean
Result = Number? ( vExpression ) AS Boolean
```

Result, η τιμή που επιστρέφει η συνάρτηση είναι τιμή boolean.
vExpression, το όρισμα της συνάρτησης είναι μια έκφραση.

Παράδειγμα:

```
PRINT IsNumber(TRUE)
PRINT IsNumber(FALSE)
PRINT IsNumber("hello")
PRINT IsNumber(1)
PRINT IsNumber(-1)
PRINT IsNumber(Now)
PRINT IsNumber(Pi)
PRINT IsNumber(1.23)
PRINT IsNumber(-9223372036854775808)
PRINT IsNumber(NULL)
```

Έξοδος:

```
True
True
False
True
True
False
True
True
True
False
```

SizeOf

Επιστρέφει το μέγεθος μνήμης που χρησιμοποιεί ένα τύπος δεδομένων.

Σύνταξη:

```
Size = SizeOf ( Datatype )
```

Ένας τύπος δεδομένων Datatype μπορεί να είναι μια σταθερά από τις παρακάτω:

Datatype	Σταθερά
Boolean	gb.Boolean
Byte	gb.Byte
Short	gb.Short
Integer	gb.Integer
Long	gb.Long
Single	gb.Single
Float	gb.Float
Date	gb.Date
String	gb.String
Variant	gb.Variant
Object	gb.Object
Pointer	gb.Pointer
Function	gb.Function
Class	gb.Class

Για να δείτε στο σύστημα μας το μέγεθος των τύπων δεδομένων μπορείτε να τρέξετε το Project29

Παράδειγμα:

```
1   ' Gambas module file
2
3   Public Sub Main()
4
5     Print SizeOf(gb.Boolean) & " bytes";;
6     Print SizeOf(gb.Byte) & " bytes";;
7     Print SizeOf(gb.Short) & " bytes";;
8     Print SizeOf(gb.Integer) & " bytes";;
9     Print SizeOf(gb.Long) & " bytes";;
10    Print SizeOf(gb.Single) & " bytes";;
11    Print SizeOf(gb.Float) & " bytes";;
12    Print SizeOf(gb.Date) & " bytes";;
13    Print SizeOf(gb.String) & " bytes";;
14    Print SizeOf(gb.Variant) & " bytes";;
15    Print SizeOf(gb.Object) & " bytes";;
16    Print SizeOf(gb.Pointer) & " bytes";;
17    Print SizeOf(gb.Function) & " bytes";;
18    Print SizeOf(gb.Class) & " bytes";;
19
20  End
```

Έξοδος:

```
1 bytes 1 bytes 2 bytes 4 bytes 8 bytes 4 bytes 8 bytes 8 bytes 8 bytes 16 bytes 8 bytes 8 bytes 0 bytes 0 bytes
```

TypeOf

Επιστρέφει τον τύπο δεδομένων μιας έκφρασης.

Σύνταξη:

```
Type = TypeOf ( Expression )
```

Type, η τιμή που επιστρέφει η συνάρτηση είναι ένας ακέραιος που αντιστοιχεί σε ένα τύπο δεδομένων. Για την αντιστοιχία των σταθερών με τιμές ακεραίων δείτε στις “Σταθερές”.
Expression, το όρισμα της συνάρτησης είναι μια έκφραση.

Παράδειγμα:

```
PRINT TypeOf(TRUE)
PRINT TypeOf(FALSE)
PRINT TypeOf("hello")
PRINT TypeOf(1)
PRINT TypeOf(-1)
PRINT TypeOf(Now)
PRINT TypeOf(Pi)
PRINT TypeOf(1.23)
PRINT TypeOf(CSingle(1.23))
PRINT TypeOf(-9223372036854775808)
PRINT TypeOf(NULL)
```

Έξοδος:

```
1
1
9
4
4
8
7
7
6
5
15
```

Συναρτήσεις ημερομηνίας και χρόνου (Date & Time Functions)

Οι συναρτήσεις αυτές είναι:

`Date`	Επιστρέφει μια τιμή ημερομηνίας χωρίς το μέρος της ώρας.
`DateAdd`	Προσθέτει μια περίοδο χρόνου σε μια ημερομηνία.
`DateDiff`	Επιστρέφει την περίοδο μεταξύ δύο ημερομηνιών.
`Day`	Επιστρέφει την ημέρα από μια ημερομηνία.
`Hour`	Επιστρέφει την ώρα από μια ημερομηνία.
`Minute`	Επιστρέφει τα λεπτά από μια ημερομηνία.
`Month`	Επιστρέφει τον μήνα από μια ημερομηνία.
`Now`	Επιστρέφει την παρούσα ημερομηνία και ώρα.
`Second`	Επιστρέφει τα δευτερόλεπτα από μια ημερομηνία.
`Time`	Επιστρέφει το μέρος του χρόνου από μια τιμή ημερομηνίας.
`Timer`	Επιστρέφει τον αριθμό των δευτερολέπτων από την εκκίνηση του προγράμματος.
`Week`	Επιστρέφει την εβδομάδα μιας ημερομηνίας.
`WeekDay`	Επιστρέφει την ημέρα της εβδομάδος από μια ημερομηνία.
`Year`	Επιστρέφει τον χρόνο από μια ημερομηνία.

Date

Επιστρέφει μια τιμή ημερομηνίας χωρίς το μέρος της ώρας.

Σύνταξη:

```
Result = Date ( Expression ) AS Date
```

Result, η τιμή που επιστρέφει η συνάρτηση.
Expression, το όρισμα της συνάρτησης είναι μια έκφραση ημερομηνίας.

Παράδειγμα:

```
PRINT Now
PRINT Date(Now)
```

Έξοδος:

```
11/24/2010 10:50:48
11/24/10
```

Σύνταξη:

```
Date = Date ( Year , Month , Day [ , Hours , Minutes , Seconds ] )
AS Date
```

Επιστρέφει μια τιμή ημερομηνίας με το μέρος της ώρας.

Date, η τιμή που επιστρέφει η συνάρτηση.
Year , Month , Day, Hours , Minutes , Seconds, τιμές ακεραίων που ορίζουν τα μέρη μιας ημερομηνίας.

Παράδειγμα:

```
PRINT Date(1962, 06, 07, 2, 5, 17)
```

Έξοδος:

```
07/06/1962 02:05:17
```

DateAdd

Προσθέτει μια περίοδο χρόνου σε μια ημερομηνία.

Σύνταξη:

```
Result = DateAdd ( Date as Date , Period AS Integer , Count AS
Integer ) AS Date
```

Result, η τιμή που επιστρέφει η συνάρτηση.

Date, η αρχική ημερομηνία.

Period, ο τύπος της περιόδου του χρόνου που θα προστεθεί. Είναι μια τιμή ακεραίου που αντιστοιχεί σε σταθερές χρόνου οι οποίες είναι: gb.second, gb.minute, gb.hour, gb.day, gb.week, gb.weekday, gb.month, gb.quarter, gb.year για περισσότερα δείτε το κεφάλαιο με τις σταθερές.

Count, το ποσό του χρόνου που θα προστεθεί.

Παράδειγμα:

```
PRINT Now
PRINT DateAdd(Now, gb.Minute, 10)
```

Έξοδος:

```
11/24/2010 11:19:01
11/24/2010 11:29:01
```

DateDiff

Επιστρέφει την περίοδο μεταξύ δύο ημερομηνιών.

Σύνταξη:

```
Count = DateDiff ( Date1 AS Date , Date2 AS Date , Period AS
Integer ) AS Integer
```

Count, η τιμή που επιστρέφει η συνάρτηση.

Date1, η πρώτη ημερομηνία.
Date2, η δεύτερη ημερομηνία.

Period, ο τύπος της περιόδου του χρόνου που θα υπολογιστεί. Είναι μια τιμή ακεραίου που αντιστοιχεί σε σταθερές χρόνου οι οποίες είναι: gb.second, gb.minute, gb.hour, gb.day, gb.week, gb.weekday, gb.month, gb.quarter, gb.year για περισσότερα δείτε το κεφάλαιο με τις σταθερές.

Παράδειγμα:

```
PRINT DateDiff("11/24/2010", "11/25/2010", gb.Hour)
```

Έξοδος:

```
24
```

Day

Επιστρέφει την ημέρα από μια ημερομηνία.

Σύνταξη:

```
Result = Day ( Date AS Date ) AS Integer
```

Result, η τιμή που επιστρέφει η συνάρτηση.
Date, η ημερομηνία.

Παράδειγμα:

```
PRINT Now
PRINT Day(Now)
```

Έξοδος:

```
11/24/2010 11:33:46
24
```

Hour

Επιστρέφει την ώρα από μια ημερομηνία.

Σύνταξη:

```
Result = Hour ( Date AS Date ) AS Integer
```

Result, η τιμή που επιστρέφει η συνάρτηση.
Date, η ημερομηνία.

Παράδειγμα:

```
PRINT Now
PRINT Hour(Now)
```

Έξοδος:

```
11/24/2010 11:33:46
11
```

Minute

Επιστρέφει τα λεπτά από μια ημερομηνία.

Σύνταξη:

```
Result = Minute ( Date AS Date ) AS Integer
```

Result, η τιμή που επιστρέφει η συνάρτηση.
Date, η ημερομηνία.

Παράδειγμα:

```
PRINT Now
PRINT Minute(Now)
```

Έξοδος:

```
11/24/2010 11:33:46
35
```

Month

Επιστρέφει τον μήνα από μια ημερομηνία.

Σύνταξη:

```
Result = Month ( Date AS Date ) AS Integer
```

Result, η τιμή που επιστρέφει η συνάρτηση.
Date, η ημερομηνία.

Παράδειγμα:

```
PRINT Now
PRINT Month(Now)
```

Έξοδος:

```
11/24/2010 11:33:46
11
```

Now

Επιστρέφει την παρούσα ημερομηνία και ώρα.

Σύνταξη:

```
Result = Now [ ( ) ] AS Date
```

Result, η τιμή που επιστρέφει η συνάρτηση.

Παράδειγμα:

```
PRINT Now
```

Έξοδος:

```
11/24/2010 11:33:46
```

Second

Επιστρέφει τα δευτερόλεπτα από μια ημερομηνία.

Σύνταξη:

```
Result = Second ( Date AS Date ) AS Integer
```

Result, η τιμή που επιστρέφει η συνάρτηση.
Date, η ημερομηνία.

Παράδειγμα:

```
PRINT Now
PRINT Second(Now)
```

Έξοδος:

```
11/24/2010 11:33:46
46
```

Time

Επιστρέφει το μέρος του χρόνου από μια τιμή ημερομηνίας.

Σύνταξη:

```
Result = Time ( Date AS Date)
```

Result, η τιμή που επιστρέφει η συνάρτηση.
Date, η ημερομηνία.

Παράδειγμα:

```
PRINT Now
PRINT Time(Now)
```

Έξοδος:

```
11/24/2010 11:33:46
11:33:46
```

Timer

Επιστρέφει τον αριθμό των δευτερολέπτων από την εκκίνηση του προγράμματος.

Σύνταξη:

```
Result = Timer [ ( ) ]
```

Result, η τιμή που επιστρέφει η συνάρτηση.

Παράδειγμα:

```
PRINT Timer
```

Έξοδος:

```
0.132802009583
```

Week

Επιστρέφει την εβδομάδα μιας ημερομηνίας.

Σύνταξη:

```
Result = Week ( Date AS Date [ , FirstDayOfWeek AS Integer ,
FullWeek AS Boolean ] )
```

Result, η τιμή που επιστρέφει η συνάρτηση.
Date, η ημερομηνία από την οποία θα υπολογιστεί η εβδομάδα
FirstDayOfWeek, ακέραιος που δείχνει την πρώτη μέρα της εβδομάδος. Δείτε το κεφάλαιο με τις σταθερές.
FullWeek, Boolean που καθορίζει αν είναι TRUE ότι μόνο ολόκληρες εβδομάδες θα υπολογίζονται. Αν είναι FALSE τότε οι πρώτες εβδομάδες που δεν είναι ολόκληρες υπολογίζονται από την συνάρτηση αν έχουν 4 ημέρες και άνω.

Παράδειγμα:

```
PRINT Now
PRINT Week(Now)
```

Έξοδος:

```
11/24/2010 11:47:54
47
```

WeekDay

Επιστρέφει την ημέρα της εβδομάδος από μια ημερομηνία.

Σύνταξη:

```
Result = WeekDay ( Date AS Date ) AS Integer
```

Result, η τιμή που επιστρέφει η συνάρτηση.
Date, η ημερομηνία.

Παράδειγμα:

```
PRINT Now
PRINT WeekDay(Now)
```

Έξοδος:

```
11/24/2010 11:49:42
3
```

Year

Επιστρέφει τον χρόνο από μια ημερομηνία.

Σύνταξη:

```
Result = Year ( Date AS Date ) AS Integer
```

Result, η τιμή που επιστρέφει η συνάρτηση.
Date, η ημερομηνία.

Παράδειγμα:

```
PRINT Now
PRINT Year(Now)
```

Έξοδος:

```
11/24/2010 11:50:46
2010
```

Συναρτήσεις αρχείων και καταλόγων (File & Directory Functions)

Οι συναρτήσεις αυτές διαχειρίζονται αρχεία και καταλόγους.

Αυτές οι συναρτήσεις είναι:

Access	Ελέγχει τον τύπο πρόσβασης σε ένα αρχείο.
COPY	Αντιγράφει ένα αρχείο.
DFree	Επιστρέφει τον ελεύθερο χώρο μιας συσκευής.
Dir	Περιηγείται σε έναν κατάλογο.
Exist	Ελέγχει αν υπάρχει ένας κατάλογος ή αρχείο.
IsDir	Ελέγχει αν μία διαδρομή είναι κατάλογος.
KILL	Διαγράφει ένα αρχείο.
LINK	Δημιουργεί ένα συμβολικό δεσμό.
MKDIR	Δημιουργεί ένα κατάλογο.
MOVE	Μετονομάζει ένα αρχείο ή κατάλογο.
RDir	Περιηγείται έναν κατάλογο κατ' επανάληψη.
RMDIR	Διαγράφει έναν άδειο κατάλογο.
Stat	Επιστρέφει πληροφορίες για ένα αρχείο.
Temp$	Δημιουργεί προσωρινά ονόματα αρχείων.

Access

Ελέγχει τον τύπο πρόσβασης σε ένα αρχείο.

Σύνταξη:

```
Accessible = Access ( Path AS String [ , Mode AS Integer ] ) AS
Boolean
```

Accessible, η τιμή επιστροφής της συνάρτησης είναι ένας Boolean, δλδ TRUE ή FALSE
Path, η τιμή είναι ένα string που δηλώνει την διαδρομή του δίσκου προς έλεγχο
Mode, η τιμή είναι ένας integer, που δηλώνει ως προς ποιο στοιχείο γίνεται ο έλεγχος.

Η τιμή Mode, είναι μια από τις ακόλουθες:

- gb.Read, τιμή 4.
- gb.Write, τιμή 2.

- gb.Exec, τιμή 1.

Για να γίνει έλεγχος σε συνδυασμό τιμών Mode, γράφουμε τις σταθερές με τον τελεστή OR.

Παράδειγμα:

```
PRINT Access("/home/user", gb.Read)
PRINT Access("/home/user", gb.Read OR gb.Write)
```

Έξοδος:

```
True
True
```

COPY

Αντιγράφει ένα αρχείο.

Σύνταξη:

```
COPY Source path TO Destination path
```

Source path, η τιμή της διαδρομής που θα αντιγραφεί το αρχείο.
Destination path, η τιμή της διαδρομής που θα καταλήξει το αρχείο.

Παράδειγμα:

```
COPY "/home/user/file1" TO "/home/user/folder2/file1"
```

DFree

Επιστρέφει τον ελεύθερο χώρο μιας συσκευής.

Σύνταξη:

```
Size = DFree ( Path AS String ) AS Long
```

Size, η τιμή επιστροφής της συνάρτησης είναι μονάδες byte.
Path, η τιμή είναι ένα string που δηλώνει την διαδρομή του δίσκου προς έλεγχο

Παράδειγμα:

```
PRINT Dfree("/home/user/")
```

Έξοδος:

```
18090684416
```

Dir

Περιηγείται σε έναν κατάλογο.

Σύνταξη:

```
File name array = Dir ( Directory AS String [ , Pattern AS String
, Filter AS Integer ] ) AS String[]
```

File name array, η τιμή επιστροφής της συνάρτησης είναι ένας πίνακας από strings.
Directory, η τιμή είναι ένα string που δηλώνει την διαδρομή του δίσκου προς περιήγηση.
Pattern, είναι η μορφή των ονομάτων των αρχείων που θα επιστραφούν. Μπορεί να αποτελείται από γενικούς αλφαριθμητικούς χαρακτήρες εκτός από την λέξη LIKE.
Αν δεν οριστεί μορφή όλα τα ονόματα των αρχείων επιστρέφονται.
Filter, ακέραιος που αντιστοιχεί σε μια σταθερά αρχείων και καταλόγων όπως gb.File gb.Directory, gb.Device, gb.Pipe, gb.Socket, gb.Link.

Παράδειγμα:

```
1 DIM sFiles AS String[]
2 DIM File AS String
3
4 sFiles = Dir("/home/user", "*", gb.File)
5
6 FOR EACH File IN sFiles
7   PRINT File;;
8 NEXT
```

Έξοδος:

```
.gksu.lock .dmrc .libquicktime_codecs .ICEauthority .xsession-
errors.old .lmmsrc.xml .gpilotd.pid .x11vnc.log.ubuntu-
desktop:5900 .zynaddsubfxXML.cfg .xsession-errors examples.desktop .printer-
groups.xml .profile .recently-
used.xbel.OWGMHV .Wammu .sudo_as_admin_successful .hyperconf .bash_logo
ut .htoprc .x11vncrc .pulse-cookie .odbc.ini .gtk-bookmarks .recently-
used.xbel .bash_history .xscreensaver-
getimage.cache .bashrc .esd_auth .openme.prefs .recently-
used.xbel.8T69GV .x11vnc.log.ubuntu-desktop:5905 .nvi
dia-settings-rc .dvdstyler
```

Ανάλυση:
Στην γραμμή 1 δηλώνεται ένας πίνακας string.
Στην γραμμή 2 δηλώνεται μια μεταβλητή string.
Στην γραμμή 4 εκτελούμε την συνάρτηση Dir() για την διαδρομή “/home/user”, για όλα τα αρχεία “*”, που είναι τύπου αρχείου “gb.File”.

Στην γραμμή 6 τελούμε έναν βρόχο επανάληψης FOR EACH και τυπώνουμε κάθε τιμή από τον πίνακα sFiles.

Exist

Ελέγχει αν υπάρχει ένας κατάλογος ή αρχείο.

Σύνταξη:

```
Boolean = Exist ( Path As String, Followlink As Boolean )
```

Boolean, η τιμή επιστροφής της συνάρτησης είναι ένας boolean.
Path, η τιμή είναι ένα string που δηλώνει την διαδρομή του δίσκου προς έλεγχο
Followlink, η τιμή ενός boolean που δηλώνει αν η διαδρομή δείχνει σε έναν σύνδεσμο.

Παράδειγμα:

```
PRINT Exist("/usr/bin")
```

Έξοδος:

```
True
```

IsDir

Ελέγχει αν μία διαδρομή είναι κατάλογος.

Σύνταξη:

```
bResult = IsDir ( sPath AS String ) AS Boolean

bResult = Dir? ( sPath AS String ) AS Boolean
```

bResult, η τιμή επιστροφής της συνάρτησης είναι ένας boolean.
sPath, η τιμή είναι ένα string που δηλώνει την διαδρομή του δίσκου προς έλεγχο

Παράδειγμα:

```
PRINT IsDir("/usr/bin")
PRINT IsDir("/usr/bin/ls")
```

Έξοδος:

```
True
False
```

KILL

Διαγράφει ένα αρχείο.

Σύνταξη:

```
KILL sPath
```

sPath, η τιμή είναι ένα string που δηλώνει την διαδρομή του δίσκου προς διαγραφή.

Παράδειγμα:

```
KILL "/folder1/folder2"
```

LINK

Δημιουργεί ένα συμβολικό δεσμό.

Σύνταξη:

```
LINK sTarget AS String TO sPath AS String
```

sTarget, η τιμή του συμβολικού δεσμού που θα δημιουργηθεί.
sPath, η τιμή είναι ένα string που δηλώνει την διαδρομή του δίσκου που θα δείχνει ο δεσμός.

Παράδειγμα:

```
LINK “/home/user/mylink” TO “/usr/bin/ls”
```

MKDIR

Δημιουργεί ένα κατάλογο.

Σύνταξη:

```
MKDIR sPath AS String
```

sPath, η τιμή είναι ένα string που δηλώνει τον φάκελο που θα δημιουργηθεί.
Αν κάποιος γονικός φάκελος λείπει η εντολή αποτυγχάνει να εκτελεστεί με επιτυχία.

Παράδειγμα:

```
MKDIR "/home/user/my_directory"
```

Η συνάρτηση επιστρέφει τα εξής λάθη:

Μήνυμα	Περιγραφή
Access forbidden (#43)	Ο γονικός κατάλογος δεν επιτρέπει πρόσβαση εγγραφής, ή ένας από τους καταλόγους στην διαδρομή δεν επιτρέπει την πρόσβαση περιήγησης.
File or directory does not exist (#45)	Ένα μέρος της διαδρομής δεν υπάρχει ή είναι συμβολικός δεσμός.
Not a directory (#49)	Ένα μέρος που χρησιμοποιείται ως κατάλογος στην διαδρομή δεν είναι κατάλογος.
Out of memory (#1)	Μη επαρκή μνήμη.
File name is too long (#44)	Η διαδρομή είναι πολύ μακρυά.
Device is full (#37)	Η συσκευή που περιέχει την διαδρομή δεν έχει άλλο χώρο.
File already exists (#38)	Η διαδρομή sPath υπάρχει ήδη. Αυτό συμπεριλαμβάνει και την περίπτωση να είναι συμβολικός δεσμός.
System error... (#42)	Άλλα πιθανά προβλήματα συστήματος: • Πάρα πολλοί συμβολικοί δεσμοί. • Το σύστημα αρχείων δεν υποστηρίζει την δημιουργία καταλόγων. • Η διαδρομή sPath αναφέρεται σε αρχείο ή σε σύστημα αρχείων μόνο προς ανάγνωση.

MOVE

Μετονομάζει ένα αρχείο ή κατάλογο.

Σύνταξη:

```
MOVE sOldName AS String TO sNewName AS String
```

sOldName, η τιμή είναι ένα string που δηλώνει το αρχείο που θα μετακινηθεί ή θα μετονομαστεί.
SnewName, η τιμή είναι ένα string που δηλώνει το όνομα του νέου αρχείου.

Παράδειγμα:

```
MOVE "/home/user/my_directory1" TO "/home/user/my_directory2"
```

RDir

Περιηγείται έναν κατάλογο κατ' επανάληψη.

Σύνταξη:

```
File name array = RDir ( Directory AS String [ , Pattern AS String
, Filter AS Integer ] )
```

File name array, η τιμή επιστροφής της συνάρτησης είναι ένας πίνακας από strings.
Directory, η τιμή είναι ένα string που δηλώνει την διαδρομή του δίσκου προς περιήγηση.
Pattern, είναι η μορφή των ονομάτων των αρχείων που θα επιστραφούν. Μπορεί να αποτελείται από γενικούς αλφαριθμητικούς χαρακτήρες εκτός από την λέξη LIKE.
Αν δεν οριστεί μορφή όλα τα ονόματα των αρχείων επιστρέφονται.
Filter, ακέραιος που αντιστοιχεί σε μια σταθερά αρχείων και καταλόγων όπως gb.File gb.Directory, gb.Device, gb.Pipe, gb.Socket, gb.Link.

Παράδειγμα:

```
1 DIM sFiles AS String[]
2 DIM File AS String
3
4 sFiles = RDir("/home/user", "*", gb.File)
5
6 FOR EACH File IN sFiles
7   PRINT File;;
8 NEXT
```

Έξοδος:

```
.gksu.lock .dmrc .libquicktime_codecs .ICEauthority .xsession-
errors.old .lmmsrc.xml .gpilotd.pid .x11vnc.log.ubuntu-
desktop:5900 .zynaddsubfxXML.cfg .xsession-errors examples.desktop .....
```

Ανάλυση:
Στην γραμμή 1 δηλώνεται ένας πίνακας string.
Στην γραμμή 2 δηλώνεται μια μεταβλητή string.
Στην γραμμή 4 εκτελούμε την συνάρτηση RDir() για την διαδρομή “/home/user”, για όλα τα αρχεία “*”, που είναι τύπου αρχείου “gb.File”.

Στην γραμμή 6 τελούμε έναν βρόχο επανάληψης FOR EACH και τυπώνουμε κάθε τιμή από τον πίνακα sFiles.

RMDIR

Διαγράφει έναν άδειο κατάλογο.

Σύνταξη:

```
RMDIR Path
```

Path, μια τιμή string της διαδρομής που θα διαγραφεί.

Παράδειγμα:

```
RMDIR “/home/user/my_directory1”
```

Stat

Επιστρέφει πληροφορίες για ένα αρχείο.

Σύνταξη:

```
File info = Stat ( Path [ , FollowLink ] )
```

File info, αναφορά σε ένα αντικείμενο Stat.
Path, η διαδρομή για την οποία θα επιστραφούν πληροφορίες στο Stat αντικείμενο.
FollowLink, boolean που όταν είναι TRUE οι συμβολικοί δεσμοί ακολουθούνται.

Παράδειγμα:

```
DIM sInfo AS Object

sInfo = Stat("/home")

PRINT sInfo.Group
PRINT sInfo.Hidden
PRINT sInfo.LastAccess
PRINT sInfo.LastChange
PRINT sInfo.Type
```

Έξοδος:

```
root
False
11/24/2010 18:39:19
10/04/2010 17:02:30
2
```

Για πλήρη περιγραφή του αντικειμένου Stat δείτε στα αντικείμενα την κλάση gb.Stat

Temp$

Δημιουργεί προσωρινά ονόματα αρχείων.

Σύνταξη:

```
File name = Temp$( [ Name ] )

File name = Temp( [ Name ] )
```

File name, η διαδρομή του προσωρινού αρχείου.
Name, το όνομα του αρχείου.

Η συνάρτηση δημιουργεί ονόματα διαδρομής του τύπου:
/tmp/gambas.**[User id]**/**[Process id]**/**[Name]**.tmp

όπου:
User_id, το id του χρήστη
Process_id, το id της διεργασίας
Name, το όνομα που δίνεται στην συνάρτηση.

Αν δεν δοθεί όνομα τότε δημιουργείται ένα αρχείο με όνομα έναν ακέραιο που αυξάνει σε τιμή σε κάθε κλήση της συνάρτησης. Έτσι μπορείτε να έχετε πρόσβαση σε μοναδικά ονόματα αρχείων.

Παράδειγμα:

```
PRINT Temp$()
```

Έξοδος:

```
/tmp/gambas.1000/4156/1.tmp
```

Συναρτήσεις μορφοποίησης (Formatting functions)

Οι συναρτήσεις αυτές μορφοποιούν έναν δυαδικό αριθμό, μια ημερομηνία, έναν αριθμό ή έναν δεκαεξαδικό αριθμό.

Αυτές οι συναρτήσεις είναι:

`Bin$`	Μορφοποιεί έναν δυαδικό αριθμό.
`Format$`	Μορφοποιεί μια ημερομηνία ή έναν αριθμό.
`Hex$`	Μορφοποιεί έναν δεκαεξαδικό αριθμό.

Bin$

Μορφοποιεί έναν δυαδικό αριθμό.

Σύνταξη:

```
String = Bin$ ( Number AS Long [ , Digits AS Integer ] ) AS String
```

String, η επιστροφή της συνάρτησης είναι ένα string.
Number, ο δεκαδικός αριθμός που θα μορφοποιηθεί.
Digits, το πλήθος των ψηφίων που θα εμφανίζονται στην τελική μορφή του αριθμού.

Παράδειγμα:

```
PRINT Bin(55)
PRINT Bin(55, 64)
```

Έξοδος:

```
110111
0000000000000000000000000000000000000000000000000000000000000110111
```

Format$

Μορφοποιεί μια ημερομηνία ή έναν αριθμό.

Σύνταξη:

```
String = Format$ ( Expression [ , Format ] )

String = Format ( Expression [ , Format ] )
```

String, η επιστροφή της συνάρτησης είναι ένα string.
Expression, η έκφραση που θα μορφοποιηθεί, μπορεί να είναι ένας αριθμός ή μια ημερομηνία.
Format, η μορφή που θα έχει το αποτέλεσμα.

Η συνάρτηση μορφοποιεί μια έκφραση σε string ανάλογα τον τύπο της εκφράσεως.
Η συνάρτηση μπορεί να μορφοποιεί ημερομηνίες, τιμές χρόνου, αριθμούς, και νομισματικές μονάδες.

Η μορφή Format μπορεί να είναι μια προκαθορισμένη σταθερά ως ένας integer (δείτε το κεφάλαιο των σταθερών), ή μια μορφή ορισμένη από τον χρήστη το οποίο θα είναι ένα string.
Εάν δεν οριστεί Format τότε χρησιμοποιήται η σταθερά gb.Standard.

Η συνάρτηση χρησιμοποιεί πληροφορίες των τοπικών ρυθμίσεων του συστήματος (localization).

Μορφές ορισμένες από τον χρήστη
Μια μορφή ορισμένη από τον χρήστη είναι ένα string με ειδικούς χαρακτήρες και χαρακτήρες μη ειδικούς.
Οι μη ειδικοί χαρακτήρες τυπώνονται όπως εμφανίζονται.
Οι ειδικοί χαρακτήρες τελούν μια μορφοποίηση στην έκφραση Expression που δίνεται σαν όρισμα στην συνάρτηση.

Ειδικοί χαρακτήρες για την μορφοποίηση ορισμένη από τον χρήστη για αριθμούς.

+	Τυπώνει το πρόσημο του αριθμού.
-	Τυπώνει το πρόσημο του αριθμού μόνον όταν είναι αρνητικός.
#	Τυπώνει ένα ψηφίο μόνον όταν είναι απαραίτητο.
0	Πάντα τυπώνει ένα ψηφίο συμπληρώνοντας με το 0 όταν χρειαστεί.
.	Τυπώνει την υποδιαστολή.
,	Τυπώνει το διαχωριστικό της χιλιάδας.
%	Πολλαπλασιάζει τον αριθμό με το 100 και εμφανίζει το σύμβολο επί της εκατό (%).
E	Εισάγει το εκθετικό μέρος ένός Float αριθμού. Το πρόσημο του εκθετικού αριθμού πάντα τυπώνεται.

Παράδειγμα:

```
PRINT Format$(Pi, "#.###")
PRINT Format$(Pi, "+#.###")
PRINT Format$(324, "### %")
PRINT Format$(Pi ^ Pi, "#.#E##")
```

Έξοδος:

```
3.142
+3.142
32400 %
3.6E+1
```

Ειδικοί χαρακτήρες για την μορφοποίηση ορισμένη από τον χρήστη για νομισματικές μονάδες.

$	Τυπώνει το σύμβολο του εθνικού νομίσματος.
$$	Όταν το $ είναι διπλό, τυπώνεται το διεθνές σύμβολο νομίσματος.
(	Τυπώνει την αρνητική μορφή νομισματικών μονάδων. Αυτός ο χαρακτήρας πρέπει να είναι ο πρώτος στην μορφή ειδικών χαρακτήρων. Μπορείτε να συμπληρώσετε μια παρένθεση κλεισίματος στο τέλος της μορφής.

Παράδειγμα:

```
PRINT Format$(1956.06, "$#.###")
PRINT Format$(1956.06, "$$#.###")
PRINT Format$(1956.06, "($#.###)")
```

Έξοδος:

```
1956.06 €
USD 1956.06
1956.06 €
```

Ειδικοί χαρακτήρες για την μορφοποίηση ορισμένη από τον χρήστη για ημερομηνίες.

yy	Τυπώνει το έτος με δύο ψηφία.
yyyy	Τυπώνει το έτος με τέσσερα ψηφία.
m	Τυπώνει τον μήνα.
mm	Τυπώνει τον μήνα με δύο ψηφία.
mmm	Τυπώνει τον μήνα στην σύντομη μορφή του.
mmmm	Τυπώνει τον μήνα στην πλήρη μορφή του.
d	Τυπώνει την ημέρα.
dd	Τυπώνει την ημέρα με δύο ψηφία.
ddd	Τυπώνει την ημέρα της εβδομάδος στην σύντομη μορφή της.
dddd	Τυπώνει την ημέρα της εβδομάδος στην πλήρη μορφή της.
/	Τυπώνει το διαχωριστικό των ημερών.
h	Τυπώνει την ώρα.
hh	Τυπώνει την ώρα με δύο ψηφία.
n	Τυπώνει τα λεπτά.
nn	Τυπώνει τα λεπτά με δύο ψηφία.
s	Τυπώνει τα δευτερόλεπτα.

ss	Τυπώνει τα δευτερόλπετα με δύο ψηφία.
:	Τυπώνει το διαχωριστικό της ώρας.
u	Τυπώνει τα χιλιοστά δευτερολέπτου miliseconds, εάν διαφέρουν από το μηδέν.

Παράδειγμα:

```
PRINT Format$(Now, "mm/dd/yyyy hh:nn:ssu")
```

Έξοδος:

```
11/25/2010 10:38:22.086
```

Hex$

Μορφοποιεί έναν δεκαεξαδικό αριθμό.

Σύνταξη:

```
String = Hex$ ( Number AS Long [ , Digits AS Integer ] )
```

String, η επιστροφή της συνάρτησης είναι ένα string.
Number, ο δεκαδικός αριθμός που θα μορφοποιηθεί.
Digits, το πλήθος των ψηφίων που θα εμφανίζονται στην τελική μορφή του αριθμού.

Παράδειγμα:

```
PRINT Hex$(55)
PRINT Hex$(55, 8)
```

Έξοδος:

```
37
00000037
```

Το πρόσημο ενός integer και ενός short θα επεκταθεί διότι αυτοί οι τύποι δεδομένων μετατρέπονται σε Long.

Παράδειγμα:

```
PRINT Hex$(-32768)
```

Έξοδος:

```
FFFFFFFFFFFF8000
```

Αν η δεκαεξαδική μορφή πρέπει να περιοριστεί σε 16 bits ο τελεστής AND πρέπει να χρησιμοποιηθεί.

Παράδειγμα:

```
PRINT Hex$(CLong(-336) AND &hffff&)
```

Έξοδος:

```
FEB0
```

Συναρτήσεις μετάφρασης και τοπικής γλώσσας (Localization and Translation Functions)

Οι συναρτήσεις αυτές είναι χρήσιμες για μεταφράσεις και εργασίες για υποστήριξη ξένων γλωσσών στην εφαρμογή σας.

`Format$`	Μορφοποιεί έναν αριθμό ή μια ημερομηνία.
`Str$`	Μετατρέπει έναν αριθμό ή ημερομηνία σε κείμενο.
`Subst$`	Αντικαθιστά κείμενο βάσει μιας μορφής pattern.
`Tr$`	Μεταφράζει ένα κείμενο
`Val`	Μετατρέπει ένα κείμενο σε αριθμό ή ημερομηνία.

Η συνάρτηση Format$ έχει ήδη περιγραφεί στις συναρτήσεις μορφοποίησης.

Η συνάρτηση Str$ έχει ήδη περιγραφεί στις συναρτήσεις μετατροπής.

Η συνάρτηση Subst$ περιγράφεται στην συνέχεια στις συναρτήσεις συμβολοσειρών String.

Η συνάρτηση Val έχει ήδη περιγραφεί στις συναρτήσεις μετατροπής.

Tr$

Επιστρέφει την μετάφραση από το κείμενο String.

Σύνταξη:

```
Translation = Tr$ ( String  )

Translation = Tr ( String  )
```

Translation, το String που θα επιστραφεί.
String, το String προς μετάφραση.

Εάν δεν βρεθεί μετάφραση το ίδιο το κείμενο επιστρέφεται.
Η σύνταξη αυτή είναι η δυναμική έκφραση της σύνταξης `("...")`

Περισσότερα για την μετάφραση ενός έργου

Σε ένα σύγχρονο περιβάλλον ανάπτυξης εφαρμογών πρέπει να υπάρχει η δυνατότητα της μετάφρασης του έργου και σε άλλες γλώσσες χωρίς την ανάγκη δημιουργίας νέων εκδόσεων έργων για κάθε γλώσσα ξεχωριστά.
Η Gambas3 δίνει αυτή την δυνατότητα. Με την Gambas3 μπορείτε να γράψετε μια εφαρμογή και να ενσωματώσετε πολλές γλώσσες και τοπικές ρυθμίσεις κάθε περιοχής στο ίδιο έργο. Έτσι οι χρήστες του προγράμματος μπορούν έχουν την εφαρμογή σας με τις δικές τους τοπικές ρυθμίσεις. Στην ίδια εφαρμογή ανάλογα τις τοπικές ρυθμίσεις του λειτουργικού ένας Άγγλος θα βλέπει την Αγγλική γλώσσα, ένας Ιταλός την Ιταλική και ένας Έλληνας την Ελληνική, κτλ.

Οι μεταβλητές γλώσσας του περιβάλλοντος του λειτουργικού συστήματος.

Η συμπεριφορά ενός πολυγλωσσικού προγράμματος πρέπει να συσχετιστεί με τις μεταβλητές του συστήματος οι οποίες καθορίζουν τις τοπικές ρυθμίσεις της γλώσσας σύμφωνα με:

- Την χώρα του χρήστη του συστήματος.
- Την γλώσσα του χρήστη του συστήματος.

Στην Gambas, αυτές οι δύο πληροφορίες εξάγονται από τις μεταβλητές του λειτουργικού συστήματος: `$LC_ALL`, ή `$LANG`.

Η `$LANG` χρησιμοποιείται μόνο όταν η `$LC_ALL` δεν έχει οριστεί.

Η σύνταξη της `$LANG` είναι η ακόλουθη:

`xx_YY.ZZZZ`

- `xx` είναι ένα string δύο χαρακτήρων σε γραφή μικρών γραμμάτων και αντιπροσωπεύει την γλώσσα.
- `YY` είναι ένα string δύο χαρακτήρων σε γραφή κεφαλαίων γραμμάτων και αντιπροσωπεύει την χώρα.
- `ZZZZ` είναι το σετ χαρακτήρων του συστήματος.

Για να δείτε τις τιμές αυτών των μεταβλητών στο σύστημα σας πρέπει να γράψετε στο τερματικό την εξής εντολή.

```
$ echo $LC_ALL && echo $LANG
```

Στο σύστημα Ubuntu 10.04 έχει καθοριστεί μόνο η μεταβλητή $LANG.

Έτσι αν γράψετε την εντολή

```
$ echo $LANG
```

Θα δείτε την τιμή για παράδειγμα en_US.utf8. Η τιμή αυτή σημαίνει ότι έχετε σαν γλώσσα την Αγγλική και τοπικές ρυθμίσεις υποδιαστολής, νομίσματος, ώρας κτλ της χώρας της Αμερικής. Το δε σετ χαρακτήρων είναι το utf8.

Για να δείτε ποιες ρυθμίσεις γλώσσας και χώρας είναι διαθέσιμες στο σύστημα σας μπορείτε να γράψετε στο τερματικό την εντολή:

```
$ locale -a
```

Στο δικό μου σύστημα Ubuntu 10.04 LTS η παραπάνω εντολή δίνει τα εξής:

```
C
el_CY.utf8
el_GR.utf8
en_AG
en_AU.utf8
en_BW.utf8
en_CA.utf8
en_DK.utf8
en_GB.utf8
en_HK.utf8
en_IE.utf8
en_IN
en_NG
en_NZ.utf8
en_PH.utf8
en_SG.utf8
en_US.utf8
en_ZA.utf8
en_ZW.utf8
POSIX
zh_CN.utf8
zh_HK.utf8
zh_SG.utf8
zh_TW.utf8
```

Για να τρέξετε ένα πρόγραμμα με διαφορετικές τοπικές ρυθμίσεις θα πρέπει να γράψετε στο τερματικό τα εξής για την Ελληνική γλώσσα:

```
$ cd /path/to/my/project
$ LANG=el_GR gbx2
```

Σε μερικά συτήματα όπως το Madriva πρέπει να ορίσετε και την μεταβλητή `$LANG` και την μεταβλητή `$LANGUAGE`.

Για παράδειγμα:

```
$ cd /path/to/my/project
$ LANGUAGE=el_GR LANG=fr_FR gbx2
```

Μεταφράσεις String

Σε ένα έργο της Gambas3 υπάρχουν δύο είδη strings:

- Strings τα οποία βλέπει ο χρήστης (ένα κείμενο label, ένα κείμενο σε ένα πλαίσιο διαλόγου...), και τα οποία πρέπει να μεταφραστούν.
- Strings τα οποία είναι εσωτερικά (μία collection key, ένα πεδίο ονόματος, μια ρύθμιση του προγράμματος...), και τα οποία δεν θα πρέπει να μεταφραστούν.

Πρέπει λοιπόν να δηλώσετε στην Gambas ποια strings πρέπει να μεταφραστούν και ποια όχι.

Τα strings τα οποία δεν είναι προς μετάφραση εσωκλείονται σε διπλά εισαγωγικά.

```
PRINT "A string that must not be translated"
```

Τα strings τα οποία είναι προς μετάφραση εσωκλείονται σε παρενθέσεις πριν τα εισαγωγικά.

```
PRINT ("A string that must be translated")
```

Όλα τα στοιχεία ελέγχου control που έχουν ιδιότητες κειμένου μπορούν να μεταφραστούν.

Η Gambas χρησιμοποιεί το σύτημα μετάφρασης GNU: τα μεταφράσιμα strings αποθηκεύονται σε αρχεία `xx_YY.mo`, όπου `xx` και `YY` έχουν τον ίδιο ορισμό όπως πιο πάνω.

Διαχείριση των String

Η Gambas έχει δύο σετ συναρτήσεων που χειρίζονται χαρακτήρες, συναρτήσεις για χαρακτήρες ASCII μόνο, και συναρτήσεις που χειρίζονται συμβολοσειρές UTF-8.

Οι μεταφράσιμες συμβολοσειρές πρέπει να διαχειρίζονται από τις συναρτήσεις διαχείρισης UTF-8 χαρακτήρων. Αυτές οι συναρτήσεις είναι όλες μέλη της στατικής κλάσης String.

Έτσι η κλήση τους γίνεται με τον τρόπο

String.function()

Σε κάθε άλλη περίπτωση προτείνεται να χρησιμοποιείτε συναρτήσεις διαχείρισης χαρακτήρων ASCII επειδή είναι γρηγορότερες και εγγενείς.

Αυτές οι συναρτήσεις είναι οι: Asc, Chr$, InStr, RInStr, LCase$, Left$, Right$, Mid$, Len, LTrim$, RTrim$, Trim$... Δείτε το κεφάλαιο “Συναρτήσεις String”.

Η κλήση των εγγενών συναρτήσεων γίνεται απευθείας με το όνομα της συνάρτησης, για

παράδειγμα Asc, Chr$ κλτ

Μετατροπές String

Συχνά χρειάζετε να μεταφράσετε έναν αριθμό ή ημερομηνία σε string και το αντίθετο.

Στην συνέχεια πρέπει να γνωρίζετε αν το string που αντιπροσωπεύει την αριθμητική τιμή ή ημερομηνία θα είναι προσβάσιμο από τον χρήστη. Σε αυτή την περίπτωση το κείμενο θα πρέπει να μεταφραστεί από συναρτήσεις που χρησιμοποιούν τις τοπικές ρυθμίσεις γλωσσών.

Αυτές οι συναρτήσεις είναι οι: Str$, και Val.

Η Str$() μετατρέπει οποιαδήποτε τιμή σε ένα string με τοπικές ρυθμίσεις, ενώ η Val() μετατρέπει ένα string σε μια τιμή με το να μαντεύει τον τύπο δεδομένων της τιμής.

Αν η απεικόνιση του string δεν χρειάζεται να παρουσιαστεί στον χρήστη (για παράδειγμα μια πληροφορία για την τιμή σε ένα αρχείο) τότε καλό θα ήταν να χρησιμοποιήσετε συναρτήσεις που δεν λαμβάνουν υπόψιν τις τοπικές ρυθμίσεις. Τέτοιες είναι: CBool, CByte, CDate, CFloat, CInt, CLong, CSng, CShort, CStr.

Στην πραγματικότητα αυτές οι συναρτήσεις χρησιμοποιούν μια σταθερή ρύθμιση γλώσσας που ονομάζεται "C", η οποία είναι ακριβώς ίδια με την ρύθμιση "en_US",

Τοπικές ρυθμίσεις για το GUI

Μια δυσκολία προκύπτει για γλώσσες με γραφή από τα δεξιά προς τα αριστερά όπως η Αραβική, Εβραϊκή, Φαρσί κτλ.

Η γραφική διεπαφή του χρήστη θα πρέπει να αντιστρέψει μερικά στοιχεία για να ακολουθήσουν αυτήν την γραφή.

Αυτό γίνεται αυτόματα για στοιχεία ελέγχου όπως το HBox, VBox, HPanel και VPanel, ή δοχείς οι οποίοι έχουν την ιδιότητα Arrangement ορισμένη στην κατάλληλη τιμή.

Τα άλλα μέρη του GUI (menus, labels, widget scrollbars...) καθορίζουν την συμπεριφορά τους από ότι παρέχουν οι βιβλιοθήκες Qt η GTK+.

Για να έχετε ένα αντίστροφο GUI, θα πρέπει να θέσετε την μεταβλητή `$GB_REVERSE` πριν την εκκίνηση του προγράμματος.

Λογαριθμικές και εκθετικές συναρτήσεις (Logarithms & Exponentials Functions)

Αυτές οι συναρτήσεις είναι οι εξής:

Cbr	Κυβική ρίζα
Exp	Εκθέτης, e^x
Exp2	Εκθέτης 2^x
Exp10	Εκθέτης 10^x
Expm	Exp(x) - 1
Log	Λογάριθμος με βάση το e
Log2	Λογάριθμος με βάση το 2
Log10	Δεκαδικός λογάριθμος
Logp	Log(1+x)
Sqr	Τετραγωνική ρίζα

Cbr

Κυβική ρίζα

Σύνταξη:

```
Value = Cbr ( Number AS Float ) AS Float
```

Value, η επιστροφή της συνάρτησης είναι ένας Float.
Number, ο δεκαδικός αριθμός που θα υπολογιστεί.

Παράδειγμα:

```
PRINT Cbr(8)
PRINT Cbr(2)
```

Έξοδος:

```
2
1.259921049895
```

Exp

Εκθέτης, e^x

Σύνταξη:

```
Value = Exp ( Number AS Float ) AS Float
```

Value, η επιστροφή της συνάρτησης είναι ένας Float.
Number, ο δεκαδικός αριθμός που θα υπολογιστεί.

Παράδειγμα:

```
PRINT Exp(8)
PRINT Exp(2)
```

Έξοδος:

```
2980.95798704173
7.389056098931
```

Exp2

Εκθέτης 2^x

Σύνταξη:

```
Value = Exp2 ( Number AS Float ) AS Float
```

Value, η επιστροφή της συνάρτησης είναι ένας Float.
Number, ο δεκαδικός αριθμός που θα υπολογιστεί.

Παράδειγμα:

```
PRINT Exp2(8)
PRINT Exp2(2)
```

Έξοδος:

```
256
4
```

Exp10

Εκθέτης 10^x

Σύνταξη:

```
Value = Exp10 ( Number AS Float ) AS Float
```

Value, η επιστροφή της συνάρτησης είναι ένας Float.
Number, ο δεκαδικός αριθμός που θα υπολογιστεί.

Παράδειγμα:

```
PRINT Exp10(8)
PRINT Exp10(2)
```

Έξοδος:

```
1E+8
100
```

Expm

Υπολογίζει το Exp(x) - 1

Σύνταξη:

```
Value = Expm ( Number AS Float ) AS Float
```

Value, η επιστροφή της συνάρτησης είναι ένας Float.
Number, ο δεκαδικός αριθμός που θα υπολογιστεί.

Παράδειγμα:

```
PRINT Expm(8)
PRINT Expm(2)
```

Έξοδος:

```
2979.95798704173
6.389056098931
```

Log

Λογάριθμος με βάση το e

Σύνταξη:

```
fValue = Log ( fNumber AS Float ) AS Float
```

fValue, η επιστροφή της συνάρτησης είναι ένας Float.
fNumber, ο αριθμός που θα υπολογιστεί.

Παράδειγμα:

```
PRINT Log(8)
PRINT Log(2)
```

Έξοδος:

```
2.07944154168
0.69314718056
```

Log2

Λογάριθμος με βάση το 2

Σύνταξη:

```
fValue = Log2 ( fNumber AS Float ) AS Float
```

fValue, η επιστροφή της συνάρτησης είναι ένας Float.
fNumber, ο αριθμός που θα υπολογιστεί.

Παράδειγμα:

```
PRINT Log2(8)
PRINT Log2(2)
```

Έξοδος:

```
3
1
```

Log10

Δεκαδικός λογάριθμος

Σύνταξη:

```
fValue = Log10 ( fNumber AS Float ) AS Float
```

fValue, η επιστροφή της συνάρτησης είναι ένας Float.
fNumber, ο αριθμός που θα υπολογιστεί.

Παράδειγμα:

```
PRINT Log10(8)
PRINT Log10(2)
```

Έξοδος:

```
0.903089986992
0.301029995664
```

Logp

Υπολογίζει το Log(1+x)

Σύνταξη:

```
fValue = Logp ( fNumber AS Float ) AS Float
```

fValue, η επιστροφή της συνάρτησης είναι ένας Float.
fNumber, ο αριθμός που θα υπολογιστεί.

Παράδειγμα:

```
PRINT Logp(8)
PRINT Logp(2)
```

Έξοδος:

```
2.197224577336
1.098612288668
```

Sqr

Τετραγωνική ρίζα

Σύνταξη:

```
Value = Sqr ( Number AS Float )
```

Value, η επιστροφή της συνάρτησης.
Number, ο αριθμός που θα υπολογιστεί.

Παράδειγμα:

```
PRINT Sqr(8)
PRINT Sqr(2)
PRINT Sqr(4)
```

Έξοδος:

```
2.828427124746
1.414213562373
2
```

Τριγωνομετρικές συναρτήσεις (Trigonometric Functions)

Αυτές είναι:

`ACos`	Υπολογίζει το συνημίτονο μιας γωνίας.
`ACosh`	Υπολογίζει το υπερβολικό συνημίτονο μιας γωνίας.
`Ang`	Υπολογίζει την πολική συντεταγμένη γωνίας από δύο ορθογώνιες συντεταγμένες.
`ASin`	Υπολογίζει το ημίτονο μιας γωνίας.
`ASinh`	Υπολογίζει το υπερβολικό ημίτονο μιας γωνίας.
`ATan`	Υπολογίζει την εφαπτομένη μιας γωνίας.
`ATan2`	Υπολογίζει την εφαπτομένη δύο αριθμών.
`ATanh`	Υπολογίζει την υπερβολική εφαπτομένη μιας γωνίας.
`Cos`	Υπολογίζει το συνημίτονο μιας γωνίας.
`Cosh`	Υπολογίζει το υπερβολικό συνημίτονο μιας γωνίας.
`Deg`	Μετατρέπει τα ακτίνια σε μοίρες.
`Hyp`	Υπολογίζει την υποτείνουσα ενός τριγώνου
`Mag`	Υπολογίζει την πολική συντεταγμένη απόστασης από δύο ορθογώνιες συντεταγμένες.
`Pi`	Επιστρέφει το π ή πολλαπλάσιο του π.
`Sin`	Υπολογίζει το ημίτονο μιας γωνίας.
`Sinh`	Υπολογίζει το υπερβολικό ημίτονο μιας γωνίας.
`Tan`	Υπολογίζει την εφαπτομένη μιας γωνίας.
`Tanh`	Υπολογίζει την υπερβολική εφαπτομένη μιας γωνίας.
`Rad`	Μετατρέπει τις μοίρες στα ακτίνια.

ACos

Υπολογίζει το συνημίτονο μιας γωνίας.

Σύνταξη:

```
Value = ACos ( Number AS Float ) AS Float

Value = Acs ( Number AS Float ) AS Float
```

Value, η επιστροφή της συνάρτησης.
Number, ο αριθμός που θα υπολογιστεί. Ο αριθμός θα πρέπει να είναι μεταξύ -1 και 1, αλλιώς εμφανίζεται ένα λάθος.

Παράδειγμα:

```
PRINT ACos(0.9)
```

Έξοδος:

```
0.451026811796
```

ACosh

Υπολογίζει το υπερβολικό συνημίτονο μιας γωνίας.

Σύνταξη:

```
Value = ACosh ( Number AS Float ) AS Float

Value = Acsh ( Number AS Float ) AS Float
```

Value, η επιστροφή της συνάρτησης.
Number, ο αριθμός που θα υπολογιστεί.

Παράδειγμα:

```
PRINT ACosh(1)
```

Έξοδος:

```
0
```

Ang

Υπολογίζει την πολική συντεταγμένη γωνίας από δύο ορθογώνιες συντεταγμένες.

Σύνταξη:

```
Angle = Ang ( X AS Float , Y AS Float ) AS Float
```

Angle, η επιστροφή της συνάρτησης, το αποτέλεσμα είναι μεταξύ -π και π.
X, η πρώτη συντεταγμένη.
Y, η δεύτερη συντεταγμένη.

Η συνάρτηση είναι η ίδια με την Atan2()

Παράδειγμα:

```
PRINT Ang(-1, 1)
```

Έξοδος:

```
2.356194490192
```

ASin

Υπολογίζει το ημίτονο μιας γωνίας.

Σύνταξη:

```
Value = ASin ( Number AS Float ) AS Float

Value = Asn ( Number AS Float ) As Float
```

Value, η επιστροφή της συνάρτησης.
Number, ο αριθμός που θα υπολογιστεί. Ο αριθμός θα πρέπει να είναι μεταξύ -1 και 1, αλλιώς εμφανίζεται ένα λάθος.

Παράδειγμα:

```
PRINT Asin(0.9)
```

Έξοδος:

```
1.119769514999
```

ASinh

Υπολογίζει το υπερβολικό ημίτονο μιας γωνίας.

Σύνταξη:

```
Value = ASinh ( Number AS Float ) AS Float

Value = Asnh ( Number AS Float ) As Float
```

Value, η επιστροφή της συνάρτησης.
Number, ο αριθμός που θα υπολογιστεί.

Παράδειγμα:

```
PRINT Asinh(0.9)
```

Έξοδος:

```
0.808866935653
```

ATan

Υπολογίζει την εφαπτομένη μιας γωνίας.

Σύνταξη:

```
Value = ATan ( fNumber AS Float ) AS Float

Value = Atn ( fNumber AS Float ) AS Float
```

Value, η επιστροφή της συνάρτησης.
fNumber, ο αριθμός που θα υπολογιστεί.

Παράδειγμα:

```
PRINT ATan(0.9)
```

Έξοδος:

```
0.732815101787
```

ATan2

Υπολογίζει την εφαπτομένη δύο αριθμών.

Σύνταξη:

```
Angle = ATan2 ( Y AS Float , X AS Float ) AS Float

Angle = Atn2 ( Y AS Float , X AS Float ) AS Float
```

Angle, η επιστροφή της συνάρτησης, το αποτέλεσμα είναι μεταξύ -π και π.
Y, η πρώτη συντεταγμένη.
X, η δεύτερη συντεταγμένη.

Η συνάρτηση είναι η ίδια με την Ang()

Παράδειγμα:

```
PRINT Ang(-1, 1)
PRINT ATan2(1, -1)
```

Έξοδος:

```
2.356194490192
2.356194490192
```

ATanh

Υπολογίζει την υπερβολική εφαπτομένη μιας γωνίας.

Σύνταξη:

```
Value = ATanh ( Number AS Float ) AS Float

Value = Atnh ( Number AS Float ) AS Float
```

Value, η επιστροφή της συνάρτησης.
Number, ο αριθμός που θα υπολογιστεί.

Παράδειγμα:

```
PRINT ATanh(0.9)
```

Έξοδος:

```
1.472219489583
```

Cos

Υπολογίζει το συνημίτονο μιας γωνίας.

Σύνταξη:

```
Value = Cos ( Angle AS Float ) AS Float
```

Value, η επιστροφή της συνάρτησης.
Angle, η γωνία που θα υπολογιστεί.

Παράδειγμα:

```
PRINT Cos(Pi)
```

Έξοδος:

```
-1
```

Cosh

Υπολογίζει το υπερβολικό συνημίτονο μιας γωνίας.

Σύνταξη:

```
Value = Cosh ( Number AS Float ) AS Float
```

Value, η επιστροφή της συνάρτησης.
Number, ο αριθμός που θα υπολογιστεί.

Παράδειγμα:

```
PRINT Cosh(0.9)
```

Έξοδος:

```
1.433086385449
```

Deg

Μετατρέπει τα ακτίνια σε μοίρες.

Σύνταξη:

```
Value = Deg ( Angle AS Float ) AS Float
```

Value, η επιστροφή της συνάρτησης.
Angle, τα ακτίνια που θα υπολογιστούν.

Παράδειγμα:

```
PRINT Deg(0.9)
```

Έξοδος:

```
51.566201561774
```

Hyp

Υπολογίζει την υποτείνουσα ενός τριγώνου

Σύνταξη:

```
Hypotenuse = Hyp ( X AS Float , Y AS Float ) AS Float
```

Hypotenuse, η επιστροφή της συνάρτησης, το αποτέλεσμα είναι Sqr((X ^ 2)+(Y ^ 2)).
X, το μήκος της πρώτης πλευράς του τριγώνου.
Y, το μήκος της δεύτερης πλευράς του τριγώνου.

Η συνάρτηση είναι η ίδια με την Mag()

Παράδειγμα:

```
PRINT Hyp(1, 1)
```

Έξοδος:

```
1.414213562373
```

Mag

Υπολογίζει την πολική συντεταγμένη απόστασης από δύο ορθογώνιες συντεταγμένες.

Σύνταξη:

```
Distance = Mag ( X AS Float , Y AS Float )
```

Distance, η επιστροφή της συνάρτησης, το αποτέλεσμα είναι Sqr((X ^ 2)+(Y ^ 2)).
X, το μήκος της πρώτης πλευράς του τριγώνου.
Y, το μήκος της δεύτερης πλευράς του τριγώνου.

Η συνάρτηση είναι η ίδια με την Hyp()

Παράδειγμα:

```
PRINT Mag(1, 1)
```

Έξοδος:

```
1.414213562373
```

Pi

Επιστρέφει το π ή πολλαπλάσιο του π.

Σύνταξη:

```
fResult = Pi ( [ fNumber AS Float ] )
```

fResult, η επιστροφή της συνάρτησης, το αποτέλεσμα είναι π*fNumber.
Αν δεν οριστεί fNumber, τότε το προκαθορισμένο είναι 1.

Παράδειγμα:

```
PRINT Pi
```

Έξοδος:

```
3.14159265359
```

Sin

Υπολογίζει το ημίτονο μιας γωνίας.

Σύνταξη:

```
Value = Sin ( Angle AS Float )
```

Value, η επιστροφή της συνάρτησης.
Angle η γωνία σε ακτίνια.

Παράδειγμα:

```
PRINT Sin(Pi/2)
```

Έξοδος:

```
1
```

Sinh

Υπολογίζει το υπερβολικό ημίτονο μιας γωνίας.

Σύνταξη:

```
Value = Sinh ( Number AS Float )
```

Value, η επιστροφή της συνάρτησης.
Number ο αριθμός.

Παράδειγμα:

```
PRINT Sinh(Pi/2)
```

Έξοδος:

```
2.301298902307
```

Tan

Υπολογίζει την εφαπτομένη μιας γωνίας.

Σύνταξη:

```
Value = Tan ( Angle AS Float )
```

Value, η επιστροφή της συνάρτησης.
Angle η γωνία σε ακτίνια.

Παράδειγμα:

```
PRINT Tan(Pi/4)
```

Έξοδος:

```
1
```

Tanh

Υπολογίζει την υπερβολική εφαπτομένη μιας γωνίας.

Σύνταξη:

```
Value = Tanh ( Number AS Float )
```

Value, η επιστροφή της συνάρτησης.
Number ο αριθμός.

Παράδειγμα:

```
PRINT Tanh(Pi/4)
```

Έξοδος:

```
0.655794202633
```

Rad

Μετατρέπει τις μοίρες στα ακτίνια.

Σύνταξη:

```
fValue = Rad ( fAngle AS Float ) AS Float
```

fValue, η επιστροφή της συνάρτησης.
fAngle η γωνία σε ακτίνια.

Παράδειγμα:

```
PRINT Rad(30)
```

Έξοδος:

```
0.523598775598
```

Συναρτήσεις τυχαίων αριθμών (Random Numbers Functions)

Οι συναρτήσεις αυτές είναι:

RANDOMIZE	Αρχικοποιεί την γεννήτρια ψευδο-τυχαίων αριθμών.
Rnd	Επιστρέφει έναν ψευδο-τυχαίο αριθμό.

RANDOMIZE

Αρχικοποιεί την γεννήτρια ψευδο-τυχαίων αριθμών.

Σύνταξη:

```
RANDOMIZE [ Seed AS Integer ]
```

Seed, το όρισμα της συνάρτησης.

Εάν δεν οριστεί όρισμα τότε το προκαθορισμένο όρισμα είναι η ημερομηνία και ο χρόνος της τρέχουσας στιγμής.

Το ίδιο όρισμα θα σας δίνει τις ίδιες τιμές.

Rnd

Επιστρέφει έναν ψευδο-τυχαίο αριθμό.

Σύνταξη:

```
Value = Rnd ( [ Min [ , Max ] )
```

Value, η επιστροφή της συνάρτησης, είναι ένας Float.
Min το κατώτατο όριο τιμών των τυχαίων αριθμών.
Max το ανώτατο όριο τιμών των τυχαίων αριθμών.

- Αν δεν οριστούν τιμές Min, Max, οι προκαθορισμένες τιμές είναι 0, 1. Επιστρέφεται μια τιμή από το εύρος [0, 1].
- Αν οριστεί μια τιμή τότε αυτή χρησιμοποιείται ως τμέγιστη, η συνάρτηση επιστρέφεται μια τιμή από το εύρος [0, Min].
  Προσοχή αν διοχετευτεί μία τιμή αυτή εκχωρείται στο πρώτο όρισμα της συνάρτησης δλδ το Min. Στην περίπτωση αυτή όμως η Rnd() την χρησιμοποιεί ως το ανώτερο όριο τιμών δλδ ως Max. Έτσι μια έκφραση Rnd(2), είναι ισοδύναμη με την Rnd(0, 2),

παρόλο που το 2 εκχωρείται στην μεταβλητή Min.

- Αν οριστούν δύο τιμές η πρώτη είναι η Min και η δεύτερη η Max. Επιστρέφεται μια τιμή από το εύρος [Min, Max].

Η ανώτερη τιμή Max δεν επιστρέφεται ποτέ από την συνάρτηση.
Αν θέλετε να επιστραφούν τιμές μεταξύ 1 και 6 θα πρέπει η μέγιστη τιμή να είναι 7.

Παράδειγμα:

```
PRINT Rnd(2) 'a random number between 0 and 1.9999999999999
```

Παράδειγμα:

```
PRINT Int(Rnd(5)) 'a random number between 0 and 4
```

Παράδειγμα:

```
RANDOMIZE
PRINT Rnd(1, 7) 'a random number between 1 and 6.9999999999999
```

Έξοδος:

```
2.275750088524
```

Εξομοίωση ενός ζαριού.
Παράδειγμα:

```
DIM Dice AS Integer

 RANDOMIZE

 Dice = Int(Rnd(1, 7)) 'a random integer between 1 and 6

 'Throws the dice between 1 and 6
 PRINT "You threw a " & dice
```

Έξοδος:

```
You threw a 3
```

Συναρτήσεις διαχείρισης διεργασιών (Process Management)

Οι συναρτήσεις διαχείρισης διεργασιών εκτελούν μια εντολή του συστήματος.

Αυτές είναι:

EXEC	Εκτελεί μια εξωτερική διεργασία.
SHELL	Εκτελεί μια εξωτερική διεργασία σε ένα κέλυφος.
Shell$	Εσωκλείει ένα string με εισαγωγικά έτσι ώστε να είναι ένα ασφαλές όρισμα τερματικού.

EXEC

Εκτελεί μια εξωτερική διεργασία.

Σύνταξη:

```
[ Process = ] EXEC Command [ WAIT ] [ FOR { { READ | INPUT } | {
WRITE | OUTPUT } } ] [ AS Name ]



EXEC Command TO Variable
```

Process, η συνάρτηση επιστρέφει ένα αντικείμενο Process.

Command, απαραίτητος πίνακας που ορίζει τα μέρη της εξωτερικής διεργασίας. Το πρώτο στοιχείο του πίνακα είναι το όνομα της εντολής και είναι απαραίτητο, τα υπόλοιπα στοιχεία είναι οι παράμετροι της εντολής και μπορεί να είναι προαιρετικοί.

Αν οριστεί το **WAIT** ο διερμηνευτής περιμένει την εντολή να τελειώσει. Αλλιώς η εντολή εκτελείται στο παρασκήνιο.

Αν οριστεί το **FOR** τότε η είσοδος και έξοδος του προγράμματος ανακατευθύνονται ώστε το πρόγραμμα σας να μπορέσει να επικοινωνήσει με την διεργασία.

> Εάν το **WRITE** έχει οριστεί τότε μπορείτε να στείλετε δεδομένα στην stdin χρησιμοποιώντας το αντικείμενο Process και τις εντολές εξόδου **PRINT**, **WRITE** ... Σημειώστε ότι θα πρέπει να έχετε μια αναφορά στο αντικείμενο Process.
>
> Εάν το **READ** έχει οριστεί τότε όταν η διεργασία στέλνει δεδομένα στην stdout και stderr και ενεργοποιείται το συμβάν Read και Error του αντικειμένου Process. Έτσι μπορείτε να τα διαβάζετε τα δεδομένα από την διεργασία με τις συναρτήσεις ροής (Stream & Input/Output functions).
>
> Εάν χρησιμοποιήσετε τα **INPUT** και **OUTPUT** αντί των **READ**, **WRITE** τότε η

διεργασία τρέχει σε ένα εικονικό τερματικό. Αυτό σημαίνει ότι η διεργασία θα νομίζει ότι τρέχει σε πραγματικό τερματικό.

Name, είναι το όνομα του συμβάντος που χρησιμοποιείται από το αντικείμενο Process. Αν δεν οριστεί το προκαθορισμένο όνομα είναι *Process*.

Η δεύτερη σύνταξη της συνάρτησης περιμένει να τελειώσει η διεργασία και επιστρέφει την έξοδο της στην μεταβλητή Variable.

Variable, είναι το όνομα της μεταβλητής που θα περιέχει την έξοδο της διεργασίας στην δεύτερη σύνταξη της συνάρτησης EXEC.

Αν θέλετε να υπολογίσετε πόσα bytes διαβάζετε από την Read ρουτίνα χρησιμοποιείστε την συνάρτηση Lof().

Οι παράμετροι που περνάτε με την EXEC δεν χρειάζεται να κλειστούν σε διπλά εισαγωγικά όπως γίνεται σε ένα κανονικό τερματικό.

Παράδειγμα:

```
' Get the contents of a directory
EXEC ["ls", "-la", "/tmp"] WAIT
```

Παράδειγμα:

```
' Get the contents of a directory into a string
DIM sOutput AS String
EXEC ["ls", "-la", "/tmp"] TO sOutput
```

Παράδειγμα:
Δημιουργείστε το Project30 με μία μονάδα Module ορισμένη ως κλάση εκκίνησης.

```
1  ' Gambas module file
2  PRIVATE sOutput AS String
3
4  PUBLIC SUB Main()
5
6  ' A specific event name is used
7  EXEC ["ls", "-la", "/tmp"] FOR READ AS "Contents"
8
9  END
10
11 PUBLIC SUB Contents_Read()
12
13   DIM sLine AS String
14
15   READ #LAST, sLine, -256
16
17   sOutput &= sLine
```

```
18
19  END
20
21  PUBLIC SUB Contents_Kill()
22
23    PRINT sOutput
24
25  END
```

Έξοδος:

```
total 148
drwxrwxrwt 25 root root 36864 2010-11-25 14:40 .
drwxr-xr-x 24 root root  4096 2010-11-25 08:40 ..
drwx------  2 user user  4096 2010-11-25 14:40 acroread_1000_1000
drwxrwxrwt  2 user user  4096 2010-11-25 09:30 _avast4_
-rw-r--r--  1 user user     6 2010-11-25 08:40 blueman-applet-1000
drwx------  2 user user  4096 2010-11-25 08:40 .esd-1000
.....
```

Ανάλυση:
Στην γραμμή 1 δηλώνεται μια ιδιωτική καθολική μεταβλητή αλφαριθμητικών. Η μεταβλητή αυτή θα περιέχει την έξοδο της διεργασίας.

Στις γραμμές 4 με 9 έχουμε την συνάρτηση Main().

Στην γραμμή 7 εκτελείται η εντολή EXEC με ορίσματα τα “ls”, “-la”, “/tmp”.

Το πρώτο όρισμα είναι η εντολή ls.
Το δεύτερο όρισμα είναι οι παράμετροι και το τρίτο όρισμα είναι ο κατάλογος που θα εμφανιστούν τα περιεχόμενα του.

Η εντολή EXEC έχει οριστεί με την READ και έτσι είναι δυνατόν να διαβάσουμε τα δεδομένα που θα επιστρέψει η διεργασία με το συμβάν Read.

Το όνομα του συμβάντος είναι Contents. Και η ρουτίνα που διαβάζει την έξοδο της διεργασίας είναι Contents_Read().

Στις γραμμές 11 με 19 έχουμε την ρουτίνα του συμβάντος Contents_Read().

Για να κατανοηθεί το παράδειγμα θα πρέπει να διαβάσετε και τις συναρτήσεις ροής (Stream & Input/Output functions) που έπονται.

Στην γραμμή 13 ορίζεται μια μεταβλητή η οποία θα περιέχει την γραμμή από την κάθε φορά που θα ενεργοποιείται το συμβάν Contents_Read().

Η γραμμή 15 διαβάζει την γραμμή που δίνεται στην stdout από την διεργασία και τοποθετεί το

περιεχόμενο της στην τοπική μεταβλητή sLine.

Στην γραμμή 17 η καθολική μεταβλητή λαμβάνει αθροιστικά όλες τις τιμές της sLine κάθε φορά που εγείρεται το συμβάν Contents_Read().

Στις γραμμές 21 με 25 είναι η ρουτίνα του συμβάντος τερματισμού της διεργασίας.

Έτσι όταν η διεργασία τελειώσει η γραμμή 23 τυπώνει το περιεχόμενο της καθολικής μεταβλητής που είναι όπως είδαμε όλες οι γραμμές που έχουν τυπωθεί στην stdout.

SHELL

Εκτελεί μια εξωτερική διεργασία σε ένα κέλυφος.

Σύνταξη:

```
[ Process = ] SHELL Command [ WAIT ] [ FOR { { READ | INPUT } | { WRITE | OUTPUT } } ] [ AS Name ]

SHELL Command TO Variable
```

Process, η συνάρτηση επιστρέφει ένα αντικείμενο Process.

Command, απαραίτητο string που ορίζει τηη εξωτερικής διεργασίας με τις παραμέτρους της.

Αν οριστεί το **WAIT** ο διερμηνευτής περιμένει την εντολή να τελειώσει. Αλλιώς η εντολή εκτελείται στο παρασκήνιο.

Αν οριστεί το **FOR** τότε η είσοδος και έξοδος του προγράμματος ανακατευθύνονται ώστε το πρόγραμμα σας να μπορέσει να επικοινωνήσει με την διεργασία.

> Εάν το **WRITE** έχει οριστεί τότε μπορείτε να στείλετε δεδομένα στην stdin χρησιμοποιώντας το αντικείμενο Process και τις εντολές εξόδου **PRINT**, **WRITE** ... Σημειώστε ότι θα πρέπει να έχετε μια αναφορά στο αντικείμενο Process.
>
> Εάν το **READ** έχει οριστεί τότε όταν η διεργασία στέλνει δεδομένα στην stdout και stderr και ενεργοποιείται το συμβάν Read και Error του αντικειμένου Process. Έτσι μπορείτε να τα διαβάζετε τα δεδομένα από την διεργασία με τις συναρτήσεις ροής (Stream & Input/Output functions).
>
> Εάν χρησιμοποιήσετε τα **INPUT** και **OUTPUT** αντί των **READ**, **WRITE** τότε η διεργασία τρέχει σε ένα εικονικό τερματικό. Αυτό σημαίνει ότι η διεργασία θα νομίζει ότι τρέχει σε πραγματικό τερματικό.

Name, είναι το όνομα του συμβάντος που χρησιμοποιείται από το αντικείμενο Process. Αν δεν οριστεί το προκαθορισμένο όνομα είναι *Process*.

Η δεύτερη σύνταξη της συνάρτησης περιμένει να τελειώσει η διεργασία και επιστρέφει την έξοδο της στην μεταβλητή Variable.

Variable, είναι το όνομα της μεταβλητής που θα περιέχει την έξοδο της διεργασίας στην δεύτερη σύνταξη της συνάρτησης SHELL.

Αν θέλετε να υπολογίσετε πόσα bytes διαβάζετε από την Read ρουτίνα χρησιμοποιείστε την συνάρτηση Lof().

Οι παράμετροι που περνάτε με την SHELL χρειάζεται να κλειστούν σε εισαγωγικά όπως γίνεται σε ένα κανονικό τερματικό.

Παράδειγμα:

```
' Get the content of a directory
SHELL "ls -la /tmp" WAIT
```

Παράδειγμα:

```
' Get the contents of a directory into a string
DIM sOutput AS String
SHELL "ls -la /tmp" TO sOutput
```

Παράδειγμα:
Δημιουργείστε το Project31 με μία μονάδα Module ορισμένη ως κλάση εκκίνησης.

```
1  ' Gambas module file
2  PRIVATE sOutput AS String
3
4  PUBLIC SUB Main()
5
6  ' A specific event name is used
7  SHELL "ls -la /tmp" FOR READ AS "Contents"
8
9  END
10
11 PUBLIC SUB Contents_Read()
12
13   DIM sLine AS String
14
15   READ #LAST, sLine, -256
16
17   sOutput &= sLine
18
19 END
20
21 PUBLIC SUB Contents_Kill()
22
```

```
23    PRINT sOutput
24
25  END
```

Το παράδειγμα αυτό είναι το ίδιο με το παράδειγμα του Project30 μόνο που στην γραμμή 7 η διεργασία εκτελείται με την SHELL.
Μπορείτε να αποθηκεύσετε το παράδειγμα αυτό σαν Project31.

Έξοδος:

```
total 148
drwxrwxrwt 25 root root 36864 2010-11-25 14:40 .
drwxr-xr-x 24 root root  4096 2010-11-25 08:40 ..
drwx------  2 user user  4096 2010-11-25 14:40 acroread_1000_1000
drwxrwxrwt  2 user user  4096 2010-11-25 09:30 _avast4_
-rw-r--r--  1 user user     6 2010-11-25 08:40 blueman-applet-1000
drwx------  2 user user  4096 2010-11-25 08:40 .esd-1000
.....
```

SHELL$

Εσωκλείει ένα string σε εισαγωγικά έτσι ώστε αυτό να είναι ένα ασφαλές όρισμα για την συνάρτηση SHELL.

Σύνταξη:

```
QuotedString = Shell ( String )

QuotedString = Shell$ ( String )
```

QuotedString, είναι το string που έχει εσωκλειστεί σε εισαγωγικά και μπορεί να αποτελέσει ένα ασφαλές όρισμα για την συνάρτηση SHELL.
String, το string προς επεξεργασία.

Συναρτήσεις ροής και εισόδου/εξόδου (Stream & Input/Output functions)

Λίγα λόγια για την είσοδο και έξοδο.

Σε προηγούμενο κεφάλαιο είδαμε κάποια θέματα για τους περιγραφείς αρχείων. Εδώ θα επαναλάβουμε αυτό το θέμα μιας και αφορά την είσοδο και έξοδο προς και από ένα πρόγραμμα.

Στον προγραμματισμό ένας περιγραφέας (descriptor) είναι ένας δείκτης προς ένα αρχείο. Ένα αρχείο σε ένα σύστημα POSIX μπορεί να είναι

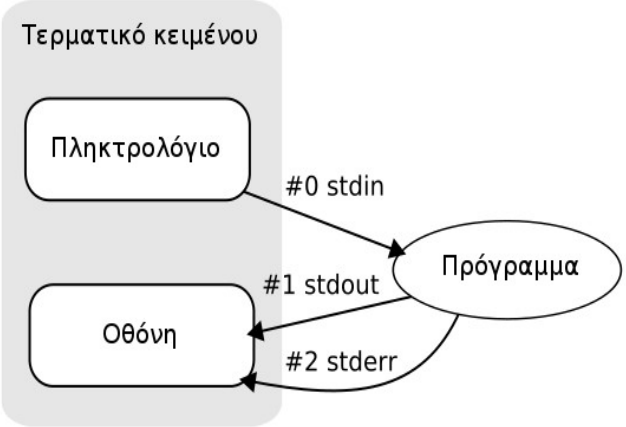
- ένα απλό file
- ένα folder
- μια συσκευή π.χ. /dev/sda
- ένα pipe
- ένα socket
- ένα link

Στα συστήματα POSIX όπως το Linux, υπάρχουν τρεις βασικοί περιγραφείς που κάθε ένας τους ορίζεται με έναν ακέραιο.

Τιμή ακεραίου	**Όνομα**	**Μετάφραση**
0	Standard Input (stdin)	Προκαθορισμένη είσοδος.
1	Standard Output (stdout)	Προκαθορισμένη έξοδος.
2	Standard Error (stderr)	Προκαθορισμένη έξοδος λαθών.

- Η προκαθορισμένη είσοδος λαμβάνει δεδομένα από όλες τις συσκευές που εισάγουν δεδομένα στον η/υ όπως το ποντίκι, το πληκτρολόγιο, το scanner κτλ.
- Η προκαθορισμένη έξοδος είναι συνήθως το τερματικό, όπου γράφουμε εντολές και εμφανίζονται τα αποτελέσματα τους.
- Η προκαθορισμένη έξοδος λαθών, είναι συνήθως πάλι το τερματικό, αποτελεί ξεχωριστό περιγραφέα από την προκαθορισμένη έξοδο και εδώ εμφανίζονται τα λάθη των εφαρμογών εξ' ορισμού.

Όταν ο χρήστης πληκτρολογεί κάτι στο τερματικό προς ένα πρόγραμμα, το πρόγραμμα διαβάζει την stdin και δέχεται είσοδο προς αυτό.
Το πρόγραμμα εκτελείται και τυπώνει μηνύματα στην stdout.
Αν το πρόγραμμα δημιουργήσει ένα λάθος τότε τυπώνονται μηνύματα λάθους στην stderr.
Συνήθως τόσο η stdout όσο και η stderr τυπώνουν μηνύματα στην οθόνη.



Εικόνα 70: Είσοδος και έξοδος ροής δεδομένων

Έτσι μπορούμε να μιλήσουμε για ροή δεδομένων προς ένα πρόγραμμα και ροή δεδομένων από ένα πρόγραμμα. Η ροή δεδομένων (stream) έχει τελικό αποδέκτη κάποιο αρχείο σε ένα σύστημα Linux.
Στο παραπάνω παράδειγμα το πρόγραμμα δέχεται δεδομένα από το /dev/tty1 και στέλνει δεδομένα προς το /dev/tty1.
Το ίδιο ακριβώς μπορεί να κάνει και για τον εκτυπωτή (/dev/lp1), για ένα αρχείο (/home/user/file1.txt), για ένα process κτλ.

Στην Gambas υπάρχουν κάποια αντικείμενα που αντιμετωπίζονται ως αρχεία και τα δεδομένα προς αυτά ή από αυτά τα διαχειρίζονται συναρτήσεις ροής δεδομένων.

Τα αντικείμενα αυτά είναι:

- Compress
- Curl
- File
- Process
- SerialPort
- Socket
- Uncompress
- VideoDevice

Οι συναρτήσεις ροής δεδομένων

Οι συναρτήσεις αυτές είναι:

OPEN	Ανοίγει ένα αρχείο για διάβασμα ή γράψιμο και δημιουργεί ένα stream για αυτό.
CLOSE	Κλείνει ένα stream
PRINT	Τυπώνει μια έκφραση σε ένα stream.
Eof	Επιστρέφει αν το τέλος του αρχείου έχει φτάσει.
Lof	Επιστρέφει το μήκος του stream.
ERROR	Τυπώνει μια έκφραση στην προκαθορισμένη έξοδο λαθών (stderr).
FLUSH	Αδειάζει την έξοδο μιας προσωρινής περιοχής αποθήκευσης από ένα stream.
INPUT	Διαβάζει αλφαριθμητικά από ένα stream κειμένου και τα μετατρέπει σε τιμές.
LINE INPUT	Διαβάζει μια γραμμή από ένα stream κειμένου.
READ	Διαβάζει δυαδικά δεδομένα από ένα stream.
MEMORY	Διαβάζει δυαδικά δεδομένα από ένα stream από την μνήμη.
WRITE	Γράφει δυαδικά δεδομένα σε ένα stream.
OUTPUT TO	Ανακατευθύνει την stdout.
INPUT FROM	Ανακατευθύνει την stdin.
ERROR TO	Ανακατευθύνει την stderr.
PIPE	Ανοίγει έναν αγωγό με όνομα για διάβασμα ή γράψιμο και δημιουργεί ένα stream για αυτό.
LOCK	Κλειδώνει ένα ανοιχτό stream.
UNLOCK	Ξεκλειδώνει ένα ανοιχτό stream.
SEEK	Αλλάζει την θέση ενός δείκτη αρχείου ενός stream.
Seek	Λαμβάνει την θέση ενός δείκτη αρχείου ενός stream.

OPEN

Ανοίγει ένα αρχείο για διάβασμα ή γράψιμο και δημιουργεί ένα stream για αυτό.

Σύνταξη:

```
hStream = OPEN sFileName FOR [ READ | INPUT ] [ WRITE | OUTPUT ] [
CREATE | APPEND ] [ WATCH ]
```

hStream, ένα αντικείμενο stream το οποίο επιστρέφει η συνάρτηση.
sFileName, το όνομα του αρχείου που θα ανοιχτεί

FOR, απαραίτητη λέξη που δηλώνει την εργασία που θα διενεργηθεί στο αντικείμενο stream.

READ | **INPUT**, όταν οριστούν οι λέξεις αυτές μπορούμε να διαβάσουμε από το stream.

WRITE | **OUTPUT**, όταν οριστούν αυτές οι λέξεις μπορούμε να γράψουμε στο stream.

CREATE, όταν οριστεί η λέξη αυτή αν το αρχείο δεν υπάρχει δημιουργείται ή αν υπάρχει καθαρίζεται από το περιεχόμενο του.

APPEND, όταν οριστεί η λέξη αυτή ο δείκτης του αρχείου τοποθετείται στο τέλος του αρχείου και έτσι μπορούμε να γράψουμε στο αρχείο συμπληρωματικά.

READ, WRITE, αν οριστούν οι λέξεις αυτές τότε η είσοδος και έξοδος δεν αποθηκεύονται προσωρινά. (not buffered)

INPUT, **OUTPUT**, αν οριστούν οι λέξεις αυτές τότε η είσοδος και έξοδος αποθηκεύονται προσωρινά. (buffered)

WATCH, αν οριστεί η λέξη αυτή τότε ο διερμηνευτής παρακολουθεί το αρχείο.

- Αν ένα byte μπορεί να διαβαστεί από το αρχείο τότε εγείρεται το συμβάν File_Read().
- Αν ένα byte μπορεί να γραφεί στο αρχείο τότε εγείρεται το συμβάν File_Write().

Εξ' ορισμού η ροή των δεδομένων αποθηκεύεται σε προσωρινή μνήμη. (buffer).
Αν θέλετε να μην αποθηκεύεται σε buffer θα πρέπει να χρησιμοποιήσετε τις λέξεις **READ** ή **WRITE**

Παράδειγμα:

```
DIM hFile AS File
hFile = OPEN "/etc/passwd" FOR INPUT
```

CLOSE

Κλείνει ένα stream

Σύνταξη:

```
CLOSE [ # ] Stream
```

Η εντολή αυτή δεν αποτυγχάνει ποτέ.
Εάν έχετε ανοίξει ένα stream Process τότε η εντολή αυτή κλείνει την stdin, σαν να είχατε πληκτρολογήσει CTRL+D σε ένα τερματικό.

Παράδειγμα:

```
DIM hFile AS File
hFile = OPEN "/etc/passwd" FOR INPUT
...
...
...
CLOSE #hFile
```

PRINT

Τυπώνει μια έκφραση σε ένα stream.

Σύνταξη:

```
PRINT [ # hStream , ] Expression [ { ; | ;; | , } Expression ... ]
[ { ; | ;; | , }  ]
```

Η εντολή PRINT τυπώνει την έκφραση Expression στο ρεύμα # hStream, αν δεν οριστεί ρεύμα τότε χρησιμοποιείται η προκαθορισμένη έξοδος του τερματικού. Η προκαθορισμένη έξοδος μπορεί να ανακατευθυνθεί από την OUTPUT TO.
Η έκφραση Expression μετατρέπεται σε string από την συνάρτηση Str$()

- Αν δεν υπάρχει Ελληνικό ερωτηματικό τότε ένας χαρακτήρας νέας γραμμής τυπώνεται (\n).
- Αν υπάρχει Ελληνικό ερωτηματικό η έκφραση που ακολουθεί τυπώνεται αμέσως μετά.
- Αν υπάρχει διπλό Ελληνικό ερωτηματικό τότε μεταξύ των εκφράσεων τυπώνεται το κενό διάστημα.
- Αν υπάρχει ένα κόμμα ανάμεσα στις εκφράσεις τότε τυπώνεται ένας χαρακτήρας TAB.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project32.

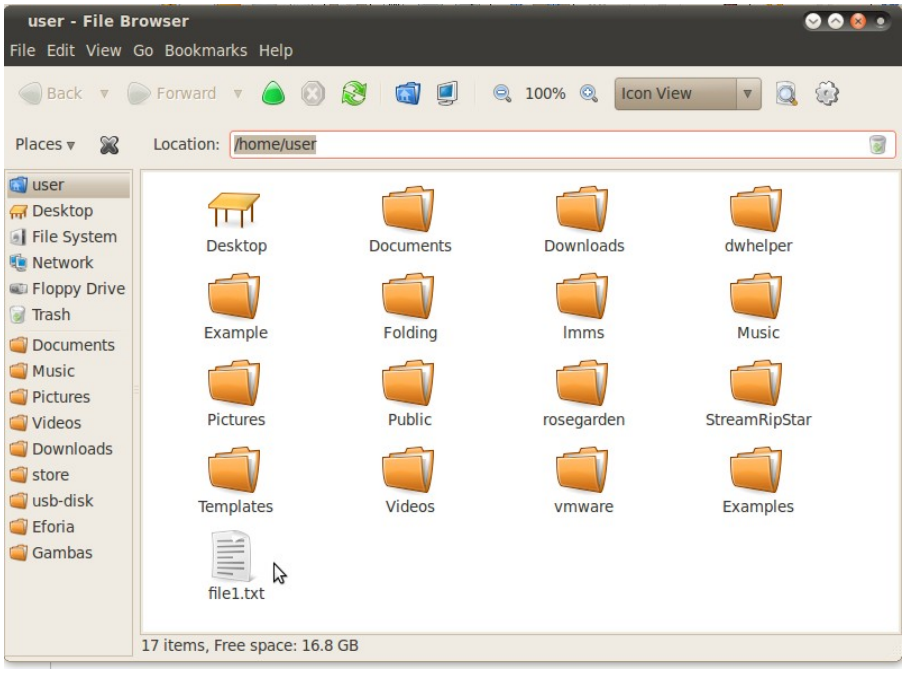
```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  DIM hFile AS File
6  DIM sFile AS String
7
8  sFile = User.Home &/ "file1.txt"
9  hFile = OPEN sFile FOR OUTPUT CREATE
10
11 PRINT #hFile, "I print a line in file1.txt"
12
13 CLOSE #hFile
14
15 END
```

Ανάλυση:
Στην γραμμή 5 ορίζουμε μια μεταβλητή hFile ως ένα αντικείμενο File.
Στην γραμμή 6 ορίζουμε μια αλφαριθμητική μεταβλητή sFile.

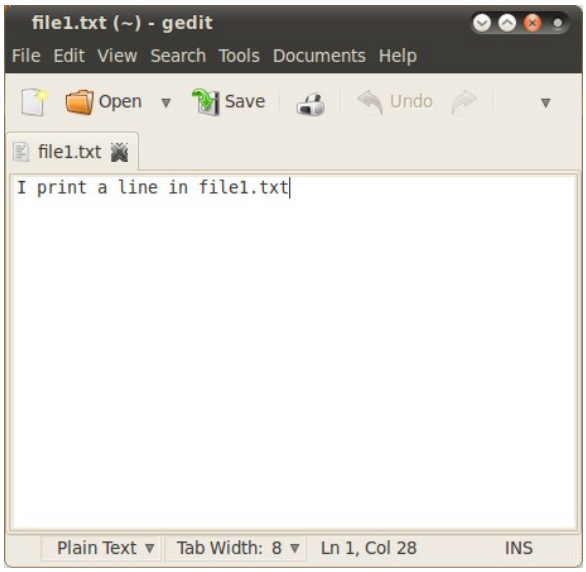
Στην γραμμή 8 εκχωρούμε στην μεταβλητή sFile την διαδρομή του home καταλόγου του χρήστη συν ένα αρχείο file1.txt. Αν για παράδειγμα έχουμε τον χρήστη user τότε η διαδρομή που εκχωρείται στην μεταβλητή είναι “/home/user/file1.txt”.

Στην γραμμή 9 ανοίγουμε το αρχείο file1.txt για να γράψουμε σε αυτό, αν το αρχείο δεν υπάρχει τότε δημιουργείται.
Στην γραμμή 11 γράφουμε στο αρχείο μία γραμμή.
Στην γραμμή 13 κλείνουμε το αρχείο.

Αν πάμε στο home φάκελο του χρήστη τότε θα δούμε το αρχείο file1.txt με την γραμμή τυπωμένη μέσα σε αυτό.



Εικόνα 71: Το δημιουργημένο αρχείο file1.txt στον κατάλογο home του χρήστη



Εικόνα 72: Το αρχείο file1.txt με το περιεχόμενο του

Eof

Επιστρέφει TRUE αν το τέλος του αρχείου έχει φτάσει.

Σύνταξη:

```
bResult = Eof ( [ hStream AS Stream ] ) AS Boolean
```

Όταν εκτελούμε εργασίες σε stream πρέπει να ξέρουμε αν ο εσωτερικός δείκτης έχει φτάσει στο τέλος του αρχείου. Η συνάρτηση αυτή επιστρέφει TRUE αν ο εσωτερικός δείκτης έχει φτάσει στο τέλος του αρχείου.

Αν δεν οριστεί stream τότε η stdin χρησιμοποιείται.

Lof

Επιστρέφει το μήκος του stream.

Σύνταξη:

```
lLength = Lof ( hFile AS Stream ) AS Long
```

Llength, το μήκος του stream σε byte.
Hfile, το stream που θα υπολογιστεί το μήκος του.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project33.

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  DIM hFile AS File
6  DIM sFile AS String
7
8  sFile = User.Home &/ "file1.txt"
9  hFile = OPEN sFile FOR READ
10
11 PRINT Lof(hFile)
12
13 CLOSE #hFile
14
15 END
```

Έξοδος:

```
28
```

Η Lof(hFile) μας επιστρέφει ότι το αρχείο είναι 28 bytes.

ERROR

Τυπώνει μια έκφραση στην προκαθορισμένη έξοδο λαθών (stderr).

Σύνταξη:

```
ERROR Expression [ { ; | ;; | , } Expression ... ] [ { ; | ;; | , }
]
```

Η εντολή αυτή λειτουργεί ακριβώς όπως η PRINT μόνο που τυπώνει εκφράσεις στην προκαθορισμένη έξοδο λαθών (stderr).

FLUSH

Αδειάζει την έξοδο μιας προσωρινής περιοχής αποθήκευσης από ένα stream.

Σύνταξη:

```
FLUSH [ [ # ] Stream ]
```

Αδειάζει την έξοδο από ένα buffered stream.
Εάν δεν δοθεί συγκεκριμένο stream τότε κάθε ανοιχτό stream αδειάζει.

Το να αδειάσετε την είσοδο ενός stream δεν έχει νόημα.

INPUT

Διαβάζει αλφαριθμητικά από ένα stream κειμένου και τα μετατρέπει σε τιμές.

Σύνταξη:

```
INPUT [ # hStream , ] Variable [ , Variable ... ]
```

Διαβάζει από το stream από την θέση του εσωτερικού δείκτη αρχείου.
Μετατρέπει τα στοιχεία που διαχωρίζονται με space ή χαρακτήρα new line με την συνάρτηση Val() προτού εκχωρήσει τις τιμές στις μεταβλητές Variable.

Εάν δεν οριστεί συγκεκριμένο stream τότε χρησιμοποιείται η προκαθορισμένη είσοδος (stdin).

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project34.

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  DIM hFile AS File
6  DIM sFile AS String
7  DIM sString AS String
8
9  sFile = User.Home &/ "file1.txt"
10 hFile = OPEN sFile FOR READ
11
12 WHILE (NOT Eof(hFile))
13   INPUT #hFile, sString
14   PRINT sString
15 WEND
16
17 CLOSE #hFile
18
19 END
```

Έξοδος:

```
I
print
a
line
in
file1.txt
```

Ανάλυση:
Στην γραμμή 5 ορίζεται ένα αντικείμενο τύπου File.
Στις γραμμές 6, 7 ορίζονται μεταβλητές τύπου string

Στην γραμμή 9 ορίζεται η διαδρομή του αρχείου που θα ανοιχτεί.
Στην γραμμή 10 ανοίγεται το αρχείο hFile για ανάγνωση.

Στις γραμμές 12 με 15 έχουμε έναν βρόχο που διαβάζει κάθε λέξη που διαχωρίζεται με κενό από την θέση του εσωτερικού δείκτη του αρχείου. Όταν διαβαστεί ένας χαρακτήρας ο δείκτης τοποθετείται μια θέση εμπρός.

Στην γραμμή 13 διαβάζεται ο χαρακτήρας της θέσης του δείκτη από το stream hFile και εκχωρείται στην μεταβλητή sString.

Στην γραμμή 14 τυπώνεται το περιεχόμενο της μεταβλητής στην stdout.

Αφού ο βρόχος διαβάσει όλο το αρχείο τέλος στην γραμμή 17 κλείνουμε το αρχείο.

LINE INPUT

Διαβάζει μια γραμμή από ένα stream κειμένου.

Σύνταξη:

```
LINE INPUT [ # hStream , ] sVariable
```

Η εντολή αυτή διαβάζει μια ολόκληρη γραμμή από ένα stream κειμένου και την εκχωρεί στην μεταβλητή Variable.

Εάν δεν οριστεί συγκεκριμένο stream τότε χρησιμοποιείται η προκαθορισμένη είσοδος.

Διαβάζεται ολόκληρη η γραμμή εκτός από τον χαρακτήρα τέλους γραμμής.

Δεν θα πρέπει να χρησιμοποιείτε αυτήν την εντολή για να διαβάσετε ένα δυαδικό αρχείο.
Αντί αυτής της εντολής για διάβασμα ενός δυαδικού αρχείου θα πρέπει να χρησιμοποιείται την READ.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project35.

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  DIM hFile AS File
6  DIM sFile AS String
7  DIM sString AS String
8
9  sFile = User.Home &/ "file1.txt"
10 hFile = OPEN sFile FOR READ
11
12 WHILE (NOT Eof(hFile))
13   LINE INPUT #hFile, sString
14   PRINT sString
15 WEND
16
17 CLOSE #hFile
18
19 END
```

Έξοδος:

```
I print a line in file1.txt
```

Ανάλυση:
Το παράδειγμα αυτό είναι το ίδιο με αυτό της εντολής INPUT μόνο που εδώ στην γραμμή 13 χρησιμοποιούμε την LINE INPUT.
Έτσι το αποτέλεσμα είναι να έχουμε στην έξοδο όχι τις λέξεις αλλά μία ολόκληρη γραμμή.

READ

Διαβάζει δυαδικά δεδομένα από ένα stream των οποίων ο τύπος καθορίζεται από την μεταβλητή εκχώρησης.

Σύνταξη:

```
Variable = READ [ # Stream ] AS Datatype

Variable = READ [ # Stream , ] Length
```

Variable, η μεταβλητή που θα εκχωρήσουμε τα δεδομένα
Stream, το stream από το οποίο θα διαβαστούν τα δυαδικά δεδομένα
Datatype, ο τύπος δεδομένων που περιγράφει τα δυαδικά δεδομένα που διαβάζονται.
Length, τα bytes που θα διαβαστούν

Η πρώτη σύνταξη διαβάζει δυαδικά δεδομένα, όπως αυτά που γράφονται από την εντολή WRITE και τα οποία εκχωρεί στην μεταβλητή Variable. Τα δεδομένα έχουν τον τύπο που καθορίζει η έκφραση Datatype.

Η δεύτερη σύνταξη διαβάζει δεδομένα στο αντίστοιχο μήκος και επιστρέφει ένα string.

Η εντολή READ επιστρέφει τους εξής τύπους δεδομένων: NULL, Boolean, Byte, Short, Integer, Long, Pointer, Single, Float, Date, String, Variant, οποιοδήποτε Array ή Collection.

Αν δεν καθοριστεί stream τότε η stdin χρησιμοποιείται.

Για να διαβάσετε δυαδικά δεδομένα από την μνήμη μέσω ενός stream θα πρέπει να χρησιμοποιήσετε την εντολή MEMORY.

Παράδειγμα:
Στο παρακάτω παράδειγμα ανοίγουμε ένα wav αρχείο και διαβάζουμε με την σειρά τις τιμές της επικεφαλίδας του. Οι τιμές θα πρέπει να διαβαστούν με συγκεκριμένο τύπο δεδομένων, άλλες ως Short, άλλες ως Integer, διότι η τιμή τους είναι διαφορετικά bytes.

```
PUBLIC Type AS Short
PUBLIC Channels AS Short
PUBLIC SampleRate AS Integer     ' Hz
PUBLIC BytesPerSecond AS Integer ' Bytes
PUBLIC Alignment AS Short
PUBLIC SampleSize AS Short       ' Bits
PUBLIC FileSize AS Integer       ' Bytes
PUBLIC Duration AS Float         ' Seconds

PUBLIC SUB ButtonRead_Click()
  DIM wav AS File
  Dialog.Filter = ["*.wav", "WAVE music files"]
  IF Dialog.OpenFile() THEN RETURN
  ' We need the file size to calculate the duration
  FileSize = Stat(Dialog.Path).Size
  wav = OPEN Dialog.Path FOR READ
  SEEK #wav, 20 ' The info we want starts 20 bytes into the file
  READ #wav, Type
  READ #wav, Channels
  READ #wav, SampleRate
  READ #wav, BytesPerSecond
  READ #wav, Alignment
  READ #wav, SampleSize
  CLOSE #wav
  ' Calc duration in seconds
  Duration = FileSize / BytesPerSecond
  ' Display results
  PRINT "Type                " & Type
  PRINT "Channels            " & Channels
  PRINT "Sample Rate (Hz)    " & SampleRate
```

```
  PRINT "Bytes Per Second     " & BytesPerSecond
  PRINT "Alignment            " & Alignment
  PRINT "Sample Size (Bytes) " & SampleSize
  PRINT "File Size (Bytes)    " & FileSize
  PRINT "Duration (Seconds)   " & Duration
CATCH
  PRINT Error.Text
END
```

MEMORY

Γράφει δυαδικά δεδομένα σε ένα stream.

Σύνταξη:

```
Stream = MEMORY Pointer FOR [ READ ] [ WRITE ]
```

Stream, το stream που επιστρέφει η εντολή MEMORY
Pointer, ένας δείκτης προς ανάγνωση ή γράψιμο.

- Εάν η λέξη **READ** έχει καθοριστεί, τότε το διάβασμα επιτρέπεται. Η λέξη αυτή είναι προαιρετική αφού για μια θέση μνήμης πάντα επιτρέπεται το διάβασμα.
- Εάν η λέξη **WRITE** έχει καθοριστεί, τότε το γράψιμο επιτρέπεται

Αν προσπαθήσετε να γράψετε σε μια απαγορευμένη περιοχή μνήμης ένα λάθος θα εμφανιστεί και ο διερμηνευτής δεν θα καταρεύσει.

WRITE

Γράφει δυαδικά δεδομένα σε ένα stream.

Σύνταξη:

```
WRITE [ # Stream , ] Expression AS Datatype

WRITE [ # Stream , ] String [ , Length ]
```

Stream, το stream στο οποίο θα γραφτούν τα δυαδικά δεδομένα
Datatype, ο τύπος δεδομένων που περιγράφει τα δυαδικά δεδομένα που γράφονται.
Expression, η έκφραση που θα γραφτεί
Length, τα bytes που θα γραφτούν

Η πρώτη σύνταξη γράφει δυαδικά δεδομένα. Τα δεδομένα έχουν τον τύπο που καθορίζει η έκφραση Datatype.

Η δεύτερη σύνταξη γράφει δεδομένα string στο αντίστοιχο μήκος.

Η εντολή WRITE υποστηρίζει τους εξής τύπους δεδομένων: NULL, Boolean, Byte, Short, Integer, Long, Pointer, Single, Float, Date, String, Variant, οποιοδήποτε Array ή Collection, ή οποιαδήποτε δομή.

Εάν η Expression είναι μια συλλογή (collection), ή διάταξη, ή δομή, τότε τα δεδομένα γράφονται αναδρομικά μέχρι το τέλος της απαρίθμησης τους.

Όταν γράφετε μια δομή, η δομή πρέπει να καθοριστεί ως τύπος δεδομένων Datatype.

Αν ένας μη έγκυρος τύπος δεδομένων γραφτεί, η μια κυκλική αναφορά ανιχνευτεί τότε ένα λάθος εμφανίζεται.

Εάν δεν καθοριστεί stream τότε η stdout χρησιμοποιείται.
Εάν η έκφραση είναι ένα string μπορείτε να ορίσετε το Length που δηλώνει τον αριθμό των bytes που θα γραφτούν.

Αυτή η εντολή χρησιμοποιεί την σειρά των bytes του stream για να γράψει δεδομένα.

Παράδειγμα:
Στο παρακάτω παράδειγμα γράφονται δυαδικά δεδομένα σε ένα αρχείο και μετά διαβάζονται.

```
PUBLIC SUB ButtonWriteBinary_Click()
  DIM filePath AS String
  ' Use a temporary file
  filePath = Temp()
  ' Write binary file
  BinaryWrite(filePath)
  ' Display binary file
  BinaryRead(filePath)
  ' Remove temporary file
  KILL filePath
CATCH
  Message.Error(Error.Text)
END

PRIVATE SUB BinaryWrite(FilePath AS String)
  DIM binaryFile AS File
  DIM i AS Integer = 10
  DIM b AS Byte = 4
  DIM s AS Short = 23
  DIM s1 AS String = "This is string 1"
  DIM s2 AS String = "Another string"
  ' Open as create so we get a new file
  binaryFile = OPEN FilePath FOR CREATE
  WRITE #binaryFile, i
  WRITE #binaryFile, b
  WRITE #binaryFile, s
  WRITE #binaryFile, s1
  WRITE #binaryFile, s2
  CLOSE #binaryFile
END
```

```
PRIVATE SUB BinaryRead(FilePath AS String)
  DIM binaryFile AS File
  DIM i AS Integer
  DIM b AS Byte
  DIM s AS Short
  DIM s1 AS String
  DIM s2 AS String
  ' Read binary file
  binaryFile = OPEN FilePath FOR READ
  READ #binaryFile, i
  READ #binaryFile, b
  READ #binaryFile, s
  READ #binaryFile, s1
  READ #binaryFile, s2
  CLOSE #binaryFile
  ' Display results
  PRINT i
  PRINT b
  PRINT s
  PRINT s1
  PRINT s2
END
```

OUTPUT TO

Ανακατευθύνει την stdout προς ένα stream.

Σύνταξη:

	OUTPUT TO Stream

Για να ανακατευθύνετε την stdout στην προκαθορισμένη έξοδο θα πρέπει να γράψετε

```
OUTPUT TO DEFAULT
```

INPUT FROM

Ανακατευθύνει την stdin να λάβει δεδομένα από ένα stream.

Σύνταξη:

	INPUT FROM Stream

Για να ανακατευθύνετε την stdin στην προκαθορισμένη είσοδο θα πρέπει να γράψετε

```
INPUT FROM DEFAULT
```

ERROR TO

Ανακατευθύνει την stderr προς ένα stream.

Σύνταξη:

```
ERROR TO Stream
```

Για να ανακατευθύνετε την stderr στην προκαθορισμένη είσοδο θα πρέπει να γράψετε

```
ERROR TO DEFAULT
```

PIPE

Ανοίγει έναν αγωγό με όνομα για διάβασμα ή γράψιμο και δημιουργεί ένα stream για αυτό.

Σύνταξη:

```
hStream = PIPE sPipeName FOR [ READ ] [ WRITE ] [ WATCH ]
```

Ένας αγωγός παρέχει ημιαφμίδρομη ροή δεδομένων μεταξύ των διεργασιών στο ίδιο σύστημα. Τα δεδομένα ρέουν προς μία κατεύθυνση.

Ένα σημαντικό χαρακτηριστικό γνώρισμα ενός σωλήνα είναι ότι τα δεδομένα που διατρέχουν το μέσο επικοινωνίας είναι παροδικά - τα δεδομένα που διαβάζονται μια φορά από το περιγραφέα ανάγνωσης δεν μπορούν να διαβαστούν πάλι. Επίσης, εάν γράψουμε δεδομένα συνεχώς στον περιγραφέα εγγραφής, κατόπιν θα είμαστε σε θέση να διαβάσουμε τα δεδομένα με την σειρά στην οποία τα δεδομένα γράφτηκαν.

Ο αγωγός θα πρέπει να ανοίξει και στις δύο άκρες προτού διαβάσετε ή γράψετε δεδομένα.
Αν ανοίξετε έναν αγωγό για διάβασμα φυσιολογικά θα μπλοκαριστεί μέχρι μια άλλη διεργασία ανοίξει τον ίδιο αγωγό για γράψιμο.

Η Gambas υλοποιεί το πρότυπο FIFO (First in – First out) για τους αγωγούς. Για περισσότερα δείτε στην τεκμηρίωση του Linux MKFIFO.

- Αν οριστεί η λέξη READ τότε ο αγωγός ανοίγετε για διάβασμα δεδομένων.
- Αν οριστεί η λέξη WRITE τότε ο αγωγός ανοίγετε για γράψιμο δεδομένων.
- Αν οριστεί η λέξη WATCH τότε ο αγωγός παρακολουθείτε από τον διερμηνευτή.
    - Αν διαβαστεί ένα byte καλείται το συμβάν `Pipe_Read()`.
    - Αν γραφτεί ένα byte καλείται το συμβάν `Pipe_Write()`.

Εάν ανοιχτεί επιτυχώς ένας αγωγός ένα αντικείμενο stream εκχωρείται στην μεταβλητή hStream.

Τα αντικείμενα pipe streams ποτέ δεν έχουν buffer.

Παράδειγμα:
Το παρακάτω παράδειγμα δημιουργεί έναν αγωγό για διάβασμα. Στην συνέχεια μπορούμε σε ένα τερματικό να δώσουμε την εντολή "`ls > /tmp/FIFO1`" και να δούμε τα περιεχόμενα του φακέλου στην κονσόλα της Gambas.
Το παρακάτω παράδειγμα είναι το Project36.

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  ' Prints the messages sent to a pipe
6
7  Dim hFile As File
8  Dim sLine As String
9
10 hFile = Pipe "/tmp/FIFO1" For Read
11
12 Do
13   Line Input #hFile, sLine
14   If hFile.EndOfFile Then Break
15   Print sLine
16 Loop
17
18 END
```

LOCK

Κλειδώνει ένα ανοιχτό stream.

Σύνταξη:

```
LOCK [ # ] hStream
```

Όταν ένα stream κλειδώνει από μία διεργασία δεν μπορεί άλλη διεργασία να γράψει δεδομένα σε αυτό.

Αν το stream είναι ήδη κλειδωμένο από άλλη διεργασία η εντολή αποτυγχάνει.

Παράδειγμα:

```
1  ' Try to lock the file
2  TRY LOCK #TheFile
3  IF ERROR THEN
4    PRINT "File is already locked. Try again later."
5    RETURN
6  ENDIF
7  ' File is locked, you can do the job now!
8  ...
9
```

UNLOCK

Ξεκλειδώνει ένα ανοιχτό stream.

Σύνταξη:

	UNLOCK [**#**] Stream

SEEK

Αλλάζει την θέση του δείκτη εντός του αρχείου ενός stream.

Σύνταξη:

	SEEK [**#**] Stream **,** Position

Ορίζει την θέση του δείκτη εντός ενός αρχείου για την επόμενη ενέργεια διαβάσματος / εγγραφής.
Αν το Position είναι αρνητικός αριθμός ο δείκτης μετακινείται σχετικά από το τέλος του αρχείου.
Για να μετακινήσετε τον δείκτη του αρχείου πέρα από το τέλος του πρέπει να χρησιμοποιήσετε την Lof().

Παράδειγμα:

```
1  ' Move to the beginning of the file
2  SEEK #hFile, 0
3
4  ' Move after the end of the file
5  SEEK #hFile, Lof(#hFile)
6
7  ' Move 150 bytes before the end of the file
8  SEEK #hFile, -150
```

Seek

Λαμβάνει την θέση του δείκτη εντός ενός αρχείου stream.

Σύνταξη:

```
Position = Seek ( Stream )
```

Επιστρέφει την τρέχουσα θέση του δείκτη εντός ενός stream
Ο τύπος δεδομένων του Position είναι ένας Long.

Σημείωση: Πολλά είδη stream όπως τα Process, Socket κτλ δεν έχουν δείκτη.

Συναρτήσεις συμβολοσειρών (String Functions)

Οι συναρτήσεις String μεταχειρίζονται μόνο ASCII χαρακτήρες.
Για λεπτομέρειες για τον πίνακα ASCII δείτε στο Παράρτημα Β.
Οι συμβολοσειρές ονομάζονται και αλφαριθμητικά.

Οι συναρτήσεις αυτές είναι:

`Asc`	Επιστρέφει τον κωδικό ASCII ενός χαρακτήρα σε ένα string.
`Chr$`	Επιστρέφει έναν χαρακτήρα από έναν κωδικό ASCII.
`Comp`	Συγκρίνει δύο strings.
Html$	Εσωκλείει σε διπλά εισαγωγικά ένα string για να τοποθετηθεί σε μια σελίδα HTML.
`InStr`	Αναζητά ένα string εντός άλλου string.
`LCase$`	Μετατρέπει ένα string σε γραφή με μικρά γράμματα.
`Left$`	Επιστρέφει τους πρώτους χαρακτήρες από ένα string.
`Len`	Επιστρέφει το μήκος ενός string.
`LTrim$`	Αφαιρεί τους χαρακτήρες κενού από τα αριστερά ενός string.
`Mid$`	Επιστρέφει μέρος από ένα string.
Quote$	Εσωκλείει σε διπλά εισαγωγικά ένα string
`Replace$`	Αντικαθιστά σε ένα string μέρος με ένα άλλο string.
`Right$`	Επιστρέφει τους τελευταίους χαρακτήρες από ένα string.
`RInStr`	Αναζητά ένα string σε ένα άλλο string από τα δεξιά του.
`RTrim$`	Αφαιρεί τους χαρακτήρες κενού από τα δεξιά ενός string.
`Scan`	Διαχωρίζει ένα string βάσει μιας έκφρασης.
`Space$`	Επιστρέφει ένα string το οποίο περιέχει μόνο ένα κενό.
`Split`	Διαιρεί ένα string σε μέρη.
`String$`	Επιστρέφει το ίδιο string ενωμένο πολλές φορές.

`Subst$`	Αντικαθιστά ένα string με ένα μοτίβο.
`Trim$`	Αφαιρεί τα κενά από ένα string.
`UCase$`	Μετατρέπει ένα string σε μορφή γραφής κεφαλαίων.
Unquote$	Αναιρεί τον εγκλεισμό σε διπλά εισαγωγικά ενός string.

Asc

Επιστρέφει τον κωδικό ASCII ενός χαρακτήρα σε ένα string.

Σύνταξη:

```
Code = Asc ( Input AS String [ , Position AS Integer ] ) AS Integer
```

Η Asc δέχεται ένα string, Input και έναν ακέραιο Position. Ο ακέραιος δείχνει την θέση του χαρακτήρα από το string Input για τον οποίο θα υπολογιστεί η αντίστοιχη τιμή του από τον πίνακα ASCII.

Αν δεν οριστεί ο ακέραιος Position τότε επιστρέφεται ο κωδικός ASCII για τον πρώτο χαρακτήρα του string.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project37.

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  PRINT Asc("This is a string", 3)
6
7  END
```

Έξοδος:

```
105
```

Πράγματι ο χαρακτήρας “i” έχει κωδικό ASCII 105 και είναι ο τρίτος χαρακτήρας στο string “This is a string”.

Chr$

Επιστρέφει έναν χαρακτήρα από έναν κωδικό ASCII.

Σύνταξη:

```
Character = Chr$ ( Code AS Integer ) AS String

Character = Chr ( Code AS Integer ) AS String
```

Η συνάρτηση λαμβάνει έναν ακέραιο που είναι η τιμή του κωδικού ASCII.
Τιμές code μεγαλύτερες του 127 δεν έχουν νόημα και καλό θα ήταν όταν χρησιμοποιείτε διαφορετικά σετ χαρακτήρων όπως ISO 8859-1 να μετατρέπετε το string σε ASCII με την συνάρτηση Conv$().

Οι εκτυπώσιμοι χαρακτήρες στο σετ χαρακτήρων ASCII είναι μεταξύ των τιμών 32 και 126.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project38.

```
1  PUBLIC SUB Main()
2
3  DIM i AS Integer
4
5  FOR i = 32 TO 126
6    PRINT Chr$(i);;
7  NEXT
8
9  END
```

Έξοδος:

```
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJK
LMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuv
wxyz{|}~
```

Ανάλυση:
Στο παράδειγμα αυτό τυπώνουμε με την βοήθεια ενός βρόχου τους χαρακτήρες με κωδικούς από το 32 μέχρι το 126 από τον πίνακα ASCII.

Comp

Συγκρίνει δύο strings.

Σύνταξη:

```
iResult = Comp ( String1 AS String , String2 AS String [ , Mode AS Integer ] ) AS Integer
```

Η συνάρτηση δέχεται δύο string τα String1 και String2 και τα συγκρίνει.
Αν τα strings είναι ίσα επιστρέφει 0
Αν το String1 είναι μεγαλύτερο του String2 επιστρέφει 1
Αν το String2 είναι μεγαλύτερο του String1 επιστρέφει -1

Η παράμετρος Mode ορίζει να η σύγκριση θα είναι δυαδικού τύπου, gb.Binary ή τύπου κειμένου για μια σύγκριση κειμένου χωρίς ευαισθησία στα κεφαλαία-μικρά.

Αν δεν οριστεί τύπος σύγκρισης ο προκαθορισμένος τύπος είναι gb.Binary.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project39.

```
1 PUBLIC SUB Main()
2
3 PRINT Comp("This is", "This is a string") 'return -1, String2 is greater
4 PRINT Comp("This is", "This is a string", gb.Text) 'return -1, String2 is greater
5 PRINT Comp("This is greater string", "This is a string") 'return 1, String1 is greater
6
7 END
```

Έξοδος:

```
-1
-1
1
```

Html$

Εσωκλείει ένα string σε εισαγωγικά για να τοποθετηθεί σε μια σελίδα HTML.

Σύνταξη:

```
HtmlString = Html ( String )

HtmlString = Html$ ( String )
```

InStr

Αναζητά ένα string εντός άλλου string.

Σύνταξη:

```
Position = InStr ( String AS String , Substring AS String [ , Start
AS Integer , Comparison AS Integer ] )
```

Η συνάρτηση επιστρέφει την θέση στην οποία βρίσκεται το Substring όταν αυτό βρεθεί πρώτη φορά μέσα στο String.

Αν οριστεί το Start τότε η σύγκριση αρχίζει από την θέση αυτή.
Ο τύπος της σύγκρισης μπορεί να είναι δυαδικός ή κειμένου, gb.Binary, gb.Text.

Αν δεν οριστεί τύπος σύγκρισης ο προκαθορισμένος είναι gb.Binary.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project40.

```
1 PUBLIC SUB Main()
2
3 PRINT InStr("This is a string", "is")
4
5 END
```

Έξοδος:

```
3
```

LCase$

Μετατρέπει ένα string σε γραφή με μικρά γράμματα.

Σύνταξη:

```
sResult = LCase$ ( sExpr AS String ) AS String

sResult = Lower$ ( sExpr AS String ) AS String
```

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project41

```
1 ' Gambas module file
2
3 PUBLIC SUB Main()
4
5 PRINT LCase$("THIS IS A STRING")
6
7 END
```

Έξοδος:

```
this is a string
```

Left$

Επιστρέφει τους πρώτους χαρακτήρες από ένα string.

Σύνταξη:

```
sResult = Left$ ( sArg AS String [ , iLength AS Integer ] ) AS
String

sResult = Left ( sArg AS String [ , iLength AS Integer ] ) AS
String
```

Επιστρέφει τους πρώτους iLength σε αριθμό, χαρακτήρες από το string sArg.

Αν δεν οριστεί ο ακέραιος iLength επιστρέφεται ο πρώτος χαρακτήρας.
Αν ο iLength είναι αρνητικός επιστρέφεται όλο το string μείον τους iLength χαρακτήρες από το τέλος.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project42.

```
1  PUBLIC SUB Main()
2
3  PRINT Left$("This is a string")
4  PRINT Left$("This is a string", 4)
5  PRINT Left$("This is a string", - 1)
6  PRINT Left$("This is a string", - 2)
7
8  END
```

Έξοδος:

```
T
This
This is a strin
This is a stri
```

Len

Επιστρέφει το μήκος ενός string.

Σύνταξη:

```
iLength = Len ( sArg AS String ) AS Integer
```

Επιστρέφει τα bytes που μετριούνται σε ένα string.
Ένας χαρακτήρας ASCII με κωδικό από το 0 έως το 127 καταλαμβάνει 1 byte.
Ένας χαρακτήρας UTF-8 καταλαμβάνει 2 ή περισσότερα bytes.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project43

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  PRINT Len("A string")
6  PRINT Len("Ένα κείμενο")
7
8  END
```

Έξοδος:

```
8
21
```

LTrim$

Αφαιρεί τους χαρακτήρες κενού από τα αριστερά ενός string.

Σύνταξη:

```
sResult = LTrim$ ( sExpr AS String ) AS String

sResult = LTrim ( sExpr AS String ) AS String
```

Η συνάρτηση αφαιρεί οποιοδήποτε χαρακτήρα ASCII με κωδικό μικρότερο από 32.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project44.

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  PRINT " This is a string "
6  PRINT LTrim(" This is a string ")
7
8
9  END
```

Έξοδος:

```
This is a string
This is a string
```

Mid$

Επιστρέφει μέρος από ένα string.

Σύνταξη:

```
Result = Mid$ ( sString AS String , iStart AS Integer [ , iLength
AS Integer ] ) AS String

sResult = Mid ( sString AS String , iStart AS Integer [ , iLength
AS Integer ] ) AS String
```

Η συνάρτηση επιστρέφει ένα μέρος από το sString το οποίο ξεκινάει από την θέση iStart και έχει μήκος σε χαρακτήρες iLength.

Αν δεν οριστεί iLength επιστρέφεται όλο το string από την θέση iStart και μετά.
Αν το iLength είναι αρνητικό επιστρέφεται το string από την θέση iStart και μετά μείον τους χαρακτήρες iLength από το τέλος.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project45

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  PRINT Mid("This is a string", 2, 3)
6  PRINT Mid("This is a string", 1, -3)
7
8  END
```

Έξοδος:

```
his
This is a str
```

Quote$

Εσωκλείει ένα string σε εισαγωγικά.

Σύνταξη:

```
Quoted string = Quote ( String )

Quoted string = Quote$ ( String )
```

Replace$

Αντικαθιστά σε ένα string μέρος με ένα άλλο string.

Σύνταξη:

```
Result = Replace$ ( String , Pattern , ReplaceString [ , Comparison ] )

Result = Replace ( String , Pattern , ReplaceString [ , Comparison ] )
```

Η συνάρτηση αντικαθιστά στο String τους χαρακτήρες Pattern με το ReplaceString και επιστρέφει το τελικό αποτέλεσμα string.

Εάν το String είναι κενό τότε κενό string επιστρέφεται.
Εάν το Pattern είναι κενό τότε String επιστρέφεται.

Η σύγκριση μπορεί να γίνει δυαδικά ή με τύπο κειμένου θέτοντας την τιμή Comparison σε gb.Binary ή gb.Text αντίστοιχα.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project46

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  PRINT Replace("This is a string", "is", "IS")
6
7  END
```

Έξοδος:

```
ThIS IS a string
```

Right$

Επιστρέφει τους τελευταίους Length χαρακτήρες από ένα string.

Σύνταξη:

```
Result = Right$ ( String AS String [ , Length AS Integer ] )

Result = Right ( String AS String [ , Length AS Integer ] )
```

Εάν το Length δεν οριστεί ο τελευταίος χαρακτήρας επιστρέφεται.
Εάν το Length είναι αρνητικός αριθμός επιστρέφεται όλο το string εκτός από τους πρώτους Length χαρακτήρες.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project47

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  PRINT Right$("This is a string", 5)
6  PRINT Right$("This is a string", -5)
7
8  END
```

Έξοδος:

```
tring
is a string
```

RInStr

Αναζητά ένα string σε ένα άλλο string από τα δεξιά του.

Σύνταξη:

```
Position = RInStr ( String AS String , Substring AS String [ ,
Start AS Integer , Comparison AS Integer ] )
```

Επιστρέφει την θέση του πρώτου ευρισκόμενου Substring εντός ενός String ξεκινώντας την εύρεση από την θέση Start.

Αν δεν οριστεί θέση εκκίνησης Start η αναζήτηση ξεκινά από τον πρώτο χαρακτήρα. Μπορεί να οριστεί η μέθοδος σύγκρισης σε δυαδικό τύπο αναζήτησης ή τύπο κειμένου, gb.Binary και gb.Text αντίστοιχα.

Αν δεν βρεθεί το Substring τότε επιστρέφεται 0.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project48

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  PRINT RInStr("This is a string", "string")
6  PRINT RInStr("This is a string", "nothing")
7
8  END
```

Έξοδος:

```
11
0
```

RTrim$

Αφαιρεί τους χαρακτήρες κενού από τα δεξιά ενός string.

Σύνταξη:

```
sResult = RTrim$ ( sExpr AS String ) AS String

sResult = RTrim ( sExpr AS String ) AS String
```

Ένας κενός χαρακτήρας είναι οποιοσδήποτε χαρακτήρας που ο κωδικός ASCII αυτού είναι μικρότερος του 32.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project49

```
1 ' Gambas module file
2
3 PUBLIC SUB Main()
4
5 PRINT RTrim("This is a string  ")
6
7 END
```

Έξοδος:

```
This is a string
```

Scan

Διαχωρίζει ένα string βάσει μιας έκφρασης.

Σύνταξη:

```
String Array = Scan ( String AS String , Pattern AS String )
```

Η συνάρτηση επιστρέφει έναν πίνακα string.
Βρίσκει τα strings που ταιριάζουν βάση μιας έκφρασης Pattern και επιστρέφει όλα τα strings

που συνδέονται με ένα χαρακτήρα γενικού τύπου *.

Το Pattern δημιουργείται σύμφωνα με τους παρακάτω κανόνες.

Γενικός χαρακτήρας	Ταίριασμα
*	Οποιοσδήποτε αριθμός ή χαρακτήρας.
?	Ένας χαρακτήρας.
[abc]	Οποιοσδήποτε χαρακτήρας μεταξύ αυτών στις αγκύλες.
[x-y]	Οποιοσδήποτε χαρακτήρας μεταξύ του εύρους τιμών.
[^x-y]	Οποιοσδήποτε χαρακτήρας που δεν είναι στο εύρος τιμών.
space	Οποιοσδήποτε χαρακτήρας με κωδικό ASCII μικρότερο του 32.
\	Ο χαρακτήρας \ αποτρέπει ο επόμενος χαρακτήρας να θεωρηθεί γενικός χαρακτήρας.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project50

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  DIM sString AS String = "This is a string"
6  DIM sArray AS String[]
7  DIM element AS String
8
9  sArray = Scan(sString, "*")
10
11 FOR EACH element IN sArray
12   PRINT element
13 NEXT
14
15 END
```

Έξοδος:

```
This is a string
```

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project51

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  DIM sString AS String = "This is a string"
6  DIM sArray AS String[]
7  DIM element AS String
8
9  sArray = Scan(sString, "?[a-z]*")
10
11 FOR EACH element IN sArray
12   PRINT element
13 NEXT
14
15 END
```

Έξοδος:

```
is is a string
```

Ανάλυση:
Στην γραμμή 5 δηλώνεται μια μεταβλητή τύπου string με το κείμενο “This is a string”.
Στην γραμμή 6 δηλώνεται ένας δυναμικός πίνακας string.
Στην γραμμή 7 δηλώνεται μια μεταβλητή string για την απαρίθμηση του πίνακα με τον βρόχο FOR ... EACH.

Στην γραμμή 9 έχουμε την έκφραση Scan().
Στο πρώτο παράδειγμα η sArray = Scan(sString, "*") επιστρέφει όλους τους χαρακτήρες του string Sstring.
Στο δεύτερο παράδειγμα η sArray = Scan(sString, "?[a-z]*") επιστρέφει τους χαρακτήρες που είναι μικρά γράμματα στο εύρος a-z και οτιδήποτε άλλο.

Στις γραμμές 11-13 έχουμε τον βρόχο FOR .. EACH και τυπώνονται τα στοιχεία του πίνακα που επιστρέφει η Scan().

Space$

Επιστρέφει ένα string το οποίο περιέχει μόνο το κενό τόσες φορές όσο ο ακέραιος Length.

Σύνταξη:

```
String = Space$ ( Length AS Integer )
```

Split

Διαιρεί ένα string σε μέρη.

Σύνταξη:

```
StringArray = Split ( String AS String [ , Separators AS String ,
Escape AS String , IgnoreVoid AS Boolean , KeepEscape AS Boolean ]
)
```

Διαιρεί ένα String με διαχωριστικό το Separators και χαρακτήρες διαφυγής το Escape.

String, είναι το string που θα διαιρεθεί.
Separators, είναι οι χαρακτήρες βάσει των οποίων θα διαιρεθεί το string.
Escape, είναι οι χαρακτήρες διαφυγής. Αν ένας χαρακτήρας Separator είναι μεταξύ δύο χαρακτήρων Escape αγνοείται στην διαδικασία διαχωρισμού του string. Αν έχουν οριστεί δύο χαρακτήρες διαφυγής τότε ο πρώτος είναι ο χαρακτήρας εκκίνησης και ο δεύτερος είναι ο χαρακτήρας τέλους.
IgnoreVoids, αν οριστεί η τιμή αυτή σε TRUE τότε η συνάρτηση δεν επιστρέφει το κενό.
KeepEscape, αν οριστεί η τιμή αυτή σε TRUE τότε διατηρούνται οι χαρακτήρες διαφυγήςστο επιστρεφόμενο string.

Η συνάρτηση χωρίζει το string σε μέρη που τα εκχωρεί σε έναν πίνακα.
Οι χαρακτήρες Separators και Escape δεν επιστρέφονται στα μέρη που εκχωρούνται στον πίνακα.
Εξ ορισμού ο χαρακτήρας διαχωρισμού είναι το κόμμα και δεν υπάρχουν χαρακτήρες διαφυγής.

Αν το string περιέχει χαρακτήρες διαφυγής οι χαρακτήρες αυτοί θα πρέπει να εξαιρεθούν με τον χαρακτήρα \

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project52

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  DIM sString AS String = "This is a string"
6  DIM sArray AS String[]
7  DIM element AS String
8
9  sArray = Split(sString, "i")
10
11 FOR EACH element IN sArray
12   PRINT element
13 NEXT
14
15 END
```

Έξοδος:

```
Th
s
s a str
ng
```

Ανάλυση:
Στην γραμμή 5 δηλώνεται μια μεταβλητή τύπου string με το κείμενο “This is a string”.
Στην γραμμή 6 δηλώνεται ένας δυναμικός πίνακας string.
Στην γραμμή 7 δηλώνεται μια μεταβλητή string για την απαρίθμηση του πίνακα με τον βρόχο FOR ... EACH.

Στην γραμμή 9 διαχωρίζεται το sString βάσει του χαρακτήρα i.

Στις γραμμές 11-13 τυπώνονται στην stdout τα στοιχεία του πίνακα.

Όπως βλέπουμε το string έχει χωριστεί σε μέρη και ο χαρακτήρας διαχωρισμού δεν τυπώνεται στα μέρη που έχουν εκχωρηθεί στον πίνακα.

String$

Επιστρέφει το ίδιο string ενωμένο πολλές φορές.

Σύνταξη:

```
String = String$ ( Length AS Integer , Pattern AS String )
```

Επιστρέφει ένα string Length φορές ένα Pattern

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project53

```
1 ' Gambas module file
2
3 PUBLIC SUB Main()
4
5 PRINT String$(5, "@")
6
7 END
```

Έξοδος:

```
@@@@@
```

Subst$

Αντικαθιστά ένα string με ένα μοτίβο.

Σύνταξη:

```
Result = Subst$ ( Pattern , ReplaceString [ , ReplaceString ] )

Result = Subst ( Pattern , ReplaceString [ , ReplaceString ] )
```

Αντικαθιστά σε ένα string τα &1, &2, &3 κτλ με το πρώτο, δεύτερο, τρίτο κτλ string αντικατάστασης.
Αφού γίνει η αντικατάσταση η συνάρτηση επιστρέφει το τελικό string.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project54.

```
1 ' Gambas module file
2
3 PUBLIC SUB Main()
4
5 PRINT Subst$("This is a &1 with a &2 and a &3", "replace1",
  "replace2", "replace3")
6
7 END
```

Έξοδος:

```
This is a replace1 with a replace2 and a replace3
```

Trim$

Αφαιρεί τα κενά από ένα string.

Σύνταξη:

```
sResult = Trim$ ( sExpr AS String ) AS String

sResult = Trim ( sExpr AS String ) AS String
```

Η συνάρτηση αφαιρεί τα κενά από τα δεξιά και τα αριστερά ενός string.
Κενό θεωρείται οποιοσδήποτε χαρακτήρας με κωδικό ASCII κάτω από 32.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project55.

```
1 ' Gambas module file
2
3 PUBLIC SUB Main()
4
5 PRINT Trim$("  This is a string  ")
6
7 END
```

Έξοδος:

```
This is a string
```

UCase$

Μετατρέπει ένα string σε μορφή γραφής κεφαλαίων.

Σύνταξη:

```
Result = UCase$ ( String )

Result = Upper$ ( String )
```

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project56.

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  PRINT Upper$("This is a string")
6
7  END
```

Έξοδος:

```
THIS IS A STRING
```

Unquote$

Αναιρεί τον εγκλεισμό σε διπλά εισαγωγικά ενός string.

Σύνταξη:

```
Result = UCase$ ( String )

Result = Upper$ ( String )
```

Διάφορες άλλες συναρτήσεις (Miscellaneous Functions)

Μερικές συναρτήσεις που δεν εντάσσονται σε κάποια κατηγορία περιγράφονται σε αυτό το μέρος του βιβλίου.

Τέτοιες συναρτήσεις είναι:

`Eval`	Υπολογίζει μια έκφραση.
`Inline Arrays`	Δημιουργεί και επιστρέφει έναν πίνακα.
`Inline Collections`	Δημιουργεί και επιστρέφει μια συλλογή.

Eval

Υπολογίζει μια έκφραση.

Σύνταξη:

```
Value = Eval ( Expression AS String [ , Context AS Collection ] )
AS Variant
```

Expression, είναι η έκφραση που θα υπολογιστεί
Context, προαιρετική παράμετρος. Είναι μια συλλογή η οποία θα περιέχει την τιμή κάθε μη δηλωμένου συμβόλου της έκφρασης.

Παράδειγμα:

```
1  DIM Context AS New Collection
2
3  Context["X"] = 2
4  Context["Y"] = "Gambas"
5
6  PRINT Eval("X * Len(Y)", Context)
```

Έξοδος:

```
12
```

Για να υπολογίσετε μια έκφραση εκχώρησης θα πρέπει να χρησιμοποιήσετε την λέξη **LET**.
Σε αυτήν την περίπτωση η τιμή εκχώρησης επιστρέφεται.
Η έκφραση εκχώρησης μπορεί να χρησιμοποιήσει οποιοδήποτε τελεστή εκχώρησης αλλά όχι συναρτήσεις που επιστρέφουν αντικείμενα όπως η **OPEN**, **NEW** κτλ.

Παράδειγμα:

```
Dim Context AS New Collection

Context["X"] = "Liberation Serif"

Print Application.Font.ToString()
Print Eval("Let Application.Font = Font[X]", Context)
Print Application.Font.ToString()
```

Έξοδος:

```
Sans,10
(Font 0x8fae9e8)
Liberation Serif,10
```

Inline Arrays

Δημιουργεί και επιστρέφει έναν πίνακα.

Σύνταξη:

```
Array = [ Expression [ , ... ] ]
```

Ο τύπος δεδομένου του πρώτου στοιχείου καθορίζει τον τύπο δεδομένων του πίνακα.
Τα υπόλοιπα στοιχεία του πίνακα μετατρέπονται στο τύπο δεδομένων του πρώτου στοιχείου.

Αν τα στοιχεία δεν είναι όλα του ίδιου τύπου δεδομένων τότε ο πίνακας είναι τύπου Variant[].

Παράδειγμα:

```
PRINT Object.Type([ 2.4, 3, 3.2 ])
PRINT Object.Type([ "a string", 3, 3.2 ])
```

Έξοδος:

```
Float[]
Variant[]
```

Inline Collections

Δημιουργεί και επιστρέφει μια συλλογή.

Σύνταξη:

```
Collection = [ Key : Expression [ , ... ] ]
```

Μια συλλογή (Collection) είναι ένας πίνακας με κλειδιά strings και στοιχεία variant.
Είναι με άλλα λόγια ένας πίνακας κατακερματισμού (hash table).
Ο κατακερματισμός είναι μια τεχνική που χρησιμοποιείται για την βελτιστοποίηση της ταχύτητας ανεύρεσης μιας πληροφορίας που περιέχεται σε μια δομή δεδομένων.

Εάν ένα κλειδί είναι NULL τότε το στοιχείο αυτό αφαιρείται από τον πίνακα.
Ο πίνακας είναι δυναμικός και μεγαλώνει σε μέγεθος όσο στοιχεία προστίθενται σε αυτόν.

Παράδειγμα:

```
1  Dim cCol As Collection
2
3  cCol = [ "A": 1, "B": 2, "C": 3 ]
4  Print cCol["B"]
```

Έξοδος:

```
2
```

Διαχείριση εξωτερικών συναρτήσεων (External Function Management)

Η Gambas μας δίνει την δυνατότητα να καλέσουμε συναρτήσεις που χρησιμοποιεί η γλώσσα C. Συναρτήσεις της γλώσσας C βρίσκονται σε αρχεία του συστήματος που ονομάζονται βιβλιοθήκες, αυτές οι συναρτήσεις αποτελούν εξωτερικές συναρτήσεις για την Gambas. Έτσι μπορούμε να έχουμε μια μεγάλη ποικιλία συναρτήσεων με μεγάλη δύναμη και έλεγχο στο υπολογιστικό σύστημα.

Οι συναρτήσεις που διαχειρίζονται τις εξωτερικές συναρτήσεις είναι οι εξής:

LIBRARY	Δηλώνει την βιβλιοθήκη όπου η εξωτερική συνάρτηση βρίσκεται.
EXTERN	Δηλώνει μια εξωτερική συνάρτηση.
Alloc	Δεσμεύει ένα μέρος μνήμης.
Free	Αποδεσμεύει ένα μέρος μνήμης που είχε δεσμευτεί με την Alloc.
Realloc	Μικραίνει ή μεγαλώνει ένα τμήμα μνήμης που έχει δεσμευτεί με την συνάρτηση Alloc.
StrPtr	Επιστρέφει ένα αντίγραφο ενός μηδενικού string που βρίσκεται σε μια διεύθυνση μνήμης.
VarPtr	Επιστρέφει έναν δείκτη που δείχνει στο περιεχόμενο μιας μεταβλητής.
Boolean@	Επιστρέφει την τιμή Boolean που βρίσκεται σε μια διεύθυνση μνήμης.
Byte@	Επιστρέφει την τιμή Byte που βρίσκεται σε μια διεύθυνση μνήμης.
Float@	Επιστρέφει την τιμή Float που βρίσκεται σε μια διεύθυνση μνήμης.
Integer@	Επιστρέφει την τιμή Integer που βρίσκεται σε μια διεύθυνση μνήμης.
Long@	Επιστρέφει την τιμή Long που βρίσκεται σε μια διεύθυνση μνήμης.
Pointer@	Επιστρέφει την τιμή Pointer που βρίσκεται σε μια διεύθυνση μνήμης.
Short@	Επιστρέφει την τιμή Short που βρίσκεται σε μια διεύθυνση μνήμης.
Single@	Επιστρέφει την τιμή Single που βρίσκεται σε μια διεύθυνση μνήμης.
String@	Επιστρέφει την τιμή String που βρίσκεται σε μια διεύθυνση μνήμης.

Προτού δούμε τις συναρτήσεις διαχείρισης εξωτερικών συναρτήσεων θα πρέπει να αναφέρουμε κάποια πράγματα για το πώς η γλώσσα C βλέπει τις συναρτήσεις και τα δεδομένα. Και αυτό γίνεται διότι οι συναρτήσεις διαχείρισης εξωτερικών συναρτήσεων της Gambas θα πρέπει να εξομοιώνει στην γλώσσα BASIC τα χαρακτηριστικά της γλώσσας C.

Η μνήμη και οι δείκτες στην γλώσσα C

Η γλώσσα C 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 C 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.

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.

RAM	**Διεύθυνση**	1567	**1568**	1569	1570	1571	1572
	Δεδομένα		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** δλδ 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 που είδαμε σε προηγούμενο κεφάλαιο 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 C 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 λάθους 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 σε 32 bit συστήματα θα είναι μια μεταβλητή Integer και σε 64 bit συστήματα θα είναι μια μεταβλητή τύπου Long στην γλώσσα προγραμματισμού της Gambas.

Τύπος δεδομένων

Τύπος δεδομένων	Περιγραφή	Προκαθορισμένη τιμή	Μέγεθος στην μνήμη
Δείκτης (Pointer)	Μια διεύθυνση μνήμης.	0	4 bytes σε 32 bits συστήματα, 8 bytes σε 64 bits συστήματα.

Οι δείκτες στην Gambas3

Η δήλωση ενός δείκτη γίνεται με τον εξής τρόπο.

Σύνταξη:

```
DIM pVar AS Pointer
```

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

Στην Gambas η προσπέλαση ενός δείκτη γίνεται χρησιμοποιώντας το όνομα της μεταβλητής.
Για να δεσμεύσουμε ένα μέρος μνήμης χρησιμοποιούμε την εντολή Alloc().
Αφού δεσμεύσουμε μια διεύθυνση μνήμης στην συνέχεια δημιουργούμε ένα stream μνήμης χρησιμοποιώντας τον δείκτη αυτόν.
Πριν διαβάσουμε ή γράψουμε σε ένα stream μνήμης θα πρέπει να δημιουργήσουμε αυτό με την εντολή:

Stream = **MEMORY** Pointer **FOR** [**READ**] [**WRITE**]

η οποία δημιουργεί ένα stream μνήμης.
Στην συνέχεια μπορούμε να διαβάσουμε και να γράψουμε στην μνήμη διαβάζοντας και γράφοντας δεδομένα στο stream αυτό.
Στην Gambas3 πλέον δεν χρησιμοποιούμε τις εντολές READ, WRITE για να διαβάσουμε ή να γράψουμε σε έναν Pointer απ' ευθείας αλλά για να διαβάσουμε ή να γράψουμε δεδομένα σε ένα stream μνήμης.
Αφού διαβάσουμε ή γράψουμε τα δεδομένα στην μνήμη θα πρέπει να απελευθερώσουμε την μνήμη αυτή με την εντολή Free().

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project57.

```
1   ' Gambas module file
2
3   Public Sub Main()
4
5   Dim pPointer As Pointer
6   Dim hMemory As Stream
7   Dim iA As Integer
8
9   pPointer = Alloc(SizeOf(gb.Integer))
10  hMemory = Memory pPointer For Read Write
11
12  Write #hMemory, 10 As Integer
13  iA = Integer@(pPointer)
14  Print iA
15
16  Free(pPointer)
17
18  End
```

Έξοδος:

```
10
```

Ανάλυση:
Στην γραμμή 5 δηλώνουμε ένα δείκτη.
Στην γραμμή 6 δηλώνουμε ένα stream.
Στην γραμμή 7 δηλώνουμε μια μεταβλητή ακεραίου.
Στην γραμμή 9 δεσμεύουμε μια περιοχή μνήμης (memory block) με μήκος όσο ένας Integer. Δλδ δεσμεύουμε 4 bytes στην μνήμη.
Αυτό γίνεται διότι στην περιοχή αυτή της μνήμης θα γράψουμε μια τιμή ακεραίου.
Στην γραμμή 10 δημιουργούμε το stream μνήμης για τον δείκτη pPointer.
Στην γραμμή 12 γράφουμε την τιμή 10 στην περιοχή μνήμης που δείχνει ο pPointer.
Στην γραμμή 13 διαβάζουμε την περιοχή που δείχνει ο pPointer και εκχωρούμε την τιμή στην μεταβλητή iA.
Στην γραμμή 14 τυπώνουμε στην stdout την τιμή που διαβάσαμε από την μνήμη.
Στην γραμμή 16 απελευθερώνουμε την περιοχή μνήμης που δεσμεύσαμε.

Το παραπάνω παράδειγμα ήταν μια απλή επίδειξη του πώς μπορούμε να διαβάσουμε και να γράψουμε στην ελεύθερη μνήμη. Ο τρόπος αυτός φαινομενικά μοιάζει σαν να χρησιμοποιούμε μεταβλητές αλλά η χρήση των δεικτών είναι διαφορετική από την γλώσσα C.
Σε έναν δείκτη δεν εκχωρούμε τιμές όπως σε μια μεταβλητή.
Με μια μεταβλητή θα γράφαμε intVar = 3.
Με έναν δείκτης δεν θα γράφαμε pVar = 3. Ο δείκτης περιέχει την τιμή της διεύθυνσης μιας μνήμης και όχι τα δεδομένα που η μνήμη αυτή έχει. Για αυτό τον λόγο για να γράψουμε και να

διαβάσουμε δεδομένα από την μνήμη χρησιμοποιούμε τις εντολές εγγραφής και ανάγνωσης από την μνήμη.

Για να γράψουμε στην μνήμη χρησιμοποιούμε την εντολή WRITE.
Για να διαβάσουμε από την μνήμη έχουμε δύο τρόπους.
Ο πρώτος είναι να χρησιμοποιήσουμε τις εντολές ανάγνωσης τιμών όπως οι Integer@(), Byte@(), Boolean@() κτλ.
Ο δεύτερος τρόπος είναι να μετακινούμε στις κατάλληλες θέσεις τον εσωτερικό δείκτη του αντικειμένου stream μνήμης και να διαβάζουμε απευθείας από το stream μνήμης.
Τα περισσότερα stream όπως αυτό της μνήμης που δημιουργείται με την εντολή MEMORY έχουν έναν εσωτερικό δείκτη που η θέση του καθορίζει σε πιο σημείο θα διαβάσει ή θα γράψει τιμές. Μετά την εγγραφή μιας τιμής στην μνήμη με την εντολή WRITE μπορούμε να τοποθετήσουμε τον εσωτερικό αυτόν δείκτη στο σημείο που θέλουμε και στην συνέχεια να διαβάσουμε το stream με την εντολή READ.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project58.

```
1  ' Gambas module file
2
3  Public Sub Main()
4
5  Dim pPointer As Pointer
6  Dim hMemory As Stream
7  Dim iA As Integer
8
9  pPointer = Alloc(SizeOf(gb.Integer))
10 hMemory = Memory pPointer For Read Write
11
12 Print "Stream pointer potition at:"; Seek(hMemory)
13 Write #hMemory, 10 As Integer
14 Print "Stream pointer potition at:"; Seek(hMemory)
15 Seek #hMemory, 0
16 Print "Stream pointer potition at:"; Seek(hMemory)
17 iA = Read #hMemory As Integer
18
19 Print iA
20
21 Free(pPointer)
22
23 End
```

Έξοδος:

```
Stream pointer potition at:0
Stream pointer potition at:4
Stream pointer potition at:0
10
```

Ανάλυση:
Στην γραμμή 5 δηλώνουμε ένα δείκτη.
Στην γραμμή 6 δηλώνουμε ένα stream.
Στην γραμμή 7 δηλώνουμε μια μεταβλητή ακεραίου.
Στην γραμμή 9 δεσμεύουμε μια περιοχή μνήμης (memory block) με μήκος όσο ένας Integer.
Αυτό γίνεται διότι στην περιοχή αυτή της μνήμης θα γράψουμε μια τιμή ακεραίου.
Στην γραμμή 10 δημιουργούμε το stream μνήμης για τον δείκτη pPointer.
Στην γραμμή 12 εμφανίζουμε την θέση του εσωτερικού δείκτη του stream, η οποία είναι η αρχική δλδ το 0.
Στην γραμμή 13 γράφουμε στην μνήμη την τιμή 10.
Επειδή η τιμή 10 γράφεται ως ακέραιος, ο τύπος δεδομένων integer καταλαμβάνει 4 bytes και έτσι ο εσωτερικός δείκτης του stream μετά την εγγραφή μετακινείται στην θέση 4.
Αυτό φαίνεται στην γραμμή 14 όπου τυπώνουμε στην stdout την θέση του.
Στην γραμμή 15 τοποθετούμε τον εσωτερικό δείκτη του stream ξανά στην αρχή, στην θέση 0.
Αυτό φαίνεται στην γραμμή 16 όπου τυπώνουμε στην stdout την θέση του.
Αφού τοποθετήσαμε τον εσωτερικό δείκτη στην αρχή τώρα μπορούμε να διαβάσουμε το stream.
Αυτό γίνεται στην γραμμή 17 όπου εκχωρούμε την τιμή που διαβάσαμε στην μεταβλητή iA.
Στην γραμμή 19 τυπώνουμε την τιμή της μεταβλητής iA.
Στην γραμμή 21 ελευθερώνουμε την δεσμευμένη μνήμη.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project59. Σε αυτό το παράδειγμα βλέπουμε τον τρόπο όπου μπορούμε να μετακινηθούμε μέσα στην μνήμη και να διαβάσουμε τα πρώτα 4 bytes και στην συνέχεια τα επόμενα 8 bytes.

```
1  ' Gambas module file
2
3  Public Sub Main()
4
5  Dim pPointer As Pointer
6  Dim hMemory As Stream
7  Dim iA As Integer
8  Dim fA As Float
9
10 pPointer = Alloc(12)
11 hMemory = Memory pPointer For Read Write
12
13 Print "Stream pointer potition at:"; Seek(hMemory)
14 Print "Writing to memory stream a 4 byte integer"
15 Write #hMemory, 10 As Integer
16 Print "Stream pointer potition at:"; Seek(hMemory)
17
18 Print "Writing to memory stream a 8 byte float"
19 Write #hMemory, 4.2 As Float
20 Print "Stream pointer potition at:"; Seek(hMemory)
21
22 Print "Set stream potition at the beginning"
23 Seek #hMemory, 0
24
25 Print "Stream pointer potition at:"; Seek(hMemory)
26
27 Print "Read the integer"
28 iA = Read #hMemory As Integer
29 Print iA
30 Print "Stream pointer potition at:"; Seek(hMemory)
31
32 Print "Read the float"
33 fA = Read #hMemory As Float
34 Print fA
35 Print "Stream pointer potition at:"; Seek(hMemory)
36
37 Free(pPointer)
38
39 End
```

Έξοδος:

```
Stream pointer potition at:0
Writing to memory stream a 4 byte integer
Stream pointer potition at:4
Writing to memory stream a 8 byte float
Stream pointer potition at:12
Set stream potition at the beginning
Stream pointer potition at:0
Read the integer
10
Stream pointer potition at:4
Read the float
4.2
Stream pointer potition at:12
```

Ανάλυση:
Στην γραμμή 5 δηλώνουμε ένα δείκτη.
Στην γραμμή 6 δηλώνουμε ένα stream.
Στην γραμμή 7 δηλώνουμε μια μεταβλητή ακεραίου.
Στην γραμμή 8 δηλώνουμε μια μεταβλητή πραγματικού αριθμού.
Στην γραμμή 10 δεσμεύουμε μια περιοχή μνήμης (memory block) με μήκος 12 bytes.
Αυτό γίνεται διότι στην περιοχή αυτή της μνήμης θα γράψουμε μια τιμή ακεραίου και μετά μια τιμή πραγματικού αριθμού.
Στην γραμμή 11 δημιουργούμε το stream μνήμης για τον δείκτη pPointer.
Στην γραμμή 13 εμφανίζουμε την θέση του εσωτερικού δείκτη του stream, η οποία είναι η αρχική δλδ το 0.
Στην γραμμή 15 γράφουμε την ακέραιη τιμή 10 στο memory stream.
Στην γραμμή 16 εμφανίζουμε την θέση του εσωτερικού δείκτη του stream, η οποία είναι η 4 αφού έχει γραφτεί μια τιμή που καταλαμβάνει 4 bytes.
Στην γραμμή 19 γράφουμε την πραγματική τιμή 4.2 στο memory stream.
Στην γραμμή 20 εμφανίζουμε την θέση του εσωτερικού δείκτη του stream, η οποία είναι η 12 αφού έχει γραφτεί μια τιμή που καταλαμβάνει 8 bytes και ήδη ο δείκτης βρισκόταν στην θέση 4.

Στην γραμμή 23 τοποθετούμε τον εσωτερικό δείκτη του stream στην αρχή για να διαβάσουμε στην συνέχεια τις τιμές που τοποθετήσαμε στην μνήμη.

Στην γραμμή 28 διαβάζουμε τα πρώτα 4 bytes δλδ την τιμή του integer και εκχωρούμε την τιμή στην μεταβλητή iA.
Ο εσωτερικός δείκτης του stream μετακινείται στην θέση 4.
Στην γραμμή 29 τυπώνουμε στην stout την τιμή της iA.

Στην γραμμή 33 διαβάζουμε τα επόμενα 8 bytes δλδ την τιμή του float και εκχωρούμε την τιμή στην μεταβλητή fA.

Ο εσωτερικός δείκτης του stream μετακινείται στην θέση 12.
Στην γραμμή 34 τυπώνουμε στην stout την τιμή της fA.

Στην γραμμή 37 ελευθερώνουμε την δεσμευμένη μνήμη.

Αναφορές σε αντικείμενα

Σε έναν δείκτη μπορεί να εκχωρηθεί:

- η τιμή NULL,
- ένας άλλος δείκτης ,
- τιμή που επιστρέφει μια συνάρτηση η οποία είναι ένας δείκτης.
- Για παράδειγμα η Alloc() είναι μια συνάρτηση που επιστρέφει έναν δείκτη.

Στην μνήμη εκτός από τιμές μπορούμε να γράψουμε αναφορές σε αντικείμενα και μεταβλητές. Όταν δημιουργούμε ένα αντικείμενο με την λέξη **NEW** στην ουσία δημιουργούμε έναν δείκτη για το αντικείμενο αυτό και το αντικείμενο αρχικοποιείται, δλδ δημιουργείται στην μνήμη και λαμβάνει κάποιες αρχικές τιμές.

Αν θέλουμε να δούμε την διεύθυνση μνήμης ενός αντικειμένου μπορούμε να τυπώσουμε το αντικείμενο στην stdout. Η Gambas θα επιστρέψει ένα string που θα αναφέρει τον τύπο δεδομένων του αντικειμένου και την διεύθυνση μνήμης στην οποία βρίσκεται.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project60.

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  DIM aArray AS NEW String[]
6
7  PRINT aArray
8
9  END
```

Έξοδος:

```
(String[] 0x21ce4a0)
```

Για να προσπελαστεί ένα αντικείμενο το οποίο βρίσκεται στην μνήμη χρησιμοποιούνται οι δείκτες.
Το όνομα του αντικειμένου aArray στην ουσία λειτουργεί σαν ένας δείκτης.

Στην γλώσσα C ένα δείκτης έχει όνομα της μορφής *pPointerName.
Στην γλώσσα C ένα δείκτης σε ένα αντικείμενο δημιουργείται με την λέξη **new**.
Για να προσπελαστούν οι ιδιότητες και οι μέθοδοι ενός αντικειμένου σε ένα δείκτη θα

μπορούσε κάποιος να γράψει (*pPointerName).MyMethod(), αλλά η C έχει έναν ειδικό τελεστή για την προσπέλαση αυτή το -> και έτσι το σωστό είναι pPointerName->MyMethod().

Στην Gambas ένας δείκτης σε αντικείμενο έχει όνομα της μορφής pName.
Η λέξη **NEW** δημιουργεί ένα αντικείμενο στην μνήμη και επιστρέφει μια αναφορά σε αυτό σε μια μεταβλητή.
Για να προσπελαστούν οι ιδιότητες ενός αντικειμένου χρησιμοποιείται ο τελεστής . Για παράδειγμα myForm.Close().

Η Gambas προσφέρει αυτόν τον έμμεσο τρόπο διαχείρισης των αντικειμένων κάνοντας αυτόματη χρήση των δεικτών σε αντικείμενα.

Ωστόσο επιτρέπει την χρήση δεικτών για να μπορεί να διαχειρίζεται τις εξωτερικές συναρτήσεις της C οι οποίες χρησιμοποιούν δείκτες ή επιστρέφουν δείκτες.

Οι συναρτήσεις δεικτών

Alloc

Δεσμεύει ένα μέρος μνήμης και επιστρέφει έναν δείκτη προς το μέρος της μνήμης που έχει δεσμευτεί.

Σύνταξη:

```
Pointer = Alloc ( Size AS Integer [ , Count AS Integer ] )

Pointer = Alloc ( String AS String)
```

Η συνάρτηση δεσμεύει μέρος της μνήμης μεγέθους Size* Count σε bytes και επιστρέφει έναν δείκτη.
Εξ' ορισμού η τιμή Count είναι 1.

Η δεύτερη σύνταξη της Alloc() δεσμεύει μια περιοχή μνήμης στην οποία εκχωρεί την τιμή ενός string. Η συνάρτηση επιστρέφει έναν δείκτη.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project61. Σε αυτό το παράδειγμα η Alloc() γράφει το string απευθείας στην μνήμη μόλις την δεσμεύσει.

```
1  ' Gambas module file
2
3  Public Sub Main()
4
5  Dim p As Pointer
6  Dim s As String
7
8  p = Alloc("this is a string")
9
10 s = String@(p)
11 Print s
12 Free(p)
13
14 End
```

Έξοδος:

```
this is a string
```

Ανάλυση:
Στην γραμμή 5 δηλώνουμε έναν δείκτη
Στην γραμμή 6 δηλώνουμε μια μεταβλητή string
Στην γραμμή 8 δεσμεύουμε μια περιοχή μνήμης και εκχωρούμε την τιμή “this is a string”
Στην γραμμή 10 διαβάζουμε την περιοχή μνήμης που δείχνει ο δείκτης p και εκχωρούμε την τιμή στην μεταβλητή s.
Στην γραμμή 11 τυπώνουμε την τιμή της μεταβλητής s
Τέλος στην γραμμή 12 απελευθερώνουμε την δεσμευμένη μνήμη που δείχνει ο δείκτης p.

Σε αυτό το σημείο θα ξαναδούμε ένα προηγούμενο παράδειγμα.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project62. Αυτή την φορά η Alloc() δεσμεύει 4 bytes στα οποία γράφουμε αργότερα με την WRITE.

```
1   ' Gambas module file
2
3   Public Sub Main()
4
5   Dim pPointer As Pointer
6   Dim hMemory As Stream
7   Dim iA As Integer
8
9   pPointer = Alloc(SizeOf(gb.Integer))
10  hMemory = Memory pPointer For Read Write
11
12  Write #hMemory, 10 As Integer
13  iA = Integer@(pPointer)
14
15  Print iA
16
17  Free(pPointer)
18
19  End
```

Έξοδος:

```
10
```

Ανάλυση:
Στην γραμμή 5 δηλώνουμε ένα δείκτη.
Στην γραμμή 6 δηλώνουμε ένα stream.
Στην γραμμή 7 δηλώνουμε μια μεταβλητή ακεραίου.
Στην γραμμή 9 δεσμεύουμε μια περιοχή μνήμης (memory block) με μήκος όσο ένας Integer. Αυτό γίνεται διότι στην περιοχή αυτή της μνήμης θα γράψουμε μια τιμή ακεραίου.
Στην γραμμή 12 γράφουμε την τιμή 10 στην περιοχή μνήμης που δείχνει ο pPointer.
Στην γραμμή 13 διαβάζουμε την περιοχή που δείχνει ο pPointer και εκχωρούμε την τιμή στην μεταβλητή iA.
Στην γραμμή 15 τυπώνουμε στην stdout την τιμή που διαβάσαμε από την μνήμη.
Στην γραμμή 17 απελευθερώνουμε την περιοχή μνήμης που δεσμεύσαμε.

Free

Αποδεσμεύει ένα μέρος μνήμης που είχε δεσμευτεί με την Alloc.

Σύνταξη:

```
Free ( Pointer )
```

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project63.

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  DIM pPointer AS Pointer
6
7  pPointer = Alloc(4)
8
9  Free(pPointer)
10
11 END
```

Realloc

Μικραίνει ή μεγαλώνει ένα τμήμα μνήμης που έχει δεσμευτεί με την συνάρτηση Alloc.

Σύνταξη:

```
New pointer = Realloc ( Old pointer AS Pointer , Size AS Integer ,
Count AS Integer )
```

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project64.

```
1  ' Gambas module file
2
3  PUBLIC SUB Main()
4
5  DIM pPointer AS Pointer
6
7  'Allocate an integer 4 bytes
8  pPointer = Alloc(4)
9
10 'Allocate a short 2 bytes
11 pPointer = Realloc(pPointer, 2)
12
13 Free(pPointer)
14
15 END
```

StrPtr

Επιστρέφει ένα αντίγραφο ενός string που βρίσκεται σε μια διεύθυνση μνήμης.
Η συνάρτηση είναι συνώνυμη με την String@

Σύνταξη:

```
Result = StrPtr ( Pointer AS Pointer )

Result = String@ ( Pointer AS Pointer )

Result = String@ ( Pointer AS Pointer , Length AS Integer )
```

Pointer, ένας δείκτης που δείχνει σε μια τιμή string.
Length, το μήκος του string που θα διαβαστεί.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project65.

```
1  ' Gambas module file
2
3  Public Sub Main()
4
5  Dim p As Pointer
6  Dim s As String
7
8  p = Alloc("this is a string")
9
10 s = Str@(p)
11 Print s
12 Free(p)
13
14 End
```

Έξοδος:

```
this is a string
```

Ανάλυση:
Στην γραμμή 5 δηλώνουμε έναν δείκτη.
Στην γραμμή 6 δηλώνουμε μια μεταβλητή string.
Στην γραμμή 8 εκχωρούμε στην μνήμη την τιμή “this is a string”.
Η γραμμή 10 είναι ανενεργή διότι ξεκινά με ένα χαρακτήρα σχολίου.
Αυτή η γραμμή θα διάβαζε την μνήμη που δείχνει ο δείκτης p.
Αλλά σε αυτό το παράδειγμα η μνήμη θα διαβαστεί όχι από την εντολή READ αλλά από την εντολή Str@() που είναι στην επόμενη γραμμή 11.
Η Str@() διαβάζει την περιοχή μνήμης που δείχνει ο δείκτης p και εκχωρεί τα δεδομένα στην μεταβλητή s.
Στην γραμμή 12 τυπώνουμε στην stdout την τιμή που έχει η μεταβλητή s.
Στην γραμμή 13 απελευθερώνουμε την περιοχή μνήμης που δείχνει ο δείκτης p.

VarPtr

Επιστρέφει ένα δείκτη που δείχνει στο περιεχόμενο μιας μεταβλητής στην μνήμη.

Σύνταξη:

```
Pointer = VarPtr ( Variable )
```

Πολλές φορές οι εξωτερικές συναρτήσεις της C χρησιμοποιούν δείκτες που δείχνουν σε άλλους δείκτες. Την λειτουργία αυτή τελεί και η VarPtr.

Variable, είναι μια μεταβλητή η οποία μπορεί να είναι τοπική ή καθολική της ίδιας κλάσης.

Ο τύπος της μεταβλητής μπορεί να είναι αριθμητικός, δείκτης ή αλφαριθμητικός (numeric, pointer, string).
Χρησιμοποιείστε την συνάρτηση αυτή όταν μια εξωτερική συνάρτηση χρειάζεται έναν δείκτη σε μια αριθμητική μεταβλητή όπως για παράδειγμα έναν δείκτη σε έναν άλλο δείκτη όπως 'int **', 'void **'.

Ωστόσο μην χρησιμοποιείτε αυτήν την συνάρτηση για έναν δείκτη σε δείκτη αλφαριθμητικό δλδ όταν η εξωτερική συνάρτηση έχει όρισμα 'char **' επειδή τα περιεχόμενα μια μεταβλητής string της Gambas είναι μόνο προς ανάγνωση.

Boolean@

Επιστρέφει μια τιμή Boolean που είναι αποθηκευμένη σε μια διεύθυνση μνήμης που δείχνει ένας δείκτης.

Σύνταξη:

```
Result = Boolean@ ( Pointer AS Pointer )
```

Αν η διεύθυνση μνήμης είναι μη αναγνώσιμη ένα λάθος ορίσματος (20, Bad argument) εμφανίζεται.

Byte@

Επιστρέφει μια τιμή Byte που είναι αποθηκευμένη σε μια διεύθυνση μνήμης που δείχνει ένας δείκτης.

Σύνταξη:

```
Result = Byte@ ( Pointer AS Pointer )
```

Αν η διεύθυνση μνήμης είναι μη αναγνώσιμη ένα λάθος ορίσματος (20, Bad argument) εμφανίζεται.

Float@

Επιστρέφει μια τιμή Byte που είναι αποθηκευμένη σε μια διεύθυνση μνήμης που δείχνει ένας δείκτης.

Σύνταξη:

```
Result = Float@ ( Pointer AS Pointer )
```

Αν η διεύθυνση μνήμης είναι μη αναγνώσιμη ένα λάθος ορίσματος (20, Bad argument) εμφανίζεται.

Long@

Επιστρέφει μια τιμή Long που είναι αποθηκευμένη σε μια διεύθυνση μνήμης που δείχνει ένας δείκτης.

Σύνταξη:

```
Result = Long@ ( Pointer AS Pointer )
```

Αν η διεύθυνση μνήμης είναι μη αναγνώσιμη ένα λάθος ορίσματος (20, Bad argument) εμφανίζεται.

Integer@

Επιστρέφει μια τιμή Integer που είναι αποθηκευμένη σε μια διεύθυνση μνήμης που δείχνει ένας δείκτης.

Σύνταξη:

```
Result = Int@ ( Pointer AS Pointer )

Result = Integer@ ( Pointer AS Pointer )
```

Αν η διεύθυνση μνήμης είναι μη αναγνώσιμη ένα λάθος ορίσματος (20, Bad argument) εμφανίζεται.

Long@

Επιστρέφει μια τιμή Long που είναι αποθηκευμένη σε μια διεύθυνση μνήμης που δείχνει ένας δείκτης.

Σύνταξη:

```
Result = Long@ ( Pointer AS Pointer )
```

Αν η διεύθυνση μνήμης είναι μη αναγνώσιμη ένα λάθος ορίσματος (20, Bad argument) εμφανίζεται.

Pointer@

Επιστρέφει μια τιμή Pointer που είναι αποθηκευμένη σε μια διεύθυνση μνήμης που δείχνει ένας δείκτης.

Σύνταξη:

```
Result = Pointer@ ( Pointer AS Pointer )
```

Αν η διεύθυνση μνήμης είναι μη αναγνώσιμη ένα λάθος ορίσματος (20, Bad argument) εμφανίζεται.

Short@

Επιστρέφει μια τιμή Short που είναι αποθηκευμένη σε μια διεύθυνση μνήμης που δείχνει ένας δείκτης.

Σύνταξη:

```
Result = Short@ ( Pointer AS Pointer )
```

Αν η διεύθυνση μνήμης είναι μη αναγνώσιμη ένα λάθος ορίσματος (20, Bad argument) εμφανίζεται.

Single@

Επιστρέφει μια τιμή Single που είναι αποθηκευμένη σε μια διεύθυνση μνήμης που δείχνει ένας δείκτης.

Σύνταξη:

```
Result = Single@ ( Pointer AS Pointer )
```

Αν η διεύθυνση μνήμης είναι μη αναγνώσιμη ένα λάθος ορίσματος (20, Bad argument) εμφανίζεται.

String@

Επιστρέφει μια τιμή String που είναι αποθηκευμένη σε μια διεύθυνση μνήμης που δείχνει ένας δείκτης.

Σύνταξη:

```
Result = String@ ( Pointer AS Pointer , Length AS Integer )
```

Το μήκος του string που θα διαβαστεί καθορίζεται από την παράμετρο Length.
Αν η διεύθυνση μνήμης είναι μη αναγνώσιμη τότε NULL επιστρέφεται.

Όλες οι συναρτήσεις XXX@() όπως είδαμε διαβάζουν την τιμή μιας διεύθυνσης που δείχνει ένας δείκτης.

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 καταγράφει 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 Alloc() θα πρέπει να την απελευθερώνετε μετά την χρήση των δεδομένων με την Free().

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.

Η εκχώρηση νέας τιμής σε υπάρχων δείκτη δημιουργεί διαρροή μνήμης.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project66.

```
1  ' Gambas module file
2
3  Public Sub Main()
4
5  Dim pPointer1 As Pointer
6  Dim pPointer2 As Pointer
7
8  pPointer1 = Alloc(4)
9  pPointer2 = Alloc(4)
10
11 Print "Address of pPointer1"; pPointer1
12 Print "Address of pPointer2"; pPointer2
13
14 pPointer2 = pPointer1
15
16 Print "Address of pPointer1"; pPointer1
17 Print "Address of pPointer2"; pPointer2
18
19 Free(pPointer1)
20 Free(pPointer2)
21
22 End
```

Έξοδος:

```
Address of pPointer1(Pointer 0xaf5928)
Address of pPointer2(Pointer 0xaf5948)
Address of pPointer1(Pointer 0xaf5928)
Address of pPointer2(Pointer 0xaf5928)
```

Ανάλυση:
Στις γραμμές 5, 6 δηλώνουμε δύο δείκτες.
Στις γραμμές 8, 9 δεσμεύουμε περιοχές στην μνήμη για 4 bytes.
Στην γραμμή 11 τυπώνουμε στην stdout την διεύθυνση που έχουμε δεσμεύσει για τον pPointer1, η οποία είναι η 0xaf5928.
Ομοίως για τον pPointer2 έχουμε δεσμεύσει την διεύθυνση 0xaf5948.

Ένας δείκτης όμως είναι μια μεταβλητή και μπορούμε να εκχωρήσουμε όποτε θέλουμε μια νέα τιμή.
Στην γραμμή 14 εκχωρούμε μια νέα τιμή στον pPointer2, την διεύθυνση που έχει ο pPointer1.
Στις γραμμές 16, 17 τυπώνουμε ξανά τις διευθύνσεις των δεικτών.
Στο σημείο αυτό βλέπουμε ότι οι δείκτες έχουν την ίδια τιμή και έτσι δείχνουν στην ίδια διεύθυνση.

Η διεύθυνση όμως 0xaf5948 είναι δεσμευμένη αλλά πλέον μη προσβάσιμη και όταν στην γραμμή 20 απελευθερώνουμε τον δείκτη pPointer2 δεν απελευθερώνουμε την διεύθυνση

0xaf5948 !
Επειδή η διεύθυνση 0xaf5948 είναι δεσμευμένη αλλά μη προσβάσιμη είναι σαν να έχουμε διαρροή μνήμης από τον υπολογιστή μας.

Μη απελευθέρωση δείκτη οδηγεί σε διαρροή μνήμης.

Όταν δεσμεύσετε μια περιοχή μνήμης με την Alloc() σε μια συνάρτηση και η συνάρτηση τελειώσει αλλά εσείς δεν έχετε απελευθερώσει την μνήμη με την Free() 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:
Το παράδειγμα αυτό είναι το Project67.

```
1  ' Gambas module file
2
3  Public Sub Main()
4
5    UsePointer()
6
7  End
8
9  Public Sub UsePointer()
10
11 Dim pPointer1 As Pointer
12
13 pPointer1 = Alloc(4)
14
15 Print "Address of pPointer1"; pPointer1
16
17 End
```

Έξοδος:

```
Address of pPointer1(Pointer 0xaf5928)
```

Ανάλυση:
Το πρόγραμμα ξεκινά από την Main() στην γραμμή 3.
Στην γραμμή 5 καλείται η υπορουτίνα UsePointer().
Στην γραμμή 9 δηλώνουμε την υπορουτίνα UsePointer().
Στην γραμμή 11 δηλώνουμε έναν δείκτη.
Στην γραμμή 13 δεσμεύουμε μια διεύθυνση στην μνήμη και 4 bytes.
Στην γραμμή 15 τυπώνουμε στην stdout την διεύθυνση που έχει εκχωρηθεί στον δείκτη pPointer1.
Στην υπορουτίνα UsePointer() δεν απελευθερώνουμε τον δείκτη και μετά το τέλος της συνάρτησης παύουμε να έχουμε πρόσβαση στην διεύθυνση της μνήμης αυτής και οδηγούμαστε

σε μια διαρροή μνήμης.

Εγγραφή δεδομένων με μεγαλύτερο μέγεθος μνήμης από αυτό που έχει δεσμευθεί.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project68.

```
1  ' Gambas module file
2
3  Public Sub Main()
4
5  Dim pPointer As Pointer
6  Dim hMemory As Stream
7
8  pPointer = Alloc(4)
9  hMemory = Memory pPointer For Read Write
10
11 Write #hMemory, CByte(0) As Byte
12 Write #hMemory, CByte(1) As Byte
13 Write #hMemory, CByte(0) As Byte
14 Write #hMemory, CByte(1) As Byte
15 Write #hMemory, CByte(0) As Byte
16 Close #hMemory
17
18 Free(pPointer)
19
20 End
```

Ανάλυση:
Στην γραμμή 5 δηλώνουμε έναν δείκτη.
Στην γραμμή 6 δηλώνουμε ένα stream.
Στην γραμμή 8 δεσμεύουμε 4 bytes στην μνήμη.
Στην γραμμή 9 δημιουργούμε το memory stream.
Στις γραμμές 11 με 14 γράφουμε 4 bytes στην μνήμη.
Στην γραμμή 15 γράφουμε ένα 5ο byte.
Σε αυτό το σημείο έχουμε γράψει στην μνήμη περισσότερο από το χώρο που δεσμεύσαμε.
Μόλις έχουμε δημιουργήσει ένα **Buffer overflow**.
Στην γραμμή 16 κλείνουμε το stream.
Και τέλος στην γραμμή 18 απελευθερώνουμε την μνήμη που δεσμεύσαμε.

Χρησιμοποίηση δείκτη που ήδη έχει απελευθερωθεί.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project69.

```
1  ' Gambas module file
2
3  Public Sub Main()
4
5  Dim pPointer As Pointer
6  Dim hMemory As Stream
7
8  pPointer = Alloc(4)
9  hMemory = Memory pPointer For Read Write
10
11 Write #hMemory, CByte(0) As Byte
12 Write #hMemory, CByte(1) As Byte
13 Write #hMemory, CByte(0) As Byte
14 Write #hMemory, CByte(1) As Byte
15 Close #hMemory
16
17 Print Byte@(pPointer)
18
19 Free(pPointer)
20
21 Print Byte@(pPointer)
22
23 End
```

Έξοδος:

```
0
0
```

Ανάλυση:
Στην γραμμή 5 δηλώνουμε έναν δείκτη.
Στην γραμμή 6 δηλώνουμε ένα stream.
Στην γραμμή 8 δεσμεύουμε 4 bytes στην μνήμη.
Στην γραμμή 9 δημιουργούμε το memory stream.
Στις γραμμές 11 με 14 γράφουμε 4 bytes στην μνήμη.
Στην γραμμή 15 κλείνουμε το stream.
Στην γραμμή 17 τυπώνουμε τα δεδομένα που βρίσκονται στην μνήμη.
Στην συνέχεια στην γραμμή 19 απελευθερώνουμε τον δείκτη pPointer.
Η διεύθυνση που είχε δεσμευθεί περνά στην ελεύθερη μνήμη αλλά τα δεδομένα σε αυτήν παραμένουν.
Κανονικά δεν θα πρέπει να χρησιμοποιήσουμε από εδώ και πέρα την διεύθυνση μνήμης του pPointer.
Στην γραμμή 21 όμως χρησιμοποιούμε τον pPointer ανακτώντας τα δεδομένα που έχει η διεύθυνση αυτή. Αυτό όμως είναι ένα λάθος διότι χρησιμοποιούμε έναν δείκτη που έχει απελευθερωθεί.

Απελευθέρωση δείκτη που ήδη έχει απελευθερωθεί.

Σε αυτήν την περίπτωση ο διερμηνευτής εμφανίζει μια προειδοποίηση.
gbx3: warning: -1 allocation(s) non freed.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project70.

```
1  ' Gambas module file
2
3  Public Sub Main()
4
5  Dim pPointer As Pointer
6
7  pPointer = Alloc(4)
8
9  Free(pPointer)
10
11 Free(pPointer)
12
13 End
```

Έξοδος:

```
gbx3: warning: -1 allocation(s) non freed.
```

Ανάλυση:
Στην γραμμή 5 δηλώνουμε έναν δείκτη.
Στην γραμμή 7 δεσμεύουμε 4 bytes στην μνήμη.
Στην γραμμή 9 απελευθερώνουμε την μνήμη.
Στην γραμμή 10 επιχειρούμε να απελευθερώσουμε ξανά την μνήμη.
Ο διερμηνευτής εμφανίζει μια προειδοποίηση.

Απελευθέρωση μνήμης που δεν έχει δεσμευτεί

Σε αυτήν την περίπτωση το πρόγραμμα καταρρέει.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project71.

```
1  ' Gambas module file
2
3  Public Sub Main()
4
5  Dim pPointer As Pointer
6
7  pPointer = Alloc(4)
8
9  Print pPointer
10
11 pPointer += 10
12
13 Print pPointer
14
15 'Free(pPointer)
16
17 End
```

Έξοδος:

```
(Pointer 0xdca5a8)
(Pointer 0xdca5b2)
gbx3: warning: 1 allocation(s) non freed.
```

Ανάλυση:
Στην γραμμή 5 δηλώνουμε έναν δείκτη.
Στην γραμμή 7 δεσμεύουμε 4 bytes στην μνήμη.
Στην γραμμή 9 τυπώνουμε τον δείκτη στην stdout. Αυτό έχει ως αποτέλεσμα να δούμε την διεύθυνση που κρατά.
Στην γραμμή 11 μετακινούμαστε στην μνήμη κατά 10 bytes.
Άν προσθέσουμε στο $dca5a8_{16}=14460328_{10}$ το 10 έχουμε το 14460338_{10}=DCA5B2 που είναι αυτό που τυπώνεται στην stdout από την γραμμή 13.
Η 0xdca5b2 είναι η νέα διεύθυνση της pPointer μετά την μετακίνηση κατά 10 bytes στην μνήμη.
Η γραμμή 15 βρίσκεται σε σχόλιο και είναι η αποδέσμευση της μνήμης που δείχνει ο pPointer.
Αν βγάλουμε το σχόλιο της γραμμής 15 και τρέξουμε ξανά το πρόγραμμα αυτήν την φορά θα καταρρεύσει δίνοντας την παρακάτω έξοδο.

Έξοδος:

```
*** glibc detected *** Project89: free(): invalid pointer: 0x00000000010a95aa ***
======= Backtrace: =========
/lib/libc.so.6(+0x775b6)[0x7f076a6fe5b6]
/lib/libc.so.6(cfree+0x73)[0x7f076a704e83]
...
```

Αυτήν την φορά η μεταβλητή pPointer έλαβε διαφορετικές τιμές αλλά στην έξοδο βλέπουμε στην πρώτη γραμμή ότι έχουμε μη έγκυρο δείκτη τον 0x00000000010a95aa.
Αυτή είναι η τιμή της pPointer μετά την μετακίνηση των 10 bytes σε αυτήν την εκτέλεση του προγράμματος.
Η κατάρρευση του προγράμματος έγινε διότι απελευθερώσαμε μνήμη που δεν είχαμε δεσμεύσει.

Απόπειρα εγγραφής σε δείκτη που έχει ελευθερωθεί

Η περίπτωση αυτή εμπίπτει στην περίπτωση χρήσης δείκτη που έχει ήδη απελευθερωθεί.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project72.

```
1  ' Gambas module file
2
3  Public Sub Main()
4
5  Dim pPointer As Pointer
6  Dim hMemory As Stream
7
8  pPointer = Alloc(SizeOf(gb.Integer))
9  hMemory = Memory pPointer For Read Write
10
11 Print pPointer
12
13 Free(pPointer)
14
15 Write #hMemory, 10 As Integer
16
17 Print pPointer
18 Print Int@(pPointer)
19 Print pPointer
20
21 End
```

Έξοδος:

```
(Pointer 0x1e1d5a8)
(Pointer 0x1e1d5a8)
10
(Pointer 0x1e1d5a8)
```

Στην γραμμή 13 απελευθερώνουμε τον δείκτη pPointer.
Στην συνέχεια όμως στην γραμμή 15 γράφουμε στην διεύθυνση που δείχνει ο pPointer.
Αυτό είναι ένα λάθος διότι η περιοχή μνήμης που δείχνει ο pPointer δεν είναι πλέον δεσμευμένη από το δικό μας πρόγραμμα. Έχει περάσει στην ελεύθερη μνήμη και μπορεί να χρησιμοποιηθεί ή να χρησιμοποιείται από άλλο πρόγραμμα.
Εμείς παρόλο αυτά μπορούμε να διαβάσουμε και να γράψουμε σε αυτήν την περιοχή μνήμης αλλά αυτό θα ήταν ένα σφάλμα.

Εγγραφή σε περιοχή μνήμης που ποτέ δεν δεσμεύτηκε

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project73.

```
1   ' Gambas module file
2
3   Public Sub Main()
4
5   Dim pPointer As Pointer
6   Dim hMemory As Stream
7
8   pPointer = Alloc(SizeOf(gb.Integer))
9   hMemory = Memory pPointer For Read Write
10
11  Print pPointer
12
13  pPointer += 10
14
15  Print pPointer
16
17  Write #hMemory, 10 As Integer
18
19  Free(pPointer)
20
21  End
```

Έξοδος:

```
(Pointer 0x263b5d8)
(Pointer 0x263b5e2)
*** glibc detected *** Project91: free(): invalid pointer: 0x000000000263b5da
***
```

Ανάλυση:
Αφού δηλώσουμε τον δείκτη και το stream και δημιουργήσουμε το memory stream, στην γραμμή 11 τυπώνουμε την διεύθυνση που έχει η μεταβλητή pPointer η οποία είναι η 0x263b5d8.
Στην γραμμή 13 μετακινούμαστε 10 bytes μπροστά στην μνήμη και μετακινούμαστε στην διεύθυνση 0x263b5e2.
Στην γραμμή 17 γράφουμε σε αυτήν την διεύθυνση. Μόλις κάναμε ένα λάθος και γράψαμε σε περιοχή μνήμης που δεν δεσμεύσαμε.

Εκτός από αυτό ο pPointer τώρα δείχνει στην διεύθυνση 0x263b5e2 και η παλιά δεσμευμένη διεύθυνση 0x263b5d8 δεν μπορεί να απελευθερωθεί στην γραμμή 19 όπως θα θέλαμε.

Αποτέλεσμα όλων αυτών είναι το πρόγραμμα να καταρρεύσει με ένα μήνυμα ότι προσπαθούμε να ελευθερώσουμε λάθος περιοχή μνήμης.

```
*** glibc detected *** Project91: free(): invalid pointer: 0x000000000263b5da ***
```

Ασφαλής χρήση δεικτών.

Κανονικά μετά το τέλος χρήσης ενός δείκτη θα πρέπει να κλείνουμε το memory stream, και να απελευθερώνουμε τον δείκτη και τέλος να του εκχωρούμε την τιμή 0.

Το παρακάτω παράδειγμα δείχνει έναν προτεινόμενο τρόπο ασφαλούς προγραμματισμού με έναν δείκτη.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project74.

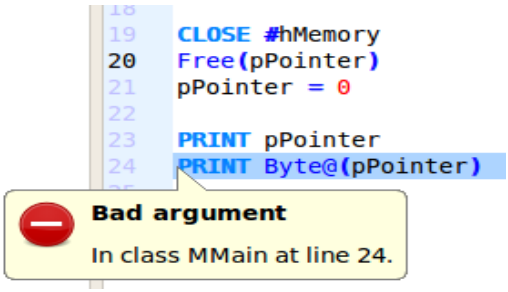
```
1  ' Gambas module file
2
3  Public Sub Main()
4
5  Dim pPointer As Pointer
6  Dim hMemory As Stream
7
8  pPointer = Alloc(4)
9
10 Print pPointer
11
12 hMemory = Memory pPointer For Read Write
13
14 Write #hMemory, CByte(0) As Byte
15 Write #hMemory, CByte(1) As Byte
16 Write #hMemory, CByte(0) As Byte
17 Write #hMemory, CByte(1) As Byte
18
19 Close #hMemory
20 Free(pPointer)
21 pPointer = 0
```

```
22
23  Print pPointer
24  Print Byte@(pPointer)
25
26  End
```

Έξοδος:

```
(Pointer 0xdca5a8)
```

Ανάλυση:
Στην γραμμή 10 τυπώνουμε στην stdout την διεύθυνση που δείχνει ο δείκτης.
Στην γραμμή 19 μετά την χρήση του δείκτη κλείνουμε το memory stream.
Στην γραμμή 20 απελευθερώνουμε την διεύθυνση που δείχνει ο δείκτης.
Τέλος στην γραμμή 21 εκχωρούμε την τιμή 0 στο δείκτη.
Στην γραμμή 23 δεν υπάρχει πλέον κάποια διεύθυνση να τυπωθεί στην stdout.
Στην γραμμή 24 αν προσπαθήσουμε να χρησιμοποιήσουμε τον δείκτη ξανά εγείρεται ένα λάθος.



Εικόνα 73: Αφού μηδενίσουμε έναν δείκτη ένα λάθος εγείρεται στην προσπάθεια χρήσης του

Χρησιμοποιώντας δείκτες στις εξωτερικές συναρτήσεις της C

Αφού είδαμε κάποια θέματα σχετικά με την μνήμη και τους δείκτες τώρα θα δούμε τις συναρτήσεις της Gambas, οι οποίες διαχειρίζονται εξωτερικές βιβλιοθήκες, με λεπτομέρειες.

Στην C ένας δείκτης δηλώνεται με τον τύπο της μεταβλητής που θα δείχνει έπεται ένα * και το όνομα του δείκτη.

Για παράδειγμα αν θέλουμε να δηλώσουμε έναν δείκτη για έναν ακέραιο γράφουμε το εξής:

int *pPointer

Με αυτήν την δήλωση λέμε να δημιουργηθεί μια μεταβλητή pPointer η οποία θα είναι δείκτης (*) και θα εκτείνεται 4 bytes στην μνήμη, δλδ θα δείχνει στην μνήμη και θα καταλαμβάνει θέση για έναν integer.

Μέχρι τώρα είδαμε πώς δηλώνεται ένας δείκτης στην Gambas3 και πως γράφουμε και διαβάζουμε την μνήμη με τους δείκτες και ένα stream μνήμης.

Αλλά στην C γιατί χρειάζονται οι δείκτες;
Η χρήση των δεικτών δεν είναι όπως των μεταβλητών, δλδ για να γράφουμε απευθείας τιμές στην μνήμη.
Η C χρησιμοποιεί δείκτες για να χειρίζεται δεδομένα στην ελεύθερη μνήμη. Πολλές φορές οι παράμετροι που διοχετεύονται στις συναρτήσεις της C είναι δείκτες και όχι μεταβλητές.

Παραδείγματα:
Υπάρχει η συνάρτηση “get_current_dir_name()” η οποία δηλώνεται ως
“char *get_current_dir_name(void);”
εδώ όπως βλέπουμε η συνάρτηση επιστρέφει έναν δείκτη σε ένα string.
Η συνάρτηση αυτή βρίσκει το όνομα του τρέχοντος καταλόγου και την τιμή αυτή την τοποθετεί στην ελεύθερη μνήμη. Η Gambas3 μπορεί να έχει πρόσβαση στην ελεύθερη μνήμη και να διαβάσει την τιμή του string και έτσι να χρησιμοποιήσει την συνάρτηση get_current_dir_name() και το αποτέλεσμα της.

Υπάρχει η συνάρτηση “getloadavg(double loadavg[], int nelem)” η οποία δηλώνεται ως “int getloadavg(double loadavg[], int nelem);”
Η συνάρτηση δείχνει τον φόρτο του συστήματος.
Η συνάρτηση επιστρέφει έναν integer και δέχεται ως ορίσματα ένα πίνακα πραγματικών αριθμών και ένα ακέραιο.
Η συνάρτηση εδώ τον πίνακα loadavg[] τον μεταχειρίζεται ως δείκτη.
Αν η Gambas3 διαβάσει την διεύθυνση που δείχνει ο δείκτης αυτός θα λάβει από την ελεύθερη μνήμη την τιμή φόρτου του συστήματος.

Μια άλλη περίπτωση, υπάρχει η συνάρτηση “cuserid(char *string)” η οποία επιστρέφει σε δείκτη το όνομα του χρήστη της διεργασίας που τρέχει.

Η συνάρτηση δηλώνεται ως “char *cuserid(char *string);”
Η συνάρτηση επιστρέφει την τιμή ενός string στην ελεύθερη μνήμη. Επίσης δέχεται ως παράμετρο έναν άλλο δείκτη. Αν η Gambas3 διαβάσει την τιμή που επιστρέφει η συνάρτηση στην ελεύθερη μνήμη τότε θα διαβάσει την τιμή του string που δείχνει τον χρήστη της διεργασίας που τρέχει. Για να γίνει αυτό η Gambas3 θα πρέπει να διαβάσει τον δείκτη που επιστρέφει η συνάρτηση.

Υπάρχουν περιπτώσεις που η C χρησιμοποιεί για τους ίδιους τους δείκτες άλλους δείκτες. Δλδ χρησιμοποιεί έναν δείκτη σε δείκτη.
Ένας δείκτης σε δείκτη εμφανίζεται με διπλό * δλδ ο “int **pPointer” είναι ένας δείκτης σε δείκτη ενός integer.

Για παράδειγμα υπάρχει η συνάρτηση snd_seq_open(snd_seq_t **seqp, const char * name, Int streams, Int mode) η οποία δηλώνεται ως
“int snd_seq_open(snd_seq_t **seqp, const char * name, Int streams, Int mode);”

Η συνάρτηση αυτή επιστρέφει έναν integer και το πρώτο της όρισμα είναι ένας δείκτης σε δείκτη, το δεύτερο ένας δείκτης σε string ο οποίος είναι σταθερά κτλ.

Η Gambas3 για να χειριστεί τους δείκτες σε δείκτες χρησιμοποιεί την εντολή VarPtr η οποία επιστρέφει έναν δείκτη από μια μεταβλητή που δείχνει σε μια διεύθυνση στην μνήμη. Δλδ η VarPtr υλοποιεί τον δείκτη σε δείκτη με άλλα λόγια.

Εντολές διαχείρισης εξωτερικών συναρτήσεων

LIBRARY

Δηλώνει την βιβλιοθήκη όπου η εξωτερική συνάρτηση βρίσκεται.

Σύνταξη:

```
LIBRARY Library name [:LibVersion]
```

Library name, είναι το όνομα της βιβλιοθήκης που περιέχει την εξωτερική συνάρτηση. Το όνομα της βιβλιοθήκης είναι το όνομα του αρχείου βιβλιοθήκης χωρίς επέκταση ή αριθμό έκδοσης.
Αν χρειαστεί να δηλώσουμε συγκεκριμένη έκδοση βιβλιοθήκης μετά το όνομα γράφουμε άνω κάτω τελεία και τον αριθμό έκδοσης.

Παράδειγμα:
Αν θέλουμε να χρησιμοποιήσουμε την βιβλιοθήκη libGL.so.1 γράφουμε:

```
1 LIBRARY “libGL:1" 'ή
2 LIBRARY “libGL"
```

EXTERN

Δηλώνει μια εξωτερική συνάρτηση η οποία υπάρχει σε μια βιβλιοθήκη του συστήματος.

Σύνταξη:

```
{ PUBLIC | PRIVATE } EXTERN

Identifier
(
[ Parameter AS Datatype [ , ... ] ]
)
[ AS Datatype ]
[ IN Library ] [ EXEC Alias ]
```

- Identifier, είναι το όνομα της εξωτερικής συνάρτησης όπως αυτό δηλώνεται στην γλώσσα C.
- Parameter, είναι το όνομα μιας παραμέτρου που δέχεται η συνάρτηση. Το όνομα αυτό δεν είναι απαραίτητο να είναι το ίδιο με το όνομα της παραμέτρου που έχει η συνάρτηση της C.
- Datatype, είναι ο τύπος δεδομένων των παραμέτρων της συνάρτησης και της τιμής επιστροφής της συνάρτησης. Η τύπος δεδομένων της παραμέτρου της συνάρτησης μπορεί να είναι οποιοσδήποτε τύπος δεδομένων της Gambas εκτός από Variant. Αν ο

τύπος δεδομένων είναι Object ένας δείκτης περνά σαν παράμετρος. Αν ο τύπος δεδομένων είναι μια κλάση τότε η συνάρτηση λαμβάνει έναν δείκτη στα στατικά δεδομένα της κλάσης. Αν χρειαστεί να περάσετε σαν παράμετρο έναν δείκτη, δημιουργείστε έναν με την συνάρτηση Alloc() και περάστε στην συνάρτηση αυτόν τον δείκτη σαν παράμετρο. Στην Gambas3 δεν είναι απαραίτητο αυτό. Αυτή η ενέργεια γίνεται με την συνάρτηση VarPtr().

- Library, είναι ένα string που δηλώνει το όνομα της βιβλιοθήκης όπου υπάρχει η συνάρτηση. Η σύνταξη του ονόματος αυτού ακολουθεί τους κανόνες της εντολής LIBRARY. Αν δεν δηλωθεί όνομα βιβλιοθήκης χρησιμοποιείται η βιβλιοθήκη που έχει δηλωθεί με την τελευταία δήλωση της εντολής LIBRARY.
- Alias, αν το όνομα της συνάρτησης είναι δεσμευμένο σαν όνομα από την Gambas μπορείτε να δηλώσετε άλλο όνομα αλλά θα πρέπει να δηλώσετε το πραγματικό όνομα της συνάρτησης από την βιβλιοθήκη μετά το EXEC. Το πραγματικό όνομα της συνάρτησης είναι η τιμή της λέξης Alias.

Η τιμή επιστροφής της συνάρτησης μπορεί να είναι τύπου δεδομένων από οποιοδήποτε τύπο δεδομένων της Gambas εκτός από Object και Variant.
Εάν η συνάρτηση επιστρέφει ένα string η Gambas επιστρέφει ένα αντίγραφο από αυτό.
Αν θέλετε να έχετε πρόσβαση στο καθεαυτό string χρησιμοποιείστε τον τύπο δεδομένων Pointer και την συνάρτηση StrPtr().

Παράδειγμα:

```
LIBRARY "libc:6"
'char *get_current_dir_name(void);
PUBLIC EXTERN get_current_dir_name() AS Pointer

'ή

PUBLIC EXTERN get_current_dir_name() AS Pointer IN "libc:6"

'char *cuserid(char *string);
Public Extern cuserid(pString As Pointer) As Pointer In "libc:6"

'int getloadavg(double loadavg[], int nelem);
Extern getloadavg(ploadavg As Pointer, nelem As Integer) As
Integer In "libc:6"

' This function name is already a Gambas reserved word!
EXTERN SysOpen(Name AS String, Flags AS Integer, Mode AS Integer)
AS Integer IN "libc:6" EXEC "open"
```

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project75.

```
1  ' Gambas module file
2  'int getloadavg(double loadavg[], int nelem);
3  Extern getloadavg(ploadavg As Pointer, nelem As Integer) As
   Integer In "libc:6"
4
5  Public Sub Main()
6
7    Dim pLoad As Pointer
8    Dim fLoad As Float
9    Dim hMemory As Stream
10   Dim i As Integer
11
12 'getloadavg
13   pLoad = Alloc(SizeOf(gb.Float), 3)
14   hMemory = Memory pLoad For Read
15
16   If getloadavg(pLoad, 3) = 3 Then
17     For i = 1 To 3
18      fLoad = Read #hMemory As Float
19      Print fLoad
20    Next
21     Free(pLoad)
22   Else
23     fLoad = Float@(pLoad)
24     Print fLoad
25    Free(pLoad)
26   Endif
27
28 Close #hMemory
29 pLoad = 0
30
31 End
```

Έξοδος:

```
1.23
1.13
1.09
```

Ανάλυση:
Στην γραμμή 2 έχουμε σε σχόλιο την δήλωση της συνάρτησης της C.
Η συνάρτηση αυτή δηλώνεται ως εξής,
“int getloadavg(double *loadavg[]*, int *nelem*);”

Η συνάρτηση **getloadavg**() λαμβάνει από 1 έως 3 δείγματα φόρτου του συστήματος.
Το πλήθος των δειγμάτων καθορίζεται από την παράμετρο *nelem*.
Και επιστρέφει τον αριθμό των δειγμάτων που έλαβε επίσης εκχωρεί στον πίνακα *loadavg*[] τον φόρτο του συστήματος για τα δείγματα που έλαβε. Τα στοιχεία του πίνακα *loadavg*[] είναι

τα δείγματα φόρτου του συστήματος για 1, 5, και 15 λεπτά αντίστοιχα.

Εμείς διοχετεύουμε σαν παράμετρο nelem το 3 ώστε να ληφθούν και τα τρία δείγματα.

Στην γραμμή 7 δηλώνουμε ένα δείκτη
Στην γραμμή 8 δηλώνουμε έναν float.
Στην γραμμή 9 δηλώνουμε το memory stream.
Στην γραμμή 10 δηλώνουμε έναν ακέραιο.
Στην γραμμή 13 δεσμεύουμε μια περιοχή μνήμης όσο 3 Float. Η περιοχή αυτή είναι για τα τρία δείγματα που θα εκχωρήσει η συνάρτηση στον πίνακα *loadavg*[].
Στην γραμμή 14 δημιουργούμε το memory stream.

Στην συνέχεια ελέγχουμε στην γραμμή 16 την επιστροφή της συνάρτησης getloadavg().
Αν η συνάρτηση επιστρέφει 3 τότε τρία δείγματα έχουν ληφθεί.
Αλλιώς έχει ληφθεί μόνο ένα.
Στην περίπτωση που έχουν ληφθεί 3 δείγματα τρέχουμε έναν βρόχο FOR...NEXT τρεις φορές στην γραμμή 17 για να διαβάσουμε την μνήμη στην γραμμή 18 και να τυπώσουμε το αποτέλεσμα στην γραμμή στην γραμμή 19.

Οι τιμές που διαβάζονται είναι οι τρεις Float στην σειρά που έχει τοποθετήσει η συνάρτηση στην μνήμη.

Όταν τελειώσει ο βρόχος απελευθερώνουμε τον δείκτη.

Στην περίπτωση που η συνάρτηση επιστρέψει ένα δείγμα απλά διαβάζουμε μία τιμή Float από την μνήμη, τυπώνουμε το αποτέλεσμα και απελευθερώνουμε την μνήμη.

Τέλος κλείνουμε το stream και μηδενίζουμε τον δείκτη.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project76.

```
1  ' Gambas module file
2  'char *getwd(char *buf);
3  Extern getwd(buf As Pointer) As Pointer In "libc:6"
4
5  Public Sub Main()
6
7    Dim pBuf As Pointer
8    Dim pFunc As Pointer
9    Dim sWorkingDirectory As String
10
11 'getwd
12   pBuf = Alloc(4096)
13
14   pFunc = getwd(pBuf)
15
16   sWorkingDirectory = String@(pFunc)
17   Print sWorkingDirectory
```

```
18
19    Print pBuf
20    Print pFunc
21
22    Free(pBuf)
23    pFunc = 0
24    pBuf = 0
25
26  End
```

Έξοδος:

```
/mnt/store/LINUX/Gambas3/Projects/Project86
(Pointer 0x7c1978)
(Pointer 0x7c1978)
```

Ανάλυση:
Στην γραμμή 2 έχουμε σε σχόλιο την δήλωση της συνάρτησης της C.
Η συνάρτηση getwd() δέχεται ένα δείκτη string και επιστρέφει έναν δείκτη string το οποίο είναι ο τρέχων κατάλογος της καλούμενης διεργασίας.
Οι δείκτες της συνάρτησης αναφέρονται στην ίδια διεύθυνση μνήμης, για τον λόγο αυτό ενώ εμείς θα δηλώσουμε δύο δείκτες, τους pBuf και pFunc θα δεσμεύσουμε μνήμη μόνο στον ένα και αυτήν την μνήμη στο τέλος θα την απελευθερώσουμε.
Στην γραμμή 3 δηλώνεται η συνάρτηση getwd().
Στις γραμμές 7 και 8 δηλώνονται δύο δείκτες, ο pFunc και pBuf.
Στην γραμμή 9 δηλώνεται μία μεταβλητή string που θα κρατά την επιστροφή της συνάρτησης, δλδ τον τρέχοντα κατάλογο.
Στο manual της getwd διαβάζουμε ότι το μήκος του buffer που διοχετεύεται σαν παράμετρος στην συνάρτηση θα πρέπει να έχει μήκος όσο ορίζει η σταθερά **PATH_MAX**. Για να βρούμε την τιμή της σταθεράς θα πρέπει να ψάξουμε την έκφραση **PATH_MAX** στα αρχεία του καταλόγου /usr/include.

Με μια εύρεση βρίσκουμε στο αρχείο /usr/include/limits.h την πληροφορία,
#define PATH_MAX 4096 /* # chars in a path name including nul */

Έτσι στην γραμμή 12 δεσμεύουμε 4096 bytes για τον δείκτη pBuf.
Στην γραμμή 14 καλούμε την συνάρτηση getwd().
Στην γραμμή 16 διαβάζουμε την τιμή της μνήμης στην οποία η συνάρτηση έχει επιστρέψει το αποτέλεσμα.
Στην γραμμή 17 τυπώνουμε το αποτέλεσμα στην stdout.

Για να δούμε τι διευθύνσεις έχουν εκχωρηθεί στους δείκτες τυπώνουμε τις τιμές τους στην stdout.
Βλέπουμε ότι η συνάρτηση getwd() έχει θέσει την ίδια τιμή στους δύο δείκτες.

Έτσι στην γραμμή 22 απελευθερώνουμε την μνήμη από την διεύθυνση που δείχνει ο δείκτης

pBuf.

Σε αυτό το σημείο δεν έχει νόημα να γράψουμε Free(pFunc) διότι η διεύθυνση έχει ήδη απελευθερωθεί μιας και οι δύο δείκτες έδειχναν στην ίδια διεύθυνση και από την άλλη στην γραμμή 12 δεσμεύσαμε με την Alloc() μόνο μία διεύθυνση μνήμης.

Μετά την χρήση των δεικτών, την απελευθέρωση της μνήμης είναι σειρά να μηδενίσουμε τους δείκτες, στις γραμμές 23 και 24, ακολουθώντας το πρότυπο ασφαλούς προγραμματισμού.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project77. Το παράδειγμα αυτό επιδεικνύει ένα λάθος.

```
1   ' Gambas module file
2   'char *get_current_dir_name(void);
3   Extern get_current_dir_name() As Pointer In "libc:6"
4
5   Public Sub Main()
6
7     Dim pFunc As Pointer
8     Dim sWorkingDirectory As String
9
10  'get_current_dir_name
11    pFunc = Alloc(4096)
12    Print pFunc
13
14    pFunc = get_current_dir_name()
15    Print pFunc
16
17    sWorkingDirectory = Str@(pFunc)
18    Print sWorkingDirectory
19
20    Free(pFunc)
21    pFunc = 0
22
23  End
```

Έξοδος:

```
(Pointer 0x9d9918)
(Pointer 0x9db780)
/mnt/store/LINUX/Gambas3/Projects/Project86
*** glibc detected *** Project87: free(): invalid next size (fast):
0x00000000009db720 ***
======= Backtrace: =========
....
```

Ανάλυση:
Στην γραμμή 2 έχουμε σε σχόλιο την δήλωση της συνάρτησης της C.
Η συνάρτηση get_current_dir_name() δεσμεύει μια περιοχή μνήμης και επιστρέφει την διεύθυνση της σε έναν δείκτη όπου το περιεχόμενο της είναι ένα string και πιο συγκεκριμένα ο τρέχων κατάλογος της διεργασίας.

Το παράδειγμα αυτό επιδεικνύει ένα λάθος προγραμματισμού.
Οι συναρτήσεις της C μπορεί να επιστρέφουν έναν δείκτη με το να δεσμεύουν ή όχι μνήμη.
Όταν δεν δεσμεύουν μνήμη θα πρέπει εμείς στην Gambas να δεσμεύουμε μια περιοχή μνήμης με την Alloc() και να την απελευθερώνουμε μετά την χρήση με την Free().

Όταν όμως η συνάρτηση της C εσωτερικά δεσμεύει μια περιοχή μνήμης και επιστρέφει έναν δείκτη δεν θα πρέπει εμείς να δεσμεύουμε μνήμη.
Αυτό γίνεται διότι η συνάρτηση της C θα εκχωρήσει εκ νέου νέο δείκτη στον υπάρχων που εμείς είχαμε εκχωρήσει με την Alloc(). Αυτό σημαίνει ότι η διεύθυνση που θα έχει ο δείκτης δεν θα έχει δεσμευτεί από την Gambas και δεν θα μπορεί να απελευθερωθεί με την Free() αλλά και η πρώτη διεύθυνση θα γίνει πλέον μη προσβάσιμη για να απελευθερωθεί με την Free().

Στην γραμμή 3 δηλώνουμε την συνάρτηση get_current_dir_name() η οποία επιστρέφει έναν δείκτη.
Στην γραμμή 7 δηλώνουμε έναν δείκτη.
Στην γραμμή 8 δηλώνουμε ένα string.
Στην γραμμή 11, φυσιολογικά όπως θα κάναμε, δεσμεύουμε μνήμη για τον δείκτη που δηλώσαμε.
Στην γραμμή 12 τυπώνουμε την διεύθυνση έχει εκχωρηθεί σαν τιμή στον δείκτη.
Στην γραμμή 14 καλούμε την συνάρτηση.
Σε αυτό το σημείο η συνάρτηση όπως διαβάζουμε στο manual δεσμεύει μνήμη και επιστρέφει έναν δείκτη. Έτσι η τιμή του pFunc **αλλάζει**. Η προηγούμενη διεύθυνση που είχαμε εκχωρήσει γίνεται μη προσβάσιμη και δεν θα μπορεί να απελευθερωθεί. Από την άλλη η νέα διεύθυνση δεν έχει δεσμευτεί από την Gambas και δεν θα μπορεί να απελευθερωθεί.
Στην γραμμή 17 εκχωρούμε στην sWorkingDirectory την τιμή που υπάρχει στην διεύθυνση που δείχνει ο δείκτης pFunc.
Στην γραμμή 18 τυπώνουμε την τιμή αυτή.
Στην γραμμή 20 προσπαθούμε να απελευθερώσουμε έναν δείκτη που όμως δεν έχει δεσμευτεί από την Gambas και έτσι το πρόγραμμα μας καταρρέει.

Οι επόμενες γραμμές δεν εκτελούνται.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project78. Το παράδειγμα αυτό δείχνει την διόρθωση του Project77.

```
1   ' Gambas module file
2   'char *get_current_dir_name(void);
3   Extern get_current_dir_name() As Pointer In "libc:6"
4   'void free(void *ptr);
5   Extern free_ptr(ptr As Pointer) In "libc:6" Exec "free"
6
7
8   Public Sub Main()
9
10    Dim pFunc As Pointer
11    Dim sWorkingDirectory As String
12
13  'get_current_dir_name
14
15    pFunc = get_current_dir_name()
16    Print pFunc
17
18    sWorkingDirectory = Str@(pFunc)
19    Print sWorkingDirectory
20
21    free_ptr(pFunc)
22    pFunc = 0
23
24  End
```

Έξοδος:

```
(Pointer 0x9d9918)
/mnt/store/LINUX/Gambas3/Projects/Project86
```

Ανάλυση:
Στην γραμμή 2 έχουμε σε σχόλιο την δήλωση της συνάρτησης της C.
Στην γραμμή 3 δηλώνουμε την συνάρτηση get_current_dir_name() η οποία επιστρέφει έναν δείκτη.
Στην γραμμή 4 έχουμε σε σχόλιο την δήλωση της free της συνάρτησης C.
Στην γραμμή 5 την δήλωση της συνάρτησης free.
Στο παράδειγμα αυτό θα πρέπει να απελευθερώσουμε την δεσμευμένη μνήμη με την καλώντας μια άλλη συνάρτηση της C.
Στην γραμμή 10 δηλώνουμε έναν δείκτη.
Στην γραμμή 11 δηλώνουμε ένα string.
Από την στιγμή που η συνάρτηση δεσμεύει μνήμη και επιστρέφει έναν δείκτη δεν θα πρέπει εμείς να εκχωρήσουμε κάποια τιμή στον δείκτη αυτόν αλλά θα πρέπει να αφήσουμε η συνάρτηση να θέσει την δικιά της τιμή.
Έτσι δεν θα χρησιμοποιήσουμε τις συναρτήσεις Alloc() και κατά συνέπεια την Free() της

Gambas. Ωστόσο στο manual αναφέρεται ότι θα πρέπει να απελευθερωθεί η διεύθυνση χρησιμοποιώντας την συνάρτηση free της C.
Στην γραμμή 15 καλείται η συνάρτηση η οποία εκχωρεί μια τιμή δείκτη στον δείκτη.
Στην γραμμή 16 τυπώνουμε την διεύθυνση που έχει ο δείκτης.
Στην γραμμή 18 διαβάζουμε την περιοχή αυτή που είναι αποδεσμευμένη από την συνάρτηση.
Στην γραμμή 19 τυπώνουμε την τιμή της μεταβλητής sWorkingDirectory.
Στην συνέχεια απελευθερώνουμε τον δείκτη με την συνάρτηση free_ptr().
Τέλος μηδενίζουμε τον δείκτη στην γραμμή 18.

Έτσι είδαμε πως υπάρχουν περιπτώσεις που χρησημοποιώντας δείκτες σε εξωτερικές συναρτήσεις θα πρέπει να δεσμεύουμε μνήμη και να την απελευθερώνουμε με την εντολή Free() της Gambas, άλλες φορές δεν θα πρέπει να δεσμεύουμε μνήμη διότι αυτό γίνεται από τις συναρτήσεις της C. Σε αυτές τις περιπτώσεις δεν μπορούμε να απελευθερώσουμε την μνήμη με την εντολή Free() της Gambas. Θα πρέπει σε αυτές τις περιπτώσεις να διαβάσουμε στο manual της συνάρτησεις αν θα πρέπει να απελευθερωθεί η μνήμη και αν ναι με ποιο τρόπο, π.χ. Με την κλήση άλλης συνάρτησης ή με την εντολή free() της C.

Μέρος 3ο, Αντικειμενοστραφής προγραμματισμός (OOP)

Κεφάλαιο 1ο – Τι είναι οι κλάσεις και τα αντικείμενα

Μέχρι τώρα είδαμε στα προηγούμενα κεφάλαια του βιβλίου τις εντολές – οδηγίες (instructions) της γλώσσας Gambas όπως οι λέξεις DIM, LET, PRINT, είδαμε τον ορισμό μεταβλητών, πινάκων, δομών, τον έλεγχο ροής του προγράμματος με εντολές σαν τις IF, FOR...NEXT, DO...WHILE, SELECT, καθώς επίσης και τις εγγενής συναρτήσεις που εκτελούν συγκεκριμένες εργασίες όπως οι Abs, Acos, Dir κτλ.

Όμως όλα αυτά δεν είναι επαρκή για να γράψουμε σύγχρονες εφαρμογές όπως αυτές με γραφική διεπαφή με φόρμες, κουμπιά, πλαίσια κειμένου και άλλα.
Η Gambas είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που σημαίνει ότι διαχειρίζεται αντικείμενα. Ένα αντικείμενο είναι για παράδειγμα μια φόρμα ή ένα κουμπί σε μια φόρμα. Ένα αντικείμενο δεν είναι απαραίτητο να έχει γραφική διεπαφή όπως το να είναι ένα κουμπί με σχήμα ορθογώνιο παραλληλόγραμμο και κείμενο ή κάποια εικόνα.

	Τα αντικείμενα είναι μια δομή δεδομένων που παρέχουν **ιδιότητες, μεταβλητές, σταθερές, μεθόδους και συμβάντα**.

Κάθε αντικείμενο για να δημιουργηθεί χρειάζεται μια κλάση η οποία περιγράφει την λειτουργία των ιδιοτήτων, μεθόδων και των συμβάντων.
Η κλάση δεν έχει διεύθυνση στην μνήμη του υπολογιστή ούτε γραφική διεπαφή.
Τα αντικείμενα που δημιουργούνται από μια κλάση έχουν διευθύνσεις στην μνήμη και μπορεί να έχουν ή όχι γραφική διεπαφή.

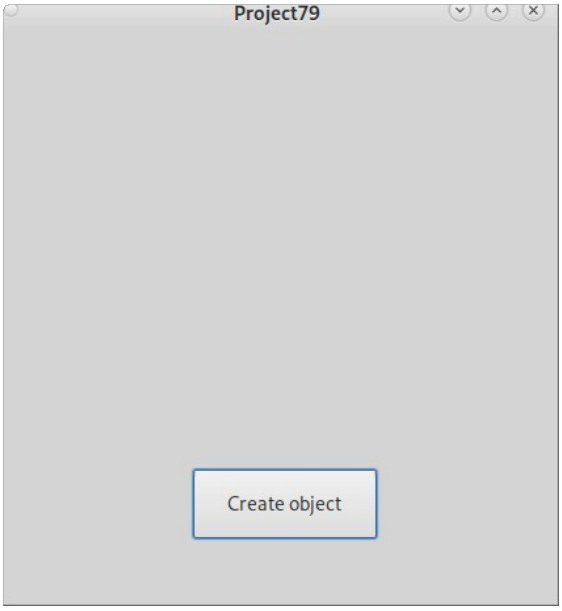
Η Gambas παρέχει εγγενείς κλάσεις, δλδ έτοιμες κλάσεις προς χρήση αλλά δίνει την δυνατότητα στον προγραμματιστή να δημιουργήσει και αυτός τις δικές του.

Για παράδειγμα, η κλάση Textbox μας δίνει την δυνατότητα να δημιουργήσουμε πλαίσια κειμένου σε μια φόρμα. Αφού δημιουργήσουμε ένα αντικείμενο πρέπει να του δώσουμε αρχικές τιμές όπως όνομα, θέση x,y πάνω στην φόρμα, ύψος και πλάτος. Μετά την αρχικοποίηση το αντικείμενο είναι έτοιμο να προβληθεί πάνω στην φόρμα.
Αν δεν δώσουμε αρχικές τιμές η Gambas δίνει κάποιες δικές της ή αφήνει κάποιες κενές, για αυτό το αντικείμενο χωρίς αρχικές τιμές δεν θα γίνει ορατό στην φόρμα θα υπάρχει ωστόσο στην μνήμη του υπολογιστή. Ο προγραμματισμός που χειρίζεται αντικείμενα ονομάζεται αντικειμενοστραφής προγραμματισμός, **Object-oriented programming** (**OOP**).

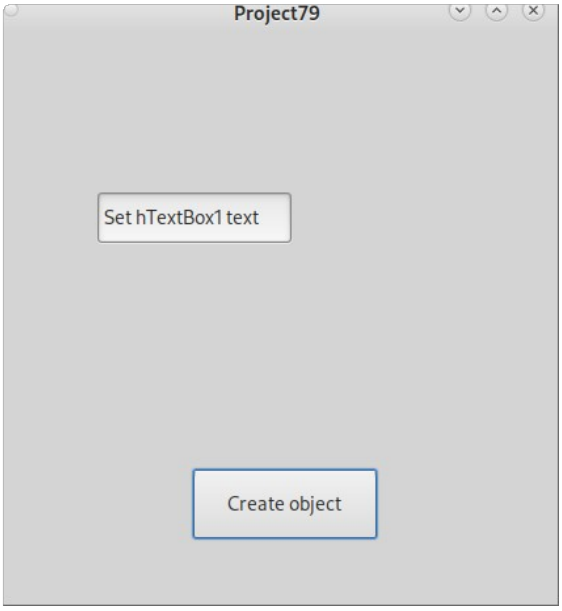
Παράδειγμα:
Το παράδειγμα αυτό είναι το Project79. Το παράδειγμα αυτό δείχνει την δημιουργία αντικειμένων από μια κλάση.

```
1  ' Gambas class file
2
3  Public Sub Button1_Click()
4
5    Dim hTextBox1 As TextBox
6
7    hTextBox1 = New TextBox(FMain)
8    hTextBox1.Text = "Set hTextBox1 text"
9    hTextBox1.X = 80
10   hTextBox1.Y = 120
11   hTextBox1.Width = 150
12   hTextBox1.Height = 40
13
14 End
```

Έξοδος:



Εικόνα 74: Πριν να πατηθεί το Button1



Εικόνα 75: Αφού έχει πατηθεί το Button1

Ανάλυση:
Στην γραμμή 5 δηλώνουμε ένα αντικείμενο TextBox το οποίο έχει το όνομα `hTextBox1`.
Στην γραμμή 7 δημιουργούμε αυτό το αντικείμενο με πατρική κλάση την φόρμα FMain.
Τα αντικείμενα τα οποία τοποθετούνται πάνω σε άλλα αντικείμενα ονομάζονται “παιδιά” και τα δεύτερα “γονείς”. Έτσι η φόρμα FMain είναι το πατρικό αντικείμενο για το αντικείμενο `hTextBox1`.
Το αντικείμενο τύπου TextBox έχει κάποιες ιδιότητες όπως κείμενο (Text), συντεταγμένες X,Y πάνω στην φόρμα, πλάτος (Width) και ύψος (Height).
Αυτές τις ιδιότητες τις αρχικοποιούμε στις γραμμές 8-12.

Αφού γίνει ο ορισμός των αρχικών τιμών (αρχικοποίηση) το αντικείμενο προβάλλεται πάνω στην φόρμα.

Στο παράδειγμα αυτό στην γραμμή 5 το hTextbox1 είναι το αντικείμενο και το TextBox είναι ο τύπος του, δλδ τι αντικείμενο είναι. Ο τύπος του αντικειμένου είναι η κλάση του.

Δημιουργία αντικειμένων από κλάσεις

Από μια κλάση μπορούμε να δημιουργήσουμε πολλά αντικείμενα. Είναι με άλλα λόγια το καλούπι με το οποίο δημιουργούμε άλλα αντικείμενα όπως η μήτρα ενός χρυσοχόου που βγάζει δαχτυλίδια ή η σφραγίδα που αφήνει το αποτύπωμα της. Μια μήτρα μπορεί να βγάλει πολλά δαχτυλίδια ή μια σφραγίδα πολλά αποτυπώματα. Το ίδιο και με μια κλάση μπορούμε να δημιουργήσουμε πολλά αντικείμενα.

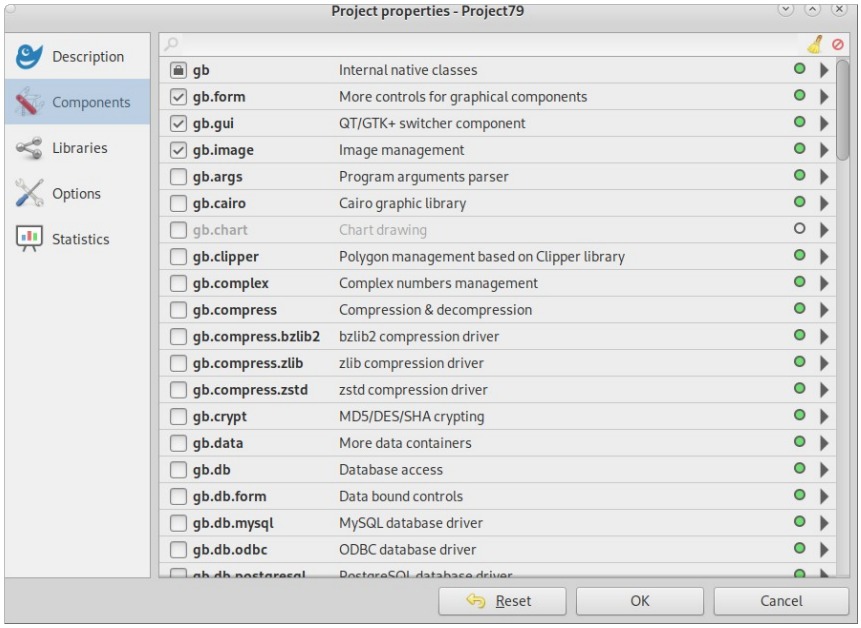
Στον αντικειμενοστραφή προγραμματισμό στην Gambas δεν δημιουργούμε on the fly τα αντικείμενα και την γραφική διεπαφή του χρήστη. Δλδ ο προγραμματιστής δεν κάθεται με κώδικα δημιουργίας αντικειμένων και αρχικοποιήσεων να κατασκευάσει μια σύνθετη φόρμα με κουμπιά, πλαίσια κειμένου κτλ. Αυτό θα ήταν αρκετά χρονοβόρο και εύκολα θα δημιουργούνταν bugs. Για αυτό ο σχεδιασμός του interface γίνεται πριν τον προγραμματισμό μέσα από το ίδιο το IDE κατά την προβολή σχεδίασης.
Ωστόσο η Gambas μας δίνει την δυνατότητα να δημιουργούμε αντικείμενα και με συγγραφή κώδικα. Αυτό βέβαια γίνεται περισσότερο για την συγγραφή δικών μας κλάσεων και την δημιουργία δικών μας αντικειμένων.

Για παράδειγμα σε ένα πρόγραμμα εξομοίωσης ψύξης θα μπορούσαμε να έχουμε την κλάση “Ψυγείο” και με αυτήν να δημιουργήσουμε αντικείμενα όπως ΨυγείοΑ, ΨυγείοΒ κτλ. Στην συνέχεια η κλάση θα δίνει σε κάθε αντικείμενο συγκεκριμένες ιδιότητες και μεθόδους για την προσομοίωση ενός δικτύου ψύξης ή κλιματισμού.
Μια τέτοια κλάση δεν είναι αναγκαστικό να παρέχει αντικείμενα με γραφική διεπαφή. Αρκεί η λειτουργία των αντικειμένων και η παρακολούθηση των τιμών τους όπως θερμοκρασία, στάθμη φρέον κτλ τα οποία μπορεί να είναι αριθμητικές τιμές.
Από την άλλη αν θέλουμε να κατασκευάσουμε κλάσεις που θα παρέχουν γραφική απεικόνιση όπως κουμπιά για ένα παιχνίδι τότε η Gambas μας δίνει και αυτήν την δυνατότητα.

Εγγενείς κλάσεις (Native classes)

Οι κλάσεις που παρέχει η Gambas οργανώνονται σε συλλογές κλάσεων οι οποίες ονομάζονται “εξαρτήματα” (Componets).
Ποια εξαρτήματα θα χρησιμοποιήσουμε στην εφαρμογή μας εξαρτάται από τις ανάγκες μας. Αν δημιουργήσουμε γραφική διεπαφή τότε θα χρειαστούμε το εξάρτημα gb.qt4 ή gb.gtk. Αν δημιουργήσουμε εφαρμογή για δίκτυο τότε θα πρέπει να χρησιμοποιήσουμε το εξάρτημα gb.net κτλ.

Τα εξαρτήματα που παρέχει η Gambas τα διαλέγουμε από την λίστα εξαρτημάτων η οποία εμφανίζεται από το μενού **Project -> Properties -> Components**.



Εικόνα 76: Η λίστα των εξαρτημάτων

Ορισμένα από τα εξαρτήματα που παρέχει η Gambas είναι τα παρακάτω:

Εμπλουτισμένα συστατικά διερμηνευτή

gb	Εσωτερικές εγγενείς κλάσεις
gb.debug	Βοηθητικές κλάσεις αποσφαλματωτή.
gb.eval	Κλάσεις υπολογισμού εκφράσεων
gb.info	Πληροφορίες σχετικά με τα εξαρτήματα
gb.option	Αναλυτής γραμμής εντολών μορφής GNU.
gb.pcre	Συμβατές κανονικές εκφράσεις της Perl. (Perl Compatible Regular Expression, PCRE)
gb.settings	Διαχείριση ρυθμίσεων εφαρμογής.
gb.signal	Διαχείριση σημάτων λειτουργικών τύπου POSIX
gb.vb	Συμβατότητα με την Visual-Basic.

Επιφάνεια εργασίας

gb.desktop	Ρουτίνες για το Desktop βασισμένες στο έργο Portland

Γραφική διεπαφή χρήστη

gb.form	Βελτιωμένα στοιχεία ελέγχου
gb.form.dialog	Βελτιωμένη έκδοση των βασικών παραθύρων διαλόγου.
gb.form.mdi	Διεπαφή πολλών εγγράφων
gb.gtk	Υλοποίηση εργαλείων GTK+
gb.gui	QT/GTK+ συστατικό εναλλαγής
gb.pdf	PDF αποτυπωτής βασισμένος στο Poppler
gb.qt4	Υλοποίηση εργαλείων QT
gb.qt4.ext	Επέκταση εργαλείων QT

Συμπίεση & Κρυπτογράφιση

gb.compress	Βιβλιοθήκη συμπίεσης
gb.crypt	MD5/DES κρυπτογράφηση

SDL προγραμματισμός

gb.sdl	Βιβλιοθήκη διεπαφής SDL
gb.sdl.sound	Βιβλιοθήκη ήχου βασισμένη στο SDL

Επικοινωνίες και δίκτυο

gb.dbus	D-Bus διεπαφή
gb.net	Πρόσβαση δικτύου.
gb.net.curl	Διαχείριση υψηλού επιπέδου δικτύου βασισμένη σε βιβλιοθήκη Curl.
gb.net.smtp	Πελάτης SMTP
gb.qt4.webkit	Στοιχείο του web-browser βασισμένο στο WebKit
gb.web	Ανάπτυξη εφαρμογής Web.

2D Γραφικά

gb.cairo	Διανυσματική σχεδίαση βασισμένη στο Cairo
gb.image	Επεξεργασία εικόνας
gb.image.imlib	Ρουτίνες βασισμένες στη βιβλιοθήκη Imlib2
gb.image.io	Ρουτίνες φόρτωσης και αποθήκευσης γραφικών
gb.image.effect	Ρουτίνες φίλτρων εικόνων
gb.v4l	Σύλληψη Video βασισμένη στο Video4Linux

3D Γραφικά

gb.opengl	Βιβλιοθήκη διεπαφής OpenGL
gb.qt4.opengl	OpenGL για την εργαλειοθήκη QT
gb.opengl.glsl	OpenGL shaders βιβλιοθήκη

Βάση δεδομένων

gb.db	Πρόσβαση βάσης δεδομένων.
gb.db.form	Στοιχεία ελέγχου σύνδεσης δεδομένων
gb.mysql	MySQL ειδικές ρουτίνες

XML προγραμματισμός

gb.xml	Εργαλεία XML βασισμένα στο libxml
gb.xml.xslt	Εργαλεία XSLT βασισμένα στο libxml
gb.xml.rpc	Υλοποίηση πρωτοκόλλου XML-RPC

Αν και εδώ παρουσίασαμε ορισμένα από τα εξαρτήματα, η Gambas3 παρέχει και άλλα.
Για περισσότερες πληροφορίες δείτε την ιστοσελίδα https://gambaswiki.org/wiki/comp

Για παράδειγμα το εξάρτημα **gb.qt4** παρέχει τις εξής κλάσεις:

Action	Drag	Line	RadioButton	TextLabel
Align	Draw	ListBox	Rect	ToggleButton
Application	DrawingArea	ListView	Screen	ToolButton
Arrange	Embedder	Menu	Screens	TrayIcon
Border	Fill	Message	Scroll	TrayIcons
Button	Font	Mouse	ScrollBar	TreeView
CheckBox	Fonts	MovieBox	ScrollView	UserContainer
Clipboard	Form	Paint	Select	UserControl
Color	Frame	PaintBrush	Separator	VBox
ColumnView	GridView	PaintExtents	Slider	VPanel
ComboBox	HBox	PaintMatrix	SpinBox	VSplit
Container	HPanel	Panel	Style	Watcher
Control	HSplit	Picture	SvgImage	Window
Cursor	Image	PictureBox	TabStrip	WindowType
Desktop	Key	Printer	TextArea	Windows
Dialog	Label	ProgressBar	TextBox	

Η αναλυτική περιγραφή των κλάσεων των εξαρτημάτων γίνεται στην wiki-εγκυκλοπαίδεια της Gambas που λειτουργεί σαν online help system.

Κάθε αντικείμενο έχει ιδιότητες, μεθόδους, συμβάντα, μεταβλητές και σταθερές.

Όπως είπαμε κάθε αντικείμενο έχει ιδιότητες, μεθόδους, συμβάντα μεταβλητές και σταθερές. Τα στοιχεία αυτά τα παρέχει μια κλάση και αποτελούν όπως ονομάζονται “μέλη” της κλάσης.

Μεταβλητές (Variables)

Μια μεταβλητή είναι ένα μέλος μιας κλάσης στο οποίο μπορούμε να εκχωρήσουμε τιμές και έχει τύπο δεδομένων έναν από τους υποστηριζόμενους τύπους δεδομένων της Gambas όπως για παράδειγμα Integer, Float, String, Boolean κτλ.

Μια μεταβλητή μέλος μιας κλάσης μπορεί να είναι ιδιωτική (Private) ή δημόσια (Public). Οι μεταβλητές μέλη των κλάσεων πάντα είναι καθολικές και όχι τοπικές όπως οι μεταβλητές μιας συνάρτησης ή μιας ρουτίνας.

Για να προσπελάσουμε μια μεταβλητή μέλος ενός αντικειμένου γράφουμε το όνομα του αντικειμένου μετά μια τελεία και στην συνέχεια το μέλος μεταβλητή.

Παράδειγμα:

```
1   ' Gambas class file
2
3   Public Name As String
4   Public X As Integer
5   Public Y As Integer
6   Private ID As Integer
7
8   Public Function Sum(a As Integer, b As Integer) As Integer
9
10  Dim c As Integer
11
12  c = a + b
13  Return c
14
15  End
```

Ανάλυση:
Στο παραπάνω παράδειγμα έχουμε μια κλάση η οποία έχει δημόσιες και ιδιωτικές μεταβλητές τις Name, X, Y, ID οι οποίες είναι καθολικές και την εντολή c η οποία είναι τοπική στην συνάρτηση Sum().

Στην κλάση αυτή οι καθολικές μεταβλητές είναι μέλη της κλάσης και μπορούν να προσπελαστούν. Αν παράδειγμα έχουμε δημιουργήσει ένα αντικείμενο MyObject με την κλάση αυτή θα μπορούσαμε να προσπελάσουμε τις μεταβλητές μέλη ως εξής:

```
MyObject.Name=”Kostas”
PRINT MyObject.Name
```

Κάτι που δεν μπορούμε να κάνουμε με την τοπική μεταβλητή της συνάρτησης Sum().
Από τις τέσσερις καθολικές μεταβλητές η ID είναι ιδιωτική και έτσι δεν μπορούμε να την προσπελάσουμε από κάποιο αντικείμενο. Δλδ δεν θα μπορούσαμε να γράψουμε κάτι σαν και το παρακάτω.

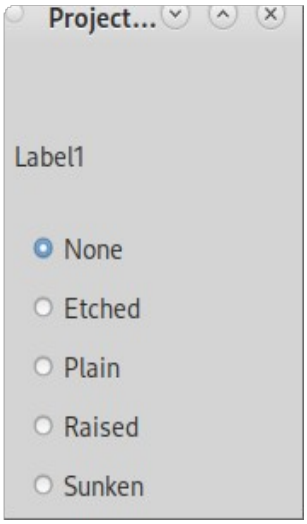
```
MyObject.ID=4
```

Σταθερές (Constants)

Μια σταθερά είναι ένα μέλος μιας κλάσης το οποίο όμως δεν αλλάζει την τιμή που του έχει εκχωρηθεί.

Για να προσπελάσουμε μια σταθερά μέλος ενός αντικειμένου γράφουμε το όνομα του αντικειμένου μετά μια τελεία και στην συνέχεια το μέλος σταθερά.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project80.
Η εφαρμογή αυτή έχει μια ετικέτα (Label) Label1 και 5 ραδιοπλήκτρα (RadioButtons) τα rdbNone, rdbEtched, rdbPlain, rdbRaised, rdbSunken. Ο χρήστης όταν κάνει κλικ σε ένα από τα ραδιοπλήκτρα αλλάζει το περίγραμμα της ετικέτας Label1.
Οι ετικέτες για να καθορίσουν την μορφή του περιγράμματος τους χρησιμοποιούν σταθερές από την κλάση Border.
Η κλάση Border παρέχει τις σταθερές **None**, **Plain**, **Sunken**, **Raised**, **Etched** οι οποίες είναι ακέραιοι με τιμές 0, 1, 2, 3, 4. Για να θέσουμε μια τιμή στην ιδιότητα Border μιας ετικέτας χρησιμοποιούμε αυτές τις σταθερές.



Εικόνα 77: Η διεπαφή του Project80

Ο κώδικας της εφαρμογής είναι ο παρακάτω:

```
1   ' Gambas class file
2
3   Public Sub Form_Open()
4
5   Me.Center
6
7   End
8
9   Public Sub rdbNone_Click()
10
11    Label1.Border = Border.None
12
13  End
14
15  Public Sub rdbEtched_Click()
16
17    Label1.Border = Border.Etched
18
19  End
20
21  Public Sub rdbPlain_Click()
22
23    Label1.Border = Border.Plain
24
25  End
26
```

```
27  Public Sub rdbRaised_Click()
28
29    Label1.Border = Border.Raised
30
31  End
32
33  Public Sub rdbSunken_Click()
34
35    Label1.Border = Border.Sunken
36
37  End
```

Ανάλυση:
Στις γραμμές 3-7 τοποθετούμε την φόρμα στο κέντρο της οθόνης όταν η φόρμα ανοίγει.
Στις γραμμές 9-37 έχουμε τις ρουτίνες των ραδιοπλήκτρων όταν αυτά τα κάνει ο χρήστης Click.

Κάθε ρουτίνα θέτει το Border του Label1, ανάλογα το ποιο ραδιοπλήκτρο είναι, σε μια τιμή σταθεράς από την κλάση Border.
Το Label1.Border είναι η ιδιότητα του αντικειμένου Label1. Τα Border.Sunken, Border.Raised, Border.Plain κτλ είναι σταθερές μέλη της κλάσης Border.
Οι σταθερές αυτές έχουν τιμές ακεραίων που δεν αλλάζουν.

Ιδιότητες (Properties)

Μια ιδιότητα είναι ένα μέλος μιας κλάσης στο οποίο μπορούμε να εκχωρήσουμε τιμές όπως ακριβώς σε μια μεταβλητή. Μια ιδιότητα μπορεί να έχει τύπο δεδομένων από τους τύπους δεδομένων που υποστηρίζει η Gambas για τις μεταβλητές.
Οι ιδιότητες ενός αντικειμένου είναι αυτές που εμφανίζονται στα δεξιά στην λίστα ιδιοτήτων στο περιβάλλον ανάπτυξης.

Η διαφορά μιας ιδιότητας και μια μεταβλητής ως μέλη της κλάσης είναι ότι οι ιδιότητες που παρέχει μια κλάση μπορούν να οριστούν και να προγραμματιστούν κατά την σχεδίαση της διεπαφής μιας εφαρμογής από την λίστα ιδιοτήτων στο περιβάλλον ανάπτυξης εφαρμογών ενώ οι μεταβλητές δεν εμφανίζονται σε αυτήν την λίστα.

Όταν θα δημιουργήσετε δικές σας κλάσεις για να διαβάσετε την τιμή μιας ιδιότητας θα πρέπει να δημιουργήσετε μια ρουτίνα που θα αποτελείται από το όνομα της ιδιότητας μια κάτω παύλα και την λέξη READ.
Η ρουτίνα αυτή δεν δέχεται παραμέτρους και πρέπει να επιστρέφει μια τιμή με τύπο δεδομένων τον ίδιο με τον τύπο δεδομένων της ιδιότητας.
Για παράδειγμα αν έχετε μια ιδιότητα Enabled που δείχνει αν το αντικείμενο είναι ενεργό ή όχι και ο τύπος δεδομένων είναι Boolean τότε η ρουτίνα ανάγνωσης θα πρέπει να επιστρέφει ένα Boolean.

Ομοίως για να γράψετε μια τιμή σε μια ιδιότητα θα πρέπει να δημιουργήσετε μια ρουτίνα που θα αποτελείται από το όνομα της ιδιότητας μια κάτω παύλα και την λέξη WRITE.
Η ρουτίνα αυτή δεν επιστρέφει κάποια τιμή και δέχεται μόνο μια παράμετρο με τύπο δεδομένων τον ίδιο με τον τύπο δεδομένων της ιδιότητας.
Για παράδειγμα αν έχετε μια ιδιότητα Enabled που δείχνει αν το αντικείμενο είναι ενεργό ή όχι και ο τύπος δεδομένων είναι Boolean τότε η ρουτίνα εγγραφής θα πρέπει να δέχεται ένα Boolean για παράμετρο της.

Μια ιδιότητα μπορεί να είναι μόνο για ανάγνωση τότε δεν μπορεί ο χρήστης να γράψει σε αυτήν και δεν έχει ρουτίνα εγγραφής.

Για να προσπελάσουμε μια ιδιότητα μέλος ενός αντικειμένου γράφουμε το όνομα του αντικειμένου μετά μια τελεία και στην συνέχεια το μέλος ιδιότητα.

Για παράδειγμα:

```
Label1.Border
```

Το Label1 είναι το αντικείμενο μας. Το Border είναι η ιδιότητα του αντικειμένου. Για να θέσουμε ή να διαβάσουμε την τιμή της ιδιότητας γράφουμε το όνομα του αντικειμένου μια τελεία και το όνομα της ιδιότητας.

Μέθοδοι (Methods)

Μια μέθοδος είναι ένα μέλος μιας κλάσης το οποίο υλοποιεί μια ρουτίνα η οποία είναι δημόσια και μπορεί να επιστρέφει ή όχι μια τιμή.

Για παράδειγμα το εξάρτημα gb.qt4 παρέχει την κλάση Form η οποία έχει τις εξείς μεθόδους: Center Close Delete Drag Find Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled SetFocus Show ShowDialog ShowModal ShowPopup.

Οι μέθοδοι αυτοί είναι δημόσιες ρουτίνες της κλάσης Form.
Οι περισσότερες δεν επιστρέφουν κάποια τιμή απλά εκτελούν ένα μέρος κώδικα όπως για παράδειγμα η μέθοδος Hide που κρύβει το αντικείμενο χωρίς να επιστρέφει κάποια τιμή.
Ώμος η μέθοδος Find επιστρέφει μια τιμή όταν κληθεί.

Οι μέθοδοι μπορούν να δέχονται παραμέτρους όπως και οι συναρτήσεις ή όχι.
Οι παράμετροι μπορούν να είναι προαιρετικοί ή όχι να περνιούνται ByValue ή ByReference.

Μια μέθοδος μπορούμε να την προσπελάσουμε γράφοντας το όνομα του αντικειμένου μια τελεία και το όνομα της μεθόδου με παρενθέσεις. Εντός των παρενθέσεων γράφουμε τις παραμέτρους που η μέθοδος δέχεται.

Για παράδειγμα:

```
FMain.Hide()
```

Το FMain είναι το αντικείμενο μας. Το Hide() είναι η μέθοδος του αντικειμένου.

Συμβάντα (Events)

Ένα συμβάν είναι ένα μέλος μια κλάσης και αποτελεί ένα σήμα (signal) το οποίο ένα αντικείμενο μπορεί να εγείρει και να μεταβιβαστεί σε άλλο αντικείμενο το οποίο ονομάζεται παρατηρητής συμβάντων (event observer) και να το διαχειριστεί.
Ένα συμβάν είναι στην πραγματικότητα μια ρουτίνα που την διαχειρίζεται ο επόπτης συμβάντων (event listener) όταν αυτή κληθεί.

Τα συμβάντα εγείρονται από πράξεις που γίνονται επί τον αντικειμένων.
Για παράδειγμα αν ο χρήστης κάνει κλικ σε ένα κουμπί, αν πατήσει ένα πλήκτρο από το πληκτρολόγιο κτλ.

Ένα συμβάν μπορούμε να το προσπελάσουμε γράφοντας το όνομα του αντικειμένου μια κάτω παύλα και το όνομα του συμβάντος με παρενθέσεις.

Για παράδειγμα:

```
FMain_Click()
```

Τα συμβάντα επειδή καλούνται να εκτελέσουν κώδικα της εφαρμογής γράφονται σε συνδυασμό με την δήλωση ρουτινών.

Για παράδειγμα:

```
Public Sub btnArc_Click()
...
End
```

Το αντικείμενο `btnArc` έχει το συμβάν Click() το οποίο για να το προγραμματίσουμε το δηλώνουμε σαν δημόσια ρουτίνα.

Στατικές κλάσεις (Static classes)

Οι κλάσεις συνήθως αποτελούν τον τύπο δεδομένων με τον οποίο δημιουργούνται τα αντικείμενα. Υπάρχουν περιπτώσεις όμως που οι κλάσεις είναι στατικές και δεν παρέχουν την δυνατότητα δημιουργίας αντικειμένων. Τις στατικές κλάσεις τις χρησιμοποιούμε απευθείας με το όνομα τους και δεν δημιουργούμε με αυτές αντικείμενα.

Για παράδειγμα:

```
Dim hTextBox1 As TextBox

hTextBox1 = New TextBox(FMain)
hTextBox1.Text = "Set hTextBox1 text"

Draw.Begin(drawArea)
```

Το εξάρτημα gb.qt4 παρέχει τις κλάσεις TextBox και Draw.
Η πρώτη κλάση δεν είναι στατική και έτσι μπορούμε να δημιουργήσουμε το αντικείμενο hTextbox1.
Η δεύτερη κλάση είναι στατική και δεν μπορούμε να δημιουργήσουμε αντικείμενα.
Αντί αυτού μπορούμε να χρησιμοποιήσουμε απευθείας την κλάση και να προσπελάσουμε τις μεθόδους της κατευθείαν.

Η κλάση Draw είναι στατική και έχει την μέθοδο Begin την οποία την προσπελάζουμε απευθείας χωρίς αναφορά σε κάποιο αντικείμενο.

Εικονικές κλάσεις (Virtual Classes)

Μια εικονική κλάση είναι μια υπό-κλάση μιας άλλης κλάσης η οποία δεν μπορεί να δημιουργηθεί και να εκχωρηθεί σε μια μεταβλητή, αλλά προσπελάζεται μέσω της αρχικής κλάσης.

Εξαρτήματα που γράφονται στην C/C++
Στα εγγενή εξαρτήματα οι εικονικές κλάσεις είναι μηχανισμοί σχεδιασμένοι ώστε ο χρήστης να διαχειρίζεται προσωρινά αντικείμενα χωρίς να τα δημιουργεί. Αυτό είναι από μόνο του πολύ γρήγορο!

Το όνομα μιας εικονικής κλάσης αρχίζει με μια τελεία. Για παράδειγμα εξάρτημα gb.qt4 παρέχει την κλάση ListView. Η κλάση αυτή έχει την ιδιότητα Item που αντιπροσωπεύει ένα στοιχείο στην λίστα. Κάθε αντικείμενο είναι μια εικονική κλάση του τύπου .ListViewItem.
Η εικονική κλάση .ListViewItem παρέχει τις δικές της ιδιότητες και μεθόδους που αντιστοιχούν στις ιδιότητες και μεθόδους ενός στοιχείου από την λίστα.

Οι εικονικές κλάσεις είναι συνηθισμένοι τύποι δεδομένων για τον διερμηνευτή. Ωστόσο το αντικείμενο που τις παρέχει είναι το πραγματικό αντικείμενο που βρίσκεται από πίσω.

Τα εξαρτήματα που γράφονται στην Gambas
Η Gambas, μας επιτρέπει να δημιουργούμε και εξαρτήματα που παρέχουν αντικείμενα με εικονικές κλάσεις. Το όνομα αυτών των κλάσεων ξεκινά με μια κάτω παύλα.

Προσωπικές κλάσεις (Custom classes)

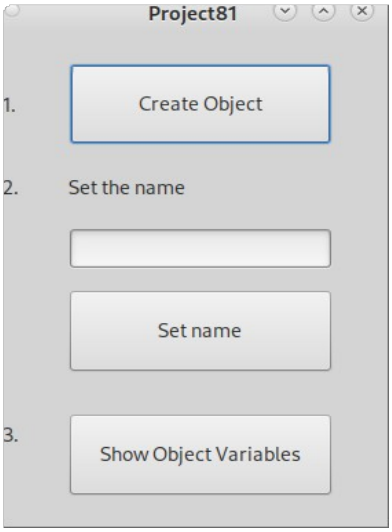
Η Gambas μας δίνει την δυνατότητα να χρησιμοποιήσουμε τις προκαθορισμένες κλάσεις των εξαρτημάτων που παρέχει αλλά και να δημιουργήσουμε δικές μας κλάσεις.

Για να δημιουργήσουμε μια προσωπική κλάση κάνουμε δεξί κλικ στον περιηγητή αρχείων και επιλέγουμε **New -> Class**.

Συνήθως τα αρχεία κλάσεων τα ονομάζουμε τοποθετώντας ένα c μπροστά από το όνομα της κλάσης που σημαίνει ότι είναι ένα αρχείο class π.χ. cName, cMachine, cCar, cThing.

Για να δούμε όλα όσα μέχρι τώρα περιγράψαμε δημιουργούμε το Project81.
Στο πρόγραμμα αυτό θα δημιουργήσουμε μια κλάση η οποία θα έχει μια ιδιότητα.
Σε μια φόρμα θα πατάμε ένα κουμπί και θα δημιουργούμε ένα αντικείμενο από την κλάση αυτή. Στην συνέχεια η φόρμα θα έχει ένα Textbox στο οποίο θα γράφουμε το όνομα του αντικειμένου και με ένα κουμπί θα εκχωρούμε την τιμή αυτή στην ιδιότητα του. Τέλος με ένα άλλο κουμπί θα προβάλλουμε ένα μήνυμα που θα περιέχει την τιμή της ιδιότητας του αντικειμένου.

Η γραφική διεπαφή του προγράμματος είναι η παρακάτω.



Εικόνα 78: Η διεπαφή του Project81

Δημιουργούμε το Project81 και ένα αρχείο class με το όνομα cMachine.

Ο κώδικας της εφαρμογής είναι ο παρακάτω:

Το αρχείο cMachine.class:

```
1  ' Gambas class file
2
3  Property Name As String
4
5  Private $sName As String
6
7  Function Name_Read() As String
8
9    Return $sName
10
11 End
12
13 Sub Name_Write(sName As String)
14
15   $sName = sName
16
17 End
```

Ανάλυση:
Στην γραμμή 3 δηλώνουμε μια ιδιότητα με το όνομα Name και τον τύπο String.
Στην γραμμή 5 δηλώνουμε μια ιδιωτική καθολική μεταβλητή τύπου String. Η μεταβλητή αυτή θα χρησιμοποιηθεί από το πρόγραμμα για να εκχωρείται η τιμή τύπου string στην ιδιότητα Name και για να διαβάζεται η ιδιότητα Name.
Στις γραμμές 7-11 υπάρχει η ρουτίνα ανάγνωσης της ιδιότητας. Η ρουτίνα αυτή επιστρέφει το περιεχόμενο της μεταβλητής $sName.
Στις γραμμές 13-17 υπάρχει η ρουτίνα εγγραφής της ιδιότητας. Η ρουτίνα αυτή δέχεται ως παράμετρο σε μια τοπική μεταβλητή sName την τιμή της ιδιότητας και στην γραμμή 15 η τιμή αυτή εκχωρείται στην καθολική ιδιωτική μεταβλητή $sName.

Το αρχείο Fmain.class

```
1  ' Gambas class file
2
3  Private hCompressor As CMachine
4
5  Public Sub Form_Open()
6
7    Me.Center
8
9  End
```

```
10
11
12 Public Sub btnCreateObject_Click()
13
14   hCompressor = New CMachine
15
16 End
17
18 Public Sub btnShowVars_Click()
19
20 If hCompressor <> Null Then
21   Message(hCompressor.Name)
22 Else
23   Message("Create first the object")
24 Endif
25
26 End
27
28 Public Sub btnSetName_Click()
29
30 If hCompressor <> Null Then
31   hCompressor.Name = txtName.Text
32 Else
33   Message("Create first the object")
34 Endif
35
36
37 End
```

Ανάλυση:
Στην γραμμή 3 ορίζουμε ένα καθολικό αλλά ιδιωτικό αντικείμενο τύπου CMachine.
Στις γραμμές 5-9 τοποθετούμε την εφαρμογή στο κέντρο της οθόνης όταν η φόρμα ανοίγει.
Στις γραμμές 12-16 υπάρχει η ρουτίνα του συμβάντος Click του κουμπιού btnCreateObject.
Στην γραμμή 14 δημιουργούμε ένα αντικείμενο CMachine.
Στις γραμμές 18-26 υπάρχει η ρουτίνα του συμβάντος Click του κουμπιού btnShowVars.
Εάν το αντικείμενο έχει δημιουργηθεί και δεν είναι κενό, εμφανίζουμε την ιδιότητα Name σε ένα μήνυμα.
Στις γραμμές 28-37 υπάρχει η ρουτίνα του συμβάντος Click του κουμπιού btnSetName.
Στην γραμμή 31 εάν το αντικείμενο έχει δημιουργηθεί και δεν είναι κενό, εκχωρούμε στην ιδιότητα Name του αντικειμένου CMachine την τιμή της ιδιότητας Text του αντικειμένου txtName.
Εάν το αντικείμενο CMachine δεν έχει δημιουργηθεί τότε οι ρουτίνες btnShowVars_Click(), btnSetName_Click() εμφανίζουν ένα μήνυμα λάθους.

Κεφάλαιο 2ο – Gambas Object Model

Οι αντικειμενοστραφείς γλώσσες προγραμματισμού έχουν ένα μοντέλο υλοποίησης των αντικειμένων στην γλώσσα. Το μοντέλο αντικειμένων αυτό ονομάζεται Gambas Object Model το οποίο περιγράφει τις κλάσεις, τα αντικείμενα, τις ιδιότητες, τις μεθόδους και ότι αφορά τον αντικειμενοστραφή προγραμματισμό.

Αντικείμενα (Objects) & Κλάσεις (Classes)

Έχουμε ήδη αναφερθεί στα αντικείμενα και στις κλάσεις στο προηγούμενο κεφάλαιο.
Σε αυτό το κεφάλαιο θα περιγράψουμε το μοντέλο υλοποίησης των αντικειμένων στην Gambas.
Ένα αντικείμενο στην Gambas είναι μια δομή δεδομένων η οποία παρέχει *ιδιότητες*, *μεταβλητές*, *σταθερές*, *μεθόδους* και *συμβάντα*.
Τα αντικείμενα προσπελάζονται με αναφορές σε αυτά. Για παράδειγμα χρησιμοποιώντας έναν δείκτη ή μια μεταβλητή της οποίας η τιμή δείχνει στην διεύθυνση μνήμης που βρίσκεται το αντικείμενο.

Για παράδειγμα:

```
1  Dim hCompressor As CMachine
2
3  hCompressor = New CMachine
```

Η μεταβλητή hCompressor λειτουργεί σαν μια αναφορά σε ένα αντικείμενο του τύπου CMachine.

Το νέο αντικείμενο hCompressor περιγράφεται από τον τύπο δεδομένων της κλάσης CMachine.

Για να δούμε σε ποια διεύθυνση στην μνήμη υπάρχει το αντικείμενο μπορούμε απλά να χρησιμοποιήσουμε την εντολή PRINT.

Για παράδειγμα:

```
1  Dim hCompressor As CMachine
2
3  hCompressor = New CMachine
4
5  PRINT hCompressor
```

Η εντολή PRINT θα δώσει στην προκαθορισμένη έξοδο την διεύθυνση μνήμης που βρίσκεται το αντικείμενο.

Έξοδος:

```
(CMachine 0x10e0b18)
```

Κλάσεις (Classes)

Κάθε αντικείμενο στην Gambas έχει μια κλάση που περιγράφει τις δημόσιες ιδιότητες, μεθόδους και συμβάντα. Η κλάση αποτελεί για την ίδια την γλώσσα ένα ακόμη αντικείμενο του οποίου η κλάση είναι αυτή που παρέχει το εξάρτημα gb και έχει το όνομα Class.

Οι κλάσεις έχουν ιδιότητες, μεταβλητές, σταθερές, μεθόδους και συμβάντα. Όλα αυτά αποτελούν τα μέλη της κλάσης.

Υπάρχουν τρεις κατηγορίες κλάσεων. Η δυναμικές κλάσεις (dynamic classes), οι στατικές (static classes) και οι εικονικές (virtual classes).

Μια δυναμική κλάση (dynamic class) μπορεί να δημιουργεί νέα αντικείμενα.

Μια στατική κλάση (static class) έχει όλα τα μέλη της είναι στατικά και στην Gambas ονομάζεται module. Μια στατική κλάση δεν μπορεί να δημιουργεί νέα αντικείμενα. Για τον λόγο αυτό χρησιμοποιείται απευθείας.

Για παράδειγμα, η κλάση System από το εξάρτημα gb.eval είναι στατική και δεν μπορείτε να δημιουργήσετε ένα αντικείμενο από αυτήν. Για να κάνετε χρήση αυτής την χρησιμοποιείτε απευθείας.

```
1 PRINT System.Path
```

Έξοδος:

```
/usr/local
```

Μια εικονική κλάση (virtual class) είναι μια κρυμένη ψευδο-κλάση μέσα σε ένα αντικείμενο την οποία δεν μπορείτε να δημιουργήσετε απευθείας παρά μόνο δημιουργώντας το αντικείμενο από την κλάση γονέα στην οποία ανήκει.

Ιδιότητες (properties), μέθοδοι (methods) και μεταβλητές (variables)

Οι ιδιότητες και οι μέθοδοι επιτρέπουν την διαχείριση της δομής δεδομένων.
Στην ουσία οι ιδιότητες είναι δημόσιες μεταβλητές και οι μέθοδοι ρουτίνες που εκτελούν κώδικα και εντολές της γλώσσας.

Μια ιδιότητα, μέθοδος ή μεταβλητή μπορεί να είναι στατική, **static**:

- Μια στατική μεταβλητή είναι προσβάσιμη από όλα τα στιγμιότυπα (instances) μιας κλάσης.
- Μια στατική ιδιότητα ή μέθοδος μπορεί να αλλάξει μόνο στατικές μεταβλητές.

Μια μέθοδος ή μια μεταβλητή μπορεί να είναι δημόσια (**public**) ή ιδιωτική (**private**). Μια ιδιότητα πάντα είναι δημόσια (**public**).
Ιδιωτικά σύμβολα (μεταβλητές, ρουτίνες κτλ) μπορούν να χρησιμοποιηθούν μόνο εσωτερικά από την κλάση όπως οι τοπικές μεταβλητές χρησιμοποιούνται μόνο εντός μιας συνάρτησης στην οποία έχουν δηλωθεί. Δημόσια σύμβολα μπορούν να χρησιμοποιηθούν παντού.

Αναφορές (References)

Τα αντικείμενα που δημιουργούνται από κλάσεις τοποθετούνται στην μνήμη του υπολογιστή και ο προγραμματιστής έχει πρόσβαση σε αυτά με αναφορές όπως δείκτες και μεταβλητές αντικειμένων. Όταν ένα αντικείμενο παύει να υπάρχει ή για οποιοδήποτε λόγο δεν είναι δυνατή η πρόσβαση στην περιοχή της μνήμης που υπήρχε γίνεται ένα ανενεργό αντικείμενο και παραμένει στην μνήμη μέχρι το πρόγραμμα μας να τερματιστεί.
Για την διαχείριση τέτοιων αντικειμένων στην μνήμη του υπολογιστή τα οποία θεωρούνται “αντικείμενα – σκουπίδια” πολλές γλώσσες προγραμματισμού κάνουν χρήση ενός συστήματος διαχείρισης που ονομάζεται garbage collector.
Η Gambas δεν έχει σύστημα διαχείρισης μη ενεργών αντικειμένων. Έτσι κάθε αντικείμενο έχει έναν μετρητή αναφοράς (**reference counter)** ο οποίος αυξάνει κάθε φορά που το αντικείμενο δημιουργείται και εκχωρείται σε μια μεταβλητή ή σε πίνακα ή σε συλλογή ή άλλο αντικείμενο. Ο μετρητής αυτός μειώνεται όταν κάποιο αντικείμενο παύει να υπάρχει στην μνήμη. Όταν δημιουργείται ένα αντικείμενο ο μετρητής έχει την τιμή μηδέν και γίνεται πάλι μηδέν μετά την απελευθέρωση της μνήμης και την παύση ύπαρξης του αντικειμένου.

Μη έγκυρα αντικείμενα (Invalid objects)

Τα αντικείμενα στην Gambas μπορεί να γίνουν μη έκυρα (invalid).
Όταν ένα αντικείμενο γίνεται invalid δεν καταστρέφεται αλλά δεν μπορεί να χρησιμοποιηθεί πλέον.
Συνήθως ένα αντικείμενο γίνεται μη έγκυρο επειδή η λειτουργία του εξαρτάται από ένα άλλο αντικείμενο ή σε κάποιο άλλο ξένο στοιχείο της Gambas το οποίο δεν έχει καταστραφεί.

Ειδικές μέθοδοι (Special methods)

Στον κύκλο ζωής ενός αντικειμένου από την δημιουργία μέχρι την καταστροφή του και την αφαίρεση του από την μνήμη υπάρχουν ειδικές μέθοδοι τις οποίες κάποιος μπορεί να χρησιμοποιήσει. Αυτές οι μέθοδοι δηλώνονται στις κλάσεις και το όνομα τους αρχίζει με μια κάτω παύλα (_). Τις μεθόδους αυτές τις καλεί ο διερμηνευτής (interpreter) στις παρακάτω περιπτώσεις:

- Όταν μια κλάση φορτώνεται στην μνήμη.
- Όταν ένα αντικείμενο δημιουργείται.
- Όταν ένα αντικείμενο απελευθερώνεται από την μνήμη.
- Όταν η κλάση του αντικειμένου ξεφορτώνεται στην μνήμη.
- Όταν γίνεται χρήση του αντικειμένου σαν να ήταν πίνακας.
- Όταν το αντικείμενο απαριθμείται (enumerated).
- Όταν χρησιμοποιείται το αντικείμενο σαν συνάρτηση.
- Όταν γίνεται προσπάθεια για χρήση άγνωστης μεθόδου ή ιδιότητας του αντικειμένου.
- Όταν συγκρίνονται δύο αντικείμενα της ίδιας κλάσης.
- Όταν ένα αντικείμενο προσαρτείται σε άλλη κλάση.

Υπάρχουν έντεκα διαφορετικές ειδικές μέθοδοι για τα αντικείμενα και όλες είναι προαιρετικές.

_init	Αρχικοποίηση της κλάσης.
_exit	Καθαρισμός της κλάσης από την μνήμη.
_new	Δημιουργία αντικειμένου.
_free	Απελευθέρωση αντικειμένου από την μνήμη.
_next	Απαρίθμηση αντικειμένου ή κλάσης.
_get	Ανάγνωση πίνακα.
_put	Εγγραφή πίνακα.
_call	Χρήση του αντικειμένου ή κλάσης σαν μέθοδο.
_unknown	Κλήση άγνωστης μεθόδου ή ιδιότητας.
_compare	Σύγκριση δύο αντικειμένων της ίδιας κλάσης
_attach	Όταν ένα αντικείμενο προσαρτείται ή αποπροσατρείται από την πατρική κλάση

_init

Σύνταξη:

```
Static Public Sub _init()
```

Η μέθοδος _init καλείται όταν η κλάση φορτώνεται από τον διερμηνευτή.
Η μέθοδος δεν έχει παραμέτρους και δεν επιστρέφει κάποια τιμή.
Μπορείτε να χρησιμοποιείτε αυτήν την μέθοδο για να αρχικοποιήσετε σε καθολικό επίπεδο την κλάση.

_exit

Σύνταξη:

```
Static Public Sub _exit()
```

Η μέθοδος _exit καλείται όταν η κλάση ξεφορτώνεται από τον διερμηνευτή.
Η μέθοδος δεν έχει παραμέτρους και δεν επιστρέφει κάποια τιμή.
Μπορείτε να χρησιμοποιείτε αυτήν την μέθοδο για να κάνετε το ακριβώς αντίστροφο από την αρχικοποίηση που γίνεται στην μέθοδο _init.

_new

Σύνταξη:

```
Public Sub _new()
```

Η μέθοδος _new καλείται όταν ένα νέο αντικείμενο μιας κλάσης δημιουργείται.
Η μέθοδος αυτή μπορεί να λάβει οποιαδήποτε παράμετρο και πρέπει να μην επιστρέφει κάποια τιμή.
Το δημιουργημένο αντικείμενο επιστρέφεται και το διαχειρίζεται ο διερμηνευτής.

Οι παράμετροι της μεθόδου προέρχονται από τις παραμέτρους του τελεστή NEW.
Μπορείτε να χρησιμοποιήσετε αυτήν την μέθοδο για να αρχικοποιήσετε ένα αντικείμενο που μόλις έχει δημιουργηθεί.

Η μέθοδος αυτή ονομάζεται και μέθοδος δημιουργίας του αντικειμένου (constructor)

_free

Σύνταξη:

```
Public Sub _free()
```

Η μέθοδος _free καλείται όταν ένα αντικείμενο μιας κλάσης καταστρέφεται και παύει να υπάρχει.
Η μέθοδος δεν έχει παραμέτρους και δεν επιστρέφει κάποια τιμή.
Χρησιμοποιήστε αυτήν την μέθοδο για να καθαρίσετε τιμές της κλάσης πριν την αφαίρεση του αντικειμένου από την μνήμη.
Για παράδειγμα αν ένα αντικείμενο περιέχει αναφορές σε άλλα αντικείμενα πρέπει να απελευθερώσετε αυτά τα αντικείμενα πριν την καταστροφή του αντικειμένου αλλιώς οι αναφορές αυτές θα μείνουν στην μνήμη χωρίς να είναι δυνατή η πρόσβαση και η χρήση τους δημιουργώντας έτσι διαρροή μνήμης.

Η μέθοδος αυτή ονομάζεται και μέθοδος καταστροφής του αντικειμένου (destructor)

_next

Σύνταξη:

```
Public Sub _next()

Static Public Sub _next()
```

Η μέθοδος _next καλείται όταν το πρόγραμμα χρησιμοποιεί τον βρόχο FOR EACH ... IN για να απαριθμήσει το αντικείμενο.
Η μέθοδος δεν δέχεται παραμέτρους και μπορεί να επιστρέψει απ αριθμημένα δεδομένα.
Η μέθοδος μπορεί να είναι Static τότε η κλάση απαριθμείται και όχι το αντικείμενο.
Η μέθοδος μπορεί να μην επιστρέψει κάποια τιμή, τότε γίνεται απαρίθμηση του αντικειμένου με τον βρόχο FOR EACH χωρίς το IN.

_get

Σύνταξη:

```
Public Sub _get()

Static Public Sub _get()
```

Η μέθοδος _get καλείται όταν ο τελεστής [] χρησιμοποιείται στο αντικείμενο για να εξάγει δεδομένα.
Η μέθοδος μπορεί να είναι στατική, τότε ο τελεστής [] χρησιμοποιείται στην κλάση και όχι στο αντικείμενο.
Η μέθοδος μπορεί να δεχθεί οποιαδήποτε παράμετρο. Οι παράμετροι αυτοί θα είναι αυτοί που θα διοχετευτούν μεταξύ των τελεστών [].
Για παράδειγμα η εντολή Value = MyObject[X, Y] θα καλέσει την μέθοδο _get με παραμέτρους X και Y.
Η μέθοδος πρέπει να επιστρέψει τιμές που εξάγονται από την κλάση ή το αντικείμενο.

_put

Σύνταξη:

```
Public Sub _put()

Static Public Sub _put()
```

Η μέθοδος _put καλείται όταν ο τελεστής [] χρησιμοποιείται στο αντικείμενο για να γράψει δεδομένα σε αυτό.
Η μέθοδος μπορεί να είναι στατική, τότε ο τελεστής [] χρησιμοποιείται στην κλάση και όχι στο αντικείμενο.
Η μέθοδος μπορεί να δεχθεί οποιαδήποτε παράμετρο. Οι παράμετροι αυτοί θα είναι αυτοί που θα διοχετευτούν μεταξύ των τελεστών [] και αποτελούν τις τιμές προς εγγραφή στο αντικείμενο.
Για παράδειγμα η εντολή Value = MyObject[X, Y] θα καλέσει την μέθοδο _put με παραμέτρους X και Y.
Η μέθοδος δεν επιστρέφει κάποια τιμή.

_call

Σύνταξη:

```
Public Sub _call()

Static Public Sub _call()
```

Η μέθοδος _call καλείται όταν η κλάση ή το αντικείμενο χρησιμοποιείται σαν συνάρτηση.
Η μέθοδος μπορεί να είναι στατική, τότε η κλάση χρησιμοποιείται σαν συνάρτηση και όχι το αντικείμενο.
Η μέθοδος μπορεί να δεχθεί οποιαδήποτε παράμετρο και επιστρέφει οποιαδήποτε τύπο δεδομένων.
Η κλάση Message του εξαρτήματος gb.qt είναι ένα καλό παράδειγμα της χρήσης αυτής της μεθόδου.

_unknown

Σύνταξη:

```
Public Sub _unknown()
```

Η μέθοδος _unknown καλείται όταν ο διερμηνευτής δεν μπορεί να βρει μια μέθοδο ή ιδιότητα ή άλλο σύμβολο στην δήλωση της κλάσης.
Η μέθοδος αυτή δέχεται παραμέτρους και επιστρέφει μια τιμή τύπου Variant.

Η κλάση Application του εξαρτήματος gb.qt.kde είναι ένα καλό παράδειγμα της χρήσης αυτής της μεθόδου.
Όταν κάνετε μια κλήση DCOP, ο διερμηνευτής δεν γνωρίζει εάν η εφαρμογή έχει φορτωθεί και αν η μέθοδος υπάρχει, ποιες είναι οι παράμετροι κτλ. Έτσι κάνει χρήση της μεθόδου _unknown.

_compare

Σύνταξη:

```
Public Sub _compare ( OtherObject As Datatype ) As Integer
```

Η μέθοδος _compare καλείται όταν δύο αντικείμενα της ίδιας κλάσης πρέπει να συγκριθούν. Για παράδειγμα όταν γίνεται ταξινόμηση ενός πίνακα αντικειμένων ή όταν αναζητείται ένα συγκεκριμένο αντικείμενο εντός ενός πίνακα.
Η μέθοδος αυτή είναι προαιρετική και αν δεν υλοποιηθεί, τα αντικείμενα συγκρίνονται χρησιμοποιώντας την εσωτερική τους διεύθυνση μνήμης.

Η μέθοδος αυτή επιστρέφει έναν ακέραιο:

- 0 εάν τα δύο αντικείμενα είναι ίσα.
- 1 εάν το τρέχον αντικείμενο είναι μεγαλύτερο από το άλλο αντικείμενο.
- -1 εάν το τρέχον αντικείμενο είναι μικρότερο από το άλλο αντικείμενο.

_attach

Σύνταξη:

```
Public Sub _attach ( Parent As Object , EventName As String )
```

Η μέθοδος αυτή καλείται όταν ένα αντικείμενο προσαρτείται ή αποπροσαρτείται σε ένα γονικό αντικείμενο.

- Parent είναι το γονικό αντικείμενο.
- EventName είναι το όνομα του συμβάντος, για παράδειγμα το πρόθεμα των event handlers.

Εάν το αντικείμενο αποπροσαρτηθεί τότε το EventName είναι NULL.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project82.
Το πρόγραμμα έχει την φόρμα FMain την οποία απλά φορτώνει κατά την εκκίνηση.

```
1   ' Gambas class file
2
3   Public Sub _new()
4
5   Print "I am in _new"
6
7   End
8
9    Static Public Sub _init()
10
11  Print "I am in _init"
12
13  End
14
15  Static Public Sub _exit()
16
17  Print "I am in _exit"
18
19  End
20
21  Public Sub _free()
22
23  Print "I am in _free"
24
25  End
26
27  Public Sub Form_Open()
28
29  Print "I am in Form_open"
30
31  End
```

Έξοδος:

```
I am in _init
I am in _new
I am in Form_open
I am in _free
I am in _exit
```

Ανάλυση:
Κάθε ειδική μέθοδος τυπώνει ένα μήνυμα στην προκαθορισμένη έξοδο.
Έτσι βλέπουμε ότι η σειρά κλήσης των μεθόδων είναι:
_init, κατά την αρχικοποίηση της κλάσης από τον διερμηνευτή.
_new, κατά την δημιουργία του αντικειμένου.

_free, κατά την απελευθέρωση του αντικειμένου από την μνήμη.
_exit, κατά την απελευθέρωση της κλάσης από την μνήμη.

Κεφάλαιο 3ο Συμβάντα και Παρατηρητές (Events & Observers)

Συμβάντα (Events)

Τα συμβάντα είναι σήματα που στέλνωνται από ένα αντικείμενο όταν κάτι συμβαίνει σε αυτό.

Εάν ένα αντικείμενο εγείρει κάποιο συμβάν τότε μια αναφορά αυτού θα δημιουργηθεί στον παρατηρητή του (**observer)** ή στο γονικό αντικείμενο (**parent object**)

Ο παρατηρητής (**observer**) είναι ένα άλλο αντικείμενο το οποίο υλοποιεί διαχειριστές συμβάντων (**event handlers**). Ο διαχειριστής συμβάντος είναι μια δημόσια μέθοδος που καλείται κάθε φορά που ένα συμβάν εγείρεται.

Εξ' ορισμού ο παρατηρητής είναι το τρέχον αντικείμενο όπου το νέο δημιουργημένο αντικείμενο έχει δηλωθεί.

Για να εγείρετε κάποιο συμβάν ένα αντικείμενο πρέπει να έχει ένα όνομα συμβάντος (**event name**). Το όνομα του συμβάντος ανατίθεται στο αντικείμενο κατά την δημιουργία του όταν χρησιμοποιείται η λέξη NEW και η λέξη AS και αποτελεί το πρόθεμα όλων των μεθόδων του διαχειριστή συμβάντος

Παράδειγμα:

```
1  DIM hButton AS Button
2
3  hButton = NEW Button(ME) AS "ButtonEventName"
```

Το παράδειγμα αυτό δημιουργεί ένα κουμπί (Button) το οποίο εγείρει συμβάντα.

Εάν δεν οριστεί όνομα συμβάντος τότε το αντικείμενο δεν εγείρει κάποιο συμβάν.

Κλειδώνοντας αντικείμενα (Locking Objects)

Ένα αντικείμενο μπορεί να κλειδώσει και να σταματήσει να εγείρει συμβάντα αλλά και να ξεκλειδώσει για να συνεχίσει να εγείρει συμβάντα ξανά. Δείτε στην wiki-εγκυκλοπαίδεια για περισσότερα στις μεθόδους Object.Lock και Object.Unlock.

Μερικά συμβάντα μπορεί να ακυρωθούν από τον διαχειριστή συμβάντων με την εντολή STOP EVENT.

Ένα αντικείμενο κλειδώνει αυτόματα κατά την διάρκεια της εκτέλεσης της μεθόδου δημιουργίας του, έτσι δεν μπορεί να λάβει κάποιο συμβάν.

Παρατηρητές (Observers)

Οι παρατηρητές (**Observers**) είναι αντικείμενα τα οποία επιτρέπουν να διαχειρίζεστε συμβάντα που έχουν εγερθεί από αντικείμενα.

Μπορείτε να διαχειρίζεστε συμβάντα πριν εγερθούν ή μετά.

Για κάθε συμβάν που παρατηρείται ο παρατηρητής εγείρει ένα συμβάν με το ίδιο όνομα και τις ίδιες παραμέτρους.

Χρησιμοποιώντας την εντολή STOP EVENT εντός ενός παρατηρητή ακυρώνεται το αρχικό συμβάν.

Εάν ακυρώσετε ένα συμβάν στον παρατηρητή το συμβάν αυτό δεν θα εγερθεί στο αρχικό αντικείμενο.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project83. Το πρόγραμμα αποτελείται από μια φόρμα την FMain στην οποία κατά το άνοιγμα δημιουργούμε ένα κουμπί το οποίο παρακολουθείται από έναν παρατηρητή.

```
1   ' Gambas class file
2
3   Private $hButton As Button
4   Private $hObserver As Observer
5
6   Public Sub Form_Open()
7     $hButton = New Button(Me) As "Button"
8     $hButton.X = 10
9     $hButton.Y = 50
10    $hButton.Width = 100
11    $hButton.Height = 50
12    $hButton.Text = "Click me"
13
14    $hObserver = New Observer($hButton) As "Observer"
15  End
16
```

```
17 Public Sub Observer_Click()
18   Debug "The button has been clicked. I cancel the event!"
19   Stop Event
20 End
21
22 Public Sub Button_Click()
23   Debug "You should not see me."
24 End
```

Έξοδος:

```
FMain.Observer_Click.18: The button has been clicked. I cancel the event!
```

Ανάλυση:
Στην γραμμή 3 δηλώνεται η μεταβλητή που θα δηλωθεί το αντικείμενο Button.
Στην γραμμή 4 δηλώνεται η μεταβλητή που θα δηλωθεί ο παρατηρητής.
Στην γραμμή 7 όταν φορτώνεται η φόρμα δημιουργούμε το νέο αντικείμενο κουμπί.
Στις γραμμές 8-12 θέτουμε μερικές ιδιότητες του νέου αντικειμένου όπως μήκος, πλάτος, θέση X,Y στην φόρμα πάνω και κείμενο.
Στην γραμμή 14 δημιουργούμε ένα νέο αντικείμενο τύπου παρατηρητή το οποίο θα παρατηρεί το αντικείμενο κουμπί.
Το αντικείμενο `$hObserver` κατά την δημιουργία του έχει και τον διαχειριστή συμβάντων με το όνομα Observer.
Έτσι όταν ένα συμβάν εγείρεται στο κουμπί όπως για παράδειγμα ένα κλικ ο διαχειριστής συμβάντων αναλαμβάνει την διαχείριση του συμβάντος παρακάμπτοντας την ρουτίνα συμβάντος με το ίδιο όνομα συμβάντος. Με άλλα λόγια για το συμβάν Click() όταν αυτό εγερθεί το πρόγραμμα θα καλέσει την ρουτίνα του παρατηρητή `Observer_Click()`στην γραμμή 17 και όχι την ρουτίνα `Button_Click()` της γραμμής 22. Έτσι το πρόγραμμα δεν θα πρέπει να φτάσει την γραμμή 17 ποτέ.

Κεφάλαιο 4ο Κληρονομικότητα (Inheritance)

Μέχρι στιγμής είδαμε τι είναι μια κλάση, ότι η Gambas παρέχει ένα πλήθος από κλάσεις και ότι μπορούμε να δημιουργήσουμε και εμείς τις δικές μας κλάσεις.

Αν για παράδειγμα δημιουργήσουμε μια κλάση 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 κλάσεων που παίρνουν τα χαρακτηριστικά άλλων κλάσεων έρχεται να λύσει η κληρονομικότητα.

Η κληρονομικότητα των κλάσεων είναι ένας τρόπος για μια κλάση να γίνει η ειδική έκδοση μιας άλλης κλάσης.

Όταν μια κλάση κληρονομεί μια άλλη, η νέα κλάση λέμε ότι απορρέει από την πρώτη και ότι η πρώτη κλάση είναι η κλάση βάσης της νέας κλάσης.
Η νέα κλάση θα έχει όλα τα μέλη της κλάσης βάσης όπως τις ιδιότητες, μεθόδους, μεταβλητές, σταθερές και συμβάντα και ότι επιπλέον εμείς προσθέσουμε.

Για παράδειγμα αν έχουμε μια κλάση 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.

Μια κλάση κληρονομεί μια άλλη με την εντολή `INHERITS`.
Εντός της νέας κλάσης μπορείτε να προσπελάζεται τα μέλη που έχουν κληρονομηθεί με την λέξη ME.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project84. Το παράδειγμα αποτελείται από τρία αρχεία.
Την κλάση cCar, την κλάση που απορρέει από την cCar την cStarCar και το module Main ως κλάση εκκίνησης.

Ο κώδικας του module Main είναι:

```
1   ' Gambas module file
2
3   Public Sub Main()
4
5     Dim hCar As CStarCar
6     hCar = New CStarCar
7
8     hCar.sColor = "Red"
9     hCar.iBrakes = 2
10    hCar.iDoors = 4
11    hCar.iSeats = 5
12    hCar.Run
13    hCar.Stop
14    hCar.TurnOnAlarm
15    hCar.TurnOffAlarm
16    hCar.Run
17    hCar.Stop
18    Print hCar.sColor
19    hCar.SetColor
20    Print hCar.sColor
21
22  End
```

Ο κώδικας της κλάσης cCar είναι:

```
1   ' Gambas class file
2
3   Public sColor As String
4   Public iDoors As Integer
5   Public iSeats As Integer
6
7   Public Sub Run()
8
9     Print "Now car is running"
10
11  End
12
13  Public Sub Stop()
14
```

```
15    Print "Now car is stopped"
16
17  End
```

Ο κώδικας της κλάσης cStarCar είναι:

```
1   ' Gambas class file
2
3   Inherits CCar
4
5   Public iBrakes As Integer
6
7   Public Sub TurnOnAlarm()
8
9     Print "Alarm is on"
10
11  End
12
13  Public Sub TurnOffAlarm()
14
15    Print "Alarm is off"
16
17  End
18
19  Public Sub SetColor()
20
21    Me.sColor = "Green"
22
23  End
```

Έξοδος:

```
Now car is running
Now car is stopped
Alarm is on
Alarm is off
Now car is running
Now car is stopped
Red
Green
```

Ανάλυση:
Στον κώδικα της κλάσης cCar, στις γραμμές 3-5 δηλώνουμε 3 δημόσιες μεταβλητές.
Στην συνέχεια στις γραμμές 7-17 δηλώνουμε δύο μεθόδους οι οποίες τυπώνουν ένα μήνυμα στην προκαθορισμένη έξοδο.

Στον κώδικα της κλάσης cStarCar, στην γραμμή 3 η κλάση κληρονομεί την κλάση cCar.
Στην γραμμή 5 δηλώνεται μια δημόσια μεταβλητή και στις γραμμές 5-23 τρεις μέθοδοι.

Στον κώδικα του module Main στις γραμμές 5-6 δημιουργούμε ένα αντικείμενο τύπου cStarCar.
Στις γραμμές 8-11 θέτουμε τιμές στις μεταβλητές του αντικειμένου.
Στις γραμμές 12-17 καλούμε τις μεθόδους του αντικειμένου.
Στην γραμμή 18 τυπώνουμε στην προκαθορισμένη έξοδο την τιμή της μεταβλητής sColor.
Στην γραμμή 19 καλούμε την μέθοδο SetColor που εκχωρεί νέα τιμή στην μεταβλητή sColor.
Τέλος στην γραμμή 20 τυπώνουμε στην προκαθορισμένη έξοδο την τιμή της μεταβλητής sColor και πάλι.

Ποιες κλάσεις μπορούν να είναι η βάση για άλλες κλάσεις;

Μπορείτε να κληρονομείτε οποιαδήποτε κλάση ακόμη και τις εγγενείς κλάσεις.
Για παράδειγμα μπορείτε να δημιουργήσετε μια custom MyListBox κλάση η οποία κληρονομεί την κλάση ListBox αλλά επιτρέπει την χρήση ενός tag για κάθε στοιχείο της λίστας.

Ωστόσο δεν μπορείτε να κάνετε χρήση της εντολής INHERITS μέσα σε μια φόρμα, διότι οι φόρμες κληρονομούν ήδη την κλάση Form.

Μια κλάση μπορεί να κληρονομεί μια άλλη και αυτό να συνεχίζεται μέχρι 16 φορές. Πάνω από 16 κληρονομήσεις δεν επιτρέπονται διότι αυτό έχει δηλωθεί ως μια σταθερά στον διερμηνευτή της Gambas.

Εικονική διανομή (Virtual dispatching)

Όταν καλείται μια μέθοδος ή προσπελάζεται μια ιδιότητα από την αναφορά ενός αντικειμένου, η Gambas πάντα χρησιμοποιεί την εικονική διανομή των δεδομένων αυτών.

Αυτό σημαίνει ότι χρησιμοποιείται η πραγματική κλάση του αντικειμένου και όχι ο τύπος της μεταβλητής στο οποίο αναφέρεται το αντικείμενο, όπως γινόταν στην Gambas 1.0.

Κληρονομικότητα και μέθοδος δημιουργίας (Inheritance and constructor)

Η σειρά κλήσης των μεθόδων δημιουργίας

Μέχρι στιγμής είδαμε ότι οι κλάσεις έχουν δύο ειδικές μεθόδους για την δημιουργία και την καταστροφή των αντικειμένων που δημιουργούν. Όταν δημιουργείται ένα αντικείμενο από μια κλάση καλείται η μέθοδος δημιουργίας _new() και όταν ένα αντικείμενο καταστρέφεται καλείται η μέθοδος καταστροφής _free().
Όταν μια κλάση κληρονομεί μια άλλη και η δεύτερη μια άλλη κτλ τότε το αντικείμενο που θα δημιουργηθεί από την τελευταία θα καλέσει με την σειρά τις μεθόδους δημιουργίας από την αρχική κλάση μέχρι την τελευταία.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project85. Το παράδειγμα αποτελείται από τέσσερα αρχεία.
Την κλάση cClassA, την κλάση cClassB, την κλάση cClassC και το module Main ως κλάση εκκίνησης. Η κλάση cClassC κληρονομεί την cClassB και η cClassB την cClassA.

Ο κώδικας του module Main είναι:

```
1   ' Gambas module file
2
3   Public Sub Main()
4
5     Dim hObject As CClassC
6
7     hObject = New CClassC
8
9     hObject.a = 16
10    hObject.b = 21
11    hObject.c = 321
12
13    Print hObject.a
14
15  End
```

Ο κώδικας της κλάσης cClassA είναι:

```
1   ' Gambas class file
2
3   Public a As Integer
4
5   Public Sub _new()
6
7     Print "I am in Class A constructor"
8
9   End
```

Ο κώδικας της κλάσης cClassB είναι:

```
1   ' Gambas class file
2
3   Inherits CClassA
4
5   Public b As Integer
6
7   Public Sub _new()
8
9     Print "I am in Class B constructor"
10
11  End
```

Ο κώδικας της κλάσης cClassC είναι:

```
1   ' Gambas class file
2
3   Inherits CClassB
4
5   Public c As Integer
6
7   Public Sub _new()
8
9     Print "I am in Class C constructor"
10
11  End
```

Έξοδος:

```
I am in Class A constructor
I am in Class B constructor
I am in Class C constructor
16
```

Ανάλυση:
Κάθε κλάση έχει μια μεταβλητή ακεραίου. Η cClassA έχει την a, η cClassB την b, η cClassC την c.
Επίσης κάθε κλάση τυπώνει ένα μήνυμα στην προκαθορισμένη έξοδο από την μέθοδο δημιουργίας της, όταν δηλαδή δημιουργείται ένα αντικείμενο από την κλάση.

Η κλάση cClassB κληρονομεί την cClassA και η κλάση cClassC την κλάση cClassB.
Δηλαδή με βάση την κλάση cClassA έχουμε την κληρονομικότητα του παρακάτω σχήματος.

cClassA --> cClassB --> cClassC

Στο module Main στην γραμμή 5 δηλώνουμε μια μεταβλητή του τύπου cClassC.
Στην γραμμή 7 δημιουργούμε ένα νέο αντικείμενο του τύπου cClassC.
Στις γραμμές 9-11 εκχωρούμε τιμές στις μεταβλητές τους.
Και στην γραμμή 13 τυπώνουμε στην προκαθορισμένη έξοδο την τιμή μιας μεταβλητής του αντικειμένου.

Βλέπουμε ότι όταν δημιουργήσαμε το αντικείμενο μας στην προκαθορισμένη έξοδο τυπώθηκαν τα μηνύματα από τις μεθόδους δημιουργίας κάθε κλάσης που η cClassC κληρονομεί.

Η σειρά προτεραιότητας κλήσης των μεθόδων αυτών βλέπουμε ότι είναι από την πρώτη κλάση προς την τελευταία. Έτσι πρώτα καλείται η _new() της κλάσης A μετά η _new() της κλάσης B και τέλος η _new() της κλάσης C.

Η σειρά διοχέτευσης των παραμέτρων στην μέθοδο δημιουργίας

Κατά την δημιουργία του αντικειμένου από μια κλάση μπορούμε να διοχετεύσουμε παραμέτρους στην μέθοδο δημιουργίας _new().
Όταν όμως μια κλάση κληρονομεί άλλες πρέπει να γνωρίζουμε την σειρά με την οποία οι παράμετροι αυτοί διοχετεύονται.

Στην Gambas3 η σειρά διοχέτευσης παραμέτρων ακολουθεί την σειρά κλήσης των μεθόδων δημιουργίας. Έτσι αν η πρώτη κλάση βάση δέχεται την παράμετρο A η δεύτερη την παράμετρο B και η τρίτη την παράμετρο C τότε η τελική μέθοδος δημιουργίας θα δέχεται τις παραμέτρους αυτές με αυτήν την σειρά.
Δλδ η σειρά διοχέτευσης των παραμέτρων θα είναι _new(A, B, C).

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project86. Το παράδειγμα αποτελείται από τέσσερα αρχεία. Την κλάση cClassA, την κλάση cClassB, την κλάση cClassC και το module Main ως κλάση εκκίνησης. Η κλάση cClassC κληρονομεί την cClassB και η cClassB την cClassA. Οι μέθοδοι δημιουργίας εδώ δέχονται από μια παράμετρο.

Ο κώδικας του module Main είναι:

```
1   ' Gambas module file
2
3   Public Sub Main()
4
5     Dim hObject As CClassC
6
7     hObject = New CClassC("A", "B", "C", 15)
8
9     hObject.a = 16
10    hObject.b = 21
11    hObject.c = 321
12
13    Print hObject.a
14
15  End
```

Ο κώδικας της κλάσης cClassA είναι:

```
1   ' Gambas class file
2
3   Public a As Integer
4
5   Public Sub _new(name As String)
6
7     Print "I am in Class " & name & " constructor"
8
9   End
```

Ο κώδικας της κλάσης cClassB είναι:

```
1
```

Ο κώδικας της κλάσης cClassC είναι:

```
1   ' Gambas class file
2
3   Inherits CClassB
4
```

```
5   Public c As Integer
6
7   Public Sub _new(name As String)
8
9     Print "I am in Class " & name & " constructor"
10
11  End
```

Έξοδος:

```
I am in Class A constructor
I am in Class B constructor
My age is 15
I am in Class C constructor
16
```

Ανάλυση:
Το πρόγραμμα είναι το ίδιο με αυτό του Project85. Μόνο που σε κάθε κλάση η μέθοδος δημιουργίας της δέχεται μια παράμετρο και τυπώνει την παράμετρο αυτή σαν όνομα στο μήνυμα στην προκαθορισμένη έξοδο.

Η κλάση B δέχεται επιπλέον μια προαιρετική παράμετρο και τυπώνει ένα μήνυμα με την τιμή αυτής στην προκαθορισμένη έξοδο.

Στο module Main στην γραμμή 7 δημιουργεί το νέο αντικείμενο και διοχετεύει τις παραμέτρους A, B, C, 15 με την σειρά αυτή διότι η πρώτη παράμετρος διοχετεύεται στην μέθοδο δημιουργίας της κλάσης A η δεύτερη παράμετρος στην κλάση B, η τρίτη στην κλάση C και στην συνέχεια με την ίδια σειρά από τις γηραιότερες κλάσεις προς τις νεότερες διοχετεύονται οι προαιρετικές παράμετροι.

Το παράδειγμα αυτό δείχνει ότι η σειρά διοχέτευσης ακολουθεί την σειρά παλαιότητας κάθε κλάσης. Πρώτα δλδ καλούνται οι μέθοδοι δημιουργίας των παλαιών κλάσεων και μετά των νεότερων έτσι και οι παράμετροι διοχετεύονται πρώτα στις παλαιότερες κλάσεις και μετά στις νεότερες. Αν υπάρχουν προαιρετικές παράμετροι αυτές ακολουθούν την ίδια σειρά προτεραιότητας αμέσως μετά από την διοχέτευση των απαιτούμενων παραμέτρων.

Κυκλική κληρονομικότητα (Circular Inheritance)

Όταν μια κλάση κληρονομεί μια άλλη η δεύτερη κλάση δεν μπορεί να κληρονομήσει την κλάση βάση. Για παράδειγμα αν έχουμε την κλάση cClassB και κληρονομεί την cClassC η κλάση cClassC δεν μπορεί να κληρονομήσει την cClassB.

Αν γράψουμε ένα τέτοιο κώδικα δημιουργούμε μια κυκλική κληρονομικότητα με αποτέλεσμα το πρόγραμμα μας να μην μπορεί να διαβαστεί από τον διερμηνευτή.

Υπέρβαση συμβόλων (Symbol Overriding)

Τα σύμβολα όπως είδαμε είναι τα μέλη μιας κλάσης, είναι δλδ οι ιδιότητες, μεταβλητές, σταθερές, μέθοδοι και συμβάντα.

Κατά την κληρονομικότητα μιας κλάσης η κλάση που κληρονομεί την κλάση βάση μπορεί και κάνει χρήση τα σύμβολα της πρώτης με τα ονόματα που παρέχονται από την κλάση βάση.

Υπάρχει όμως η δυνατότητα η κλάση που κληρονομεί την κλάση βάση να έχει σύμβολα με ονόματα ίδια με αυτά που κληρονομεί. Σε αυτήν την περίπτωση λέμε ότι γίνεται υπέρβαση συμβόλων στην κλάση. Τα σύμβολα που υπερβαίνουν τα σύμβολα της κλάσης βάσης πρέπει να έχουν τον ίδιο τύπο δεδομένων.

Οι κανόνες που ισχύουν είναι:

- Ένα δυναμικό σύμβολο πρέπει να υπερβαίνει ένα δυναμικό σύμβολο.
- Ένα στατικό σύμβολο πρέπει να υπερβαίνει ένα στατικό σύμβολο.
- Μια μέθοδος πρέπει να υπερβαίνει μια μέθοδο.
- Μια ιδιότητα πρέπει να υπερβαίνει μια ιδιότητα.
- Μια ιδιότητα που είναι για ανάγνωση ή εγγραφή μπορεί να υπερβαίνει μια ιδιότητα που είναι μόνο για ανάγνωση.
- Μια σταθερά ή ιδιότητα μπορεί να υπερβαίνει μια σταθερά.

Μια μέθοδος αποτελείται από το όνομα και την λίστα των παραμέτρων αυτό ονομάζεται υπογραφή της συνάρτησης ή πρωτότυπο.
Όταν δημιουργούμε μια μέθοδο στην νέα κλάση με το ίδιο όνομα μιας μεθόδου από μια κλάση βάση πολλές φορές χρησιμοποιούμε την ίδια υπογραφή της μεθόδου, δλδ το όνομα και το πλήθος των παραμέτρων.

Στην Gambas3 η υπογραφή της μεθόδου της κλάσης που κληρονομεί δεν ελέγχεται να είναι η ίδια με την υπογραφή της κλάσης βάσης.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project87. Το παράδειγμα αποτελείται από τρία αρχεία.
Την κλάση cClassA, την κλάση cClassB, και το module Main ως κλάση εκκίνησης. Η κλάση cClassB κληρονομεί την cClassA.

Ο κώδικας του module Main είναι:

```
1   ' Gambas module file
2
3   Public Sub Main()
4
5     Dim hObject As CClassB
6
7     hObject = New CClassB
8
9     hObject.iAge = 23
10    hObject.PrintAge(24)
11
12  End
```

Ο κώδικας της κλάσης cClassA είναι:

```
1   ' Gambas class file
2
3   Public iAge As Integer
4
5   Public Sub PrintAge()
6
7     Print "My age is " & iAge
8
9   End
10
```

Ο κώδικας της κλάσης cClassB είναι:

```
1   ' Gambas class file
2
3   Inherits CClassA
4
5   Public iAge As Integer
6
7   Public Sub PrintAge(iNewAge As Integer)
8
9     Print "Now my age is " & iNewAge
10
11  End
```

Έξοδος:

```
Now my age is 24
```

Ανάλυση:
Η κλάση cClassA έχει δύο σύμβολα. Μια δημόσια μεταβλητή ακεραίου την iAge που δηλώνεται στην γραμμή 3 και την μέθοδο PrintAge() που δηλώνεται στις γραμμές 5-9.

Η κλάση cClassB κληρονομεί τη κλάση cClassA αλλά δημιουργεί υπέρβαση για τα σύμβολα iAge και PrintAge() δημιουργώντας αυτά με το ίδιο όνομα αλλά με διαφορετική υλοποίηση. Η μέθοδος PrintAge() επίσης δεν έχει την ίδια υπογραφή με την μέθοδο PrintAge() της κλάσης cClassA. Στην κλάση cClassB η PrintAge() δέχεται μια παράμετρο ακεραίου την iNewAge.

Στο module Main δημιουργούμε το αντικείμενο hObject στις γραμμές 5,7 και στην συνέχεια, γραμμή 9, εκχωρούμε μια τιμή στην μεταβλητή του iAge η οποία εδώ είναι η μεταβλητή υπέρβασης της κλάσης cClassB. Στην γραμμή 10 καλούμε την μέθοδο PrintAge() αφού διοχετεύσουμε την τιμή 24.
Έτσι το αντικείμενο hObject τυπώνει στην προκαθορισμένη έξοδο το μήνυμα “Now my age is 24”.

Κεφάλαιο 5ο Πίνακες συμβόλων

Τα εξαρτήματα στην Gambas είναι εξωτερικές βιβλιοθήκες γραμμένες στην C, C++ ή στην Gambas οι οποίες προσθέτουν νέες συναρτήσεις ή και κλάσεις στον διερμηνευτή της Gambas.

Οι κλάσεις είναι ομαδοποιημένες σύμφωνα με το εξάρτημα από το οποίο προέρχονται.

Το προκαθορισμένο εσωτερικό εξάρτημα

Ο διερμηνευτής περιέχει ένα εσωτερικό εξάρτημα με το όνομα gb το οποίο δηλώνει όλες τις προκαθορισμένες κλάσεις της γλώσσας.

Πίνακες συμβόλων

Κάθε εξάρτημα έχει τον δικό του προσωπικό πίνακα συμβόλων των κλάσεων έτσι ώστε τα ονόματα των κλάσεων να μην συγχέονται.
Επίσης υπάρχει ένα γενικός πίνακας συμβόλων όπου όλες οι κλάσεις εξάγονται από τα εξαρτήματα και όλες οι κλάσεις εξάγονται από το τρέχον έργο.
Εάν υπάρχει μια διένεξη στον γενικό πίνακα συμβόλων η τελευταία κλάση που έχει φορτωθεί κληρονομεί και υπερβαίνει οποιαδήποτε άλλη με το ίδιο όνομα έτσι η τελευταία κλάση αποτελεί μια επέκταση όλων των άλλων κλάσεων με το ίδιο όνομα.
Αυτό το τελευταίο χαρακτηριστικό μπορεί να χρησιμοποιηθεί για:

- Επέκταση μια ήδη δηλωμένης κλάσης με το να προστεθούν νέες μέθοδοι ή ιδιότητες σε αυτή. Για παράδειγμα η κλάση Application του εξαρτήματος gb.qt υλοποιεί ξανά την κλάση Application του εξαρτήματος gb.
- Υπέρβαση των μεθόδων μιας ήδη δηλωμένης κλάσης. Για παράδειγμα το εξάρτημα gb.form.dialog υπερβαίνει τις περισσότερες στατικές μεθόδους της κλάσης Dialog.

Ο πίνακας συμβόλων του έργου

Κάθε έργο (project) έχει τα δικά του ιδιωτικά σύμβολα, όπως κάθε εξάρτημα, και μπορεί να εξάγει οποιαδήποτε από τις κλάσεις του στο γενικό πίνακα συμβόλων χρησιμοποιώντας την λέξη EXPORT.
Οι κλάσεις του έργου φορτώνονται μετά από την φόρτωση όλων των εξαρτημάτων. Έτσι οι κλάσεις του έργου μπορούν να υπερβαίνουν τις κλάσεις των εξαρτημάτων.

Κεφάλαιο 6ο – Διαχείριση κλάσεων και αντικειμένων

Η Gambas δίνει μια σειρά από εντολές για την διαχείριση των κλάσεων και των αντικειμένων. Οι εντολές αυτές είναι:

CLASS	Δηλώνει την χρήση μιας κλάσης
Class	Η κλάση με το όνομα Class δίνει αρκετές στατικές μεθόδους για εξαγωγή πληροφοριών σχετικά με τις κλάσεις.
CREATE_PRIVATE	Δηλώνει μια κλάση ως μη ικανή να δημιουργηθεί.
CREATE STATIC	Δηλώνει μια κλάση ως αυτόματα δημιουργήσιμη.
EXPORT	Δηλώνει μια κλάση ως εξαγώμενη.
INHERITS	Δηλώνει ότι η κλάση κληρονομεί μια άλλη.
IS	Επιστρέφει αλήθεια αν ένα αντικείμενο είναι υλοποίηση μιας κλάσης.
LAST	Επιστρέφει μια αναφορά στο αντικείμενο το οποίο έχει εγείρει το τελευταίο συμβάν.
ME	Επιστρέφει μια αναφορά στο τρέχον αντικείμενο.
NEW	Υλοποιεί ένα αντικείμενο.
Object	Η κλάση Object δίνει αρκετές μεθόδους σχετικές με την διαχείριση αντικειμένων.
SUPER	Επιστρέφει μια αναφορά στο τρέχον αντικείμενο το οποίο θα κάνει χρήση δημόσια σύμβολα μιας κληρονομούμενης κλάσης.

CLASS

Σύνταξη:

```
CLASS Identifier
```

Δηλώνει στον compiler ότι ο προσδιορισμός Identifier μπορεί να χρησιμοποιηθεί ως όνομα κλάσης.
Εξ' ορισμού όλες οι κλάσεις του τρέχοντος έργου και όλες οι εξαγόμενες κλάσεις από τα εξαρτήματα που χρησιμοποιούνται αυτόματα δηλώνονται.

Αυτή η λέξη μπορεί να είναι χρήσιμη εάν θέλετε να γράψετε προαιρετικό κώδικα σε ένα εξάρτημα το οποίο χρειάζεται μια κλάση από εξάρτημα το οποίο δεν φορτώνεται απαραίτητα από το έργο απευθείας.

Class

Η κλάση με το όνομα Class αποτελεί μέλος του εξαρτήματος gb.
Η κλάση αυτή παρέχει μια κλάση που μπορεί να χρησιμοποιηθεί από τον διερμηνευτή.

Η κλάση αυτή μπορεί να είναι εγγενής, για παράδειγμα να υλοποιηθεί σε ένα εξάρτημα ή μια κλάση γραμμένη στην Gambas.

Η κλάση δεν μπορεί να δημιουργηθεί.
Η κλάση λειτουργεί σαν ένας πίνακας μόνο για ανάγνωση.

Στατικές μέθοδοι	**Μεταβλητές**	**Ιδιότητες**
Load	Constant Event Method Property Variable	Component Count Hidden Instance Name Native Parent Symbols

Η περιγραφή των μεθόδων, μεταβλητών και ιδιοτήτων γίνεται στην wikipedia της τεκμηρίωσης της Gambas.

CREATE PRIVATE

Σύνταξη:

```
CREATE PRIVATE
```

Οι λέξεις αυτές όταν τοποθετηθούν μόνες τους στην αρχή μιας κλάσης λένε στον διερμηνευτή ότι η κλάση δεν μπορεί να υλοποιηθεί.
Αυτό είναι χρήσιμο για κλάσεις οι οποίες είναι σχεδιασμένες για να είναι κλάσεις βάσης για άλλες κλάσεις.

CREATE STATIC

Σύνταξη:

```
CREATE STATIC
```

Οι λέξεις αυτές όταν τοποθετηθούν μόνες τους στην αρχή μιας κλάσης λένε στον διερμηνευτή ότι όταν εφαρμόζει μη στατικά σύμβολα στην κλάση μια αυτόματη υλοποίηση από αυτά να δημιουργείται απευθείας.

Αυτό το χαρακτηριστικό επιτρέπει την υλοποίηση αντικειμενοστραφούς προγραμματισμού με μονήρη μοτίβα (singleton pattern).

Εάν χρησιμοποιείται αυτό το χαρακτηριστικό η μέθοδος δημιουργίας της κλάσης (η ειδική μέθοδος _new) θα κληθεί χωρίς παραμέτρους
Το χαρακτηριστικό αυτό χρησιμοποιείται στις κλάσεις Form και Settings.

EXPORT

Σύνταξη:

```
EXPORT [ OPTIONAL ]
```

Η λέξη αυτή όταν τοποθετηθεί μόνη της στην αρχή μια κλάσης δηλώνει ότι η κλάση εξάγεται. Αυτό σημαίνει ότι η κλάση θα είναι ορατή απέξω όταν δημιουργείτε ένα εξάρτημα.
Εάν δηλωθεί η λέξη **OPTIONAL** τότε η κλάση εξάγεται μόνο εάν δεν έχει φορτωθεί άλλη κλάση με το ίδιο όνομα από κάποιο εξάρτημα.

Παράδειγμα:

```
' Gambas class file
'
' This is an implementation of a new ListBox control put in a
component

EXPORT
INHERITS ListBox

...
```

INHERITS

Σύνταξη:

```
INHERITS ParentClass
```

Η λέξη αυτή όταν δηλωθεί στην αρχή της κλάσης λέει στον διερμηνευτή ότι η τρέχουσα κλάση κληρονομεί την βάση κλάση ParentClass.
Πρέπει να χρησιμοποιείτε την λέξη ME για να προσπελάζετε τα μέλη μιας κληρονομούμενης κλάσης όταν γράφετε κώδικα στην κλάση που κληρονομεί μια άλλη κλάση.

IS

Σύνταξη:

```
Result = Expression IS Class
```

Επιστρέφει TRUE εάν το αντικείμενο είναι υλοποίηση μιας κλάσης ή απόγονου μιας κλάσης

- Result είναι η τιμή που επιστρέφεται.
- Expression είναι οποιαδήποτε έκφραση η οποία επιστρέφει μια αναφορά σε αντικείμενο.
- Class είναι το όνομα της κλάσης

Παραδείγματα:

```
PRINT ["Gambas", "rules!"] IS String[]
```

Έξοδος:

```
True
```

```
DIM myTextBox AS NEW TextBox(ME)

PRINT myTextBox IS Control
```

Έξοδος:

```
True
```

```
PRINT ["Gambas", "rules!"] IS Collection
```

Έξοδος:

```
False
```

LAST

Σύνταξη:

```
LAST
```

Επιστρέφει μια αναφορά στο αντικείμενο που έχει εγείρει τελευταίο το πιο πρόσφατο συμβάν.

Παράδειγμα:

```
DIM hButton[3] AS Button

hButton[0] = NEW Button(ME) AS "MyButtons"
hButton[0].Text = "Red";

hButton[1] = NEW Button(ME) AS "MyButtons"
hButton[1].Text = "Green"

hButton[2] = NEW Button(ME) AS "MyButtons"
hButton[2].Text = "Blue"

...

PUBLIC SUB MyButtons_Click()

  PRINT LAST.Text

END
```

ME

Σύνταξη:

	ME

Η λέξη επιστρέφει μια αναφορά στο τρέχον αντικείμενο.
Η λέξη **ME** είναι απαραίτητη όταν θέλετε να καλέσετε μια μέθοδο ή ιδιότητα ή μεταβλητή που έχει κληρονομηθεί.

Παραδείγματα:

```
' Gambas form

PUBLIC SUB SetTitle(Title AS String)

ME.Text = Title

END

PUBLIC SUB MoveRight(Step AS Integer)

ME.Move(ME.X + Step, ME.Y)

END
```

Το επόμενο παράδειγμα δείχνει ότι η λέξη ME επιστρέφει την φόρμα ενώ ή λέξη LAST επιστρέφει το κουμπί που έχει εγείρει ένα συμβάν.

```
PUBLIC SUB Button5_Click()
PRINT ME;; LAST
END
```

Έξοδος:

```
(Form1 0x80d8090) (Button 0x80fa060)
```

NEW

Σύνταξη:

```
Object = NEW Class [ ( Constructor parameters... ) ] [ AS Name ]
```

Η λέξη NEW υλοποιεί ένα αντικείμενο από μια κλάση με το όνομα Class.
Εάν δηλωθεί το όνομα Name το νέο αντικείμενο θα μπορεί να εγείρει συμβάντα με το να καλέι μια δημόσια μέθοδο ή συνάρτηση στο πατρικό του αντικείμενο

- Αυτό το πατρικό αντικείμενο, ή ο προκαθορισμένος παρατηρητής είναι το αντικείμενο ή η κλάση όπου το νέο αντικείμενο υλοποιείται.
- Το όνομα του διαχειριστή συμβάντων είναι το όνομα του αντικειμένου ακολουθούμενο από μια κάτω παύλα (_) και το όνομα του συμβάντος.

Εάν δεν δηλωθεί το μέρος με το όνομα Name, το αντικείμενο δεν θα εγείρει συμβάντα.

Η λέξη **NEW** δεν είναι τελεστής. Μπορεί να χρησιμοποιηθεί μόνο για νέες εκχωρήσεις.

Η λέξη NEW δηλώνει νέα αντικείμενα την ώρα του compile.
Για να δηλώσετε νέα αντικείμενα την ώρα εκτέλεσης (run time) μπορείτε να χρησιμοποιήσετε την συνάρτηση Object.New από την κλάση Object.

Δυναμική υλοποίηση (Dynamic instanciation)

Σύνταξη:

```
Object = NEW ( ClassName [ , Constructor parameters... ] ) [ AS
Name ]
```

Η δεύτερη σύνταξη επιτρέπει να οριστεί η κλάση δυναμικά ως ένα αλφαριθμητικό string.

- ClassName είναι ένα αλφαριθμητικό που δηλώνει το όνομα της κλάσης
- Constructor parameters είναι οι παράμετροι της μεθόδου δημιουργίας

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project88

```
1   ' Gambas module file
2
3   Public Sub Main()
4
5   ' This creates an 3x3 array of float.
6
7   Dim MyArray1 As New Float[3, 3]
8   Dim MyArray2 As Object
9   Dim MyArray3 As Object
10
11  Dim MyClassName As String
12
13  MyArray2 = Object.New("Float[]", [3, 3])
14
15  MyClassName = "Float[]"
16  MyArray3 = New (MyClassName, 3, 3)
17
18  Print MyArray1
19  Print MyArray2
20  Print MyArray3
21
22  End
```

Έξοδος:

```
(Float[] 0x102c1f8)
(Float[] 0x102c448)
(Float[] 0x102c388)
```

Ανάλυση:
Στην γραμμή 7 υλοποιείται απευθείας ένα αντικείμενο Float[] με διαστάσεις [3,3].
Στις γραμμές 8,9 δηλώνονται δύο αναφορές αντικειμένων.
Στην γραμμή 11 δηλώνεται μια αλφαριθμητική μεταβλητή η οποία θα κρατά το όνομα της κλάσης που θα υλοποιηθεί με τον τρίτο τρόπο της δυναμικής υλοποίησης.

Στην γραμμή 13 υλοποιείται ένα αντικείμενο Float[], με την μέθοδο Object.New().

Στην γραμμή 15 εκχωρείται στην αλφαριθμητική μεταβλητή το όνομα της κλάσης.
Στην γραμμή 16 υλοποιείται δυναμικά ένα αντικείμενο από την κλάση Float[].

Στις γραμμές 18-20 τυπώνονται στην προκαθορισμένη έξοδο τα αντικείμενα που έχουν υλοποιηθεί με τους τρεις αυτούς τρόπους.

Object

Η στατική κλάση Object παρέχει πολλές μεθόδους οι οποίες επιτρέπουν την διαχείριση των αντικειμένων που δημιουργούνται.

Στατικές μέθοδοι
Attach Call Class Count Detach GetProperty Is IsLocked IsValid Lock New Parent SetProperty Type Unlock

SUPER

Σύνταξη:

SUPER

Επιστρέφει μια αναφορά στο τρέχον αντικείμενο το οποίο επιστρέφει τον πίνακα συμβόλων της κλάσης που κληρονομείται.

Η διαφορά της λέξης ME από την λέξη SUPER είναι ότι η πρώτη επιστρέφει τα μέλη της τρέχουσας κλάσης που είναι τα κληρονομούμενα μέλη και τα νέα μέλη της κλάσης ενώ η λέξη SUPER επιστρέφει τα μέλη μόνο της κλάσης που κλειρονομείται.

Έτσι για παράδειγμα αν έχουμε την κλάση Class1, με ένα μέλος, μια αλφαριθμητική μεταβλητή sName και μια κλάση Class2 που κληρονομεί αυτήν και η οποία έχει άλλα μέλη όπως νέες μεταβλητές και μεθόδους, η λέξη SUPER θα επιστρέψει μόνο τα μέλη της Class1.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project89.
Το έργο αποτελείται από τρία αρχεία. Το Class1, Class2 και Main.

Ο κώδικας του Class1 είναι:

```
1  ' Gambas class file
2
3  Public sName As String
```

Η κλάση Class1 έχει ένα μόνο μέλος μια αλφαριθμητική μεταβλητή.

Ο κώδικας του Class2 είναι:

```
1   ' Gambas class file
2
3   Inherits Class1
4
5   Public sLastName As String
6
7   Public Sub PrintName()
8
9   Super.sName = "Kostas"
10  Print Super.sName
11
12  End
```

Η κλάση Class2 κληρονομεί την Class1 στην γραμμή 3.
Έχει ένα νέο μέλος, μια αλφαριθμητική μεταβλητή στην γραμμή 5.
Και μια μέθοδο στις γραμμές 7-12.
Στις γραμμές 9 και 10 η λέξη SUPER επιστρέφει μόνο το μέλος sName από την κλάση Class1.

Ο κώδικας του module Main είναι:

```
1   ' Gambas module file
2
3   Public Sub Main()
4
5     Dim hObject As New Class2
6
7     hObject.PrintName
8
9   End
```

Τέλος στο module Main δηλώνουμε στην γραμμή 5 ένα νέο αντικείμενο και στην γραμμή 7 χρησιμοποιούμε την μέθοδο του hObject.PrintName.

Έξοδος:

```
Kostas
```

Κεφάλαιο 7ο – Διαχείριση ιδιοτήτων

Μια ιδιότητα (Property) μιας κλάσης μπορεί να δηλωθεί με την παρακάτω σύνταξη:

Σύνταξη:

	PROPERTY [**READ**] Identifier **AS** Datatype

Όπου Identifier είναι το όνομα της ιδιότητας και Datatype ο τύπος δεδομένων αυτής.
Αν δηλωθεί με την λέξη READ τότε η ιδιότητα είναι μόνο για ανάγνωση.

Μόλις δηλωθεί μια ιδιότητα πρέπει να υλοποιηθούν οι δυναρτήσεις ανάγνωσης και εγγραφής των δεδομένων στην ιδιότητα.

Το όνομα της συνάρτησης ανάγνωσης είναι το όνομα της ιδιότητας ακολουθούμενο με μια κάτω παύλα και την λέξη Read. Η συνάρτηση αυτή δεν δέχεται παραμέτρους πρέπει όμως να επιστρέψει δεδομένα με τύπο δεδομένων ίδιο με αυτόν της ιδιότητας.

Το όνομα της συνάρτησης εγγραφής είναι το όνομα της ιδιότητας ακολουθούμενο από μια κάτω παύλα και την λέξη Write. Η συνάρτηση αυτή δεν επιστρέφει κάποια τιμή αλλά δέχεται μια μόνο παράμετρο με τύπο δεδομένων τον ίδιο με αυτόν της ιδιότητας.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project90. Το έργο αποτελείται από δύο αρχεία. Το άρθρωμα Main και την κλάση CClass1.

Ο κώδικας του CClass1 είναι:

```
1   ' Gambas class file
2
3   Property Enabled As Boolean
4   Property Read Handle As Integer
5
6   Private $bEnabled As Boolean
7   Private $iHandle As Integer
8
9   ' Implements the Enabled property
10
11  Function Enabled_Read() As Boolean
12
13    Return $bEnabled
14
15  End
16
17  Sub Enabled_Write(bEnabled As Boolean)
18
19    $bEnabled = bEnabled
20
21  End
22
23  ' Implements the Handle property
24
25  Function Handle_Read() As Integer
26
27    Return $iHandle
28
29  End
```

Ανάλυση:
Στις γραμμές 3,4 δηλώνονται δύο ιδιότητες εκ των οποίων η δεύτερη είναι μόνο προς ανάγνωση.
Στις γραμμές 6,7 δηλώνονται δύο ιδιωτικές μεταβλητές οι οποίες θα χρησιμοποιηθούν εσωτερικά στην κλάση στις συναρτήσεις Read, Write των ιδιοτήτων.
Στις γραμμές 11-15 υπάρχει η συνάρτηση ανάγνωσης της ιδιότητας Enabled.
Η συνάρτηση αυτή δεν δέχεται παραμέτρους αλλά επιστρέφει την τιμή της μεταβλητής $bEnabled.
Στις γραμμές 17-21 υπάρχει η συνάρτηση εγγραφής της ιδιότητας Enabled.
Η συνάρτηση αυτή δέχεται μια τιμή τύπου Boolean στην παράμετρο bEnabled και εκχωρεί την τιμή αυτή στην ιδιωτική μεταβλητή της κλάσης $bEnabled.

Στις γραμμές 25-29 υπάρχει η συνάρτηση ανάγνωσης της ιδιότητας Handle η οποία επιστρέφει μόνο την τιμή της ιδιωτικής μεταβλητής $iHandle. Η τιμής αυτής της μεταβλητής θα πρέπει να τεθεί εσωτερικά στην κλάση αφού η ιδιότητα είναι μόνο για ανάγνωση.
Ο κώδικας του αρθρώματος Main είναι:

```
1   ' Gambas module file
2
3   Public Sub Main()
4
5     Dim hObject As CClass1
6     hObject = New CClass1
7
8     hObject.Enabled = True
9
10   Print hObject.Handle
11   Print hObject.Enabled
12
13  End
```

Ανάλυση:
Στην γραμμή 5 δηλώνουμε ένα αντικείμενο τύπου CClass1.
Στην γραμμή 6 δημιουργούμε το αντικείμενο αυτό.
Στην γραμμή 8 εκχωρούμε μια τιμή στην ιδιότητα του Enabled.
Στις γραμμές 10,11 τυπώνουμε στην προκαθορισμένη έξοδο τις τιμές των ιδιοτήτων του αντικειμένου.

Έξοδος:

```
0
True
```

Κεφάλαιο 8ο – Διαχείριση μεθόδων

Σε μια κλάση μπορούμε να έχουμε μεθόδους και συναρτήσεις.
Η μέθοδος είναι ένα μέλος μιας κλάσης η οποία δεν επιστρέφει κάποια τιμή συνήθως, ενώ η συνάρτηση είναι ένα μέλος μιας κλάσης η οποία επιστρέφει μια τιμή.
Για παράδειγμα μια μέθοδος που τυπώνει ένα μήνυμα στην προκαθορισμένη έξοδο δεν επιστρέφει κάποια τιμή ενώ μια συνάρτηση ανάγνωσης μιας ιδιότητας επιστρέφει μια τιμή ενός τύπου δεδομένων.

Οι μέθοδοι μπορεί να είναι:

- *Δημόσιες (public)* – Τότε αυτές γίνονται μέρος της διεπαφής της κλάσης.
- *Ιδιωτικές (private)* – Τότε μόνο άλλες μέθοδοι της κλάσης μπορούν να τις χρησιμοποιήσουν.
- *Στατικές (static)* – Τότε οι μέθοδοι γίνονται καθολικές και μπορούν να χρησιμοποιήσουν μόνο άλλα στατικά στοιχεία της κλάσης.

Μια μέθοδος μπορεί να επιστρέφει μια τιμή όπως οι συναρτήσεις. Ο τύπος δεδομένων της τιμής που επιστρέφεται είναι ίδιος με τον τύπο δεδομένων της δήλωσης της μεθόδου.

Μια μέθοδος μπορεί να δέχεται παραμέτρους που η κάθε μία να έχει ένα διαφορετικό τύπο δεδομένων ή όχι. Οι παράμετροι μπορούν να είναι προαιρετικοί.

Η σύνταξη μιας μεθόδου ή μιας συνάρτησης είναι η παρακάτω.

Σύνταξη:
Η σύνταξη μιας μεθόδου.

```
[ STATIC ] { PUBLIC | PRIVATE } { PROCEDURE | SUB }
Identifier
(
[ Parameter AS Datatype [ , ... ] ] [ , ]
[ OPTIONAL Optional Parameter AS Datatype [ , ... ] ] [ , ] [ ... ]
)
...
END
```

Σύνταξη:
Η σύνταξη μιας συνάρτησης.

```
[ STATIC ] { PUBLIC | PRIVATE } { FUNCTION | PROCEDURE | SUB }
Identifier
(
[ Parameter AS Datatype [ , ... ] ] [ , ]
[ OPTIONAL Optional Parameter AS Datatype [ , ... ] ] [ , ] [ ... ]
)
AS Datatype
...

RETURN Expression

END
```

Μια συνάρτηση στο τέλος επιστρέφει μια τιμή μιας έκφρασης η οποία αποδίδει τον ίδιο τύπο δεδομένων με αυτόν που έχει δηλωθεί η συνάρτηση.

Παράδειγμα:
Ο κώδικας μιας κλάσης CClass1 είναι:

```
1   ' Gambas class file
2
3   Property Enabled As Boolean
4
5   Private $bEnabled As Boolean
6
7   ' Implements the Enabled property
8
9   Function Enabled_Read() As Boolean
10
11    Return $bEnabled
12
13  End
14
15  Sub Enabled_Write(bEnabled As Boolean)
16
17    $bEnabled = bEnabled
18
19  End
```

Ανάλυση:
Στην γραμμή 9 δηλώνεται μια συνάρτηση τύπου δεδομένων Boolean. Η συνάρτηση επιστρέφει την τιμή της μεταβλητής $bEnabled.
Στην γραμμή 15 δηλώνεται μια μέθοδος η οποία δέχεται μια παράμετρο.

Κεφάλαιο 9ο – Διαχείριση συμβάντων

Η Gambas δίνει μια σειρά από εντολές για την διαχείριση των συμβάντων των αντικειμένων. Οι εντολές αυτές είναι:

EVENT	Δηλώνει ένα συμβάν.
RAISE	Εγείρει ένα συμβάν.
Object.Attach	Προσαρτεί ένα αντικείμενο σε έναν παρατηρητή συμβάντων.
Object.Detach	Αποπροσαρτεί ένα αντικείμενο από έναν παρατηρητή συμβάντων.
STOP EVENT	Ακυρώνει ένα συμβάν.
WAIT	Καλεί τον βρόχο συμβάντων για συγκεκριμένο χρόνο.

EVENT

Η λέξη EVENT δηλώνει σε μια κλάση ένα συμβάν.

Σύνταξη:

```
EVENT Name ( [ Parameter #1 [ , Parameter #2 ... ] )
```

Το συμβάν εγείρεται χρησιμοποιώντας την λέξη RAISE.

Η λέξη RAISE επιστρέφει έναν boolean που δείχνει αν ο διαχειριστής συμβάντων θέλει να ακυρώσει το συμβάν.

Διαχειριστής συμβάντος (Event Handler)

Το προκαθορισμένο όνομα της μεθόδου που καλείται όταν ένα συμβάν εγείρεται είναι, *Name_EventName.*

Για παράδειγμα αν έχετε μια κλάση CCar με ένα συμβάν Run και δημιουργήσετε ένα αντικείμενο hCar τότε η μέθοδος που θα καλείται όταν εγείρεται το συμβάν θα είναι η hCar_Run().

```
PUBLIC SUB hCar_Run(...)
```

Η προκαθορισμένη συμπεριφορά ωστόσο μπορεί να αλλάξει κάνοντας χρήση της μεθόδου Attach της κλάσης Object (Object.Attach) και των άλλων μεθόδων από την κλάση Object.
Παράδειγμα:

Το παράδειγμα είναι το Project91. Αποτελείται από μια κλάση CCar και ένα άρθρωμα Main.

Ο κώδικας της κλάσης CCar είναι:

```
1   ' Gambas class file
2
3   Event Run()
4
5
6   Public Sub StartEngine()
7
8   Print "Engine is started!"
9
10  End
11
12  Public Sub IncreaseSpeed()
13
14  Dim bResult As Boolean
15
16  ' Raises the event
17
18  bResult = Raise Run
19
20  If bResult = True Then
21    Print "Car is not running"
22  Else
23    Print "Car is running faster"
24  Endif
25
26  End
```

Ανάλυση:
Στην γραμμή 3 δηλώνουμε ένα συμβάν το Run().
Στην γραμμή 6 δηλώνουμε μια μέθοδο την StartEngine η οποία τυπώνει ένα μήνυμα στην προκαθορισμένη έξοδο.
Στην γραμμή 12 δηλώνουμε μια άλλη μέθοδο την IncreaseSpeed.
Στην γραμμή 14 δηλώνουμε μια μεταβλητή τύπου Boolean η οποία θα κρατά το αποτέλεσμα της εντολής RAISE από την γραμμή 18.
Αν το συμβάν εκτελεστεί επιτυχώς η γραμμή 23 θα εμφανίζει ένα μήνυμα αλλιώς η γραμμή 21 θα εμφανίζει ένα άλλο.

Ο κώδικας του module Main είναι:

```
1   ' Gambas module file
2
3   Private hCar As CCar
4
5   Public Sub Main()
6
7     hCar = New CCar
8
9     Object.Attach(hCar, Me, "hCar")
10    hCar.StartEngine
11    hCar.IncreaseSpeed
12
13  End
14
15  Public Sub hCar_Run()
16
17    Print "Car is running"
18
19  End
```

Ανάλυση:
Στην γραμμή 1 δηλώνουμε ένα καθολικό ιδωτικό αντικείμενο τύπου CCar.
Η Main() αρχίζει στην γραμμή 5.
Στην γραμμή 7 δημιουργούμε το αντικείμενο hCar που είναι τύπου CCar.
Στην γραμμή 9 προσαρτούμε στο αντικείμενο hCar τον διαχειριστή συμβάντων hCar.
Στην γραμμή 10 καλούμε την μέθοδο StartEngine και εμφανίζεται το μήνυμα στην προκαθορισμένη έξοδο “Engine is started!”.
Στην γραμμή 11 καλούμε την μέθοδο IncreaseSpeed.
Η μέθοδος IncreaseSpeed στην γραμμή 18 της κλάσης καλεί το συμβάν Run() έτσι ο διαχειριστής συμβάντων hCar καλεί την μέθοδο στην γραμμή 15 του module Main και εμφανίζεται το μήνυμα “Car is running”.
Επειδή το συμβάν δεν διακόπτεται στην γραμμή 18 της κλάσης η μεταβλητή bResult έχει την τιμή FALSE.
Έτσι στην δομή ελέγχου στις γραμμές 20-24 τυπώνεται το μήνυμα της γραμμής 23 “Car is running faster”

Έξοδος:

```
Engine is started!
Car is running
Car is running faster
```

RAISE

Σύνταξη:

```
[ ReturnVariable = ] RAISE Event ( [ Parameter1 [ ,
Parameter2 ... ] )
```

Η λέξη αυτή εγείρει ένα συμβάν που έχει δηλωθεί με την λέξη EVENT.
Μπορείτε να δηλώσετε μια μεταβλητή τύπου boolean η οποία θα δέχεται το αποτέλεσμα της RAISE. Όταν η RAISE επιστρέφει TRUE ο διαχειριστής συμβάντων έχει ακυρώσει το συμβάν με την οδηγία STOP EVENT.

Object.Attach

Σύνταξη:

```
STATIC SUB Attach ( Object AS Object, Parent AS Object, Name AS
String )
```

Προσαρτεί ένα αντικείμενο σε ένα πατρικό αντικείμενο.

`Object` είναι το αντικείμενο που θα προσαρτηθεί.

`Parent` είναι το αντικείμενο γονέας στο οποίο θα προσαρτηθεί το αντικείμενο `Object`.

Name είναι το όνομα που χρησιμοποιείται για την εύρεση του διαχειριστή συμβάντος (event-handler) στο πατρικό αντικείμενο

Κάθε συμβάν που εγείρεται από ένα αντικείμενο το διαχειρίζεται ο event-handler που βρίσκεται στο γονικό αντικείμενο.
Εάν ο γονέας είναι μια κλάση τότε οι διαχειριστές συμβάντων θα είναι στατικές μέθοδοι αυτής της κλάσης.

Object.Detach

Σύνταξη:

```
STATIC SUB Detach ( Object AS Object )
```

Αποπροσαρτεί ένα αντικείμενο από τον γονέα του.

Τα συμβάντα που εγείρονται από το αντικείμενο αγνοούνται.

STOP EVENT

Σύνταξη:

	`STOP EVENT`

Η δήλωση αυτή χρησιμοποιείται εντός ενός διαχειριστή συμβάντος για να ακυρώση το συμβάν.

Μόνο μερικά συμβάντα μπορούν να ακυρωθούν.

Παράδειγμα:
Το παράδειγμα αυτό είναι η αντιγραφή του Project91 με μια μικρή αλλαγή. Το νέο έργο είναι το Project92. Η αλλαγή υπάρχει στο αρχείο Main.module όπου στον διαχειριστή συμβάντων τοποθετούμε την οδηγία STOP EVENT στην γραμμή 17.

Ο κώδικας του module Main είναι:

```
1   ' Gambas module file
2
3   Private hCar As CCar
4
5   Public Sub Main()
6
7     hCar = New CCar
8
9     Object.Attach(hCar, Me, "hCar")
10    hCar.StartEngine
11    hCar.IncreaseSpeed
12
13  End
14
15  Public Sub hCar_Run()
16
17    Stop Event
18
19    Print "Car is running"
20
21  End
```

Ανάλυση:
Στην γραμμή 17 υπάρχει η οδηγία STOP EVENT.
Το πρόγραμμα εκτελώντας την οδηγία αυτή δεν θα προχωρήσει στην γραμμή 19.

Από την άλλη η οδηγία RAISE Run θα επιστρέψει την τιμή TRUE διότι το συμβάν ακυρώθηκε. Έτσι το πρόγραμμα θα κάνει τον έλεγχο στις γραμμές 20-24 στο αρχείο CCar και θα τυπώσει το μήνυμα “Car is not running”.
Ο κώδικας του αρχείου CCar.

```
18  bResult = Raise Run
19
20  If bResult = True Then
21    Print "Car is not running"
22  Else
23    Print "Car is running faster"
24  Endif
```

WAIT

Σύνταξη:

```
WAIT [ Delay ]
```

Η εντολή WAIT καλεί σε επανάληψη τον βρόχο των συμβάντων (event loop).
Εάν έχει οριστεί η παράμετρος Delay η συνάρτηση δεν επιστρέφει έως ότου περάσει ο χρόνος των Delay δευτερολέπτων.

Εάν δεν δηλωθεί χρόνος Delay η συνάρτηση επεξεργάζεται όλα τα συμβάντα που βρίσκονται στην ουρά επεξεργασίας αμέσως. Σε αυτήν την ειδική περίπτωση συμβάντα από το πληκτρολόγιο και το ποντίκι αγνοούνται.

Ο χρόνος Delay είναι μια τιμή float και έτσι μπορείτε να ορίσετε και ms για παράδειγμα 100 ms ορίζονται ως: `WAIT 0.1`

Παράδειγμα:

```
' Waits a little, letting the user interacts with the GUI
WAIT 0.1

' Waits, but the user can just watch what happens...
WAIT
```

Η κλήση της εντολής WAIT δεν θα πρέπει να γίνεται από έναν διαχειριστή συμβάντων.
Αν γίνει αυτό η εντολή θα δημιουργήσει ατέρμονα βρόχο κλήσης του διαχειριστή συμβάντος.

Παράδειγμα:

```
PUBLIC SUB MySerialPort_Read()

  DIM sData, sTemp AS String
  DIM iTries AS Integer

  FOR iTries = 1 TO 5
     ' wait a bit and read response.
    WAIT 0.01
     ' see if we got some data.
    READ #LAST, sTemp, Lof(LAST)
    sData &= sTemp
    ...
 NEXT

END
```

Η εντολή `WAIT` καλείται χωρίς να διαβάζει κάτι από την σειριακή θύρα. Μόλις τα δεδομένα περάσουν από την σειριακή και υπάρχει κάτι να διαβαστεί καλείται το συμβάν Read σε επανάληψη ξανά και ξανά από την εντολή WAIT μέχρι η στοίβα γεμίσει.
Σε αυτήν την περίπτωση αντί της εντολής WAIT θα έπρεπε να χρησιμοποιήσετε την εντολή SLEEP.

Μέρος 4ο, Δουλεύοντας με τις κλάσεις

Κεφάλαιο 1ο – Γραφική διεπαφή χρήστη (GUI)

gb.qt4

Το εξάρτημα αυτό υλοποιεί την γραφική διεπαφή του χρήστη. Είναι βασισμένο στην βιβλιοθήκη QT.

Κλάσεις

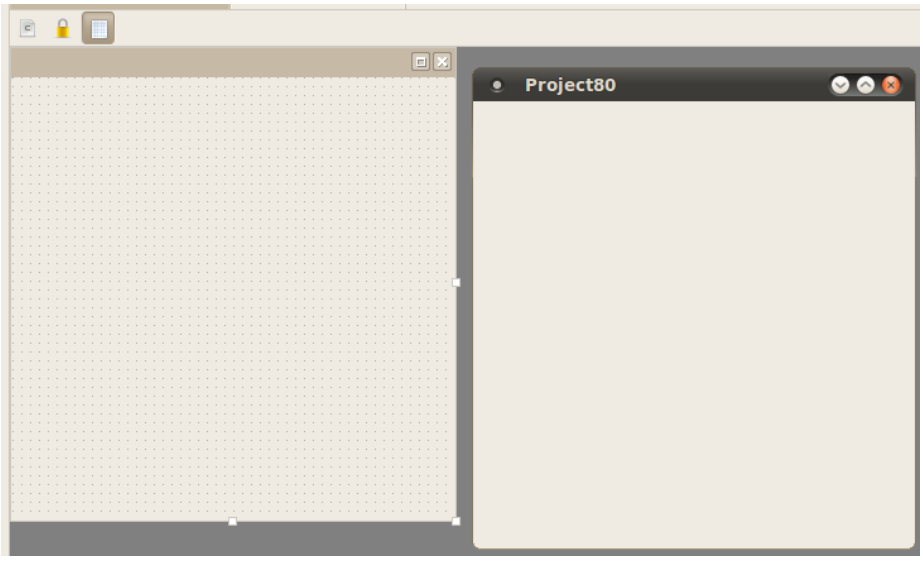
Περιέχει τις εξής κλάσεις

Action	Drag	Label	ProgressBar	ToggleButton
Align	Draw	Line	RadioButton	ToolButton
Application	DrawingArea	ListBox	Screen	TrayIcon
Arrange	Embedder	ListView	Screens	TrayIcons
Border	Fill	Menu	Scroll	TreeView
Button	Font	Message	ScrollBar	UserContaine
CheckBox	Fonts	Mouse	ScrollView	r
Clipboard	Form	MovieBox	Select	UserControl
Color	Frame	Paint	Separator	VBox
ColumnView	GridView	PaintBrush	Slider	VPanel
ComboBox	HBox	PaintExtents	SpinBox	VSplit
Container	HPanel	PaintMatrix	SvgImage	Watcher
Control	HSplit	Panel	TabStrip	Window
Cursor	IconView	Picture	TextArea	WindowType
Desktop	Image	PictureBox	TextBox	Windows
Dialog	Key	Printer	TextLabel	

Δουλεύοντας με τις φόρμες, η κλάση Form

Μια φόρμα, Form, είναι η πατρική κλάση για κάθε στοιχείο ελέγχου (Control). Οτιδήποτε τοποθετείται σε μια “πατρική κλάση” (parent class) είναι μια “κλάση παιδί” (child class) έτσι τα στοιχεία ελέγχου (Controls) είναι οι κλάσεις παιδιά με κλάση γονέα μια φόρμα. Η φόρμα κληρονομεί την κλάση Window.

Η κλάση έχει την παρακάτω απεικόνιση.



Εικόνα 79: Η απεικόνιση μιας φόρμας.

Στα αριστερά φαίνεται η προβολή σχεδίασης και στα δεξιά η προβολή εκτέλεσης της εφαρμογής.

Μια φόρμα αποτελείται από την επικεφαλίδα που φέρει το όνομα της φόρμας, τα κουμπιά ελαχιστοποίησης, μεγιστοποίησης και κλεισίματος και από την επιφάνεια πάνω στην οποία θα τοποθετηθούν τα στοιχεία ελέγχου όπως κουμπιά, λίστες αρχείων, πλαίσια, κουτιά κειμένου κτλ.

Στην προβολή σχεδίασης εμφανίζεται ένα πλέγμα με σημεία στην επιφάνεια της φόρμας για να στοιχίζουμε τα στοιχεία ελέγχου που θα τοποθετήσουμε πάνω σε αυτήν.

Το αντικείμενο Form μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hForm AS Form

hForm = NEW Form ( ) [ AS "event name" ]
```

Η φόρμα έχει ιδιότητες (**Properties**), μεθόδους (**Methods**), συμβάντα (**Events**), στατικές μεθόδους (**Static methods**), και σταθερές (**Constants**).

Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Γεγονότα (Events)**
Action Arrangement AutoResize Background Border Caption Children ClientH ClientHeight ClientW ClientWidth ClientX ClientY Closed Controls Cursor Design Drop Enabled Expand Font Foreground FullScreen H Handle HasFocus Height Hovered Icon Id Ignore Left Margin Mask Maximized Menus Minimized Modal Mouse Name Next Opacity Padding Parent Persistent Picture PopupMenu Previous Proxy Resizable Screen ScreenX ScreenY SkipTaskbar Spacing Stacking Sticky Tag Text Title ToolTip Top TopLevel TopOnly Tracking Type Utility Visible W Width Window X Y	Center Close Delete Drag Find Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show ShowDialog ShowModal ShowPopup	Activate Arrange BeforeArrange Close DblClick Deactivate Drag DragMove Drop Enter GotFocus Hide Icon Insert KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel Move Open Resize Show Title
	Στατικές μέθοδοι (Static Medods)	
	Load Main	
	Σταθερές (Constants)	
	Above Below Normal	

Κάθε ένα μέλος της κλάσης περιγράφεται στην wiki-εγκυκλοπαίδεια της Gambas αναλυτικά. Εδώ θα δούμε ορισμένα παραδείγματα με τις φόρμες.

Ιδιότητες (Properties)

Action

PROPERTY **Action** AS String

Επιστρέφει ή θέτει το κείμενο string που θα συσχετιστεί με την λειτουργία Action.

Η λειτουργία Action αναφέρεται στην κλάση **Action**. Με αυτήν την κλάση μπορούμε να θέσουμε ένα string στην ιδιότητα Action σε ορισμένα στοιχεία ελέγχου που θέλουμε να εκτελούν τον ίδιο κώδικα.
Για παράδειγμα ένα κουμπί σε μια μπάρα ελέγχου με το όνομα “Αποθήκευση”, σε ένα μενού με το όνομα “Αποθήκευση” και σε ένα κουμπί με το όνομα “Αποθήκευση”.
Ο τελικός χρήστης όποιο από αυτά τα στοιχεία ελέγχου πατήσει θα θέλαμε να τρέξει τον ίδιο κώδικα αποθήκευσης ενός αρχείου.
Όλα αυτά τα στοιχεία ελέγχου θα πρέπει να έχουν το ίδιο string στην ιδιότητα τους Action.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project93.
Η φόρμα FMain έχει ένα μενού File και ένα υπομενού Save που έχει δημιουργηθεί με τον Menu Editor. Επίσης έχει ένα κουμπί Button1. Τόσο το κουμπί όσο και το μενού έχουν κοινή τιμή στην ιδιότητα Action έτσι ώστε να εκτελούν τον ίδιο κώδικα.

Ο κώδικας της FMain είναι:

```
1   ' Gambas class file
2
3   Public Sub Form_Open()
4    Me.Center
5    Menu1.Action = "save"
6    Button1.Action = "save"
7   End
8
9   Public Sub Action_Activate(hkey As String) As Boolean
10   Select Case hkey
11   Case "save"
12    ' call save sub at here
13     Print "save to a file."
14   End Select
15  End
```

Ανάλυση:
Στην γραμμή 3 με 7 υπάρχει η ρουτίνα Form_Open() η οποία καλείται όταν η φόρμα ανοίγει.
Στην γραμμή 4 τοποθετούμε στο κέντρο της οθόνης την φόρμα.
Στις γραμμές 5,6 θέτουμε την ιδιότητα Action τόσο του Menu1 όσο και του Button1.

Στις γραμμές 9 με 15 υπάρχει η ρουτίνα που διαχειρίζεται την ενεργοποίηση της ιδιότητας Action. Η ρουτίνα αυτή είναι η Action_Activate().
Η ρουτίνα αυτή δέχεται μια παράμετρο key η οποία είναι η τιμή Menu1.Action ή η Button1.Action.

Στις γραμμές 10 με 14 υπάρχει μια δομή ελέγχου SELECT ... CASE η οποία ελέγχει την τιμή της μεταβλητής hkey. Στην περίπτωση που η τιμή της μεταβλητής είναι “save” τυπώνεται ένα μήνυμα στην προκαθορισμένη έξοδο.

Στην θέση της εντολής PRINT θα μπορούσαμε να βάλουμε κώδικα ο οποίος να αποθηκεύει δεδομένα σε ένα αρχείο.

Στο παράδειγμα αυτό βλέπουμε ότι ο κώδικας για την διεργασία αποθήκευσης σε αρχείο έχει γραφτεί μια φορά και καλείται είτε από το πάτημα του κουμπιού είτε από την επιλογή ενός μενού.

Cursor

```
PROPERTY Cursor AS Cursor
```

Με αυτήν την ιδιότητα μπορείτε να ορίσετε έναν δείκτη ποντικιού.

Ένα αντικείμενο Cursor έχει την εξής δήλωση.

Σύνταξη:

```
DIM hCursor AS Cursor

hCursor = NEW Cursor ( Picture AS Picture [ , X AS Integer, Y AS
Integer ] )
```

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project94.

```
1  PUBLIC SUB Form_Open()
2
3  DIM hCursor AS Cursor
4  DIM pic AS Picture = Picture["hand.png"]
5
6  hCursor = NEW Cursor(pic)
7  FMain.Cursor = hCursor
8
9  END
```

Ανάλυση:
Στην γραμμή 3 δηλώνεται ένα αντικείμενο Cursor
Στην γραμμή 4 δηλώνεται ένα αντικείμενο Picture και του εκχωρείται σαν εικόνα η εικόνα "hand.png"
Στην γραμμή 6 δημιουργείται ένα αντικείμενο Cursor και τίθεται σαν εικονίδιο του η εικόνα pic.
Στην γραμμή 7 τίθεται σαν κέρσορας της φόρμας ο κέρσορας hCursor.

Font

PROPERTY **Font** AS Font

Επιστρέφει ή θέτει την γραμματοσειρά που θα χρησιμοποιήσει το αντικείμενο.

Για να αλλάξετε την γραμματοσειρά στην κατάσταση εκτέλεσης του προγράμματος μπορείτε να:

- χρησιμοποιήσετε την ιδιότητα Font από ένα άλλο αντικείμενο
- να δημιουργήσετε ένα αντικείμενο Font

Ένα αντικείμενο Font έχει την εξής σύνταξη

Σύνταξη:

```
DIM hFont AS Font

hFont = NEW Font ( [ Font AS String ] )
```

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project95.

```
1   ' Gambas class file
2
3   Public Sub Form_Open()
4
5   Dim hFont As Font
6   hFont = Font["DejaVu Sans Mono"]
7   hFont.Bold = True
8   hFont.Italic = True
9   hFont.Size = 10
10  FMain.Font = hFont
11
12  End
```

Ανάλυση:
Η φόρμα όταν ανοίγει καλεί την ρουτίνα Form_Open() στην γραμμή 3.
Στην γραμμή 5 δηλώνεται ένα αντικείμενο τύπου Font.
Στις γραμμές 6 με 9 τίθενται διάφορες τιμές της γραμματοσειράς όπως αν είναι Bold, Italic και μέγεθος.
Αφού έχει δημιουργηθεί και οριστεί η γραμματοσειρά στην γραμμή 10 εκχωρούμε το αντικείμενο Font στην ιδιότητα Font της φόρμας.
Κάθε αντικείμενο που έχει κείμενο και τοποθετείται πάνω στην φόρμα όπως το Label1 θα έχει ως γραμματοσειρά αυτή που έχουμε ορίσει στις γραμμές 5 με 9.

Icon

PROPERTY **Icon** AS Picture

Επιστρέφει ή θέτει την διαδρομή της εικόνας που θα χρησιμοποιηθεί από το αντικείμενο ως εικονίδιο.
Τα αρχεία εικόνας τοποθετούνται στον φάκελο του έργου και το περιβάλλον ανάπτυξης τα εμφανίζει στον κατάλογο Data.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project96.

```
1   ' Gambas class file
2
3   Public Sub Form_Open()
4
5   Me.Icon = Picture["icon.png"]
6
7   End
```

Ανάλυση:
Η γραμμή 5 φορτώνει για εικονίδιο της φόρμας την εικόνα icon.png

Mask

```
PROPERTY Mask AS Boolean
```

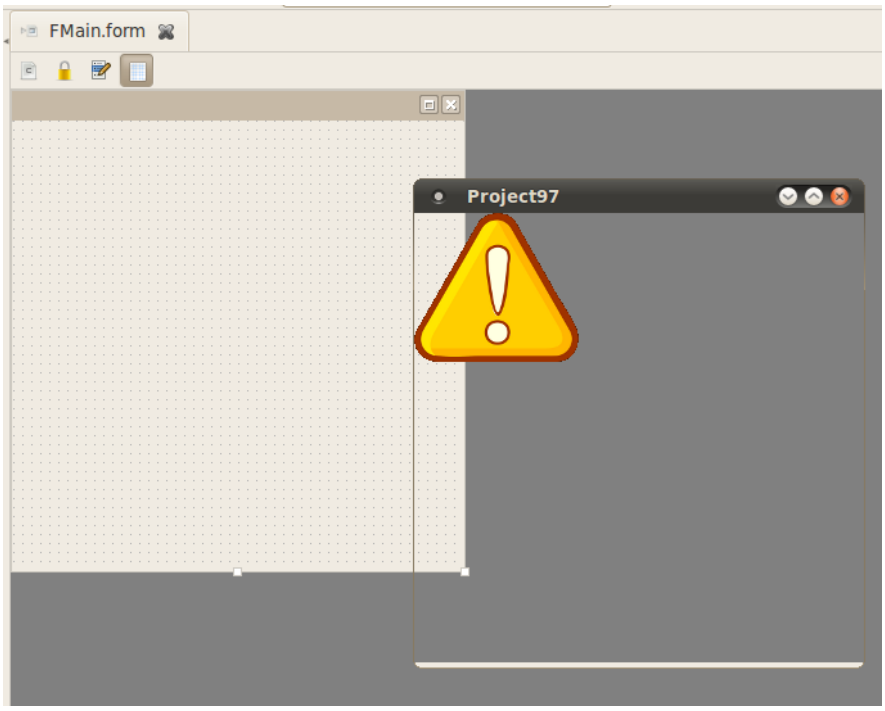
Η ιδιότητα αυτή συνδυάζεται με την ιδιότητα Picture.

Πρώτα ο προγραμματιστής θέτει μια εικόνα στην ιδιότητα Picture.

Επιστρέφει ή θέτει έναν Boolean ο οποίος εφαρμόζει την εικόνα της ιδιότητας Picture σαν μια μάσκα στο αντικείμενο.

Όταν εφαρμοστεί μια μάσκα σε ένα αντικείμενο στην κατάσταση εκτέλεσης το αντικείμενο είναι διάφανο εκτός από την περιοχή της εικόνας.

Ένα παράδειγμα φαίνεται στην παρακάτω εικόνα.



Εικόνα 80: Η εφαρμογή της μάσκας σε μια φόρμα

Το παράδειγμα αυτό είναι το Project97.
Σε αυτό το παράδειγμα θέτουμε τις ιδιότητες Mask σε TRUE και θέτουμε στην ιδιότητα Picture μια εικόνα.

Modal

```
PROPERTY READ Modal AS Boolean
```

Επιστρέφει έναν Boolean ο οποίος αναφέρει αν το αντικείμενο είναι σε αποκλειστική χρήση όταν η τιμή της ιδιότητας είναι TRUE.

Για να εμφανιστεί μια φόρμα σε αποκλειστική χρήση πρέπει να εμφανιστεί με την μέθοδο ShowModal().
Όταν μια φόρμα είναι σε αποκλειστική χρήση ο χρήστης δεν μπορεί να κάνει χρήση των άλλων φορμών.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project98.

Ο κώδικας της FMain είναι:

```
1  ' Gambas class file
2
3  Public Sub Button1_Click()
4
5  Form1.ShowModal
6
7  End
```

Ο κώδικας της Form1 είναι:

```
1  ' Gambas class file
2
3  Public Sub Form_Open()
4
5    Print Me.Modal
6
7  End
```

Ανάλυση:
Έχουμε δύο φόρμες την FMain και την Form1. Στην FMain έχουμε ένα κουμπί Button1 το οποίο όταν πατηθεί εμφανίζει σε αποκλειστική χρήση την φόρμα Form1 στην γραμμή 5 της FMain.
Μέχρι να κλείσουμε την φόρμα Form1 η FMain δεν δέχεται συμβάντα και χρήση από τον χρήστη.
Όταν η φόρμα Form1 ανοίγει τυπώνει στην προκαθορισμένη έξοδο την τιμή της ιδιότητας Modal.

Έξοδος:

```
True
```

ToolTip

```
PROPERTY Tooltip AS String
```

Επιστρέφει ή θέτει έναν String το οποίο εμφανίζεται όταν ο δείκτης του ποντικιού παραμένει μερικά δευτερόλεπτα πάνω από ένα αντικείμενο.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project99.

Εικόνα 81: Αφήνωντας τον δείκτη του ποντικιού για λίγα δευτερόλεπτα πάνω από την φόρμα εμφανίζεται το κείμενο Tooltip

Συμβάντα (Events)

KeyPress

EVENT **KeyPress** ( )

Το γεγονός αυτό εγείρεται όταν ένα αντικείμενο έχει την εστίαση και πατηθεί κάποιο πλήκτρο από το πληκτρολόγιο.
Μπορείτε να δείτε ποιο πλήκτρο πατήθηκε με την κλάση Key.

Παράδειγμα:
Το παράδειγμα αυτό είναι το Project100.
Το Project έχει μια φόρμα την FMain που είναι και η κλάση εκκίνησης.

```
1  ' Gambas class file
2  PUBLIC SUB Form_KeyPress()
3
4  PRINT "Key Code:";; Key.Code
5  PRINT "Key text:";; Key.Text
6
7  END
```

Έξοδος:

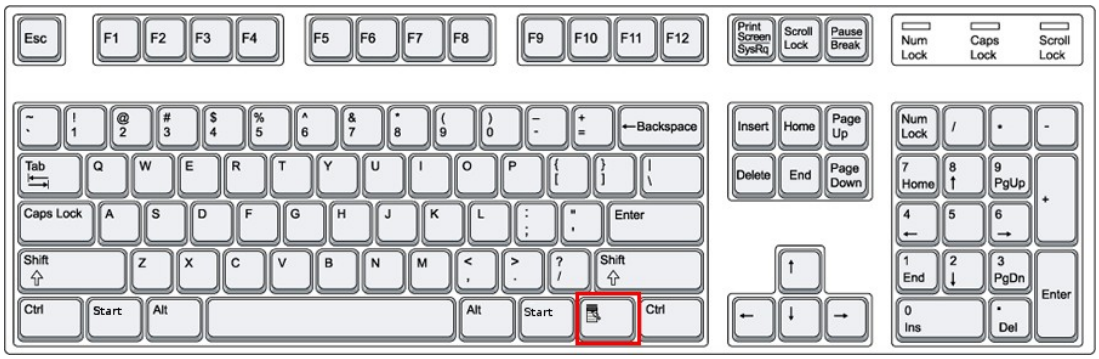
```
Key Code: 113
Key text: q
Key Code: 97
Key text: a
Key Code: 122
Key text: z
```

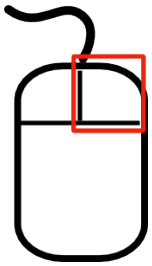
Ανάλυση:
Η φόρμα έχει κώδικα γραμμένο για το συμβάν KeyPress() της φόρμας.
Στην γραμμή 4 τυπώνουμε τον κώδικα του πλήκτρου που θα πατηθεί.
Στην γραμμή 5 τυπώνουμε το κείμενο του πλήκτρου που θα πατηθεί.
Όταν το πρόγραμμα τρέχει και πατήσουμε τα πλήκτρα q, a, z. Λαμβάνουμε το αποτέλεσμα που φαίνεται στην έξοδο.
Κάθε πλήκτρο έχει έναν μοναδικό αριθμό που καθορίζει στην Gambas ποιο πλήκτρο είναι. Τον κωδικό του πλήκτρου τον λαμβάνουμε από την ιδιότητα Code της κλάσης Key.
Ομοίως κάθε πλήκτρο έχει και το κείμενο που επιστρέφει όταν πατιέται. Το κείμενο αυτό το λαμβάνουμε από την ιδιότητα Text της κλάσης Key.

Menu

EVENT **Menu** ( )

Το γεγονός εγείρεται όταν ο χρήστης κάνει δεξί κλικ στο αντικείμενο ή πατήσει από το πληκτρολόγιο το κουμπί για την ανάδυση του μενού.

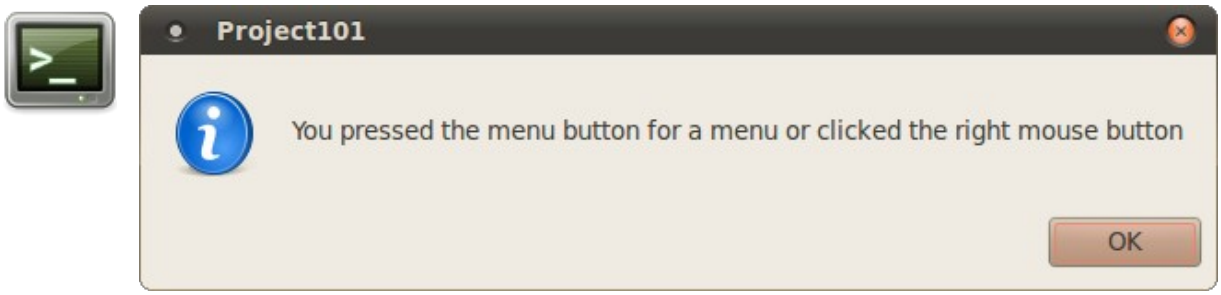




Παράδειγμα:
Το παράδειγμα αυτό είναι το Project101.
Το Project έχει μια φόρμα την FMain που είναι και η κλάση εκκίνησης.

```
1  ' Gambas class file
2
3  PUBLIC SUB Form_Menu()
4
5    Message("You pressed the menu button for a menu or clicked
   the right mouse button")
6
7  END
```

Έξοδος:



Ανάλυση:
Στην γραμμή 5 εμφανίζουμε ένα μήνυμα στον χρήστη όταν το συμβάν Menu εγερθεί.
Για να εγείρουμε το συμβάν όταν τρέχει το πρόγραμμα πατάμε το δεξί κουμπί από το ποντίκι ή το κουμπί για το μενού από το πληκτρολόγιο.

Εφαρμογή 1

Σε αυτό το παράδειγμα αλλάζουμε τα χρώματα του παρασκηνίου μιας φόρμας με κάποιο χρώμα που επιλέγεται τυχαία.

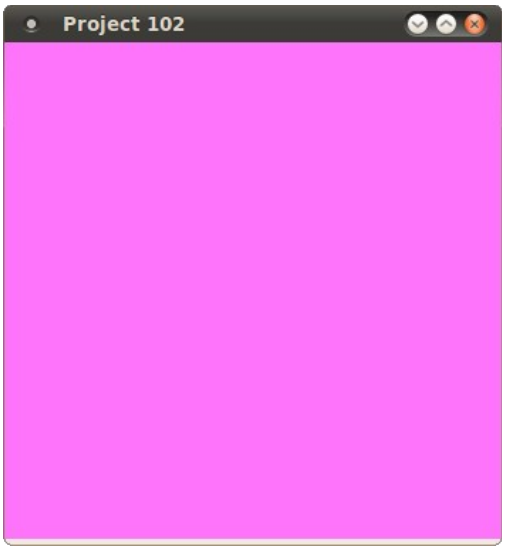
Όπως είδαμε ένα χρώμα είναι της μορφής RGB (Red, Green, Blue) και κάθε τιμή του κόκκινου, πράσινου, μπλε είναι μεταξύ του 0-255.

Το πρόγραμμα αυτό παράγει τρεις τυχαίους αριθμούς από το 0 μέχρι το 255.
Ο πρώτος αριθμός είναι το κόκκινο, ο δεύτερος το πράσινο και ο τρίτος το μπλε.
Οι τιμές αυτές μετατρέπονται σε δεκαεξαδική μορφή και εκχωρούνται σαν τιμή της ιδιότητας Background.
Ο χρήστης κάθε φορά που κάνει κλικ στην φόρμα αλλάζει χρώμα το παρασκήνιο της φόρμας.

Το παράδειγμα αυτό είναι το Project102.

```
1   ' Gambas class file
2
3   PUBLIC FUNCTION RndColor() AS Integer
4
5     DIM Red AS Integer
6     DIM Green AS Integer
7     DIM Blue AS Integer
8     DIM sColor AS String
9     DIM iColor AS Integer
10
11    Red = Rnd(0, 256)
12    Green = Rnd(0, 256)
13    Blue = Rnd(0, 256)
14
15    sColor = "&" & Hex(Red, 2) & Hex(Green, 2) & Hex(Blue, 2)
      & "&"
16    PRINT sColor
17    iColor = Val(sColor)
18    RETURN iColor
19
20  END
21
22  PUBLIC SUB Form_MouseDown()
23
24    ME.Background = RndColor()
25
26  END
```

Έξοδος:




```
&D8E466&
&BA17B0&
&A29F35&
&AEA0F5&
&AC0FB6&
```

Εικόνα 82: Το Background της φόρμας έχει χρώμα μωβ

Ανάλυση:
Στις γραμμές 3 με 20 υπάρχει η συνάρτηση RndColor().
Στις γραμμές 22 με 26 υπάρχει το συμβάν Form_MouseDown().

Ας δούμε αναλυτικά αυτά τα μέρη του προγράμματος μας.
Στις γραμμές 5 με 9 δηλώνονται κάποιες μεταβλητές.
Θέλουμε τρεις μεταβλητές ακεραίου όπου θα εκχωρούνται οι τυχαίοι αριθμοί από το εύρος 0 με 255.
Κάθε ένας από τους τρεις αριθμούς αυτούς θα αντιστοιχεί σε ένα χρώμα. Το Red, Green, Blue.
Αφού οριστούν οι τρεις αριθμοί η δεκαεξαδική τους αναπαράσταση θα είναι η τιμή της μεταβλητής του χρώματος. Η τιμή αυτή όμως είναι ένας string αφού η συνάρτηση Hex$() η οποία μετατρέπει τους αριθμούς σε δεκαεξαδική μορφή επιστρέφει ένα string.
Έτσι ορίζουμε την μεταβλητή τύπου string sColor.

Στις γραμμές 11 με 13 οι μεταβλητές λαμβάνουν τυχαίες τιμές μεταξύ του εύρους 0 και 255.
Στην γραμμή 15 κάθε τιμή χρώματος μετατρέπεται σε δεκαεξαδική μορφή η οποία θέλουμε να έχει δύο ψηφία. Αυτό γίνεται διότι οι τιμές των μεταβλητών αυτών από το 1 έως το 15 στο δεκαεξαδικό σύστημα είναι με ένα ψηφίο. Εμείς όμως ως τιμή χρώματος θέλουμε να έχει πάντα δύο ψηφία δλδ οι τιμές να είναι από το 00 έως το FF.
Έτσι μετατρέπουμε τις τυχαίες τιμές των Red, Green, Blue σε δεκαεξαδικούς αριθμούς δύο ψηφίων καλώντας την συνάρτηση Hex$(integer, 2).

Στην συνέχεια ενώνουμε όλες τις δεκαεξαδικές τιμές και προσθέτουμε στην αρχή και στο τέλος τους τελικού String το σύμβολο & το οποίο δείχνει ότι το string αυτό είναι ένας δεκαεξαδικός

αριθμός.
Αυτή την εργασία τελεί η έκφραση

```
sColor = "&" & Hex(Red, 2) & Hex(Green, 2) & Hex(Blue, 2) & "&"
```

στην γραμμή 15.

Κάθε μέρος του string ενώνεται με τον τελεστή ένωσης αλφαριθμητικών που είναι το **&**.
Προσοχή το "&" είναι ένα αλφαριθμητικό διότι εσωκλείεται εντός διπλών εισαγωγικών και είναι διαφορετικό από τον τελεστή ένωσης string ο οποίος είναι το **&**.

Η sColor έχει λοιπόν εκχωρημένη την τελική τιμή του χρώματος σε δεκαεξαδική μορφή.
Στην γραμμή 16 τυπώνουμε στην stdout την τιμή αυτή για να την δούμε.
Επειδή όμως η ιδιότητα Background της φόρμας λαμβάνει τιμή ακεραίου θα πρέπει να μετατρέψουμε την sColor σε ακέραιο.
Αυτό γίνεται με την πρόταση στην γραμμή 17.
Τέλος στην γραμμή 18 η συνάρτηση επιστρέφει την τιμή της μεταβλητής iColor.

Το συμβάν Form_MouseDown() ξεκινά στην γραμμή 22.
Στην γραμμή 24 θέτουμε την ιδιότητα του χρώματος παρασκηνίου της φόρμας στην τιμή που επιστρέφει η συνάρτηση Color().
Η λέξη ME επιστρέφει το τρέχον αντικείμενο. Στην προκειμένη περίπτωση την FMain.

Έτσι κάθε φορά που ο χρήστης κάνει κλικ στην φόρμα αυτή θα αλλάζει χρώμα με τυχαία σειρά.

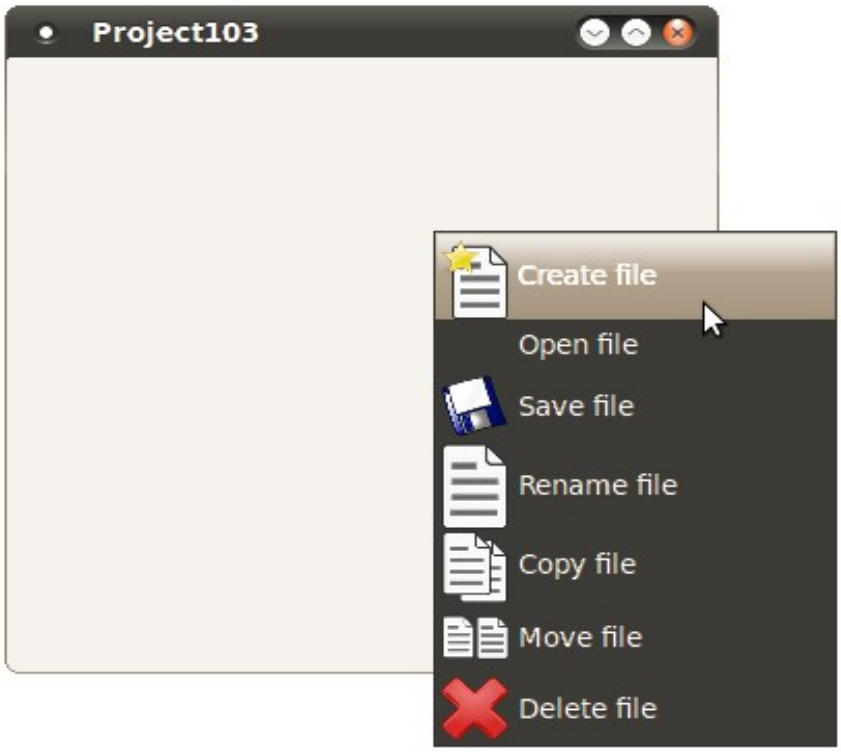
Εφαρμογή 2
Το επόμενο παράδειγμα είναι το Project103.
Σε αυτό το παράδειγμα φαίνεται πώς μπορούμε να κάνουμε ένα μενού εντολών το οποίο να εμφανίζεται κάνοντας δεξί κλικ στην φόρμα.
Το πρόγραμμα έχει μία κλάση την FMain.

```
1   ' Gambas class file
2
3   PRIVATE Mnu AS Menu
4   PRIVATE Opt0 AS Menu
5   PRIVATE Opt1 AS Menu
6   PRIVATE Opt2 AS Menu
7   PRIVATE Opt3 AS Menu
8   PRIVATE Opt4 AS Menu
9   PRIVATE Opt5 AS Menu
10  PRIVATE Opt6 AS Menu
11
12  PUBLIC SUB _new()
13   'initializing popup menu
14   Mnu = NEW Menu(ME)
15   Mnu.Name = "Mnu"
16
17   Opt0 = NEW Menu(Mnu) AS "PopUpMenu"
18   Opt0.Name = "Opt0"
19   Opt0.Caption = "Create file"
20   Opt0.Visible = TRUE
21   Opt0.Picture = Picture["icons/create_file.png"]
22
23   Opt1 = NEW Menu(Mnu) AS "PopUpMenu"
24   Opt1.Name = "Opt1"
25   Opt1.Caption = "Open file"
26   Opt1.Visible = TRUE
27   Opt1.Picture = Picture["icons/open_file.png"]
28
29   Opt2 = NEW Menu(Mnu) AS "PopUpMenu"
30   Opt2.Name = "Opt2"
31   Opt2.Caption = "Save file"
32   Opt2.Visible = TRUE
33   Opt2.Picture = Picture["icons/save_file.png"]
34
35   Opt3 = NEW Menu(Mnu) AS "PopUpMenu"
36   Opt3.Name = "Opt3"
37   Opt3.Caption = "Rename file"
38   Opt3.Visible = TRUE
39   Opt3.Picture = Picture["icons/rename_file.png"]
40
```

```
41   Opt4 = NEW Menu(Mnu) AS "PopUpMenu"
42   Opt4.Name = "Opt4"
43   Opt4.Caption = "Copy file"
44   Opt4.Visible = TRUE
45   Opt4.Picture = Picture["icons/copy_file.png"]
46
47   Opt5 = NEW Menu(Mnu) AS "PopUpMenu"
48   Opt5.Name = "Opt5"
49   Opt5.Caption = "Move file"
50   Opt5.Visible = TRUE
51   Opt5.Picture = Picture["icons/move_file.png"]
52
53   Opt6 = NEW Menu(Mnu) AS "PopUpMenu"
54   Opt6.Name = "Opt6"
55   Opt6.Caption = "Delete file"
56   Opt6.Visible = TRUE
57   Opt6.Picture = Picture["icons/delete_file.png"]
58
59 END
60
61 PUBLIC SUB PopUpMenu_Click()
62
63   SELECT CASE LAST.name
64     CASE "Opt0"
65         'create file code
66         Message("You clicked create file menu")
67
68     CASE "Opt1"
69         'open file code
70         Message("You clicked open file menu")
71
72     CASE "Opt2"
73         'save file code
74         Message("You clicked save file menu")
75
76     CASE "Opt3"
77         'rename file code
78         Message("You clicked rename file menu")
79
80     CASE "Opt4"
81         'copy file code
82         Message("You clicked copy file menu")
83
84     CASE "Opt5"
85         'move file code
86         Message("You clicked move file menu")
```

```
87
88      CASE "Opt6"
89          'delete file code
90          Message("You clicked delete file menu")
91    END SELECT
92
93  END
94
95  PUBLIC SUB Form_Menu()
96
97    Mnu.Popup
98
99  END
```

Έξοδος:



Ανάλυση:
Το Project έχει έναν φάκελο Data ο οποίος φαίνεται στα αριστερά στον περιηγητή αρχείων του έργου. Αυτός ο φάκελος κανονικά δεν υπάρχει στην πραγματικότητα, είναι ένας τρόπος το περιβάλλον ανάπτυξης εφαρμογών να ξεχωρίζει τα αρχεία των κλάσεων με τα αρχεία που εισάγει ο προγραμματιστής.
Σε αυτόν τον φάκελο μπορούμε να δημιουργήσουμε έναν νέο φάκελο με το όνομα icons και

στο οποίο να βάλουμε τα εικονίδια που θέλουμε να εμφανίζονται στο menu.
Τα εικονίδια αυτά αποθηκεύονται στον σκληρό δίσκο στον φάκελο icons που υπάρχει εντός του φακέλου Project103.

Ξεκινάμε το πρόγραμμα με δηλώσεις ιδιωτικών καθολικών αντικειμένων menu στον κώδικα της κλάσης FMain.
Έτσι δηλώνουμε τα αντικείμενα Mnu, και Opt0 έως Opt6.

Πριν να εμφανιστεί η φόρμα FMain όταν το αντικείμενο πρόκειται να δημιουργηθεί καλείται η ρουτίνα PUBLIC SUB _new(). Σε αυτήν την ρουτίνα μπορούμε να δημιουργήσουμε το menu το οποίο θα εμφανίζεται κάνοντας δεξί κλικ στην φόρμα.

Στην γραμμή 14 δημιουργείται ένα αντικείμενο Menu.
Η σύνταξη δημιουργίας ενός μενού είναι:

Σύνταξη:

```
DIM hMenu AS Menu

hMenu = NEW Menu ( Parent AS Object [ , Hidden AS Boolean ] ) [ AS "event name" ]
```

Τις δηλώσεις αντικειμένων Menu τις κάναμε στην περιοχή του κώδικα δήλωσης καθολικών αντικειμένων και μεταβλητών με τις εκφράσεις

```
PRIVATE Mnu AS Menu
PRIVATE Opt0 AS Menu
PRIVATE Opt1 AS Menu
PRIVATE Opt2 AS Menu
PRIVATE Opt3 AS Menu
PRIVATE Opt4 AS Menu
PRIVATE Opt5 AS Menu
PRIVATE Opt6 AS Menu
```

Τώρα στην ρουτίνα δημιουργίας της FMain, PUBLIC SUB _new() θα δημιουργήσουμε τα αντικείμενα menu.

Όπως βλέπουμε ένα αντικείμενο menu δημιουργείται θέτοντας ένα γονικό αντικείμενο Parent.
Έτσι το μενού μας θα έχει γονικό αντικείμενο την FMain.
Αυτό γίνεται στην γραμμή 14 όπου η έκφραση

```
Mnu = NEW Menu(ME)
```

δημιουργεί ένα αντικείμενο μενού με γονικό αντικείμενο την FMain.

Κάθε αντικείμενο επιλογής το οποίο θα είναι και αυτό ένα menu θα έχει γονικό αντικείμενο το

πρώτο μενού, δλδ το Mnu.
Έτσι τα επόμενα αντικείμενα Opt0, Opt1 κτλ δημιουργούνται με τις γραμμές

```
Opt0 = NEW Menu(Mnu) AS "PopUpMenu"
```

Η λέξη PopUpMenu είναι δικός μας προσδιορισμός για τα αντικείμενα menu που είναι οι επιλογές του βασικού μενού Mnu.

Κάθε μενού έχει ιδιότητες όπως

```
Opt0.Name
Opt0.Caption
Opt0.Visible
Opt0.Picture
```

οι οποίες τίθενται αμέσως μετά την δημιουργία ενός μενού.

Αυτό ακριβώς γίνεται στις γραμμές 17 έως 57.

Για να θέσουμε μια εικόνα σε ένα μενού εκχωρούμε ένα αντικείμενο Picture στην ιδιότητα Picture του μενού.

Αυτό γίνεται με τις εκφράσεις

```
Opt0.Picture = Picture["icons/create_file.png"]
Opt1.Picture = Picture["icons/open_file.png"]
...
```

εντός των αγκυλών δηλώνουμε την σχετική διαδρομή του αρχείου εικόνας που θέλουμε να έχει το μενού.

Αφού λοιπόν έχουμε δημιουργήσει τα αντικείμενα μενού θα πρέπει τώρα να γράψουμε κώδικα για τα αντικείμενα αυτά για το συμβάν click. Δλδ τι θα γίνεται όταν ο χρήστης κάνει κλικ σε ένα αντικείμενο μενού.

Όπως είδαμε όλες οι επιλογές μενού είναι αντικείμενα με το όνομα PopUpMenu.
Το αντικείμενο PopUpMenu έχει και αυτό το συμβάν Click().
Επομένως ο κώδικας μας θα γραφτεί στην ρουτίνα συμβάντος PUBLIC SUB PopUpMenu_Click().
Για να δούμε ποιο από όλα τα μενού έχει κάνει κλικ ο χρήστης χρησιμοποιούμε την έκφραση LAST.name.
Η έκφραση αυτή επιστρέφει το τελευταίο όνομα του αντικειμένου PopUpMenu.

Έτσι στην ρουτίνα συμβάντος PUBLIC SUB PopUpMenu_Click() μπορούμε να χρησιμοποιήσουμε έναν βρόχο SELECT CASE για την έκφραση LAST.name.
Κάθε περίπτωση διαχειρίζεται και μια επιλογή ονόματος από τα αντικείμενα PopUpMenu.
Ο βρόχος SELECT CASE εκτείνεται στις γραμμές 62 μέχρι 92.

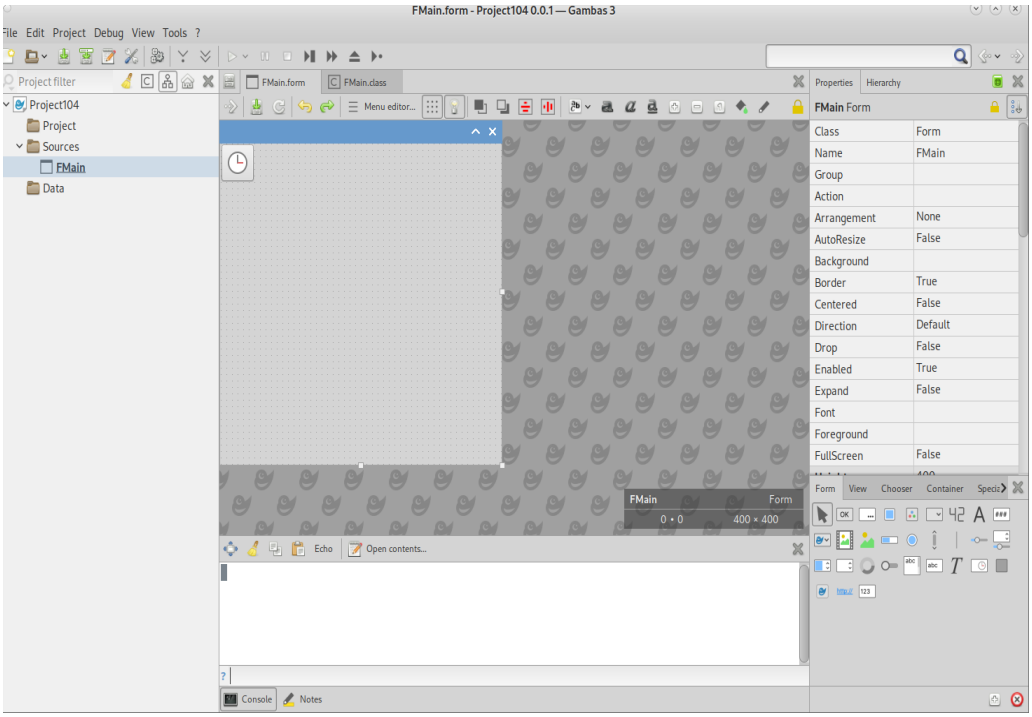
Κάθε επιλογή εμφανίζει ένα κιβώτιο μηνύματος με σχετικό μήνυμα.

Τέλος θα πρέπει να εμφανίσουμε το μενού Mnu.
Αυτό γίνεται στην ρουτίνα συμβάντος PUBLIC SUB Form_Menu().
Η ρουτίνα αυτή καλείται κάθε φορά που ο χρήστης κάνει δεξί κλικ στην φόρμα ή πατάει το πλήκτρο μενού από το πληκτρολόγιο.

Έτσι ο κώδικας που πρέπει να γραφτεί σε αυτήν την ρουτίνα είναι.

```
PUBLIC SUB Form_Menu()

  Mnu.Popup

END
```

Εφαρμογή 3
Το παράδειγμα αυτό είναι το Project104.
Το παράδειγμα αυτό έχει μια κλάση την FMain.
Σε αυτό το πρόγραμμα έχουμε ένα στοιχείο ελέγχου μέτρησης χρόνου, έναν Timer.



Εικόνα 83: Η σχεδίαση του Project104

Θέλουμε ο Timer να καλεί κάθε ένα δευτερόλεπτο μια ρουτίνα η οποία θα αλλάζει τον δείκτη του ποντικιού. Οι δείκτες του ποντικιού έχουν τιμές σταθερών ακεραίους από το -2 έως το 13.
Ο κώδικας της FMain είναι ο εξής:

```
1   ' Gambas class file
2   PRIVATE i AS Integer = 13
3
4   PUBLIC SUB Timer1_Timer()
5
6     IF i < -2 THEN i = 13
7
8     IF i > 13 THEN i = -2
9
10    ME.Mouse = i
11    PRINT i;;
12
13    DEC i
14
15  END
```

Έξοδος:

```
13 12 11 10 9 8 7 6 5 4 3 2 1 0 -1 -2 13 12 11...
```

Ανάλυση:
Στην γραμμή 2 δηλώνεται ένας ακέραιος με αρχική τιμή 13.
Εφόσον έχουμε τοποθετήσει τον Timer στην φόρμα FMain θέτουμε την ιδιότητα του Delay στην τιμή 1000 και την ιδιότητα του Enabled σε TRUE.

Η ιδιότητα Delay είναι ο χρόνος των milisecond που βάσει αυτού ο Timer θα καλεί την ρουτίνα PUBLIC SUB Timer1_Timer(). Έτσι κάθε 1000 ms ή αλλιώς κάθε 1 sec θα καλείται η ρουτίνα PUBLIC SUB Timer1_Timer().

Στην ρουτίνα αυτή μειώνουμε την τιμή της μεταβλητής i από το 13 στο -2. Αν οι τιμή ξεπεράσει το -2 ξαναγίνεται 13 και αν ξεπεράσει το 13 γίνεται -2.

Στην γραμμή 10 αλλάζουμε τον δείκτη του ποντικιού σύμφωνα με την τιμή της μεταβλητής i.
Στην γραμμή 11 τυπώνουμε την τιμή της μεταβλητής i.
Στην γραμμή 13 μειώνουμε την τιμή της μεταβλητής i.

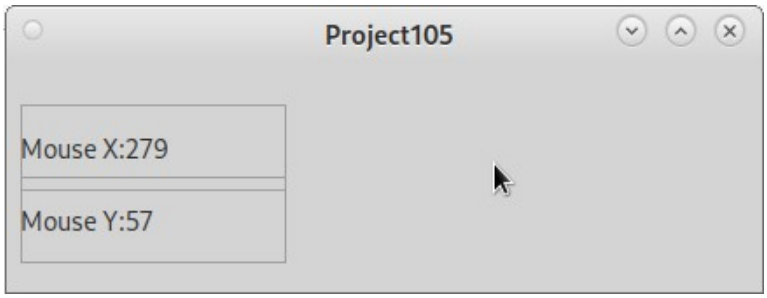
Όταν τρέχουμε το πρόγραμμα και έχουμε τον δείκτη του ποντικιού πάνω από την φόρμα, τότε βλέπουμε πως κάθε 1 sec αλλάζει εικονίδιο.

Εφαρμογή 4
Το παράδειγμα αυτό είναι το Project105.
Έχουμε μια φόρμα την FMain με δύο Labels. Καθώς είναι πατημένο ένα κουμπί του ποντικιού και κινείται ο κέρσορας στην φόρμα εμφανίζονται οι συντεταγμένες του στις ετικέτες Label1 η συντεταγμένη X, Label2 η συντεταγμένη Y.

Ο κώδικας της FMain είναι ο εξής:

```
1  ' Gambas class file
2
3  Public Sub Form_MouseMove()
4
5    Label1.Text = "Mouse X:" & Mouse.X
6    Label2.Text = "Mouse Y:" & Mouse.Y
7
8  End
```

Έξοδος:

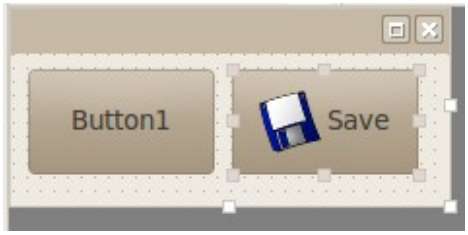


Ανάλυση:
Στην γραμμή 3 έχουμε την υπορουτίνα Form_MouseMove().
Η υπορουτίνα αυτή είναι το συμβάν που εγείρεται όταν ο δείκτης του ποντικιού κινείται και κρατιέται ένα κουμπί του ποντικιού πατημένο.
Στις γραμμές 5 και 6 οι ετικέτες Label1 και Label2 διαμορφώνουν το κείμενο τους λαμβάνοντας υπόψιν την θέση X, Y του δείκτη του ποντικιού.

Δουλεύοντας με τα κουμπιά, η κλάση Button.

Ένα κουμπί είναι ένα στοιχείο ελέγχου μιας φόρμας το οποίο υλοποιεί ένα πλήκτρο που μπορεί να πιεστεί.
Η κλάση Button κληρονομεί χαρακτηριστικά από την κλάση Control.
Ένα κουμπί μπορεί να έχει ένα κείμενο στην όψη του ή μια εικόνα ή και τα δύο.

Ένα κουμπί σχεδιάζεται σε μια φόρμα επιλέγοντας το από την εργαλειοθήκη και κρατώντας πατημένο το αριστερό πλήκτρο του ποντικιού σέρνουμε το ποντίκι στην φόρμα. Κατά το σύρσιμο βλέπουμε το περίγραμμα του αντικειμένου κουμπιού να σχηματίζεται πάνω στην φόρμα.



Εικόνα 84: Φόρμα σε προβολή σχεδίασης με δύο κουμπιά. Το ένα εκ των οποίων έχει εικονίδιο.

Εικόνα 85: Επιλέγοντας το στοιχείο ελέγχου "Κουμπί"

Μεταξύ των κουμπιών το ένα μπορεί να οριστεί ως Default και το άλλο ως Cancel.
Το κουμπί που θα οριστεί ως Default μπορεί να ενεργοποιηθεί πατώντας το Enter.
Το κουμπί που θα οριστεί ως Cancel μπορεί να ενεργοποιηθεί πατώντας το Escape.

Η κλάση αυτή μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hButton AS Button

hButton = NEW Button ( Parent AS Container ) [ AS "event name" ]
```

Όπου Parent είναι το αντικείμενο δοχέας που θα τοποθετηθεί το κουμπί.

Το αντικείμενο Button έχει τις εξής ιδιότητες, μεθόδους και συμβάντα.

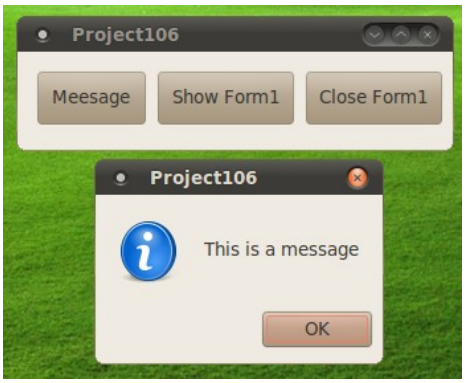
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Control		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Background Cursor Design Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show	DblClick Drag DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Button		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
AutoResize Border Cancel Caption Default Picture Text Value		Click

Παράδειγμα1
Το παράδειγμα αυτό είναι το Project106.
Σε αυτό το παράδειγμα έχουμε μια φόρμα με τρία κουμπιά.
Το πρώτο κουμπί εμφανίζει ένα μήνυμα.
Το δεύτερο κουμπί εμφανίζει μια άλλη φόρμα
Το τρίτο κουμπί κλείνει την φόρμα που έχει εμφανιστεί.

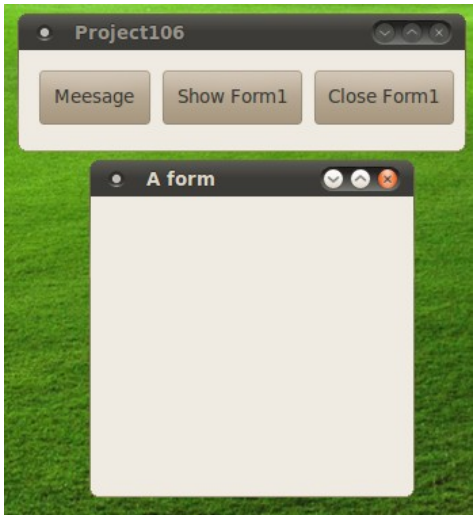
Ο κώδικας της FMain είναι ο εξής:

```
1   ' Gambas class file
2
3   PUBLIC SUB Form_Open()
4
5     ME.Center
6
7   END
8
9   PUBLIC SUB Button1_Click()
10
11    Message.Info("This is a message")
12
13  END
14
15  PUBLIC SUB Button2_Click()
16
17    Form1.Show
18
19  END
20
21  PUBLIC SUB Button3_Click()
22
23    Form1.Close
24
25  END
```

Έξοδος:



Εικόνα 86: Πατώντας το κουμπί Message



Εικόνα 87: Πατώντας το κουμπί Show Form1

Ανάλυση:
Υπάρχουν τέσσερις ρουτίνες.
Η PUBLIC SUB Form_Open(),
η PUBLIC SUB Button1_Click(),
η PUBLIC SUB Button2_Click(),
και η PUBLIC SUB Button3_Click().

Η PUBLIC SUB Form_Open() καλείται όταν εμφανίζεται η φόρμα FMain.
Στην γραμμή 5 η FMain τοποθετείται στο κέντρο της οθόνης.
Όταν πατηθεί το πρώτο κουμπί καλείται η PUBLIC SUB Button1_Click().
Η γραμμή 11 εμφανίζει ένα μήνυμα στην οθόνη.
Όταν πατηθεί το δεύτερο κουμπί καλείται η PUBLIC SUB Button2_Click().
Η γραμμή 17 εμφανίζει την φόρμα Form1.
Όταν πατηθεί το τρίτο κουμπί καλείται η PUBLIC SUB Button3_Click().
Η γραμμή 23 κλείνει την φόρμα Form1.

Δουλεύοντας με κουτιά κειμένου, η κλάση TextBox

Η κλάση υλοποιεί ένα πλαίσιο κειμένου μιας γραμμής.
Η κλάση κληρονομεί χαρακτηριστικά από την κλάση Control.
Η κλάση αυτή μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hTextBox AS TextBox

hTextBox = NEW TextBox ( Parent AS Container ) [ AS "event name" ]
```

Παρακάτω αναφέρονται οι ιδιότητες, μέθοδοι και συμβάντα του αντικειμένου TextBox.

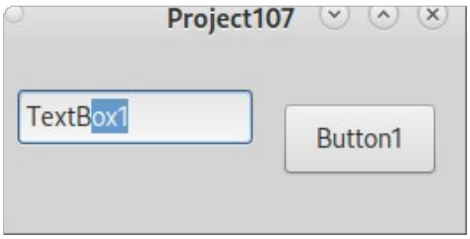
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Control		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Background Cursor Design Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show	DblClick Drag DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Textbox		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Alignment Border Length MaxLength Password Pos ReadOnly Selected Selection Text	Clear Insert Select SelectAll Unselect	Activate Change

Εφαρμογή 1
Στο παράδειγμα Project107, έχουμε ένα TextBox στο οποίο επιλέγουμε μέρος του κειμένου. Στην συνέχεια πατάμε ένα κουμπί Button1 και εμφανίζουμε στην stdout τις πληροφορίες για το επιλεγμένο κείμενο.

Ο κώδικας της FMain είναι ο εξής:

```
1   ' Gambas class file
2
3   PUBLIC SUB Form_Open()
4
5   ME.Center
6
7   END
8
9   PUBLIC SUB Button1_Click()
10
11  PRINT "Text: ";; TextBox1.Selection.Text
12  PRINT "Length";; TextBox1.Selection.Length
13  PRINT "Pos:";; TextBox1.Selection.Pos
14  TextBox1.Selection.Hide
15
16  END
```

Έξοδος:




```
Text:  ox1
Length -3
Pos: 5
```

Ανάλυση:
Έχουμε δύο υπορουτίνες τις PUBLIC SUB Form_Open(), PUBLIC SUB Button1_Click(). Στην πρώτη υπορουτίνα τοποθετούμε την φόρμα FMain στο κέντρο της οθόνης με την πρόταση ME.Center.

Στην δεύτερη υπορουτίνα έχουμε το συμβάν Click του κουμπιού Button1.
Εδώ εμφανίζουμε στην stdout το επιλεγμένο κείμενο
PRINT "Text: ";; TextBox1.Selection.Text

εμφανίζουμε το μήκος του επιλεγμένου κειμένου

PRINT "Length";; TextBox1.Selection.Length

εμφανίζουμε την θέση εκκίνησης της επιλογής
PRINT "Pos:";; TextBox1.Selection.Pos

και τέλος κάνουμε αποεπιλογή του κειμένου
TextBox1.Selection.Hide

Εφαρμογή 2
Στο παράδειγμα Project108, έχουμε δύο TextBox στα οποία εισάγουμε αριθμούς. Στην συνέχεια έχουμε 4 κουμπιά με τις 4 βασικές μαθηματικές πράξεις, πρόσθεση, αφαίρεση, πολλαπλασιασμός και διαίρεση.

Όταν πατάμε ένα κουμπί εκτελείται μια πράξη από αυτές μεταξύ των δύο αριθμών και το αποτέλεσμα εμφανίζεται σε ένα πέμπτο TextBox

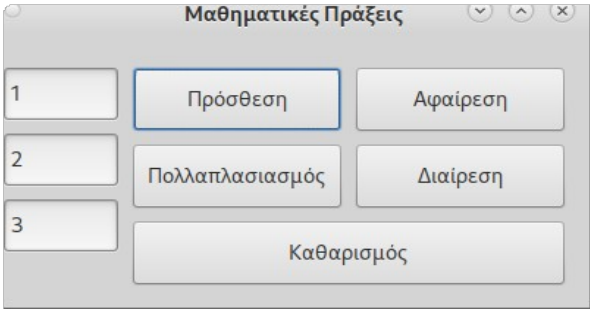
Τέλος έχουμε ένα κουμπί καθαρισμού των τιμών για να μπορούμε εύκολα να κάνουμε νέες πράξεις.

Ο κώδικας της FMain είναι ο εξής:

```
1   ' Gambas class file
2
3   Public Sub Form_Open()
4
5     Me.Center
6
7   End
8
9   Public Sub txtVal1_Change()
10
11    If Not IsNull(txtVal1.Text) And Not IsNumber(txtVal1.Text)
    Then
12    MessageNotNumber()
13    CorrectValue(txtVal1)
14    Endif
15
16  End
17
18  Public Sub txtVal2_Change()
19
20    If Not IsNull(txtVal2.Text) And Not IsNumber(txtVal2.Text)
    Then
21    MessageNotNumber()
22    CorrectValue(txtVal2)
23    Endif
24
25  End
26
27  Public Sub MessageNotNumber()
28
29    Message.Error("Παρακαλώ πληκτρολογείστε μόνο αριθμούς")
30
```

```
31  End
32
33  Public Sub CorrectValue(objTextBox As Object)
34
35     objTextBox.Text = Mid(objTextBox.Text, 1,
    Len(objTextBox.Text) - 1)
36
37  End
38
39  Public Sub btnClear_Click()
40
41    txtVal1.Clear
42    txtVal2.Clear
43    txtResult.Clear
44
45  End
46
47  Public Sub btnSum_Click()
48
49    txtResult.Text = Val(txtVal1.Text) + Val(txtVal2.Text)
50
51  End
52
53  Public Sub btnAbs_Click()
54
55      txtResult.Text = Val(txtVal1.Text) - Val(txtVal2.Text)
56
57  End
58
59  Public Sub btnMulti_Click()
60
61      txtResult.Text = Val(txtVal1.Text) * Val(txtVal2.Text)
62
63  End
64
65  Public Sub btnDiv_Click()
66
67      If Val(txtVal2.Text) <> 0 Then
68      txtResult.Text = Val(txtVal1.Text) / Val(txtVal2.Text)
69      Else
70      Message.Error("Διαίρεση με το 0. Παρακαλώ διορθώστε την
    δεύτερη τιμή.")
71      Endif
72
73  End
```

Έξοδος:



Εικόνα 88: Η εφαρμογή Project108

Ανάλυση:
Ξεκινάμε κλασσικά με το να τοποθετήσουμε την φόρμα στο κέντρο της οθόνης με την πρόταση
ME.Center στην υπορουτίνα PUBLIC SUB Form_Open().

Στην συνέχεια ακολουθούν οι υπορουτίνες ελέγχου τιμών των δύο TextBoxes.
Θέλουμε καθώς ο χρήστης πληκτρολογεί τιμές αν πληκτρολογήσει μια τιμή που δεν είναι κείμενο το πρόγραμμα να τον ενημερώσει για το λάθος του και να διορθώσει την τιμή.
Για αυτό τον σκοπό ελέγχουμε τις πληκτρολογημένες τιμές την στιγμή που εισάγονται στο TextBox.
Αυτό γίνεται στις υπορουτίνες PUBLIC SUB txtVal1_Change() και PUBLIC SUB txtVal2_Change().

Καθώς λοιπόν ο χρήστης εισάγει τιμές από το πληκτρολόγιο ελέγχουμε αν η τιμή του TextBox είναι αριθμός και αν είναι κενό το TextBox χωρίς τιμή.
Αυτό γίνεται με την πρόταση IF NOT IsNull(txtVal1.Text) AND NOT IsNumber(Val(txtVal1.Text)) THEN .

Όταν λοιπόν η τιμή δεν είναι αριθμός και κενή τιμή καλούμε τις υπορουτίνες
MessageNotNumber()
CorrectValue(txtVal1)

Η πρώτη εμφανίζει ένα μήνυμα λάθους
Message.Error("Παρακαλώ πληκτρολογείστε μόνο αριθμούς")

Η δεύτερη λαμβάνει σαν όρισμα το αντικείμενο txtVal1 ή txtVal2 και αφαιρεί το τελευταίο αλφαρηθμικό από την τιμή Text του TextBox. Το τελευταίο αλφαριθμητικό είναι αυτό που ο χρήστης εισήγαγε και έθεσε την τιμή του TextBox ως μη αριθμό. Έτσι αν αφαιρεθεί η τιμή που θα μείνει θα είναι ένας αριθμός.
Αυτό γίνεται με την έκφραση objTextBox.Text = Mid(objTextBox.Text, 1, Len(objTextBox.Text) – 1).

Η πρόταση objTextBox.Text αντιστοιχεί στις προτάσεις txtVal1.Text και txtVal2.Text όταν στην υπορουτίνα PUBLIC SUB CorrectValue(objTextBox AS Object) το όρισμα objTextBox είναι το αντικείμενο TextBox.

Η έκφραση Mid(objTextBox.Text, 1, Len(objTextBox.Text) – 1) επιστρέφει το κείμενο objTextBox.Text αφαιρώντας τον τελευταίο χαρακτήρα.
Πιο αναλυτικά, η Mid(objTextBox.Text, 1, Len(objTextBox.Text) – 1) λέει πάρε το κείμενο objTextBox.Text και επέστρεψε από αυτό το κείμενο που ξεκινά από το χαρακτήρα 1 και έχει μήκος όσο το μήκος του objTextBox.Text μείων 1 χαρακτήρα.

Mid(objTextBox.Text, 1, Len(objTextBox.Text) – 1)	
objTextBox.Text	Το κείμενο εισαγωγής στην συνάρτηση Mid
1	Ο χαρακτήρας εκκίνησης
Len(objTextBox.Text) – 1	Το μήκος του κειμένου

Στην συνέχεια η υπορουτίνα PUBLIC SUB btnClear_Click() είναι το συμβάν Click του κουμπιού btnClear το οποίο καθαρίζει τις τιμές από όλα τα TextBoxes.

Στην συνέχεια ακολουθούν οι υπορουτίνες συμβάντων κλικ των κουμπιών btnSum, btnAbs, btnMulti, btnDiv.

Σε αυτές τις υπορουτίνες αντίστοιχα γίνεται η πρόσθεση,
txtResult.Text = Val(txtVal1.Text) + Val(txtVal2.Text)

η αφαίρεση,
txtResult.Text = Val(txtVal1.Text) – Val(txtVal2.Text)

ο πολλαπλασιασμός,
txtResult.Text = Val(txtVal1.Text) * Val(txtVal2.Text)

και τέλος η διαίρεση.

Στην διαίρεση πρέπει να ελέγξουμε την τιμή txtVal2.Text να μην είναι μηδενική.
Αν είναι εμφανίζουμε ένα μήνυμα λάθους.

Αυτό γίνεται στον κώδικα

```
IF Val(txtVal2.Text) <> 0 THEN
txtResult.Text = Val(txtVal1.Text) / Val(txtVal2.Text)
ELSE
Message.Error("Διαίρεση με το 0. Παρακαλώ διορθώστε την δεύτερη τιμή.")
ENDIF
```

Σε αυτό τον κώδικα ελέγχουμε αν η τιμή του txtVal2.Text είναι διάφορη του 0 κάνουμε την διαίρεση.
Αλλιώς εμφανίζουμε ένα μήνυμα λάθους.

Εφαρμογή 3
Στο παράδειγμα Project109, έχουμε δύο φόρμες. Την FMain και την Form1.
Στην FMain έχουμε δύο TextBoxes ένα για το username και ένα για το password.
Επίσης έχουμε δύο κουμπιά ένα για τον έλεγχο του κωδικού και ονόματος και ένα για να καθαρίζει τα TextBoxes.

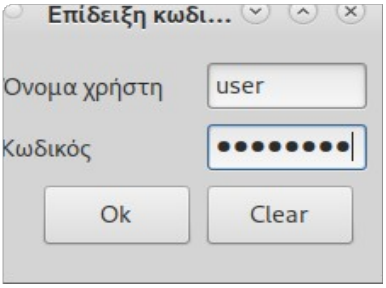
Επίσης έχουμε θέσει στα δύο TextBoxes την ιδιότητα MaxLength σε 8 που σημαίνει ότι ο χρήστης μπορεί να πληκτρολογήσει το μέγιστο οκτώ χαρακτήρες.

Στο txtPassword έχουμε θέσει την ιδιότητα Password σε TRUE και έτσι κατά την πληκτρολόγηση το κείμενο θα εμφανίζεται ως αστεράκια.

Ο κώδικας της FMain είναι ο εξής:

```
1   ' Gambas class file
2
3   PUBLIC SUB Form_Open()
4
5     ME.Center
6
7   END
8
9   PUBLIC SUB btnClear_Click()
10
11    txtUser.Clear
12    txtPassword.Clear
13
14  END
15
16  PUBLIC SUB btnOk_Click()
17
18    IF txtUser.Text = "username" AND txtPassword.Text =
    "password" THEN
19     Form1.Show
20    ELSE
21      Message.Error("Λάθος!")
22    ENDIF
23
24  END
```

Έξοδος:



Εικόνα 89: Το Project109

Ανάλυση:
Ξεκινάμε τοποθετώντας τη FMain στο κέντρο της οθόνης όταν η φόρμα εμφανίζεται.

```
PUBLIC SUB Form_Open()

  ME.Center

END
```

Όταν πιέζεται το κουμπί btnClear καθαρίζουμε το κείμενο από τα TextBoxes.

```
PUBLIC SUB btnClear_Click()

  txtUser.Clear
  txtPassword.Clear

END
```

Όταν πατηθεί το κουμπί btnOk ελέγχουμε αν οι τιμές κειμένου των TextBoxes είναι αυτές που επιθυμούμε.
Αν οι τιμές του κειμένου είναι "username" και "password" αντίστοιχα για το κείμενο στα TextBoxes txtUser και txtPassword τότε εμφανίζουμε την φόρμα Form1. Αλλιώς εμφανίζουμε ένα μήνυμα λάθους.

```
PUBLIC SUB btnOk_Click()

  IF txtUser.Text = "username" AND txtPassword.Text = "password" THEN
   Form1.Show
  ELSE
    Message.Error("Λάθος!")
  ENDIF

END
```

Δουλεύοντας με κουτιά επιλογής, η κλάση CheckBox

Η κλάση υλοποιεί ένα πλαίσιο επιλογής που μπορεί να επιλέγεται ή όχι ή βρίσκεται στην κατάσταση μη αλλαγής. Αυτή η τρίτη κατάσταση μπορεί να επιλεχθεί όταν έχει οριστεί η ιδιότητα Tristate.
Η κλάση κληρονομεί χαρακτηριστικά από την κλάση Control.
Η κλάση αυτή μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hCheckBox AS CheckBox

hCheckBox = NEW CheckBox ( Parent AS Container ) [ AS "event name" ]
```

Παρακάτω αναφέρονται οι ιδιότητες, μέθοδοι και συμβάντα του αντικειμένου CheckBox.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Control		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Background Cursor Design Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show	DblClick Drag DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Checkbox		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Autoresize Caption Text Tristate Value		Click
Σταθερές		
False None True		

Εφαρμογή 1
Στο παράδειγμα Project110, ο χρήστης κάνει κλικ σε ένα πλαίσιο επιλογής και εμφανίζεται ένα μήνυμα που πληροφορεί την κατάσταση του πλαισίου επιλογής. Στην φόρμα FMain υπάρχουν 4 πλαίσια επιλογής με ενεργοποιημένη την ιδιότητα Tristate.

Ο κώδικας της FMain είναι ο εξής:

```
1   ' Gambas class file
2
3   PUBLIC SUB Form_Open()
4
5   ME.Center
6
7   END
8
9   PUBLIC SUB Status(sName AS String, iState AS Integer)
10
11    DIM sState AS String
12
13    IF iState = -1 THEN sState = "On"
14    IF iState = 0 THEN sState = "Off"
15    IF iState = 1 THEN sState = "Disable"
16
17    Message.Info(sName & " is " & sState)
18
19  END
20
21  PUBLIC SUB CheckBox1_Click()
22
23    Status(CheckBox1.Text, CheckBox1.Value)
24
25  END
26
27  PUBLIC SUB CheckBox2_Click()
28
29    Status(CheckBox2.Text, CheckBox2.Value)
30
31  END
32
33  PUBLIC SUB CheckBox3_Click()
34
35    Status(CheckBox3.Text, CheckBox3.Value)
36
37  END
38
39  PUBLIC SUB CheckBox4_Click()
```

```
40
41    Status(CheckBox4.Text, CheckBox4.Value)
42
43  END
```

Έξοδος:

Εικόνα 90: Το Project110

Ανάλυση:
Η εφαρμογή ξεκινά τοποθετώντας την φόρμα FMain στο κέντρο της οθόνης όταν εμφανίζεται η φόρμα.

```
PUBLIC SUB Form_Open()

ME.Center

END
```

Στην γραμμή 9 έχουμε δημιουργήσει την ρουτίνα PUBLIC SUB Status(sName AS String, iState AS Integer).

Η ρουτίνα αυτή δέχεται δύο παραμέτρους ένα αλφαριθμητικό sName το οποίο είναι το όνομα του CheckBox που θα γίνεται κλικ και την κατάσταση του Checkbox ως ακέραιο που αποθηκεύεται στην μεταβλητή iState.

Στην γραμμή 11 δηλώνουμε μια μεταβλητή αλφαριθμητική.

```
DIM sState AS String
```

Στις γραμμές 12 με 15 εκχωρούμε την τιμή στην sState ανάλογα την τιμή που έχει η iState.

```
IF iState = -1 THEN sState = "On"
```

```
IF iState = 0 THEN sState = "Off"
IF iState = 1 THEN sState = "Disable"
```

Τέλος στην γραμμή 17 εμφανίζουμε ένα μήνυμα που πληροφορεί ότι το συγκεκριμένο checkbox έχει την συγκεκριμένη κατάσταση.

```
Message.Info(sName & " is " & sState)
```

Ακολουθούν τα συμβάντα κλικ των checkboxes.

```
PUBLIC SUB CheckBox1_Click()

  Status(CheckBox1.Text, CheckBox1.Value)

END
```

Σε κάθε ένα από αυτά καλούμε την ρουτίνα Status αφού περάσουμε τις παραμέτρους κειμένου και τιμής του checkbox.

Οι ρουτίνες των συμβάντων αυτών έχουν την ίδια μορφή.

Δουλεύοντας με πλαίσια, η κλάση Frame

Η κλάση υλοποιεί ένα πλαίσιο που μπορεί να χρησιμοποιηθεί ως δοχέας και να τοποθετηθούν σε αυτό άλλα στοιχεία ελέγχου που θα αποτελούν μια ομάδα.
Η κλάση κληρονομεί χαρακτηριστικά από την κλάση Container.
Η κλάση αυτή μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hFrame AS Frame

hFrame = NEW Frame ( Parent AS Container ) [ AS "event name" ]
```

Παρακάτω αναφέρονται οι ιδιότητες, μέθοδοι και συμβάντα του αντικειμένου CheckBox.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Container		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Background Children ClientH ClientHeight ClientW ClientWidth ClientX ClientY Cursor Design Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Find Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show	Arrange BeforeArrange DblClick Drag DragMove Drop Enter GotFocus Insert KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Frame		
Ιδιότητες (Properties)		
Caption Text Title		

Τα πλαίσια ομαδοποιούν τα αντικείμενα που βρίσκονται πάνω τους. Έτσι σε αυτά μπορούμε να τοποθετήσουμε αντικείμενα ραδιοπλήκτρων όπως θα δούμε στην συνέχεια που θα λειτουργούν ως μία ομάδα.

Δουλεύοντας με ραδιοπλήκτρα, η κλάση RadioButton

Η κλάση υλοποιεί ένα ραδιοπλήκτρο ή αλλιώς όπως ονομάζεται κουμπί αλληλοαποκλειόμενων επιλογών.
Η κλάση κληρονομεί χαρακτηριστικά από την κλάση Control.
Η κλάση αυτή μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hRadioButton AS RadioButton

hRadioButton = NEW RadioButton ( Parent AS Container ) [ AS "event name" ]
```

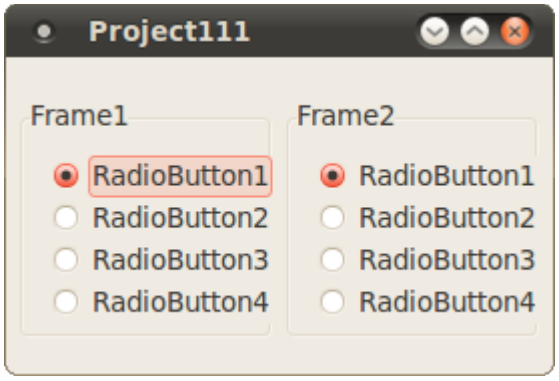
Παρακάτω αναφέρονται οι ιδιότητες, μέθοδοι και συμβάντα του αντικειμένου RadioButton.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Control		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Background Cursor Design Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show	DblClick Drag DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Radiobutton		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Autoresize Caption Text Value		Click

Το αντικείμενο RadioButton μοιάζει με το αντικείμενο Checkbox.
Η διαφορά τους είναι ότι μια ομάδα από αντικείμενα RadioButtons προσφέρουν μια μοναδική επιλογή για αυτό τα αντικείμενα ονομάζονται και κουμπιά αλληλοαποκλειόμενων επιλογών.
Αυτό σημαίνει ότι αν έχουμε για παράδειγμα 3 RadioButton ο χρήστης όταν θα επιλέγει το ένα η τιμή Value θα τεθεί TRUE αλλά όταν επιλέξει κάποιο άλλο το πρώτο radiobutton θα αποκτήσει την τιμή Value=False και το επιλεγμένο την τιμή Value=True.
Έτσι κάθε φορά που θα επιλέγεται ένα radiobutton τα υπόλοιπα θα αποκλείονται.

Εφαρμογή 1
Στο παράδειγμα Project111, υπάρχουν στην FMain δύο πλαίσια (Frames) με τέσσερα RadioButtons.
Κάθε πλαίσιο είναι ένας δοχέας για τα τέσσερα ραδιοπλήκτρα και έτσι αυτά λειτουργούν σαν δύο διαφορετικές ομάδες. Οι επιλογές στην μία ομάδα ραδιοπλήκτρων δεν επηρεάζει την άλλη ομάδα.

Έξοδος:



Εικόνα 91: Το Project111, δύο ομάδες ραδιοπλήκτρων

Αν είχαμε τα οκτώ ραδιοπλήκτρα στην φόρμα τοποθετημένα τότε αυτά θα είχαν δοχέα την φόρμα FMain και θα λειτουργούσαν σαν μια ομάδα οκτώ ραδιοπλήκτρων.
Επομένως για να ορίσουμε ομάδες ραδιοπλήκτρων σε μια φόρμα θα πρέπει να χρησιμοποιούμε έναν άλλο δοχέα όπως ένα πλαίσιο.

Δουλεύοντας με ετικέτες, η κλάση Label

Η κλάση υλοποιεί μια ετικέτα η οποία δείχνει ένα κείμενο.
Η κλάση κληρονομεί χαρακτηριστικά από την κλάση Control.
Η κλάση αυτή μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hLabel AS Label

hLabel = NEW Label ( Parent AS Container ) [ AS "event name" ]
```

Παρακάτω αναφέρονται οι ιδιότητες, μέθοδοι και συμβάντα του αντικειμένου Label.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Control		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Background Cursor Design Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show	DblClick Drag DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Label		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Alignment AutoResize Border Caption Padding Text Transparent	Adjust	

Ιδιότητες (Properties)

Alignment

```
PROPERTY Alignment AS Integer
```

Επιστρέφει ή θέτει έναν ακέραιο που αναφέρεται σε μια σταθερά που ορίζει την στοίχιση του κειμένου.
Οι σταθερές που χρησιμοποιούνται είναι από την κλάση Align και είναι οι εξής:

Σταθερά	Τιμή	Περιγραφή

Normal	0	Τοποθετεί το κείμενο στο κέντρο κάθετα και οριζόντια κατά την φορά γραφής.
Left	1	Τοποθετεί το κείμενο στα δεξιά και στο κέντρο κάθετα.
Right	2	Τοποθετεί το κείμενο στα αριστερά και στο κέντρο κάθετα.
Center	3	Τοποθετεί το κείμενο στο κέντρο της επιφάνειας τόσο κάθετα όσο και οριζόντια.
Justify	4	Τοποθετεί το κείμενο στα δεξιά και στο κέντρο κάθετα.
TopNormal	16	Τοποθετεί το κείμενο κατά την διάταξη γραφής της γλώσσας και στο πάνω μέρος της επιφάνειας.
TopLeft	17	Τοποθετεί το κείμενο στα αριστερά και στο πάνω μέρος της επιφάνειας.
TopRight	18	Τοποθετεί το κείμενο στα δεξιά και στο πάνω μέρος της επιφάνειας.
Top	19	Τοποθετεί το κείμενο στο κέντρο και στο πάνω μέρος της επιφάνειας.
BottomNormal	32	Τοποθετεί το κείμενο κατά την διάταξη γραφής της γλώσσας και στο κάτω μέρος της επιφάνειας.
BottomLeft	33	Τοποθετεί το κείμενο στα αριστερά και στο κάτω μέρος της επιφάνειας.
BottomRight	34	Τοποθετεί το κείμενο στα δεξιά και στο κάτω μέρος της επιφάνειας.
Bottom	35	Τοποθετεί το κείμενο στο κέντρο και στο κάτω μέρος της επιφάνειας.

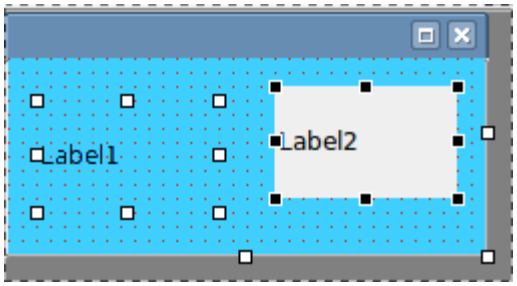
Το Project112 δείχνει αυτές τις στοιχίσεις του αντικειμένου Label. Σε αυτό το παράδειγμα έχουμε θέσει την ιδιότητα Border των ετικετών στην τιμή Plain για να φανεί πιο καλά η στοίχιση του κειμένου στα όρια της ετικέτας.

Εικόνα 92: Οι στοίχιση του κειμένου των ετικετών

Transparent

```
PROPERTY Transparent AS Boolean
```

Επιστρέφει ή θέτει έναν Boolean που ορίζει όταν είναι TRUE ότι η επιφάνεια του αντικειμένου θα είναι διάφανη. Όταν η επιφάνεια είναι διάφανη τότε φαίνεται το αντικείμενο δοχέας μέσα από το αντικείμενο Label.
Παρακάτω φαίνονται δύο ετικέτες κατά την σχεδίαση, εκ των οποίων η πρώτη είναι διάφανη ενώ η δεύτερη δεν είναι.



Εικόνα 93: Η ιδιότητα Transparent στο αντικείμενο Label

Δουλεύοντας με περιοχές κειμένου, η κλάση TextArea

Η κλάση υλοποιεί ένα πλαίσιο κειμένου με πολλές σειρές.
Η κλάση κληρονομεί χαρακτηριστικά από την κλάση Control.
Η κλάση αυτή μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hTextArea AS TextArea

hTextArea = NEW TextArea ( Parent AS Container ) [ AS "event name" ]
```

Παρακάτω αναφέρονται οι ιδιότητες, μέθοδοι και συμβάντα του αντικειμένου TextArea.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Control		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Background Cursor Design Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show	DblClick Drag DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης TextArea		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**

Border Column Length Line Pos ReadOnly ScrollBar Selected Selection Text Wrap	Clear Copy Cut EnsureVisible Insert Paste Redo Select SelectAll ToColumn ToLine ToPos Undo Unselect	Change Cursor

Εφαρμογή 1
Σε αυτό το παράδειγμα θα δημιουργήσουμε μια μικρή εφαρμογή επεξεργαστή κειμένου.
Σε αυτήν την εφαρμογή θα αξιοποιήσουμε τις περισσότερες ιδιότητες, μεθόδους και συμβάντα από την κλάση περιοχής κειμένου. Επίσης θα χρησιμοποιήσουμε και άλλες κλάσεις όπως κουμπιά κτλ.
Επειδή το Project είναι αρκετά μεγάλο σε έκταση η περιγραφή του θα γίνει βήμα βήμα καθώς σχεδιάζεται και υλοποιείται και όχι όπως τα προηγούμενα παραδείγματα όπου παρατίθονταν όλος ο κώδικας και εξηγούνταν στην συνέχεια. Σε αυτό το παράδειγμα ο κώδικας θα γράφεται και θα εξηγείται σε στάδια κατά την δημιουργία του έργου.

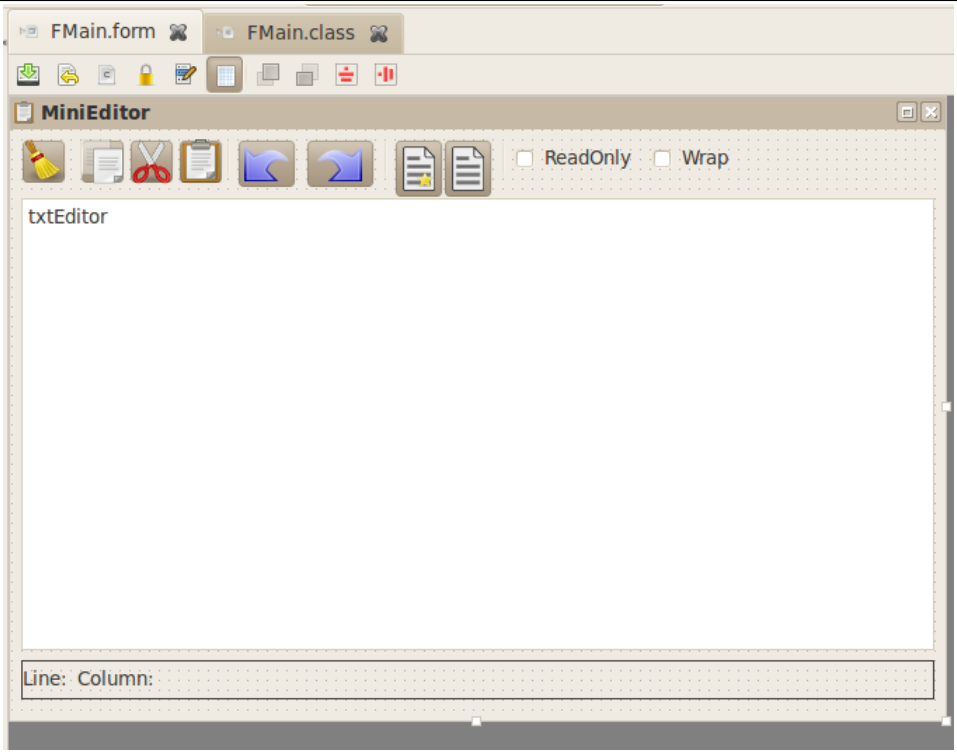
Σχεδιασμός της γραφικής διεπαφής.
Η εφαρμογή που θα αναπτύξουμε θα ονομάζεται MiniEditor και αυτό θα είναι και το όνομα της φόρμας FMain.

Θα δημιουργήσουμε 8 κουμπιά με αντίστοιχα εικονίδια για τις λειτουργίες Clear, Copy, Cut, Paste, Redo, Undo, SelectAll, Unselect. Τα εικονίδια μπορούμε να τα βρούμε στο διαδίκτυο από την διεύθυνση http://www.openclipart.org/.

Θέλουμε να μπορεί ο χρήστης να ορίζει αν το κείμενο θα είναι μόνο για ανάγνωση ή όχι και αν το κείμενο θα αναδιπλώνεται. Αυτήν την λειτουργία θα την υλοποιήσουμε με δύο Checkbox που όταν θα είναι επιλεγμένα το κείμενο θα είναι μόνο για ανάγνωση και με το δεύτερο το κείμενο θα αναδιπλώνεται.

Τέλος θέλουμε να εμφανίζεται σε ένα Label στο κάτω μέρος της φόρμας η θέση του δείκτη κειμένου στην περιοχή κειμένου. Στην ετικέτα θα εμφανίζουμε σε ποια κολόνα και σε ποια σειρά βρίσκεται ο δρομέας.

Έτσι μπορούμε να σχεδιάσουμε την φόρμα μας όπως παρακάτω. Μεταξύ των κουμπιών μπορούμε να βάλουμε το αντικείμενο Separator που ξεχωρίζει τις εντολές ομαδοποιημένες.



Εικόνα 94: Ο σχεδιασμός της εφαρμογής MiniEditor

Παρακάτω φαίνονται τα ονόματα των αντικειμένων της φόρμας.

Όνομα	**Αντικείμενο**
Fmain	Η φόρμα
btnClear	Το κουμπί καθαρισμού
btnCopy	Το κουμπί αντιγραφής
btnCut	Το κουμπί αποκοπής
btnPaste	Το κουμπί επικόλλησης
btnUndo	Το κουμπί αναίρεσης
btnRedo	Το κουμπί επανάλληψης
btnSellectAll	Το κουμπί επιλλογής όλων
btnUnsellect	Το κουμπί αποεπιλογής
chkReadOnly	Η επιλογή μόνο για ανάγνωση
chkWrap	Η επιλογή αναδίπλωσης
txtEditor	Η περιοχή κειμένου
lblPosition	Η ένδειξη θέσης του δείκτη κειμένου

Συγγραφή κώδικα

Αφού σχεδιάσαμε την εφαρμογή μας τώρα θα πρέπει να προγραμματίσουμε κάθε στοιχείο της.

Για το κουμπί καθαρισμού έχουμε

```
1  PUBLIC SUB btnClear_Click()
2
3    txtEditor.Clear
4
5  END
```

Για το κουμπί αντιγραφής έχουμε

```
1  PUBLIC SUB btnCopy_Click()
2
3    txtEditor.Copy
4
5  END
```

Για το κουμπί αποκοπής έχουμε

```
1  PUBLIC SUB btnCut_Click()
2
3    txtEditor.Cut
4
5  END
```

Για το κουμπί επικόλλησης έχουμε

```
1  PUBLIC SUB btnPaste_Click()
2
3    txtEditor.Paste
4
5  END
```

Για το κουμπί αναίρεσης έχουμε

```
1  PUBLIC SUB btnUndo_Click()
2
3    txtEditor.Undo
4
5  END
```

Για το κουμπί επανάληψης έχουμε

```
1  PUBLIC SUB btnRedo_Click()
2
3    txtEditor.Redo
4
5  END
```

Σημείωση, στο γραφικό περιβάλλον GTK+ οι μέθοδοι Undo και Redo δεν έχουν υλοποιηθεί ακόμη.

Για το κουμπί επιλογής όλων έχουμε

```
1  PUBLIC SUB btnSellectAll_Click()
2
3    txtEditor.SelectAll
4
5  END
```

Για το κουμπί αποεπιλογής έχουμε

```
1  PUBLIC SUB btnUnselect_Click()
2
3    txtEditor.Unselect
4
5  END
```

Για την επιλογή μόνο για ανάγνωση έχουμε έχουμε

```
1  PUBLIC SUB chkReadOnly_Click()
2
3    IF chkReadOnly.Value = TRUE
4     txtEditor.ReadOnly = TRUE
5    ELSE
6     txtEditor.ReadOnly = FALSE
7    ENDIF
8
9  END
```

Για την επιλογή αναδίπλωσης έχουμε έχουμε

```
PUBLIC SUB chkWrap_Click()

  IF chkWrap.Value = TRUE
   txtEditor.Wrap = TRUE
  ELSE
   txtEditor.Wrap = FALSE
  ENDIF

END
```

Για να εμφανίσουμε την θέση του δείκτη του κειμένου στο κάτω μέρος της φόρμας στην ετικέτα ένδειξης δημιουργούμε μια ρουτίνα η οποία θέτει το κείμενο της ετικέτας σε σχέση με την θέση του δείκτη κειμένου.

Η ρουτίνα λήψης της θέσης του δείκτη είναι η εξής

```
PUBLIC SUB GetPosition()

  lblPosition.Text = " Line: " & txtEditor.Line & " Column:
" & txtEditor.Column

END
```

Την ρουτίνα αυτή θα την καλούμε κάθε φορά που ο χρήστης αλλάζει θέση τον δείκτη κειμένου, δλδ κατά το συμβάν _Cursor() και όταν ο δείκτης πατάει ένα πλήκτρο δλδ κατά το συμβάν _KeyPress().

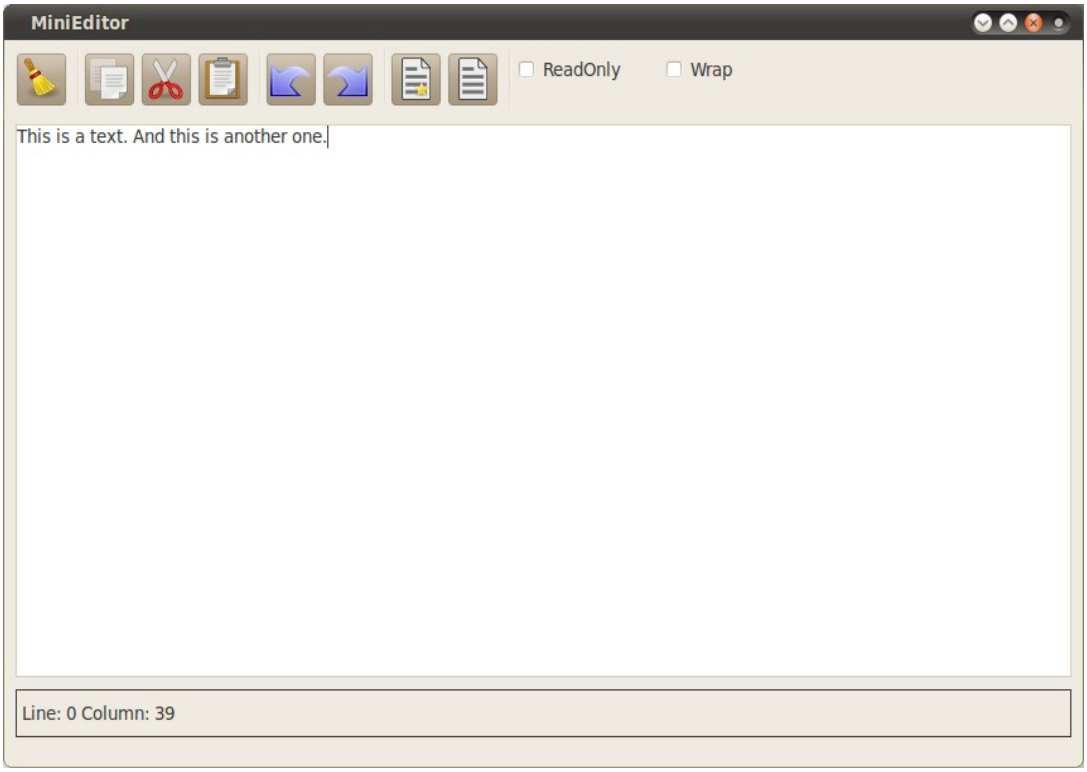
Έτσι έχουμε,

```
PUBLIC SUB txtEditor_Cursor()

  GetPosition()

END

PUBLIC SUB txtEditor_KeyPress()

  GetPosition()

END
```

Τέλος τοποθετούμε την φόρμα στο κέντρο της οθόνης όταν αυτή εμφανίζεται.

```
1  PUBLIC SUB Form_Open()
2
3  ME.Center
4
5  END
```

Η εφαρμογή μας είναι έτοιμη και μπορούμε να την τρέξουμε πατώντας το κουμπί εκτέλεσης.



Εικόνα 95: Η εφαρμογή MiniEditor

Δουλεύοντας με κουμπιά εναλλαγής, η κλάση ToggleButton

Η κλάση υλοποιεί ένα κουμπί εναλλαγής. Ένα κουμπί εναλλαγής μπορεί να πατηθεί και να παραμείνει πατημένο ενώ όταν πατηθεί ξανά επαναφέρεται στην πρότερη του κατάσταση. Το κουμπί εναλλαγής δηλώνει δύο καταστάσεις, την κατάσταση “πατημένου κουμπιού” και την κατάσταση “μη πατημένου κουμπιού”.
Η κλάση κληρονομεί χαρακτηριστικά από την κλάση Control.
Η κλάση αυτή μπορεί να δημιουργηθεί.

Σύνταξη:

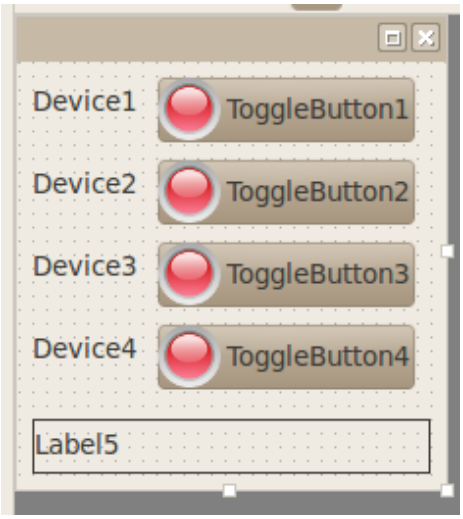
```
DIM hToggleButton AS ToggleButton

hToggleButton = NEW ToggleButton ( Parent AS Container ) [ AS "event name" ]
```

Παρακάτω αναφέρονται οι ιδιότητες, μέθοδοι και συμβάντα του αντικειμένου ToggleButton.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Control		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Background Cursor Design Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show	DblClick Drag DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης ToggleButton		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
AutoResize Border Caption Picture Radio Text Value		Click

Εφαρμογή 1
Το παράδειγμα αυτό είναι το Project114
Σε αυτό το παράδειγμα έχουμε 4 εικονικές συσκευές οι οποίες ενεργοποιούνται από 4 διακόπτες On/Off.
Στο κάτω μέρος της φόρμας υπάρχει ένδειξη της κατάστασης της συσκευής.
Κάθε κουμπί έχει ένα εικονίδιο LED που δείχνει αν η συσκευή είναι ενεργή ή απενεργοποιημένη.
Όταν η συσκευή είναι ενεργοποιημένη το εικονίδιο είναι ένα πράσινο LED.
Όταν η συσκευή είναι απενεργοποιημένη το εικονίδιο είναι ένα κόκκινο LED.

Η σχεδίαση της φόρμας είναι η παρακάτω:



Εικόνα 96: Ο σχεδιασμός του Project114

Κάθε φορά που πατάμε ένα κουμπί εναλλαγής αλλάζει το εικονίδιο του LED και εμφανίζεται το αντίστοιχο μήνυμα στην ετικέτα στο κάτω μέρος της φόρμας.

Για να μην επαναλάβουμε τον ίδιο κώδικα σε όλα τα κουμπιά εναλλαγής θα δημιουργήσουμε δύο ρουτίνες οι οποίες θα θέτουν το κατάλληλο εικονίδιο και θα εμφανίζουν το μήνυμα στην ετικέτα.

Ρουτίνα εναλλαγής του εικονιδίου LED.

```
1   PUBLIC SUB SetLed(bStatus AS Boolean, oButton AS Object)

2     IF bStatus = TRUE THEN oButton.Picture =
    Picture["Anonymous_led_circle_green.png"]
3     IF bStatus = FALSE THEN oButton.Picture =
    Picture["Anonymous_led_circle_red.png"]
4
5   END
```

Η ρουτίνα αυτή λαμβάνει δύο παραμέτρους. Έναν Boolean που είναι η τιμή Value του πλήκτρου εναλλαγής και το ίδιο το πλήκτρο εναλλαγής ως αντικείμενο.

Ελέγχουμε την τιμή bStatus, αν είναι TRUE ορίζουμε εικόνα στο αντικείμενο το πράσινο LED αλλιώς το κόκκινο LED.

Ρουτίνα εμφάνισης του μηνύματος στην ετικέτα

```
1   PUBLIC SUB SetStatus(bStatus AS Boolean, sDevice AS String)
2
3     DIM sStatus AS String
4
5     IF bStatus = TRUE THEN sStatus = "On"
6     IF bStatus = FALSE THEN sStatus = "Off"
7
8     Label5.Text = sDevice & " is " & sStatus
9
10  END
```

Η ρουτίνα αυτή δέχεται δύο παραμέτρους. Έναν Boolean και ένα String.
Ο Boolean είναι η τιμή Value του πλήκτρου εναλλαγής.
Το String είναι το όνομα της συσκευής.

Η ρουτίνα ορίζει μια μεταβλητή αλφαριθμητική την sString.
Στη μεταβλητή αυτή εκχωρείται, στις γραμμές 5,6, η τιμή On ή Off ανάλογα την τιμή που έχει η bStatus.

Αφού έχουμε ορίσει την sStatus στην γραμμή 8 εμφανίζουμε το μήνυμα, ποια συσκευή είναι σε ποια κατάσταση.

Κάθε κουμπί εναλλαγής έχει τον ακόλουθο κώδικα,

```
1   PUBLIC SUB ToggleButton1_Click()
2
3     SetStatus(ToggleButton1.Value, "Device1")
4     SetLed(ToggleButton1.Value, ToggleButton1)
5
6   END
7
8
9   PUBLIC SUB ToggleButton2_Click()
10
11    SetStatus(ToggleButton2.Value, "Device2")
12    SetLed(ToggleButton2.Value, ToggleButton2)
13
14  END
15
16  PUBLIC SUB ToggleButton3_Click()
17
18    SetStatus(ToggleButton3.Value, "Device3")
19    SetLed(ToggleButton3.Value, ToggleButton3)
20
21  END
22
23  PUBLIC SUB ToggleButton4_Click()
24
25    SetStatus(ToggleButton4.Value, "Device4")
26    SetLed(ToggleButton4.Value, ToggleButton4)
27
28  END
```

Το Project114 κατά την εκτέλεση είναι το εξής,

Εικόνα 97: Το Project114 σε κατάσταση εκτέλεσης

Δουλεύοντας με ετικέτες κειμένου, η κλάση TextLabel

Η κλάση υλοποιεί μία ετικέτα εμπλουτισμένου κειμένου. Το κείμενο έχει την μορφοποίηση της HTML (HyperText Markup Language).
Η κλάση κληρονομεί χαρακτηριστικά από την κλάση Control.
Η κλάση αυτή μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hTextLabel AS TextLabel

hTextLabel = NEW TextLabel ( Parent AS Container ) [ AS "event name" ]
```

Παρακάτω αναφέρονται οι ιδιότητες, μέθοδοι και συμβάντα του αντικειμένου TextLabel.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Control		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Background Cursor Design Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show	DblClick Drag DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης TextLabel		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Alignment AutoResize Border Caption Padding Text Transparent Wrap		Adjust

Εφαρμογή 1
Το Project115 είναι ένα παράδειγμα με ένα TextLabel.

Η τιμή της ιδιότητας Text είναι η εξής:

“<h1>This is a HTML text.</h1>

This <a>text</a> is in a TextLabel control!”

Στην HTML ότι συμπεριλαμβάνεται μεταξύ <h1>...</h1> είναι ένα κείμενο επικεφαλίδας νούμερο 1 (header1). Ότι συμπεριλαμβάνεται μεταξύ <a>...</a> είναι ένας σύνδεσμος, το κείμενο μεταξύ ... είναι πλάγιο (italics) και τέλος το κείμενο μεταξύ ... είναι έντονο (bold). Για περισσότερες λεπτομέρειες με την σύνταξη της HTML δείτε ένα εγχειρίδιο της HTML.

Το αποτέλεσμα αυτής της τιμής που εκχωρείται στην ιδιότητα Text του TexLabel είναι το εξής,

Εικόνα 98: Το Project115

Δουλεύοντας με κουτιά εικόνων, η κλάση PictureBox

Η κλάση υλοποιεί ένα κουτί εικόνας. Ένα κουτί εικόνας προβάλει μια εικόνα.
Η κλάση κληρονομεί χαρακτηριστικά από την κλάση Control.
Η κλάση αυτή μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hPictureBox AS PictureBox

hPictureBox = NEW PictureBox ( Parent AS Container ) [ AS "event name" ]
```

Παρακάτω αναφέρονται οι ιδιότητες, μέθοδοι και συμβάντα του αντικειμένου PictureBox.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Control		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Background Cursor Design Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show	DblClick Drag DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης PictureBox		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Alignment AutoResize Border Picture Stretch		

Σε ένα αντικείμενο Picturebox θέτουμε απευθείας την εικόνα που θέλουμε ορίζοντας την ιδιότητα Picture.
Αν θέλουμε η εικόνα να καταλάβει όλη την έκταση του αντικειμένου και αυτόματα να παραμορφωθεί κατά μήκος και πλάτος θέτουμε την ιδιότητα Stretch να έχει την τιμή TRUE.

Δουλεύοντας με εικόνες, η κλάση Picture

Τα δεδομένα ενός αντικειμένου Picture αποθηκεύονται στον διακομιστή οθόνης (display server) και όχι στην μνήμη της διεργασίας όπως γίνεται σε ένα αντικείμενο Image.

Αν και η κλάση δεν διαχειρίζεται πλήρως την κατάσταση διαφάνειας, κάθε εικόνα μπορεί να έχει μια μάσκα. Το χαρακτηριστικό αυτό μπορεί να υλοποιηθεί κατά την αρχικοποίηση του αντικειμένου ή κατά την εκχώρηση μια εικόνας στο αντικείμενο η οποία έχει ένα επίπεδο διαφάνειας όπως ένα αρχείο PNG.

Όταν σχεδιάζετε σε μια εικόνα που έχει μάσκα η εικόνα και η μάσκα τροποποιούνται αντίστοιχα.

Η κλάση μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hPicture AS Picture

hPicture = NEW Picture ( [ Width AS Integer, Height AS Integer,
Transparent AS Boolean ] )
```

Η κλάση Picture έχει τις ακόλουθες ιδιότητες και μεθόδους.

Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Στατικές Μέθοδοι (Static Methods)**
Depth H Height Image Transparent W Width	Clear Copy Fill Resize Save	Flush Load

Δουλεύοντας με εικόνες, η κλάση Image

Η κλάση υλοποιεί ένα αντικείμενο Image.
Τα δεδομένα ενός αντικειμένου Image αποθηκεύονται στην μνήμη της διεργασίας και όχι στον διακομιστή οθόνης (display server).
Για να πάρετε τα δεδομένα ενός pixel μπορείτε να διαχειριστείτε το αντικείμενο Image σαν έναν δυσδιάστατο πίνακα Image[x,y].
pixel = Image[x,y]

Η κλάση μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hImage AS Image

hImage = NEW Image ( [ Width AS Integer, Height AS Integer, Color
AS Integer, Format AS Integer ] )
```

Αν οριστεί η παράμετρος Color τότε δημιουργείται μια εικόνα γεμισμένη με το χρώμα αυτό.
Η παράμετρος Format μπορεί να έχει τις τιμές:

- **Standard**.
- **Premultiplied**.

Η παράμετρος **Standard** είναι μια σταθερά με τιμή 0.

CONST **Standard** AS Integer = 0

Κάθε pixel της εικόνας καταλαμβάνει 4 bytes στην μνήμη.

Το πρώτο byte αντιστοιχεί στην τιμή aplha. Η τιμή 0 καθιστά το pixel πλήρως διαφανές. Η τιμή 255 πλήρως αδιαφανές.

Το δεύτερο, τρίτο και τέταρτο byte αντιστοιχούν στις τιμές του Red, Green, Blue.

Η παράμετρος **Premultiplied** είναι μια σταθερά με τιμή 1.

CONST **Premultiplied** AS Integer = 1

Κάθε pixel της εικόνας καταλαμβάνει 4 bytes στην μνήμη.

Το πρώτο byte αντιστοιχεί στην τιμή aplha. Η τιμή 0 καθιστά το pixel πλήρως διαφανές. Η τιμή 255 πλήρως αδιαφανές.

Το δεύτερο, τρίτο και τέταρτο byte αντιστοιχούν στις τιμές του Red, Green, Blue πολλαπλασιασμένες από την τιμή alpha και διαιρούμενη από το 255.

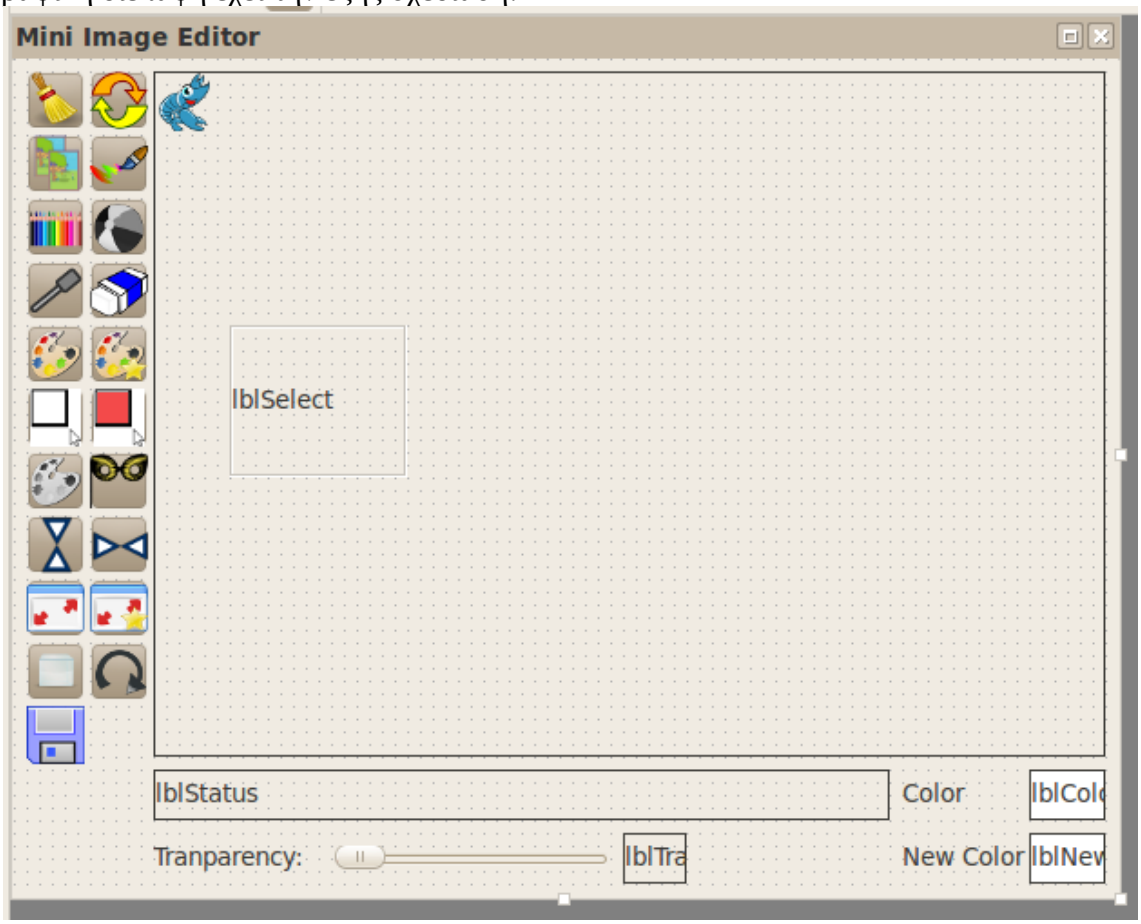
Η κλάση Image υλοποιεί ξανά το αντικείμενο Image του συστατικού gb.image

Η κλάση Image έχει τις ακόλουθες ιδιότητες και μεθόδους.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Image		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Data Depth Format H Height W Width	Clear Colorize Copy Desaturate Drawalpha Drawrect Erase Fill Fillrect Gray Mask Mirror Replace Resize Transparent	
Constants		
Premultiplied Standard		
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Image		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Picture	Draw Rotate Save Stretch	
Static methods		
Load		

Εφαρμογή 1
Το Project116 είναι ένα παράδειγμα με αντικείμενα εικόνας.
Στο παράδειγμα αυτό φορτώνουμε ένα αρχείο εικόνας σε ένα αντικείμενο Image, εφαρμόζουμε διάφορες ενέργειες σε αυτό και εμφανίζουμε το αποτέλεσμα σε ένα αντικείμενο Picture.
Μπορούμε επίσης αν θέλουμε να φορτώσουμε ξανά το αρχείο εικόνας για να αρχίσουμε από την αρχή ή να σώσουμε το αποτέλεσμα σε ένα άλλο αρχείο.

Η γραφική διεπαφή έχει την εξής σχεδίαση:



Εικόνα 99: Η διεπαφή του Project116

Στα αριστερά έχουμε Buttons που εκτελούν και από μια εργασία στην εικόνα.
Στο κέντρο έχουμε ένα αντικείμενο PictureBox και ένα Label που θα χρησιμοποιηθεί σαν περίγραμμα για ένδειξη της επιλογής μιας περιοχής από την εικόνα.
Από κάτω έχουμε ένα Label που εμφανίζει πληροφορίες για την επιλογή περιοχής της εικόνας.
Πιο κάτω έχουμε ένα Slidebar με μέγιστη τιμή το 255 και ελάχιστη το 0 για να θέσουμε την τιμή διαφάνειας για ένα χρώμα. Η τιμή διαφάνειας από το Sledebar εμφανίζεται δίπλα στο lblTransparency.
Στα δεξιά έχουμε 2 Label που που με εμφανίζουν τα χρώματα επιλογής και αντικατάστασης και τα οποία χρησιμοποιούνται από τις συναρτήσεις του αντικειμένου Image.

Το αντικείμενο Image δεν φαίνεται στην διεπαφή αλλά δημιουργείται και διαχειρίζεται από τον κώδικα της εφαρμογής.
Αφού εκτελέσουμε κάποιες συναρτήσεις επεξεργασίας της εικόνας στο αντικείμενο Image στην συνέχεια εμφανίζουμε το αποτέλεσμα στο αντικείμενο PictureBox.

Το Project έχει τις εξής φόρμες.

FColorChooser
FColorChooser2
FMain
FResize
FRotate
FSave
FStretch

Η φόρμα FColorChooser θέτει την τιμή του χρώματος που θα χρησιμοποιηθεί από τις συναρτήσεις Colorize, Fill, DrawRect, Erase, FillRect, Mask, Replace, Transparent.

Η φόρμα FColorChooser θέτει την τιμή του χρώματος που θα χρησιμοποιηθεί από την συνάρτηση Replace.
Η φόρμα FResize λαμβάνει τις τιμές Width, Height για την συνάρτηση Resize.
Η φόρμα FRotate λαμβάνει την τιμή γωνίας περιστροφής για την συνάρτηση Rotate.
Η φόρμα FSave εμφανίζει ένα FileChooser για την αποθήκευση του αρχείου.
Η φόρμα FStretch λαμβάνει τις τιμές Width, Height για την συνάρτηση Stretch.

Ας δούμε όμως τον κώδικα σε κάθε μία φόρμα.

<u>FMain</u>
Στο τμήμα των καθολικών δηλώσεων έχουμε τις δηλώσεις των μεταβλητών

```
Public hImage As Image
Private hCopyImage As Image
Private iStartX As Integer
Private iEndX As Integer
Private iStartY As Integer
Private iEndY As Integer
Public iColor As Integer = &FFFFFF&
Public iNewColor As Integer = &FFFFFF&
Private bPickColor As Boolean = False
Private cursorPickColor As New
Cursor((Picture.Load("ftcolorpicker.png")))
```

Με το που ανοίγει η φόρμα την τοποθετούμε στο κέντρο της οθόνης και καλούμε τις συναρτήσεις InitImage(), ShowImage().

```
14  Public Sub Form_Open()
15
16  Me.Center
17  InitImage()
18  ShowImage()
19
20  End
```

Η συνάρτηση InitImage() δημιουργεί ένα αντίγραφο του αρχείου εικόνας που θα χρησιμοποιηθεί έτσι ώστε αν ο χρήστης σώσει αυτό η εφαρμογή να μπορεί να φορτώνει το πρώτο αρχείο για κάθε επίδειξη των συναρτηεων του αντικειμένου Image που θα γίνεται. Πριν όμως δημιουργήσει το αντίγραφο αφαιρεί οποιοδήποτε παλιό αρχείο έχει δημιουργηθεί από προηγούμενη χρήση.
Αφού γίνει το αντίγραφο του αρχείου δημιουργούμε ένα αντικείμενο Image και φορτώνουμε το αρχείο σε αυτό.

```
23  Public Sub InitImage()
24
25  Try Kill Application.Path & "/nature_.png"
26  Copy Application.Path & "/nature.png" To Application.Path &
    "/nature_.png"
27  hImage = New Image(525, 378, False)
28  hImage = Image.Load(Application.Path & "/nature_.png")
29
30  End
```

Η συνάρτηση ShowImage() εμφανίζει την εικόνα του αντικειμένου Image στο αντικείμενο PictureBox.

```
32  Public Sub ShowImage()
33
34  PictureBox1.Picture = hImage.Picture
35
36  End
```

Το κουμπί btnClear καθαρίζει την εικόνα.

```
39  Public Sub btnClear_Click()
40
41    hImage.Clear
42    ShowImage()
43
44  End
```

Το κουμπί btnReload φορτώνει ξανά την αρχική εικόνα.

```
223  Public Sub btnReload_Click()
224
225    hImage = Image.Load(Application.Path & "/nature_.png")
       ShowImage
226
227  End
228
```

Το κουμπί btnCopy αντιγράφει μια περιοχή από την εικόνα.

```
137  Public Sub btnCopy_Click()
138
139    If lblSelect.Visible = True Then
140     hCopyImage = hImage.Copy(lblSelect.X - PictureBox1.X,
     lblSelect.Y - PictureBox1.Y, lblSelect.Width,
     lblSelect.Height)
141    Else
142      Message.Info("Select an area to copy")
143    Endif
144
145  End
```

Στην γραμμή 139 ελέγχουμε αν το lblSelect είναι ορατό. Αν είναι αυτό σημαίνει ότι ο χρήστης έχει επιλέξει μια περιοχή της εικόνας κάνοντας κλικ και σύρσιμο του ποντικιού στην περιοχή. Σε αυτήν την περίπτωση αντιγράφουμε την περιοχή της επιλογής σε ένα αντικείμενο Image το hCopyImage.
Οι τιμές X,Y της συνάρτησης Copy υπολογίζονται αφού αφαιρεθεί η τιμή μετατόπισης του αντικειμένου PictureBox1 κατά X,Y για αυτό διορθώνονται με τις τιμές lblSelect.X - PictureBox1.X, lblSelect.Y – PictureBox1.Y.

Αλλιώς αν δεν έχει γίνει επιλογή εμφανίζουμε ένα μήνυμα λάθους στον χρήστη.

Το κουμπί btnDraw ζωγραφίζει την επιλογή που κάναμε με το κουμπί btnCopy στην θέση που έχουμε κάνει κλικ πάνω στην οθόνη.

```
147  Public Sub btnDraw_Click()
148
149    If lblSelect.Visible = True Then
150      hImage.Draw(hCopyImage, lblSelect.X - PictureBox1.X,
     lblSelect.Y - PictureBox1.Y)
151    Else
152      Message.Info("Select an area to paste")
153    Endif
154
155    ShowImage()
156
157  End
```

Ο κώδικα εμφάνισης του περιγράμματος επιλογής είναι ο εξής:
ο χρήστης αρχικά κάνει κλικ στο PictureBox1 και καλείται το γεγονός PictureBox1_MouseDown().

```
66   Public Sub PictureBox1_MouseDown()
67
68     iStartX = Mouse.X + PictureBox1.X
69     iStartY = Mouse.Y + PictureBox1.Y
70
71     lblSelect.X = iStartX
72     lblSelect.Y = iStartY
73     lblSelect.Width = 0
74     lblSelect.Height = 0
75     lblSelect.Resize(0, 0)
76     lblStatus.Text = ""
77     lblSelect.Refresh
78
79     If bPickColor = True Then
80       iColor = hImage[Mouse.X, Mouse.Y+32]
81       lblColor.Background = iColor
82       bPickColor = False
83       Me.Mouse = Mouse.Default
84     Endif
85
86   End
```

Στις γραμμές 68, 69 υπολογίζεται το σημείο με τις συντεταγμένες iStartX, iStartY που ο χρήστης κάνει κλικ. Στις συντεταγμένες X, Y του ποντικιού προστίθενται οι συντεταγμένες X,Y του PictureBox1 αφού αυτό είναι τοποθετημένο κατά X,Y σε θέση πάνω στην FMain.

Στις γραμμές 71, 72 το Label λαμβάνει τις υπολογισμένες τιμές X,Y.
Στις γραμμές 73, 74, 75 ορίζεται το ύψος και το πλάτος σε 0 αφού με απλό κλικ θέλουμε να

μην επιλέγεται κάποια περιοχή.

Στην γραμμή 76 ενημερώνουμε το κάτω Label που δείχνει την κατάσταση επιλογής.
Εδώ αφαιρούμε κάθε πληροφορία αφού το απλό κλικ δεν ορίζει κάποια επιλογή περιοχής.

Στην γραμμή 77 ανανεώνουμε την σχεδίαση του lblSelect.

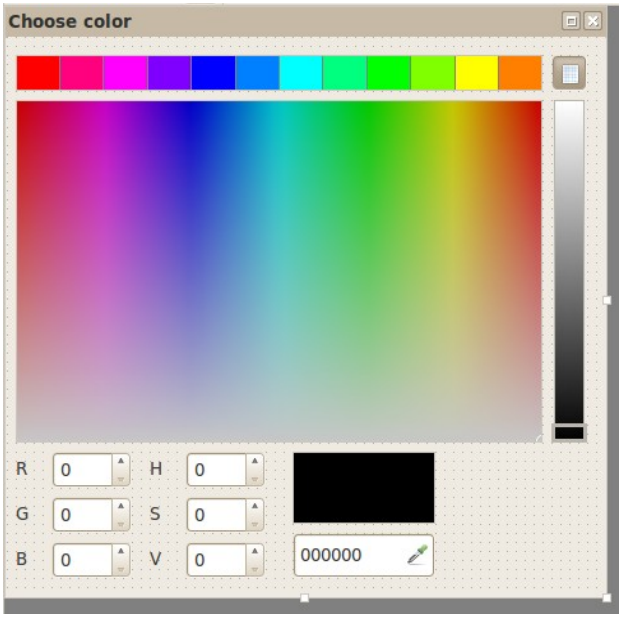
Στις γραμμές 79 με 84 εκτελούμε τον κώδικα επιλογής χρώματος από το σημείο επιλογής από την εικόνα και την επαναφοράς του δείκτη του ποντικιού σε βελάκι από σταγονόμετρο.

Για να επιλέξουμε ένα χρώμα το οποίο θα χρησιμοποιηθεί από τις συναρτήσεις του αντικειμένου Image κάνουμε κλικ στα lblColor και lblNewColor.

Σε κάθε περίπτωση εμφανίζεται η φόρμα FColorChooser και FcolorChooser2 σε αποκλειστική χρήση (ShowModal). Αυτό σημαίνει ότι δεν επιτρέπεται να γίνει χρήση άλλων φορμών μέχρι να κλείσει η FColorChooser ή η FcolorChooser2.

```
122  Public Sub lblColor_MouseDown()
123
124    FColorChooser.ShowModal
125    lblColor.Background = iColor
126
127  End
128
129  Public Sub lblNewColor_MouseDown()
130
131    FColorChooser2.ShowModal
132    lblNewColor.Background = iNewColor
133
134  End
```

Κάθε φόρμα φέρει ένα αντικείμενο επιλογής χρώματος, ColorChooser.



Εικόνα 100: Η διεπαφή των φορμών FColorChooser και FColorChooser2

Ο κώδικας της κάθε φόρμας είναι ο εξής:

FColorChooser

```
3   Public Sub ColorChooser1_Change()
4
5   FMain.iColor = ColorChooser1.SelectedColor
6
7   End
```

FColorChooser2

```
3   Public Sub ColorChooser1_Change()
4
5     FMain.iNewColor = ColorChooser1.SelectedColor
6
7   End
```

Κάθε φόρμα όταν αλλάζει τιμή το χρώμα του αντικειμένου επιλογής χρώματος θέτει την τιμή αυτή στις δημόσιες μεταβλητές iColor και iNewColor της FMain.

Τις δημόσιες αυτές μεταβλητές στην συνέχεια τις χρησιμοποιούν οι συναρτήσεις του αντικειμένου Image.

Το κουμπί btnColorize στρώνει μια επιφάνεια χρώματος από ένα χρώμα επιλογής.

```
88   Public Sub btnColorize_Click()
89
90     hImage.Colorize(iColor)
91     ShowImage()
92
93   End
```

Στην γραμμή 90 καλείται η συνάρτηση Colorize η οποία χρησιμοποιεί το χρώμα iColor.

Το κουμπί btnDesaturate καλεί την συνάρτηση hImage.Desaturate.

```
95   Public Sub btnDesaturate_Click()
96
97     hImage.Desaturate
98     ShowImage()
99
100  End
```

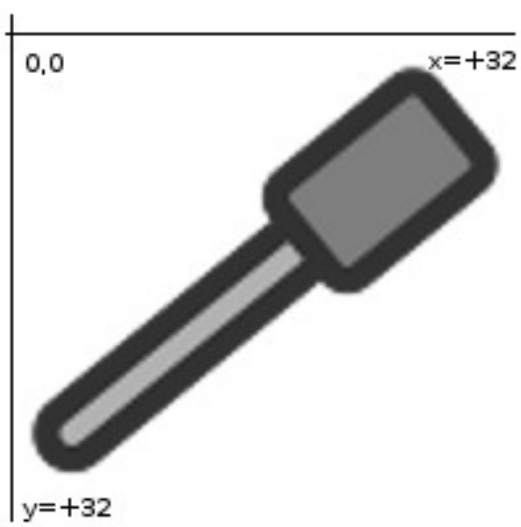
Το κουμπί btnPickColor αλλάζει την εικόνα του δείκτη του ποντικιού σε ένα σταγονόμετρο και θέτει την τιμή TRUE στην μεταβλητή bPickColor. Ο χρήστης μετά μπορεί να κάνει κλικ στην εικόνα και να ορίσει την τιμή χρώματος της μεταβλητής iColor λαμβάνοντας την τιμή χρώματος του σημείου της εικόνας που έκανε κλικ.

```
166  Public Sub btnPickColor_Click()
167
168    Me.Mouse = Mouse.Custom
169    Me.Cursor = cursorPickColor
170    bPickColor = True
171
172  End
```

Κάνοντας κλικ στην εικόνα εκτελείται το εξής μέρος κώδικα όταν η μεταβλητή bPickColor έχει την τιμή TRUE.

```
66      Public Sub PictureBox1_MouseDown()
67
...     ...
78
79        If bPickColor = True Then
80          iColor = hImage[Mouse.X, Mouse.Y+32]
81          lblColor.Background = iColor
82          bPickColor = False
83          Me.Mouse = Mouse.Default
84        Endif
85
86      End
```

Στην γραμμή 80 χρησιμοποιούμε το αντικείμενο hImage σαν πίνακα και λαμβάνουμε από ένα σημείο του την τιμή χρώματος. Προσθέτουμε την τιμή 32 στην συντεταγμένη Y διότι θέλουμε να λάβουμε το σημείο της εικόνας από το κάτω αριστερό σημείο του δείκτη του ποντικιού. Η εικόνα του δείκτη του ποντικιού έχει διαστάσεις 32X32 pixels.



Εικόνα 101: Οι διαστάσεις της εικόνας του δείκτη του ποντικιού

Στην γραμμή 81 θέτουμε την τιμή χρώματος του Background, του αντικειμένου lblColor.
Στην γραμμή 82 θέτουμε στην μεταβλητή bPickColor την τιμή False.
Τέλος στην γραμμή 83 επαναφέρουμε στην προκαθορισμένη μορφή το εικονίδιο του δείκτη του ποντικιού.

Το κουμπί btnErase καλεί την συνάρτηση Erase.

```
Public Sub btnErase_Click()

  hImage.Erase(iColor)
  ShowImage()

End
```

Το κουμπί btnFill καλεί την συνάρτηση Fill().

```
Public Sub btnFill_Click()

  hImage.Fill(iColor)
  ShowImage()

End
```

Το κουμπί btnReplace καλεί την συνάρτηση Replace()

```
Public Sub btnReplace_Click()

  hImage = hImage.Replace(iColor, iNewColor)
  ShowImage()

End
```

Το κουμπί btnDrawRect καλεί την συνάρτηση DrawRect().

```
Public Sub btnDrawRect_Click()

  If lblSelect.Visible = True Then
    hImage.DrawRect(lblSelect.X - PictureBox1.X,
lblSelect.Y - PictureBox1.Y, lblSelect.Width,
lblSelect.Height, Color.SetAlpha(iColor,
sldTranparency.Value))
    ShowImage()
  Else
    Message.Info("Please select an area first")
  Endif

End
```

Η συνάρτηση DrawRect λαμβάνει τις συντεταγμένες X,Y του πλαισίου επιλογής οι οποίες είναι το X, Y του lblSelect αφού αφαιρεθεί το PictureBox1.X και Y αντίστοιχα.
Επίσης σαν χρώμα λαμβάνει το χρώμα που επιστρέφει η συνάρτηση Color.SetAlpha().
Η συνάρτηση αυτή λαμβάνει ένα χρώμα σε τύπο δεδομένων integer και μια τιμή από το 0 μέχρι το 255 που δηλώνει το ποσοστό διαφάνειας. Η τιμή 0 επιστρέφει χρώμα πλήρως συμπαγές ενώ η τιμή 255 επιστρέφει χρώμα πλήρως διαφανές. Στην συνάρτηση Color.SetAlpha() διοχετεύουμε την τιμή του αντικειμένου Slider.

Εάν ο χρήστης δεν έχει κάνει επιλογή δλδ η ιδιότητα lblSelect.Visible = True τότε εμφανίζουμε ένα μήνυμα λάθους.

Το κουμπί btnFillRect ομοίως καλεί την συνάρτηση FillRect().

```
175  Public Sub btnFillRect_Click()
176
177    If lblSelect.Visible = True Then
178      hImage.FillRect(lblSelect.X - PictureBox1.X,
     lblSelect.Y - PictureBox1.Y, lblSelect.Width,
     lblSelect.Height, iColor)
179      ShowImage()
180    Else
181      Message.Info("Please select an area first")
182    Endif
183
184  End
```

Το κουμπί btnGrayScale καλεί την συνάρτηση Grey().

```
186  Public Sub btnGrayScale_Click()
187
188    hImage.Gray()
189    ShowImage()
190
191  End
```

Το κουμπί btnMask καλεί την συνάρτηση Mask().

```
193  Public Sub btnMask_Click()
194
195    hImage = hImage.Mask(iColor)
196    ShowImage()
197
198  End
```

Το κουμπί btnMirrorH καλεί την συνάρτηση Mirror() και δημιουργεί έναν οριζόντιο κατοπτρισμό.

```
Public Sub btnMirrorH_Click()

  hImage = hImage.Mirror(False, True)
  ShowImage()

End
```

Το κουμπί btnMirrorV καλεί την συνάρτηση Mirror() και δημιουργεί έναν κάθετο κατοπτρισμό.

```
Public Sub btnMirrorV_Click()

  hImage = hImage.Mirror(True, False)
  ShowImage()

End
```

Το κουμπί btnResize εμφανίζει σε αποκλειστική χρήση την φόρμα FResize.

```
Public Sub btnResize_Click()

  FResize.ShowModal
  ShowImage()

End
```

Η φόρμα FResize έχει την εξής διεπαφή:

Εικόνα 102: Η διεπαφή της φόρμας FResize

Η φόρμα λαμβάνει με τα textboxes, txtNewH και txtNewW τις διαστάσεις Height και Width

για την εικόνα. Η φόρμα έχει δύο ανενεργά textboxes που εμφανίζει τις αρχικές διαστάσεις Height και Width.

Ο κώδικας της φόρμας είναι:

FResize

```
1   ' Gambas class file
2
3
4   Public Sub btnCancel_Click()
5
6     Me.Close
7
8   End
9
10  Public Sub btnOk_Click()
11
12    If IsNumber(txtNewH.Text) And IsNumber(txtNewW.Text)
    Then
13    FMain.hImage = FMain.hImage.Resize(Val(txtNewW.Text),
    Val(txtNewH.Text))
14    Endif
15
16    Me.Close
17
18  End
19
20  Public Sub Form_Open()
21
22    txtOldH.Text = FMain.hImage.Height
23    txtOldW.Text = FMain.hImage.Width
24
25  End
```

Στο event btnCancel_Click() όταν πατηθεί το κουμπί Cancel κλείνουμε την φόρμα.

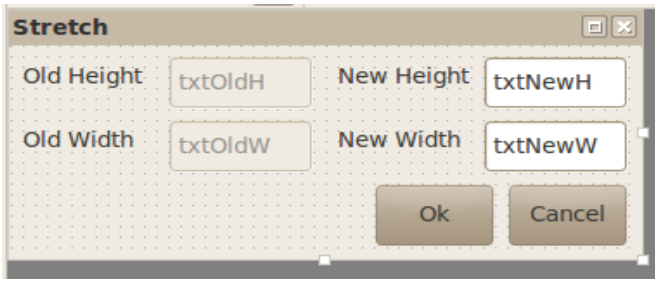
Όταν πατηθεί το πλήκτρο OK ελέγχουμε αν είναι αριθμοί οι τιμές κειμένου των txtNewH και txtNewW στην γραμμή 12.
Αν είναι μετατρέπουμε αυτές σε αριθμούς με την συνάρτηση Val() και καλούμε την συνάρτηση Resize() του αντικειμένου hImage της φόρμας FMain.
Στην συνέχεια κλείνουμε την φόρμα.

Στην γραμμή 20 στο άνοιγμα της φόρμας θέτουμε τις τιμές ύψους, μήκους της εικόνας στις τιμές κειμένου των textboxes, txtOldH και txtOldW.

Το κουμπί btnStretch εμφανίζει σε αποκλειστική χρήση την φόρμα FStretch.

```
234 Public Sub btnStretch_Click()
235
236   FStretch.ShowModal
237   ShowImage()
238
239 End
```

Η φόρμα FStretch έχει την εξής διεπαφή:



Εικόνα 103: Η διεπαφή της φόρμας FStretch

Η φόρμα εμφανίζει τις διαστάσεις της εικόνας σε απενεργοποιημένα textboxes και λαμβάνει τις νέες διαστάσεις Height, Width για να κάνει stretch την εικόνα.

Ο κώδικας της φόρμας είναι:

<u>FStretch</u>

```
1  ' Gambas class file
2
3
4  Public Sub btnCancel_Click()
5
6    Me.Close
7
8  End
9
10 Public Sub btnOk_Click()
11
12   If IsNumber(txtNewH.Text) And IsNumber(txtNewW.Text)
   Then
13   FMain.hImage = FMain.hImage.Stretch(Val(txtNewW.Text),
   Val(txtNewH.Text))
14   Endif
15
16   Me.Close
```

```
17
18  End
19
20  Public Sub Form_Open()
21
22    txtOldH.Text = FMain.hImage.Height
23    txtOldW.Text = FMain.hImage.Width
24
25  End
```

Ο κώδικας της φόρμας είναι παρόμοιος με αυτόν την FResize και δεν θα γίνει άλλη περιγραφή.

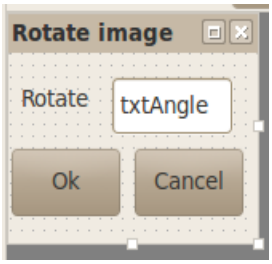
Το κουμπί btnTransparent καλεί την συνάρτηση Transparent() του αντικειμένου hImage.

```
241  Public Sub btnTransparent_Click()
242
243    hImage = hImage.Transparent(iColor)
244    ShowImage()
245
246  End
```

Το κουμπί btnRotate εμφανίζει σε αποκλειστική χρήση την φόρμα FRotate.

```
248  Public Sub btnRotate_Click()
249
250    FRotate.ShowModal
251    ShowImage()
252
253  End
```

Η φόρμα FRotate έχει την εξής διεπαφή:



Εικόνα 104: Η φόρμα FRotate

Η φόρμα λαμβάνει σε ένα textbox μια αριθμητική τιμή Float και στην συνέχεια πατώντας το κουμπί OK στρέφει την εικόνα κατά την αριθμητική τιμή που έλαβε.

Ο κώδικας της φόρμας FRotate είναι ο εξής:

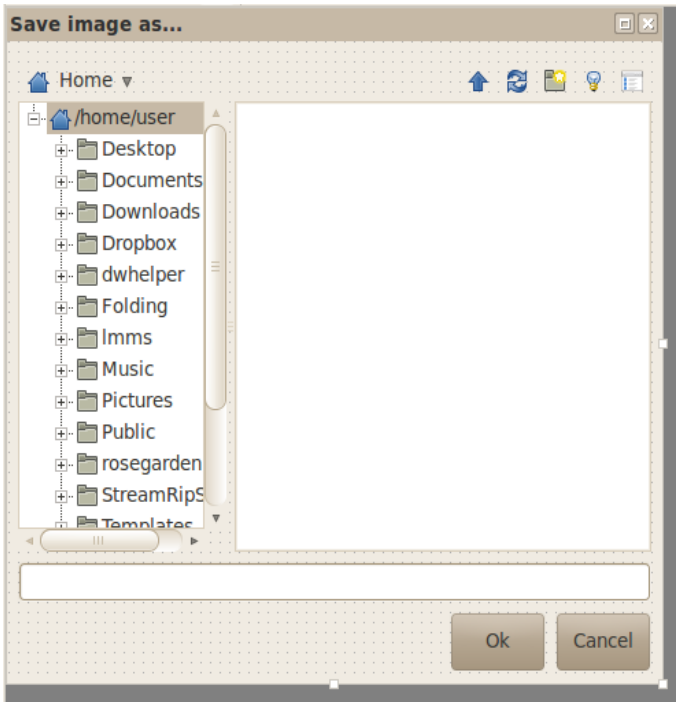
```
1   ' Gambas class file
2
3
4   Public Sub btnOk_Click()
5
6     If IsNumber(txtAngle.Text) Then
7       FMain.hImage = FMain.hImage.Rotate(-
    Rad(Val(txtAngle.Text)))
8   Endif
9
10    Me.Close
11
12  End
13
14  Public Sub btnCancel_Click()
15
16    Me.Close
17
18  End
```

Πατώντας το κουμπί OK καλείται το γεγονός btnOk_Click() στην γραμμή 4.
Στην γραμμή 6 ελέγχουμε αν η τιμή του textbox είναι αριθμός. Αν είναι, μετατρέπουμε την τιμή αυτή σε αριθμητική και στην συνέχεια την μετατρέπουμε σε ακτίνια (Radians) με την συνάρτηση Rad() και την διοχετεύουμε στην συνάρτηση Rotate του αντικειμένου hImage της φόρμας FMain. Η συνάρτηση Rotate() στρέφει την εικόνα προς τα αριστερά για αυτό βάζουμε το πρόσημο – στην τιμή διοχέτευσης για να γίνεται η περιστροφή σύμφωνα με τους δείκτες του ρολογιού όπως άλλωστε δείχνει και το εικονίδιο. Στην συνέχεια κλείνουμε την φόρμα.

Πατώντας το κουμπί Cancel καλείται το γεγονός btnCancel_Click(). Εδώ εκτελείται ο κώδικας κλεισίματος της φόρμας χωρίς άλλη ενέργεια.

Το κουμπί btnSave εμφανίζει σε αποκλειστική χρήση την φόρμα FSave.

Η φόρμα FSave έχει την εξής διεπαφή.



Εικόνα 105: Η διεπαφή της φόρμας FSave

Η φόρμα περιέχει ένα FileChooser και δύο κουμπιά.
Εδώ θα χρησιμοποιήσουμε τον FileChooser για να επιλέξουμε την τοποθεσία αποθήκευσης του αρχείου.

Ο κώδικας της φόρμας είναι ο εξής:

```
1   ' Gambas class file
2
3
4   Public Sub btnCancel_Click()
5
6     Me.Close
7
8   End
9
10  Public Sub Form_Open()
11
12    FileChooser1.Dir = Application.Path
13
14  End
15
16  Public Sub btnOk_Click()
17
18    FMain.hImage.Save(FileChooser1.SelectedPath)
19    Me.Close
20
21  End
```

Πατώντας το κουμπί Close εκτελείται ο κώδικας στις γραμμές 4-8 όπου στην γραμμή 6 κλείνουμε την φόρμα.

Ανοίγοντας η φόρμα εκτελεί το γεγονός Form_Open().
Στην γραμμή 12 θέτουμε σαν φάκελο εμφάνισης στο FileChooser το φάκελο που βρίσκεται η εφαρμογή.

Πατώντας το κουμπί OK καλείται η ρουτίνα btnOk_Click().
Στην γραμμή 18 αποθηκεύεται η εικόνα του αντικειμένου hImage σύμφωνα με την τιμή που έχει το αντικείμενο FileChooser1.
Στην συνέχεια κλείνουμε την φόρμα στην γραμμή 19.

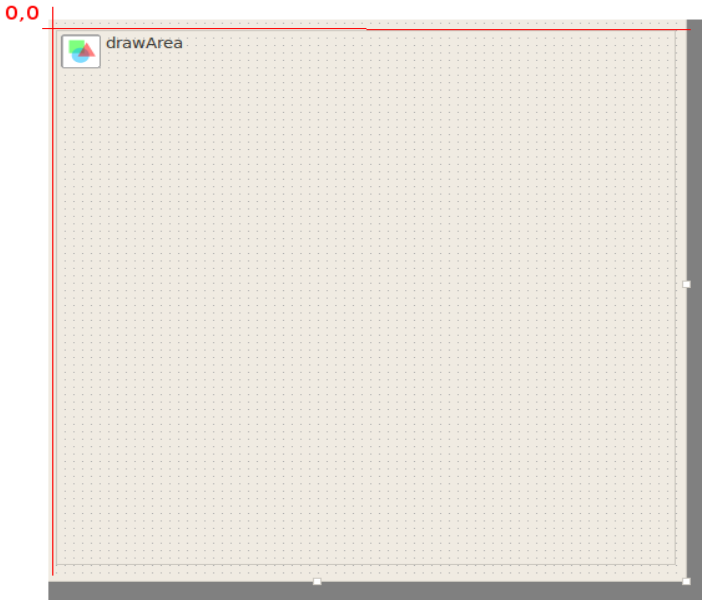
Στο σημείο ολοκληρώθηκε η περιγραφή της εφαρμογής Project116.

Δουλεύοντας με εικόνες, η κλάση DrawingArea

Η κλάση υλοποιεί ένα αντικείμενο DrawingArea το οποίο είναι μια περιοχή σχεδίασης όπου η εφαρμογή μπορεί να σχεδιάσει γεωμετρικά σχήματα όπως γραμμές, κύκλους και τόξα κτλ. Λειτουργεί σε δύο καταστάσεις την standard ή την cached όπως ορίζεται από την ιδιότητα Cached.
Στην κατάσταση standard το αντικείμενο καλεί το Draw event κάθε φορά που χρειάζεται ανανέωση.
Στην κατάσταση cached η σχεδίαση γίνεται μια φορά χρησιμοποιώντας το αντικείμενο ως συσκευή σχεδίασης. Αυτή η σχεδίαση αποθηκεύεται σε μια εσωτερική κλάση Picture που χρησιμοποιείται από το αντικείμενο DrawingArea.

Η σχεδίαση γίνεται με την κλάση Draw ή Paint.
Ποια κλάση θα χρησιμοποιηθεί ορίζεται από την ιδιότητα Painted.
Αν χρησιμοποιείσετε την κλάση Paint τότε γίνεται εξομάλυνση της σχεδίασης (anti-aliased).

Οι συντεταγμένες της περιοχής σχεδίασης αρχίζουν από την πάνω αριστερή γωνία.



Η κλάση μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hDrawingArea AS DrawingArea

hDrawingArea = NEW DrawingArea ( Parent AS Container ) [ AS "event name" ]
```

Η κλάση κληρονομεί την κλάση Container

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Container		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Children ClientH ClientHeight ClientW ClientWidth ClientX ClientY Cursor Design Drop Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Find Grab Hide Lower Move MoveScaled Raise Reparent Resize ResizeScaled Screenshot SetFocus Show	Arrange BeforeArrange DblClick Drag DragMove Drop Enter GotFocus Insert KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης DrawingArea		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Arrangement AutoResize Background Border Cached Enabled Focus Indent Margin NoBackground Padding Painted Spacing	Clear Refresh	Draw

Δουλεύοντας με εικόνες, η κλάση Draw.

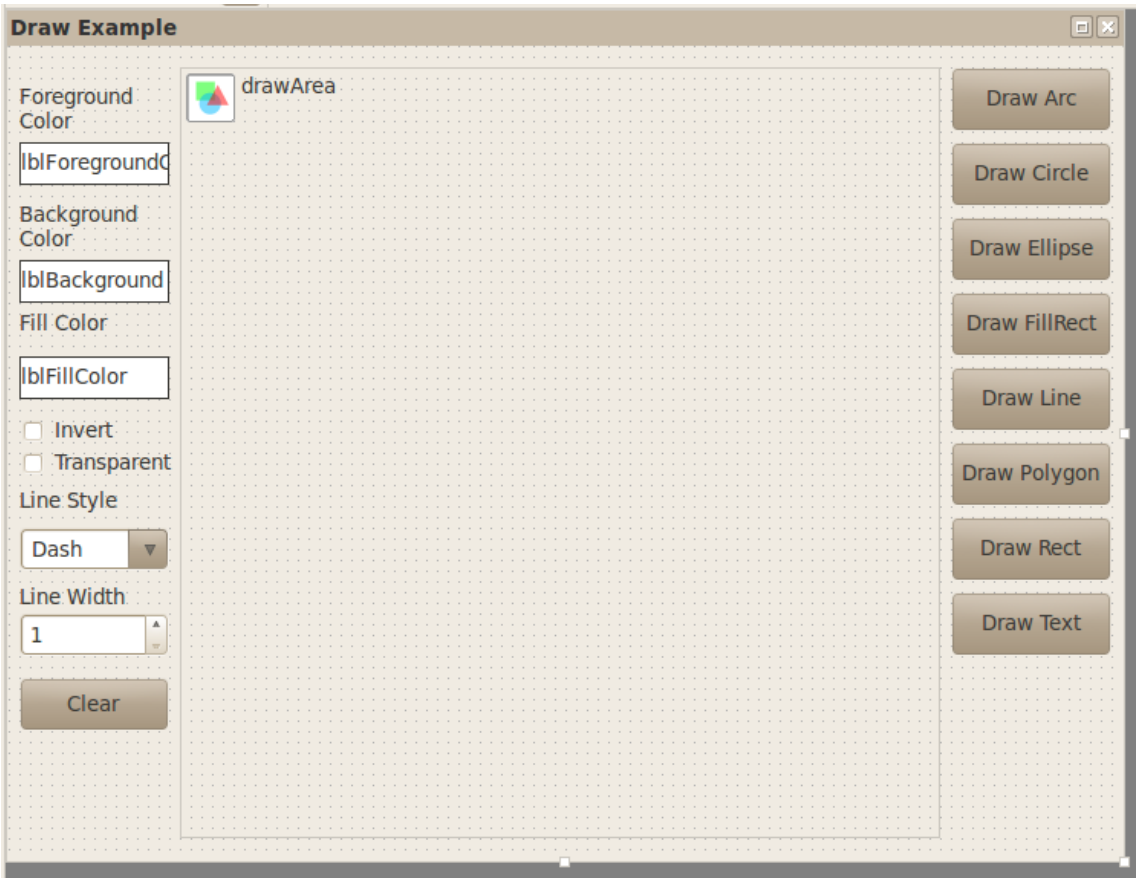
Η κλάση σχεδιάζει σχήματα στα αντικείμενα Picture, ή Window ή DrawingArea.
Πριν την εκκίνηση σχεδίασης πρέπει να κληθεί η μέθοδος Begin() και να της διοχετευτεί το αντικείμενο στο οποίο θα γίνει η σχεδίαση.
Στην συνέχεια μπορεί να κληθεί οποιαδήποτε μέθοδος σχεδίασης της κλάσης Draw.
Όταν τελειώσει η σχεδίαση πρέπει να κληθεί η μέθοδος End().
Η σχεδίαση πλέον στην Gambas3 γίνεται στο event Draw σε όποιο αντικείμενο το έχει.
Αν ένα αντικείμενο δεν έχει το συμβάν Draw δεν μπορεί να χρησιμοποιηθεί η κλάση Draw. Για παράδειγμα μια φόρμα, Form, δεν έχει συμβάν Draw και δεν είναι δυνατή η σχεδίαση με την κλάση Draw, ενώ ένα αντικείμενο DrawingArea έχει συμβάν Draw και είναι δυνατή η σχεδίαση με την κλάση Draw.

Σημείωση, δεν μπορείτε να σχεδιάσετε σε ένα αντικείμενο DrawingArea καλώντας μεθόδους σχεδίασης έξω από το συμβάν Draw, εκτός και αν η ιδιότητα Cache έχει τεθεί στην τιμή TRUE. Τότε το αντικείμενο DrawingArea λειτουργεί σαν εικόνα.

Η κλάση είναι στατική δεν μπορεί να δημιουργηθεί.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Draw		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Σταθερές (Constants)**
Background Clip ClipRect Device FillColor FillStyle FillX FillY Font Foreground H Height Invert LineStyle LineWidth Resolution Style Transparent W Width	Arc Begin Circle Clear Ellipse End FillRect Image Line Picture Point Polygon Polyline Pop Push Rect Reset Restore RichText RichTextHeight RichTextWidth Save Scale Text TextHeight TextWidth Tile Translate Zoom	Chord Disabled Focus Hover Normal Pie

Εφαρμογή 1
Το Project117 είναι ένα παράδειγμα για την κλάση Draw.
Η διεπαφή του προγράμματος είναι η παρακάτω:



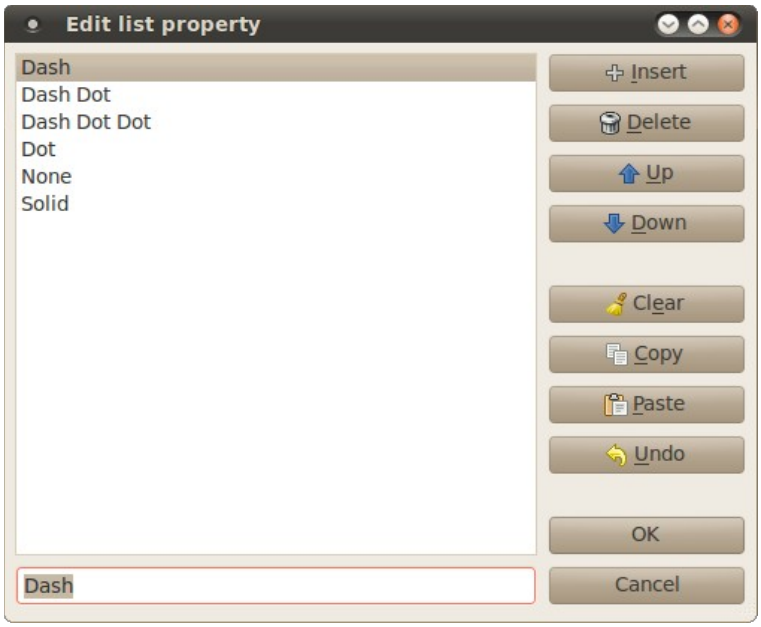
Στα αριστερά υπάρχουν τρία Labels, ένα για το Foreground χρώμα, ένα για το Background χρώμα και ένα για το FillColor χρώμα.

Από κάτω υπάρχουν δύο Checkboxes που όταν είναι σε κατάσταση TRUE θέτουν σε TRUE αντίστοιχα τις ιδιότητες Invert και Transparent.

Στην συνέχεια υπάρχει ένα αντικείμενο ComboBox. Σε αυτό έχουμε προσθέσει τις επιλογές Dash, Dash Dot, Dash Dot Dot, Dot, None, Solid.

Η προσθήκη των επιλογών αυτών γίνεται από το πλαίσιο διαλόγου τις ιδιότητας List του αντικειμένου.

Το πλαίσιο διαλόγου είναι το παρακάτω:



Εικόνα 106: Το πλαίσιο διαλόγου προσθήκης επιλογών στο αντικείμενο ComboBox

Με αυτό το πλαίσιο διαλόγου μπορούμε να εισάγουμε μια νέα επιλογή, να διαγράψουμε μια υπάρχουσα, να μετακινήσουμε τις επιλογές πάνω ή κάτω, να καθαρίσουμε την λίστα επιλογών, να αντιγράψουμε ή να επικολλήσουμε ή να αναιρέσουμε μια επιλογή.

Κάτω από το αντικείμενο ComboBox βρίσκεται ένα αντικείμενο SpinBox.
Το αντικείμενο SpinBox έχει μια ελάχιστη αριθμητική τιμή και μια μέγιστη αριθμητική τιμή. Ο χρήστης μπορεί να αυξάνει ή να μειώνει την τιμή με τα βελάκια από την δεξιά μεριά. Στο παράδειγμα αυτό η τιμή του SpinBox ορίζει την ιδιότητα Draw.LineWidth, δλδ το πλάτος της γραμμής που θα σχεδιαστεί.

Κάτω από το SpinBox βρίσκεται ένα κουμπί που όταν πατηθεί καθαρίζει την περιοχή σχεδίασης.

Στο κέντρο της φόρμας βρίσκεται ένα αντικείμενο DrawingArea.
Το αντικείμενο DrawingArea έχει την ιδιότητα Cached στην τιμή TRUE.
Αυτό σημαίνει ότι μπορούμε να χρησιμοποιούμε το αντικείμενο ως μια εικόνα που θα σχεδιάζουμε πάνω της προσθέτοντας σχήματα. Σε κάθε άλλη περίπτωση θα έπρεπε να καλούμε την ιδιότητα DrawingArea.Refresh() και να σχεδιάζουμε το επιθυμητό σχήμα ή σχήματα από την αρχή.

Στο παράδειγμα αυτό όμως δεν είναι επιθυμητή αυτή η λειτουργία. Εδώ θέλουμε ο χρήστης να μπορεί να σχεδιάζει στην περιοχή σχεδίασης χωρίς να χάνονται τα παλιά σχήματα και χωρίς να καλούνται όλες οι συναρτήσεις σχεδίασης από την αρχή προκειμένω να σχεδιαστεί κάτι.

Στα δεξιά υπάρχουν 8 κουμπιά με μια από τις βασικές μεθόδους σχεδίασης.

Στην εφαρμογή αυτή ο χρήστης πριν σχεδιάσει κάτι επιλέγει τις ιδιότητες χρωμάτων, πλάτους γραμμής κτλ και στην συνέχεια πατά ένα κουμπί σχεδίασης.

Η εφαρμογή έχει 4 φόρμες. Τις FMain, FcolorChooser1, FcolorChooser2, FcolorChooser3.

Οι FColorChooser1, FColorChooser2, FColorChooser3 είναι όμοιες φόρμες και η κάθε μία ορίζει το χρώμα για τις ιδιότητες Background, FillColor, ForeColor της κλάσης Draw.

Ο κώδικας της φόρμας FMain είναι ο εξής:

```
1  ' Gambas class file
2
3  Public iBgColor As Integer
4  Public iFillColor As Integer
5  Public iForeColor As Integer
6  Private aPolygon As New Integer[]
7  Private iShape As Integer
```

Δηλώνουμε 5 δημόσιες μεταβλητές.
Η iBgColor, iFillCorol, iForeColor είναι τύπου Integer και θα χρησιμοποιηθούν για την εκχώρηση των τιμών των χρωμάτων.

Η aPolygon είναι ένας δυναμικός πίνακας ακεραίων και θα χρησιμοποιηθεί ως πίνακας σημείων X,Y για ένα πολύγωνο με 4 σημεία οπότε θα έχει 8 στοιχεία ο πίνακας. Κάθε ζεύγος στοιχείων θα είναι ένα σημείο X,Y του πολυγώνου. Ο πίνακας δηλώνεται δημόσιος αλλά Private διότι δεν θα χρησιμοποιηθεί από άλλες φόρμες.

Η μεταβλητή iSphape είναι μια δημόσια μεταβλητή τύπου Integer και θα χρησιμοποιηθεί σε μια δομή SELECT CASE. Με το πάτημα των κουμπιών σχεδίασης θα της εκχωρείται μια τιμή και στην συνέχεια θα καλείται το συμβάν Draw του αντικειμένου DrawingArea. Στο συμβάν αυτό υπάρχει μια δομή SELECT CASE όπου ανάλογα την τιμή της iShape θα σχεδιάζει το αντίστοιχο σχήμα.

Όταν ο χρήστης θέλει να ορίσει κάποιο χρώμα κάνει κλικ στο lblForegroundColor ή στο lblBackground ή στο lblFillColor.

```
77  Public Sub lblBackground_MouseDown()
78
79    FColorChooser1.ShowModal
80    lblBackground.Background = iBgColor
81
82  End
83
84  Public Sub lblFillColor_MouseDown()
85
86    FColorChooser2.ShowModal
87    lblFillColor.Background = iFillColor
88
89  End
90
91  Public Sub lblForegroundColor_MouseDown()
92
93    FColorChooser3.ShowModal
94    lblForegroundColor.Background = iForeColor
95
96  End
```

Τότε εμφανίζεται σε κατάσταση αποκλειστικής χρήσης η αντίστοιχη φόρμα π.χ. FColorChooser1.ShowModal.

Οι φόρμες αυτές έχουν ένα αντικείμενο ColorChooser όπου ο χρήστης επιλέγει ένα χρώμα το οποίο είναι ένας integer που τον εκχωρεί στις δημόσιες μεταβλητές της φόρμας FMain, iBgColor, iFillColor, iForeColor αντίστοιχα.

Ο κώδικας των φορμών αυτών είναι όμοιος.
Για παράδειγμα ο κώδικας της φόρμας FcolorChooser1 είναι:

```
1   ' Gambas class file
2
3
4   Public Sub btnCancel_Click()
5
6     Me.Close
7
8   End
9
10  Public Sub ColorChooser1_Change()
11
12    FMain.iBgColor = ColorChooser1.SelectedColor
13
14  End
15
16  Public Sub btnOK_Click()
17
18    Me.Close
19
20  End
```

Όταν πατηθεί το κουμπί btnCancel ή btnOK η φόρμα κλείνει.
Όταν γίνει μια αλλαγή στο ColorChooser1 τότε ορίζουμε την δημόσια μεταβλητή της FMain με το επιλεγμένο χρώμα.

Ομοίως οι άλλες φόρμες FColorChooser2, FColorChooser3 εκχωρούν σε κάθε αλλαγή τους την τιμή του χρώματος επιλογής στις δημόσιες μεταβλητές της FMain.

```
10  Public Sub ColorChooser1_Change()
11
12    FMain.iFillColor = ColorChooser1.SelectedColor
13
14  End
```

```
10  Public Sub ColorChooser1_Change()
11
12    FMain.iForeColor = ColorChooser1.SelectedColor
13
14  End
```

Πριν σχεδιαστεί ένα σχήμα στην περιοχή σχεδίασης η εφαρμογή θέτει τις τιμές σχεδίασης από τις επιλογές του χρήστη όπως τα χρώματα, το πλάτος γραμμής, το είδος γραμμής κτλ.

Αυτές οι ενέργειες γίνονται από την υπορουτίνα SetValues().

```
10  Public Sub SetValues()
11
12  'Set Polygon points x,y
13  'point1
14  aPolygon.Add(320)
15  aPolygon.Add(310)
16
17  'point2
18  aPolygon.Add(380)
19  aPolygon.Add(310)
20
21  'point3
22  aPolygon.Add(350)
23  aPolygon.Add(380)
24
25  'point4
26  aPolygon.Add(310)
27  aPolygon.Add(380)
28
29  'set other values
30  CheckInvert()
31  CheckTransparent()
32  SelectLineStyle()
33
34  Draw.LineWidth = spnLineWidth.Value
35  Draw.Foreground = lblForegroundColor.Background
36  Draw.Background = lblBackground.Background
37  Draw.FillColor = lblFillColor.Background
38
39  End
```

Στις γραμμές 12 -27 προστίθενται στον πίνακα aPolygon τα σημεία X,Y του πολυγώνου που θα σχεδιαστεί.

Στην συνέχεια η υπορουτίνα CheckInvert() ελέγχει την κατάσταση του checkbox chkInvert και θέτει αντίστοιχα την τιμή Draw.Invert.

```
63  Public Sub CheckInvert()
64
65  If chkInvert.Value = True Then Draw.Invert = True
66  If chkInvert.Value = False Then Draw.Invert = False
67
68  End
```

Η υπορουτίνα CheckTransparent() κάνει ομοίως το ίδιο για την ιδιότητα Draw.Transparent.

```
70  Public Sub CheckTransparent()
71
72  If chkTransparent.Value = True Then Draw.Transparent =
    True
73  If chkTransparent.Value = False Then Draw.Transparent =
    False
74
75  End
```

Η υπορουτίνα SelectLineStyle() ελέγχει την ιδιότητα κειμένου του cmbLineStyle, cmbLineStyle.Text και βάσει αυτής με μια δομή SELECT CASE ορίζει την ιδιότητα Draw.LineStyle.

```
42  Public Sub SelectLineStyle()
43
44  Select Case cmbLineStyle.Text
45    Case "Dash"
46      Draw.LineStyle = Line.Dash
47    Case "Dash Dot"
48      Draw.LineStyle = Line.DashDot
49    Case "Dash Dot Dot"
50      Draw.LineStyle = Line.DashDotDot
51    Case "Dot"
52      Draw.LineStyle = Line.Dot
53    Case "None"
54      Draw.LineStyle = Line.None
55    Case "Solid"
56      Draw.LineStyle = Line.Solid
57  End Select
58
59  End
```

Τέλος ορίζονται οι τιμές

```
34  Draw.LineWidth = spnLineWidth.Value
35  Draw.Foreground = lblForegroundColor.Background
36  Draw.Background = lblBackground.Background
37  Draw.FillColor = lblFillColor.Background
```

Αφού οριστούν οι βασικές ιδιότητες σχεδίασης μπορούμε να δούμε τον κώδικα σχεδίασης ενός σχήματος.

Όταν πατηθεί ένα κουμπί σχεδίασης, π.χ. το btnArc τότε έχουμε τον ακόλουθο κώδικα:

```
98   Public Sub btnArc_Click()
99
100  iShape = 1
101  LetsDraw()
102
103  End
```

Ορίζουμε την τιμή της μεταβλητής iShape. Κάθε κουμπί εκχωρεί διαφορετική τιμή στην μεταβλητή αυτή.

Στην συνέχεια καλούμε την υπορουτίνα LetsDraw().

```
159  Public Sub LetsDraw()
160
161  Draw.Begin(drawArea)
162  drawArea_Draw()
163  Draw.End
164
165  End
```

Η υπορουτίνα αυτή είναι δημιουργημένη από εμάς.
Στην γραμμή 161 ξεκινάμε την σχεδίαση με συσκευή σχεδίασης την drawArea.
Στην γραμμή 162 καλούμε την ρουτίνα του συμβάντος Draw της περιοχής σχεδίασης.
Τέλος στην γραμμή 163 δηλώνουμε το τέλος σχεδίασης.

Οι συναρτήσεις σχεδίασης υλοποιούνται στην υπορουτίνα
Public Sub drawArea_Draw().

```
167  Public Sub drawArea_Draw()
168
169  SetValues()
170
171  Select Case iShape
172    Case 1 'draw arc
173     Draw.Arc(10, 10, 100, 150, 14.5, 28.6)
174
175    Case 2 'draw circle
176     Draw.Circle(150, 141, 43)
177
178    Case 3 'draw ellipse
179     Draw.Ellipse(250, 250, 145, 56)
180
181    Case 4 'draw fill rect
182     Draw.FillRect(82, 50, 130, 80)
183
184    Case 5 'draw line
185      Draw.Line(55, 47, 289, 367)
186
187    Case 6 'draw polygon
188     Draw.Polygon(aPolygon)
189
190    Case 7 'draw rectangle
191     Draw.Rect(300, 200, 54, 74)
192
193    Case 8 'draw text
194     Draw.Text("Hello World!", 250, 300)
195  End Select
196
197  End
```

Στην γραμμή 169 καλείται η υπορουτίνα SetValues() και τίθενται οι τιμές σχεδίασης της κλάσης Draw.

Στην γραμμή 171 ξεκινά η δομή SELECT CASE για την τιμή της μεταβλητής iShape.

Σε κάθε περίπτωση της τιμής της iShape σχεδιάζεται και ένα σχήμα.
Η τιμή της iShape εκχωρείται από το πάτημα κάθε κουμπιού από την δεξιά μεριά της φόρμας FMain.

Τέλος το κουμπί btnClear καθαρίζει την περιοχή σχεδίασης.

```
199  Public Sub btnClear_Click()
200
201    drawArea.Clear
202
203  End
```

Δουλεύοντας με εικόνες, η κλάση SvgImage.

Η κλάση αντιπροσωπεύει ένα αρχείο SVG το οποίο φορτώνεται στην μνήμη.
Τα αντικείμενα SvgImage μπορούν να σχεδιαστούν με την κλάση Paint και να αποθηκευτούν οι οποιεσδήποτε αλλαγές πίσω πάλι στο αρχείο.

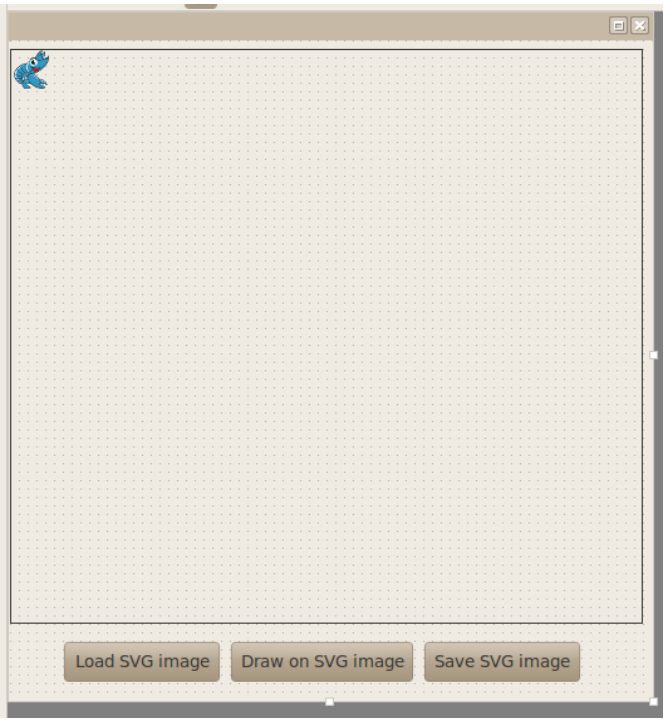
Η κλάση μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hSvgImage AS SvgImage

hSvgImage = NEW SvgImage ( [ Width AS Float, Height AS Float ] )
```

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης SvgImage		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Στατικές μέθοδοι (Static methods)**
Height Width	Clear Paint Resize Save	Load

Εφαρμογή 1
Το Project118 είναι ένα πρόγραμμα για την κλάση SvgImage.
Η διεπαφή του προγράμματος είναι η παρακάτω.



Εικόνα 107: Η διεπαφή του Project118

Αποτελείται από τρία αντικείμενα Button και ένα αντικείμενο Picturebox.

Το πρώτο κουμπί φορτώνει μια εικόνα SVG στο PictureBox.
Το δεύτερο κουμπί σχεδιάζει πάνω στην εικόνα SVG στην συνέχεια αποθηκεύει το αποτέλεσμα σε ένα προσωρινό αρχείο SVG και φορτώνει αυτό το αρχείο στο αντικείμενο PictureBox.
Το τρίτο κουμπί αποθηκεύει την εικόνα SVG.

Αλλά ας δούμε τα μέρη του προγράμματος.

```
1   ' Gambas class file
2
3   Private hSVGImage As SvgImage
4
5   Public Sub Form_Open()
6
7    Me.Center
8
9   End
```

Δηλώνουμε ένα καθολικό ιδιωτικό αντικείμενο τύπου SvgImage, το hSVGImage στο οποίο θα φορτωθεί η εικόνα SVG.
Κατά το άνοιγμα της φόρμας τοποθετούμε την φόρμα στο κέντρο της οθόνης.

```
11  Public Sub btnLoad_Click()
12
13    hSVGImage = SvgImage.Load("sun.svg")
14    picBox.Picture = Picture.Load("sun.svg")
15    Message.Info("SVG image loaded")
16
17  End
```

Κάνοντας κλικ στο κουμπί `btnLoad` στην γραμμή 13 φορτώνουμε την εικόνα sun.svg στο αντικείμενο `hSVGImage`.
Επίσης φορτώνουμε την εικόνα στο αντικείμενο `picBox` στην γραμμή 14.
Στην συνέχεια, στην γραμμή 15, ενημερώνουμε τον χρήστη ότι η εικόνα φορτώθηκε.

```
19  Public Sub btnDraw_Click()
20
21   Paint.Begin(hSVGImage)
22   Paint.Brush = Paint.RadialGradient(200, 140, 40, 215, 115,
    [Color.RGB(255, 0, 0, 64), Color.White], [1.0, 0.1])
23   Paint.Arc(200, 140, 40)
24   Paint.Fill
25   Paint.End
26
27   hSVGImage.Save(Application.Path &/ "_tmp_sun.svg")
28   picBox.Picture = Picture.Load(Application.Path &/
```

```
   "_tmp_sun.svg")
29  Try Kill Application.Path &/ "_tmp_sun.svg"
30
31  Message.Info("SVG image was drawn")
32
33 End
```

Η ρουτίνα btnDraw_Click() καλείται όταν πατηθεί το κουμπί btnDraw.
Στην γραμμή 21 ξεκινάμε την σχεδίαση στο αντικείμενο hSVGImage.
Στην γραμμή 22 ορίζουμε μια βούρτσα από την κλάση Paint.
Στην γραμμή 23 σχεδιάζουμε ένα τόξο.
Στην γραμμή 24 γεμίζουμε την περιοχή σχεδίασης.
Στην γραμμή 25 τελειώνουμε την σχεδίαση.
Στην γραμμή 27 αποθηκεύουμε την εικόνα στην διαδρομή της εφαρμογής με το όνομα _tmp_sun.svg.
Στην γραμμή 28 εμφανίζουμε το αποτέλεσμα στο Picturebox picBox φορτώνοντας την εικόνα που αποθηκεύσαμε.
Στην γραμμή 29 διαγράφουμε την προσωρινή εικόνα _tmp_sun.svg.

```
35 Public Sub btnSave_Click()
36
37  hSVGImage.Save(Application.Path &/ "sun2.svg")
38  Message.Info("SVG image saved")
39
40 End
```

Η υπορουτίνα btnSave_Click() καλείται όταν πατηθεί το κουμπί btnSave.
Στην γραμμή 37 αποθηκεύουμε την εικόνα στο αρχείο sun2.svg
Και στην γραμμή 38 εμφανίζουμε ένα μήνυμα προς τον χρήστη.

Δουλεύοντας με λίστες, η κλάση ListView.

Το στοιχείο ελέγχου αυτό υλοποιεί μια λίστα κειμένου όπου κάθε μέλος της λίστας μπορεί να έχει και μια εικόνα.
Τα στοιχεία της λίστας είναι τοποθετημένα σε ευρετήριο (index) και το κάθε ένα έχει ένα δικό του μοναδικό κλειδί ακριβώς όπως τα στοιχεία σε έναν πίνακα.
Το control έχει ένα εσωτερικό δείκτη που χρησιμοποιείται για την προσπέλαση των στοιχείων της λίστας. Χρησιμοποιείστε τις μεθόδους MoveFirst, MoveNext, MoveTo... για να μετακινήσετε τον εσωτερικό δείκτη και την ιδιότητα Item για να προσπελάσετε το στοιχείο στο οποίο δείχνει.
Η κλάση αυτή συμπεριφέρεται σαν ένας πίνακας μόνο για ανάγνωση.
Η κλάση μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hListView AS ListView

hListView = NEW ListView ( Parent AS Container ) [ AS "event name" ]
```

Η κλάση αυτή κληρονομεί την κλάση Control.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Control		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Background Cursor Design Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show	DblClick Drag DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Listview		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Available Border ClientH ClientHeight ClientW ClientWidth Compare Count Current Editable Item Key Mode Renaming ScrollBar Sorted	Add Clear Exist Find MoveAbove MoveBack MoveBelow MoveCurrent MoveFirst MoveLast MoveNext MovePrevious MoveTo Remove SelectAll UnselectAll	Activate Cancel Click Compare Rename Select

Εφαρμογή 1
Το Project119 δείχνει ένα παράδειγμα με ένα αντικείμενο Listview.

Η γραφική διεπαφή του προγράμματος αποτελείται από 3 Labels που δίνουν πληροφορίες στον χρήστη, ένα αντικείμενο Listview, ένα αντικείμενο Textarea και τρία Buttons.

Εικόνα 108: Η γραφική διεπαφή του Project101

Στο πρόγραμμα αυτό θέλουμε λέξεις που διαχωρίζονται με κόμμα από το στοιχείο ελέγχου Textarea να προστεθούν ως αντικείμενα στο στοιχείο ελέγχου Listview.

Το πρώτο κουμπί καθαρίζει τις τιμές των αντικειμένων Textarea, Listview, το δεύτερο κουμπί προσθέτει τις λέξεις ως στοιχεία στην λίστα και το τρίτο κουμπί κλείνει την εφαρμογή.

Ο κώδικας της εφαρμογής είναι:

```
1   ' Gambas class file
2
3   Public Sub Form_Open()
4
5   Me.Center
6
7   End
8
9   Public Sub btnClose_Click()
10
11  Me.Close
12
13  End
14
15  Public Sub btnAddItems_Click()
16
17    Dim sWords As New String[]
18    Dim i As Integer
19
```

```
20    sWords = Split(txaItems.Text, ",")
21
22    For i = 0 To sWords.Count - 1
23      If lswItems.Exist(i) = False Then
24      lswItems.Add(i, Trim(sWords[i]),
Picture["krig_Check_item_Aqua_style.png"])
25      End If
26    Next
27
28
29  End
30
31  Public Sub btnClear_Click()
32
33    txaItems.Clear
34    lswItems.Clear
35
36  End
```

Στις γραμμές 3-7 έχουμε την ρουτίνα `Form_Open()` που εκτελείται όταν η φόρμα ανοίγει. Στην γραμμή 5 τοποθετούμε την φόρμα στο κέντρο της οθόνης.

Στις γραμμές 9-13 έχουμε την ρουτίνα `btnClose_Click()` που εκτελείται όταν πατηθεί το κουμπί btnClose. Στην γραμμή 11 κλείνουμε την φόρμα.

Στις γραμμές 15-29 έχουμε την ρουτίνα `btnAddItems_Click()` που εκτελείται όταν πατηθεί το κουμπί btnAddItems. Στην γραμμή 17 ορίζουμε έναν δυναμικό πίνακα συμβολοσειράς τον `sWords` και στην γραμμή 18 μια μεταβλητή εκεραίου.

Στην γραμμή 20 εκχωρούμε στον πίνακα `sWords` το αποτέλεσμα διαχωρισμού του κειμένου `txaItems.Text` με διαχωριστικό το κόμμα. Η συνάρτηση Split επιστρέφει έναν πίνακα και έτσι αυτός εκχωρείται στον πίνακα `sWords`.

Στην συνέχεια θα προσπελάσουμε ένα ένα τα στοιχεία του πίνακα `sWords` και θα τα τοποθετήσουμε σαν μέλη του αντικειμένου `lswItems`.

Για να το κάνουμε αυτό χρησιμοποιούμε ένα βρόχο For...Next. Η ακέραια μεταβλητή i χρησιμοποιείται σαν μετρητής του βρόχου. Η απαρίθμηση γίνεται από το στοιχείο 0 μέχρι το τελευταίο στοιχείο που είναι το `sWords.Count - 1`.

Εντός του βρόχου ελέγχουμε αν υπάρχει στην λίστα ήδη κάποιο στοιχείο στην γραμμή 23. Αν δεν υπάρχει το προσθέτουμε στην λίστα με την γραμμή 24.

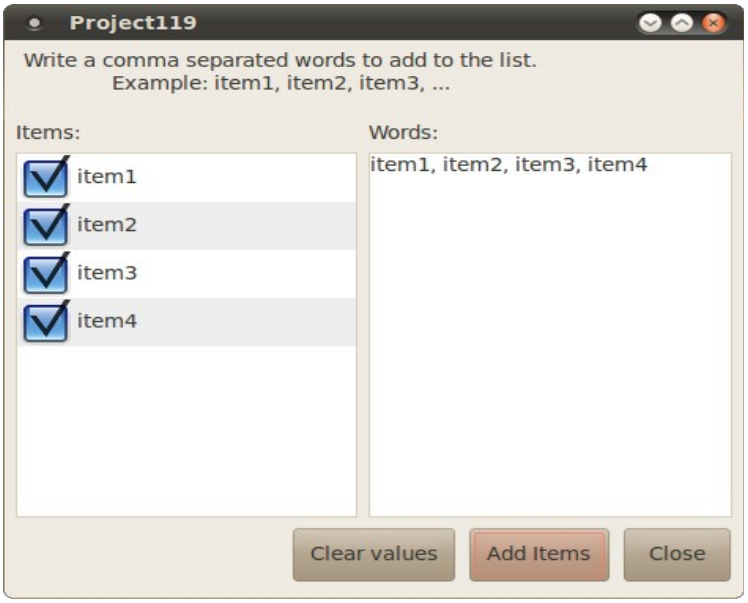
```
lswItems.Add(i, Trim(sWords[i]), Picture["krig_Check_item_Aqua_style.png"])
```

Η μέθοδος Add του αντικειμένου lswItems δέχεται σαν παραμέτρους τα εξής:

- i, το κλειδί του νέου μέλους
- `Trim(sWords[i])`, το κείμενο χωρίς χαρακτήρες space από κάθε στοιχείο του πίνακα `sWords`.

- `Picture["krig_Check_item_Aqua_style.png"])`, μια εικόνα που εμφανίζεται δίπλα στο κείμενο κάθε μέλους στην λίστα.

Τέλος στις γραμμές 31-36 είναι η ρουτίνα `btnClear_Click()` που εκτελείται όταν πατηθεί το κουμπί `btnClear`. Στην ρουτίνα αυτή καθαρίζονται οι τιμές των αντικειμένων `txaItems` και `lswItems`.



Εικόνα 109: Η εφαρμογή κατά την εκτέλεση

Δουλεύοντας με κουτιά από λίστες, η κλάση ListBox

Η κλάση υλοποιεί μια λίστα από μέλη με κείμενο. Τα οποία μέλη της λίστας μπορούν να επιλεχθούν.

Τα στοιχεία της λίστας είναι τοποθετημένα σε ευρετήριο (index) και το κάθε ένα έχει ένα δικό του μοναδικό κλειδί ακριβώς όπως τα στοιχεία σε έναν πίνακα.

Η κλάση αυτή συμπεριφέρεται σαν ένας πίνακας μόνο για ανάγνωση.
Η κλάση μπορεί να δημιουργηθεί.

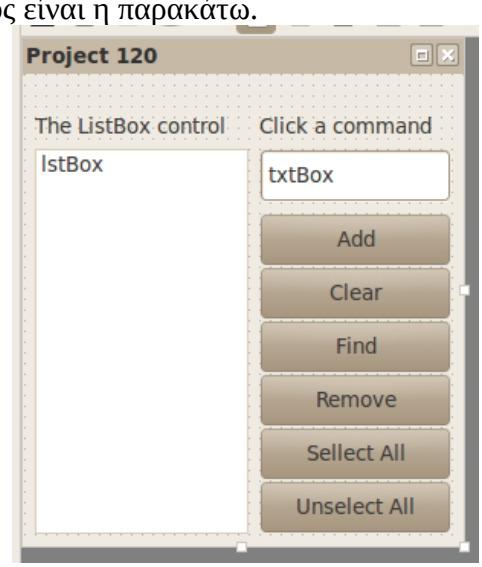
Σύνταξη:

```
DIM hListBox AS ListBox

hListBox = NEW ListBox ( Parent AS Container ) [ AS "event name" ]
```

Η κλάση αυτή κληρονομεί την κλάση Control.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Control		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Background Cursor Design Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show	DblClick Drag DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Listview		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Border Count Current Index List Mode Sorted Text	Add Clear Find Remove SelectAll UnselectAll	Activate Click Select

Εφαρμογή 1
Το Project120 δείχνει ένα παράδειγμα με ένα αντικείμενο Listbox.
Η διεπαφή του προγράμματος είναι η παρακάτω.



Εικόνα 110: Η διεπαφή του Project102

Αποτελείται από ένα ListBox, ένα TextBox και έξι Buttons. Κάθε κουμπί υλοποιεί μια από τις έξι μεθόδους του control. Για παράδειγμα αν θέλουμε να προσθέσουμε ένα στοιχείο στην λίστα γράφουμε το κείμενο στο textbox και πατάμε Add. Αν θέλουμε να βρούμε ένα στοιχείο της λίστας γράφουμε το κείμενο στο textbox και πατάμε το κουμπί Find. Μπορούμε να επιλέξουμε διάφορα στοιχεία από την λίστα και να τα αφαιρέσουμε με το κουμπί Remove.

Ο κώδικας της εφαρμογής είναι:

```
1   ' Gambas class file
2
3   Public Sub Form_Open()
4
5     Me.Center
6
7   End
8
9   Public Sub btnAdd_Click()
10
11    lstBox.Add(txtBox.Text)
12
13  End
14
15  Public Sub btnClear_Click()
16
17    lstBox.Clear
18
19  End
20
21  Public Sub btnFind_Click()
22
23    Dim iKey As Integer
24    iKey = lstBox.Find(txtBox.Text)
25    If iKey <> -1 Then
26      lstBox.Index = iKey
27    Else
28      Message("Nothing found")
29    End If
30
31  End
32
33  Public Sub btnRemove_Click()
34
35    Dim i As Integer
36
37    While i < lstBox.Count
38        'is it selected?
39       If lstBox[i].Selected Then
40          'remove
41         lstBox.Remove(i)
42         Dec i 'decrease relevant index
43       Endif
```

```
44      Inc i 'next item in lstBox
45    Wend
46
47  End
48
49  Public Sub btnSelectAll_Click()
50
51    lstBox.SelectAll
52
53  End
54
55  Public Sub btnUnsellectAll_Click()
56
57    lstBox.UnselectAll
58
59  End
```

Στις γραμμές 3-7 τοποθετούμε το παράθυρο στο κέντρο της οθόνης με το που φορτώνεται η φόρμα και καλείται η ρουτίνα για το συμβάν Form_Open().

Στις γραμμές 9-13 υπάρχει ο κώδικας για το συμβάν btnAdd_Click(). Όταν πατηθεί το κουμπί Add καλείται αυτή η ρουτίνα η οποία προσθέτει ένα στοιχείο στο lstBox.

Η γραμμή lstBox.Add(txtBox.Text) προσθέτει ένα στοιχείο στην λίστα το οποίο είναι το κείμενο του textbox txtBox.

Στις γραμμές 15-19 καλείται η ρουτίνα btnClear_Click() όταν πατηθεί το κουμπί Clear.
Η ρουτίνα αυτή καθαρίζει το ListBox με την μέθοδο lstBox.Clear.

Στις γραμμές 21-29 καλείται η ρουτίνα btnFind_Click() όταν πατηθεί το κουμπί Find.
Το lstBox συμπεριφέρεται σαν ένας πίνακας με μια λίστα στοιχείων. Κάθε στοιχείο έχει ένα κλειδί (index) και μια τιμή, όπως όλοι οι πίνακες.

Η μέθοδος Find του lstBox επιστρέφει το κλειδί από το στοιχείο που θα βρεθεί από την αναζήτηση στην λίστα βάσει του κειμένου txtBox.Text. Αυτό κάνει η γραμμή 24.
iKey = lstBox.Find(txtBox.Text)
Αν δεν βρεθεί το στοιχείο στην λίστα τότε η Find επιστρέφει -1.
Έτσι στην γραμμή 25 ελέγχουμε την τιμή της μεταβλητής iKey. Αν δεν είναι -1 τότε θέτουμε σαν τρέχουσα επιλογή το στοιχείο από την λίστα που βρέθηκε.
Αν είναι -1 ενημερώνουμε τον χρήστη ότι δεν έχει βρεθεί η τιμή που αναζητά.

Στις γραμμές 33-47 καλείται η ρουτίνα btnRemove_Click() όταν πατηθεί το κουμπί Remove.
Στην ρουτίνα αυτή εκτελούμε έναν βρόχο While...Wend όσο η τιμή i είναι μικρότερη από το πλήθος των στοιχείων στο lstBox.
Όταν αφαιρούμε ένα αντικείμενο από την λίστα αμέσως το πλήθος των στοιχείων μειώνεται κατά ένα. Αν η μεταβλητή i δεν μειώνεται και αυτή τότε κατά την αρίθμηση του πίνακα θα

βγούμε εκτός ορίων το οποίο θα οδηγήσει στην κατάρρευση του προγράμματος με το μήνυμα λάθους “Bad Index”. Για τον λόγο αυτό κάθε φορά που το πρόγραμμα βρίσκει ένα επιλεγμένο στοιχείο και το αφαιρεί εμείς θα πρέπει να αναπροσαρμόζουμε την τιμή της μεταβλητής i.

Στην γραμμή 39 ελέγχουμε αν ένα στοιχείο στην λίστα είναι επιλεγμένο.
Αν είναι το αφαιρούμε και μειώνουμε την τιμή της μεταβλητής i.

Στην γραμμή 44 αυξάνουμε την τιμή της μεταβλητής i για να ελέγξουμε το επόμενο στοιχείο στην λίστα.

Στις γραμμές 49-53 καλείται η ρουτίνα btnSelectAll_Click() όταν πατηθεί το κουμπί Select All η οποία επιλέγει όλα τα αντικείμενα.

Στις γραμμές 55-59 καλείται η ρουτίνα btnUnsellectAll_Click() όταν πατηθεί το κουμπί Unselect All η οποία καταργεί την επιλογή σε όλα τα αντικείμενα.

Δουλεύοντας με αναδυόμενα κουτιά επιλογών, η κλάση ComboBox

Η κλάση υλοποιεί ένα text box που συνδυάζει και μια αναδυόμενη λίστα με στοιχεία κειμένου.

Τα στοιχεία της λίστας είναι τοποθετημένα σε ευρετήριο (index) και το κάθε ένα έχει ένα δικό του μοναδικό κλειδί ακριβώς όπως τα στοιχεία σε έναν πίνακα.

Η κλάση αυτή συμπεριφέρεται σαν ένας πίνακας μόνο για ανάγνωση.
Η κλάση μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hComboBox AS ComboBox

hComboBox = NEW ComboBox ( Parent AS Container ) [ AS "event name" ]
```

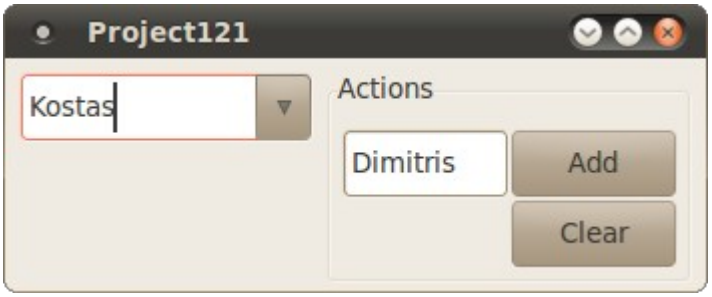
Η κλάση αυτή κληρονομεί την κλάση Control.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Control		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Background Cursor Design Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show	DblClick Drag DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Listview		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Count Current Index Length List MaxLength Password Pos ReadOnly Selected Selection Sorted Text	Add Clear Find Insert Popup Remove Select SelectAll Unselect	Activate Change Click

Εφαρμογή 1
Το παράδειγμα αυτό είναι το Project121.
Στην φόρμα FMain υπάρχει ένα combobox με το όνομα cmbList και ένα Frame που περιέχει ένα textbox και δύο κουμπιά. Το ένα κουμπί προσθέτει ένα στοιχείο στο combobox ενώ το άλλο αφαιρεί όλα τα στοιχεία από το combobox.
Όταν ο χρήστης επιλέξει ένα στοιχείο από το combobox εμφανίζεται ένα μήνυμα.

Τα στοιχεία ελέγχου που περιέχει η φόρμα είναι:

Όνομα	**Περιγραφή**
cmbList	Το combobox
Frame1	Το πλαίσιο που περιέχει το textbox και τα κουμπιά
txtAdd	Το textbox
btnAdd	Το κουμπί προσθήκης
btnClear	Το κουμπί καθαρισμού



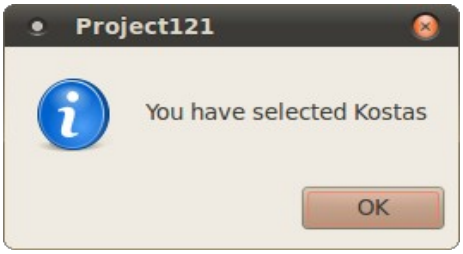
Εικόνα 111: Η εφαρμογή κατά την λειτουργία της

Ο κώδικας της εφαρμογής είναι ο παρακάτω:

```
1   ' Gambas class file
2
3   Public Sub Form_Open()
4
5     Me.Center
6
7   End
8
9   Public Sub btnAdd_Click()
10
11    cmbList.Add(txtAdd.Text)
12
13  End
14
15  Public Sub btnClear_Click()
16
17    cmbList.Clear
18
19  End
20
21  Public Sub cmbList_Click()
22
23    Message.Info("You have selected " & cmbList.Current.Text)
24
25  End
```

Ανάλυση:
Στην γραμμή 5 τοποθετούμε την φόρμα στο κέντρο της οθόνης κατά το άνοιγμα της φόρμας.
Στις γραμμές 9 – 13 βρίσκεται η ρουτίνα του κουμπιού btnAdd όταν κάνουμε σε αυτό κλικ με το ποντίκι.
Στην γραμμή 11 προσθέτουμε ένα στοιχείο στο combobox όταν πατηθεί το κουμπί btnAdd.
Στις γραμμές 15 – 19 βρίσκεται η ρουτίνα του κουμπιού btnClear όταν κάνουμε σε αυτό κλικ με το ποντίκι.
Στην γραμμή 17 αφαιρούμε όλα τα στοιχεία από το combobox όταν πατηθεί το κουμπί btnClear.
Στις γραμμές 21 – 25 βρίσκεται η ρουτίνα του combobox που καλείται όταν ο χρήστης επιλέξει ένα στοιχείο από αυτό.
Στην γραμμή 23 εμφανίζουμε ένα μήνυμα στον χρήστη που περιέχει το τρέχον κείμενο από το επιλεγμένο στοιχείο του combobox.

Στο παράδειγμα μας έχουμε προσθέσει σαν στοιχεία στο combobox ονόματα και όταν επιλέγουμε ένα όνομα εμφανίζεται το παρακάτω μήνυμα.



Εικόνα 112: Το μήνυμα που εμφανίζεται στον χρήστη ύστερα από την επιλογή του στοιχείου του combobox

Δουλεύοντας με λίστες πλέγματος, η κλάση GridView

Η κλάση υλοποιεί ένα στοιχείο ελέγχου το οποίο εμφανίζει δεδομένα σε ένα πλέγμα.
Μπορείτε να γεμίσετε το πλέγμα είτε προσπελάζοντας ένα ένα τα κελιά του πλέγματος ή απευθείας με το συμβάν Data.
Η τελευταία μέθοδος ενδεικνύεται όταν υπάρχουν χιλιάδες εγγραφές που πρέπει να περαστούν στο πλέγμα με κάποιο αυτόματο τρόπο.
Αν θέλετε να μπορείτε να επεξεργαστείτε ένα κελί τότε θα πρέπει να χρησιμοποιήσετε το control TableView του εξαρτήματος gb.form.
Η κλάση αυτή συμπεριφέρεται σαν ένας πίνακας μόνο για ανάγνωση.
Η κλάση μπορεί να δημιουργηθεί.

Σύνταξη:

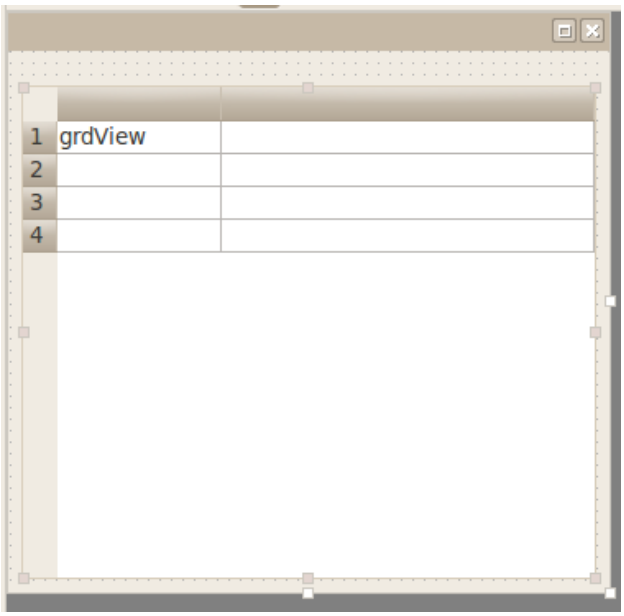
```
DIM hGridView AS GridView

hGridView = NEW GridView ( Parent AS Container ) [ AS "event name" ]
```

Η κλάση αυτή κληρονομεί την κλάση Control.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Control		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Background Cursor Design Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show	DblClick Drag DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης GridView		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
AutoResize Border ClientH ClientHeight ClientW ClientWidth ClientX ClientY Column Columns Current Data Grid Header Mode Resizable Row Rows ScrollBar ScrollX ScrollY	Clear ColumnAt Find MoveTo RowAt	Activate Change Click ColumnClick ColumnResize Data RowClick RowResize Scroll Select
Σταθερές (Constants)	Both Horizontal None Vertical	

Εφαρμογή 1
Το παράδειγμα είναι το Project122.
Η φόρμα FMain έχει ένα GridView control.
Έχουμε θέσει την ιδιότητα Header στην τιμή Both ώστε να εμφανίζονται οι επικεφαλίδες των γραμμών και των στηλών.

Η διεπαφή του προγράμματος είναι η εξής:



Εικόνα 113: Η διεπαφή του Project122

Στο άνοιγμα της φόρμας κάνουμε setup το control αυτό.

```
1   ' Gambas class file
2
3   Public Sub Form_Open()
4   'setup gridview
5   grdView.Columns.Count = 2
6   grdView.Rows.Count = 4
7   grdView.Columns.Width = 52
8   grdView.Rows.Height = 52
9
10  'setup headers
11  grdView.Columns[0].Text = "Picture"
12  grdView.Columns[1].Text = "Text"
13
14  'add rows [row, column]
15  grdView[0, 0].Picture = Picture["green32.png"]
16  grdView[0, 1].Text = "green"
```

```
17
18  grdView[1, 0].Picture = Picture["blue32.png"]
19  grdView[1, 1].Text = "blue"
20
21  grdView[2, 0].Picture = Picture["orange32.png"]
22  grdView[2, 1].Text = "orange"
23
24  grdView[3, 0].Picture = Picture["red32.png"]
25  grdView[3, 1].Text = "red"
26
27  End
```

Ανάλυση:
Ο κώδικας έχει γραφτεί στην ρουτίνα Form_Open() έτσι ώστε να εκτελείται κατά το άνοιγμα της φόρμας και να αρχικοποιεί το στοιχείο ελέγχου GridView.

Στις γραμμές 5,6 ορίζουμε το πλήθος των στηλών και των γραμμών.
Στις γραμμές 7,8 ορίζουμε το πλάτος των στηλών και το ύψος των γραμμών αντίστοιχα.

Επειδή στις ιδιότητες έχουμε ορίσει να έχουμε επικεφαλίδες (Headers) στις γραμμές 11,12 ορίζουμε το κείμενο των επικεφαλίδων αυτών. Κάθε επικεφαλίδα προσπελάζεται ως ανάγνωση ενός πίνακα. Ο πίνακας αυτός είναι η εικονική κλάση Columns.
Έτσι για να προσπελάζουμε την πρώτη επικεφαλίδα γράφουμε grdView.Columns[0] για την δεύτερη grdView.Columns[1] κτλ. Η εικονική κλάση Columns μας παρέχει στην συνέχεια τις ιδιότητες της κάθε επικεφαλίδας όπως το κείμενο, για παράδειγμα grdView.Columns[0].Text = "Picture"

Στις γραμμές 14 με 25 προσθέτουμε μία μία γραμμή.
Στην πρώτη στήλη θέλουμε να εμφανίζουμε ένα εικονίδιο ενώ στην δεύτερη ένα κείμενο.
Η προσπέλαση των γραμμών γίνεται όμοια με την προσπέλαση ενός πίνακα με διαστάσεις [x,y] όπου x είναι το πλήθος των γραμμών και y το πλήθος των στηλών.
Έτσι για παράδειγμα το κελί στην πρώτη γραμμή (x=0) και πρώτη στήλη (y=0) προσπελάζεται ως grdView[0, 0], για το κελί στην πρώτη γραμμή (x=0) και δεύτερη στήλη (y=1) έχουμε αντίστοιχα grdView[0, 1] κτλ.

Κάθε κελί μας δίνει τις δικές του ιδιότητες μέσω της εικονικής κλάσης .GridViewCell. Παρακάτω αναφέρονται τα μέλη της εικονικής κλάσης αυτής.

Κλάση	.GridViewCell
Ιδιότητες	Alignment BackColor Background Font ForeColor Foreground H Height Left Padding Picture RichText Text Top W Width X Y
Μέθοδοι	Clear EnsureVisible Refresh

Έτσι η γραμμές 15, 18, 21, 24 θέτουν μια εικόνα στα κελιά της πρώτης στήλης ενώ οι γραμμές

16, 19, 22, 25 θέτουν ένα κείμενο στα κελιά της δεύτερης στήλης.

Εφαρμογή 2
Το παράδειγμα είναι το Project123.
Η φόρμα FMain έχει ένα GridView control.
Έχουμε θέσει την ιδιότητα Header στην τιμή Both ώστε να εμφανίζονται οι επικεφαλίδες των γραμμών και των στηλών.

Η διεπαφή του προγράμματος είναι η εξής:

Εικόνα 114: Η διεπαφή του Project123

Στο άνοιγμα της φόρμας κάνουμε setup το control αυτό και θέτουμε τις τιμές των κελιών με το συμβάν Data.

```
1   ' Gambas class file
2
3   Private sNames As New String[]
4
5   Public Sub Form_Open()
6
7   Dim i As Integer
8   sNames.Add("Nikos")
9   sNames.Add("Petros")
10  sNames.Add("Dimos")
11  sNames.Add("Kostas")
12  sNames.Add("Dimitris")
13  sNames.Add("Vagelis")
```

```
14  sNames.Add("Giannis")
15  sNames.Add("Giorgos")
16
17  'setup gridview
18  grdView.Columns.Count = 2
19  grdView.Rows.Count = sNames.Count
20  grdView.Columns.Width = 52
21  grdView.Rows.Height = 52
22
23  'setup headers
24  grdView.Columns[0].Text = "Picture"
25  grdView.Columns[1].Text = "Text"
26
27  'add rows [row, column]
28  For i = 0 To sNames.Count - 1
29    'setup first column
30    grdView_Data(i, 0)
31
32    'setup second column
33    grdView_Data(i, 1)
34  Next
35
36  End
37
38  Public Sub grdView_Data(Row As Integer, Column As Integer)
39
40    If Column = 0 Then
41      grdView[Row, Column].Picture = Picture["blue32.png"]
42    Endif
43
44    If Column = 1 Then
45      grdView[Row, Column].Text = sNames[Row]
46    Endif
47
48  End
```

Ανάλυση:
Ο κώδικας έχει γραφτεί στην ρουτίνα Form_Open() έτσι ώστε να εκτελείται κατά το άνοιγμα της φόρμας και να αρχικοποιεί το στοιχείο ελέγχου GridView.

Στην γραμμή 3 δηλώνουμε ένα καθολικό και ιδιωτικό πίνακα τύπου αλφαρηθμικών με το όνομα sNames.

Στην γραμμή 7 δηλώνουμε μια μεταβλητή τύπου ακεραίου για να την χρησιμοποιήσουμε για την απαρίθμηση του πίνακα sNames.

Στις γραμμές 8-15 προσθέτουμε στοιχεία στον πίνακα sNames.
Στις γραμμές 17-21 αρχικοποιούμε το gridview control ορίζοντας το πλήθος των στηλών και των γραμμών του όπως και το μήκος και το ύψος αυτών.
Η γραμμή 19 θέτει ως πλήθος γραμμών το πλήθος των στοιχείων του πίνακα sNames.

Στις γραμμές 23-25 ορίζουμε το κείμενο των επικεφαλίδων των στηλών.

Στις γραμμές 28-34 υπάρχει ένας βρόχος επανάληψης FOR...NEXT.
Ο βρόχος μετρά από το 0 μέχρι το πλήθος των στοιχείων του πίνακα μείον 1.
Αυτό γίνεται διότι η αρίθμηση του πίνακα ξεκινά από το 0 και όχι από το 1.
Έτσι το η απαρίθμηση 8 στοιχείων θα γίνει από το στοιχείο 0 μέχρι το 7.
Εντός του βρόχου καλούμε την ρουτίνα grdView_Data().
Σαν παραμέτρους διοχετεύουμε την γραμμή και την στήλη οι οποίες ορίζουν πιο κελί θα προσπελαστεί.

Η γραμμή grdView_Data(i, 0) προσπέλαζει όλες τις γραμμές της πρώτης στήλης.
Ενώ η γραμμή grdView_Data(i, 1) προσπελάζει όλες τις γραμμές της δεύτερης στήλης.

Στις γραμμές 38-48 βρίσκεται η ρουτίνα grdView_Data(Row As Integer, Column As Integer).
Εδώ όταν η μεταβλητή Column έχει την τιμή 0, δηλαδή την πρώτη στήλη τότε εκτελούμε τον κώδικα ο οποίος προσθέτει μια εικόνα σε όλες τις γραμμές της πρώτης στήλης.

Όταν η μεταβλητή Column έχει την τιμή 1, δηλαδή την δεύτερη στήλη τότε εκτελούμε τον κώδικα ο οποίος προσθέτει το κείμενο του κάθε στοιχείου του πίνακα sNames σε όλες τις γραμμές της δεύτερης στήλης.

Με παρόμοιο τρόπο μπορούμε να προσπελάσουμε κάθε πίνακα όπως τα αντικείμενα Result που περιέχουν τις απαντήσεις από ερωτήματα (queries) σε βάσεις δεδομένων.

Δουλεύοντας με λίστες δένδρου, η κλάση TreeView

Η κλάση υλοποιεί ένα στοιχείο ελέγχου το οποίο εμφανίζει δεδομένα σε μορφή δένδρου με κείμενο και εικονίδια.
Τα στοιχεία του δένδρου βρίσκονται σε ευρετήριο (index) και έχουν ένα κλειδί (key) όπως ακριβώς ένας πίνακας. Κάθε στοιχείο έχει ένα εικονίδιο και ένα κείμενο.
Το στοιχείο ελέγχου έχει έναν εσωτερικό δείκτη για να προσπελάζει τα στοιχεία του.
Μπορείτε να μετακινείτε τον εσωτερικό δείκτη με τις μεθόδους `MoveFirst`, `MoveNext`, `MoveTo`
Η κλάση αυτή συμπεριφέρεται σαν ένας πίνακας μόνο για ανάγνωση.
Η κλάση μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hTreeView AS TreeView

hTreeView = NEW TreeView ( Parent AS Container ) [ AS "event name" ]
```

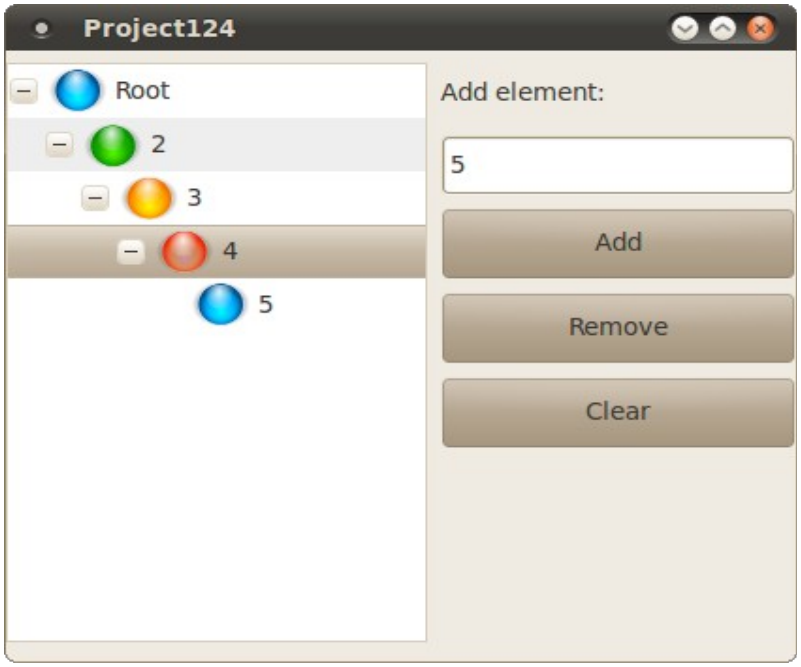
Η κλάση αυτή κληρονομεί την κλάση Control.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Control		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Background Cursor Design Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show	DblClick Drag DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης TreeView		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Available Border ClientH ClientHeight ClientW ClientWidth Compare Count Current Editable Item Key Mode Renaming ScrollBar Sorted	Add Clear Exist Find MoveAbove MoveBack MoveBelow MoveChild MoveCurrent MoveFirst MoveLast MoveNext MoveParent MovePrevious MoveTo Remove SelectAll Unselectall	Activate Cancel Click Collapse Compare Expand Rename Select

Εφαρμογή 1
Το παράδειγμα είναι το Project124.
Η φόρμα FMain έχει ένα TreeView control.
Ο χρήστης μπορεί να προσθέτει, να αφαιρεί στοιχεία στο Treview control όπως και να το καθαρίζει από όλα τα στοιχεία. Κάθε στοιχείο έχει ένα εικονίδιο και ένα κείμενο. Το κείμενο μπορεί να το ορίζει ο χρήστης από ένα textbox ενώ τα εικονίδια ανάλογα το χρώμα τους ορίζουν το επίπεδο στο οποίο τα στοιχεία έχουν καταχωρηθεί. Για παράδειγμα τα στοιχεία πρώτου επιπέδου έχουν εικονίδιο μια γαλάζια σφαίρα, του δευτέρου μια πράσινη, του τρίτου επιπέδου μια πορτοκαλί σφαίρα και του τέταρτου επιπέδου μια κόκκινη. Τα εικονίδια των επόμενων επιπέδων επαναλαμβάνονται με αυτήν την σειρά.

Η γραφική διεπαφή αποτελείται από τα εξής controls:

Control	**Περιγραφή**
trvList	Το treeview control
Label1	Ετικέτα κειμένου
txtElement	Το textbox εισαγωγής στοιχείων στο control treeview
btnAdd	Το κουμπί προσθήκης
btnRemove	Το κουμπί αφαίρεσης
btnClear	Το κουμπί καθαρισμού



Εικόνα 115: Η γραφική διεπαφή του Project124

Ο κώδικας της εφαρμογής είναι:

```
1   ' Gambas class file
2
3   Private pic1 As Picture
4   Private pic2 As Picture
5   Private pic3 As Picture
6   Private pic4 As Picture
7
8   Public Sub Form_Open()
9     Me.Center
10
11    pic1 = Picture["blue32.png"]
12    pic2 = Picture["green32.png"]
13    pic3 = Picture["orange32.png"]
14    pic4 = Picture["red32.png"]
15
16    trvList.Add("root", "Root", pic1)
17
18  End
19
20  Public Sub btnAdd_Click()
21
22    If trvList.Exist(txtElement.Text) Then
23      Message.Error("Element exists!")
24      Return
25    Endif
26
27    If IsNull(txtElement.Text) = True Then
28      Message.Error("Set some text")
29      Return
30    Endif
31
32    If trvList.Current.Picture = pic1 Then
33        trvList.Add(txtElement.Text, txtElement.Text, pic2,
    trvList.Current.Key)
34    Endif
35
36    If trvList.Current.Picture = pic2 Then
37        trvList.Add(txtElement.Text, txtElement.Text, pic3,
    trvList.Current.Key)
38    Endif
39
40    If trvList.Current.Picture = pic3 Then
41        trvList.Add(txtElement.Text, txtElement.Text, pic4,
    trvList.Current.Key)
```

```
42    Endif
43
44    If trvList.Current.Picture = pic4 Then
45        trvList.Add(txtElement.Text, txtElement.Text, pic1,
trvList.Current.Key)
46    Endif
47
48  End
49
50  Public Sub btnRemove_Click()
51
52    If trvList.Current.key = "root" Then
53      Message.Error("Cannot remove root")
54      Return
55    Endif
56
57    trvList.Remove(trvList.Current.Key)
58
59  End
60
61  Public Sub btnClear_Click()
62
63    trvList.Clear
64    trvList.Add("root", "Root", pic1)
65
66  End
```

Ανάλυση:
Στις γραμμές 3 – 6 δηλώνουμε τέσσερα καθολικά και ιδιωτικά αντικείμενα τύπου Picture.
Στην γραμμή 8 ξεκινάει η ρουτίνα φόρτωσης της φόρμας FMain.
Στην γραμμή 9 τοποθετούμε την φόρμα στο κέντρο της οθόνης.
Στις γραμμές 11-14 θέτουμε στα αντικείμενα εικόνας ένα αρχείο εικόνας.
Το αρχείο αυτό είναι μια χρωματιστή σφαίρα που θα χρησιμοποιηθεί για εικονίδιο σε κάθε επίπεδο του treeview control.

Στην γραμμή 16 θέτουμε το αρχικό επίπεδο Root στο treeview control. Τα υπόλοιπα επίπεδα και στοιχεία του στοιχείου ελέγχου θα υλοποιηθούν με βάση αυτό το αρχικό επίπεδο.
Αν δεν υπάρχει κάποιο αρχικό πατρικό επίπεδο, οι γραμμές προσθήκης στοιχείων 33, 37, 41, 45 που χρησιμοποιούν ως παράμετρο πατρικού στοιχείου την τιμή του τρέχοντος επιπέδου trvList.Current.Key θα αποτύχουν. Έτσι πάντα θα πρέπει να υπάρχει ένα τουλάχιστον στοιχείο στο treeview control και αυτό το στοιχείο το έχουμε ονομάσει root.

Στην γραμμή 20 ξεκινά η ρουτίνα του κουμπιού btnAdd με το συμβάν Click.

Στις γραμμές 22-25 γίνεται ένας έλεγχος εάν προσπαθούμε να εισάγουμε ένα στοιχείο που ήδη υπάρχει. Η έκφραση trvList.Exist(txtElement.Text) ελέγχει αν υπάρχει κάποιο στοιχείο με κλειδί το κείμενο του control txtElement. Αν υπάρχει ήδη κάποιο στοιχείο η γραμμή 23 εμφανίζει ένα μήνυμα λάθους και η γραμμή 24 επιστρέφει την ροή του προγράμματος εκτός ρουτίνας έτσι ο κώδικας που βρίσκεται παρακάτω και εισάγει ένα νέο στοιχείο ελέγχου δεν εκτελείται.

Στις γραμμές 27 – 30 γίνεται έλεγχος αν το κείμενο του txtElement είναι κενό. Αν είναι κενό δεν μπορεί να εισαχθεί ένα νέο στοιχείο στο treeview control. Αν δεν υπάρχει αυτός ο έλεγχος και προσπαθήσουμε να εισάγουμε ένα νέο στοιχείο τότε ένα λάθος θα εμφανιστεί.
Στις γραμμές 32-34 ελέγχουμε ποια εικόνα υπάρχει στο τρέχον στοιχείο του treview control. Αν η εικόνα είναι αυτή του αντικειμένου pic1 τότε προσθέτουμε ένα στοιχείο στο treeview control με την εικόνα του δεύτερου επιπέδου δλδ την pic2.
Η σύνταξη και οι παράμετροι της μεθόδου trvList.Add() είναι:

Σύνταξη:

```
FUNCTION Add ( Key AS String, Text AS String [ , Picture AS Picture,
Parent AS String, After AS String ] ) AS .TreeViewItem
```

Έτσι έχουμε:

Όνομα Παραμέτρου	**Παράμετρος**
Key	txtElement.Text
Text	txtElement.Text
Picture	pic2
Parent	trvList.Current.Key

Η μέθοδος Add με τις παραμέτρους αυτές προσθέτει ένα νέο στοιχείο στο treeview control το οποίο έχει κλειδί και κείμενο την τιμή της ιδιότητας txtElement.Text, εικονίδιο την τιμή του αντικειμένου pic2 και πατρικό στοιχείο το τρέχον στοιχείο του treeview control.

Το τρέχον στοιχείο trvList.Current είναι ένα αντικείμενο της εικονικής κλάσης .TreeViewItem. Η κλάση αυτή έχει τις δικές της ιδιότητες και μεθόδους για κάθε ένα στοιχείο του treview control.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης .TreeViewItem	
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**
Children Count Editable Expanded H Height Key Parentkey Picture Selected Text W Width X Y	Clear Delete EnsureVisible MoveAfter MoveBefore MoveFirst MoveLast Rename

Οι γραμμές 36 – 46 προσθέτουν νέα στοιχεία στο control με διαφορετικά εικονίδια για κάθε επίπεδο. Ο έλεγχος του κάθε επιπέδου γίνεται βάσει της εικόνας του όπως εξηγήσαμε πιο πάνω. (Πρώτο επίπεδο εικονίδιο γαλάζια σφαίρα, δεύτερο πράσινη, τρίτο επίπεδο πορτοκαλί και τέταρτο επίπεδο κόκκινη.) Από το 6ο επίπεδο και μετά τα εικονίδια επαναλαμβάνονται.

Στην γραμμή 50 ξεκινά η ρουτίνα του κουμπιού btnRemove για το συμβάν Click.
Στις γραμμές 52 -55 ελέγχουμε αν το τρέχον στοιχείο είναι το πρώτο. Επειδή το πρώτο στοιχείο δεν πρέπει να αφαιρεθεί κατά τον έλεγχο αυτό αν γίνει κάτι τέτοιο εμφανίζουμε ένα μήνυμα λάθους και επιστρέφουμε την ροή του προγράμματος με την εντολή RETURN.
Σε κάθε άλλη περίπτωση που το τρέχον στοιχείο δεν είναι το πρώτο μπορούμε να το αφαιρέσουμε με την γραμμή 57, trvList.Remove(trvList.Current.Key).

Στην γραμμή 61 ξεκινά η ρουτίνα του κουμπιού btnClear για το συμβάν Click.
Στην γραμμή 63 αφαιρούμε όλα τα στοιχεία από το control.
Στην γραμμή 64 προσθέτουμε πάλι το πρώτο στοιχείο.

Δουλεύοντας με λίστες σε στήλες, η κλάση ColumnView

Η κλάση υλοποιεί ένα στοιχείο ελέγχου το οποίο εμφανίζει δεδομένα ενός treeview control σε στήλες.
Τα στοιχεία του columnview βρίσκονται σε ευρετήριο (index) και έχουν ένα κλειδί (key) όπως ακριβώς ένας πίνακας. Κάθε στοιχείο μπορεί στην πρώτη στήλη να έχει ένα εικονίδιο και σε όλες τις στήλες ένα κείμενο.
Το στοιχείο ελέγχου έχει έναν εσωτερικό δείκτη για να προσπελάζει τα στοιχεία του.
Μπορείτε να μετακινείτε τον εσωτερικό δείκτη με τις μεθόδους `MoveFirst`, `MoveNext`, `MoveTo`
Για να αλλάξετε τον αριθμό των στηλών και το layout τους χρησιμοποιήστε την ιδιότητα Columns.
Η κλάση αυτή συμπεριφέρεται σαν ένας πίνακας μόνο για ανάγνωση.
Η κλάση μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hColumnView AS ColumnView

hColumnView = NEW ColumnView ( Parent AS Container ) [ AS "event name" ]
```

Η κλάση αυτή κληρονομεί την κλάση Control.

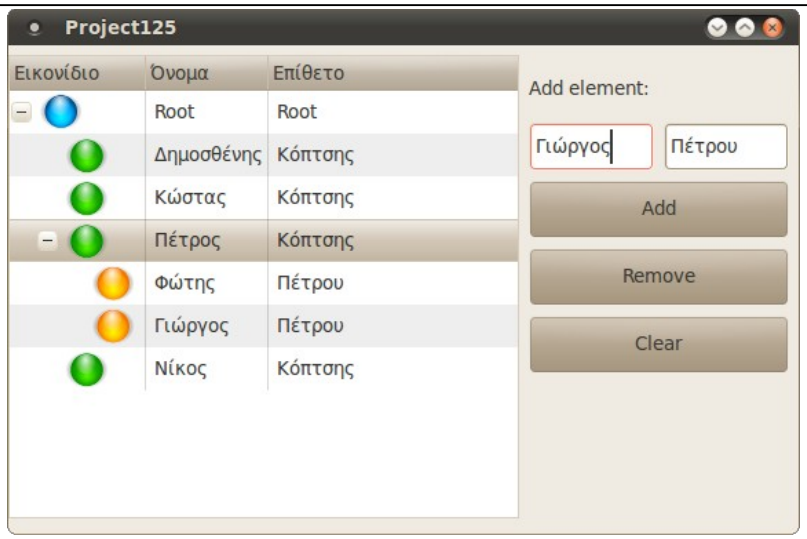
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Control		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Background Cursor Design Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show	DblClick Drag DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης ColumnView		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
AutoResize Available Border ClientH ClientHeight ClientW ClientWidth Columns Compare Count Current Editable Header Item Key Mode ./renaming Resizable ScrollBar Settings	Add Clear Exist Find MoveAbove MoveBack MoveBelow MoveChild MoveCurrent MoveFirst MoveLast MoveNext MoveParent MovePrevious MoveTo Remove SelectAll	Activate Cancel Click Collapse Compare Expand Rename Select

Sorted	Unselectall	

Εφαρμογή 1
Το παράδειγμα είναι το Project125.
Η φόρμα FMain έχει ένα ColumnView control με τρεις στήλες.
Ο χρήστης μπορεί να προσθέτει, να αφαιρεί στοιχεία στο ColumnView control όπως και να το καθαρίζει από όλα τα στοιχεία. Κάθε στοιχείο έχει ένα εικονίδιο στην πρώτη στήλη και κείμενο στις άλλες δύο. Το κείμενο μπορεί να το ορίζει ο χρήστης από δύο textboxes ενώ τα εικονίδια ανάλογα το χρώμα τους ορίζουν το επίπεδο στο οποίο τα στοιχεία έχουν καταχωρηθεί. Για παράδειγμα τα στοιχεία πρώτου επιπέδου έχουν εικονίδιο μια γαλάζια σφαίρα, του δευτέρου μια πράσινη, του τρίτου επιπέδου μια πορτοκαλί σφαίρα και του τέταρτου επιπέδου μια κόκκινη. Τα εικονίδια των επόμενων επιπέδων επαναλαμβάνονται με αυτήν την σειρά.

Η γραφική διεπαφή αποτελείται από τα εξής controls:

Control	**Περιγραφή**
clmView	Το columnview control
Label1	Ετικέτα κειμένου
txtElement1, txtElement2	Τα textboxεσ εισαγωγής στοιχείων στο control treeview
btnAdd	Το κουμπί προσθήκης
btnRemove	Το κουμπί αφαίρεσης
btnClear	Το κουμπί καθαρισμού



Εικόνα 116: Η γραφική διεπαφή του Project124

Ο κώδικας της εφαρμογής είναι:

```
1   ' Gambas class file
2
3   Private pic1 As Picture
4   Private pic2 As Picture
5   Private pic3 As Picture
6   Private pic4 As Picture
7
8   Public Sub Form_Open()
9     Me.Center
10
11    pic1 = Picture["blue32.png"]
12    pic2 = Picture["green32.png"]
13    pic3 = Picture["orange32.png"]
14    pic4 = Picture["red32.png"]
15
16    clmView.Columns.Count = 3
17    clmView.Columns[0].Text = "Εικονίδιο"
18    clmView.Columns[1].Text = "Όνομα"
19    clmView.Columns[2].Text = "Επίθετο"
20
21    clmView.Add("root", "", pic1)
22    clmView["root"][1] = "Root"
23    clmView["root"][2] = "Root"
24
25  End
```

```
26
27  Public Sub btnAdd_Click()
28
29    If clmView.Exist(txtElement1.Text) Then
30      Message.Error("Element exists!")
31      Return
32    Endif
33
34    If IsNull(txtElement1.Text) = True Then
35      Message.Error("Set some text")
36      Return
37    Endif
38
39    If clmView[clmView.Current.Key].Picture = pic1 Then
40        clmView.Add(txtElement1.Text, "", pic2,
    clmView.Current.Key)
41    Endif
42
43    If clmView[clmView.Current.Key].Picture = pic2 Then
44        clmView.Add(txtElement1.Text, "", pic3,
    clmView.Current.Key)
45    Endif
46
47    If clmView[clmView.Current.Key].Picture = pic3 Then
48        clmView.Add(txtElement1.Text, "", pic4,
    clmView.Current.Key)
49    Endif
50
51    If clmView[clmView.Current.Key].Picture = pic4 Then
52        clmView.Add(txtElement1.Text, "", pic1,
    clmView.Current.Key)
53    Endif
54
55    clmView[txtElement1.Text][1] = txtElement1.Text
56    clmView[txtElement1.Text][2] = txtElement2.Text
57
58  End
59
60  Public Sub btnRemove_Click()
61
62    If clmView.Current.key = "root" Then
63      Message.Error("Cannot remove root")
64      Return
65    Endif
66
```

```
67    clmView.Remove(clmView.Current.Key)
68
69  End
70
71  Public Sub btnClear_Click()
72
73    clmView.Clear
74    clmView.Add("root", "", pic1)
75    clmView["root"][1] = "Root"
76    clmView["root"][2] = "Root"
77
78  End
```

Ανάλυση:
Στις γραμμές 3 – 6 δηλώνουμε τέσσερα καθολικά και ιδιωτικά αντικείμενα τύπου Picture.
Στην γραμμή 8 αρχίζει η ρουτίνα της FMain όταν φορτώνεται.
Στην γραμμή 9 τοποθετούμε την φόρμα στο κέντρο της οθόνης.
Στις γραμμές 11-14 ορίζουμε τις τιμές των εικόνων στα αντικείμενα pic1-pic4.
Στην γραμμή 16 ορίζουμε το πλήθος των στηλών του control.
Στις γραμμές 17-19 θέτουμε τις τιμές των επικεφαλίδων του control.
Η ιδιότητα clmView.Columns συμπεριφέρεται ως ένας πίνακας για ανάγνωση ο οποίος επιστρέφει αντικείμενα τύπου .ColumnViewColumns. Η κλάση .ColumnViewColumns είναι μια εικονική κλάση για κάθε στήλη του control.

Στην γραμμή 21 προσθέτουμε το αρχικό στοιχείο με ονομασία κλειδιού root.
Το στοιχείο αυτό δεν θέλουμε να έχει κείμενο και έτσι αφήνουμε την δεύτερη παράμετρο του κενή ενώ θέλουμε να έχει εικονίδιο το αντικείμενο pic1.
Στις γραμμές 22, 23 ορίζουμε το κείμενο των στηλών του αρχικού στοιχείου root.
Το αντικείμενο clmView συμπεριφέρεται ως ένας πίνακας ο οποίος προσπελάζεται βάσει των κλειδιών του. Έτσι για να προσπελάσουμε το πρώτο στοιχείο γράφουμε clmView["root"].
Στην συνέχεια για να προσπελάσουμε κάθε στήλη γράφουμε μέσα σε αγκύλες τον αριθμό της στήλης. Έτσι το clmView["root"][1] επιστρέφει την πρώτη στήλη του στοιχείου root.

Στις γραμμές 29-32 γίνεται ένας έλεγχος εάν προσπαθούμε να εισάγουμε ένα στοιχείο που ήδη υπάρχει. Η έκφραση clmView.Exist(txtElement1.Text) ελέγχει αν υπάρχει κάποιο στοιχείο με κλειδί το κείμενο του control txtElement1. Αν υπάρχει ήδη κάποιο στοιχείο η γραμμή 30 εμφανίζει ένα μήνυμα λάθους και η γραμμή 31 επιστρέφει την ροή του προγράμματος εκτός ρουτίνας έτσι ο κώδικας που βρίσκεται παρακάτω και εισάγει ένα νέο στοιχείο ελέγχου δεν εκτελείται.

Στις γραμμές 34 – 37 γίνεται έλεγχος αν το κείμενο του txtElement1 είναι κενό. Αν είναι κενό δεν μπορεί να εισαχθεί ένα νέο στοιχείο στο columnview control. Αν δεν υπάρχει αυτός ο έλεγχος και προσπαθήσουμε να εισάγουμε ένα νέο στοιχείο τότε ένα λάθος θα εμφανιστεί.

Οι γραμμές 39-41 ελέγχουν ποιο εικονίδιο υπάρχει στην τρέχουσα γραμμή του control columnview. Όταν το τρέχον εικονίδιο είναι το pic1 η γραμμή 40 προσθέτει ένα στοιχείο με κλειδί την τιμή του txtElement1.Text, πατρικό στοιχείο το τρέχον στοιχείο clmView.Current.Key και εικονίδιο το pic2.

Αντίστοιχα στις γραμμές 43-53 ο κώδικας επαναλαμβάνεται για τα εικονίδια pic2, pic3, pic4.

Στις γραμμές 55 και 56 προσθέτουμε το κείμενο των txtElement1.Text και txtElement2.Text αντίστοιχα στις στήλες 1 και 2 για το στοιχείο με το κλειδί txtElement1.Text.

Το control clmView συμπεριφέρεται ως ένας πίνακας για ανάγνωση. Έτσι το clmView[txtElement1.Text] επιστρέφει το στοιχείο με το κλειδί txtElement1.Text ενώ το clmView[txtElement1.Text][1] επιστρέφει την πρώτη στήλη του στοιχείου αυτού. Αντίστοιχα και η έκφραση clmView[txtElement1.Text][2] επιστρέφει την δεύτερη στήλη του στοιχείου αυτού.

Δουλεύοντας με γραμμές προόδου, η κλάση ProgressBar

Η κλάση υλοποιεί ένα στοιχείο ελέγχου το οποίο εμφανίζει μια γραμμή προόδου μιας εργασίας με μια αριθμητική τιμή ποσοστού προόδου.
Η τιμή της αριθμητικής τιμής τίθεται από την ιδιότητα Value. Η τιμή αυτή είναι ένας Float μεταξύ των τιμών 0 και 1. Έτσι για παράδειγμα το 50% αντιστοιχεί στην τιμή 0,5.

Η κλάση μπορεί να δημιουργηθεί.

Σύνταξη:

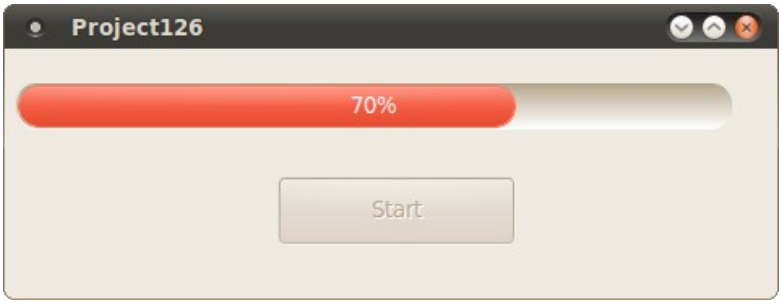
```
DIM hProgressBar AS ProgressBar

hProgressBar = NEW ProgressBar ( Parent AS Container ) [ AS "event name" ]
```

Η κλάση αυτή κληρονομεί την κλάση Control.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Control		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Background Cursor Design Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show	DblClick Drag DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης ProgressBar		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Label Value	Reset	-

Εφαρμογή 1
Το παράδειγμα είναι το Project126.
Η εφαρμογή αυτή προσομοιώνει την λειτουργία μια διεργασίας που χρειάζεται χρόνο να εκπληρωθεί. Στην φόρμα της εφαρμογής έχουμε την γραμμή προόδου pbarProccess ένα χρονομετρητή Timer1 που κάθε 0.10 seconds αυξάνει την τιμή της γραμμής προόδου και ένα κουμπί που ξεκινά την λειτουργία της γραμμής προόδου.



Εικόνα 117: Η εφαρμογή κατά την εκτέλεση

Ο κώδικας της εφαρμογής είναι:

```
1   ' Gambas class file
2
3   Public Sub Form_Open()
4     Me.Center
5   End
6
7   Public Sub btnStart_Click()
8
9     pbarProccess.Value = 0
10    Timer1.Start
11    btnStart.Enabled = False
12
13  End
14
15  Public Sub Timer1_Timer()
16
17    If pbarProccess.Value < 1 Then
18      pbarProccess.Value += 0.10
19    Else
20      Message.Info("Proccess completed!")
21      Timer1.Stop
```

```
22      btnStart.Enabled = True
23    Endif
24
25  End
```

Ανάλυση:
Στην γραμμή 4 τοποθετούμε την φόρμα FMain στο κέντρο της οθόνης κατά το άνοιγμα της φόρμας.
Στην γραμμή 7 ξεκινά η ρουτίνα του κουμπιού btnStart του συμβάντος Click.
Στην γραμμή 9 αρχικοποιούμε την γραμμή προόδου ορίζοντας σαν αρχική τιμή το 0.
Έτσι κάθε φορά που ο χρήστης θα πατά το κουμπί btnStart θα επαναφέρει την κατάσταση της γραμμής προόδου στην αρχική κατάσταση.
Στην γραμμή 10 ξεκινάμε τον χρονομετρητή Timer1 και στην γραμμή 11 απενεργοποιούμε το κουμπί εκκίνησης. Αυτό γίνεται διότι θέλουμε να αποκλείσουμε το ενδεχόμενο ο χρήστης να οδηγήσει το πρόγραμμα για δεύτερη φορά στην ρουτίνα Timer1_Timer().

Στην γραμμή 15 ξεκινά η ρουτίνα του χρονομετρητή. Όταν η ιδιότητα Enabled του χρονομετρητή έχει την τιμή TRUE ή κληθεί η μέθοδος Start εκτελείται η ρουτίνα αυτή κάθε 1000 ms όπως ορίζεται από την ιδιότητα Delay.
Η ρουτίνα αυτή έχει έναν λογικό έλεγχο IF...THEN...ELSE ο οποίος ελέγχει την τιμή pbarProccess.Value.
Όσο η τιμή αυτή είναι μικρότερη του 1 αυξάνουμε την τιμή της γραμμής προόδου.
Όταν η τιμή αυτή γίνει μεγαλύτερη ή ίση με 1 εμφανίζουμε ένα μήνυμα ότι η διεργασία ολοκληρώθηκε, σταματάμε τον χρονομετρητή και ενεργοποιούμε το κουμπί έναρξης.

Δουλεύοντας με γραμμές κύλισης, η κλάση ScrollBar

Η κλάση υλοποιεί ένα στοιχείο ελέγχου το οποίο εμφανίζει μια γραμμή κύλισης ενός δείκτη που αντιστοιχεί σε αριθμητική τιμή μεταξύ ενός εύρους τιμών ορισμένες από τον χρήστη.
Το εύρος των τιμών είναι ακέραιοι για τις ιδιότητες MaxValue, MinValue.
Η τιμή της αριθμητικής τιμής τίθεται από την ιδιότητα Value και είναι ένας ακέραιος.
Υπάρχει και η ιδιότητα Step που αντιστοιχεί στο βήμα μεταβολής της τιμής Value όταν ο χρήστης κάνει click στο control στις περιοχές αύξησης ή μείωσης (βελάκια δεξιά και αριστερά).

Η κλάση μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hScrollBar AS ScrollBar

hScrollBar = NEW ScrollBar ( Parent AS Container ) [ AS "event name" ]
```

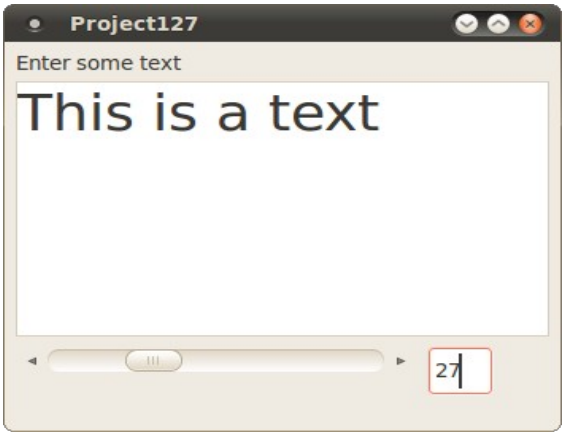
Η κλάση αυτή κληρονομεί την κλάση Control.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Control		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Background Cursor Design Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show	DblClick Drag DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης ScrollBar		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Defaultsize MaxValue MinValue PageStep Step Tracking Value	-	Change

Εφαρμογή 1
Το παράδειγμα είναι το Project127.
Η εφαρμογή έχει στην φόρμα FMain ένα label, ένα textarea control, ένα progressbar control και ένα textbox control.

Στο άνοιγμα της φόρμας αρχικοποιούμε την τιμή του progressbar control και σε κάθε αλλαγή της τιμής του αλλάζουμε το μέγεθος της γραμματοσειράς του textarea control. Η τιμή του progressbar control φαίνεται αριθμητικά δίπλα στο textbox control.

Η γραφική διεπαφή της εφαρμογής είναι η παρακάτω:



Εικόνα 118: Η εφαρμογή κατά την εκτέλεση

Ο κώδικας της εφαρμογής είναι ο εξής:

```
1   ' Gambas class file
2
3   Public Sub Form_Open()
4     Me.Center
5     scrBar.Value = 10
6   End
7
8   Public Sub scrBar_Change()
9
10    txtValue.Text = scrBar.Value
11    txtArea.Font.Size = scrBar.Value
12
13  End
```

Ανάλυση:
Στην γραμμή 3 ξεκινά η ρουτίνα ανοίγματος της φόρμας.
Στην γραμμή 4 τοποθετούμε την φόρμα στο κέντρο της οθόνης.

Στην γραμμή 5 ορίζουμε μια αρχική τιμή στην τιμή της γραμμής κύλισης.
Στην γραμμή 8 ξεκινά η ρουτίνα αλλαγής της τιμής της γραμμής κύλισης.
Κάθε φορά που η τιμή αλλάζει καλείται αυτή η ρουτίνα.
Στην γραμμή 10 θέτουμε το κείμενο του textbox txtValue να είναι η τιμή της γραμμής κύλισης.
Στην γραμμή 11 ορίζουμε το μέγεθος της γραμματοσειράς του textarea control txtArea να είναι η αριθμητική τιμή της γραμμής κύλισης.

Ο χρήστης αν αλλάξει την τιμή της γραμμής κύλισης, με το να σύρει τον βραχίονα αυτής ή με το να πατήσει τα βελάκια στα αριστερά δεξιά, τότε αυτόματα το μέγεθος της γραμματοσειράς της περιοχής κειμένου αλλάζει ανάλογα.

Δουλεύοντας με περιοχές κύλισης, η κλάση ScrollView

Η κλάση υλοποιεί ένα στοιχείο ελέγχου το οποίο εμφανίζει μια περιοχή δοχέα για άλλα αντικείμενα η οποία εμφανίζει γραμμές κύλισης και μεταβάλει την περιοχή που εμφανίζει. Ο χρήστης εμφανίζει το μέρος της περιοχής που θέλει χρησιμοποιώντας τις γραμμές κύλισης.

Η κλάση μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hScrollView AS ScrollView

hScrollView = NEW ScrollView ( Parent AS Container ) [ AS "event name" ]
```

Η κλάση κληρονομεί την κλάση Container.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης ScrollView		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Arrangement Border ClientH ClientHeight ClientW ClientWidth ClientX ClientY Contentsh Contentsheight Contentsw Contentswidth Indent Margin Padding ScrollBar ScrollH ScrollHeight ScrollW ScrollWidth ScrollX ScrollY Spacing	EnsureVisible Scroll	Scroll

Εφαρμογή 1
Το παράδειγμα είναι το Project128.
Η εφαρμογή στην φόρμα FMain έχει ένα scrollview control το οποίο περιέχει ένα PictureBox control με μια εικόνα φορτωμένη.
Η εικόνα είναι επίτηδες μεγάλη ώστε οι διαστάσεις του PictureBox control να είναι μεγάλες και έτσι ο χρήστης να αναγκαστεί να κυλίσει την περιοχή εμφάνισης για να δει όλη την εικόνα.

Εικόνα 119: Η εφαρμογή κατά την κατάσταση εκτέλεσης

Δουλεύοντας με καρτέλες, η κλάση TabStrip

Η κλάση υλοποιεί ένα στοιχείο ελέγχου το οποίο εμφανίζει μια σειρά από καρτέλες οι οποίες λειτουργούν ως δοχείς για άλλα αντικείμενα. Κάθε καρτέλα μπορεί να γίνει ενεργή προς χρήση με την ιδιότητα Index.

Η κλάση λειτουργεί ως ένας πίνακας προς ανάγνωση.
Η κλάση μπορεί να δημιουργηθεί.

Σύνταξη:

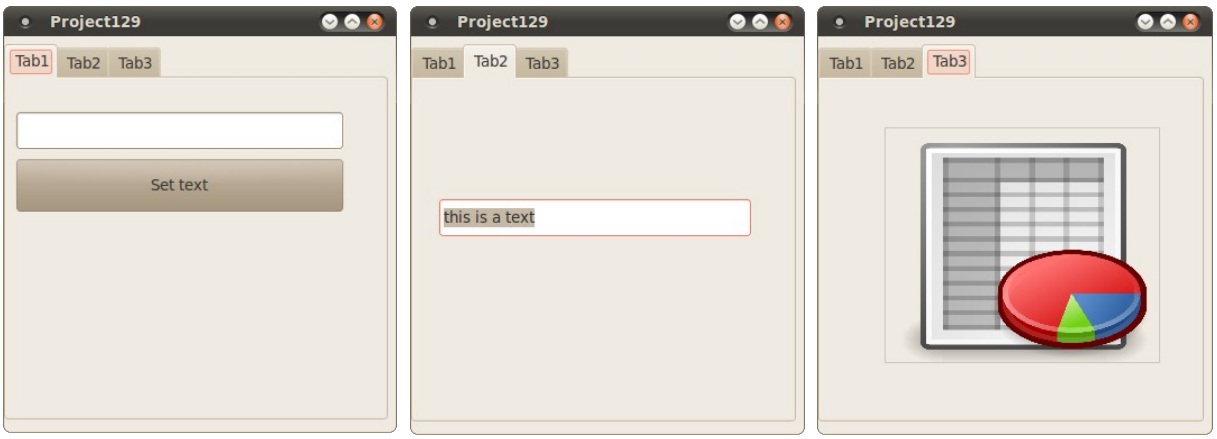
```
DIM hTabStrip AS TabStrip

hTabStrip = NEW TabStrip ( Parent AS Container ) [ AS "event name" ]
```

Η κλάση αυτή κληρονομεί την κλάση Container.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης TabStrip		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Arrangement AutoResize Caption ClientH ClientHeight ClientW ClientWidth ClientY ClientX Closable Count Current Enabled Indent Index Margin Orientation Padding Picture Spacing Text Textfont	-	Click Close

Εφαρμογή 1
Το παράδειγμα είναι το Project129.
Η εφαρμογή περιέχει τρεις καρτέλες. Για να προσθέσουμε καρτέλες ορίζουμε μια αριθμητική τιμή στην ιδιότητα Count. Κάθε καρτέλα έχει ένα ενδεικτικό κείμενο για ονομασία όπως Tab1, Tab2, Tab3. Για να θέσουμε το κείμενο της κάθε καρτέλας επιλέγουμε αυτήν και ορίζουμε την τιμή της ιδιότητας Text.
Η πρώτη καρτέλα έχει ένα textbox control και ένα κουμπί.
Η δεύτερη καρτέλα έχει ένα textbox control.
Η τρίτη καρτέλα έχει ένα PictureBox control με μια εικόνα φορτωμένη.
Σκοπός εδώ είναι να δείξουμε ότι κάθε καρτέλα λειτουργεί σαν δοχέας για άλλα αντικείμενα.

Στην πρώτη καρτέλα μπορούμε να γράψουμε ένα κείμενο στο textbox και πατώντας το κουμπί να αντιγράψουμε το κείμενο αυτό στο textbox της δεύτερης καρτέλας. Την ίδια στιγμή αλλάζουμε καρτέλα από την πρώτη στην δεύτερη θέτοντας την ιδιότητα Index.



Εικόνα 120: Η εφαρμογή κατά την κατάσταση εκτέλεσης

Ο κώδικας της εφαρμογής είναι ο εξής:

```
1   ' Gambas class file
2
3   Public Sub Form_Open()
4     Me.Center
5   End
6
7   Public Sub btnSetText_Click()
8
9     txtBox2.Text = txtBox1.Text
10    tabStrip.Index = 1
11
12  End
```

Ανάλυση:
Στην γραμμή 3 ξεκινά η ρουτίνα ανοίγματος της φόρμας.
Στην γραμμή 4 τοποθετούμε στο κέντρο της οθόνης την φόρμα.
Στην γραμμή 7 ξεκινά η ρουτίνα του κουμπιού btnSetText για το συμβάν Click.
Κάθε φορά που κάνουμε κλικ στο κουμπί καλείται αυτή η ρουτίνα.
Στην γραμμή 9 αντιγράφουμε το κείμενο του txtBox1 στο txtBox2.
Στην γραμμή 10 ορίζουμε σαν καρτέλα προς χρήση το Tab2.
Η αρίθμηση των καρτελών ξεκινά από το 0 έτσι το Index=1 θέτει προς χρήση την δεύτερη καρτέλα.

Δουλεύοντας με γραμμές εργαλείων, η κλάση ToolButton

Η κλάση υλοποιεί ένα toolbar button control.
Ένα στοιχείο ελέγχου toolbar button εμφανίζει το περίγραμμα του μόνο όταν ο δείκτης του ποντικιού εισέρχεται στο control.

Η κλάση μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hToolButton AS ToolButton

hToolButton = NEW ToolButton ( Parent AS Container ) [ AS "event name" ]
```

Η κλάση αυτή κληρονομεί την κλάση Control.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Control		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action Background Cursor Design Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Mouse Name Next Parent PopupMenu Previous Proxy ScreenX ScreenY Tag ToolTip Top Tracking Visible W Width Window X Y	Delete Drag Grab Hide Lower Move MoveScaled Raise Refresh Reparent Resize ResizeScaled Screenshot SetFocus Show	DblClick Drag DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης ScrollBar		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
AutoResize Border Caption Picture Radio Text Toggle Value	-	Click

Εφαρμογή 1
Το παράδειγμα είναι το Project130.
Η εφαρμογή έχει ένα αντικείμενο Hbox ως δοχέα των επτά ToolButtons controls.
Το Hbox είναι ένας δοχέας που τοποθετεί αυτόματα οριζόντια όσα αντικείμενα βρίσκονται εντός του.
Κάθε Toolbutton έχει κοινή την τιμή της ιδιότητας Action ώστε να καλεί την ίδια μέθοδο όταν ο χρήστης πατήσει ένα από αυτά. Όταν πατηθεί ένα toolbutton ένα μήνυμα εμφανίζεται.

Εικόνα 121: Η εφαρμογή σε κατάσταση εκτέλεσης

Ο κώδικας της εφαρμογής είναι ο εξής:

```
1   ' Gambas class file
2
3   Public Sub Form_Open()
4
5   Me.Center
6
7   End
8
9   Public Sub Action_Activate(key As String) As Boolean
10   Select Case key
11   Case "Inform"
12   ' call inform sub here
13   Inform()
14   End Select
15  End
16
17  Public Sub Inform()
```

```
18
19    Message.Info("A toolbutton was clicked")
20
21  End
```

Ανάλυση:
Στην γραμμή 3 ξεκινά η ρουτίνα ανοίγματος της φόρμας.
Στην γραμμή 5 τοποθετούμε στο κέντρο της οθόνης την φόρμα.
Στην γραμμή 9 ξεκινά η ρουτίνα ενεργοποίησης του αντικειμένου Action.
Στην ρουτίνα αυτή διοχετεύεται η τιμή της ιδιότητας Action των Toolbuttons controls.
Στις γραμμές 10-14 υπάρχει μια δομή SELECT CASE η οποία ελέγχει την τιμή της παραμέτρου key.
Όταν η παράμετρος έχει την τιμή “Inform” καλεί την ρουτίνα Inform().
Στην γραμμή 17 ξεκινά η ρουτίνα Inform() η οποία στην γραμμή 19 εμφανίζει ένα μήνυμα στον χρήστη ότι έχει πατήσει ένα κουμπί εργαλειοθήκης.

Δουλεύοντας με εικονίδια στην περιοχή ειδοποιήσεων, η κλάση TrayIcon

Η κλάση υλοποιεί ένα εικονίδιο στην περιοχή ειδοποιήσεων (Tray).
Ένα αντικείμενο TrayIcon τυπικά ανταποκρίνεται στο συμβάν διπλό κλικ DblClick για να επαναφέρει στο προσκήνιο μια ελαχιστοποιημένη φόρμα ή με δεξί κλικ εμφανίζει ένα μενού με επιλογές σχετικά με την εφαρμογή. Επίσης μπορείτε να χρησιμοποιήσετε την ιδιότητα ToolTip του TrayIcon για να εμφανίσετε την κατάσταση της εφαρμογής.

Η κλάση μπορεί να δημιουργηθεί.

Σύνταξη:

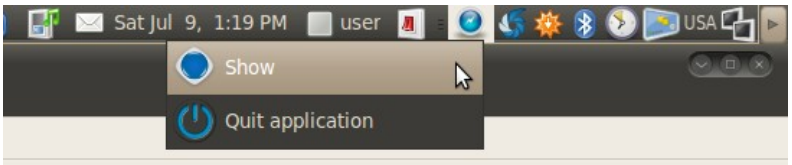
```
DIM hTrayIcon AS TrayIcon

hTrayIcon = NEW TrayIcon ( ) [ AS "event name" ]
```

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης TrayIcon		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
H Height Icon Picture Popupmenu ScreenX ScreenY Tag Text Tooltip Visible W Width	Delete Hide Show	DblClick Enter GotFocus Leave LostFocus Menu MouseDown MouseMove MouseUp MouseWheel

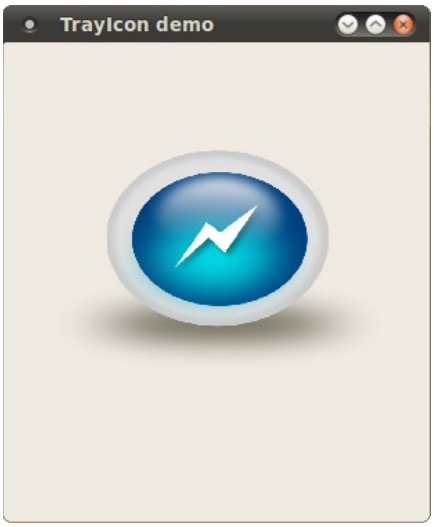
Εφαρμογή 1
Το παράδειγμα είναι το Project131.
Η εφαρμογή δεν περιέχει αντικείμενα εκτός από ένα PictureBox το οποίο εμφανίζει μια εικόνα απλά για εφέ και έναν Timer που ελέγχει συνεχώς αν έχουμε ελαχιστοποιήσει την φόρμα.
Αν ελαχιστοποιήσουμε την φόρμα τότε την κρύβουμε και αφήνουμε μόνο το TrayIcon.

Ο χρήστης αν κάνει διπλό κλικ στο εικόνίδιο στο Tray η φόρμα εμφανίζεται και εξαφανίζεται αντίστοιχα αν είναι αόρατη ή ορατή.

Αν κάνει δεξί κλικ στο Tray εμφανίζεται ένα μενού με δύο επιλογές. Η πρώτη εμφανίζει την φόρμα και η δεύτερη τερματίζει την εφαρμογή.



Εικόνα 122: Το εικονίδιο με τις επιλογές του στο Tray



Εικόνα 123: Η φόρμα FMain

Ο κώδικας της εφαρμογής είναι ο εξής:

```
1   ' Gambas class file
2
3   Private hTrayIcon As TrayIcon
4   Private Mnu As Menu
5   Private Opt1 As Menu
6   Private Opt2 As Menu
7
8   Public Sub _new()
9    'create trayicon
10   hTrayIcon = New TrayIcon As "hTrayIcon"
11   hTrayIcon.Picture = Picture["app.png"]
12   hTrayIcon.Tooltip = "The demo trayicon application"
13   hTrayIcon.Show
14
15   InitMenu()
16
17  End
18
19  Public Sub Form_Open()
20    Me.Center
21  End
22
23  Public Sub Form_Close()
24    ExitApp()
25  End
```

```
26
27
28  Public Sub InitMenu()
29   'setup popup menu
30   Mnu = New Menu(Me)
31   Mnu.Name = "Mnu"
32
33   'setup option1
34   Opt1 = New Menu(Mnu) As "MenuTray"
35   Opt1.Name = "Opt1"
36   Opt1.Caption = "Show"
37   Opt1.Visible = True
38   Opt1.Picture = Picture["show.png"]
39
40   'setup option2
41   Opt2 = New Menu(Mnu) As "MenuTray"
42   Opt2.Name = "Opt2"
43   Opt2.Caption = "Quit application"
44   Opt2.Picture = Picture["exit.png"]
45   Opt2.Visible = True
46
47  End
48
49  Public Sub MenuTray_Click()
50   Select Case Last.name
51     Case "Opt1"
52       Me.Visible = True
53       Me.Minimized = False
54     Case "Opt2"
55       ExitApp()
56   End Select
57
58  End
59
60  Public Sub hTrayIcon_Click()
61   Mnu.Popup
62  End
63
64  Public Sub ExitApp()
65    Quit
66  End
67
68  Public Sub Timer1_Timer()
69   If Me.Minimized = True Then
70     Me.Hide
```

```
71   Endif
72 End
```

Ανάλυση:
Στην γραμμή 3 δηλώνουμε ένα καθολικό ιδιωτικό αντικείμενο τύπου TrayIcon.
Στις γραμμές 4-6 δηλώνουμε τρία καθολικά ιδιωτικά αντικείμενα τύπου Menu.
Στην γραμμή 8 αρχίζει η ειδική μέθοδος _new η οποία αρχικοποιεί την εφαρμογή.
Σε αυτήν την μέθοδο δημιουργείται το αντικείμενο TrayIcon και το μενού του.

Στην γραμμή 10 δημιουργούμε ένα αντικείμενο TrayIcon με όνομα hTrayIcon το οποίο έχει έναν παρατηρητή συμβάντων με το όνομα επίσης hTrayIcon.

Στις γραμμές 11-12 θέτουμε διάφορες ιδιότητες του αντικειμένου hTrayIcon όπως εικόνα και κείμενο tooltip.
Στην γραμμή 13 εμφανίζουμε το αντικείμενο hTrayIcon στο Tray.

Στην γραμμή 15 καλούμε την μέθοδο InitMenu η οποία είναι μια μέθοδος που έχει δημιουργηθεί από εμάς. Σε αυτήν την μέθοδο θα δημιουργήσουμε το μενού του αντικειμένου hTrayIcon.

Η ροή του προγράμματος συνεχίζει στην σειρά 28 όπου ξεκινά η ρουτίνα InitMenu().
Στην γραμμή 30 δημιουργούμε ένα νέο αντικείμενο τύπου Menu το οποίο έχει πατρική κλάση την φόρμα FMain.
Στην γραμμή 31 του ορίζουμε ένα όνομα.

Στην γραμμή 34 δημιουργούμε ένα νέο αντικείμενο μενού που αποτελεί μια επιλογή στο κεντρικό μενού Mnu και για αυτό έχει πατρική κλάση την κλάση Mnu.
Στις γραμμές 35, 36, 37, 38 ορίζουμε το όνομα του μενού αυτού, το κείμενο προς εμφάνιση, το καθιστούμε ορατό, και ορίζουμε και ένα εικονίδιο.

Τον ίδιο κώδικα γράφουμε και για την δεύτερη επιλογή του μενού στις γραμμές 40-45.

Η ροή του προγράμματος επιστρέφει στην γραμμή 17 όπου τελειώνει η ρουτίνα _new.

Στην γραμμή 19 ξεκινά η ρουτίνα ανοίγματος της φόρμας.
Στην γραμμή 20 τοποθετούμε στο κέντρο της οθόνης την φόρμα FMain.
Στην γραμμή 23 ξεκινά η ρουτίνα κλεισίματος της φόρμας.
Στην γραμμή 24 καλούμε την ρουτίνα ExitApp(), η οποία τερματίζει την εφαρμογή.

Στην γραμμή 49 ξεκινά η ρουτίνα MenuTray_Click() η οποία καλείται όταν ο χρήστης κάνει κλικ στο εικονίδιο του Tray. Εδώ η ρουτίνα αυτή καλείται όταν ο χρήστης κάνει κλικ σε κάποιο μενού του TrayIcon.

Στις γραμμές 50-56 υπάρχει μια δομή SELECT CASE.

Η γραμμή 50 ελέγχει το όνομα του τελευταίου αντικειμένου. Στην προκειμένη περίπτωση το τελευταίο αντικείμενο είναι ένα αντικείμενο Menu που έχει επιλέξει ο χρήστης από το αναδυόμενο μενού. Αυτό το αντικείμενο μπορεί να είναι είτε το Opt1 είτε το Opt2.

Στην περίπτωση που είναι το Opt1 στην γραμμή 51 ο χρήστης έχει επιλέξει το μενού Show και έτσι στις γραμμές 52, 53 εμφανίζουμε την φόρμα και θέτουμε την ιδιότητα Minimized στην τιμή False. Το τελευταίο γίνεται διότι ο Timer1 παρακολουθεί συνεχώς την κατάσταση της φόρμας αν είναι minimized και όταν είναι την κρύβει. Για αυτόν τον λόγο αλλάζουμε και την τιμή της ιδιότητας Minimized.

Στην γραμμή 54 ο χρήστης έχει επιλέξει το δεύτερο μενού το οποίο είναι το Quit.
Έτσι στην γραμμή 55 καλούμε την ρουτίνα ExitApp() η οποία τερματίζει την εφαρμογή.
Η ρουτίνα ExitApp() είναι μια custom ρουτίνα που χρησιμοποιεί την εντολή Quit όπως φαίνεται στις γραμμές 71-73.

Στις γραμμές 60-62 ξεκινά η ρουτίνα hTrayIcon_Menu() η οποία καλείται όταν ο χρήστης κάνει δεξί κλικ στο tray icon.
Σε αυτήν την περίπτωση εμφανίζουμε με ανάδυση το μενού Mnu όπως δείχνει η γραμμή 56.

Την ελαχιστοποίηση της φόρμας την παρακολουθεί ο Timer1 και η ροή του προγράμματος σε αυτήν την περίπτωση πηγαίνει στην γραμμή στην γραμμή 68 όπου βρίσκεται η μέθοδος Timer1_Timer(). Ο Timer1 κάθε 100 ms τρέχει αυτήν την ρουτίνα.
Στην γραμμή 69 ελέγχει αν φόρμα είναι ελαχιστοποιημένη. Αν είναι τότε στην γραμμή 70 την κρύβει.

Ο παραπάνω κώδικας είδαμε ότι αλλάζει ροή από την γραμμή 63 στην γραμμή 75.

Στην περίπτωση που η FMain δεν είναι ορατή η ροή του προγράμματος συνεχίζεται στην γραμμή 64. Εδώ στις γραμμές 65 και 66 υπάρχουν δύο εκφράσεις. Η πρώτη επαναφέρει την φόρμα από την ελαχιστοποίηση και η δεύτερη εμφανίζει αυτήν.

Δουλεύοντας με την κλάση Application

Η κλάση υλοποιεί ένα αντικείμενο Application το οποίο δίνει ένα πλήθος πληροφοριών για την εφαρμογή.
Η κλάση υλοποιεί ξανά την κλάση Application του εξαρτήματος gb.
Η κλάση δεν μπορεί να δημιουργηθεί, είναι στατική.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Application
Ιδιότητες (Properties)
Args Daemon Dir Env Handle Id Name Path Return Startup Title Version ActiveControl ActiveWindow Busy Embedder Font MainWindow Theme ToolTip

Εφαρμογή 1
Το παράδειγμα είναι το Project132.
Η εφαρμογή αποτελείται από το άρθρωμα Main.

Ο κώδικας της εφαρμογής είναι ο εξής:

```
1   ' Gambas module file
2
3   Public Sub Main()
4
5     Dim sArg As String
6
7     Print "~~~ Arguments ~~~"
8     For Each sArg In Application.Args
9       Print sArg;;
10    Next
11
12    Print "\n\n~~~ Env ~~~"
13    For Each sArg In Application.Env
14      Print sArg & ": " & Application.Env[sArg]
15    Next
16
17    Print "\n\n~~~ Dir ~~~"
18    Print "Dir: " & Application.Dir
19
20    Print "\n\n~~~ Handle ~~~"
21    Print "Handle: " & Application.Handle
22
23    Print "\n\n~~~ Name ~~~"
24    Print "Name: " & Application.Name
25
26    Print "\n\n~~~ Path ~~~"
27    Print "Path: " & Application.Path
28
29    Print "\n\n~~~ Title ~~~"
30    Print "Title: " & Application.Title
31
32    Print "\n\n~~~ Version ~~~"
33    Print "Version: " & Application.Version
34
35    Print "\n\n~~~ Busy ~~~"
36    Print "Enter busy state"
37    Inc Application.Busy
38    Wait 5
39    Print "Enter normal state"
```

```
40    Dec Application.Busy
41
42    Print "\n\n~~~ Font ~~~"
43    Print "Font: " & Application.Font.Name
44
45  End
```

Έξοδος:

```
~~~ Arguments ~~~
Project132

~~~ Env ~~~
GDM_KEYBOARD_LAYOUT: us
GNOME_KEYRING_PID: 3578
LANGUAGE: en
USER: user
HOME: /home/user
XDG_SESSION_COOKIE: e80bfa1ebdb3c198efe74ad54bf08a1d-
1310363999.150227-1063889980
DESKTOP_SESSION: gnome
GTK_MODULES: canberra-gtk-module
GNOME_KEYRING_CONTROL: /tmp/keyring-tWZfld
MANDATORY_PATH: /usr/share/gconf/gnome.mandatory.path
LOGNAME: user
DEFAULTS_PATH: /usr/share/gconf/gnome.default.path
USERNAME: user
GDM_LANG: en_US.utf8
PATH: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
DISPLAY: :0.0
LANG: en_US.utf8
XAUTHORITY: /var/run/gdm/auth-for-user-2PTavc/database
SHELL: /bin/bash
GDMSESSION: gnome
SPEECHD_PORT: 7560
PWD: /home/user
XDG_CONFIG_DIRS: /etc/xdg/xdg-gnome:/etc/xdg
XDG_DATA_DIRS: /usr/share/gnome:/usr/local/share/:/usr/share/
SSH_AUTH_SOCK: /tmp/keyring-tWZfld/ssh
SSH_AGENT_PID: 3634
GPG_AGENT_INFO: /tmp/gpg-Meq5wm/S.gpg-agent:3635:1
DBUS_SESSION_BUS_ADDRESS: unix:abstract=/tmp/dbus-
9ule8DA6ZP,guid=98bbe8b05983fa88cb70c7184e1a915f
GNOME_DESKTOP_SESSION_ID: this-is-deprecated
SESSION_MANAGER:
```

```
local/ubuntu-desktop:@/tmp/.ICE-unix/3596,unix/ubuntu-desktop:/tmp/.ICE-
unix/3596
ORBIT_SOCKETDIR: /tmp/orbit-user
APPMENU_DISPLAY_BOTH: 1


~~~ Dir ~~~
Dir: /home/user


~~~ Handle ~~~
Handle: 6184


~~~ Name ~~~
Name: Project132


~~~ Path ~~~
Path:
/home/user/Store/LINUX/DOCUMENTS/DEVELOPMENT/Gambas/Προγραμματ
ισμός σε Gambas/Το βιβλίο της Gambas3/1 - Το βιβλίο της
Gambas3/Projects/Project132


~~~ Return ~~~
Return: 0


~~~ Title ~~~
Title: Project132


~~~ Version ~~~
Version: 0.0.1


~~~ Busy ~~~
Enter busy state
Enter normal state


~~~ Font ~~~
Font: Sans
```

Ανάλυση:
Στην γραμμή 5 δηλώνουμε μια αλφαριθμητική μεταβλητή με το όνομα sArg.
Στις γραμμές 7-10 και 12-15 απαριθμούμε τα αντικείμενα Application.Args και Application.Env τα οποία συμπεριφέρονται ως πίνακες.
Κάθε τιμή του των στοιχείων των αντικειμένων αυτών διοχετεύεται στην μεταβλητή sArg.

Στην γραμμή 9 τυπώνουμε στην προκαθορισμένη έξοδο την τιμή της sArg.

Στην γραμμή 12 οι χαρακτήρες \n\n τυπώνουν δύο νέες κενές γραμμές και στην συνέχεια το string "~~~ Env ~~~". Αυτό το μοτίβο επαναλαμβάνεται στις γραμμές 17, 20, 23, 26, 29, 32, 35, 38, 41 και 48.

Στην γραμμή 14 η sArg περιέχει το όνομα των μεταβλητών του συστήματος.
Η έκφραση `Print sArg` τυπώνει το όνομα της μεταβλητής ενώ η έκφραση `Application.Env[sArg]` επιστρέφει την τιμή της μεταβλητής αυτής.
Η έκφραση `Print sArg & ": " & Application.Env[sArg]` τυπώνει στην προκαθορισμένη έξοδο το όνομα της μεταβλητής και την τιμή της.
Για παράδειγμα GDM_KEYBOARD_LAYOUT: us.

Στην γραμμή 18 τυπώνουμε τον κατάλογο στον οποίο βρίσκεται η εφαρμογή.
Στην γραμμή 21 τυπώνουμε τον αριθμό της διεργασίας (pid).
Στην γραμμή 24 τυπώνουμε το όνομα της εφαρμογής.
Στην γραμμή 27 τυπώνουμε την διαδρομή του καταλόγου που βρίσκεται η εφαρμογή (PATH).
Στην γραμμή 30 τυπώνουμε τον τίτλο της εφαρμογής.
Στην γραμμή 33 τυπώνουμε την έκδοση της εφαρμογής.

Στην γραμμή 40 αυξάνουμε κατά μία μονάδα την ιδιότητα Application.Busy.
Όταν η τιμή της ιδιότητας Busy είναι μεγαλύτερη από το 0 τότε ο δείκτης του ποντικιού κλειδώνει στην κατάσταση απασχόλησης (busy) για να δείξει ότι η εφαρμογή είναι απασχολημένη και δεν δέχεται συμβάντα από τον χρήστη.
Αυξάνετε την ιδιότητα αυτή όταν η εφαρμογή ξεκινά μια μεγάλη εργασία και μειώνετε την ιδιότητα αυτή όταν τελειώσει η εργασία αυτή.

Στην γραμμή 37 επειδή αυξάνεται η ιδιότητα αυτή η εφαρμογή εισέρχεται στην κατάσταση απασχόλησης. Στην γραμμή 38 κάνουμε προσομοίωση μιας τέτοιας κατάστασης με το να θέτουμε σε αναμονή την εφαρμογή για 5 δευτερόλεπτα.
Στην συνέχεια στην γραμμή 40 μειώνουμε την ιδιότητα Busy και επαναφέρουμε την εφαρμογή στην κατάσταση κανονικής λειτουργίας.

Στην γραμμή 43 τυπώνουμε στην προκαθορισμένη έξοδο το όνομα της γραμματοσειράς που χρησιμοποιεί η εφαρμογή.

Η ιδιότητα Deamon

Πριν περιγράψουμε την ιδιότητα Deamon πρέπει να πούμε λίγα λόγια για την δικράνωση των διεργασιών (fork).

Η δικράνωση fork() είναι μια ιδέα η οποία χρησιμοποιείται στα λειτουργικά συστήματα UNIX. Κάθε πρόγραμμα που τρέχει αποτελεί μια διεργασία process, και ο μόνος τρόπος για να εκτελέσετε ένα νέο πρόγραμμα είναι με το να διχαλώσετε ένα ήδη υπάρχον.
Η διχάλωση ή δικράνωση είναι εκείνη η ενέργεια όπου μια διεργασία αποτελεί ένα αντίγραφο μιας άλλης. Η πρώτη διεργασία αποτελεί την πατρική διεργασία (parent) και η δεύτερη διεργσία το παιδί (child).

Η διεργασία παιδί ξεκινά την ύπαρξη της ως ένα αντίγραφο της πατρικής διεργασίας και μπορεί να αντικατασταθεί.

Στο UNIX, το πρώτο πρόγραμμα που εκτελείται μετά την εκκίνηση είναι η διεργασία init. Αυτή η διεργασία δικρανώνεται και φορτώνει ένα άλλο πρόγραμμα όπως το getty πάνω στην διεργασία παιδί. Η διεργασία παιδί που αποτελεί ένα αντίγραφο και έχει αντικατασταθεί από άλλο πρόγραμμα (το getty) τώρα χρησιμοποιείται για να επιτρέψει στον χρήστη να εισάγει το username και password του και να κάνει login στο σύστημα.

Όταν το fork της διεργασία έχει πετύχει η διεργασία επιστρέφει το id της διεργασίας παιδί στην διεργασία πατέρα και 0 στην διεργασία παιδί.

Έτσι μπορείτε να χρησιμοποιήσετε την νέα διεργασία για να εκτελέσετε μια κλήση του συστήματος (system call) όπως να τρέξετε ένα άλλο πρόγραμμα.

Η αντιγραφή και αντικατάσταση μιας διεργασίας σε ένα σύστημα LINUX ονομάζεται δικράνωση (fork).

Η σύνταξη της ιδιότητας Deamon είναι η παρακάτω:

```
STATIC PROPERTY Daemon AS Boolean
```

Αν θέσετε στην ιδιότητα αυτή την τιμή TRUE κάνετε την τρέχουσα διεργασία ένα *daemon*.

Όταν μια διεργασία γίνεται daemon τότε:

- Η εφαρμογή δικρανώνεται, και τερματίζει την διεργασία πατέρα έτσι ώστε η εφαρμογή να μην αναμένει τερματισμό.
- Ο τρέχων κατάλογος τίθεται στην τιμή `"/"`.
- Η προκαθορισμένη είσοδος, έξοδος και έξοδος λάθους κλείνονται.

Δεν μπορείτε να θέσετε την ιδιότητα αυτή στην τιμή `FALSE` από την στιγμή που η διεργασία γίνει daemon.
Ο προσωρινός κατάλογος της εφαρμογής μετονομάζεται και ο προσδιορισμός (pid) της διεργασίας αλλάζει μετά την δικράνωση.

Δουλεύοντας με τον εκτυπωτή, η κλάση Printer

Η κλάση αυτή επιτρέπει να τυπωθούν έγγραφα στον εκτυπωτή ή να σωθούν αυτά σε αρχεία μέσω του εκτυπωτή.
Για να τυπώσετε ένα έγγραφο θα πρέπει:

- Να υλοποιήσετε ένα αντικείμενο Printer.
- Να καλέσετε την μέθοδο Configure. Ένα πλαίσιο διαλόγου για τον εκτυπωτή εμφανίζεται.
- Καλέστε την μέθοδο Print για να τρέξει η διεργασία εκτύπωσης.

Όταν καλείτε την μέθοδο Print, ένα τοπικός βρόχος συμβάντων τρέχει και η μέθοδος επιστρέφει μόνο όταν η εκτύπωση τελειώνει ή ακυρώνεται.
Κατά την διάρκεια του βρόχου τα ακόλουθα συμβάντα εγείρονται:

- Το συμβάν Begin εγείρεται όταν η εκτύπωση αρχίζει.
- Το συμβάν Paginate εγείρεται για την σελιδοποίηση του εγγράφου στο παρασκήνιο
- Το συμβάν Draw εγείρεται για κάθε σελίδα που εκτυπώνεται.
- Το συμβάν End εγείρεται όταν η εκτύπωση τελειώνει.

Το συμβάν που πρέπει αναγκαστικά να εγερθεί είναι το Draw.
Τα άλλα συμβάντα είναι προαιρετικά.

Το συμβάν Begin

Το συμβάν αυτό εγείρεται όταν η εκτύπωση ξεκινά.
Μπορείτε να ορίσετε την ιδιότητα Printer.Count αν γνωρίζεται πόσες σελίδες θα εκτυπωθούν.
Αλλιώς θα πρέπει να υλοποιήσετε το συμβάν Paginate για να σελιδοποιήσετε το έγγραφο και να υπολογίσετε τις σελίδες προς εκτύπωση.
Εξορισμού μόνο μια σελίδα εκτυπώνεται.

Το συμβάν Paginate

Το συμβάν αυτό εγείρεται ώστε να σελιδοποιήσετε το έγγραφο στο παρασκήνιο.
Εάν δεν ορίσετε διαχειριστή συμβάντων το συμβάν αυτό δεν θα εγερθεί και θα πρέπει να ορίσετε την ιδιότητα Count μέσα στον διαχειριστή συμβάντος Begin.
Αλλιώς αν διαχειριστείτε αυτό το συμβάν ο διαχειριστής συμβάντος θα καλείται ξανά και ξανά μέχρι να ορίσετε την ιδιότητα Count.

Το συμβάν Draw

Αυτό το συμβάν εγείρεται μια φορά για κάθε σελίδα που τυπώνεται.
Η τρέχουσα σελίδα επιστρέφεται από την ιδιότητα Printer.Page.
Η σχεδίαση γίνεται χρησιμοποιώντας την κλάση Paint.

The End event

Το συμβάν αυτό εγείρεται όταν η διεργασία εκτύπωσης τελειώνει.

Πώς να σχεδιάσετε μια σελίδα.

Διαστάσεις της σελίδας

Η σχεδίαση γίνεται μέσα σε ένα παραλληλόγραμμο που αντιπροσωπεύει την σελίδα. Οι συντεταγμένες της άνω αριστερής γωνίας του είναι (0,0), και το μέγεθος του είναι σε μήκος και ύψος οι τιμές των ιδιοτήτων Paint.Width, Paint.Height.

Περιθώρια

Το προηγούμενο παραλληλόγραμμο αντιπροσωπεύει όλη την σελίδα. Αλλά οι εκτυπωτές συνήθως δεν τυπώνουν σε όλη την σελίδα διότι υπάρχουν περιθώρια όπου εκεί δεν τυπώνεται κάτι.
Στο σχεδιασμό της εκτύπωσης σας θα πρέπει να λαμβάνετε υπόψιν και αυτά τα περιθώρια!

Απόλυτες συντεταγμένες

Αντί να χρησιμοποιείται τις συντεταγμένες σχεδίασης, συνήθως χρησιμοποιούμε απόλυτες συντεταγμένες σε χιλιοστά millimeters.
Για να κάνετε αυτό θα πρέπει να χρησιμοποιήσετε τις ιδιότητες Printer.PaperWidth και Printer.PaperHeight οι οποίες επιστρέφουν το μέγεθος του χαρτιού σε millimeters.
Στην συνέχεια θα πρέπει να χρησιμοποιήσετε την μέθοδο Paint.Scale για να θέσετε υπό κλίμακα το σύστημα των συντεταγμένων ώστε κάθε συντεταγμένη που ορίζετε να είναι σε millimeters. Οι εντολές αυτές τοποθετούνται στην ρουτίνα του συμβάντος Printer_Draw() όπου σχεδιάζεται η σελίδα.

Μέγεθος γραμμάτων

Προσοχή: Το μέγεθος των γραμμάτων, καθώς επιστρέφεται από την ιδιότητα Paint.Font.Size, είναι μια απόλυτη τιμή μεγέθους.
Η μονάδα μέτρησης είναι η τυπογραφική μονάδα.
Μια τυπογραφική μονάδα είναι το 1/72 μιας ίντσας που είναι περίπου 0,353 mm.
Συνεπώς εάν 10 μονάδες γραμματοσειράς είναι καλή μέγεθος για να σχεδιάζεται ένα κείμενο στην οθόνη μπορεί να είναι πολύ μεγάλο για τον εκτυπωτή καθώς η ανάλυση εκτύπωσης είναι πολύ μεγαλύτερη από αυτήν της οθόνης. Έτσι συνήθως θα πρέπει να εκτυπώνεται σε μεγέθη μικρότερα από αυτά που εμφανίζονται στην οθόνη.

Η κλάση μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hPrinter AS Printer

hPrinter = NEW Printer ( ) [ AS "event name" ]
```

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Printer		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
CollateCopies Count Duplex FirstPage FullPage GrayScale LastPage Name NumCopies Orientation OutputFile Page Paper PaperHeight PaperWidth Resolution ReverseOrder	Cancel Configure Print	Begin Draw End Paginate
Σταθερές (Constants)		
A3 A4 A5 B5 Custom Executive Horizontal Landscape Legal Letter Portrait Simplex Vertical		

Εφαρμογή 1
Το παράδειγμα είναι το Project133.
Στο παράδειγμα αυτό έχουμε την φόρμα FMain, ένα κουμπί btnPrint, ένα αντικείμενο ptrPrinter, ένα label και ένα TextArea.

Πατώντας το κουμπί με την ετικέτα “Print” ξεκινάμε την διαδικασία εκτύπωσης.
Στην περιοχή κειμένου TextArea εμφανίζονται μηνύματα καθώς το αντικείμενο εκτυπωτής καλεί μία μία τις μεθόδους prtPrinter_Begin, prtPrinter_Paginate, prtPrinter_Draw και prtPrinter_End.

Αυτό το παράδειγμα δεν σχεδιάζει την περιοχή εκτύπωσης απλά επιδεικνύει την σειρά κλήσης των μεθόδων του αντικειμένου εκτύπωσης κατά την εκτύπωση.

```
1   ' Gambas class file
2
3   Public Sub Form_Open()
4     Me.Center
5   End
6
7   Public Sub btnPrint_Click()
8
9     If prtPrinter.Configure() Then Return
10
11    Me.Enabled = False
12    Inc Application.Busy
13    prtPrinter.Print
14    Dec Application.Busy
15    Me.Enabled = True
16
17  End
```

```
18
19  Public Sub prtPrinter_Begin()
20
21    prtPrinter.Count = 1
22    txtArea.Text &= "Set number of pages to 1\n"
23    txtArea.Text &= "Begin printing\n"
24
25  End
26
27  Public Sub prtPrinter_Paginate()
28
29    txtArea.Text &= "Begin paginate\n"
30
31  End
32
33  Public Sub prtPrinter_Draw()
34
35    txtArea.Text &= "Begin Draw\n"
36
37  End
38
39  Public Sub prtPrinter_End()
40
41    txtArea.Text &= "End printing\n"
42
43  End
```

Ανάλυση:
Στις γραμμές 3-5 τοποθετούμε την φόρμα στο κέντρο της οθόνης όταν η φόρμα ανοίγει.
Στις γραμμές 7-17 υπάρχει η ρουτίνα συμβάντος Click του κουμπιού btnPrint.
Στην γραμμή 9 εμφανίζουμε το πλαίσιο διαλόγου εκτύπωσης και επιστρέφουμε την ροή του προγράμματος στην συνέχεια μόλις αυτό κλείσει.
Αν η έκφραση prtPrinter.Configure() επιστρέψει TRUE ο χρήστης έχει ακυρώσει την εκτύπωση και τότε εκτελείτε η εντολή RETURN και εξερχόμαστε από την ρουτίνα.
Με αυτόν τον τρόπο δεν εκτελούμε τις παρακάτω εντολές που τυπώνουν μια σελίδα.

Αν η έκφραση prtPrinter.Configure() επιστρέψει FALSE ο χρήστης έχει επιλέξει να εκτυπωθεί κάτι και έτσι η ροή του προγράμματος συνεχίζει στην γραμμή 11.

Στην γραμμή 11 απενεργοποιούμε την φόρμα προσωρινά.
Στην γραμμή 12 θέτουμε την εφαρμογή στην κατάσταση απασχόλησης.
Στην γραμμή 13 ξεκινάμε την διαδικασία εκτύπωσης.
Στην γραμμή 14 θέτουμε την εφαρμογή στην κατάσταση κανονικής λειτουργίας.
Στην γραμμή 15 ενεργοποιούμε πάλι την φόρμα.

Αφού ξεκινήσουμε την διαδικασία εκτύπωσης καλούνται με την σειρά οι ρουτίνες prtPrinter_Begin, prtPrinter_Paginate, prtPrinter_Draw και prtPrinter_End.

Ο κώδικας της κάθε μία μεθόδου εκτελείται στην συνέχεια.

Στην γραμμή 19 βρίσκεται η μέθοδος prtPrinter_Begin().
Στην γραμμή 21 ορίζουμε τον αριθμό των σελίδων που θα εκτυπωθούν.
Αν δεν ορίζουμε την ιδιότητα prtPrinter.Count τότε θα καλείται συνέχεια η μέθοδος prtPrinter_Paginate() για να σελιδοποιήσει την εκτύπωση.

Στις γραμμές 22, 23 τυπώνουμε μηνύματα που πληροφορούν τον χρήστη για την κατάσταση εκτύπωσης και αυτά εμφανίζονται στο TextArea.

Στην γραμμή 27 βρίσκεται η ρουτίνα prtPrinter_Paginate(). Η οποία καλείται μετά την prtPrinter_Begin(). Στην γραμμή 29 εμφανίζουμε ένα μήνυμα στο control TextArea.

Στην γραμμή 33 prtPrinter_Draw() υπάρχει η ρουτίνα που εκτυπώνει πάνω στο φύλλο χαρτιού. Εδώ θα πρέπει να τοποθετηθούν οι εκφράσεις της κλάσης Paint οι οποίες θα γράψουν πάνω στην σελίδα.
Εμείς σε αυτό το παράδειγμα απλά εμφανίζουμε ένα μήνυμα στην περιοχή κειμένου TextArea με την γραμμή 35.

Τέλος στην γραμμή 39 υπάρχει η ρουτίνα prtPrinter_End() που καλείται τελευταία.
Στην γραμμή 41 εμφανίζουμε ένα μήνυμα στο control TextArea.

Εφαρμογή 2
Το παράδειγμα είναι το Project134.
Στο παράδειγμα αυτό έχουμε την φόρμα FMain, ένα κουμπί btnPrint, ένα αντικείμενο ptrPrinter, ένα label και ένα TextArea.

Πατώντας το κουμπί με την ετικέτα “Print” ξεκινάμε την διαδικασία εκτύπωσης.
Στην περιοχή κειμένου TextArea εμφανίζονται μηνύματα καθώς το αντικείμενο εκτυπωτής καλεί μία μία τις μεθόδους prtPrinter_Begin, prtPrinter_Paginate, prtPrinter_Draw και prtPrinter_End. Στην ρουτνίνα prtPrinter_Draw() σχεδιάζουμε με κόκκινο περίγραμμα την περιοχή εκτύπωσης και τυπώνουμε και ένα κείμενο.

Αυτό το παράδειγμα σχεδιάζει την περιοχή εκτύπωσης και επιδεικνύει την σειρά κλήσης των μεθόδων του αντικειμένου εκτύπωσης κατά την εκτύπωση όπως επίσης δείχνει τον τρόπο με τον οποίο μπορούμε να γράφουμε κείμενο στην σελίδα του οποίου οι μονάδες να αντιστοιχούν σε σωστές μονάδες της συσκευής εκτύπωσης. Με άλλα λόγια όταν θέλουμε να εκτυπώσουμε μια γραμματοσειρά με μέγεθος 12 αυτό επειδή αντιστοιχεί στην ανάλυση της οθόνης πρέπει να το μετατρέψουμε σε ανάλυση εκτυπωτή.

```
1   ' Gambas class file
2
3   Public Sub Form_Open()
4     Me.Center
5   End
6
7   Public Sub btnPrint_Click()
8
9     If prtPrinter.Configure() Then Return
10
11    Me.Enabled = False
12    Inc Application.Busy
13    prtPrinter.Print
14    Dec Application.Busy
15    Me.Enabled = True
16
17  End
18
19  Public Sub prtPrinter_Begin()
20
21    prtPrinter.Count = 1
22    txtArea.Text &= "Set number of pages to 1\n"
23    txtArea.Text &= "Begin printing\n"
24
25  End
26
27  Public Sub prtPrinter_Paginate()
```

```
28
29    txtArea.Text &= "Begin paginate\n"
30    prtPrinter.FullPage = True
31
32  End
33
34  Public Sub prtPrinter_Draw()
35
36    Dim X As Float
37    Dim Y As Float
38    Dim W As Float
39    Dim H As Float
40    Dim wFactor As Float
41    Dim hFactor As Float
42
43    txtArea.Text &= "Begin Draw\n"
44
45    'Set scale
46    wFactor = Paint.Width / prtPrinter.PaperWidth
47    hFactor = Paint.Height / prtPrinter.PaperHeight
48    Paint.Scale(wFactor, hFactor)
49
50    'Set Color to red
51    Paint.Brush = Paint.Color(Color.RGB(255, 0, 0, 0))
52    'Set line width
53    Paint.LineWidth = 0.45
54
55    'Set X,Y origin to 0,0
56    X = 0
57    Y = 0
58
59    'Set Width, Height
60    W = prtPrinter.PaperWidth - Paint.LineWidth
61    H = prtPrinter.PaperHeight - Paint.LineWidth
62
63    'Draw a rectangle
64    Paint.Rectangle(X, Y, W, H)
65
66    'Write it on paper
67    Paint.Stroke
68
69    'Write text to paper
70    Paint.Font.Name = "Times New Roman"
71    Paint.Font.Size = 12 * (Desktop.Resolution /
prtPrinter.Resolution)
```

```
72    Paint.Text("Hello world!", 20, 20)
73    Paint.Fill
74
75  End
76
77  Public Sub prtPrinter_End()
78
79    txtArea.Text &= "End printing\n"
80
81  End
```

Ανάλυση:
Στις γραμμές 3-5 τοποθετούμε την φόρμα στο κέντρο της οθόνης όταν η φόρμα ανοίγει.
Στις γραμμές 7-17 υπάρχει η ρουτίνα συμβάντος Click του κουμπιού btnPrint.
Στην γραμμή 9 εμφανίζουμε το πλαίσιο διαλόγου εκτύπωσης και επιστρέφουμε την ροή του προγράμματος στην συνέχεια μόλις αυτό κλείσει.
Αν η έκφραση prtPrinter.Configure() επιστρέψει TRUE ο χρήστης έχει ακυρώσει την εκτύπωση και τότε εκτελείτε η εντολή RETURN και εξερχόμαστε από την ρουτίνα.
Με αυτόν τον τρόπο δεν εκτελούμε τις παρακάτω εντολές που τυπώνουν μια σελίδα.

Αν η έκφραση prtPrinter.Configure() επιστρέψει FALSE ο χρήστης έχει επιλέξει να εκτυπωθεί κάτι και έτσι η ροή του προγράμματος συνεχίζει στην γραμμή 11.

Στην γραμμή 11 απενεργοποιούμε την φόρμα προσωρινά.
Στην γραμμή 12 θέτουμε την εφαρμογή στην κατάσταση απασχόλησης.
Στην γραμμή 13 ξεκινάμε την διαδικασία εκτύπωσης.
Στην γραμμή 14 θέτουμε την εφαρμογή στην κατάσταση κανονικής λειτουργίας.
Στην γραμμή 15 ενεργοποιούμε πάλι την φόρμα.

Αφού ξεκινήσουμε την διαδικασία εκτύπωσης καλούνται με την σειρά οι ρουτίνες prtPrinter_Begin, prtPrinter_Paginate, prtPrinter_Draw και prtPrinter_End.

Ο κώδικας της κάθε μία μεθόδου εκτελείται στην συνέχεια.

Στην γραμμή 19 βρίσκεται η μέθοδος prtPrinter_Begin().
Στην γραμμή 21 ορίζουμε τον αριθμό των σελίδων που θα εκτυπωθούν.
Αν δεν ορίζουμε την ιδιότητα prtPrinter.Count τότε θα καλείται συνέχεια η μέθοδος prtPrinter_Paginate() για να σελιδοποιήσει την εκτύπωση.

Στις γραμμές 22, 23 τυπώνουμε μηνύματα που πληροφορούν τον χρήστη για την κατάσταση εκτύπωσης και αυτά εμφανίζονται στο TextArea.

Στην γραμμή 27 βρίσκεται η ρουτίνα prtPrinter_Paginate(). Η οποία καλείται μετά την prtPrinter_Begin(). Στην γραμμή 29 εμφανίζουμε ένα μήνυμα στο control TextArea.

Στην γραμμή 30 θέτουμε την ιδιότητα prtPrinter.FullPage στην τιμή TRUE. Η ιδιότητα αυτή ορίζει την περιοχή εκτύπωσης σε όλη την σελίδα. Αν είχε την τιμή FALSE τότε η περιοχή εκτύπωσης θα ήταν πιο μικρή αφού θα υπήρχαν δεξία, αριστερά, πάνω και κάτω περιθώρια προκαθορισμένα σε μήκος από την QT. Τα περιθώρια αυτά σε αυτήν την περίπτωση δεν μπορεί να τα αλλάξει ο χρήστης. Ο Benoît προτείνει πάντα να σχεδιάζουμε την εκτύπωση μας σε πλήρη σελίδα και να ορίζει ο χρήστης τα περιθώρια. Ενδέχεται σε μελλοντικές εκδόσεις της Gambas η ιδιότητα FullPage να είναι προκαθορισμένη στην τιμή TRUE και ο χρήστης να μην μπορεί να την αλλάξει.

Στην γραμμή 34 ξεκινά η ρουτίνα prtPrinter_Draw() που σχεδιάζει την σελίδα στον εκτυπωτή.
Στις γραμμές 36-41 ορίζουμε μεταβλητές τύπου Float. Οι X,Y είναι οι συντεταγμένες της πάνω αριστεράς γωνίας του ορθογόνιου παραλληλογράμμου που αντιστοιχεί στην περιοχή εκτύπωσης. Οι W,H είναι το πλάτος και το ύψος της περιοχής εκτύπωσης.
Οι wFactor και hFactor είναι ο λόγος Paint.Width / prtPrinter.PaperWidth και Paint.Height / prtPrinter.PaperHeight που θα διοχετευτούν στην συνάρτηση Paint.Scale().
Στις γραμμές 46-48 υπολογίζουμε τα wFactor και hFactor και ορίζουμε την κλίμακα της περιοχής εκτύπωσης σε mm.

Τα wFactor και hFactor υπολογίζονται ως εξής:
Γνωρίζουμε ότι θέλουμε 1mm στην συσκευή του εκτυπωτή η οποία μετριέται από την ιδιότητα prtPrinter.PaperWidth και prtPrinter.PaperHeight. Επομένως ποια τιμή (X) πρέπει να έχουμε στην συσκευή της κλάσης Paint που υπολογίζει το μήκος και ύψος με τις ιδιότητες Paint.Width και Paint.Height ;

Με απλή μέθοδο των τριών έχουμε:

prtPrinter.PaperHeight	1mm
Paint.Height	X
X = 1 * (Paint.Height/prtPrinter.PaperHeight)	

Έτσι έχουμε wFactor = Paint.Width / prtPrinter.PaperWidth και αντίστοιχα
hFactor = Paint.Height / prtPrinter.PaperHeight

Στην γραμμή 51 ορίζουμε το χώμα των γραμμών σε κόκκινο.
Η έκφραση Color.RGB(255, 0, 0, 0) επιστρέφει το κόκκινο χρώμα.
Η συνάρτηση RGB(Red, Green, Blue, Alpha) δέχεται αριθμητικές τιμές από το 0 έως το 255 για τα κανάλια χρωμάτων Red, Green, Blue και Alpha όπως φαίνεται. Η τιμή Alpha 0 είναι πλήρως αδιαφανές χρώμα ενώ η τιμή 255 πλήρως διαφανές.

Αφού υπολογίζουμε την αριθμητική τιμή του κόκκινου χρώματος την διοχετεύουμε στην μέθοδο Paint.Color() γράφωντας Paint.Color(Color.RGB(255, 0, 0, 0)).
Με τον τρόπο αυτό δημιουργούμε ένα αντικείμενο Brush με χρώμα κόκκινο.
Το αντικείμενο αυτό το εκχωρούμε στην στατική ιδιότητα Paint.Brush γράφωντας
Paint.Brush = Paint.Color(Color.RGB(255, 0, 0, 0))

Στην γραμμή 53 ορίζουμε το πλάτος της γραμμής σχεδίασης.

Στην συνέχεια στις γραμμές 56-61 ορίζουμε τις τιμές των X,Y,W,H.
Θέλουμε η περιοχή εκτύπωσης να είναι όλη η σελίδα οπότε το ορθογώνιο παραλληλόγραμμο μας θα ξεκινά από τις συντεταγμένες 0,0.
Θα έχει μήκος όσο το μήκος της σελίδας μειωμένο κατά το πλάτος της γραμμής σχεδίασης.
Ομοίως θα έχει ύψος το ύψος της σελίδας μειωμένο κατά το πλάτος της γραμμής σχεδίασης.

Οι τιμές αυτές υπολογίζονται από τις εκφράσεις
W = prtPrinter.PaperWidth - Paint.LineWidth
H = prtPrinter.PaperHeight - Paint.LineWidth

Τέλος στην γραμμή 64 σχεδιάζουμε το ορθογώνιο παραλληλόγραμμο σε πλάτος και μήκος όλης της σελίδας με χρώμα κόκκινο, δείχνοντας έτσι την περιοχή εκτύπωσης.

Στην γραμμή 67 τυπώνουμε το κόκκινο ορθογώνιο παραλληλόγραμμο στην σελίδα.

Στην συνέχεια στις γραμμές 68-71 ορίζουμε τις τιμές μιας γραμματοσειράς και το κείμενο που εκτυπώνουμε στην σελίδα.
Η γραμμή 68 ορίζει το όνομα της γραμματοσειράς
Η γραμμή 69 μετατρέπει τις μονάδες οθόνης της γραμματοσειράς σε μονάδες εκτύπωσης.
Έτσι για γραμματοσειρά με 12 font size στην οθόνη η έκφραση που μετατρέπει το μέγεθος αυτό στην ανάλυση του εκτυπωτή είναι
Paint.Font.Size = 12 * (Desktop.Resolution / prtPrinter.Resolution)

Εδώ πάλι με απλή μέθοδο των τριών υπολογίζουμε τις μονάδες που πρέπει να γράψουμε στην οθόνη ώστε να εκτυπωθούν στον εκτυπωτή.

Γνωρίζουμε ότι θέλουμε 12 font size στον εκτυπωτή με ανάλυση prtPrinter.Resolution.
Επομένως ποιο (X) είναι το font size που πρέπει να χρησιμοποιήσουμε στην οθόνη με ανάλυση Desktop.Resolution.

prtPrinter.Resolution	12
Desktop.Resolution	X
X = 12 * (Desktop.Resolution / prtPrinter.Resolution)	

Στην γραμμή 70 γράφουμε το κείμενο που θέλουμε στην θέση με συντεταγμένες x,y τις 20,20.
Στην γραμμή 71 τυπώνουμε το κείμενο πάνω στην σελίδα.

Τέλος στις γραμμές 75-79 υπάρχει η ρουτίνα prtPrinter_End() η οποία εμφανίζει ένα μήνυμα κατά το τέλος της εκτύπωσης.

Για να δοκιμάσουμε την εκτύπωση μας μπορούμε να δημιουργήσουμε ένα αρχείο PDF του εγγράφου εκτύπωσης. Καλή τακτική είναι οι προεπισκοπήσεις εκτύπωσης να είναι PDF έγγραφα τα οποία θα στέλνονται στον εκτυπωτή. Για την προβολή της προεπισκόπησης ως

έγγραφο PDF μπορούμε να χρησιμοποιούμε τα μέλη του εξαρτήματος gb.pdf.

Κεφάλαιο 2ο – Εμπλουτισμένα συστατικά διερμηνευτή (Interpreter enhancements)

gb.pcre

Το εξάρτημα αυτό υλοποιεί τις λεγόμενες κανονικές εκφράσεις (Regular expressions) που είναι συμβατές με τις κανονικές εκφράσεις της Perl.
Μια κανονική έκφραση (regular expression) είναι ένας τρόπος εύρεσης κειμένου που ταιριάζει σε κάποια κριτήρια ενός Pattern. Για παράδειγμα να βρεθούν όλοι οι αριθμοί σε ένα κείμενο, να βρεθούν όλα τα email, να βρεθεί κείμενο που ξεκινά με αριθμούς στην συνέχεια έχει αόριστο αριθμό χαρακτήρων μετά κάτω παύλα μετά αριθμούς και στην συνέχεια τελεία.

Η έκφραση αναζήτησης μπορεί να είναι όσο απλή ή όσο περίπλοκη επιθυμεί ο χρήστης.
Οι κανονικές εκφράσεις της Perl υλοποιούνται στην Gambas3 με το εξάρτημα gb.pcre.

Κλάσεις
Περιέχει τις εξής κλάσεις

Regexp

Δουλεύοντας με τις Regular Expressions, η κλάση Regexp

Η κλάση αυτή αντιπροσωπεύει μια κανονική έκφραση με την οποία μπορείτε να ταιριάξετε εκφράσεις πάνω σε διάφορα strings και να σας επιστρέψει το αποτέλεσμα.

Η κλάση μπορεί να δημιουργηθεί.

Σύνταξη:

```
DIM hRegexp AS Regexp

hRegexp = NEW Regexp ( [ Subject AS String, Pattern AS String,
CompileOptions AS Integer, ExecOptions AS Integer ] )
```

Όπου:
Subject, είναι το αλφαριθμητικό κείμενο προς επεξεργασία
Pattern, είναι η κανονική έκφραση η οποία θα ψάξει για ταιριάσματα αυτής μέσα στο Subject
CompileOptions, είναι ένας integer που δηλώνει μια επιλογή μεταγλώττισης.

Οι επιλογές μεταγλώττισης είναι:

Επιλογή	**Τιμή**
Anchored	16
Caseless	1
DollarEndOnly	32
DotAll	4
Extended	8
Extra	64
MultiLine	2
NoAutoCapture	4096
Ungreedy	512
UTF8	2048
NoUTF8Check	8192

Για περισσότερα για κάθε επιλογή δείτε το online εγχειρίδιο στην σελίδα
https://gambaswiki.org/wiki/comp/gb.pcre

ExecOptions, είναι ένας integer που δηλώνει μια επιλογή εκτέλεσης.
Οι επιλογές εκτέλεσης είναι:

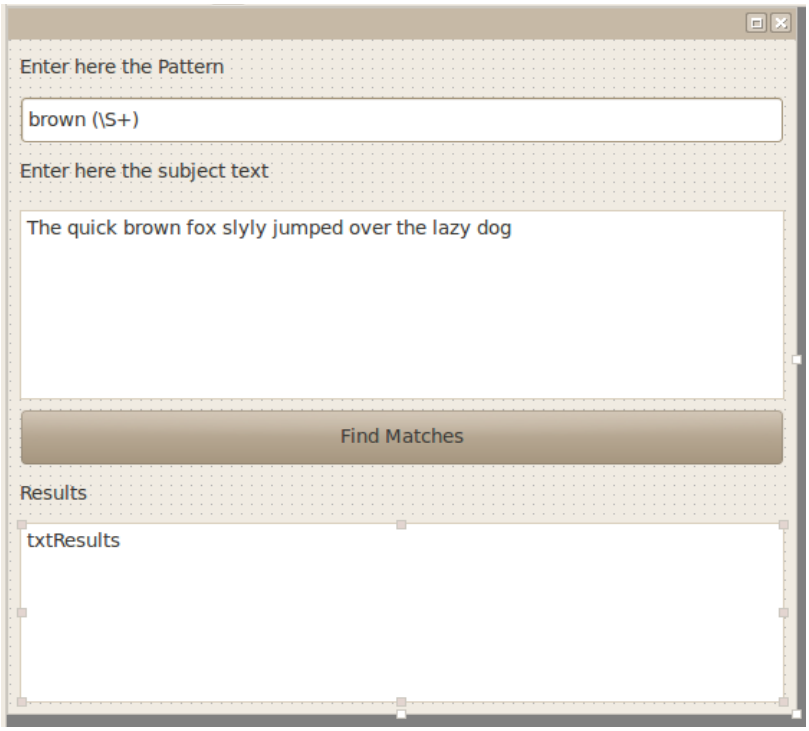
Επιλογή	**Τιμή**
Anchored	16
NotBOL	128
NotEOL	256
NotEmpty	1024
NoUTF8Check	8192

Για περισσότερα για κάθε επιλογή δείτε το online εγχειρίδιο στην σελίδα
https://gambaswiki.org/wiki/comp/gb.pcre

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Regexp		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Offset Pattern SubMatches Subject Text	Compile Exec	-
Σταθερές (Constants)		
Anchored BadMagic BadOption BadUTF8 BadUTF8Offset Callout Caseless DollarEndOnly DotAll Extended Extra MatchLimit MultiLine NoAutoCapture NoMatch NoMemory NoSubstring NoUTF8Check NotBOL NotEOL NotEmpty Null UTF8 Ungreedy UnknownNode		

Εφαρμογή1
Η εφαρμογή αυτή είναι το Project136. Αποτελείται από μια φόρμα FMain ένα textbox με όνομα txtPattern, δύο TextArea με όνομα txtSubject και txtResults και ένα κουμπί btnMatch. Ο χρήστης γράφει το regular expression pattern στο στοιχείο ελέγχου txtPattern και το κείμενο προς έλεγχο στο txtSubject. Στην συνέχεια πατάει το κουμπί btnMatch και εκτελεί το κώδικα της regular expression εμφανίζοντας τα αποτελέσματα στο στοιχείο ελέγχου txtResults.

Για να χρησιμοποιήσουμε το εξάρτημα gb.pcre στην εφαρμογή μας πρέπει να το επιλέξουμε στην λίστα των components από το μενού Project-> Properties -> Componetns
Η γραφική διεπαφή της εφαρμογής είναι η παρακάτω



Εικόνα 124: Η γραφική διεπαφή του Project136

Ο κώδικας της εφαρμογής είναι ο παρακάτω:

```
1   ' Gambas class file
2
3   Private sRegExp As Regexp
4
5   Public Sub Form_Open()
6
7     Me.Center
8
9   End
10
11  Public Sub btnMatch_Click()
12
13    txtResults.Clear
14
15    If IsNull(txtPattern.Text) Or IsNull(txtSubject.Text) Then
    Return
16
17    sRegExp = New Regexp(txtSubject.Text, txtPattern.Text, 2)
18
19    If sRegExp.Offset = -1 Then Return
20
21    txtResults.Text = sRegExp.Text
22
```

```
23  End
```

Ανάλυση:
Στην γραμμή 3 δηλώνουμε ένα ιδιωτικό αντικείμενο RegExp
Στην γραμμή 5 ξεκινά η ρουτίνα ανοίγματος της φόρμας.
Στην γραμμή 7 τοποθετούμε στο κέντρο της οθόνης την φόρμα.
Στην γραμμή 11 ξεκινά η ρουτίνα του συμβάντος click του κουμπιού btnMatch, `btnMatch_Click()`
Στην γραμμή 13 καθαρίζουμε το κείμενο του στοιχείου ελέγχου txtResults.
Στην γραμμή 15 ελέγχουμε αν οι τιμές των txtPattern.Text και txtSubject.Text είναι κενές. Αν είναι επιστρέφουμε την ροή του προγράμματος εκτός ρουτίνας.
Στην γραμμή 17 δημιουργούμε την κανονική έκφραση. Η παράμετρος 2 είναι μια παράμετρος μεταγλώττισης και λέει ότι η κανονική έκφραση είναι κείμενο σε πολλές σειρές. Αυτό κάνει τους χαρακτήρες των pattern ^ και $ να ταιριάζουν κείμενο στην αρχή και στο τέλος κάθε γραμμής οπουδήποτε στο κείμενο.

Μετά την δημιουργία της κανονικής έκφρασης στην γραμμή 19 ελέγχουμε το Offset αυτής. Αν έχει την τιμή -1 σημαίνει ότι η κανονική έκφραση δεν επέστρεψε κάποιο αποτέλεσμα και έτσι η ροή του προγράμματος επιστρέφει εκτός ρουτίνας. Αυτό γίνεται διότι σε αυτήν την περίπτωση δεν θα μπορούσαμε να προσπελάσουμε την ιδιότητα sRegExp.Text και ο μεταγλωττιστής της Gambas θα εμφάνιζε ένα λάθος.

Στην περίπτωση που η ιδιότητα Offset έχει διαφορετική τιμή από το -1 στην γραμμή 21 εκχωρούμε το κείμενο της κανονικής έκφρασης στο στοιχείο ελέγχου txtResults.

Δημιουργώντας patterns για τις Regular Expressions

Όλη η επιτυχία σε μια κανονική έκφραση είναι η δημιουργία ενός σωστού pattern.
Η δημιουργία pattern είναι ένα μεγάλο κεφάλαιο και εδώ θα αναφερθούμε μόνο σε μερικές βασικές τεχνικές. Για περισσότερες πληροφορίες μπορείτε να δείτε την ιστοσελίδα https://gambaswiki.org/wiki/comp/gb.pcre

Με την εφαρμογή Project136 μπορούμε να δοκιμάσουμε τα παρακάτω παραδείγματα.

Εύρεση εκτυπώσιμων χαρακτήρων

Patterns

- . οποιοσδήποτε χαρακτήρας εκτός τον χαρακτήρα νέας γραμμής (newline)
- \C ένα byte, ακόμη και στην κατάσταση UTF-8
- \d ένα δεκαδικό ψηφίο
- \D ένα μη δεκαδικό ψηφίο
- \h ένα οριζόντιο whitespace χαρακτήρα (π.χ. space, tab, αλλά όχι newline)
- \H ένα μη οριζόντιο whitespace χαρακτήρα
- \R μια ακολουθία από newline χαρακτήρες
- \s έναν whitespace χαρακτήρα
- \S έναν μη whitespace χαρακτήρα

- \v έναν κάθετο whitespace χαρακτήρα (π.χ. newline ή CR)
- \V έναν μη κάθετο whitespace χαρακτήρα
- \w έναν χαρακτήρα "word"
- \W έναν όχι "word" χαρακτήρα
- \X μία ακολουθία Unicode

Στις PCRE, \d, \D, \s, \S, \w, και \W αναγνωρίζουν μόνο ASCII χαρακτήρες.

Παραδείγματα

Pattern	**Subject**	**Αποτέλεσμα**	**Σχόλια**
.	This is a text	T	Επιστρέφει οποιονδήποτε χαρακτήρα που βρίσκει πρώτο
\d	This is a number 1.45	1	Επιστρέφει το πρώτο δεκαδικό ψηφίο
\D	This is a text	T	Επιστρέφει το πρώτο μη δεκαδικό ψηφίο
\S	This is a text	T	Επιστρέφει το πρώτο μη whitespace χαρακτήρα

Επαναλήψεις

Patterns
Οι επαναλήψεις μέσα σε ένα pattern καθορίζονται από σύμβολα που ονομάζονται “προσδιοριστές” (quantifiers).
Στις επαναλήψεις διακρίνουμε τρεις περιπτώσεις προσδιοριστών, τους greedy, lazy και possesive.

Ένας greedy προσδιοριστής αρχικά θα προσπαθήσει να επαναλάβει το pattern όσες φορές είναι δυνατό και να δώσει το περισσότερο δυνατό αποτέλεσμα.
Ένας lazy προσδιοριστής αρχικά θα επαναλάβει το pattern τις λιγότερες δυνατές φορές όσο χρειάζεται και θα επιστρέψει το λιγότερο δυνατό αποτέλεσμα.
Ένας possessive προσδιοριστής είναι ένας τρόπος να εμποδίσουμε την μηχανή κανονικών εκφράσεων να δοκιμάσει όλες τις μεταθέσεις σε ένα pattern για λόγους απόδοσης. Ένας possesive προσδιοριστής επιστρέφει όλο το αποτέλεσμα ή τίποτα.

Για αναλυτική παρουσίαση των προσδιοριστών δείτε
http://www.regular-expressions.info/repeat.html
http://www.regular-expressions.info/possessive.html

0 ή 1

- ? 0 ή 1, greedy
- ?+ 0 ή 1, possessive

- ?? 0 ή 1, lazy

0 ή περισσότερα
- * 0 ή περισσότερα, greedy
- *+ 0 ή περισσότερα, possessive
- *? 0 ή περισσότερα, lazy

1 ή περισσότερα
- + 1 ή περισσότερα, greedy
- ++ 1 ή περισσότερα, possessive
- +? 1 ή περισσότερα, lazy

σε πλήθος n
- {n} ακριβώς n
- {n,m} τουλάχιστον n, όχι περισσότερα από m, greedy
- {n,m}+ τουλάχιστον n, όχι περισσότερα από m, possessive
- {n,m}? τουλάχιστον n, όχι περισσότερα από m, lazy
- {n,} n ή περισσότερα, greedy
- {n,}+ n ή περισσότερα, possessive
- {n,}? n ή περισσότερα, lazy

Παραδείγματα

Pattern	Subject	Αποτέλεσμα	Σχόλια
.?	This is a text	T	Επιστρέφει οποιονδήποτε χαρακτήρα 0 ή 1 φορά, greedy
.{6}	This is a text	This i	Επιστρέφει οποιονδήποτε χαρακτήρα για ακριβώς έξι φορές
\D+	This is a text	This is a text	Επιστρέφει το πρώτο μη δεκαδικό ψηφίο μία ή περισσότερες φορές

Κλάσεις χαρακτήρων

Οι κλάσεις χαρακτήρων είναι ομάδες χαρακτήρων όπως αλφαριθμητικά, μόνο αλφάβητο, μόνο το κενό, δεκαδικά ψηφία, κεφαλαία, μικρά κτλ.

Pattern

- [...] επιστρέφει αυτήν την κλάση
- [^...] δεν επιστρέφει αυτήν την κλάση
- [x-y] εύρος χαρακτήρων

- [[:xxx:]] επιστρέφει την ομάδα POSIX
- [[:^xxx:]] δεν επιστρέφει την ομάδα POSIX

Οι ομάδες των χαρακτήρων POSIX είναι:

- alnum αλφαριθμητική ομάδα
- alpha αλφαβητική ομάδα
- ascii 0-127
- blank κενό ή tab
- cntrl control χαρακτήρας
- digit δεκαδικό ψηφίο
- graph εκτυπώσιμοι χαρακτήρες, εκτός του κενού
- lower μικρά
- print εκτυπώσιμοι χαρακτήρες, εντός και του κενού
- punct εκτυπώσιμοι χαρακτήρες, εκτός αλφαριθμητικών
- space το κενό
- upper κεφαλαία
- word το ίδιο όπως το w
- xdigit δεκαεξαδικά ψηφία

Στις κανονικές εκφράσεις PCRE, οι χαρακτήρες POSIX είναι μόνο οι ASCII χαρακτήρες.

Παραδείγματα

Pattern	**Subject**	**Αποτέλεσμα**	**Σχόλια**
[[:alnum:]]+	this is a text and this IS A NUMBER 123.43	this	Επιστρέφει τα αλφαριθμητικά μία ή περισσότερες φορές.
[[:digit:]]+	this is a text and this IS A NUMBER 123.43	123	Επιστρέφει τα δεκαδικά ψηφία μία ή περισσότερες φορές.
[[:lower:]]+	this is a text and this IS A NUMBER 123.43	this	Επιστρέφει τα μικρά μία ή περισσότερες φορές.
[[:upper:]]+	this is a text and this IS A NUMBER 123.43	IS	Επιστρέφει τα κεφαλαία μία ή περισσότερες φορές.
[a-z]+	this is a text and this IS A NUMBER 123.43	this	Επιστρέφει μία ή περισσότερες φορές χαρακτήρες από το εύρος a-z

[A-Z]+	this is a text and this IS A NUMBER 123.43	IS	Επιστρέφει μία ή περισσότερες φορές χαρακτήρες από το εύρος A-Z
[a-zA-Z]+	this is a text and this IS A NUMBER 123.43	this	Επιστρέφει μία ή περισσότερες φορές χαρακτήρες από το εύρος a-z και A-Z

Σύνθετες κανονικές εκφράσεις

Οι κανονικές εκφράσεις μπορούν να ομαδοποιηθούν με τα σύμβολα (...) και να συνδυαστούν ώστε να δημιουργήσουν σύνθετες κανονικές εκφράσεις.

Όπως είδαμε η κανονική έκφραση *[[:upper:]]+* για την πρόταση ***this is a text and this IS A NUMBER 123.43*** επιστρέφει ***IS*** και αυτό γιατί επιστρέφει μόνο τους εκτυπώσιμους χαρακτήρες και όταν βρίσκει κενό σταματά. Αν θα θέλαμε να μας επιστρέψει όλα τα κεφαλαία μαζί με τα κενά θα έπρεπε να γράψουμε μια σύνθετη κανονική πρόταση όπως την ***([[:upper:]] +\h)+*** η οποία λέει επέστρεψε τα κεφαλαία ***[[:upper:]]*** με τον κενό χαρακτήρα ***+\h*** σαν ομάδα ***([[:upper:]]+\h)*** μία ή περισσότερες φορές +.

Ας δούμε ένα άλλο παράδειγμα. Αν θέλετε να εξάγετε το email από μια πρόταση θα πρέπει να γράψετε μια κανονική πρόταση σαν την παρακάτω.

[^.\s][a-zA-Z0-9!#$%&'*+-/=?^_`{}|~]+\.*[^.\s]@[^.\s-][[:alnum:]-.]+

Αν και φαίνεται πολύπλοκη θα δείτε ότι καθώς φτιάχνουμε μια κανονική έκφραση δεν είναι τόσο δύσκολη εργασία όσο φαίνεται από το τελικό αποτέλεσμα.

Ένα email είναι της μορφής user@myserver.com

Ένα email αποτελείται από το τοπικό μέρος (local part) και το μέρος του domain (domain part). Τα δύο αυτά μέρη τα χωρίζει ένα και μόνο ένα @. (local@domain)

Οπότε η κανονική έκφραση μας θα αποτελείται από ένα μέρος που θα αναφέρεται στο local part και ένα μέρος που θα αναφέρεται στο domain part.

Το local part δημιουργείται με τους εξής χαρακτήρες.

- Κεφαλαία και μικρά των Αγγλικών γραμμάτων (a–z, A–Z)
- ψηφία από το `0` μέχρι το `9`
- οι χαρακτήρες `! # $ % & ' * + - / = ? ^ _ ` { | } ~`
- ο χαρακτήρας τελεία `.` Ο οποίος δεν μπορεί να είναι ο πρώτος ή τελευταίος χαρακτήρας

Έτσι γνωρίζοντας το format αυτό μπορούμε να γράψουμε την κανονική έκφραση:
[a-zA-Z0-9!#$%&'*+-/=?^_`{}|~]+

Η κανονική αυτή έκφραση ταιριάζει τους χαρακτήρες που βρίσκονται μέσα στις αγκύλες [...] μία ή περισσότερες φορές επειδή μετά τις αγκύλες έχουμε τον προσδιοριστή +
Έτσι μέχρι στιγμής με αυτήν την κανονική έκφραση θα μπορούμε να βρίσκουμε κείμενο της μορφής user1, user#123, user1#%23 κτλ.
Τώρα θα πρέπει να γράψουμε κάποιο pattern για τον χαρακτήρα τελεία ο οποίος δεν μπορεί να είναι πρώτος ή τελευταίος. Το pattern αυτό είναι το **\.*** Εδώ λέμε ότι θα ταιριάζει κείμενο το οποίο θα είναι μια τελεία μηδέν ή περισσότερες φορές. Επειδή το σύμβολο της τελείας αποτελεί σύμβολο των κανονικών εκφράσεων πρέπει να βάλουμε τον προσδιοριστή \ πριν από αυτό για να δηλώσουμε ότι αυτό που ακολουθεί είναι χαρακτήρας και όχι σύμβολο κανονικής έκφρασης. Ο προσδιοριστής ***** στην συνέχεια προσδιορίζει ότι η τελεία μπορεί να επαναλαμβάνεται μηδέν ή περισσότερες φορές.

Αν προσθέσουμε και στο τέλος το σύμβολο @ τότε η κανονική μας έκφραση θα είναι
[a-zA-Z0-9!#$%&'*+-/=?^_`{}|~]+\.*@
και θα επιστρέφει ταιριάσματα όπως user@, user1@, user..123@ αλλά και .user@ πράγμα το οποίο δεν θέλουμε.
Για να ορίσουμε ότι δεν θέλουμε κάποιους χαρακτήρες το κάνουμε με το σύμβολο της αρνητικής κλάσης [^....]
Εμείς στο παράδειγμα μας δεν θέλουμε στην αρχή και στο τέλος τους χαρακτήρες τελεία και whitespaces. Έτσι στην αρχή και στο τέλος της κανονικής έκφρασης προσθέτουμε το **[^.\s]** και η κανονική έκφραση μας γίνεται
[^.\s][a-zA-Z0-9!#$%&'*+-/=?^_`{}|~]+\.*[^.\s]@

Αυτή η κανονική έκφραση επιστρέφει πλέον ταιριάσματα όπως user!@, user1@, user..123@ και όχι .user@ ή user.@

Μέχρι στιγμής η κανονική μας έκφραση έχει υπολογίσει το local part από το email.
Στην συνέχεια θα πρέπει να γράψουμε την κανονική έκφραση που θα επιστρέφει το domain part.

Οι κανόνες σύνταξης ενός domain είναι:

- Ένα domain name πρέπει να έχει ένα ή περισσότερα subdomain names και ένα top-level domain name.
- Οι επιτρεπτοί χαρακτήρες είναι αλφαριθμητικοί και η παύλα (-).
- Τα domain names δεν πρέπει να ξεκινούν με παύλα (-).

Επομένως λαμβάνοντας αυτά υπόψιν μπορούμε να γράψουμε μια κανονική έκφραση όπως η παρακάτω:

[^.\s-] η κανονική έκφραση αυτή επιστρέφει ταιριάσματα που δεν είναι τελεία, παύλα και whitespaces χαρακτήρες. Αυτοί οι χαρακτήρες δεν είναι επιτρεπτοί στην αρχή του domain name.

[[:alnum:]-.]+ η κανονική έκφραση αυτή επιστρέφει ταιριάσματα αλφαριθμητικών, τελείας και παύλας μία ή περισσότερες φορές.

Επομένως η κανονική έκφραση για το μέρος του domain θα είναι
[^.\s-][[:alnum:]-.]+

Η τελική λοιπόν κανονική έκφραση εξαγωγής email από μια πρόταση θα είναι:
[^.\s][a-zA-Z0-9!#$%&'*+-/=?^_`{}|~]+\.*[^.\s]@[^.\s-][[:alnum:]-.]+

Αν και το τελικό αποτέλεσμα φαίνεται περίπλοκο όπως είδαμε τα στάδια δημιουργίας της έκφρασης είναι απολύτως απλά και κατανοητά.

gb.settings

Το εξάρτημα αυτό επιτρέπει την χρήση αρχείων που λειτουργούν σαν αρχεία ρυθμίσεων (configuration files) για τις εφαρμογές σας.

Αρχεία ρυθμίσεων

Τα αρχεία αυτά συνήθως τα ονόματα τους έχουν την μορφή "xyzzy.conf" όπου "xyzzy" είναι το όνομα της εφαρμογής.

Σύνταξη των αρχείων ρυθμίσεων.

Ένα αρχείο ρυθμίσεων είναι ένα αρχείο απλού κειμένου που έχει συγκεκριμένη σύνταξη. Κάθε γραμμή του μπορεί να είναι:

- Μια κενή γραμμή. Κάθε κενή γραμμή αγνοείται από τον διερμηνευτή.
- Μια γραμμή με σχόλιο. Οι γραμμές με σχόλια ξεκινάνε με τον χαρακτήρα '#' ή ';'. Κάθε σχόλιο επίσης αγνοείται από τον διερμηνευτή.
- Μια γραμμή που περιέχει μια τιμή. Η σύνταξη αυτής της γραμμής είναι της μορφής `"Key=Value".`
- Μια γραμμή ομάδας ρυθμίσεων. Μια ομάδα ρυθμίσεων είναι πολλές γραμμές τιμών που συγκεντρώνονται σε ένα group. Μια τέτοια γραμμή ξεκινά με τον χαρακτήρα `'['` και τελειώνει με τον χαρακτήρα `']'`.

Μια τιμή προσπελάζεται από το όνομα του κλειδιού της τιμής. Αν το κλειδί βρίσκεται σε μια ομάδα ρυθμίσεων προσπελάζεται από το όνομα της ομάδας ακολουθεί ο χαρακτήρας slash και το όνομα του κλειδιού.

Παράδειγμα σύνταξης αρχείου ρυθμίσεων:

Το παρακάτω είναι το περιεχόμενο ενός αρχείου ρύθμισης. Εδώ μπορούμε να δούμε την σύνταξη του αρχείου ρύθμισης όπως περιγράφτηκε παραπάνω.

```
[Editor]
Font="Bitstream Vera Sans Mono"
Toolbar=1
CleanOnSave=0
```

```
ShowProc=1
ShowChange=1
ShowCurrent=1
UseRelief=1
Color[0]="&FFFFFF&"
Color[1]="&000000&"
Color[2]="&00C0A0&"
Color[3]="&00C000&"

[ExistingFile]
ExplorerType=0
Splitter="249,249"
TreeSequence="0"
PreviewImg=0
ShowHidden=0

[ExistingImage]
Path="/usr/share/icons/crystalsvg/64x64/devices"
Key="1"
ExplorerType=1
Splitter="212,434"
TreeSequence="1"
PreviewImg=1
ShowHidden=0
```

Κλάσεις

Το εξάρτημα περιέχει τις εξής κλάσεις

Settings

Δουλεύοντας με τις ρυθμίσεις, η κλάση Settings

Η κλάση αυτή διαχειρίζεται τα καθολικά αρχεία ρυθμίσεων.

Η κλάση αυτή μπορεί να χρησιμοποιηθεί όπως ένα αντικείμενο με το να δημιουργούμε και να κρύβουμε στιγμιότυπα αυτής κατά βούληση.
Η κλάση αυτή συμπεριφέρεται σαν ένας πίνακας που μπορεί να διαβαστεί και να γραφτεί.
Η κλάση μπορεί να δημιουργηθεί.

Σύνταξη:

```
Dim hSettings As Settings

hSettings = New Settings ( [ Path As String, Title As String ] )
```

Δημιουργεί ένα νέο αντικείμενο ρυθμίσεων από ένα αρχείο ρυθμίσεων το οποίο βρίσκεται στην διαδρομή Path.
Εάν δεν οριστεί η διαδρομή Path τότε χρησιμοποιείται το προκαθορισμένο αρχείο ρυθμίσεων της εφαρμογής.
Ο τίτλος Title είναι ένα σχόλιο που προστίθεται στην αρχή του αρχείου ρυθμίσεων.

Ιδιότητες, Μέθοδοι της κλάσης Settings	
Στατικές Ιδιότητες (Static Properties)	**Στατικές Μέθοδοι (Static Methods)**
DefaultDir	Array
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**
Keys Path	Clear Read Readwindow Reload Save Write Writewindow

Όταν χρησιμοποιείτε την κλάση αυτή στατικά τότε το προκαθορισμένο αρχείο ρυθμίσεων της εφαρμογής θα χρησιμοποιηθεί.

Το προκαθορισμένο αρχείο ρυθμίσεων για κάθε εφαρμογής βρίσκεται στον φάκελο του χρήστη στην διαδρομή: `User.Home &/ ".config/gambas3" &/ Application.Name & ".conf"`

Για παράδειγμα αν η εφαρμογή σας ονομάζεται “MyCalculator” τότε το προκαθορισμένο αρχείο ρυθμίσεων θα βρίσκεται στην διαδρομή, `~/.config/gambas3/MyCalculator.conf`

Εφαρμογή1
Για να χρησιμοποιήσουμε το εξάρτημα gb.settings στην εφαρμογή μας πρέπει να το επιλέξουμε στην λίστα των components από το μενού Project-> Properties -> Components

Η εφαρμογή αυτή είναι το Project137.
Η εφαρμογή αυτή κατά το άνοιγμα διαβάζει το αρχείο ρυθμίσεων και λαμβάνει τις τιμές Top, Left, Height, Width της φόρμας. Στην συνέχεια εμφανίζει τις τιμές αυτές σε labels και τις εκχωρεί στις αντίστοιχες ιδιότητες της φόρμας. Με αυτόν τον τρόπο η φόρμα αποκτά κατά το άνοιγμα την θέση και τις διαστάσεις από το αρχείο ρυθμίσεων.

Κατά το κλείσιμο η εφαρμογή αποθηκεύει αυτές τις τιμές στο αρχείο ρυθμίσεων.
Έτσι όταν η φόρμα ανοίγει έχει το μέγεθος και την θέση στην οποία είχε κλείσει την τελευταία φορά.

Η γραφική διεπαφή της εφαρμογής είναι η παρακάτω:

Εικόνα 125: Η διεπαφή της εφαρμογής

Ο κώδικας της εφαρμογής είναι ο παρακάτω:

```
1   ' Gambas class file
2
3   Public Sub Form_Open()
4
5   Me.Top = Settings["Window/Top", Me.Top]
6   lblTop.Text = Settings["Window/Top", Me.Top]
7
8   Me.Left = Settings["Window/Left", Me.Left]
9   lblLeft.Text = Settings["Window/Left", Me.Left]
10
11  Me.Height = Settings["Window/Height", Me.Height]
12  lblHeight.Text = Settings["Window/Height", Me.Height]
13
14  Me.Width = Settings["Window/Width", Me.Width]
15  lblWidth.Text = Settings["Window/Width", Me.Width]
16
17  End
18
19  Public Sub Form_Close()
20
21  Settings["Window/Top"] = Me.Top
22  Settings["Window/Left"] = Me.Left
23  Settings["Window/Height"] = Me.Height
24  Settings["Window/Width"] = Me.Width
25  End
```

Ανάλυση:
Στην γραμμή 3 ξεκινά η ρουτίνα ανοίγματος της φόρμας.
Στην γραμμή 5 εκχωρούμε στην ιδιότητα Top της φόρμας την τιμή που διαβάζουμε από το αρχείο ρυθμίσεων.
Η έκφραση Settings["Window/Top", Me.Top] λέει, διάβασε το κλειδί Top της ομάδας Window και αν δεν έχει τιμή επέστρεψε την τιμή Me.Top.
Με αυτόν τον τρόπο εκχωρούμε πάντα μια υπάρχουσα τιμή στην ιδιότητα Top της φόρμας με την έκφραση, Me.Top = Settings["Window/Top", Me.Top].

Στις γραμμές 8-16 εκχωρούμε τις υπόλοιπες τιμές με τον ίδιο τρόπο για τις ιδιότητες Left, Height, Width.

Στην γραμμή 19 ξεκινά η ρουτίνα κλεισίματος της φόρμας.
Εδώ εκχωρούμε στο αντικείμενο Settings, που είναι ένας πίνακας, τις τιμές των ιδιοτήτων της φόρμας. Οι τιμές αυτές γράφονται στο αρχείο ρυθμίσεων στην ομάδα Window και στα αντίστοιχα κλειδιά Top, Left, Height, Width.

Αν ανοίξουμε το αρχείο ρυθμίσεων που βρίσκεται στον κατάλογο ~/.config/gambas3 και έχει το όνομα Project137.conf θα δούμε ότι περιέχει κείμενο όμοιο με το παρακάτω.

```
[Window]
Top=13
Left=18
Height=154
Width=235
```

Απλά οι τιμές μπορεί να είναι διαφορετικές ανάλογα την θέση και τις διαστάσεις της φόρμας στην δικιά σας οθόνη.

gb.signal

Το εξάρτημα αυτό επιτρέπει την διαχείριση των σημάτων POSIX.
Ένα σήμα POSIX μπορείτε: να το ακυρώσετε, να το συλλάβετε μέσα σε έναν διαχειριστή συμβάντων ή να ακυρώσετε μια προηγούμενη ακύρωση ή σύλληψη.
Τα σήματα διακόπτουν τις διεργασίες αλλά η Gambas είναι μονο-νηματική διεργασία και έτσι μπορεί να δεχθεί σήματα σε έναν διαχειριστή συμβάντων.
Έτσι αν χρησιμοποιήσετε την εντολή SLEEP εντός ενός διαχειριστή συμβάντων δεν θα μπορείτε να διαχειριστείτε τα σήματα POSIX.

Κλάσεις

Το εξάρτημα περιέχει τις εξής κλάσεις

Signal .SignalHandler

Δουλεύοντας με τα σήματα του πυρήνα, η κλάση Signal

Η κλάση αυτή επιτρέπει την διαχείριση σημάτων POSIX.
Η κλάση είναι στατική.
Η κλάση μπορεί να συμπεριφερθεί σαν ένας πίνακας μόνο προς ανάγνωση.

Σταθερές της κλάσης Signals
Σταθερές (Constants)
SIGABRT SIGALRM SIGBUS SIGCHLD SIGCONT SIGFPE SIGHUP SIGILL SIGINT SIGIO SIGIOT SIGKILL SIGPIPE SIGPOLL SIGPROF SIGPWR SIGQUIT SIGSEGV SIGSTOP SIGSYS SIGTERM SIGTRAP SIGTSTP SIGTTIN SIGTTOU SIGURG SIGUSR1 SIGUSR2 SIGVTALRM SIGWINCH SIGXCPU SIGXFSZ

Οι προδιαγραφές του LINUX καθορίζουν τα παρακάτω σήματα:

Σήμα	**Περιγραφή**	**Αριθμός σήματος στο Linux x86**
SIGHUP	Αποσύνδεση	1
SIGINT	Διακοπή	2
SIGQUIT	Εγκατάλειψη και αποθήκευση του πυρήνα	3
SIGILL	Παράνομη οδηγία	4
SIGTRAP	Παγίδα ίχνους/διακοπής εκτέλεσης	5
SIGABRT	Η διεργασία εγκαταλείφθηκε	6
SIGBUS	Λάθος διαύλου: "πρόσβαση σε μη ορισμένο τμήμα της μνήμης"	7
SIGFPE	Εξαίρεση πραγματικού αριθμού: "εσφαλμένη αριθμητική λειτουργία"	8
SIGKILL	Άμεσος τερματισμός	9
SIGUSR1	Καθορισμένο σήμα από τον χρήστη 1	10
SIGSEGV	Παραβίαση της τμηματοποίησης	11
SIGUSR2	Καθορισμένο σήμα από τον χρήστη 2	12
SIGPIPE	Εγγραφή σε αγωγό χωρίς κανέναν να διαβάζει	13
SIGALRM	Το σήμα έχει εγερθεί από την λήξη του χρονικού ορίου.	14
SIGTERM	Αίτηση τερματισμού	15
SIGCHLD	Η διεργασία τέκνο έχει τερματιστεί, σταματήσει ή συνεχίζεται	17
SIGCONT	Συνέχιση αν έχει η διεργασία σταμάτησε	18
SIGSTOP	Προσωρινή διακοπή εκτέλεσης	19
SIGTSTP	Σήμα σταματήματος τερματικού	20
SIGTTIN	Διεργασία στο παρασκήνιο αποπειράται να διαβάσει από το tty	21
SIGTTOU	Διεργασία στο παρασκήνιο αποπειράται να γράψει στο tty	22

SIGURG	Επείγοντα δεδομένα διαθέσιμα στην υποδοχή	23
SIGXCPU	Το όριο χρόνου της CPU έχει λήξει	24
SIGXFSZ	Το όριο του αρχείου έχει υπερβεί	25
SIGVTALRM	Ο εικονικός χρόνος έχει λήξει.	26
SIGPROF	Λήξη του χρονιστή προφίλ	27
SIGWINCH	Το μέγεθος του παραθύρου έχει αλλάξει	28
SIGPOLL	Pollable συμβάν	29
SIGIO	I/O τώρα είναι δυνατόν	29
SIGPWR	Επανεκκίνηση, αδυναμία ισχύος	30
SIGSYS	Κακή κλήση συστήματος	31

Η εικονική κλάση .SignalHandler

Η κλάση Signal έχει σαν μέλος την εικονική κλάση .SignalHandler
Η εικονική κλάση αυτή επιτρέπει την διαχείριση ενός συγκεκριμένου σήματος.
Η κλάση αυτή είναι εικονική.
Η κλάση αυτή είναι στατική.

Μέθοδοι της κλάσης .SignalHandler
Σταθερές (Constants)
Catch Ignore Reset

Εφαρμογή1
Για να χρησιμοποιήσουμε το εξάρτημα gb.signal στην εφαρμογή μας πρέπει να το επιλέξουμε στην λίστα των components από το μενού Project-> Properties -> Components

Η εφαρμογή αυτή είναι το Project138.
Η εφαρμογή αυτή αποτελείται από την ρουτίνα `Main()` που είναι η ρουτίνα εκκίνησης και μέσα σε αυτήν διαχειριζόμαστε τα σήματα με την εικονική κλάση .SignalHandler και την ρουτίνα `Application_Signal()` που είναι ο διαχειριστής συμβάντων για τα σήματα που συλλαμβάνονται με την μέθοδο Catch.

Εκτελούμε σε ένα τερματικό το εκτελέσιμο Project138 και σε ένα άλλο τερματικό στέλνουμε διάφορα σήματα με μια εφαρμογή όπως η htop. Όταν η εφαρμογή πιάσει ένα σήμα `SIGTERM` τότε καλεί την ρουτίνα `Application_Signal()`.

Ο κώδικας της εφαρμογής είναι ο παρακάτω:

```
1  ' Gambas module file
2
3  Public Sub Application_Signal(Signal As Integer)
```

```
4
5     Print "I caught signal " & Signal
6
7   End
8
9   Public Sub Main()
10
11    ' Ignore the signal sent when you hit CTRL+Z in a terminal
12    Signal[Signal.SIGTSTP].Ignore
13
14    ' Catch the signal sent when you hit CTRL+C in a terminal
15    Signal[Signal.SIGTERM].Catch
16
17  End
```

Ανάλυση:
Στις γραμμές 3 -7 υπάρχει η ρουτίνα που καλείται όταν γίνει η σύλληψη ενός σήματος με την μέθοδο Catch. Η ρουτίνα αυτή είναι η `Application_Signal(Signal As Integer)`. Σαν όρισμα έχει μια μεταβλητή ακεραίου την Signal. Όταν ένα σήμα συλλαμβάνεται καλείται η συνάρτηση και η Gambas διοχετεύει τον αριθμό που αντιστοιχεί στο σήμα στην ρουτίνα Application_Signal().

Στην γραμμή 5 τυπώνουμε στην προκαθορισμένη έξοδο το μήνυμα "I caught signal " και ακολουθεί η τιμής της μεταβλητής Signal.

Στις γραμμές 7 με 17 υπάρχει η ρουτίνα Main(). Στην ρουτίνα αυτή διαχειριζόμαστε τις ενέργειες από τα σήματα που δέχεται η εφαρμογή. Στην γραμμή 12 αγνοούμε το σήμα SIGTSTP ενώ στην γραμμή 15 συλλαμβάνουμε το σήμα SIGTERM.

Έτσι αν ο χρήστης πατήσει στο τερματικό CTRL+Z αυτό αγνοείται, ενώ αν πατήσει CTRL+C τυπώνεται το μήνυμα I cought signal 15, αφού ο αριθμός του SIGTERM είναι το 15.

gb.eval

Το εξάρτημα αυτό επιτρέπει τον υπολογισμό εκφράσεων της Gambas στην κατάσταση εκτέλεσης μιας εφαρμογής.
Το εξάρτημα αυτό φορτώνεται αυτόματα όταν γίνεται χρήση της συνάρτησης Eval()

Κλάσεις
Περιέχει τις εξής κλάσεις

Expression
Highlight
System

Κλάση Expression

Η κλάση αυτή αντιπροσωπεύει μια δυναμική έκφραση της Gambas.

Η κλάση μπορεί να δημιουργηθεί.

Σύνταξη:

```
Dim hExpression As Expression

hExpression = New Expression ( )
```

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Expression		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Environment Text Value	Compile	-
Σταθερές (Constants)		
-		

Στην κονσόλα της Gambas που βρίσκεται στο κάτω μέρος τους IDE μπορούμε να ελέγχουμε εκφράσεις της γλώσσας όταν πατήσουμε ? και στην συνέχεια την έκφραση που θέλουμε να ελέγξουμε.

Για παράδειγμα:

Έκφραση	**Αποτέλεσμα**
?1>2	False
?10+1	11

Αν όμως θέλουμε να ελέγξουμε εκφράσεις κατά την εκτέλεση μιας εφαρμογής μπορούμε να χρησιμοποιήσουμε είτε την συνάρτηση Eval() είτε την κλάση Expression από το εξάρτημα gb.eval.

Για να χρησιμοποιήσουμε την κλάση Expression πρέπει να δημιουργήσουμε ένα αντικείμενο τύπου Expression και στην συνέχεια να θέσουμε την τιμή της ιδιότητας Text στην έκφραση προς έλεγχο.
Όταν προσπελάσουμε την ιδιότητα Value ή καλέσουμε την μέθοδο Compile γίνεται ο έλεγχος της έκφρασης.

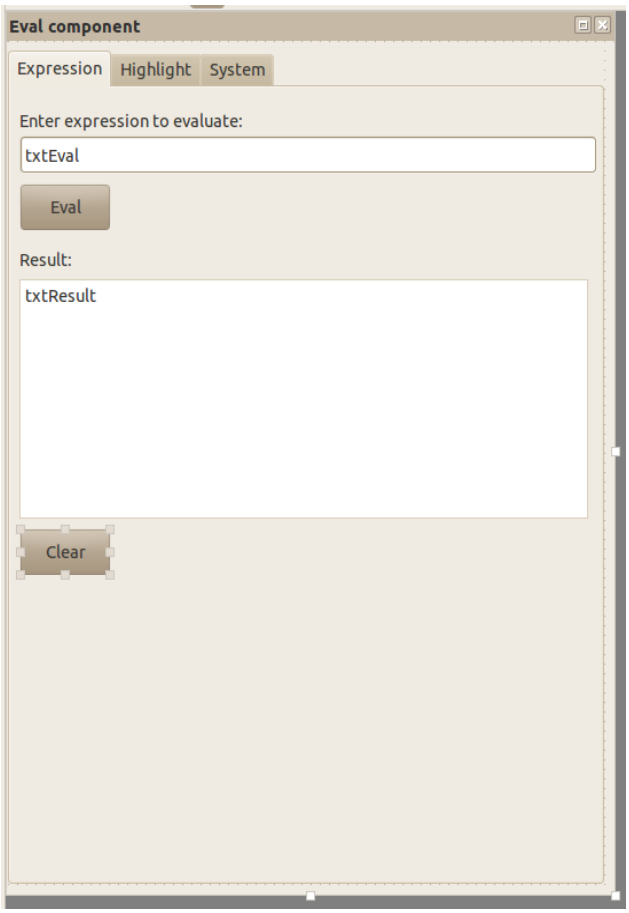
Εφαρμογή1
Για να χρησιμοποιήσουμε το εξάρτημα gb.eval στην εφαρμογή μας πρέπει να το επιλέξουμε στην λίστα των components από το μενού Project-> Properties -> Components

Η εφαρμογή αυτή είναι το Project139.
Αποτελείται από μια φόρμα Fmain με τίτλο “Eval component”, ένα αντικείμενο TabStrip με τρία tabs με τις ετικέτες Expression, Highlight, System οι οποίες αντιστοιχούν στις κλάσεις του εξαρτήματος gb.eval.
Σε κάθε ένα tab θα δημιουργήσουμε ένα GUI για κάθε μία κλάση.

Στο tab Expression έχουμε ένα textbox (txtEval) στο οποίο θα εισάγουμε την έκφραση προς έλεγχο, ένα κουμπί (btnEval) με το κείμενο Eval με το οποίο θα ελέγχουμε την έκφραση και textarea control (txtResult) στο οποίο θα εμφανίζονται τα αποτελέσματα των ελέγχων.

Τέλος το κουμπί btnClear καθαρίζει το κείμενο από τα στοιχεία ελέγχου txtEval, txtResult.

Ο χρήστης μπορεί να ελέγχει την μία έκφραση αμέσως μετά την άλλη και τα αποτελέσματα εμφανίζονται στο txtResult το ένα κάτω από το άλλο.



Εικόνα 126: Η διεπαφή του Project139

Ο κώδικας της εφαρμογής είναι ο παρακάτω:

```
1   ' Gambas class file
2
3   Public Sub Form_Open()
4     Me.Center
5     SystemInfo()
6   End
7
8   Public Sub btnEval_Click()
9
10    Dim hExpression As Expression
11    hExpression = New Expression
12
13    hExpression.Text = txtEval.Text
14    txtResult.Text &= Str(hExpression.Value) & gb.NewLine
15
16  Catch
17    txtResult.Text &= "Error code:" & Error.Code & ", " &
18  Error.Text & gb.NewLine
19    Error.Clear
20
21  End
22
23  Public Sub btnClear_Click()
24
25    txtEval.Clear
26    txtResult.Clear
27
28  End
29
30  Public Sub btnAnalyze_Click()
31
32    Dim sCode As String[]
33    Dim i As Integer = 0
34    Dim sElement As String = ""
35
36    sCode = Highlight.Analyze(txtCode.Text)
37
38    grdCode.Columns.Count = 3
39    grdCode.Rows.Count = sCode.Count
40    grdCode.Columns[0].Width = grdCode.Width / 3
41    grdCode.Columns[1].Width = grdCode.Width / 3
42    grdCode.Columns[2].Width = grdCode.Width / 3
43
44    grdCode.Columns[0].Text = "Symbols"
45    grdCode.Columns[1].Text = "Types"
```

```
46    grdCode.Columns[2].Text = "Positions"
47
48    For Each sElement In sCode
49      Print sCode[i]
50      grdCode[i, 0].Text = Highlight.Symbols[i]
51      grdCode[i, 1].Text = Highlight.Types[i]
52      grdCode[i, 2].Text = Highlight.Positions[i]
53      i += 1
54    Next
55
56    Editor1.Text = txtCode.Text
57    Editor1.Highlight = "gambas"
58    Editor1.ShowLineNumber = True
59
60  End
61
62  Public Sub btnClear2_Click()
63
64    txtCode.Clear
65    grdCode.Clear
66    Editor1.Clear
67
68  End
69
70  Public Sub SystemInfo()
71
72    Dim sElement As String = ""
73    Dim i As Integer = 0
74    Dim a As Integer = 0
75    Dim b As Integer = 0
76
77    grdSystem.Columns.Count = 2
78    grdSystem.Rows.Count = 20
79    grdSystem.Columns[0].Width = grdSystem.Width / 2
80    grdSystem.Columns[1].Width = grdSystem.Width / 2
81
82    grdSystem.Columns[0].Text = "Properties/Constants"
83    grdSystem.Columns[1].Text = "Values"
84
85    grdSystem[0, 0].Text = "Backtrace"
86
87    For Each sElement In System.Backtrace
88      grdSystem[i, 1].Text = System.Backtrace[i]
89      i += 1
90    Next
91
```

```
92    grdSystem[2, 0].Text = "ByteOrder"
93    grdSystem[2, 1].Text = System.ByteOrder
94
95    grdSystem[3, 0].Text = "Charset"
96    grdSystem[3, 1].Text = System.Charset
97
98    grdSystem[4, 0].Text = "Domain"
99    grdSystem[4, 1].Text = System.Domain
100
101   grdSystem[5, 0].Text = "Error"
102   grdSystem[5, 1].Text = System.Error
103
104   grdSystem[6, 0].Text = "FirstDayOfWeek"
105   grdSystem[6, 1].Text = System.FirstDayOfWeek
106
107   grdSystem[7, 0].Text = "Host"
108   grdSystem[7, 1].Text = System.Host
109
110   grdSystem[8, 0].Text = "Language"
111   grdSystem[8, 1].Text = System.Language
112
113   grdSystem[9, 0].Text = "Path"
114   grdSystem[9, 1].Text = System.Path
115
116   grdSystem[10, 0].Text = "RightToLeft"
117   grdSystem[10, 1].Text = Str(System.RightToLeft)
118
119   grdSystem[11, 0].Text = "User Group"
120   grdSystem[11, 1].Text = System.User.Group
121
122   grdSystem[12, 0].Text = "User Home"
123   grdSystem[12, 1].Text = System.User.Home
124
125   grdSystem[13, 0].Text = "User Id"
126   grdSystem[13, 1].Text = System.User.Id
127
128   grdSystem[14, 0].Text = "User Name"
129   grdSystem[14, 1].Text = System.User.Name
130
131   grdSystem[15, 0].Text = "Archtecture"
132   grdSystem[15, 1].Text = System.Architecture
133
134   grdSystem[16, 0].Text = "Family"
135   grdSystem[16, 1].Text = System.Family
136
```

```
137     grdSystem[17, 0].Text = "FullVersion"
138     grdSystem[17, 1].Text = System.FullVersion
139
140     grdSystem[18, 0].Text = "Version"
141     grdSystem[18, 1].Text = System.Version
142
143     grdSystem[19, 0].Text = "Datatypes"
144
145     a = grdSystem.Rows.Count
146     b = 0
147
148     For Each sElement In System.Datatypes
149       grdSystem.Rows.Count += 1
150       grdSystem[a, 1].Text = System.Datatypes[b]
151       a += 1
152       b += 1
153     Next
154
155     grdSystem.Rows.Count += 1
156     grdSystem[grdSystem.Rows.Count - 1, 0].Text = "Keywords"
157
158     a = grdSystem.Rows.Count
159     b = 0
160
161     For Each sElement In System.Keywords
162       grdSystem.Rows.Count += 1
163       grdSystem[a, 1].Text = System.Keywords[b]
164       a += 1
165       b += 1
166     Next
167
168   End
```

Ανάλυση:
Στην γραμμή 3 ξεκινά η ρουτίνα `Form_Open()` και στην γραμμή 4 τοποθετούμε την φόρμα στο κέντρο της οθόνης.

Στην γραμμή 8 ξεκινά η ρουτίνα `btnEval_Click()` που καλείται όταν πατηθεί το κουμπί btnEval.
Στις γραμμές 10 και 11 δημιουργούμε ένα αντικείμενο Expression.
Στην γραμμή 13 θέτουμε την τιμή Text του αντικειμένου `hExpression` ως την τιμή που έχει η ιδιότητα Text του αντικειμένου `txtEval`.

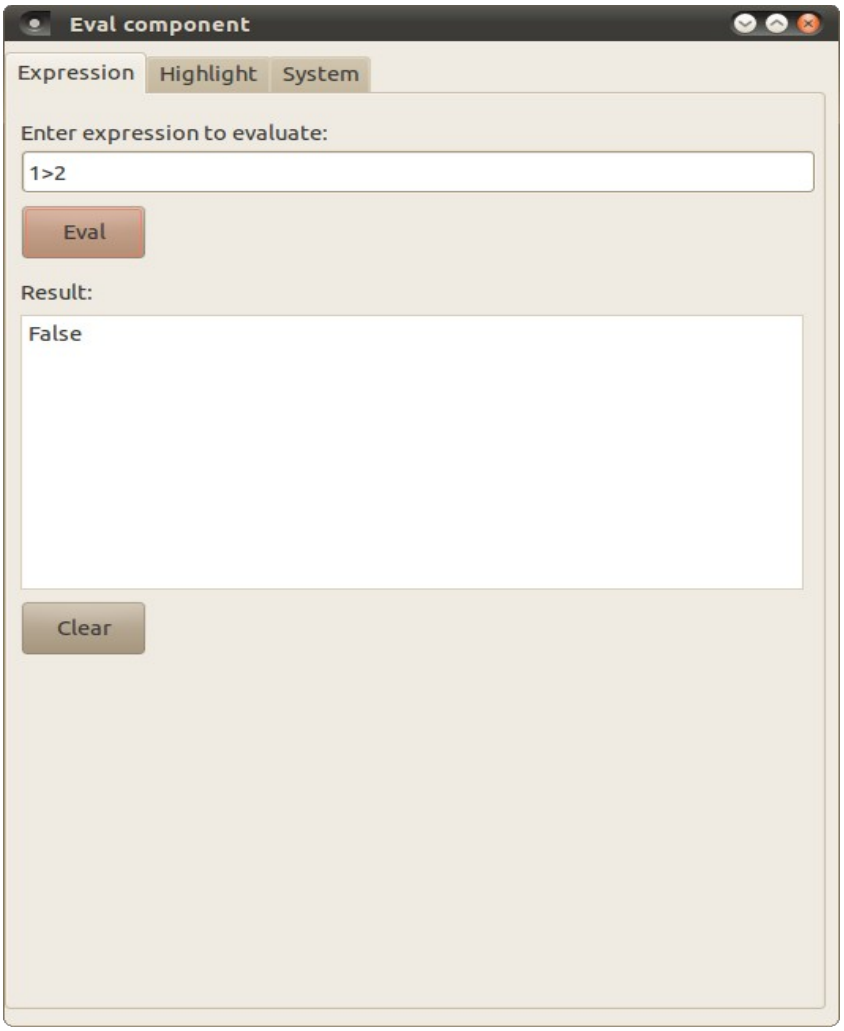
Στην γραμμή 14 προσθέτουμε στην τιμή της ιδιότητας Text του αντικειμένου txtResult την τιμή Value του αντικειμένου hExpression και ένα χαρακτήρα νέας γραμμής.

Με αυτόν τον τρόπο όταν ο χρήστης υπολογίσει μια νέα έκφραση το αποτέλεσμα θα τυπωθεί στην νέα κενή γραμμή κάτω από το προηγούμενο αποτέλεσμα.
Επίσης πρέπει να μετατρέψουμε την τιμή `hExpression.Value` που είναι τύπου Variant σε τύπο δεδομένων String για να εμφανιστεί ως κείμενο στο textArea control.
Αυτό γίνεται με την συνάρτηση Str() που μετατρέπει μια έκφραση σε κείμενο.

Έτσι τα μέρη που απαρτίζουν την έκφραση `txtResult.Text &= Str(hExpression.Value) & gb.NewLine` είναι:

Υπολόγισε την τιμή της έκφρασης	`hExpression.Value`
Μετέτρεψε την τιμή της έκφρασης σε String	`Str(hExpression.Value)`
Πρόσθεσε στο τέλος ένα χαρακτήρα νέας γραμμής	`Str(hExpression.Value) & gb.NewLine`
Πρόσθεσε στο κείμενο που υπάρχει στο `txtResult.Text` το νέο String	`txtResult.Text &= Str(hExpression.Value) & gb.NewLine`

Έξοδος:



Εικόνα 127: Παράδειγμα υπολογισμού μιας έκφρασης

Κλάση Highlight

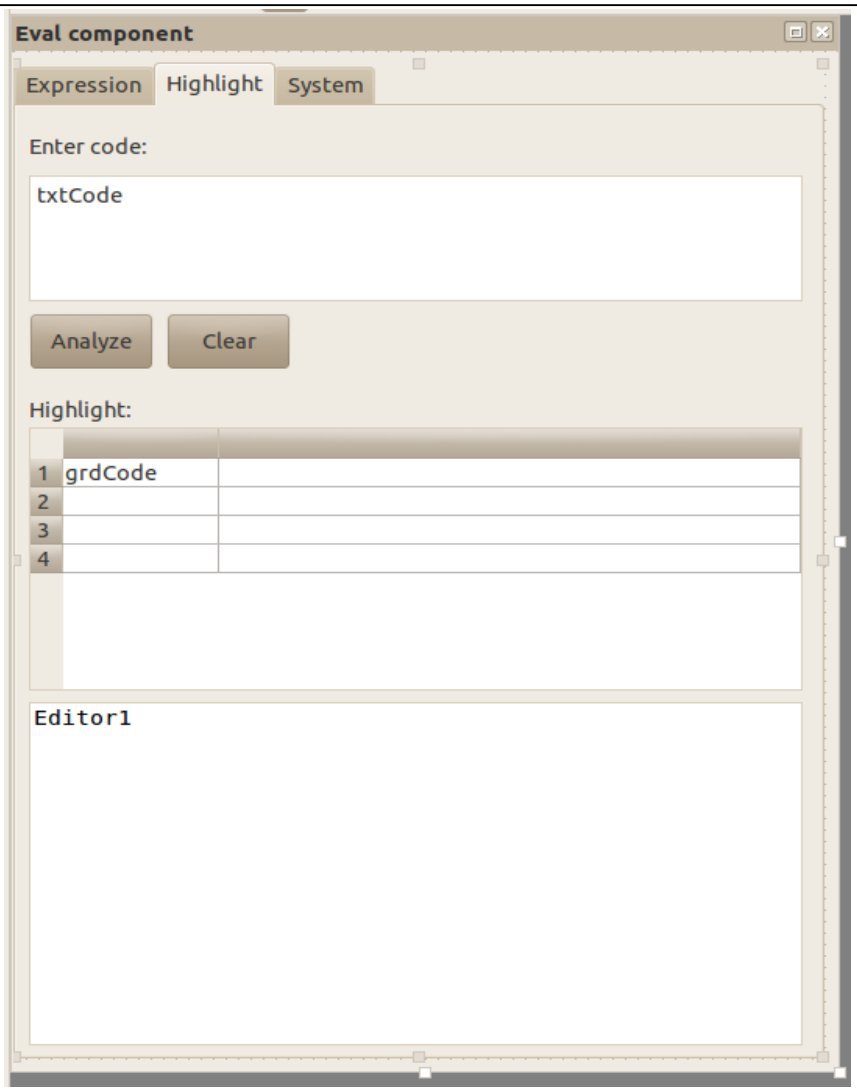
Η κλάση αυτή παρέχει μερικά χρήσιμα χαρακτηριστικά για την ανάλυση κειμένου που είναι κώδικας της γλώσσας Gambas έτσι ώστε να μπορεί ο προγραμματιστής να χρωματίσει τις λέξεις για ευκολότερη οπτική αναγνώριση.

Η κλάση δεν μπορεί να δημιουργηθεί.
Η κλάση είναι στατική.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Highlight		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Positions Symbols TextAfter Types	Analyze Purge	-
Σταθερές (Constants)		
Alternate Background Breakpoint Comment Current CurrentLine DataType Error Function Help Highlight Keyword Normal Number Operator Preprocessor Selection String Symbol		

Εφαρμογή2
Για να χρησιμοποιήσουμε το εξάρτημα gb.eval στην εφαρμογή μας πρέπει να το επιλέξουμε στην λίστα των components από το μενού Project-> Properties -> Components
Η εφαρμογή επίσης κάνει χρήση των εξαρτημάτων gb.qt4 και gb.qt4.ext. Από τα εξαρτήματα αυτά χρησιμοποιούμε το control “TextEditor” το οποίο υλοποιεί τον χρωματισμό των λέξεων.

Η εφαρμογή αυτή είναι η συνέχεια του Project139.
Στο Tab “Highlight” έχουμε μια περιοχή κειμένου TextArea στην οποία γράφουμε τον κώδικα, δύο κουμπιά εκ των οποίων το πρώτο αναλύει τον κώδικα και το δεύτερο καθαρίζει τις τιμές κειμένου από τα στοιχεία ελέγχου.
Επίσης έχουμε ένα πλέγμα τιμών grdCode, στο οποίο εμφανίζουμε το αποτέλεσμα της μεθόδου Analyze() και έναν Editor από το εξάρτημα gb.qt4.ext ο οποίος υλοποιεί τον χρωματισμό του κειμένου.


Eval component
Expression
Highlight
System
Enter code:
txtCode
Analyze
Clear
Highlight:
1 grdCode
2
3
4
Editor1

Ο κώδικας της εφαρμογής είναι ο παρακάτω:

```
29  Public Sub btnAnalyze_Click()
30
31    Dim sCode As String[]
32    Dim i As Integer = 0
33    Dim sElement As String = ""
34
35    sCode = Highlight.Analyze(txtCode.Text)
36
37    grdCode.Columns.Count = 3
38    grdCode.Rows.Count = sCode.Count
39    grdCode.Columns[0].Width = grdCode.Width / 3
40    grdCode.Columns[1].Width = grdCode.Width / 3
41    grdCode.Columns[2].Width = grdCode.Width / 3
42
43    grdCode.Columns[0].Text = "Symbols"
44    grdCode.Columns[1].Text = "Types"
45    grdCode.Columns[2].Text = "Positions"
46
47    For Each sElement In sCode
48      grdCode[i, 0].Text = Highlight.Symbols[i]
49      grdCode[i, 1].Text = Highlight.Types[i]
50      grdCode[i, 2].Text = Highlight.Positions[i]
51      i += 1
52    Next
53
54    Editor1.Text = txtCode.Text
55    Editor1.Highlight = Highlight.Gambas
56    Editor1.Flags[Editor1.ShowLineNumbers] = True
57    Editor1.HighlightAll
58
59  End
60
61  Public Sub btnClear2_Click()
62
63    txtCode.Clear
64    grdCode.Clear
65    Editor1.Clear
66
67  End
```

Ανάλυση:
Στην γραμμή 29 ξεκινά η ρουτίνα `btnAnalyze_Click()`.
Στην γραμμή 31 δηλώνουμε ένα δυναμικό πίνακα τύπου String. Ο πίνακας αυτός θα κρατά την επιστροφή της συνάρτησης Analyze() της κλάσης Highlight.
Στην γραμμή 32 δηλώνουμε μια μεταβλητή ακεραίου. Την μεταβλητή αυτή θα την χρησιμοποιήσουμε για την απαρίθμηση του πίνακα sCode που δηλώσαμε στην γραμμή 31.
Στην γραμμή 33 δηλώνουμε μια μεταβλητή τύπου String την οποία θα χρησιμοποιήσουμε στην απαρίθμηση του πίνακα sCode.

Στην γραμμή 35 αναλύουμε τον κώδικα του κειμένου από το αντικείμενο txtCode.Text.
Η συνάρτηση Analyze() επιστρέφει έναν πίνακα ο οποίος εκχωρείται στον πίνακα sCode.
Στις γραμμές 37 με 45 αρχικοποιούμε το αντικείμενο GridView της φόρμας μας.
Στην γραμμή 37 δηλώνουμε ότι θα έχει 3 στήλες. Στην γραμμή 38 δηλώνουμε ότι θα έχει τόσες γραμμές όσο το πλήθος των στοιχείων του πίνακα sCode. Αυτό γίνεται για να εκχωρηθεί κάθε τιμή του πίνακα σε μία γραμμή του πλέγματος.

Στις γραμμές 39 με 41 ορίζουμε το μήκος των στηλών στο 1/3 του μήκους του αντικειμένου πλέγματος. Έτσι οι στήλες θα έχουν το ίδιο μήκος.

Στις γραμμές 43 με 45 ορίζουμε το κείμενο των επικεφαλίδων των στηλών.

Στις γραμμές 47 με 52 απαριθμούμε τον πίνακα.
Για κάθε στοιχείο του πίνακα sCode σε κάθε γραμμή i του πλέγματος, εκχωρούμε στην 1η, 2η, 3η στήλη το i-οστό στοιχείο από τους πίνακες Highlight.Symbols[i], Highlight.Types[i], Highlight.Positions[i]

Για παράδειγμα αν η έκφραση είναι “PRINT x” τότε:
sCode[0]=”PRINT”
Highlight.Symbols[0]=”PRINT”
Highlight.Types[0]=2
Highlight.Positions[0]=0

και

sCode[1]=”x”
Highlight.Symbols[1]=”x”
Highlight.Types[1]=5
Highlight.Positions[1]=6

Κατά την απαρίθμηση του πίνακα sCode για το πρώτο στοιχείο θα έχουμε,
i=0

```
    grdCode[i, 0].Text = Highlight.Symbols[i]
    grdCode[i, 1].Text = Highlight.Types[i]
    grdCode[i, 2].Text = Highlight.Positions[i]
```

```
grdCode[0, 0].Text = Highlight.Symbols[0]      'grdCode[0, 0] 1η γραμμή 1η στήλη
grdCode[0, 1].Text = Highlight.Types[0]        'grdCode[0, 1] 1η γραμμή 2η στήλη
grdCode[0, 2].Text = Highlight.Positions[0]    'grdCode[0, 2] 1η γραμμή 3η στήλη
```

Η γραμμή του πλέγματος θα ήταν:

Symbols	**Types**	**Position**
PRINT	2	0

Όπου Type 2 είναι η σταθερά **Keyword**
Μετά την ανάλυση του κώδικα οι πίνακες Highlight.Symbols[], Highlight.Types[], Highlight.Positions[] συμπληρώνονται αυτόματα με τις ανάλογες τιμές για κάθε σύμβολο του κώδικα. Έτσι ο πίνακας Highlight.Symbols[] κρατά τα σύμβολα, ο πίνακας Highlight.Types[] κρατά τον τύπο του συμβόλου ο οποίος είναι μια αριθμητική τιμή ακεραίου που αντιστοιχεί σε μια από τις σταθερές της κλάσης Highlight, τέλος ο πίνακας Highlight.Positions[] κρατά την θέση που βρίσκεται το σύμβολο μέσα στο string κειμένου.
Εμείς για να συμπληρώσουμε μια γραμμή για στο πλέγμα grdCode λαμβάνουμε τις τιμές από τους πίνακες αυτούς για το ίδιο index. Έτσι για παράδειγμα, για index=0 θα λάβουμε τις τιμές συμβόλου, τύπου, και θέσης για το πρώτο σύμβολο.

Στην γραμμή 52 αυξάνουμε την τιμή της μεταβλητής i κατά μία μονάδα. Η αύξηση αυτή θα οδηγήσει στην εκχώρηση των τιμών στην επόμενη γραμμή του πλέγματος.

Στην γραμμή 55 εκχωρούμε την τιμή του txtCode.Text στο αντικείμενο Editor και συγκεκριμένα στην ιδιότητα του Text.

Στην γραμμή 56 ορίζουμε ότι ο χρωματισμός θα είναι του τύπου κώδικα της Gambas.
Στην γραμμή 57 ορίζουμε την σημαία της εμφάνισης των αριθμών των γραμμών στην τιμή TRUE.

Τέλος στην γραμμή 58 ενεργοποιούμε το highlight του κειμένου.

Στην γραμμή 61 ξεκινά η ρουτίνα `btnClear2_Click()` που καλείται όταν πατηθεί το κουμπί btnClear2.
Στην ρουτίνα αυτή καθαρίζουμε τις τιμές κειμένου από τα αντικείμενα txtCode, grdCode, Editor1.

Κλάση System

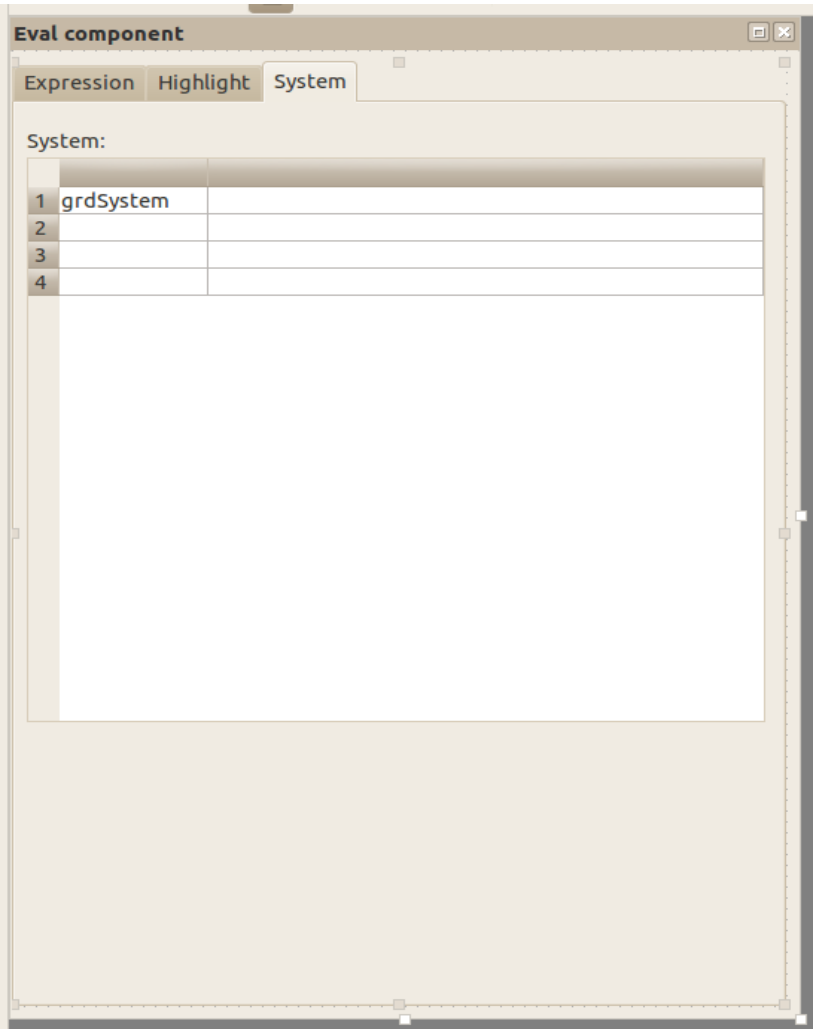
Η κλάση αυτή υλοποιεί ξανά την κλάση System του εξαρτήματος gb.
Η κλάση αυτή δίνει πληροφορίες σχετικά με το λειτουργικό σύστημα.

Η κλάση δεν μπορεί να δημιουργηθεί.
Η κλάση είναι στατική.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης System		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Backtrace ByteOrder Charset Domain Error FirstDayOfWeek Host Language Path RightToLeft User Datatypes Keywords	GetExternSymbol	-
Σταθερές (Constants)		
Architecture Family FullVersion Version		

Εφαρμογή3
Για να χρησιμοποιήσουμε το εξάρτημα gb.eval στην εφαρμογή μας πρέπει να το επιλέξουμε στην λίστα των components από το μενού Project-> Properties -> Components

Η εφαρμογή αυτή είναι η συνέχεια του Project139.
Στο Tab “System” έχουμε ένα πλέγμα τιμών grdSystem, στο οποίο θα εμφανίζουμε τις τιμές από την κλάση System.



Εικόνα 128: Η διεπαφή για την εμφάνιση τιμών από την κλάση System

Ο κώδικας της εφαρμογής είναι ο παρακάτω:

```
3    Public Sub Form_Open()
4      Me.Center
5      SystemInfo()
6    End

...  ...

70   Public Sub SystemInfo()
71
72     Dim sElement As String = ""
73     Dim i As Integer = 0
74     Dim a As Integer = 0
75     Dim b As Integer = 0
76
77     grdSystem.Columns.Count = 2
78     grdSystem.Rows.Count = 20
79     grdSystem.Columns[0].Width = grdSystem.Width / 2
80     grdSystem.Columns[1].Width = grdSystem.Width / 2
81
82     grdSystem.Columns[0].Text = "Properties/Constants"
83     grdSystem.Columns[1].Text = "Values"
84
85     grdSystem[0, 0].Text = "Backtrace"
86
87     For Each sElement In System.Backtrace
88       grdSystem[i, 1].Text = System.Backtrace[i]
89       i += 1
90     Next
91
92     grdSystem[2, 0].Text = "ByteOrder"
93     grdSystem[2, 1].Text = System.ByteOrder
94
95     grdSystem[3, 0].Text = "Charset"
96     grdSystem[3, 1].Text = System.Charset
97
98     grdSystem[4, 0].Text = "Domain"
99     grdSystem[4, 1].Text = System.Domain
100
101    grdSystem[5, 0].Text = "Error"
102    grdSystem[5, 1].Text = System.Error
103
104    grdSystem[6, 0].Text = "FirstDayOfWeek"
105    grdSystem[6, 1].Text = System.FirstDayOfWeek
106
```

```
107   grdSystem[7, 0].Text = "Host"
108   grdSystem[7, 1].Text = System.Host
109
110   grdSystem[8, 0].Text = "Language"
111   grdSystem[8, 1].Text = System.Language
112
113   grdSystem[9, 0].Text = "Path"
114   grdSystem[9, 1].Text = System.Path
115
116   grdSystem[10, 0].Text = "RightToLeft"
117   grdSystem[10, 1].Text = Str(System.RightToLeft)
118
119   grdSystem[11, 0].Text = "User Group"
120   grdSystem[11, 1].Text = System.User.Group
121
122   grdSystem[12, 0].Text = "User Home"
123   grdSystem[12, 1].Text = System.User.Home
124
125   grdSystem[13, 0].Text = "User Id"
126   grdSystem[13, 1].Text = System.User.Id
127
128   grdSystem[14, 0].Text = "User Name"
129   grdSystem[14, 1].Text = System.User.Name
130
131   grdSystem[15, 0].Text = "Archtecture"
132   grdSystem[15, 1].Text = System.Architecture
133
134   grdSystem[16, 0].Text = "Family"
135   grdSystem[16, 1].Text = System.Family
136
137   grdSystem[17, 0].Text = "FullVersion"
138   grdSystem[17, 1].Text = System.FullVersion
139
140   grdSystem[18, 0].Text = "Version"
141   grdSystem[18, 1].Text = System.Version
142
143   grdSystem[19, 0].Text = "Datatypes"
144
145   a = grdSystem.Rows.Count
146   b = 0
147
148   For Each sElement In System.Datatypes
149     grdSystem.Rows.Count += 1
150     grdSystem[a, 1].Text = System.Datatypes[b]
151     a += 1
```

```
152      b += 1
153    Next
154
155    grdSystem.Rows.Count += 1
156    grdSystem[grdSystem.Rows.Count - 1, 0].Text = "Keywords"
157
158    a = grdSystem.Rows.Count
159    b = 0
160
161    For Each sElement In System.Keywords
162      grdSystem.Rows.Count += 1
163      grdSystem[a, 1].Text = System.Keywords[b]
164      a += 1
165      b += 1
166    Next
167
168  End
```

Ανάλυση:
Στην γραμμή 5 της ρουτίνας Form_Open() καλούμε την ρουτίνα SystemInfo().
Έτσι οι τιμές στο πλέγμα grdSystem συμπληρώνονται κατά το άνοιγμα της φόρμας.

Η SystemInfo() είναι μια ρουτίνα που έχουμε γράψει για να εκχωρούμε τις τιμές από την κλάση System στα κελιά του πλέγματος grdSystem.

Στην γραμμή 72 δηλώνουμε μια μεταβλητή τύπου String. Στις γραμμές 73 – 75 δηλώνουμε τρεις ακεραίους και τους αποδίδουμε την αρχική τιμή 0.

Στις γραμμές 77 με 83 αρχικοποιούμε το πλέγμα ορίζοντας πλήθος στηλών και γραμμών και το κείμενο των επικεφαλίδων.
Στην γραμμή 77 ορίζουμε το πλήθος των στηλών, στην γραμμή 78 το πλήθος των γραμμών, στις γραμμές 79, 80 ορίζουμε το μήκος των στηλών στο 1/2 του μήκους του grdSystem.

Από την γραμμή 85 και κάτω αρχίζουμε να συμπληρώνουμε τις γραμμές του πλέγματος.
Στην πρώτη στήλη θα αναγράφεται το όνομα της σταθεράς ή της ιδιότητας.
Στην δεύτερη στήλη θα αναγράφεται η τιμή της σταθεράς ή της ιδιότητας.
Μερικές τιμές βρίσκονται εντός πινάκων όπως οι Backtrace, Datatypes και Keywords. Σε αυτήν την περίπτωση αυτοί οι πίνακες θα απαριθμούνται και οι τιμές θα αναγράφονται στην δεύτερη στήλη του πλέγματος grdSystem.

Στις γραμμές 87 με 90 απαριθμούμε τον πίνακα System.Backtrace.
Στην γραμμή 88 προσθέτουμε μία γραμμή για κάθε στοιχείο του πίνακα και στην γραμμή 89 αυξάνουμε την μεταβλητή i κατά μία μονάδα. Η μεταβλητή i ορίζει την επόμενη γραμμή στην οποία θα γραφεί η τιμή από τον πίνακα System.Backtrace.

Στις γραμμές 92 με 141 προσθέτουμε τις τιμές των σταθερών και των ιδιοτήτων στο πλέγμα

grdSystem.

Ιδιαίτερη αναφορά πρέπει να κάνουμε για την γραμμή 117.
Η τιμή της ιδιότητας System.RightToLeft είναι τύπου boolean. Για να μετατρέψουμε την τιμή αυτή σε string θα πρέπει να χρησιμοποιήσουμε την συνάρτηση Str().

Στην γραμμή 78 είχαμε ορίσει ότι το πλέγμα θα έχει 20 γραμμές.
Στην συνέχεια κατά την απαρίθμηση των πινάκων System.Datatypes και System.Keywords θα πρέπει να αυξάνουμε κατά μία γραμμή το πλήθος των γραμμών όταν θα απαριθμείτε ένα ένα τα στοιχεία από τους πίνακες αυτούς.

Για τον λόγο αυτό θα χρησιμοποιήσουμε δύο μεταβλητές, την a και b.

Η a θα κρατά τον αριθμό της τελευταίας γραμμής και η b θα κρατά το index του πίνακα που θα απαριθμείται.

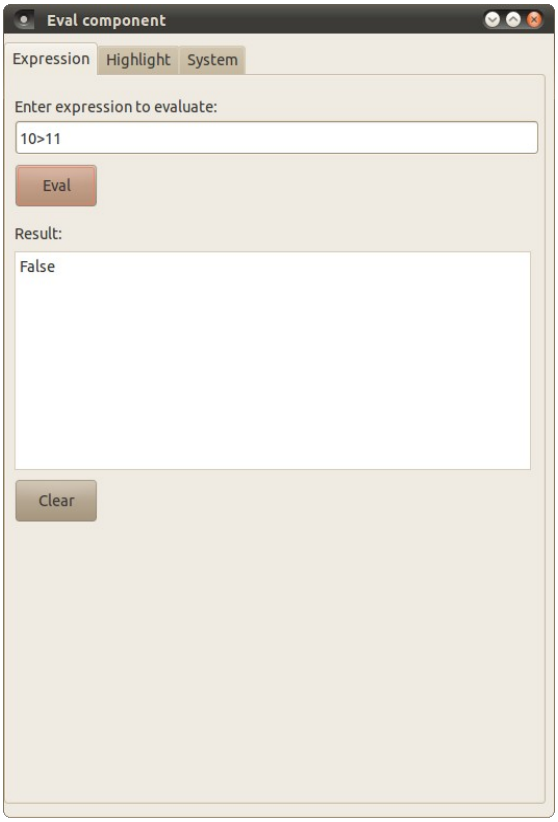
Στην γραμμή 145 εκχωρούμε την τιμή του πλήθους των γραμμών του πλέγματος στην μεταβλητή a.

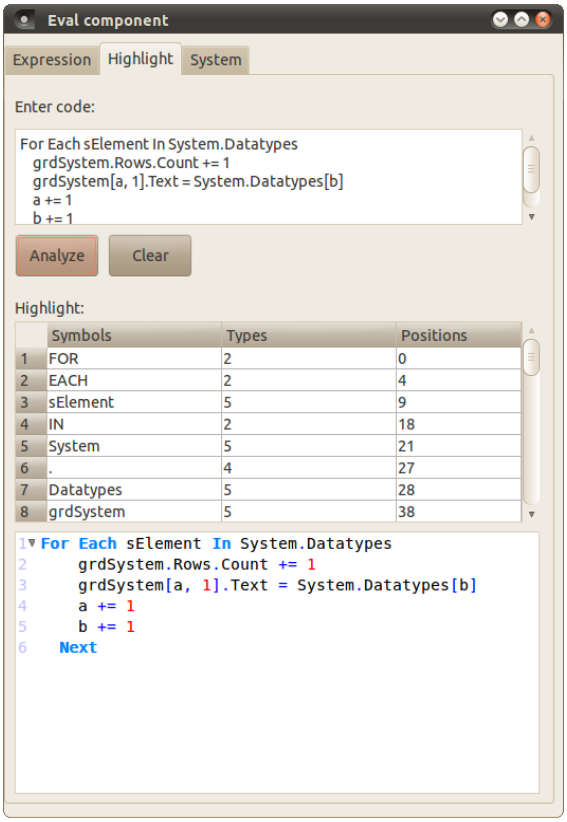
Στην γραμμή 146 αρχικοποιούμε την μεταβλητή b με την τιμή 0.

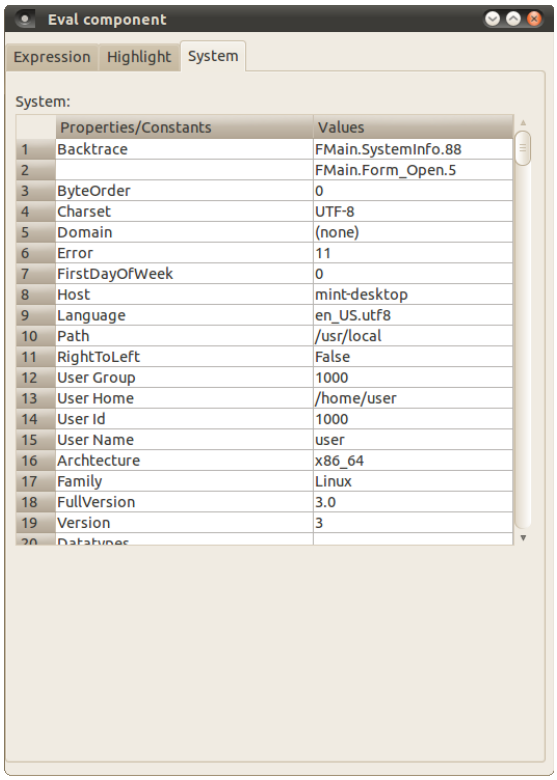
Στις γραμμές 148 με 153 γίνεται η απαρίθμηση του πίνακα System.Datatypes.
Στην γραμμή 149 αυξάνουμε το πλήθος των γραμμών του πλέγματος κατά μία μονάδα.
Στην γραμμή 150 γράφουμε στην τελευταία γραμμή την τιμή του στοιχείου του πίνακα.
Στην γραμμή 151 αυξάνουμε την μεταβλητή a κατά μία μονάδα και το ίδιο γίνεται και στην γραμμή 152 για την μεταβλητή b.
Με αυτόν τον τρόπο στην τελευταία γραμμή a του πλέγματος θα προστίθενται οι νέες τιμές από την απαρίθμηση του πίνακα.

Ακριβώς τον ίδιο αλγόριθμο χρησιμοποιούμε και για την απαρίθμηση του πίνακα System.Keywords στις γραμμές 161 – 166.

Το Project139 επιδεικνύει την λειτουργία του εξαρτήματος gb.eval.
Παρακάτω φαίνονται εικόνες κατά την εκτέλεση της εφαρμογής.






Eval component
Expression
Highlight
System
System:
Properties/Constants
Values
1 Backtrace FMain.SystemInfo.88
2 FMain.Form_Open.5
3 ByteOrder 0
4 Charset UTF-8
5 Domain (none)
6 Error 11
7 FirstDayOfWeek 0
8 Host mint-desktop
9 Language en_US.utf8
10 Path /usr/local
11 RightToLeft False
12 User Group 1000
13 User Home /home/user
14 User Id 1000
15 User Name user
16 Archtecture x86_64
17 Family Linux
18 FullVersion 3.0
19 Version 3

gb.args

Το εξάρτημα αυτό υλοποιεί έναν αναλυτή παραμέτρων της γραμμής εντολών ο οποίος βασίζεται στην συνάρτηση GNU `getopt()`.

Κλάσεις
Περιέχει τις εξής κλάσεις

Args

Κλάση Args

Η κλάσση υλοποιεί την κλάσση Args στο εξάρτημα gb.
Η κλάσση είναι στατική.
Η κλάσση λειτουργεί ως έναν στατικό πίνακα μόνο για ανάγνωση. (read-only static array).
Η κλάσση είναι στατικά απαριθμίσημη με τις λέξεις κλειδιά FOR EACH.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Args		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Abort All Count Max	Begin Copy End Get GetFloat GetInteger Has HelpText	-
Σταθερές (Constants)		
-		

Εφαρμογή1
Για να χρησιμοποιήσουμε το εξάρτημα gb.args στην εφαρμογή μας πρέπει να το επιλέξουμε στην λίστα των components από το μενού Project-> Properties -> Components

Η εφαρμογή αυτή εκτελείται από το τερματικό από τον φάκελο Project140 με την εντολή

```
./Project140.gambas.
```

Για να κάνουμε επίδειξη του αναλυτή παραμέτρων μετά το όνομα της εφαρμογής καλό είναι να περάσουμε κάποιες παραμέτρους γράφοντας μια εντολή της μορφής:

```
./Project140.gambas -a 1 -b 2 -c 3
```

όπου -a,1,-b,2-c,3 είναι οι διοχέτευση διακοπτών με τα ορίσματα τους.

Ο κώδικας της εφαρμογής είναι ο παρακάτω:

```
1   ' Gambas module file
2
3   Public Sub Main()
4
5     Dim sElement As String
6
7     For Each sElement In Args
8       Print sElement
9     Next
10  End
```

Ανάλυση:
Στην γραμμή 3 ξεκινά η ρουτίνα Main().
Στην γραμμή 5 δηλώνουμε μια μεταβλητή τύπου String.
Στην γραμμή 7-9 έχουμε ένα μπλοκ κώδικα FOR EACH...NEXT
Στο μπλοκ αυτό απαριθμούμε κάθε ένα στοιχείο της στατικής κλάσης Args.
Κάθε ένα στοιχείο της Args εκχωρείτε στην μεταβλητή sElement, την οποία τυπώνουμε στην έξοδο.

Τρέχοντας την εφαρμογή με την εντολή

```
./Project140.gambas -a 1 -b 2 -c 3
```

Θα έχουμε στην έξοδο

```
./Project140.gambas
-a
1
-b
2
-c
3
```

Βλέπουμε ότι η πρώτη παράμετρος είναι το ίδιο το όνομα της εφαρμογής.

Κεφάλαιο 3ο – Προχωρημένες τεχνικές GUI (Graphical User Interface)

gb.form.dialog

Το εξάρτημα αυτό υλοποιεί εκ νέου τα τυποποιημένα πλαίσια διαλόγου βασισμένο στα controls που έχουν οριστεί στα εξαρτήματα gb.form component.
Η Gambas προσφέρει συγκεκριμένα πλαίσια διαλόγου για την επιλογή χρώματος, γραμματοσειράς, ανοίγματος αρχείου, σώσιμο αρχείου, επιλογή ημερομηνίας και επιλογή φακέλου.

Τα πλαίσια διαλόγου είναι ανεξάρτητα παράθυρα τα οποία προσφέρουν έτοιμα στοιχεία ελέγχου πλήρως λειτουργικά για την διεκπεραίωση απλών τυποποιημένων ενεργειών όπως αυτές που αναφέρθηκαν προηγουμένως.

Κλάσεις
Περιέχει τις εξής κλάσεις

Dialog

Κλάση Dialog

Η κλάση περιέχει στατικές μεθόδους που χρησιμοποιούνται για να κληθούν συγκεκριμένα πλαίσια διαλόγου της Gambas.
Η κλάση υλοποιεί ξανά την κλάση Dialog από το εξάρτημα gb.qt4.
Η κλάση είναι στατική και δεν μπορεί να δημιουργηθεί.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Dialog		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Color Filter Font Path Title AutoExt Date Paths	SelectColor SelectFont OpenFile SaveFile SelectDate SelectDirectory	-
Σταθερές (Constants)		
-		

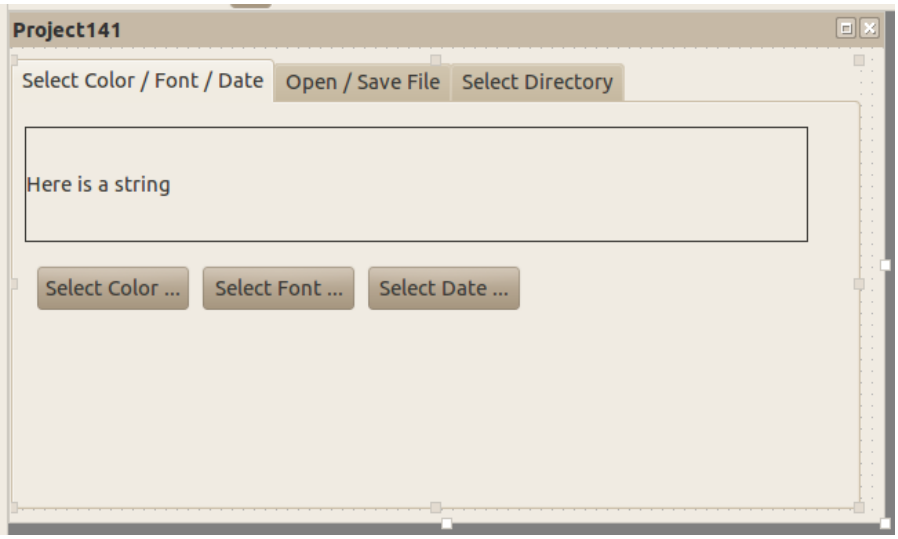
Εφαρμογή1
Για να χρησιμοποιήσουμε το εξάρτημα gb.form.dialog στην εφαρμογή μας πρέπει να το επιλέξουμε στην λίστα των components από το μενού Project-> Properties -> Components.

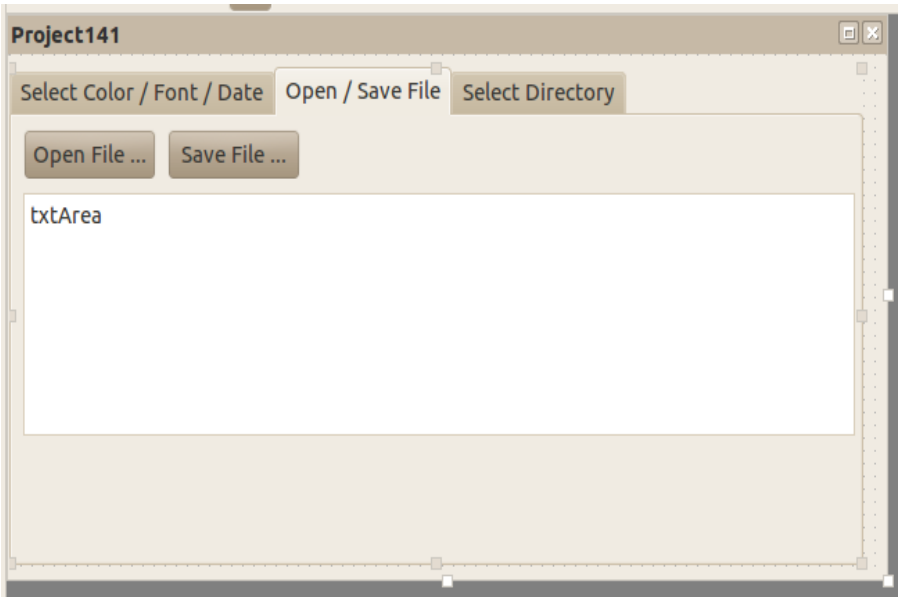
Η εφαρμογή είναι το Project141.
Η εφαρμογή έχει ένα control καρτελών Tabstrip με 3 καρτέλες. Στην πρώτη γίνεται επίδειξη των μεθόδων SelectColor, SelectFont, SelectDate. Ο χρήστης επιλέγει μια ημερομηνία που εμφανίζεται σε ένα label και στην συνέχεια επιλέγει γραμματοσειρά και χρώμα για το κείμενο στο label.

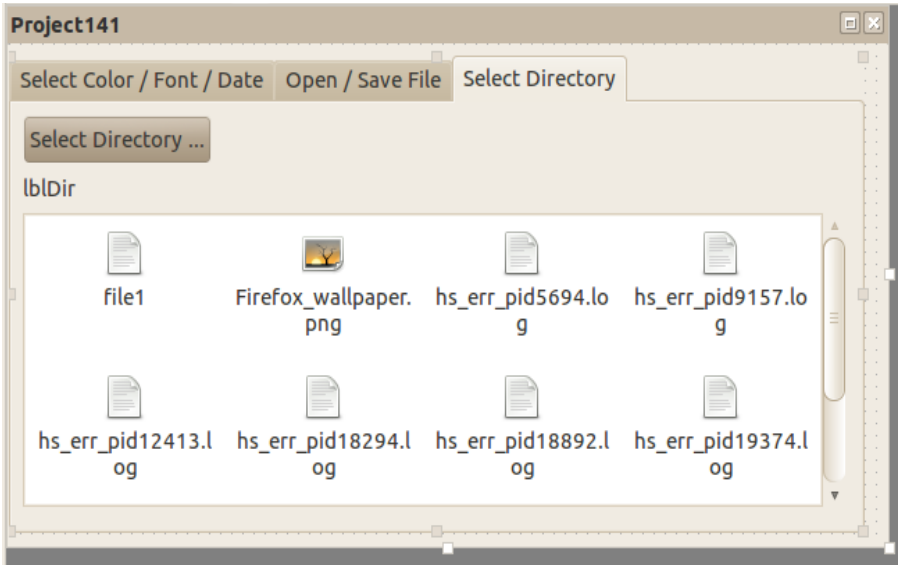
Στην δεύτερη καρτέλα γίνεται επίδειξη των μεθόδων OpenFile, SaveFile. Έχουμε δημιουργήσει ένα αρχείο test.txt στον κατάλογο της εφαρμογής με ένα κείμενο εντός του αρχείου. Στην καρτέλα αυτή εμφανίζουμε ένα πλαίσιο διαλόγου επιλογής του αρχείου και το ανοίγουμε εμφανίζοντας τα περιεχόμενα του σε ένα TextArea control.
Κατόπιν μπορούμε να αποθηκεύσουμε τις οποιεσδήποτε αλλαγές πάλι σε ένα αρχείο εμφανίζοντας ένα πλαίσιο διαλόγου Save As.

Στην τρίτη καρτέλα έχουμε ένα FileView που εμφανίζει τα περιεχόμενα ενός φακέλου.
Εξ' ορισμού εμφανίζει τα περιεχόμενα του home φακέλου του χρήστη.
Όμως με την επιλογή κάποιου φακέλου με το πλαίσιο διαλόγου SelectDirectory εμφανίζουμε περιεχόμενα από οποιοδήποτε φάκελο.

Η γραφική διεπαφή της εφαρμογής είναι η εξής:







Ο κώδικας της εφαρμογής είναι ο παρακάτω:

```
1  ' Gambas class file
2
3  Public Sub Form_Open()
4
5    Me.Center
6    fvwFileView.Dir = User.Home
7    lblDir.Text = fvwFileView.Dir
8
9  End
```

```
10
11
12  Public Sub btnColor_Click()
13
14    Dialog.Color = Color.Black
15    If Dialog.SelectColor() Then Return
16    Label1.Foreground = Dialog.Color
17
18  End
19
20  Public Sub btnFont_Click()
21
22    Dialog.Font = Label1.Font
23    If Dialog.SelectFont() Then Return
24    Label1.Font = Dialog.Font
25
26  End
27
28  Public Sub btnOpenFile_Click()
29
30    Dialog.Filter = ["*.txt", "Text Files"]
31    Dialog.Path = Application.Path
32    If Dialog.OpenFile() Then Return
33    txtArea.Text = File.Load(Dialog.Path)
34
35  Catch
36    Message.Info(Error.Text)
37
38  End
39
40  Public Sub btnSaveFile_Click()
41
42    Dialog.Filter = ["*.txt", "Text Files"]
43    Dialog.Path = Application.Path
44    If Dialog.SaveFile() Then Return
45    File.Save(Dialog.Path, txtArea.Text)
46
47  Catch
48    Message.Info(Error.Text)
49
50  End
51
52  Public Sub btnSelectDate_Click()
53
54    If Dialog.SelectDate() Then Return
```

```
55    Label1.Text = Dialog.Date
56
57  Catch
58    Message.Info(Error.Text)
59
60  End
61
62  Public Sub btnSelectDirectory_Click()
63
64    Dialog.Path = Application.Path
65    If Dialog.SelectDirectory() Then Return
66    fvwFileView.Dir = Dialog.Path
67    lblDir.Text = fvwFileView.Dir
68
69  End
```

Ανάλυση:
Στην γραμμή 3 ξεκινά η ρουτίνα `Form_Open()`. Στην ρουτίνα αυτή στην γραμμή 5 τοποθετούμε την φόρμα στο κέντρο της οθόνης. Στην γραμμή 6 αρχικοποιούμε το FileView control εκχωρώντας του σαν διαδρομή φακέλου τον φάκελο home του χρήστη.
Στην γραμμή 7 ενημερώνουμε και το κείμενο της ετικέτας lblDir που βρίσκεται στο tab Select Directory. Οι γραμμές 6,7 απλά αρχικοποιούν τα controls του tab Select Directory.

Στην γραμμή 12 ξεκινά η ρουτίνα `btnColor_Click()` του κουμπιού btnColor από το tab Select Font / Color / Date.
Στην γραμμή 14 αρχικοποιούμε το πλαίσιο διαλόγου ορίζοντας ένα προκαθορισμένο χρώμα.
Στην γραμμή 15 εμφανίζουμε το πλαίσιο διαλόγου και στην περίπτωση που πατήσει ο χρήστης Cancel επιστρέφεται η τιμή TRUE και τότε εξερχόμαστε από την ρουτίνα με την εντολή RETURN. Στην περίπτωση που πατήσει OK το πλαίσιο διαλόγου επιστρέφει FALSE και συνεχίζει η εκτέλεση του κώδικα στην γραμμή 16.
Στην γραμμή 16 ορίζουμε το χρώμα Label1.Foreground στην τιμή που έχουμε επιλέξει από το πλαίσιο διαλόγου χρώματος.

Στην γραμμή 20 ξεκινά η ρουτίνα `btnFont_Click()` και ο αλγόριθμος είναι παρόμοιος με τον προηγούμενο. Απλά εδώ ορίζουμε την τιμή Label1.Font από την επιλεγμένη τιμή του πλαίσου διαλόγου επιλογής γραμματοσειράς.

Παρόμοιος είναι και ο κώδικας στην γραμμή 52 στην ρουτίνα `btnSelectDate_Click()`
Στην γραμμή 54 εμφανίζουμε το πλαίσιο διαλόγου επιλογής ημερομηνίας και ελέγχουμε την επιστροφή της τιμής του. Αν η τιμή είναι TRUE εξερχόμαστε της ρουτίνας, αλλιώς συνεχίζει η εκτέλεση του κώδικα στην επόμενη γραμμή 55 όπου εκχωρούμε την ημερομηνία στην ιδιότητα της ετικέτας Label1.Text

Έτσι από το πρώτο tab μπορούμε να επιλέξουμε ημερομηνία και να την εμφανίσουμε σε ένα label και να αλλάξουμε την γραμματοσειρά και το χρώμα της ετικέτας.

Στο δεύτερο tab έχουμε μια περιοχή κειμένου στην οποία φορτώνουμε το κείμενο από ένα αρχείο txt. Για να ανοίξουμε το αρχείο χρησιμοποιούμε το πλαίσιο διαλόγου OpenFile και την μέθοδο File.Load().

Στην γραμμή 28 ξεκινά η ρουτίνα `btnOpenFile_Click()`
Στις γραμμές 30, 31 αρχικοποιούμε το πλαίσιο διαλόγου. Στην γραμμή 30 ορίζουμε τον τύπο αρχείων που θα εμφανίζει το πλαίσιο διαλόγου και στην γραμμή 31 ορίζουμε σαν διαδρομή ανοίγματος την διαδρομή στην οποία βρίσκεται η εφαρμογή. Στην διαδρομή αυτή βρίσκεται ένα αρχείο txt που έχουμε δημιουργήσει με έναν editor.
Στην γραμμή 32 εμφανίζουμε το πλαίσιο διαλόγου ανοίγματος αρχείου.
Στην γραμμή 33 διοχετεύουμε στην μέθοδο File.Load() την διαδρομή του αρχείου που επιλέξαμε από το πλαίσιο διαλόγου. Έτσι το αρχείο ανοίγει και τα περιεχόμενα του τα εκχωρούμε στην ιδιότητα txtArea.Text της περιοχής κειμένου.

Στην περίπτωση που εμφανιστεί κάποιο λάθος στην γραμμή 35 υπάρχει ο διαχειριστής λαθών CATCH που εμφανίζει το μήνυμα λάθους στην γραμμή 36.

Παρόμοιο αλγόριθμο χρησιμοποιούμε και για να σώσουμε ένα αρχείο στις γραμμές 40 με 50. Απλά εδώ διοχετεύουμε την τιμή που επιστρέφει το πλαίσιο διαλόγου στην μέθοδο File.Save().

Στο τρίτο tab εμφανίζουμε το πλαίσιο διαλόγου επιλογής φακέλου και εκωρούμε την επιστρεφόμενη τιμή στην ιδιότητα Dir ενός FileView control.
Στην γραμμή 62 ξεκινά η ρουτίνα `btnSelectDirectory_Click()`
Στην γραμή 64 αρχικοποιούμε το πλαίσιο διαλόγου ορίζοντας την διαδρομή που θα δείχνει όταν θα ανοίξει.
Στην γραμμή 65 εμφανίζουμε το πλαίσιο διαλόγου.
Στην γραμμή 66 εκχωρούμε στην ιδιότητα Dir του fvwFileView την τιμή που επέστρεψε το πλαίσιο διαλόγου. Έτσι στο FileView control εμφανίζονται τα αρχεία του φακέλου που επιλέξαμε.
Τέλος ενημερώνουμε το κείμενο του lblDir για να εμφανίζουμε την διαδρομή που δείχνει το fvwFileView.

gb.form.mdi

Αυτό το εξάρτημα επιτρέπει την δημιουργία διεπαφής πολλαπλών εγγράφων με το στοιχείο ελέγχου Workspace και παρέχει και ένα δοχέα ToolBar.

Κλάσεις
Περιέχει τις εξής κλάσεις

Action ToolBar Workspace

Κλάση Workspace

Το control αυτό υλοποιεί μια περιοχή εργασία ως ένα TabStrip όπου μπορείτε να εμφανίζετε άλλα παράθυρα – φόρμες.

Για να προσθέσετε ένα νέο παράθυρο στην περιοχή εργασία workspace, πρέπει πρώτα να το αρχικοποιήσετε και στην συνέχεια να χρησιμοποιήσετε την μέθοδο Add για να το παρουσιάσετε στην περιοχή εργασίας.

Ο χρήστης μπορεί να εναλλάσει τα παράθυρα – παιδιά χρησιμοποιώντας τις καρτέλες Tabs του TabStrip.

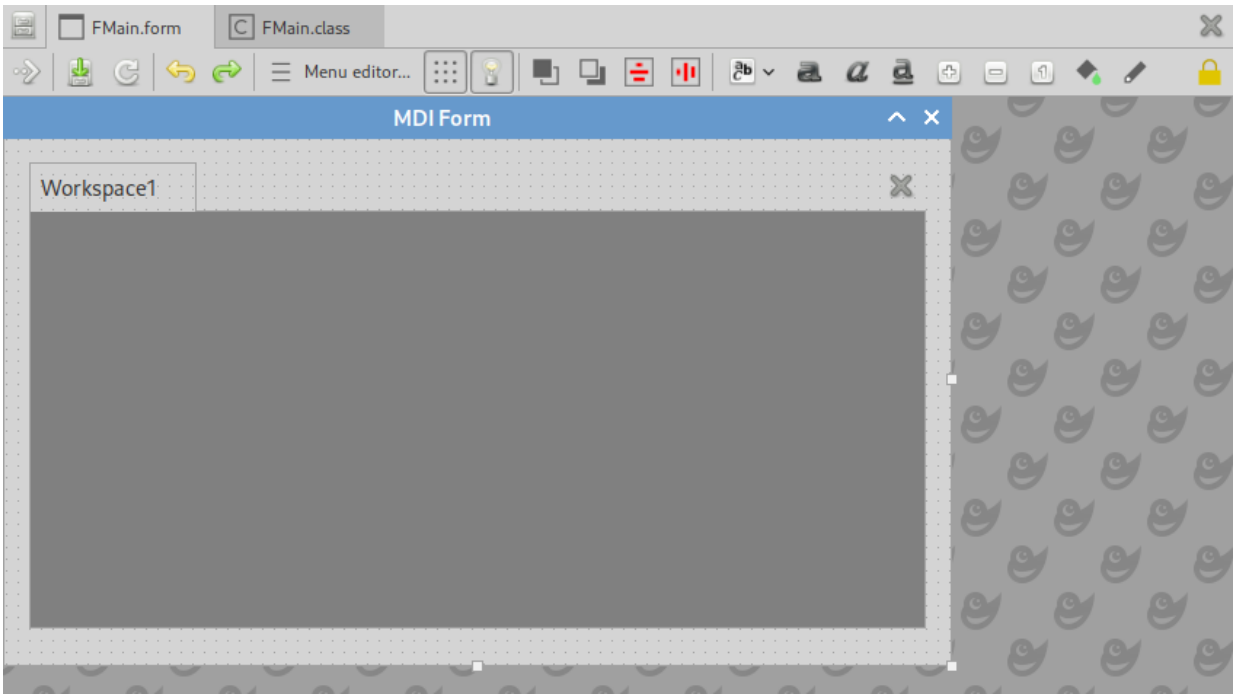
Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης Toolbar		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action ActiveWindow Background Border Children ClientH ClientHeight ClientW ClientWidth ClientX ClientY CloseButton CloseWithMouse Count Cursor Design Direction Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Image Left Mouse Name Next NoTabFocus Orientation Parent Picture PopupMenu Previous Proxy RightToLeft ScreenX ScreenY TabPanel Tag Tooltip Top Tracking Visible W Width Window Windows X Y	Add Attach CloseAll Delete Detach Drag Exist FindChild Grab Hide Lock Lower Move MoveScaled Raise Refresh Remove Reparent Resize ResizeScaled SetFocus Show Unlock	Activate Arrange Attach BeforeArrange Close CloseAll DblClick Detach Drag DragLeave DragMove Drop Enter GotFocus KeyPress KeyRelease Leave LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel NewChild

Εφαρμογή1
Για να χρησιμοποιήσουμε το εξάρτημα gb.form.mdi στην εφαρμογή μας πρέπει να το

επιλέξουμε στην λίστα των components από το μενού Project-> Properties -> Components.

Το Project142 αποτελείται από τρεις φόρμες. Την φόρμα εκκίνησης FMain, Form1, Form2.
Η FMain έχει ένα αντικείμενο Workspace1 σχεδιασμένο πάνω της.
Το αντικείμενο Workspace βρίσκεται στην καρτέλα Special της εργαλειοθήκης.
Οι φόρμες Form1 και Form2 έχουν από ένα Label με κείμενο το όνομα της φόρμας.

Η γραφική διεπαφή της εφαρμογής είναι η εξής:



Ο κώδικας της εφαρμογής είναι ο παρακάτω:

```
1  ' Gambas class file
2
3  Public Sub Form_Open()
4
5    Workspace1.Add(Form1)
6    Workspace1.Add(Form2)
7
8  End
```

Ανάλυση:
Στην γραμμή 3 ξεκινά η ρουτίνα `Form_Open()`. Στην ρουτίνα αυτή στις γραμμές 5,6 προσθέτουμε στο αντικείμενο Workspace1 τις φόρμες Form1 και Form2.

Η εφαρμογή κατά τον χρόνο εκτέλεσης είναι η παρακάτω:

gb.form.terminal

Αυτό το εξάρτημα παρέχει ένα πλήρης χαρακτηριστικών στοιχείο ελέγχου τερματικού εξομοιωτή VT-100.
Ένα τερματικό κάνει τα ακόλουθα:
Λαμβάνει χαρακτήρες από το πληκτρολόγιο και τους στέλνει στη διαδικασία που εκτελείται μέσα σε αυτό.
Λαμβάνει χαρακτήρες από την έξοδο διεργασίας που εκτελείται και τους εκτυπώνει στην οθόνη του.
Ορισμένες ακολουθίες ειδικών χαρακτήρων δεν εκτυπώνουν τίποτα αλλά ενεργοποιούν συγκεκριμένες ενέργειες. Ονομάζονται «χαρακτήρες ελέγχου» και «ακολουθίες διαφυγής ελέγχου».

Για περισσότερες πληροφορίες επισκεφθείτε την βοήθεια της Gambas3
https://gambaswiki.org/wiki/comp/gb.form.terminal

Κλάσεις
Περιέχει τις εξής κλάσεις

TerminalAttr TerminalLink TerminalView

Κλάση TerminalView

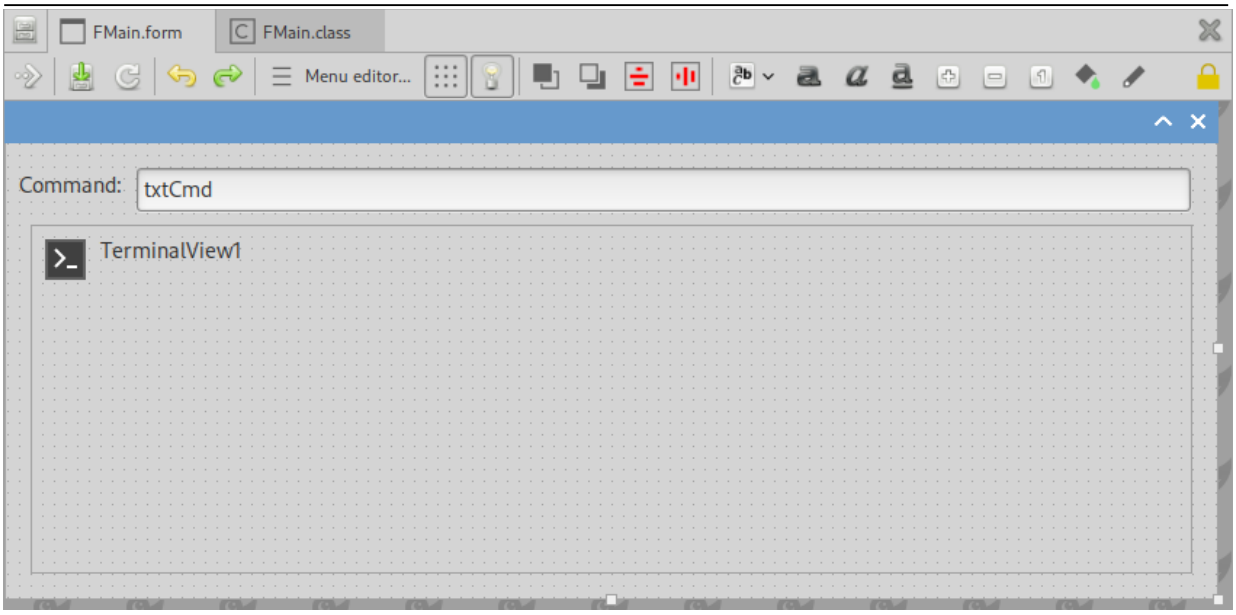
Αυτή η κλάση παρέχει ένα πλήρης χαρακτηριστικών στοιχείο ελέγχου τερματικού εξομοιωτή VT-100.

Ιδιότητες, Μέθοδοι, Συμβάντα της κλάσης TerminalView		
Ιδιότητες (Properties)	**Μέθοδοι (Methods)**	**Συμβάντα (Events)**
Action AlternateScreen Attr AutoResize Background Blink Border CharWidth Children ClientH ClientHeight ClientW ClientWidth ClientX ClientY Column Count Cursor CursorStyle Design Direction Drop Enabled Expand Font Foreground H Handle HasFocus Height Hovered Id Ignore Left Limit Line LineHeight Link Mouse Name Next NoTabFocus Parent PopupMenu Previous Proxy ReadOnly RightToLeft ScreenWidth ScreenX ScreenY ShowLink ShowScrollBar Suspended Tag Text Title Tooltip Top Tracking Type Visible W Width Window WindowCommands X Y	Clear ConnectTo Copy Delete Drag EnsureVisible Exec FindChild Goto Grab Hide Input Kill Lower Move MoveScaled Paste Print Raise Refresh Reparent Reset Resize ResizeScaled SetFocus Shell Show	Arrange BeforeArrange Bell Click DblClick Drag DragLeave DragMove Drop Enter GotFocus KeyPress KeyRelease Kill Leave Link LostFocus Menu MouseDown MouseDrag MouseMove MouseUp MouseWheel NewChild Resize Title
Σταθερές (Constants)	Block Underline VerticalLine	

Εφαρμογή1
Για να χρησιμοποιήσουμε το εξάρτημα gb.form.terminal στην εφαρμογή μας πρέπει να το επιλέξουμε στην λίστα των components από το μενού Project-> Properties -> Components.

Το Project143 δείχνει την λειτουργία της κλάσης TerminalView.
Έχουμε μια φόρμα την FMain με ένα Label με την ιδιότητα Text στο κείμενο “Command”, ένα TextBox με το όνομα txtCmd και το στοιχείο ελέγχου TerminalView1.
Το στοιχείο ελέγχου TerminalView1 μπορείτε να το βρείτε στην καρτέλα Προβολή (View) της εργαλειοθήκης.

Η διεπαφή της εφαρμογής είναι η εξής:

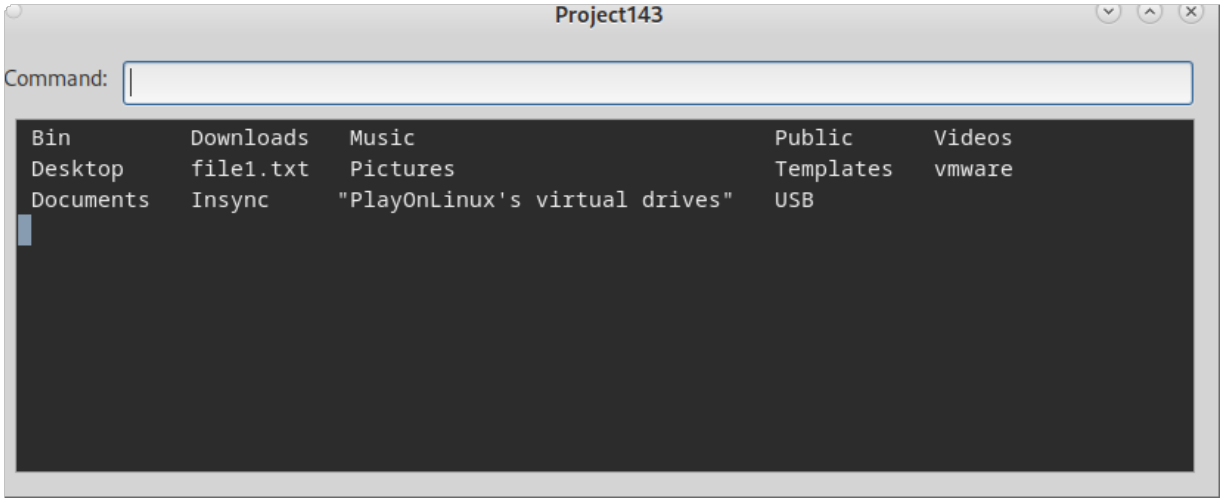


Ο κώδικας της εφαρμογής είναι ο παρακάτω:

```
1   ' Gambas class file
2
3   Public Sub txtCmd_KeyPress()
4
5     'if user press ENTER
6     If Key.Code = 65293 Then
7       TerminalView1.Shell(txtCmd.Text)
8       txtCmd.Text = ""
9     Endif
10
11  End
```

Ανάλυση:
Στην γραμμή 3 ξεκινά η ρουτίνα `txtCmd_KeyPress()`. Στην ρουτίνα αυτή στην γραμμή 6 ελέγχουμε αν ο χρήστης πάτησε το κουμπί ENTER ενώ ήταν στο πλαίσιο κειμένου txtCmd. Αν το έχει πατήσει η γραμμή 7 εκτελεί σε κέλυφος το κείμενο του πλαισίου κειμένου txtCmd και στην συνέχεια στην γραμμή 8 καθαρίζει το κείμενο του πλαισίου κειμένου για να γράψει ο χρήστης μια νέα εντολή.

Η εφαρμογή κατά τον χρόνο εκτέλεσης είναι η παρακάτω:
Εδώ ο χρήστης έχει πατήσει την εντολή “ls”.


Project143
Command:
Bin Downloads Music Public Videos
Desktop file1.txt Pictures Templates vmware
Documents Insync "PlayOnLinux's virtual drives" USB

Κεφάλαιο 4ο – Βάσεις Δεδομένων (Databases)

gb.db

Αυτό το εξάρτημα σάς επιτρέπει να έχετε πρόσβαση σε πολλά συστήματα διαχείρισης βάσεων δεδομένων. Αυτό το εξάρτημα δημιουργεί ένα ενδιάμεσο επίπεδο μεταξύ του ενός διακομιστή βάσεων δεδομένων και του προγράμματός σας, έτσι ώστε να μπορείτε να χρησιμοποιήσετε ακριβώς τον ίδιο κώδικα ό,τι βάση δεδομένων αποφασίσετε να χρησιμοποιήσετε.

Το εξάρτημα αυτό παρέχει τις εξής κλάσεις:

Κλάση	**Περιγραφή**
Blob	Αυτό το αντικείμενο επιστρέφεται από ένα πεδίο Blob και χρησιμοποιείται για την ανάγνωση ή την εγγραφή των περιεχομένων του.
Connection	Αυτή η κλάση αντιπροσωπεύει μια σύνδεση με μια βάση δεδομένων.
Connections	Αυτή η κλάση απαριθμεί όλες τις συνδέσεις που ορίζονται στο IDE στον φάκελο έργου "Συνδέσεις", (Connections).
Database	Αυτή η κλάση αντιπροσωπεύει μια βάση δεδομένων.
DatabaseUser	Αυτή η κλάση αντιπροσωπεύει έναν χρήστη μιας βάσης δεδομένων.
DB	Αυτή η κλάση αντιπροσωπεύει την τρέχουσα σύνδεση.
Field	Αυτή η κλάση αντιπροσωπεύει ένα πεδίο πίνακα.
Index	Αυτή η κλάση αντιπροσωπεύει ένα ευρετήριο πίνακα.
Result	Αυτή η κλάση αντιπροσωπεύει το αποτέλεσμα ενός αιτήματος SQL.
ResultField	Αυτή η κλάση αντιπροσωπεύει ένα από τα πεδία ενός αντικειμένου Result.
SQLRequest	Αυτή η κλάση παρέχει τη δυνατότητα δημιουργίας ενός βασικού αιτήματος SQL σε έναν μόνο πίνακα (table).
Table	Αυτή η κλάση αντιπροσωπεύει τον ορισμό ενός πίνακα βάσης δεδομένων.

Αυτό το στοιχείο μπορεί να έχει πρόσβαση στα ακόλουθα συστήματα διαχείρισης βάσης δεδομένων:
•PostgreSQL
•MySQL
•SQLite2
•SQLite3
•ODBC

Τα συστήματα διαχείρισης βάσης δεδομένων PostgreSQL και MySQL είναι πελάτη/διακομιστή (client/server databases), που σημαίνει ότι η σύνδεση γίνεται σε μια διεργασία του διακομιστή.

Από την άλλη, η SQLite είναι μια βάση δεδομένων αρχείου ή μνήμης, έτσι δεν υπάρχει

διεργασία διακομιστή να συνδεθείτε. Αυτό σημαίνει ότι ο χρήστης μπορεί να χρειαστεί να συμπεριλάβει μια πλήρως διαδρομή προς το αρχείο της βάσης δεδομένων εάν η προεπιλογή δεν είναι ικανοποιητική. Δείτε την κλάση Connection για περισσότερες πληροφορίες.

Το πρόγραμμα οδήγησης SQLite υποστηρίζει την έκδοση 3 του SQLite και επιστρέφει στο πρόγραμμα οδήγησης έκδοσης 2 εάν χρειάζεται.

Δείτε το www.sqlite.org για περισσότερες πληροφορίες.

Πώς να ανοίξετε μια σύνδεση SQLite και να τη χρησιμοποιήσετε

Συνολική διαδικασία

Δημιουργείται ένα αντικείμενο τύπου Connection, που εκχωρεί τις απαραίτητες ιδιότητες (τύπος βάσης δεδομένων και όνομα, μόνο).

Στην συνέχεια καλείται η μέθοδος Open για να ανοίξει η βάση δεδομένων.
Όταν καλείτε Άνοιγμα (Open), η βάση δεδομένων είναι ενεργή και έτοιμη για αναζήτηση.

Το ερώτημα επικοινωνίας μπορεί να γίνει χρησιμοποιώντας τη μέθοδο Exec.

Με τη σύνδεση αντικειμένου ανοιχτή και ενεργή, εκτελούνται ερωτήματα SQL στη βάση δεδομένων.

Το Αποτέλεσμα (Result) αναπαρίσταται ως αποτέλεσμα αντικειμένου του οποίου οι ιδιότητες εξαρτώνται από τον τύπο του τύπου χειριστή βάσης δεδομένων, όπως οι επηρεαζόμενες ή επιστρεφόμενες σειρές ανάλογα με τον τύπο του ερωτήματος SQL που εκτελείται.

Το αντικείμενο Αποτέλεσμα (Result) είναι αυτό στο οποίο γίνεται τελικά πρόσβαση για τη διαχείριση δεδομένων από τη βάση δεδομένων και είναι επαναλαμβανόμενο, σε κάθε επανάληψη έχει έναν πίνακα, αλλά σαν κάθε στοιχείο έχει έναν δικό του τύπο προδιαγραφών, που εξαρτώνται από το εάν υποβλήθηκε ερώτημα ή χειρισμός.

Απαιτήσεις

1. Θα πρέπει να βεβαιωθείτε ότι έχετε εγκατεστημένα όλα τα εξαρτήματα για τις βάσεις δεδομένων όπως το **gb.db** και όλα τα πακέτα που επισημαίνονται με **Sqlite**.
2. Η πρόσβαση στη βάση δεδομένων και τα δικαιώματα εξαρτώνται από το δικαίωμα αρχείου και τη διαδρομή βάσης δεδομένων sqlite.
3. Το στοιχείο **gb.db** έχει εγκατασταθεί και μεταγλωττιστεί με υποστήριξη για **gb.db.sqlite3** και έχει επιλεγεί για το έργο.

Εφαρμογή1
Το Project144 δείχνει την σύνδεση σε μια βάση SQLite3, την διεκπαιρέωση ενός ερωτήματος SQL, και την παρουσίαση των αποτελεσμάτων στην stdout αλλά και στην συμπλήρωση των στοιχείων ενός ComboBox.

Η γραφική διεπαφή του έργου αποτελείται από μια φόρμα FMain, δύο κουμπιά btnConnectDb και btnGetResult και από ένα ComboBox με το όνομα cmbResult.

Το πρώτο κουμπί, btnConnectDb, ανοίγει μια σύνδεση με το αρχείο βάσης δεδομένων test.sqlite3 που βρίσκεται στον φάκελο του έργου.

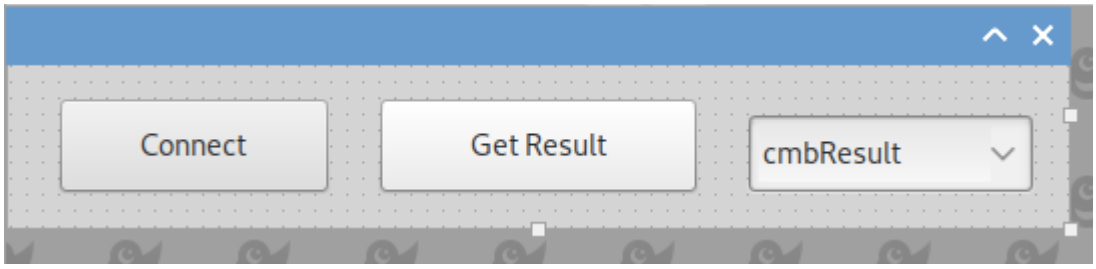
Το δεύτερο κουμπί, btnGetResult, εκτελεί ένα SQL ερώτημα στην βάση. Συγκεκριμένα επιλέγει από τον πίνακα names της βάσης, όλες τις εγγραφές με το SQL ερώτημα, "SELECT * FROM names ".

Κάθε μία εγγραφή αποτελείται από δύο κολώνες την id και την name.

Ο πίνακας names της βάσης περιέχει τέσσερα ονόματα και τέσσερεις μοναδιαίους ακέραιους αριθμούς για το id κάθε ονόματος, όπως φαίνεται παρακάτω.

Id	**Name**
1	Dim
2	Nick
3	Maria
4	Vivi

Η γραφική διεπαφή φαίνεται παρακάτω:



Αρχικά ορίζουμε ως δημόσιες καθολικές μεταβλητές, τις $con ως σύνδεση της βάσης και την rs ως Αποτέλεσμα ενός SQL ερωτήματος

Στην συνέχεια γράφουμε τον κώδικα για την σύνδεση με την βάση δεδομένων που εκτελεί το κουμπί btnConnectDb.

Ο κώδικας φαίνεται στην ρουτίνα `Public Sub btnConnectDb_Click()`.

```
1   ' Gambas class file
2
3   Public $con As New Connection
4   Public rs As Result
5
6   Public Sub btnConnectDb_Click()
7
8     Try $con.Close()
9     $con.Type = "sqlite3"
10    $con.Host = Application.path
11    $con.Name = "test.sqlite3"
12    $con.Open()
13
14    If $con.Opened Then
15      Message("Connected to database successfully !")
16    Else
17      Message("Could not connect to database !")
18    Endif
19
20  End
```

Ανάλυση:
Στην γραμμή 3 δηλώνουμε ως Δημόσια και καθολική την μεταβλητή $con ως μια σύνδεση βάσης δεδομένων.
Στην γραμμή 4 δηλώνουμε ως Δημόσια και καθολική την μεταβλητή rs ως ένα αποτέλεσμα ερωτήματος σε βάση δεδομένων.
Δηλώνουμε τις μεταβλητές αυτές ως καθολικές για να τις προσπελάσουμε από όλες τις ρουτίνες αντικειμένων της εφαρμογής.

Στην γραμμή 6, ξεκινά η ρουτίνα του κουμπιού `btnConnectDb` για το συμβάν `Click()`.

Στην γραμμή 8 προσπαθούμε να κλείσουμε την σύνδεση με την βάση εάν για παράδειγμα ήταν ανοιχτή από προηγούμενη χρήση. Αυτός είναι ένας γρήγορος τρόπος να κάνουμε reset την σύνδεση.

Στις γραμμές 9-11 ορίζουμε τις ιδιότητες της σύνδεσης για μια σύνδεση με βάση δεδομένων τύπου SQLite3.

Τέλος στην γραμμή 12 προσπαθούμε να ανοίξουμε την σύνδεση με την βάση.
Στις γραμμές 14-18 κάνουμε έναν έλεγχο αν η σύνδεση έχει επιτευχθεί και εάν είναι έτσι εμφανίζουμε ένα μήνυμα στον χρήστη.

Στην συνέχεια ο χρήστης θα πατήσει το κουμπί btnGetResult για να κάνει ένα SQL ερώτημα στην βάση και να πάρει τα αποτελέσματα.
Τα αποτελέσματα του ερωτήματος θα τα εμφανίσουμε στην προκαθορισμένη έξοδο που είναι η κοσνόλα και θα συμπληρώσουμε για κάθε εγγραφή τα στοιχεία του cmbResult, που είναι ένα ComboBox.

Ο κώδικας του συμβάντος κλικ του κουμπιού btnGetResult φαίνεται παρακάτω:

```
22 Public Sub btnGetResult_Click()
23
24   cmbResult.Clear
25
26   If $con.Opened Then
27     rs = $con.Exec("SELECT * FROM names ")
28     While (rs.Available)
29       Print "Column1:" & rs!id & ", Column2:" & rs["name"]
30       cmbResult.Add(rs!name, (rs!id - 1))
31       rs.MoveNext()
32     Wend
33   Endif
34
35   If cmbResult.Count > 0 Then
36         cmbResult.Index = 0
37   Endif
38
39 End
```

Στην γραμμή 22 ξεκινάει η ρουτίνα του συμβάντος κλικ του κουμπιού.

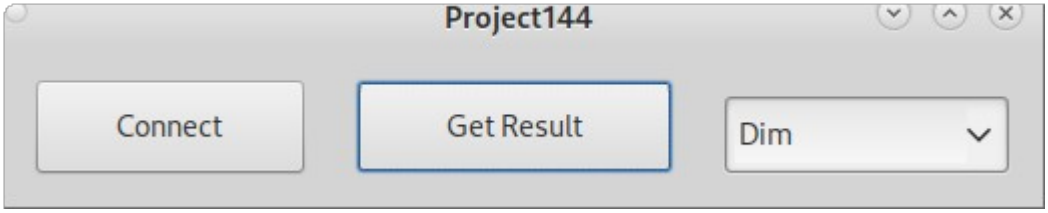
Αρχικά στην γραμμή 24 καθαρίζουμε τα πεδία του combobox cmbResult από προηγούμενη χρήση. Αυτός είναι ένας τρόπος να κάνουμε reset το combobox.

Στις γραμμές 26-33 ελέγχουμε αν η σύνδεση είναι ανοιχτή. Τότε στην γραμμή 27 εκτελούμε το SQL ερώτημα προς την βάση.
Στις γραμμές 28-32 έχουμε έναν βρόγχο While...Wend για να προσπελάσουμε κάθε εγγραφή του αποτελέσματος rs.
Έτσι όσο το αποτέλεσμα είναι διαθέσιμο, γραμμή 28, τυπώνουμε στην προκαθορισμένη έξοδο stdout, δηλαδή στην κονσόλα την πρώτη στήλη και δεύτερη στήλη του αοτελεσματος rs.
Για να κάνουμε την προσπέλαση της πρώτης στήλης που είναι το id του πίνακα γράφουμε, rs!id και για την δεύτερη rs[“name”].
Οι εκφράσεις rs!id ή rs[“id”] είναι ισοδύναμες και αποτελούν δύο τρόπους προσπέλασης των δεδομένων ενός αποτελέσματος ενός ερωτήματος SQL σε μια βάση δεδομένων.
Ομοίως ισοδύναμα είναι και οι εκφράσεις rs!name, rs[“name”].
Η έκφραση rs!id επιστρέφει τα δεδομένα της πρώτης στήλης ενώ η έκφραση rs!name τα δεδομένα της δεύτερης στήλης του πίνακα names της βάσης.

Στην γραμμή 30 προσθέτουμε κάθε εγγραφή ως στοιχείο του combobox.
Για να το κάνουμε αυτό καλούμε την μέθοδο cmbResult.Add().
Ως πρώτο όρισμα η μέθοδος cmbResult.Add(), λαμβάνει ένα String που θα αποτελέσει το κείμενο του στοιχείου του combobox.
Ως δεύτερο όρισμα η μέθοδος cmbResult.Add(), λαμβάνει έναν ακέραιο που παίζει τον ρόλο του δείκτη (index) του στοιχείου του combobox.
Εδώ η μέθοδος cmbResult.Add() είναι όμοια με έναν πίνακα που του προσθέτουμε στοιχεία.
Επειδή το πρώτο στοιχείο έχει index = 0 και το id της εγγραφής είναι 1, υπολογίζουμε το index ως (rs!id – 1).
Έτσι όλα τα ορίσματα της μεθόδου θα είναι, cmbResult.Add(rs!name, (rs!id – 1)).

Τέλος στις γραμμές 35-37, ελέγχουμε αν το combobox έχει τουλάχιστον ένα ενεργό στοιχείο, και αν έχει κάνουμε προκαθορισμένο και επιλεγμένο το πρώτο στοιχείο, στην γραμμή 36.

Η εφαρμογή κατά τον χρόνο εκτέλεσης είναι η παρακάτω:



Παρόμοια είναι και η σύνδεση και προσπέλαση δεδομένων σε βάσεις MySQL, ODBC κτλ…

Για περισσότερα δείτε τους παρακάτω συνδέσμους:
How To Open a MySQL connection and use it
How To Open a Unix ODBC connection and use it
δείτε επίσης την αντιστοιχία των τύπων των δεδομένων, Database Datatype Mapping

Κεφάλαιο 5ο – Εξαρτήματα (Components)

Η Gambas3 παρέχει μια πληθώρα εξαρτημάτων που δεν είναι δυνατόν να περιγραφούν όλα με κάθε λεπτομέρεια σε ένα βιβλίο. Εδώ θα παραθέσω τους απαραίτητους συνδέσμους προς το wiki της Gambas3 ώστε ο χρήστης να βρει περεταίρω βοήθεια για τον προγραμματισμό των εφαρμογών του.

Εξαρτήματα, μια πλήρης λίστα (Components)
Επιπρόσθετα εξαρτήματα (Additional Gambas Components)
Πώς να προγραμματίζετε εξερτήματα στην Gambas3 (How To Program Components In Gambas)
Πώς να προγραμματίζετε εξαρτήματα στην C/C++ (How To Program Components In C/C++)

Κεφάλαιο 6ο – Συμπεράσματα

Η Gambas3 είναι μια παντοδύναμη γλώσσα προγραμματισμού όπως είδαμε.
Μπορείτε να κάνετε εύκολα και γρήγορα τις δικές σας εφαρμογές, γραφικές ή τερματικού.
Επίσης μπορείτε να επεκτείνετε την λειτουργία της με τον προγραμματισμό εξαρτημάτων όπως αναφέρθηκε στο κεφάλαιο 5.
Σε αυτό το βιβλίο έγινε μια πρώτη γνωριμία με την γλώσσα και το περιβάλλον ανάπτυξης. Για κάθε βοήθεια που θέλετε για τις εφαρμογές σας μην διστάσετε να συμβουλευτείτε την βοήθεια του wiki.
Μπορείτε ακόμα να ζητήσετε βοήθεια από άλλους προγραμματιστές στην λίστα ταχυδρομείου των χρηστών Gambas, https://lists.gambas-basic.org/listinfo/user, ή να εγγραφείτε σε κάποιο forum όπως το https://forum.gambas.one/ ή http://howtogambas.org/

Για περισσότερα δείτε στο https://gambas.sourceforge.net/en/main.html#

ΤΕΛΟΣ