Πτυχιακή Εργασία

Θέμα: «Εξομοιωτής Μαθηματικών Μοντέλων»

Εισηγητής: Αθανάσιος Μαλάμος
Σπουδαστής: Αναστάσιος Γαλανόπουλος

Πίνακας Περιεχομένων

1 Εισαγωγή

1.1 Τα μαθηματικά μοντέλα

1.1.1 Ορισμός

Ο άνθρωπος σήμερα μέσα από τη συνεχή προσπάθεια κατανόησης του περιβάλλοντός του έχει καταφέρει να εξηγήσει και να αναπαραστήσει τα φυσικά φαινόμενα που συμβαίνουν γύρω του με διάφορα μαθηματικά μοντέλα. Τα μαθηματικά μοντέλα δεν περιορίζονται μόνο στην εξήγηση των φυσικών φαινομένων (π.χ. καιρικά φαινόμενα, μαγνητικά πεδία, άνωση, βαρύτητα) αλλά είναι θεμελιώδη στοιχείο όλων των εφαρμοσμένων επιστημών (όπως για παράδειγμα είναι η οικονομία, κοινωνιολογία, οι επικοινωνίες και η τεχνητή νοημοσύνη). Ένα μοντέλο είναι μια αναπαράσταση που περιέχει την ουσιώδη δομή ενός αντικειμένου η συμβάντος του πραγματικού κόσμου. Αυτή η αναπαράσταση μπορεί να γίνει με τους παρακάτω τρόπους:

- *Φυσική απεικόνιση*: Το τρισδιάστατο μοντέλο ενός αεροσκάφους ή το σχεδιάγραμμα ενός σπιτιού από έναν αρχιτέκτονα.
- *Συμβολική*: Όπως είναι για παράδειγμα ένας αλγόριθμος ή ένα σύνολο μαθηματικών συναρτήσεων.

Τα μαθηματικά μοντέλα αποτελούν μια αναπαράσταση της πραγματικότητας με τη χρήση μαθηματικών σχέσεων[1]. Γενικότερα τα μαθηματικά μοντέλα εφαρμόζονται σε κάθε διαδικασία που μπορεί να περιγραφεί από μια μαθηματική έκφραση. Καθώς τα μαθηματικά μοντέλα απαρτίζονται από μαθηματικές εκφράσεις κάποιες μεταβλητές αυτών των εκφράσεων πρέπει να είναι δεδομένες ή γνωστές. Αυτές οι *μεταβλητές* αποτελούν τις *εισόδους* των μοντέλων. Κάποια άλλα τμήματα των μαθηματικών εκφράσεων είναι άγνωστα για τα οποία ζητείται μια λύση. Αυτά είναι οι *έξοδοι* του μαθηματικού μοντέλου. Οι *παράμετροι* ενός μοντέλου είναι κάποιες *σταθερές* που επηρεάζουν τη συμπεριφορά του μοντέλου και κατά συνέπεια τις τελικές εξόδους του.

Τα μαθηματικά μοντέλα μπορούν να μετατραπούν σε έναν αλγόριθμο. Η παρούσα πτυχιακή εργασία έχει ως αντικείμενο την οπτική αναπαράσταση των μαθηματικών μοντέλων με τη χρήση αλγορίθμων. Αυτή η αναπαράσταση επιτυγχάνεται με την εισαγωγή της έννοιας του στοιχείου επεξεργασίας. Ένα στοιχείο επεξεργασίας αντιστοιχεί σε ένα μαθηματικό μοντέλο το οποίο υλοποιείται με έναν αλγόριθμο. Το στοιχείο επεξεργασίας αναπαρίσταται οπτικά στην οθόνη του υπολογιστή ως ένα κουτί με ακίδες (pins). Οι ακίδες αντιστοιχούν στις παραμέτρους και μεταβλητές του μαθηματικού μοντέλου.

1.1.2 Χαρακτηριστικά μαθηματικών μοντέλων

Όπως αναφέρεται παραπάνω τα μαθηματικά μοντέλα είναι μια συμβολική αναπαράσταση του πραγματικού κόσμου. Δεν μπορούν ωστόσο να περιλαμβάνουν όλες τις λεπτομέρειες και όλους τους παράγοντες που μπορεί να έχει το πραγματικό φαινόμενο στο οποίο αντιστοιχούν. Στην πραγματικότητα *τα μοντέλα είναι μια απλοποιημένη και κατά προσέγγιση απεικόνιση του πραγματικού κόσμου*. Όταν κάποια διαδικασία ή συμβάν ενός φαινομένου ή συστήματος δεν δημιουργεί σημαντική απόκλιση (*σφάλμα*) στις μεταβλητές του αντίστοιχου μαθηματικού μοντέλου, αυτό μπορούν να αγνοηθεί απλοποιώντας έτσι το μαθηματικό μοντέλο. Τα απλά μαθηματικά μοντέλα έχουν λίγες παραμέτρους και είναι πιο αφηρημένα σε σχέση με πολύπλοκα μαθηματικά μοντέλα. Ένας ακόμη παράγοντας που κάνει τα μαθηματικά μοντέλα, με τα οποία πραγματεύεται η παρούσα πτυχιακή εργασία, να αποκλίνουν από το πραγματικό είναι το σφάλμα που εισάγει η διακριτή φύση του υπολογιστή. Ένας αλγόριθμος που προορίζεται για έναν επεξεργαστή ψηφιακής λογικής μπορεί να αναπαραστήσει πλήρως διακριτά μαθηματικά μοντέλα. Τα συνεχή μαθηματικά μοντέλα έχουν αντίστοιχα διακριτά μαθηματικά μοντέλα όμως η μετάβαση σε αυτά εισάγει κάποιο σφάλμα ή ακόμη κάποιο περιορισμό στο εύρος τιμών και ακρίβειας των παραμέτρων και των μεταβλητών.

Ένα ακόμη χαρακτηριστικό των μαθηματικών μοντέλων είναι πως αυτά πρέπει να είναι πιο εύκολα στη χρήση σε σχέση με το σύστημα το οποίο αντιπροσωπεύουν στον πραγματικό κόσμο. Ο τελικός χρήστης πρέπει να είναι σε θέση να αλλάζει τις παραμέτρους του μοντέλου και παρατηρεί το τελικό αποτέλεσμα με μικρότερο κόστος σε χρόνο και υλικά σε σχέση με την παρατήρηση του αντίστοιχου στην πραγματικότητα.

1.1.3 Είδη Μαθηματικών μοντέλων

Τα μαθηματικά μοντέλα χωρίζονται σε δυο κατηγορίες ως προς το στάδιο υλοποίησής τους και σε δυο κατηγορίες ως προς τον τρόπο υλοποίησης τους:

Ως προς τα στάδια υλοποίησης διακρίνονται σε:

- Θεωρητικά μοντέλα (A Praori): Πρόκειται για τα μοντέλα τα οποία ήδη γνωρίζουμε και δε χρειάζονται παραπάνω πειράματα για να δημιουργηθούν καθώς αυτά έχουν καθοριστεί και επαληθευτεί από την επιστημονική κοινότητα.
- Post Praori: Πρόκειται για μαθηματικά μοντέλα τα οποία βασίζονται σε νέες παρατηρήσεις. Τα μοντέλα αυτά δεν είναι ολοκληρωμένα και χρειάζονται περισσότερα πειράματα ώστε να οριστικοποιηθούν.

Ως προς τον τρόπο υλοποίησης διακρίνονται σε:

- Περιγραφικά μοντέλα: Πρόκειται για συναρτήσεις που περιγράφουν δεδομένα χωρίς ωστόσο να τα εξηγούν. Τα μοντέλα αυτά βασίζονται περισσότερο στα δεδομένα και λιγότερο στο μαθηματικό μοντέλο που απλά τα περιγράφει. Ένα παράδειγμα ενός περιγραφικού μοντέλου είναι μια πολυωνυμική παρεμβολή βασισμένη σε δεδομένα σημεία.
- Μηχανιστικά μοντέλα: Πρόκειται για συναρτήσεις παραμέτρων συστήματος που μπορούν να εξηγήσουν τη λειτουργία ενός συστήματος. Παραδείγματα μηχανιστικών μοντέλων είναι η εξίσωση της ταχύτητας, της επιτάχυνσης, της βαρύτητας και γενικότερα των φυσικών επιστημών.

1.1.4 Δημιουργία μαθηματικών μοντέλων

Η δημιουργία των μαθηματικών μοντέλων προϋποθέτει την πλήρη κατανόηση των διαδικασιών ενός συστήματος ώστε να μπορούν αυτές να μεταφραστούν σε μαθηματικές εκφράσεις ή αλγορίθμους. Εφόσον εντοπιστεί ένα πρόβλημα η μοντελοποίηση του με μαθηματικές εκφράσεις περνά από τα ακόλουθα στάδια:

1) Προσδιορισμός: Σε αυτό το στάδιο επιλέγονται οι παράμετροι του πραγματικού κόσμου που επηρεάζουν το μοντέλο. Κατά τη φάση του προσδιορισμού το πραγματοποιείται μια απλούστευση του πραγματικού μοντέλου.

2) Αναπαράσταση / Μέτρηση: Οι επιλεγμένες παράμετροι συσχετίζονται ως αντικείμενα, συμβάντα ή σχέσεις με σύμβολα στο μαθηματικό μοντέλο δημιουργώντας τις μαθηματικές εκφράσεις του μοντέλου.

3) Μετατροπή: Ο καθορισμένος προσδιορισμός του μοντέλου μετατρέπεται σε άλλες εκφράσεις (λεκτικά) δίχως αυτές να αλλοιώνουν ή να αποκλίνουν από τον αρχικό προσδιορισμό του μοντέλου. Με αυτό τον τρόπο ανακαλύπτονται νέες συσχετίσεις που τυχόν δεν είχαν βρεθεί.

4) Επαλήθευση: Ελέγχονται εάν τα αποτελέσματα του μαθηματικού μοντέλου παράγουν είναι όμοια με αυτά στον πραγματικό κόσμο δεδομένων των παραμέτρων του μοντέλου. Δεν είναι αναγκαίο τα αποτελέσματα να είναι σε

απόλυτη συμφωνία με αυτά του πραγματικού κόσμου καθώς η απλοποίηση που έχει υποστεί το μοντέλο είναι λογικό να επηρεάζει το αποτέλεσμα. Το θέμα είναι η απόκλιση των αποτελεσμάτων να είναι εντός επιτρεπτών ορίων που καθορίζουμε οι ίδιοι.

Για να πραγματοποιήσουμε την επαλήθευση του μοντέλου θα πρέπει στο μαθηματικό μοντέλο να εισάγουμε τιμές στις διάφορες μεταβλητές του και να υπολογίσουμε τις εξόδους του. Η διαδικασία αυτή υπολογισμού τιμών θεωρείται μια απλουστευμένη μορφή εξομοίωσης.

1.2 Εξομοίωση

1.2.1 Έννοια

Σύμφωνα με τον R.E. Shannon εξομοίωση είναι η διαδικασία σχεδιασμού ενός μοντέλου πραγματικού συστήματος και η εκτέλεση πειραμάτων με αυτό το σύστημα με σκοπό την κατανόηση της συμπεριφοράς του συστήματος ή για τον προσδιορισμό διαφόρων στρατηγικών της λειτουργίας του συστήματος (εντός ορίων που επιβάλλονται από διάφορα κριτήρια). Σύστημα είναι μια συλλογή αντικειμένων ή συμβάντων που αλληλεπιδρούν μεταξύ τους με κάποιο τελικό σκοπό. Σε μια εξομοίωση τα αντικείμενα και τα συμβάντα αυτά συνήθως περιγράφονται από τα μαθηματικά μοντέλα. Εξομοιωτής είναι μια εφαρμογή που μιμείται τη συμπεριφορά ενός συστήματος του πραγματικού κόσμου και εξάγει μετρήσεις βασισμένες στη συμπεριφορά του συστήματος και τα αρχικά δεδομένα εξομοίωσης.

1.2.2 Είδη εξομοίωσης

Τα συστήματα εξομοίωσης μπορούν να χωριστούν σε

- **Στοχαστικά**: Τα μοντέλα αυτά χρησιμοποιούν γεννήτριες τυχαίων αριθμών ώστε να μοντελοποιηθεί η πιθανότητα εμφάνισης ενός συμβάντος στο σύστημα. Τα συστήματα αυτά είναι γνωστά και ως **Monte Carlo**.
- **Αιτιοκρατικά**: (Deterministic): Δεν εισάγεται ο παράγοντας της τύχης σε τέτοιου είδους συστήματα και συνεπώς η έξοδός του μπορεί να προβλεφτεί με βεβαιότητα 100%.
- **Διακριτού χρόνου**: Τα συστήματα αυτά χειρίζονται συμβάντα στο χρόνο. Σε αυτού του είδους την εξομοίωση ο εξομοιωτής διατηρεί μια ουρά γεγονότων ταξινομημένων βάσει το χρόνο τον οποίο θα πρέπει να συμβούν. Ο εξομοιωτής διαβάζει την ουρά και δημιουργεί νέα συμβάντα κατά την εκτέλεση της εξομοίωσης.
- **Συνεχή**: Τέτοια συστήματα χρησιμοποιούν διαφορικές εξισώσεις για τη μοντελοποίησή τους.
- Ένας ακόμη τύπος εξομοίωσης δίχως να υλοποιεί ένα μαθηματικό μοντέλο εξίσωσης είναι γνωστός ως **Agent Based.** Σε αυτού του είδους εξομοίωση οι ξεχωριστές οντότητες (πχ δέντρα, άτομα, πελάτες) αναπαριστώνται κατευθείαν στο σύστημα αντί να πραγματοποιείται αναπαράσταση με βάσει τη πυκνότητά τους. Οι οντότητες περιέχουν μια εσωτερική κατάσταση και ένα σύνολο κανόνων που περιγράφουν τον τρόπο με τον οποίο θα αλλάξει η κατάσταση της οντότητας από το ένα χρονικό διάστημα στο επόμενο.
- **Κατανεμημένη Εξομοίωση**: Πρόκειται για εξομοίωση που πραγματοποιείται σε ένα σύνολο υπολογιστών συνδεδεμένων μεταξύ τους συνήθως διαμέσου του Internet ή τοπικού δικτύου.

1.3 Πλεονεκτήματα εξομοιώσεων

Η πραγματοποίηση εξομοιώσεων έχει πολλά πλεονεκτήματα σε σχέση με την εκπόνηση πειράματος σε πραγματικές συνθήκες. Μερικά από αυτά είναι:

- Εκτέλεση πειράματος σε μειωμένο χρόνο
- Μειωμένες απαιτήσεις ανάλυσης

- Εύκολη αναπαράσταση και ανάλυση μοντέλων

1.3.1 Εκτέλεση πειράματος σε μειωμένο χρόνο

Το μοντέλο εξομοιώνεται σε υπολογιστή, πειραματικές εξομοιώσεις εκτελούνται κατά κανόνα σε συμπιεσμένο χρόνο. Πρόκειται για ένα σημαντικό πλεονέκτημα καθώς ορισμένες διαδικασίες απαιτούν μήνες ακόμη και έτη για την εξαγωγή συμπερασμάτων (π.χ. ρυθμός μόλυνσης λίμνης με τροποποίηση φίλτρου σε εργοστάσιο). Χρονοβόρες διαδικασίες καθιστούν δύσκολη ή αδύνατη την πραγματοποίηση ανάλυσης. Ενώ η ίδια διαδικασία σε έναν υπολογιστή μπορεί να εξομοιωθεί σε δευτερόλεπτα. Ακόμη πολλές επαναλήψεις της εξομοίωσης μπορούν να αυξήσουν την πιστότητά της.

1.3.2 Μειωμένες απαιτήσεις ανάλυσης

Πριν τη δημιουργία εξομοιώσεων με χρήση υπολογιστή οι μελετητές ήταν αναγκασμένοι να χρησιμοποιούν άλλες διαδικασίες πιο απαιτητικές ως προς την ανάλυσή τους. Ακόμη και τότε μόνο απλές διαδικασίες που αφορούσαν στοχαστικά μοντέλα ήταν δυνατό να αναλυθούν από το μέσο μελετητή. Πιο πολύπλοκα συστήματα ήταν αυστηρά θέμα των μαθηματικών ή ερευνητών επιστημόνων. Τα εξειδικευμένα λογισμικά πακέτα εξομοίωσης που έχουν δημιουργηθεί σήμερα κρύβουν από τον τελικό χρήστη τους πολύπλοκους μαθηματικούς υπολογισμούς που απαιτούνται για την ολοκλήρωση μιας εξομοίωσης δίνοντας σε αυτόν τη δυνατότητα εξομοίωσης περισσότερων τύπων συστημάτων σε σχέση με τις δυνατότητες που είχε δίχως τα εργαλεία εξομοίωσης.

1.3.3 Εύκολη αναπαράσταση και ανάλυση μοντέλων

Αρκετά λογισμικά πακέτα εξομοιώσεων διαθέτουν τη δυνατότητα δυναμικής γραφικής απεικόνισης ενός συστήματος σε λειτουργία (Αυτοκίνητο, αεροσκάφος σε λειτουργία). Η γραφική απεικόνιση επιτρέπει στο μελετητή να διαπιστώσει έγκαιρα ατέλειες του συστήματος. Δίχως τη δυνατότητα γραφικής απεικόνισης οι μελετητές θα ήταν δεσμευμένοι σε λιγότερο αποτελεσματικές αναπαραστάσεις κειμένου και αριθμών.

1.4 Μειονεκτήματα εξομοιώσεων

Οι εξομοιώσεις ωστόσο έχουν και μειονεκτήματα τα οποία πρέπει να γνωρίζει ο υποψήφιος μελετητής. Αυτά τα μειονεκτήματα δεν είναι συνδεδεμένα άμεσα με τη μοντελοποίηση και την ανάλυση του συστήματος, αλλά με τις λανθασμένες αντιλήψεις και προσδοκίες των μελετητών από τις εξομοιώσεις. Μερικά από τα μειονεκτήματα των συστημάτων εξομοιώσεων:

- Δεν είναι σε θέση να εξάγουν ακριβή αποτελέσματα όταν τα δεδομένα εισόδου είναι ανακριβή.
- Δεν είναι σε θέση να δώσουν απλή απάντηση σε πολύπλοκα προβλήματα.
- Δεν μπορούν να λύσουν προβλήματα μόνα τους

1.4.1 Ανακριβή δεδομένα εισόδου

Εάν σε ένα σύστημα εξομοίωσης εισάγουμε λανθασμένα δεδομένα, αυτό που θα εξάγουμε είναι πάλι λανθασμένα δεδομένα. Η συλλογή των δεδομένων θεωρείται από η πιο δύσκολη διαδικασία κατά τη διάρκεια μιας εξομοίωσης. Ενώ είναι γνωστό, δεν αφιερώνεται ο απαιτούμενος χρόνος στη συλλογή των αρχικών δεδομένων. Οι μελετητές έχουν τη τάση να δημιουργούν μοντέλα εξομοιώσεων αντί να φροντίζουν για τη συλλογή σωστών δεδομένων. Οι περισσότερες αποτυχημένες εξομοιώσεις βασίζονται στην ανακρίβεια των δεδομένων εισόδου.

1.4.2 Δύσκολες απαντήσεις σε δύσκολα προβλήματα

Πολλοί αναλυτές έχουν την πεποίθηση ότι ένα πολύπλοκο πρόβλημα θα καταλήξει σε μια απλή λύση κάνοντας χρήση εξομοίωσης. Ένα πολύπλοκο πρόβλημα μπορεί να απαιτεί εξομοίωση με χρήση πολλών αντικειμένων για τη μελέτη των διαφόρων παραμέτρων του συστήματος. Κατά τη διάρκεια της απλούστευσης ενός συστήματος είναι πολύ πιθανό να

πραγματοποιηθούν απλουστευμένες υποθέσεις που να έχουν μεν ως αποτέλεσμα τη γενικότερη απλούστευση του συστήματος. Οποιαδήποτε λύση εξάγει μια υπεραπλουστευμένη εξομοίωση ενδέχεται να είναι λιγότερο αποτελεσματική. Ο ενδεδειγμένος τρόπος επίλυσης δύσκολων προβλημάτων είναι το «Διαίρει και βασίλευε». Ξεχωριστή επίλυση / εξομοίωση των διαφόρων παραμέτρων ή αντικειμένων ενός πολύπλοκου συστήματος.

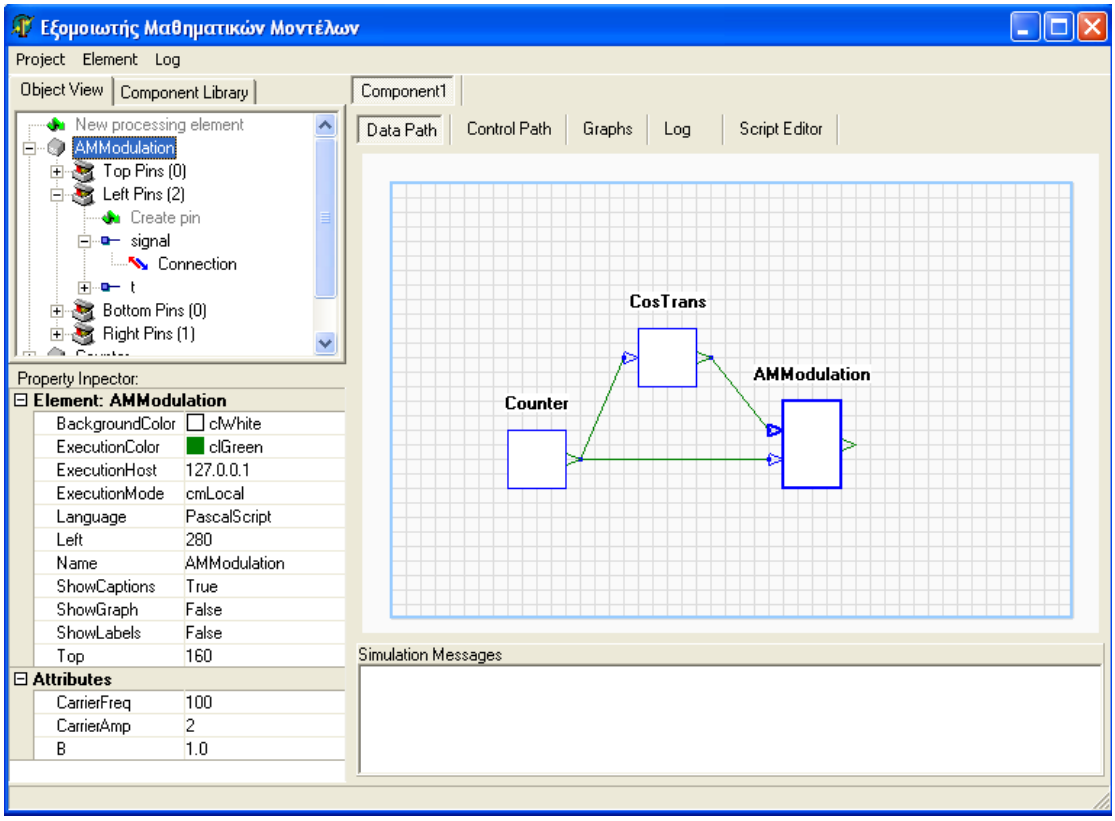
1.4.3 Η εξομοίωση από μόνη της δε λύνει προβλήματα

Μια εξομοίωση δεν λύνει ένα πρόβλημα, αλλά παρέχει πιθανές λύσεις για ένα πρόβλημα. Η υλοποίηση της προτεινόμενης λύσεις δεν αποτελεί θέμα εξομοίωσης αλλά θέμα βούλησης και κατανομής των αναγκαίων πόρων προς τη προτεινόμενη κατεύθυνση.

2 Εισαγωγή στην εφαρμογή

2.1 Συνοπτική περιγραφή

Η εφαρμογή δημιουργήθηκε με στόχο την εξομοίωση μαθηματικών μοντέλων. Υλοποιήθηκε στο περιβάλλον προγραμματισμού της Borland Delphi. Η λογική της εφαρμογής βασίζεται στην εναπόθεση και σύνδεση στοιχείων επεξεργασίας μεταξύ τους. Στοιχείο επεξεργασίας είναι μια ελάχιστη μονάδα επεξεργασίας. Σε αυτό αποθηκεύεται ένας αλγόριθμος με τον οποίον υπολογίζονται οι τιμές των εξόδων βάσει των τιμών εισόδου. Αφού σχεδιαστεί ένα στοιχείο επεξεργασίας μπορεί να αποθηκευτεί σε βιβλιοθήκη στοιχείων ώστε να μπορεί να επαναχρησιμοποιηθεί σε μελλοντικές εξομοιώσεις χωρίς τον εκ νέου σχεδιασμό και προγραμματισμό του στοιχείου. Τα στοιχεία επεξεργασίας τοποθετούνται με τέτοιον τρόπο ώστε να αλληλεπιδρούν μεταξύ τους. Υπάρχουν δυο ροές αλληλεπίδρασης, πρόκειται για την ροή δεδομένων (data path) και την ροή ελέγχου (control path).



Εικόνα 1: Το παράθυρο της εφαρμογής

2.2 Δυνατότητες

Ίσως η πιο σημαντική παράμετρος στη δημιουργία μιας εφαρμογής εξομοίωσης είναι ο τρόπος με τον οποίο θα επιτυγχάνεται η διάδραση της εφαρμογής με τον τελικό χρήστη. Αυτό μπορεί να καθοριστεί απαντώντας σε κάποια απλά ερωτήματα όπως αυτά που αναφέρονται παρακάτω:

- Ποιές είναι οι λειτουργίες που θα μπορεί να κάνει ο χρήστης στο πρόγραμμα εξομοίωσης;
- Πως θα απεικονίζεται στο χρήστη η λειτουργία του συστήματος ώστε να είναι κατανοητή;
- Τι είδους αποτελέσματα θα εξάγει η εξομοίωσή μας;

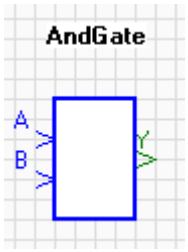
Απαντώντας συνοπτικά στα παραπάνω ερωτήματα ακολουθεί μια σύντομη παράθεση των δυνατοτήτων που έχει ο τελικός χρήστης μέσω της εφαρμογής:

- Δημιουργία, αποθήκευση και φόρτωση εργασίας.
- Δημιουργία, τροποποίηση, αποθήκευση και φόρτωση των στοιχείων επεξεργασίας ξεχωριστά.
- Αποθήκευση των στοιχείων επεξεργασίας σε μια βιβλιοθήκη. Έχοντας μια βιβλιοθήκη στοιχείων επεξεργασίας ο χρήστης μπορεί απλά να επιλέγει τα στοιχεία επεξεργασίας που επιθυμεί να χρησιμοποιήσει από τη βιβλιοθήκη.
- Παράθυρο εργασίας όπου διαμορφώνεται η ροή των δεδομένων της εξομοίωσης (Data flow).
- Δημιουργία διαγράμματος ροής ελέγχου (Control Path) των στοιχείων επεξεργασίας.
- Απεικόνιση των τοποθετημένων στο Data flow παράθυρο σε δενδρική μορφή με τις λεπτομέρειες κάθε στοιχείου (εισόδοι – εξόδοι – ενωσεις).
- Περιοχή απεικόνισης αποτελεσμάτων σε διαγράμματα (Charts).
- Παράθυρο ιδιοτήτων επιλεγμένων αντικειμένων.
- Υποστήριξη Scripting γλώσσας με την οποία γίνεται η υλοποίηση του αλγορίθμου που αντιστοιχεί στο μαθηματικό μοντέλο του στοιχείου επεξεργασίας.
- Δυνατότητα επιλογής γλώσσας scripting για υλοποίηση αλγορίθμων. Η γλώσσα υλοποίησης μπορεί να είναι PascalScript, VBScript ή Jscript
- Επεκτασιμότητα με δυνατότητα υποστήριξης COM αντικειμένων από τρίτους
- Περιβάλλον προγραμματισμού με δυνατότητα απασφαλμάτωσης (debugging) (για υλοποιήσεις σε PascalScript).
- Εκτέλεση εξομοίωσης
- Δυνατότητα κατανεμημένης εκτέλεσης εξομοίωσης.

Η βασική αρχή λειτουργίας του εξομοιωτή δίνει στον τελικό χρήστη τη δυνατότητα να τοποθετεί δομικά συστατικά (components) στην εφαρμογή. Τα components έχουν αντιστοιχούν στα «μαύρα κουτιά» όπως αυτά περιγράφηκαν πιο πάνω. Θα αντιστοιχεί σε αυτά δηλαδή ένας μαθηματικός τύπος που θα «παντρεύει» τις εισόδους με την έξοδο (ή εξόδους). Μπορούμε να κατατάξουμε τα Components της εφαρμογής σε 3 βασικές κατηγορίες:

- Στοιχεία συλλογής δεδομένων (Data Acquisition Components). Αυτά τα στοιχεία δεν έχουν connectors εισόδου
    - Γεννήτριες τυχαίων αριθμών
    - File Readers
    - Σύνδεση με βάση δεδομένων
    - Com/Serial/usb port Readers
- Στοιχεία επεξεργασίας. Αυτά τα στοιχεία περιέχουν ένα αλγόριθμο βάσει του οποίου γίνεται ο υπολογισμός στις τιμές των εξόδων δεδομένων των εισόδων του.
- Στοιχεία εξόδου
    - Γραφήματα / Στατιστικά
    - File Writers
    - Σύνδεση με βάση δεδομένων
    - Com/Serial/usb port Writers

2.3 Το στοιχείο επεξεργασίας



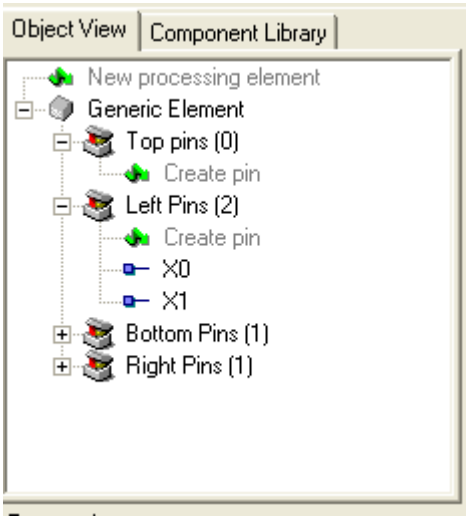
Εικόνα 2: Στοιχείο επεξεργασίας

Το στοιχείο επεξεργασίας υλοποιεί το μαθηματικό μοντέλο μέσω της scripting γλώσσας. Οι μεταβλητές του μαθηματικού μοντέλου που υλοποιεί αντιστοιχούν στις ακίδες. Οι εισερχόμενες μεταβλητές αντιστοιχούν στις ακίδες μπλε χρώματος και φαίνονται ως ένα βέλος που δείχνει προς το στοιχείο επεξεργασίας ενώ οι εξερχόμενες μεταβλητές απεικονίζονται με ένα βέλος πράσινου χρώματος και έχει φορά προς το εξωτερικό μέρος του στοιχείου επεξεργασίας. Στην διπλανή εικόνα φαίνεται ένα στοιχείο επεξεργασίας με δύο παραμέτρους και μια έξοδος που υλοποιεί τον αλγόριθμο μιας πύλης AND. Είναι δηλαδή Y ← A AND B

Ένα στοιχείο επεξεργασίας μπορεί να υλοποιεί κάτι απλό όπως για παράδειγμα μια πράξη αντιστροφής ως κάτι πολύ πιο σύνθετο. Θα μπορούσε πχ να περιγράφει τη συμπεριφορά ενός δρομολογητή, μιας οπτικής ίνας ή ενός νευρωνικού δικτύου. Για τα πιο σύνθετα components υπάρχει ένας Component Editor ενσωματωμένος στην εφαρμογή. Όταν ο χρήστης δημιουργεί ένα component ορίζει εκτός από τις εισόδους και τις εξόδους του στοιχείου το μαθηματικό μοντέλο που τα ενώνει. Υλοποιεί δηλαδή στο scripting περιβάλλον της εφαρμογής τον αλγόριθμο που περιγράφει το μαθηματικό μοντέλο. Ακόμη ορίζει κάποιες παραμέτρους (Properties) κατά το χρόνο σχεδιασμού οι οποίες θα μπορούν να αλλάξουν τη συμπεριφορά του στοιχείου κατά το χρόνο εξομοίωσης του συστήματος. Όλα τα στοιχεία που έχουν σχεδιαστεί υπάρχουν σε μια βιβλιοθήκη. Κατά το χρόνο σχεδιασμού του συστήματος (System Design Time), ο χρήστης θα έχει τη δυνατότητα να τοποθετήσει πάνω σε ένα editor οποιοδήποτε component της βιβλιοθήκης και να συνδέσει connectors εξόδου με connectors εισόδου. Επίσης είναι εφικτό να τροποποιηθούν εκ νέου κάποιες ιδιότητες των επανατοποθετημένων στοιχείων επεξεργασίας. Εφόσον όλα τα στοιχεία είναι συνδεδεμένα με σωστό τρόπο μεταξύ τους στη ροή δεδομένων αλλά και στη ροή ελέγχου μπορεί να ξεκινήσει η προσομοίωση

3 Δουλεύοντας με την εφαρμογή

3.1 Παράθυρο Object View



Εικόνα 3: Παράθυρο Object View

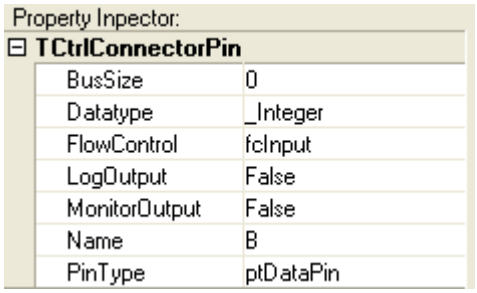
Στο παράθυρο object view απεικονίζονται σε δενδρική μορφή όλα τα στοιχεία επεξεργασίας που υπάρχουν τοποθετημένα στο παράθυρο ροής δεδομένων. Είναι ο χώρος στον οποίο πραγματοποιούνται λειτουργίες όπως:

- Προσθήκη / Αφαίρεση ενός στοιχείου επεξεργασίας.
- Προσθήκη / Αφαίρεση μεταβλητής σε στοιχείο επεξεργασίας.
- Αποθήκευση ενός στοιχείου επεξεργασίας στη βιβλιοθήκη στοιχείων

Όλες οι παραπάνω λειτουργίες είναι διαθέσιμες με δεξί κλικ πάνω σε κάποιο στοιχείο ή με κλικ πάνω στις επιλογές που έχουν το εικονίδιο . Ακόμη τα στοιχεία που είναι επιλεγμένα στο παράθυρο object view εμφανίζουν λεπτομέρειες στο παράθυρο ιδιοτήτων (property inspector).

3.2 Παράθυρο Ιδιοτήτων (Property Inspector)

3.2.1 Παράθυρο ιδιοτήτων Connector Pin



Εικόνα 4: Ιδιότητες Connector Pin

φαίνονται οι ιδιότητες μιας μεταβλητής ενός στοιχείου επεξεργασίας (connector pin). Αναλυτικότερα:

Bus Size: Είναι ο αριθμός των συνδέσεων που έχει το connector pin πάνω του. Δηλαδή ένα pin εξόδου που είναι συνδεδεμένο με 2 pins εισόδου άλλων στοιχείων επεξεργασίας θα έχει bus size ίσο με 2 ενώ ένα pin με Bus Size ίσο με 0 σημαίνει ότι δεν είναι συνδεδεμένο με κάποιο άλλο από τα στοιχεία επεξεργασίας

Datatype: Πρόκειται για τον τύπο δεδομένων του connector pin. Έχει σημασία να οριστεί ο τύπος δεδομένων του connector pin διότι του αποδίδεται τιμή μέσω της scripting γλώσσας. Ένα connector pin μπορεί να είναι ενός εκ των ακόλουθων τύπων δεδομένων:

_String: Συμβολοσειρά. Το pin μπορεί να περιέχει λέξεις ή αλφαριθμητικά δεδομένα.
_Integer: Ακέραιος αριθμός.
_float: Πραγματικός Αριθμός.
_Boolean: Δυαδικός αριθμός (τιμή True / False)

Flowcontrol: Ορίζει εάν το pin είναι pin εισόδου δεδομένων ή pin εξόδου δεδομένων. Δυνατές τιμές είναι:

fcInput: Το επιλεγμένο pin είναι pin εισόδου
fcOutput: Το επιλεγμένο pin είναι pin εξόδου

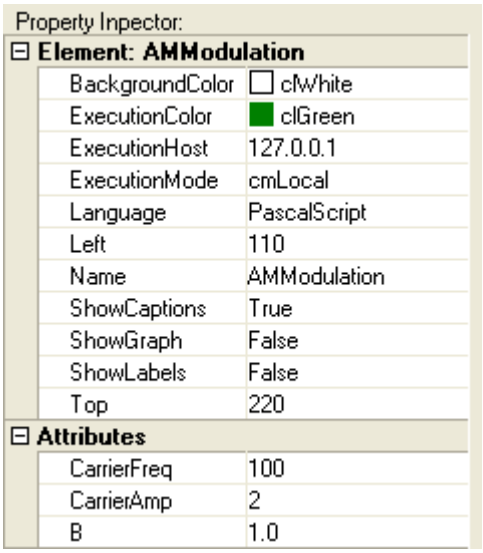
LogOutput: Boolean. Αν είναι ενεργή αυτή η επιλογή τότε η τιμές που λαμβάνει το pin κατά τη διάρκεια της εξομοίωσης σημειώνονται στο Log window.

MonitorOutput: Boolean. Αν είναι ενεργοποιημένη η συγκεκριμένη επιλογή (True) τότε δημιουργείται ένα γράφημα στο παράθυρο γραφημάτων που απεικονίζει τις τιμές που έχει πάρει το συγκεκριμένο pin κατά την εκτέλεση της εξομοίωσης. Εργαλείο χρήσιμο για την εξαγωγή συμπερασμάτων σχετικές με την εξομοίωση. Δυνατές τιμές είναι True ή False.

Name: Συμβολοσειρά που είναι το όνομα του pin. Η ονομασία που αποδίδεται στο pin υπάρχει και σαν μεταβλητή στο περιβάλλον scripting. Για το λόγο αυτό θα πρέπει να τηρούνται οι κανόνες ονοματοδοσίας μεταβλητών όπως στη γλώσσα προγραμματισμού pascal. Θα πρέπει δηλαδή η μεταβλητή να έχει όνομα με λατινικούς χαρακτήρες και αριθμούς καθώς και ο πρώτος χαρακτήρας να είναι γράμμα.

Pintype: Τύπος pin δηλαδή εάν πρόκειται για pin ροής δεδομένων ή ελέγχου. Δε χρησιμοποιείται, υπάρχει για μελλοντική χρήση.

3.2.2 Παράθυρο ιδιοτήτων Στοιχείου Επεξεργασίας



Εικόνα 5: Παράθυρο ιδιοτήτων στοιχείου επεξεργασίας

Εάν επιλέξουμε ένα στοιχείο επεξεργασίας από το object view ή από το παράθυρο ροής δεδομένων τότε εμφανίζονται οι ιδιότητες του επιλεγμένου στοιχείου όπως αυτές φαίνονται στην εικόνα 5. Υπάρχουν δυο ειδών ιδιότητες για τα στοιχεία. Το πρώτο σετ ιδιοτήτων εμφανίζεται σε όλα τα στοιχεία επεξεργασίας και είναι τα παρακάτω:

BackgroundColor: Είναι το χρώμα που θα έχει το στοιχείο επεξεργασίας όταν αυτό είναι ανενεργό.

ExecutionColor: Είναι το χρώμα που θα έχει το στοιχείο επεξεργασίας όταν εκτελείται ο αλγόριθμος που αντιστοιχεί σε αυτό το στοιχείο.

ExecutionHost: Η εφαρμογή υποστηρίζει κατανεμημένη εκτέλεση στοιχείων επεξεργασίας. Ένα στοιχείο επεξεργασίας μπορεί να εκτελεστεί σε διαφορετικό υπολογιστή εάν δοθεί τιμή στην ιδιότητα ExecutionHost διαφορετική από την τοπική ip του υπολογιστή (127.0.0.1), και τεθεί η κατάλληλη τιμή στην παράμετρο ExecutionMode. Η τιμες που μπορεί να πάρει το executionhost είναι IP διεύθυνση ή host name του απομακρυσμένου υπολογιστή.

ExecutionMode: Ορίζει εάν το στοιχείο επεξεργασίας θα εκτελεστεί τοπικά ή στον υπολογιστή που αναφέρεται στην ιδιότητα του executionhost. Οι τιμές που μπορεί να πάρει η ιδιότητα ExecutionMode είναι:

cmLocal: Εκτέλεση του στοιχείου επεξεργασίας τοπικά.
cmRemote: Εκτέλεση του στοιχείου επεξεργασίας στον υπολογιστή που αναφέρεται στο πεδίο ExecutionHost

Language: Η επιλεγμένη γλώσσα για την υλοποίηση του αλγορίθμου του στοιχείου επεξεργασίας. Η γλώσσα μπορεί να είναι μια από τις ακόλουθες:

PascalScript
Jscript
VBScript

Left: Η απόσταση του στοιχείου επεξεργασίας σε εικονοστοιχεία (pixels) από την αριστερή πλευρά του παραθύρου ροής δεδομένων

Name: Το όνομα του στοιχείου επεξεργασίας.

ShowCaption: Boolean. Εάν είναι ενεργό (true) τότε εμφανίζεται το όνομα του στοιχείου επεξεργασίας στο παράθυρο ροής δεδομένων πάνω ακριβώς από το στοιχείο.

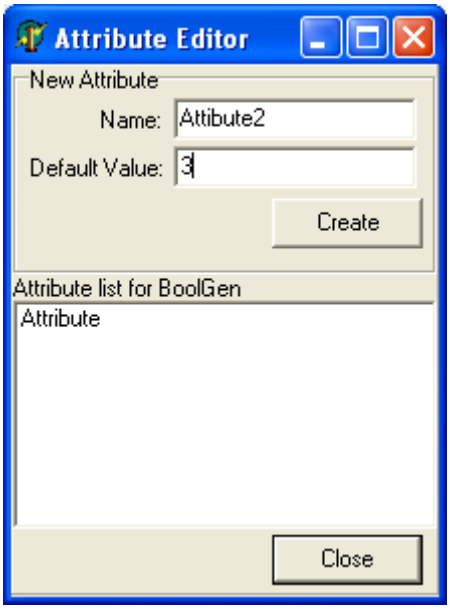
ShowGraph: Boolean. Δεσμευμένο για μελλοντική χρήση.

ShowLabels: Boolean. Εάν είναι ενεργό (true) εμφανίζονται τα ονόματα των ακροδεκτών (connector pins) του στοιχείου επεξεργασίας στο παράθυρο ροής δεδομένων.

Top: Η απόσταση του στοιχείου επεξεργασίας σε εικονοστοιχεία (pixels) από την κορυφή του παραθύρου ροής δεδομένων

Οι μεταβλητές που βρίσκονται κάτω από τον τίτλο attributes είναι επιπρόσθετες μεταβλητές που μπορεί να ορίσει ο χρήστης για ένα στοιχείο επεξεργασίας. Οι μεταβλητές αυτές είναι συνδεδεμένες με τον αλγόριθμο υλοποίησης και δίνουν στο τελικό χρήστη τη δυνατότητα να παραμετροποιεί ένα στοιχείο επεξεργασίας δίχως να επέμβει στον κώδικα, κάνοντας την εφαρμογή ακόμη πιο φιλική στον τελικό χρήστη.

3.2.3 Παράθυρο επεξεργασίας Attributes



Εικόνα 6: Παράθυρο Attribute Editor

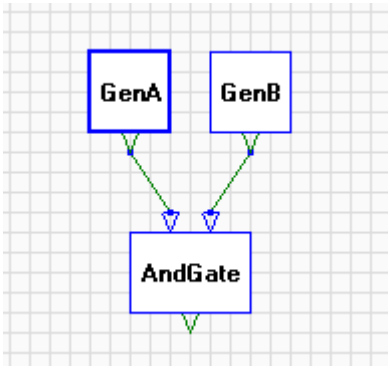
Για να εμφανίσουμε το παράθυρο επεξεργασίας των Attributes ενός στοιχείου επεξεργασίας επιλέγουμε από το μενού Element > Attribute Editor αφού προηγουμένως έχουμε επιλέξει το αντίστοιχο στοιχείο που μας ενδιαφέρει. Εμφανίζεται ένα πλαίσιο διαλόγου όπως στην εικόνα 6

Η παράμετρος Name είναι το όνομα της του Attribute όπως αυτό θα εμφανίζεται στον property Inspector. Η παράμετρος όνομα πρέπει να είναι σύμφωνη με τους κανόνες ονοματοδοσίας μεταβλητών καθώς στον αλγόριθμο δημιουργείται μια μεταβλητή με όνομα Attribute2 στο παράδειγμα της διπλανής εικόνας και προεπιλεγμένη τιμή ίση με 3. Για να διαγράψουμε ένα Attribute το επιλέγουμε από τη λίστα των Attributes και πατούμε το πλήκτρο DEL.

3.3 Παράθυρο Ροής Δεδομένων (Data Path)

3.3.1 Παράθυρο ροής δεδομένων – Γενικές λειτουργίες

Η αρχή λειτουργίας του παράθυρου ροής δεδομένων είναι η εναπόθεση στοιχείων επεξεργασίας πάνω του και η σύνδεσή τους έτσι ώστε η έξοδος ενός στοιχείου επεξεργασίας να είναι η είσοδος σε ένα άλλο. Στην εικόνα 7 φαίνονται τρία στοιχεία επεξεργασίας τοποθετημένα στο παράθυρο ροής δεδομένων τα οποία έχουν και δύο συνδέσεις μεταξύ τους. Συγκεκριμένα έχουν τοποθετηθεί δυο γεννήτριες τυχαίων δυαδικών αριθμών και μια πύλη AND. Στο παράθυρο ροής δεδομένων πραγματοποιούνται οι ακόλουθες διεργασίες:



Εικόνα 7: Παράθυρο Ροής Δεδομένων

- Η προσθήκη στοιχείων από τη βιβλιοθήκη στοιχείων επεξεργασίας.
- Η επεξεργασία / διαγραφή υπαρχόντων στοιχείων επεξεργασίας.
- Η σύνδεση στοιχείων μεταξύ τους.
- Η σωστή διάταξη των στοιχείων καθώς υπάρχει δυνατότητα μετακίνησης ή περιστροφής των στοιχείων πάνω στο γαλάζιο πλέγμα.

3.3.2 Προσθήκη στοιχείων επεξεργασίας στο Data Path

Υπάρχουν τρεις τρόποι με τους οποίους είναι εφικτό να προστεθεί ένα νέο στοιχείο επεξεργασίας στο παράθυρο ροής δεδομένων.

i. Σέρνοντας από το παράθυρο «Component Library» το στοιχείο επεξεργασίας που μας ενδιαφέρει και αφήνοντάς το πάνω στο παράθυρο του Data Path.
ii. Κάνοντας δεξί κλικ πάνω σε κενό χώρο στο παράθυρο ροής δεδομένων και επιλέγοντας από το μενού που προκύπτει το στοιχείο που μας ενδιαφέρει.
iii. Κάνοντας κλικ στην επιλογή «New Processing Element» στο Object View

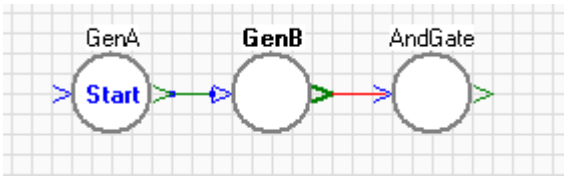
3.3.3 Σύνδεση στοιχείων μεταξύ τους

Για να πραγματοποιηθεί μια σύνδεση ανάμεσα σε δυο pins διαφορετικών στοιχείων επεξεργασίας αρκεί να σύρουμε το pin προέλευσης πάνω στο pin προορισμού. Σε κάθε περίπτωση όμως τα δυο pins θα πρέπει να βρίσκονται σε ξεχωριστά στοιχεία επεξεργασίας, το pin προέλευσης να είναι pin εξόδου και το pin προορισμού να είναι pin εισόδου ώστε να είναι εφικτή η σύνδεση των δυο pins.

3.4 Παράθυρο ροής ελέγχου (Control Path)

3.4.1 Εισαγωγή στο Control Path

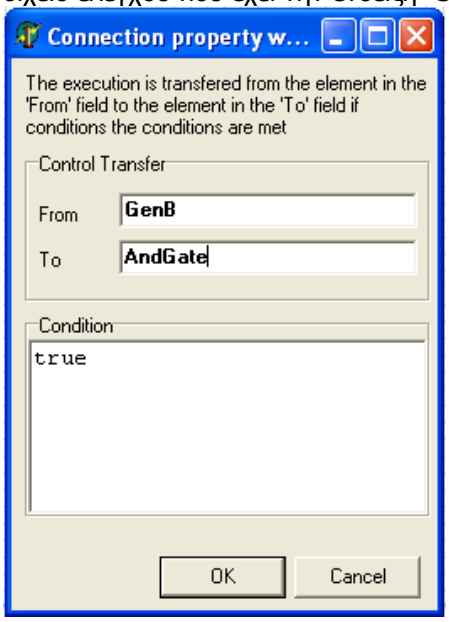
Το παράθυρο ροής ελέγχου ορίζει τη σειρά με την οποία θα εκτελεστούν τα στοιχεία επεξεργασίας. Για κάθε στοιχείο επεξεργασίας που τοποθετείται στο Data Path δημιουργείται ένα στοιχείο ελέγχου στο control path με το ίδιο όνομα. Τα στοιχεία ελέγχου του control path έχουν έναν ακροδέκτη εισόδου και έναν ακροδέκτη εξόδου. Η ροή της εξομοίωσης ξεκινά από το στοιχείο ελέγχου με την ένδειξη start. Το προεπιλεγμένο στοιχείο από όπου ξεκινά η εξομοίωση είναι αυτό που τοποθετήθηκε πρώτο στο Data Path. Για να επιλέξουμε το στοιχείο από όπου θα ξεκινήσει η εξομοίωση αρκεί να κάνουμε διπλό κλικ στο αντίστοιχο στοιχείο ελέγχου στο Control Path. Όταν ολοκληρωθεί η εκτέλεση του πρώτου στοιχείου η ροή ελέγχου μεταφέρεται στο στοιχείο που είναι συνδεδεμένο εφόσον επαληθεύει τη συνθήκη σύνδεσης μεταξύ τους.



Εικόνα 8: Συνδεσή ροής ελέγχου (Control Path)

3.4.2 Έλεγχος της ροής εκτέλεσης

Για να οριστεί η σειρά με την οποία θα εκτελεστούν τα στοιχεία επεξεργασίας θα πρέπει να συνδεθούν τα αντίστοιχα στοιχεία ελέγχου μεταξύ τους. Η ροή της εκτέλεσης ξεκινά από το στοιχείο ελέγχου που έχει την ένδειξη "Start" και συνεχίζει στα συνδεδεμένα στοιχεία ελέγχου.



Εικόνα 9: Παράθυρο διαλόγου σύνδεσης ροής ελέγχου

Έστω ότι έχουμε τοποθετήσει στο Data path τα στοιχεία όπως αυτά φαίνονται στην εικόνα 7, το αντίστοιχο control Path φαίνεται στην εικόνα 8. Ο τρόπος που έχουμε συνδέσει τα στοιχεία μεταξύ τους στην εικόνα 8 δείχνει ότι πρώτα θα εκτελεστεί το στοιχείο GenA, στη συνέχεια θα εκτελεστεί το στοιχείο GenB και τέλος θα εκτελεστεί το στοιχείο AndGate. Για να δημιουργηθεί μια σύνδεση μεταξύ δυο στοιχείων ελέγχου σέρνουμε την ακίδα εξόδου του στοιχείου ελέγχου από το οποίο μεταβιβάζουμε τον έλεγχο πάνω από την ακίδα εισόδου του στοιχείου ελέγχου προς το οποίο μεταβιβάζουμε τον έλεγχο εκτέλεσης. Με την πράξη αυτή εμφανίζεται το παράθυρο διαλόγου όπως στην εικόνα 9. Η σύνδεση της εικόνας περιγράφει τη σχέση της μεταφοράς ελέγχου από το στοιχείο GenB στο στοιχείο AndGate. Η έκφραση που υπάρχει ως συνθήκη για τη μεταφορά του ελέγχου από το GenB στο AndGate είναι σε κάθε περίπτωση αληθής συνεπώς πάντα όταν ολοκληρώνεται η εκτέλεση του κώδικα του στοιχείου GenA ο έλεγχος της εκτέλεσης μεταφέρεται στο στοιχείο AndGate.

3.4.3 Μεταφορά ελέγχου ροής υπό συνθήκες

Είναι δυνατό να πραγματοποιηθεί η μεταφορά ελέγχου εκτέλεσης της ροής του προγράμματος από ένα στοιχείο επεξεργασίας σε ένα άλλο εφόσον ισχύουν κάποιες συνθήκες. Για παράδειγμα είναι εφικτό να μεταφερθεί ο έλεγχος της ροής όπως φαίνεται στην εικόνα 8 από το στοιχείο AndGate ξανά στην αρχή εφόσον η έξοδος στο Pin Y του στοιχείου AndGate έχει τιμή 0. Το αποτέλεσμα μιας τέτοιας σύνδεσης είναι η εκτέλεση της εξομοίωσης μέχρι το στοιχείο AndGate να παρουσιάσει στο pin Y τιμή 1. Για να πραγματοποιηθεί μια τέτοια σύνδεση σέρνουμε την έξοδο του στοιχείου ελέγχου AndGate προς την είσοδο του στοιχείου ελέγχου GenA δημιουργώντας μια σύνδεση και εμφανίζοντας το παράθυρο διαλόγου. Αρκεί στο παράθυρο διαλόγου να πληκτρολογήσουμε την αντίστοιχη έκφραση δυαδικής λογικής (Boolean expression) στο πλαίσιο κειμένου με τίτλο "Condition". Η συγκεκριμένη έκφραση πρέπει να επιστρέφει τιμή true ή false και μπορούμε να αναφερθούμε σε τιμές που έχουν pins στοιχείων επεξεργασίας με την ακόλουθη σύνταξη:

```
Ονομαστοιχειου('ονομαPin')
```

Συνεπώς για να αναφερθούμε στην τιμή που έχει η ακίδα Y του στοιχείου AndGate η σύνταξη είναι:

```
AndGate('Y')
```

Άρα ολόκληρη η δυαδική έκφραση που θα μεταφέρει τον έλεγχο εκτέλεσης στο επόμενο στοιχείο ελέγχου εφόσον η τιμή της ακίδας Y του στοιχείου AndGate είναι ίσο με το μηδέν είναι η ακόλουθη:

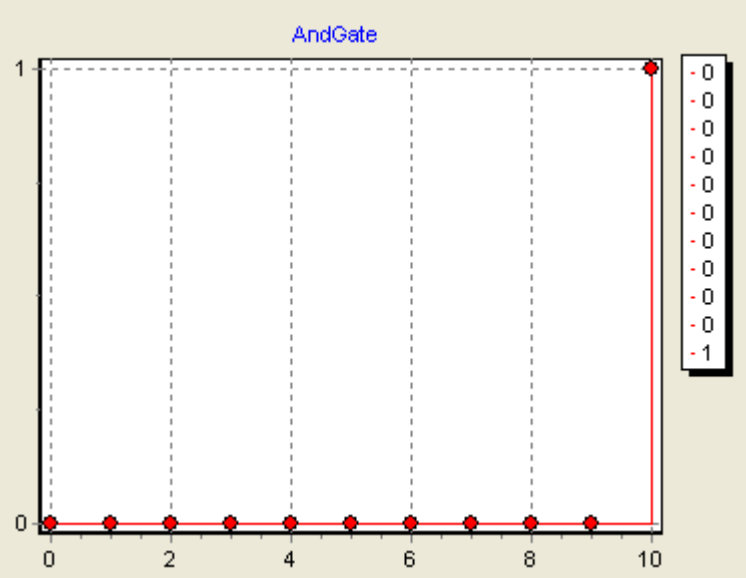
```
AndGate('Y') = 0
```

Όποτε η παραπάνω δυαδική έκφραση είναι αληθής τότε ο έλεγχος της ροής του προγράμματος θα μεταφέρεται στο παράδειγμά μας στο στοιχείο GenA, δηλαδή η εξομοίωση θα γίνεται ξανά από την αρχή έως η παραπάνω συνθήκη να είναι ψευδής να γίνει δηλαδή η τιμή του AndGate('Y') ίση με 1.

Με την παραπάνω λογική είναι εφικτό να υλοποιηθούν πολύπλοκες συνθήκες ελέγχου. Μπορεί για παράδειγμα ένα στοιχείο ελέγχου να έχει τρεις συνδέσεις στην εξόδό του και ανάλογα με τη συνθήκη ελέγχου που έχει κάθε σύνδεση η ροή της εκτέλεσης μπορεί να μεταφέρεται σε διαφορετικό κάθε φορά στοιχείο επεξεργασίας.

3.5 Παράθυρο Graphs

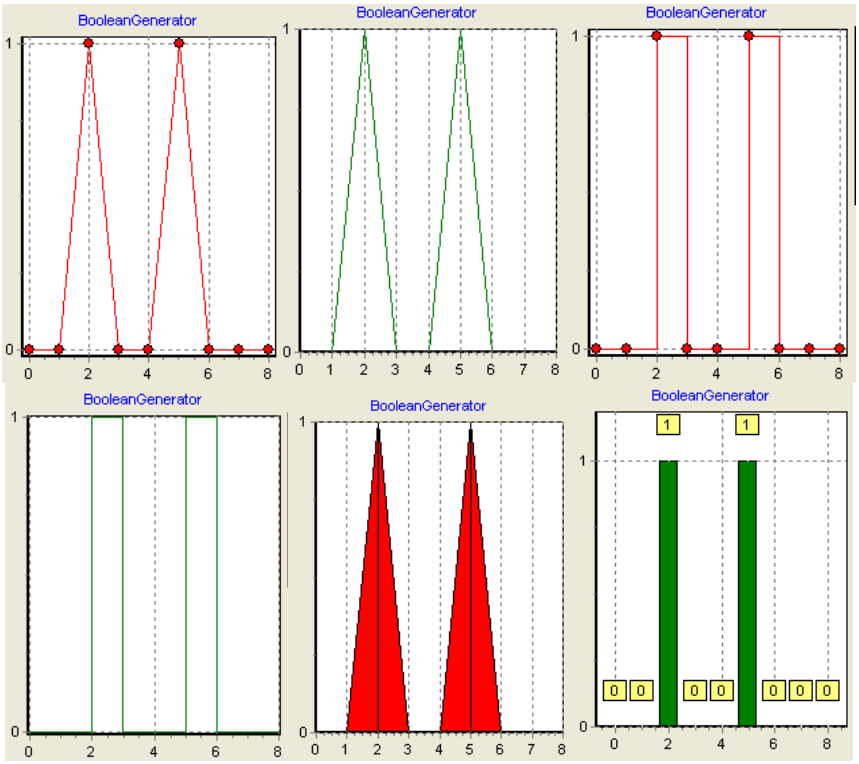
Το παράθυρο Graphs εμφανίζει γραφήματα με τις τιμές που παίρνουν οι ακροδέκτες προς παρακολούθηση (monitored pins) όπως έχουν επιλεγεί από το παράθυρο ιδιοτήτων (Κεφ. 3.2.1 Παράθυρο ιδιοτήτων Connector Pin σελ 12). Κάθε φορά που εκτελείται ένα στοιχείο επεξεργασίας οι ακροδέκτες του γίνονται δέκτες τιμών (εισόδου ή εξόδου). Εφόσον ένας ακροδέκτης έχει επιλεγεί για παρακολούθηση, σε κάθε κύκλο εκτέλεσης σημειώνεται η τιμή που έχει στο γράφημα. Δημιουργείται ένα ξεχωριστό γράφημα για κάθε στοιχείο επεξεργασίας που παρακολουθείται. Εάν παρακολουθούνται παραπάνω από μια μεταβλητές του ιδίου στοιχείου επεξεργασίας τότε αυτές απεικονίζονται στο ίδιο γράφημα του στοιχείου. Η εικόνα 10 δείχνει το γράφημα εξόδου όπως αυτό φαίνεται στο παράθυρο Graphs, εφόσον πραγματοποιηθεί μια εξομοίωση με μεταφορά ροής ξανά στην αρχή έως ότου παρουσιαστεί στην έξοδο Y τιμή 1. Στον άξονα Y φαίνεται το εύρος τιμών που λαμβάνει η μεταβλητή και στον άξονα X είναι ο αύξων κύκλος εκτέλεσης του στοιχείου.



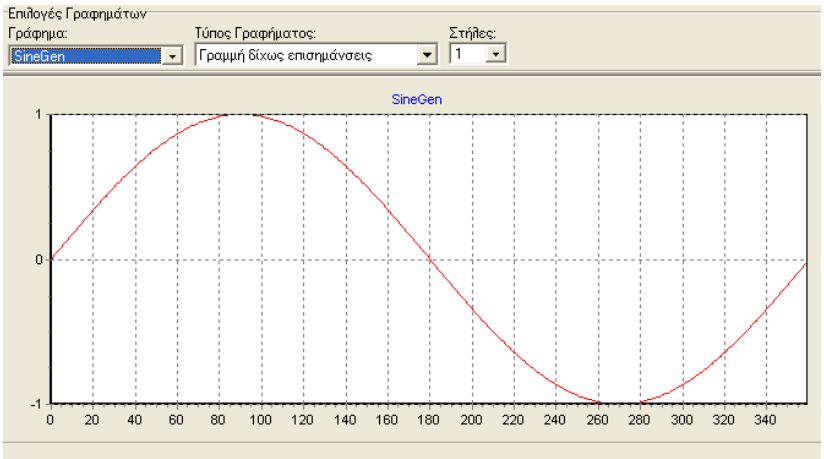
Εικόνα 10: Γράφημα εξόδου Y του στοιχείου AndGate

Υπάρχουν πρόσθετες επιλογές στο παράθυρο των γραφημάτων που ορίζουν τον τρόπο με τον οποίο θα εμφανίζεται το γράφημα στον χρήστη. Μπορεί να τροποποιηθεί ο αριθμός των στηλών των γραφημάτων ή ο τύπος του γραφήματος. Υποστηρίζονται οι ακόλουθοι τύποι γραφημάτων:

- **Γραμμή με επισημάνσεις**. Στο γράφημα εμφανίζονται χαρακτηριστικά σημάδια (τρίγωνο, ρόμβος, κύκλος, τετράγωνο), σε κάθε τιμή κάθε καμπύλης του επιλεγμένου γραφήματος.
- **Γραμμή δίχως επισημάνσεις**. Είναι το πιο γρήγορο κατά την εκτέλεση.
- **Σκαλοπάτι με επισημάνσεις**. Χρησιμοποιείται κυρίως για την εξομοίωση συστημάτων δυαδικής λογικής
- **Σκαλοπάτι δίχως επισημάνσεις**. Όπως το προηγούμενο αλλά χωρίς τα σημεία επισημάνσεως.
- **Γράφημα Περιοχή:** Γεμίζει το γράφημα με ένα χρώμα.
- **Μπάρα**. Το γράφημα αποτελείται από κατακόρυφες μπάρες.



Εικόνα 11: Επιλογές απεικόνισης Γραφημάτων



Εικόνα 12: Γράφημα γεννήτριας ημιτονικού σήματος

3.6 Παράθυρο Log

Το παράθυρο Log έχει παρόμοια λειτουργία με αυτήν του παραθύρου graphs. Η διαφορά είναι ότι σε κάθε κύκλο εκτέλεσης ενός στοιχείου επεξεργασίας απεικονίζονται οι τιμές που λαμβάνουν οι μεταβλητές όπου γίνεται παρακολούθηση με μορφή κειμένου αντί να γίνεται γραφική απεικόνιση. Για παράδειγμα εάν επιλέξουμε να κάνουμε log τη μεταβλητή Y της πύλης AndGate του παραπάνω παραδείγματος, θα έχουμε την ακόλουθη έξοδο:

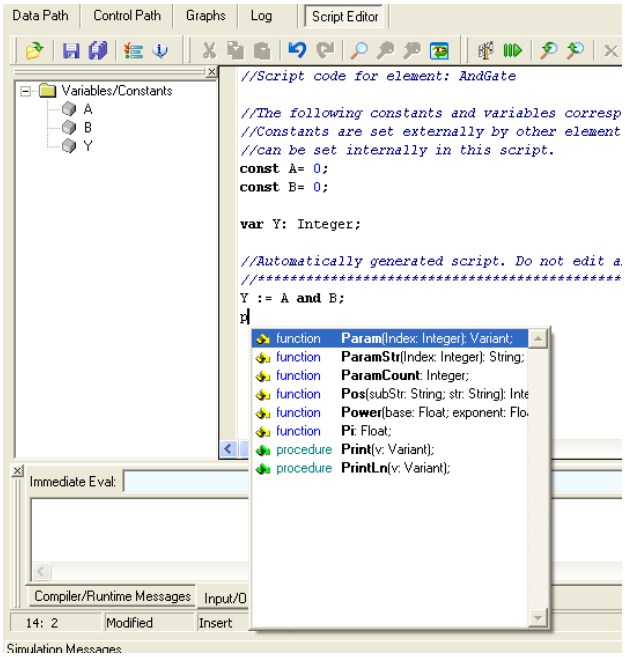
```
27/4/2006 4:44:08 μμ : AndGate('Y') = 0
27/4/2006 4:44:08 μμ : AndGate('Y') = 0
27/4/2006 4:44:08 μμ : AndGate('Y') = 0
27/4/2006 4:44:09 μμ : AndGate('Y') = 0
27/4/2006 4:44:09 μμ : AndGate('Y') = 0
27/4/2006 4:44:09 μμ : AndGate('Y') = 0
27/4/2006 4:44:09 μμ : AndGate('Y') = 0
27/4/2006 4:44:09 μμ : AndGate('Y') = 0
27/4/2006 4:44:09 μμ : AndGate('Y') = 0
27/4/2006 4:44:09 μμ : AndGate('Y') = 0
27/4/2006 4:44:09 μμ : AndGate('Y') = 1
```

Από την επιλογή Log του κεντρικού μενού υπάρχει η δυνατότητα αποθήκευσης, φόρτωσης ή καθαρισμού του παραθύρου Log επιλέγοντας αντίστοιχα "Save…", "Open..." ή "Clear".

3.7 Διεπαφή Scripting Προγραμματισμού

3.7.1 Εισαγωγή στη Scripting διεπαφή της εφαρμογής

Το scripting engine που διαθέτει η εφαρμογή ίσως είναι το πιο θεμελιώδης στοιχείο της καθώς χαρίζει επεκτασιμότητα και δυνατότητα εμπλουτισμού της εφαρμογής. Σε κάθε στοιχείο επεξεργασίας αντιστοιχεί ένας αλγόριθμος. Ο αλγόριθμος κάθε στοιχείου επεξεργασίας είναι η υλοποίηση της μαθηματικής έκφρασης που σχετίζει τις εισόδους με τις εξόδους του. Η υλοποίηση αυτής της έκφρασης μπορεί να πραγματοποιηθεί σε PascalScript, JScript ή VBScript. Στοιχεία επεξεργασίας είναι δυνατό να υλοποιηθούν σε διαφορετικές γλώσσες προγραμματισμού και να συνδεθούν μεταξύ τους στο Data Path και στο Control Path.



Εικόνα 13: Περιβάλλον Scripting Προγραμματισμού

Η εικόνα 13 παρουσιάζει την υλοποίηση μιας πύλης AND στο περιβάλλον scripting με χρήση PascalScript. Τα pins που διαθέτει το στοιχείο επεξεργασίας είναι ορισμένα αυτόματα ως μεταβλητές με το όνομα που τους έχει δώσει ο χρήστης από το παράθυρο ιδιοτήτων.
Συγκεκριμένα τα pins εισόδου ορίζονται ως σταθερές (const) καθώς δεν πρέπει να τους αποδοθεί τιμή από τον αλγόριθμο ενώ τα pins εξόδου είναι ορισμένα ως μεταβλητές (var).

3.7.2 Αυτόματοποιημένος Κώδικας (Auto generated code)

Τα attributes που ορίζονται σε ένα στοιχείο δημιουργούν αυτόματο κώδικα που περιέχει το όνομα της μεταβλητής και τιμή ίση με αυτή όπως έχει οριστεί στον Property Inspector. Για να τροποποιήσουμε την τιμή σε ένα Attribute ώστε να αποθηκευτεί η αλλαγή θα πρέπει να δοθεί τιμή στον Object Inspector. Γενικότερα ότι αλλαγές γίνουν στον κώδικα που παράγεται αυτόματα δεν αποθηκεύονται, χάνονται!

Για το λόγο αυτό ο κώδικας θα πρέπει να γράφεται κάτω από τη γραμμή σχολίου που περιέχει τους αστερίσκους. Οι μεταβλητές που έχουν δημιουργηθεί από τον αυτοματοποιημένο κώδικα μπορούν να χρησιμοποιηθούν στον αλγόριθμο.

Μια διαφορά που έχει μια υλοποίηση σε PascalScript σε σχέση με υλοποίηση σε VBScript ή JScript πέρα από τη σύνταξη είναι και ο τρόπος με τον οποίο γίνεται η ανάγνωση ή απόδοση τιμής στα Pins. Στην PascalScript δημιουργείται αυτόματος κώδικας με το όνομα του Pin και τον τύπο του. Τα Pins εισόδου ορίζονται ως Const και δεν είναι εφικτή η απόδοση τιμής σε αυτά. Τα pins Εξόδου ορίζονται ως μεταβλητές με τον τύπο δεδομένων τους και μπορεί να τους δοθεί τιμή, και αυτή αποθηκεύεται στο Pin.

Κώδικας 1: Παράδειγμα αυτόματου κώδικα σε PascalScript

```
const Elemname='AMModulation';

//The following constants and variables correspond to the pins.
//Constants are set externally by other elements while variables
//can be set internally in this script.
const signal= 0.0;
const t= 0.0;

var Y: Float;

//Attribute Declarations
const B = 1.0;
const CarrierAmp = 2;
const CarrierFreq = 100;
//Auto generated script. Add your code BELOW the star line
//*****************************************************
```

Το παραπάνω δείγμα κώδικα είναι ο αυτοματοποιημένος κώδικας που παράγεται από ένα στοιχείο με 2 pins εισόδου με όνοματα signal και t. Το στοιχείο περιέχει επίσης ένα Pin εξόδου με όνομα Y και 3 attributes. Ο κώδικας που καλείται να εισάγει ο χρήστης πρέπει να πληκτρολογηθεί κάτω από τη γραμμή με τους αστερίσκους. Το αντίστοιχο αυτοματοποιημένο κομμάτι κώδικα σε VBScript και JScript φαίνεται στους παρακάτω κώδικες (Κώδικας 2 και

```
const Elemname="AMModulation"

'Read and set pin values via pin("pinname")
'Attribute Declarations
const B = 1.0
const CarrierAmp = 2
const CarrierFreq = 100
'Auto generated script. Add your code BELOW the star line
'*****************************************************
```

Κώδικας 2: Αυτοματοποιημένος κώδικας σε VBScript

```
Elemname='AMModulation';

//Read and set pin values via pin('pinname')
//Attribute Declarations
B = 1.0;
CarrierAmp = 2;
CarrierFreq = 100;
//Auto generated script. Add your code BELOW the star line
//*****************************************************
```

Κώδικας 3: Αυτοματοποιημένος κώδικας σε JScript

Από τη σύγκριση του κώδικα παρατηρούμε ότι στην υλοποίηση με VBScript ή JScript δεν υπάρχουν ορισμένες οι μεταβλητές που να αντιστοιχούν στα pin εισόδου ή εξόδου. Αυτό συμβαίνει διότι υπάρχει διαφορετικός τρόπος επικοινωνίας με τα Pins της εφαρμογής υλοποιώντας τον αλγόριθμος σε JScript ή VBScript. Η ανάγνωση / απόδοση τιμών στα Pins γίνεται μέσου του αντικειμένου **Pin**, όπως αυτό αναφέρεται στο κεφάλαιο 5 (σελ. 34)

3.7.3 Παράθυρο Simulation Messages

Στο παράθυρο αυτό εμφανίζονται μηνύματα σφάλματος ή πληροφορίας που δημιουργούνται κατά την εκτέλεση της εξομοίωσης της εφαρμογής. Εάν υπάρχει συντακτικό λάθος ή σφάλμα κατά τη διάρκεια της εκτέλεσης της εξομοίωσης, θα εμφανιστεί σε αυτό το παράθυρο. Επίσης στο παράθυρο αυτό εμφανίζονται όλα τα μηνύματα στοιχείων επεξεργασίας που περιέχουν την εντολή print ή println στον αλγοριθμό τους.

```
PrintLn, Print
function PrintLn(msg: string);
function Print(msg: string);
```

Η συνάρτηση PrintLn εμφανίζει το μήνυμα msg στο παράθυρο Simulation Messages. Χρήσιμο για απασφαλμάτωση κώδικα ή για εμφάνιση πληροφοριών στο χρήστη.
Αντίστοιχα σε προγραμματισμό JScript ή VBScript η αντίστοιχη επικοινωνία με την εφαρμογή μπορεί να γίνει μέσω της συνάρτησης:

```
Application.println(msg: string)
```

3.8 Χαρακτηριστικά Scripting Διεπαφής

3.8.1 Χαρακτηριστικά υποστηριζόμενα από όλες τις γλώσσες

Η διεπαφή Scripting υποστηρίζει καποια βασικά χαρακτηριστικά ως επεξεργαστής κειμένου για όλες τις γλώσσες προγραμματισμού όπως:

- **Search & Replace**: Δίνει στο χρήστη τη δυνατότητα να αναζητήσει και να αντικαταστήσει ένα κομμάτι κώδικα
- **Open & Save**: Δυνατότητα αποθήκευσης και φόρτωσης αρχείου script από μονάδα αποθήκευσης.
- **Syntax Highlighting**: Χρωματίζει με διαφορετικό τρόπο τις λέξεις κλειδιά, τα σχόλια, τους αριθμούς και τις συμβολοσειρές ώστε να είναι πιο ευανάγνωστος και κατανοητός ο κώδικας του αλγόριθμου.

3.8.2 Χαρακτηριστικά υποστηριζόμενα από την PascalScript

Το ενσωματωμένο περιβάλλον προγραμματισμού περιλαμβάνει πλήρη δυνατότητες απασφαλμάτωσης κώδικα όταν η γλώσσα υλοποίησης είναι η PascalScript.

Μερικά από τα χαρακτηριστά του περιβάλλοντος scripting είναι:

- **Code Tree:** Δείχνει σε δενδρική μορφή όλες τις ορισμένες κλάσεις, συναρτήσεις και μεταβλητές του αλγορίθμου. Επιλέγοντας ένα στοιχείο από το code tree τότε εμφανίζεται η γραμμή στην οποία ορίζεται το επιλεγμένο στοιχείο.

- **Code Insight:** Πατώντας ctrl+space εμφανίζεται ένα παράθυρο με πρότασεις για τη συμπλήρωση της έκφρασης που πληκτρολογεί ο χρήστης

- **Breakpoints:** Ο χρήστης έχει τη δυνατότητα να εισάγει breakpoints. Εάν πατήσει run μέσα από το περιβάλλον απασφαλμάτωσης θα τρέξει μόνο ο κώδικας του επιλεγμένου στοιχείου επεξεργασίας και θα σταματήσει σε τυχόν Breakpoints που συναντήσει.

- **Immediate Eval:** Πληκτρολόγηση μιας εντολής και η άμεση εκτέλεσή της.

- **Compiler / Runtime Messages:** Περιοχή εμφάνισης μηνυμάτων από το scripting engine.

- **Code Completion:** Έστω η ύπαρξη διεπαφής μιας κλάσης (class interface), πατώντας ctrl+shift+c δημιουργείται ο σκελετός των μεθόδων που υπάρχουν ορισμένες στην κλάση.

Για παράδειγμα έστω το ακόλουθο απόσπασμα κώδικα:

```
type
TMyClass = class
  function ReturnInt: integer;
  procedure ManageStr(s: string);
end;
```

Κώδικας 4: Ορισμός μιας Κλάσης σε PascalScript

Πατώντας ctrl+shift+c θα δημιουργηθεί αυτόματα ο κορμός της κλάσης που είναι το παρακάτω κομμάτι κώδικα. Ανάμεσα στις συναρτήσεις χρειάζεται να γίνει η συμπλήρωση του αλγορίθμου της κάθε συνάρτησης.

```
function TMyClass.ReturnInt: Integer;
begin

end;

procedure TMyClass.ManageStr(s: String);
begin

end;
```

Κώδικας 5: Υλοποίηση κλάσης σε PascalScript

4 Προγραμματισμός με PascalScript

4.1 Εσωτερικές συναρτήσεις

Το scripting engine της PascalScript διαθέτει ένα πλήθος ενσωματωμένων συναρτήσεων που μπορούν να χρησιμοποιηθούν για διάφορες εργασίας όπως μαθηματικές συναρτήσεις ή ακόμα συναρτήσεις για διαχείριση αρχείων και σύνδεση με βάση δεδομένων. Η σύνταξη και οι υποστηριζόμενοι τύποι δεδομένων μοιάζουν με τη σύνταξη της γλώσσας Delphi.

4.1.1 Μαθηματικές συναρτήσεις

4.1.1.1 Μαθηματικές τριγωνομετρικές συναρτήσεις

Sin, Sinh

```
function Sin(X: Float): Float;
function SinH(X: Float): Float;
```

Η συνάρτηση επιστρέφει το ημίτονο της παραμέτρου σε ακτίνια. Όπου X μια έκφραση πραγματικού αριθμού. Η Sin επιστρέφει το ημίτονο σε ακτίνια. Η Sinh Επιστρέφει το υπερβολικό ημίτονο του X.

Cos, Cosh

```
function Cos(X: Float): Float;
function CosH(X: Float): Float;
```

Επιστρέφει το συνημίτονο σε ακτίνια
Η παράμετρος X είναι έκφραση πραγματικού αριθμού. Το αποτέλεσμα επιστρέφεται σε ακτίνια.

Tan, Tanh

Επιστρέφει την εφαπτομένη της γωνίας σε ακτίνια

```
(Tan(x) = Sin(x) / Cos(x))
```

ArcSin, ArcSinh

```
function ArcSin(X: Float): Float;
function ArcSinH(X: Float): Float;
```

Υπολογίζει το τόξο ημιτόνου δεδομένης της γωνίας σε ακτίνια.
ArcSin επιστρέφει το τόξο ημιτόνου X. Το X πρέπει να είναι μεταξύ -1 και 1 και η τιμή επιστροφής είναι στο διάστημα [-Pi/2..Pi/2], σε ακτίνια.

ArcCos, ArcCosh

```
function ArcCos(X: Float): Float;
function ArcCosH(X: Float): Float;
```

Υπολογίζει το τόξο συνημιτόνου δεδομένης της γωνίας σε ακτίνια.
ArcSin επιστρέφει το τόξο ημιτόνου X. Το X πρέπει να είναι μεταξύ -1 και 1 και η τιμή επιστροφής είναι στο διάστημα [0..Pi], σε ακτίνια.

ArcTan, ArcTanh

```
function ArcTan(X: Float): Float;
function ArcTanH(X: Float): Float;
```

Υπολογίζει το τόξο εφαπτομένης.
Το X πρέπει να είναι στο διάστημα [-1,1]

function CoTan(X: Float): Float;

Συνεφαπτομένη X. Ταυτόσημο με το

```
1 / Tan(X)
```

Μην καλείτε τη συνάρτηση με X = 0 !

Pi

```
function Pi: Float;
```

Επιστρέφει: 3.1415926535897932385.
Χρησιμοποιήστε το PI σε μαθηματικές εκφράσεις που το απαιτούν.

DegToRad

```
function DegToRad(x:Float):Float;
```

Μετατρέπει το όρισμα από μοίρες σε ακτίνια.
Ορισμός:

RadToDeg

```
function RadToDeg(x:Float): Float;
```

Μετατροπή από ακτίνια σε μοίρες

Hypot

```
function Hypot(x, y: Float):
Float;
```

Υπολογίζει το μήκος της υποτείνουσας τριγώνου με πλευρές x,y

4.1.1.2 Μαθηματικές συναρτήσεις πραγματικών αριθμών

Abs
```
function Abs(x: Float): Float;
```
Επιστρέφει την απόλυτη τιμή του ορίσματος X, όπου X πραγματικός αριθμός

Exp
```
function Exp(x: Float): Float;
```
Επιστρέφει την τιμή e^{x} όπου x, πραγματικός αριθμός.

Ln
```
function Ln(x: Float): Float;
```

Log2
```
function Log2(x: Float): Float;
```
Επιστρέφει το λογάριθμο του X με βάση το 2

Log10
```
function Log10(x:Float): Float;
```
Επιστρέφει το λογάριθμο του X με βάση το 10.

LogN
```
function LogN(Base, x: Float): Float;
```
Η LogN επιστρέφει το λογάριθμο του X με βάση Base.

Sqrt
```
function Sqrt(x: Float): Float;
```
Όπου X πραγματικός αριθμός. Το αποτέλεσμα είναι η τετραγωνική ρίζα του X

Sqr
```
function Sqr(x: Float): Float;
```
Επιστρέφει το τετράγωνο του X, δηλαδή επιστρέφει X*X

Int
```
function Int(x: Float): Float;
```
Επιστρέφει το ακέραιο μέρος του x, δηλαδή το x στρογγυλοποιημένο προς το 0

Frac
```
function Frac(x: Float): Float;
```
Επιστρέφει το πραγματικό μέρος του X
Το αποτέλεσμα ισοδυναμεί με την πράξη X - Int(X).

Trunc
```
function Trunc(x:Float):Integer;
```
Η συνάρτηση Trunc επιστρέφει εναν ακέραιο αριθμό που προκύπτει από το ακέραιο μέρος του ορίσματος x. Το αποτέλεσμα είναι x στρογγυλοποιημένο προς το 0

Round
```
function Round(x: Float): Integer;
```
Η συνάρτηση Round επιστρέφει ένα ακέραιο αριθμό από που προκύπτει από τη στρογγυλοποίηση του πραγματικού αριθμού x του ορίσματος προς τον κοντινότερο ακέραιο. Εάν ο x βρίσκεται ακριβώς ανάμεσα σε 2 ακεραίους τότε στρογγυλοποιείται προς τον ζυγό αριθμό. Δηλ, Round(2.5) = 2 και Round(1.5) = 2
Ο παραπάνω τρόπος υπολογισμού μπορεί να αλλάξει καλώντας πρώτα τη συνάρτηση SetRoundMode ή τη συνάρτηση Set8087CW

Power
```
function Power(Base, Exponent: Float): Float;
```
Η συνάρτηση Power επιστρέφει τον αριθμό Base στη δύναμη Exponent

Max
```
function Max(v1,v2: Float): Float;
```
Επιστρέφει το μεγαλύτερο από τα δυο ορίσματα

Min
```
function Min(v1,v2: Float): Float;
```
Επιστρέφει το μικρότερο από τα δυο ορίσματα.

4.1.1.3 Μαθηματικές συναρτήσεις τυχαίων αριθμών

Random
```
function Random: Float;
```
Επιστρέφει ένα τυχαίο πραγματικό αριθμό μεταξύ 0 <= X < 1

RandomInt
```
function RandomInt(Range: Integer): Integer;
```
Επιστρέφει ένα τυχαίο ακέραιο αριθμό στο διάστημα 0 <= X < Range

Randomize
```
procedure Randomize;
```
Αρχικοποιεί τη γεννήτρια τυχαίων αριθμών με μια τυχαία τιμή (βάσει της ώρας του συστήματος) Η γεννήτρια τυχαίων αριθμών πρέπει να αρχικοποιείται με κλήση στη Randomize ή με κλήση στη SetRandSeed
Δε χρειάζεται να κάνετε πάνω από μια κλήση σε αυτές τις συναρτήσεις, τις καλείτε μια φορά και μετά καλείτε τη συνάρτηση random όσες φορές επιθυμείτε.

RandG
```
function RandG(Mean, StdDev: Float): Float
```
Η συνάρτηση RandG παράγει τυχαίους αριθμούς με γκαουσσιανή κατανομή γύρω από την παράμετρο Mean.
Χρήσιμο στην εξομοίωση δεδομένων με σφάλματα δειγματοληψίας και και με αναμενόμενες αποκλήσεις από το Mean.

RandSeed
```
function RandSeed: Integer;
```
Η συνάρτηση επιστρέφει τον αύξων αριθμό της παρούσας ακολουθίας τυχαίων αριθμών.

SetRandSeed
```
procedure SetRandSeed(val: Integer);
```
Επιλογή της ακολουθίας τυχαίων αριθμών

4.1.2 Συναρτήσεις Συμβολοσειρών (String Functions)

IntToStr
```
function IntToStr(i: Integer): String;
```
Μετατρέπει έναν ακέραιο σε συμβολοσειρά

StrToInt
```
function StrToInt(S: String): Integer;
```
Μετατρέπει έναν αριθμό που περιέχεται στη συμβολοσειρά S (σε δεκαδική ή δεκαεξαδική μορφή) σε συμβολοσειρά. Εάν δεν υπάρχει σωστός αριθμός τότε γίνεται σφάλμα.

StrToIntDef
```
function StrToIntDef(const S: string; Default:Integer): Integer;
```
Η StrToIntDef μετατρέπει όπως παραπάνω τη συμβολοσειρά S σε ακέραιο, και σε περίπτωση σφάλματος επιστρέφεται ο αριθμός Default

IntToHex
```
function IntToHex(Value: Integer; Digits:Integer): string;
```
Η IntToHex μετατρέπει ένα ακέραιο σε συμβολοσειρά που περιέχει τον αριθμό σε δεκαεξαδική μορφή.
Όπου:
Value: Ο ακέραιος προς μετατροπή
Digits: Ο ελάχιστος αριθμός ψηφίων που θα περιέχει το String που επιστρέφεται

FloatToStr
```
function FloatToStr(Value: Float): string;
```
Μετατρέπει ένα πραγματικό αριθμό σε συμβολοσειρά, για μεγαλύτερο έλεγχο της μορφής που επιστρέφεται μπορείτε να κάνετε χρήση της FloatToStrF.

StrToFloat

```
function StrToFloat(const S:
string): float;
```

Χρησιμοποιήστε τη StrToFloat για να μετατρέψετε ένα string S, σε πραγματικό αριθμό. Το S μπορεί να περιέχει ένα προαιρετικό πρόσιμο (+ or -), μια ακολουθία ψηφίων με προαιρετική υποδιαστολή και μια προαιρετική μάντισσα. Η μάντισσα περιέχει το χαρακτήρα 'E' ή 'e' ακολουθούμενο από ένα προαιρετικό πρόσιμο (+ or -) και έναν ακέραιο αριθμό. Η μεταβλητή DecimalSeparator ορίζει το χαρακτήρα που χρησιμοποιείται ως υποδιαστολή. Δεν επιτρέπεται σύμβολο χωρισμού χιλιάδων ή σύμβολο νομίσματος στη συμβολοσειρά.

StrToFloatDef

```
function StrToFloatDef(const S:
string; def:float): float;
```

Ίδια με την παραπάνω συνάρτηση με τη διαφορά ότι σε περίπτωση σφάλματος επιστρέφεται η τιμή του ορίσματος def

Chr

```
function Chr( x: Integer):
String;
```

Chr επιστρέφει το χαρακτήρα με κωδικό ASCII ίσο με τη παράμετρο X

Ord

```
function Ord(s: String):
Integer;
```

Επιστρέφει τον αριθμό ASCII που αντιστοιχεί στον χαρακτήρα S.

CharAt

```
function CharAt(S: String; i:
Integer): String;
```

Επιστρέφει το χαρακτήρα στη θέση I στο string S.

SetCharAt

```
procedure SetCharAt(var
x:string; i: integer;value:
string);
```

Θέτει το χαρακτήρα στη θέση i ίση με την τιμή της παραμέτρου value

Delete

```
procedure Delete(var S: string;
Index, Count:Integer);
```

Διαγράφει ένα Count χαρακτήρες από το string s ξεκινώντας από το Index.
Εάν το index είναι μεγαλύτερο σε μήκος από το S ή μικρότερο του 1 τοτε δε διαγράφονται χαρακτήρες.

Insert

```
procedure Insert(Source:
string; var S: string; Index:
Integer);
```

Εισάγει το Source στο string S στη θέση index. Εάν Index είναι μικρότερο του 1, it is mapped to a 1 τότε θεωρείται ίσο με 1. Εάν είναι πέρα από το τέλος του string τότε γίνεται ίσο με το μήκος του string.

LowerCase

```
function LowerCase(S: string):
string;
```

Επιστρέφει τη συμβολοσειρά S με μόνο πεζούς χαρακτήρες. Η μετατροπή επηρεάζει μόνο 7-bit ASCII χαρακτήρες μεταξύ 'A' και 'Z'. Για τη μετατροπή 8-bit διεθνών χαρακτήρων χρησιμοποιήστε την AnsiLowerCase.

AnsiLowerCase

```
function AnsiLowerCase(S:
string): string;
```

Επιστρέφει την παράμετρο S περιέχοντας μόνο μικρούς χαρακτήρες

UpperCase

```
function UpperCase(S: string):
string;
```

Επιστρέφει τη συμβολοσειρά S με μόνο κεφαλαίους χαρακτήρες. Η μετατροπή επιρεάζει μονο 7-bit ASCII χαρακτήρες μεταξύ 'a' και 'z'. Για τη μετατροπή 8-bit διεθνών χαρακτήρων χρησιμοποιήστε την AnsiUpperCase.

AnsiUpperCase

```
function AnsiUpperCase(S:
string): string;
```

Επιστρέφει τη συμβολοσειρά S με μόνο κεφαλαίους χαρακτήρες

Pos
```
function Pos(Substr: string; S: string): Integer;
```
Επιστρέφει έναν ακέραιο που είναι η θέση του Substr εντός του string S. Το Substr και το S είναι strings. Γίνεται διάκριση μεταξύ πεζών και κεφαλαίων.
Εάν το Substr δε βρεθεί, η Pos επιστρέφει μηδέν

Length
```
function Length(S): Integer;
```
Επιστρέφει τον αριθμό των χαρακτήρων που περιέχονται στο string s.

SetLength
```
procedure SetLength(var S : String; NewLength: Integer);
```

TrimLeft
```
function TrimLeft(S: string): string;
```
Επιστρέφει ένα αντίγραφο του string S δίχως κενά ή χαρακτήρες ελέγχου στην αρχή του.

TrimRight
```
function TrimRight(S: string): string;
```
Επιστρέφει ένα αντίγραφο του string S δίχως κενά ή χαρακτήρες ελέγχου στο τέλος του.

Trim
```
function Trim(S: string): string;
```
Αφαιρεί τα κενά και τους χαρακτήρες ελέγχου από την αρχή και το τέλος.

CompareText
```
function CompareText(S1, S2: string): Integer;
```
Συγκρίνει τα S1 και S2 και επιστρέφει 0 εάν είναι ίδια. Εάν το S1 είναι μεγαλύτερο επιστρέφεται θετικός ακέραιος διαφορετικά αρνητικός ακέραιος. Δε γίνεται διάκριση μεταξύ πεζών και κεφαλαίων κατά τη σύγκριση. Μόνο για τους χαρακτήρες 7-bit ASCII. Είναι δηλαδή
S1 > S2 > 0
S1 < S2 < 0
S1 = S2 = 0

AnsiCompareText
```
function AnsiCompareText(S1, S2: string):Integer;
```
Όμοια με παραπάνω αλλά για χαρακτήρες 8-bit

CompareStr
```
function CompareStr(S1, S2: string): Integer;
```
Συγκρίνει το S1 με το S2 όπως η CompareText με τη διαφορά ότι γίνεται διάκριση μεταξύ πεζών και κεφαλαίων. Μόνο για τους χαρακτήρες 7-bit ASCII

AnsiCompareStr
```
function AnsiCompareStr( S1, S2: string):Integer;
```
AnsiCompareStr συγκρίνει το S1 με το S2, κάνοντας διάκριση μεταξύ πεζών και κεφαλαίων. Η σύγκριση επηρεάζεται από το τρέχων locale.
Σημείωση: Τα περισσότερα locale θεωρούν τους πεζούς χαρακτήρες να είναι μικρότεροι σε σχέση με τους κεφαλαίους. Σε αντίθεση με το ASCII όπου οι πεζοί χαρακτήρες θεωρούνται μεγαλύτεροι από τους κεφαλαίους. Ως εκ τούτου η συνάρτηση
AnsiCompareStr('a','A') επιστρέφει τιμή<0, ενώ η CompareStr('a','A') επιστρέφει τιμή>0

LastDelimiter
```
function LastDelimiter(Delimiters, S: string):Integer;
```
Επιστρέφει τη θέση της τελευταίας εμφάνισης ενός χαρακτήρα του string Delimiters εντός του string S.
Για παράδειγμα
```
MyIndex := LastDelimiter('\.:','c:\filename.ext');
```
Θέτει στο MyIndex την τιμή 12.

QuotedStr
```
function QuotedStr( S: string): string;
```
Τοποθετεί τον χαρακτήρα (') στην αρχή και στο τέλος του string και κάθε εμφάνιση του συγκεκριμένου χαρακτήρα εντός του string γίνεται διπλή.

Copy
```
function Copy(S: String; Index, Count: Integer):String;
```
Επιστρέφει ένα string που περιέχει Count χαρακτήρες ξεκινώντας από τη θέση S[Index].
Εάν Index > μήκος S επιστρέφεται άδειο string.

LeftStr
```
function LeftStr(AText: String; ACount: Integer):String;
```
Επιστρέφει τους πρώτους ACount χαρακτήρες του string.

RightStr
```
function RightStr(AText: String; ACount:Integer): String;
```
Επιστρέφει τους πρώτους ACount χαρακτήρες του string.

MidStr
```
function MidStr(AText: String; AStart, ACount: Integer): string;
```
Επιστρέφει τους ACount χαρακτήρες του string ξεκινώντας από τη θέση AStart.

StringOfChar
```
function StringOfChar(Ch : String; Count :Integer) : String;
```
Επιστρέφει ένα string που περιέχει Count χαρακτήρες ch.
Για παράδειγμα
```
S := StringOfChar('A', 10);
```
Θέτει στο S τιμή ίση με 'AAAAAAAAAA'

4.1.3 Συναρτήσεις Ημερομηνίας / Ώρας

Now
```
function Now: TDateTime;
```
Επιστρέφει την τρέχουσα ημερομηνία και ώρα. Ισοδυναμεί με:Date + Time.
Το TDateTime values περιέχει milliseconds αλλά η συνάρτηση στρογγυλοποιεί την ώρα στο κοντινότερο δευτερόλεπτο.

Date
```
function Date: TDateTime;
```
Επιστρέφει την τοπική ημερομηνία. Το κομμάτι τις ώρας είναι 0 (μεσάνυχτα).

Time
```
function Time: TDateTime;
```
Επιστρέφει την τρέχουσα ώρα.

DateTimeToStr
```
function DateTimeToStr(DateTime: TDateTime): string;
```
Επιστρέφει την ημερομηνία μορφοποιημένο σύμφωνα με τη μεταβλητή ShortDateFormat ακολουθούμενο από την ώρα μορφοποιημένο όπως στη μεταβλητή LongTimeFormat. Η ώρα δεν εμφανίζεται σε περίπτωση που είναι ίση με 0.
Για να τροποποιήσετε τη μορφοποίηση αρκεί να θέσετε τιμή στις μεταβλητές ShortDateFormat και LongTimeFormat

StrToDateTime
```
function StrToDateTime(const S: string): TDateTime;
```
Μετατρέπει ένα string που περιέχει ημερομηνία και ώρα σε τύπο TDateTime
Η παράμετρος S πρέπει να είναι σύμφωνη με τις ρυθμίσεις του τρέχοντος locale (τοπικές ρυθμίσεις)

DateToStr
```
function DateToStr(Date: TDateTime): string;
```
Μετατρέπει μια ημερομηνία σε string

StrToDate
```
function StrToDate(const S: string): TDateTime;
```
Μετατρέπει ένα string σε ημερομηνία

TimeToStr
```
function TimeToStr(Time: TDateTime): string;
```
Μετατρέπει το κομμάτι της ώρας του τύπου TDateTime σε string μορφοποιημένο κατά LongTimeFormat

StrToTime
```
function StrToTime(const S: string)
```
Μετατροπή string σε ώρα

DayOfWeek
```
function DayOfWeek(Date: TDateTime): Integer;
```
Επιστρέφει την ημέρα της εβδομάδας της ημερομηνίας που είναι παράμετρος. Τιμές επιστροφής είναι από 1 έως 7 με την Κυριακή να αντιστοιχεί στον αριθμό 1.
Η συνάρτηση DayOfWeek δεν είναι συμβατή με το ISO 8601 standard, όπου ορίζεται η Δευτέρα ως $1^{η}$ μέρα της εβδομάδας.

FormatDateTime
```
function FormatDateTime(Format: string;DateTime: TDateTime): string;
```

IsLeapYear
```
function IsLeapYear(Year: Integer): Boolean;
```
Επιστρέφει True εφόσον το έτος που έχει περαστεί ως παράμετρος είναι δίσεκτο.

IncMonth
```
function IncMonth(Date: TDateTime; NumberOfMonths:Integer): TDateTime;
```
Επιστρέφει μια ημερομηνία προσαυξημένη κατά NumberOfMonths μήνες.
Η παράμετρος NumberOfMonths μπορεί να είναι αρνητική επιστρέφοντας Ν μήνες πίσω.

DecodeDate
```
procedure DecodeDate(Date: TDateTime;var Year, Month, Day: Integer);
```
Η συνάρτηση σπάει την παράμετρο Date στα επιμέρους τμήματά της ημερομηνίας επιστρέφοντας το έτος, το μήνα και την ημέρα στις μεταβλητές Year, Month, Day αντίστοιχα. Εάν το έτος είναι αρνητικό, οι επιστρεφόμενες μεταβλητές επιστρέφουν όλες μηδέν..

EncodeDate
```
function EncodeDate(Year, Month, Day: Integer):TDateTime;
```
Επιστρέφει μια τιμή τύπου TDateTime όπως αυτή οριστεί από τις παραμέτρους Year, Month και Day. Εάν οι τιμές των παραμέτρων δεν είναι εντός του σωστού διαστήματος τότε δημιουργείται σφάλμα.

DecodeTime
```
procedure DecodeTime(Time: TDateTime; var Hour, Min, Sec, MSec: Integer);
```
Σπάει την παράμετρο της ώρας (δηλ, παράμετρο Time) σε ώρες, λεπτά, δευτερόλεπτα, και χιλιοστά δευτερολέπτου επιστρέφοντας τις τιμές στις μεταβλητές Hour, Min, Sec, MSec αντίστοιχα.

EncodeTime
```
function EncodeTime(Hour, Min, Sec, MSec: Integer):TDateTime;
```
Επιστρέφει τιμή τύπου TDateTime βάσει των παραμέτρων Hour, Min, Sec και MSec. Αποδεκτές τιμές ώρας είναι ακέραιες τιμές από 0 έως 24. Εάν η ώρα είναι 24 θα πρέπει τα Min, Sec, and MSec να είναι όλα 0, αποδεκτές τιμές Min και Sec είναι ακέραιοι από 0 έως και 59, ενώ για MSec από 0 έως 999. Εαν οι τιμές των παραμέτρων δεν είναι εντός των αποδεκτών τιμών δημιουργείται σφάλμα.

4.1.4 Συναρτήσεις Αρχείων

SaveStringToFile
```
procedure SaveStringToFile(FileName,Data: String);
```
Αποθηκεύει το string data στο αρχείο με όνομα filename. Δημιουργεί το αρχείο εφόσον δεν υπάρχει διαφορετικά πανωγράφει στο υπάρχων αρχείο.

LoadStringFromFile
```
function LoadStringFromFile(FileName: string): String;
```
Επιστρέφει τα περιεχόμενα του αρχείου με όνομα filename ως ένα string. Το αρχείο ανοίγεται σε mode fmShareDenyNone. Δημιουργείται σφάλμα στην περίπτωση που δεν είναι δυνατό να ανοίξει το αρχείο.

AppendStringToFile
```
procedure AppendStringToFile(FileName, Data: String);
```
Προσθέτει τα περιεχόμενα του Data στο τέλος του αρχείου με όνομα FileName.

FileExists
```
function FileExists(FileName: string):Boolean;
```
Επιστρέφει true εφόσον υπάρχει το αρχείο με με όνομα filename διαφορετικά false

DeleteFile
```
function DeleteFile(FileName: string): Boolean;
```
Διαγράφει το αρχείο με όνομα FileName από το δίσκο. Εάν το αρχείο δε δύναται να διαγραφεί η δεν υπάρχει η συνάρτηση επιστρέφει false.

RenameFile
```
function RenameFile(OldName,
NewName:string): Boolean;
```
Επιχειρεί να μετονομάσει το αρχείο OldFile σε NewFile. Επιστρέφει True εφόσον η μετονομασία πετύχει διαφορετικά False.

ChDir
```
procedure ChDir(S: string);
```
Θέτει το τρέχων φάκελο σε S.

CreateDir
```
function CreateDir(Dir:
string): Boolean;
```
Δημιουργεί ένα νέο φάκελο βάσει του ορίσματος Dir.

RemoveDir
```
function RemoveDir(Dir:
string): Boolean;
```
Διαγράφει το φάκελο του ορίσματος Dir. Επιστρέφει True εφόσον πετύχει η διαγραφή, διαφορετικά επιστρέφει False.
Ο φάκελος πρέπει να είναι άδειος για να πετύχει η διαγραφή.

GetCurrentDir
```
function GetCurrentDir: string;
```
Επιστρέφει το πλήρες όνομα του τρέχοντος φακέλου.

SetCurrentDir
```
function SetCurrentDir(Dir:
string):Boolean;
```
Θέτει το τρέχων φάκελο.

FileSearch
```
function FileSearch(Name,
DirList: string):
```
string;
Αναζητά τους φακέλους της παραμέτρου DirList για το αρχείο με όνομα FileName. Η παράμετρος DirList είναι μια λίστα φακελων (πλήρες path), χωρισμένοι με ελληνικό ερωτηματικό ';' Εάν βρεθεί το αρχείο με το όνομα filename η συνάρτηση επιστρέφει το πλήρες path για το αρχείο αυτό, διαφορετικά ένα άδειο string.

ExtractFileDrive
```
function ExtractFileDrive(
FileName: string): string;
```
Επιστρέφει το string που περιέχει το κομμάτι όπου ορίζεται το drive σε ένα πλήρες path

ExtractFileDir
```
function ExtractFileDir(const
FileName: string):string;
```
Επιστρέφει ένα string που μπορεί να περαστεί ως παράμετρο στις συναρτήσεις CreateDir, RemoveDir και SetCurrentDir. Το string είναι άδειο εφόσον το filename δεν περιέχει πληροφορία drive και φακέλου.

ExtractFileName
```
function
ExtractFileName(FileName:
string):string;
```
Επιστρέφει ένα string με μόνο το όνομα και την επέκταση του αρχείου, δίχως πληροφορία φακέλου ή drive.

ExtractFilePath
```
function ExtractFilePath(
FileName: string):string;
```
Επιστρέφει το τμήμα που είναι το path του αρχείου με όνομα FileName

ExtractFileExt
```
function ExtractFileExt(
FileName: string):string;
```
Επιστρέφει ένα string που είναι η επέκταση του αρχείου μαζί με την τελεία. Επιστρέφει Κενό string στην περίπτωση που το αρχείο δεν έχει επέκταση

ChangeFileExt
```
function ChangeFileExt(
FileName, Ext:string): string;
```
Επιστρέφει ένα string που είναι το όνομα του αρχείου FileName αλλά με την επέκταση που είναι στην παράμετρο ext. Η συνάρτηση δε μετονομάζει το αρχείο. Η παράμετρος ext, που είναι η νέα επέκταση, πρέπει να περιλαμβάνει τον χαρακτήρα της τελείας.

4.1.5 Συναρτήσεις γραφικής διεπαφής χρήστη (GUI functions)

ShowMessage
```
procedure ShowMessage(Msg: string);
```
Εμφάνιση μηνύματος με ένα κουμπί OK.
Η παράμετρος Msg είναι το μήνυμα που θα εμφανιστεί.

InputBox
```
function InputBox(ACaption, APrompt, ADefault: string): string;
```
Εμφανίζει ένα διάλογο με πλαίσιο κειμένου.
ACaption: Τίτλος του πλαισίου κειμένου.
APrompt: Το μήνυμα
ADefault: Προεπιλεγμένη τιμή στο πλαίσιο κειμένου.
Εάν ο χρήστης πατήσει Cancel, τότε η συνάρτηση επιστρέφει την τιμή ADefault. Αν πατήσει OK επιστρέφει το κείμενο του πλαισίου κειμένου.

ErrorDlg
```
procedure ErrorDlg(Msg: string);
```
Εμφανίζει ένα πλαίσιο διαλόγου με κουμπί OK. Το πλαίσιο διαλόγου εμφανίζει ένα εικονίδιο σφάλματος.
Msg: Το μήνυμα string που εμφανίζεται στο πλαίσιο διαλόγου.

InformationDlg
```
procedure InformationDlg(Msg: string);
```
Εμφανίζει πλαίσιο διαλόγου με κουμπί OK. Το πλαίσιο διαλόγου περιέχει ένα εικονίδιο πληροφόρησης.
Msg: Το μήνυμα string που εμφανίζεται στο πλαίσιο διαλόγου.

QuestionDlg
```
function QuestionDlg(Msg: string): Boolean;
```
Εμφανίζει ένα πλαίσιο διαλόγου με εικονίδιο με ερωτηματικό και δυο κουμπιά «Ναι» και «Όχι»
Msg: Το μήνυμα string που εμφανίζεται στο πλαίσιο διαλόγου.
Επιστρέφει True στην περίπτωση που πατηθεί το κουμπί «Ναι» διαφορετικά επιστρέφει False.

OkCancelDlg
```
function OkCancelDlg(Msg: string): Boolean;
```
Εμφανίζει ένα πλαίσιο διαλόγου με εικονίδιο με ερωτηματικό και δυο κουμπιά «OK» και «Cancel»
Msg: Το μήνυμα string που εμφανίζεται στο πλαίσιο διαλόγου.
Επιστρέφει True στην περίπτωση που πατηθεί το κουμπί «Ok» διαφορετικά επιστρέφει False.

4.1.6 Καθολικές Μεταβλητές (Global Variables)

Οι καθολικές μεταβλητές είναι μεταβλητές που παραμένουν στη μνήμη της εφαρμογής ακόμη και μετά το πέρας της λήξης εκτέλεσης του στοιχείου που τη δημιούργησε. Αυτές μπορούν να χρησιμοποιηθούν είτε ως περιοχή μνήμης μεταξύ των εκτελέσεων ενός στοιχείου, είτε ως ένας εναλλακτικός τρόπος για αλληλεπίδραση διαφορετικών στοιχείων μεταξύ τους. Στις Global μεταβλητές γίνεται διάκριση μεταξύ πεζών και κεφαλαίων γραμμάτων. (Οι μεταβλητές 'x' και 'X' είναι διαφορετικές)

ReadGlobalVar
```
function ReadGlobalVar(n: String): Variant;
```
Επιστρέφει την τιμή της καθολικής μεταβλητής n. Επιστρέφει ένα κενό variant, εάν η μεταβλητή δεν μπορεί να βρεθεί.

ReadGlobalVarDef
```
function ReadGlobalVarDef(n: String; d: Variant): Variant;
```
Παρόμοια με την ReadGlobalVar με τη διαφορά ότι επιστρέφεται η τιμή d στην περίπτωση που δεν βρεθεί η μεταβλητή n.

WriteGlobalVar
```
procedure WriteGlobalVar(n: String; v: Variant);
```
Θέτει στην καθολική μεταβλητή με όνομα n την τιμή v. Εάν η μεταβλητή δεν υπάρχει τότε τη δημιουργεί.

CleanupGlobalVars
```
procedure CleanupGlobalVars;
```
Απελευθερώνει όλες τις καθολικές μεταβλητές.

4.2 Υποστήριξη ADO – Σύνδεση με βάση δεδομένων με χρήση PascalScript.

Είναι εφικτό να συνδεθεί κανείς με μια βάση δεδομένων χρησιμοποιώντας τις ενδογενείς κλάσεις ADO της εφαρμογής. Το παρακάτω παράδειγμα δείχνει πως είναι εφικτό να παρουσιαστούν δεδομένα μιας βάσης δεδομένων στο παράθυρο Simulation Messages.

```
var conn: TADOConnection;
var rs: TADODataset;
var strconn : string;
var sql: string;

strConn := 'Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\dt.mdb'+
      + ';Persist Security Info=False';
conn := TADOConnection.Create(strConn);
rs := conn.Execute('select top 10 id, username from users');
println('');
while not rs.Eof do
begin
   PrintLn(rs.FieldAsString[0] + '.) ' + rs.FieldAsString[1]);
   rs.Next;
end;

rs.free;
conn.free;
```

Κώδικας 6: Σύνδεση με βάση δεδομένων

```
TADOConnetion
TADOConnetion = class
  .Create(ConnectionString: String): String;
  .Open;
  .Close;
  .Execute(sSQL: string): TADODataset;
  .BeginTrans;
  .CommitTrans;
  .RollbackTrans;
  .ExecuteSQL;
  .GetDataset(sSQL: string): TADODataset;
  .Version: string;
  .State: integer;
  .DatasetCount: integer;
  .Free;
properties
  .ConnectionString: String;
  .CommandTimeout: integer;
```

Κώδικας 7: Διάγραμμα κλάσης TADOConnection

Για να συνδεθούμε σε μία βάση δεδομένων αρκεί να δημιουργήσουμε ένα instance του αντικειμένου TADOConnection, περνώντας σαν παράμετρο το connectionstring για σύνδεση στη βάση δεδομένων όπως δείχνει στο παράδειγμα ο κώδικας 6.

```
TDBField
TDBField = class
  .IsNull: Boolean;
  .DataType: Integer;
  .DataSize: integer;
  .FieldName: string;
properties
  .Value: Variant;
  .AsString: String;
  .AsInteger: Integer;
  .AsFloat: Float;
  .AsDateTime: DateTime;
```

Κώδικας 8: Διάγραμμα κλάσης TDBField

```
TADODataset
TADODataset = class
  .Create;
  .Open;
  .Close;
  .First: boolean;
  .Next: boolean;
  .Last: boolean;
  .Eof: boolean;
  .FieldByName(Name: string): TDBField;
  .edit;
  .insert;
  .post;
  .cancel;
  .delete;
  .RecordCount: integer;
  .FieldCount: integer;
  .GetHTMLCombo: string;
  .Free;
properties
  .SQL: String;
  .Connection: TADOConnection;
  .Fields[index]: TDBField;
  .Fieldisnull[index]: boolean;
  .FieldAsDateTime[index]: dateTime;
  .FieldAsFloat[index]: Float;
  .FieldAsString[index]: String;
  .FieldAsInteger[index]: Integer;
  .FieldAsVariant[index]: Variant;
  .CommandTimeout: integer;
```

Κώδικας 9: Διάγραμμα κλάσης TADODataset

5 Προγραμματισμός με VBScript ή JScript

Όπως έχει αναφερθεί και νωρίτερα υπάρχει η δυνατότητα προγραμματισμού της διεπαφής με χρήση του Scripting Engine της VBScript ή της JScript. Αποφασίστηκε και υποστήριξη αυτών των γλωσσών προγραμματισμού καθώς είναι ήδη ευρέως διαδεδομένες. Ο σκοπός αυτού του κεφαλαίου δεν είναι να τεκμηριώσει τη σύνταξη και τις δυνατότητες αυτών των γλωσσών καθώς υπάρχει πλήρη τεκμηρίωση της σύνταξης σε πληθώρα site στο internet όπως. Μερικά από αυτά είναι:

http://www.microsoft.com/scripting/
http://www.w3schools.com/vbscript/
http://www.devguru.com/

Η εφαρμογή υποστηρίζει πλήρως τη σύνταξη αλγορίθμου σε JScript ή VBScript όπως αυτές τεκμηριώνονται από τη Microsoft με δυνατότητα δυμιουργίας και κλήσης COM Αντικειμένων. Με χρήση COM είναι δυνατό να χρησιμοποιηθούν οι βιβλιοθήκες του Matlab. Λεπτομέρειες για επικοινωνία με το Matlab στο κεφάλαιο 8 (σελ 8-104) του:
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/apiext.pdf

Στη προγραμματιστική διεπαφή με JScript ή VBScript υπάρχουν 2 αντικείμενα τα οποία μπορούν να προσπελάσουν οι χρήστες.

Το αντικείμενο **Application**

Application.ShowMessage(msg:string)
Εμφανίζει στον τελικό χρήστη το μήνυμα msg σε ένα παράθυρο διαλόγου.

Application.println(msg:string)
Εμφανίζει στο παράθυρο simulation messages το μήνυμα msg

Το αντικείμενο **Pin**

Το αντικείμενο Pin είναι το αντικείμενο από το οποίο ο χρήστης μπορεί να θέσει ή να διαβάσει την τιμή που έχει ένα pin εισόδου ή εξόδου. Για παράδειγμα έστω ότι έχουμε ένα στοιχείο επεξεργασίας με 2 pins, ένα pin εισόδου ονόματος in και ένα pin εξόδου ονόματος out, και η λογική πίσω από το στοιχείο επεξεργασίας είναι να αυξήσει την είσοδο κατά 1. τότε η υλοποίησή του αλγορίθμου του θα ήταν:

Σε *JScript ή VBScript:*

```
Pin("out") = Pin("in") + 1
```

Προσοχή: Εάν επιλέξουμε τη JScript για την υλοποίηση του αλγορίθμου πρέπει να έχουμε υπ'όψη μας ότι γίνεται διάκριση μεταξύ πεζών και κεφαλαίων (είναι Case Sensitive) οπότε τα αντικείμενα Application και Pin θα πρέπει να ξεκινούν με κεφαλαίο χαρακτήρα.

Ο κώδικας στο παράθυρο θα περιέχει ορισμένες προκαθορισμένες μεταβλητές. Πρόκειται για κώδικας που δημιουργείται αυτόματα από την εφαρμογή. Η πρώτη μεταβλητή είναι το όνομα του στοιχείου επεξεργασίας. Οι επόμενες μεταβλητές είναι τυχόν attributes που μπορεί να έχει το στοιχείο.

Ο αλγόριθμος του στοιχείου επεξεργασίας πρέπει να πληκτρολογηθεί κάτω από τη γραμμή σχόλιο με τους αστερίσκους.

6 Παραδείγματα εξομοιώσεων

6.1 Εξομοίωση πύλης AND βήμα - βήμα.

Στο παράδειγμα αυτό θα παρουσιαστεί η διαδικασία ώστε να υλοποιήσουμε μια εξοιμοίωση μιας πύλης AND μέχρι αυτή να παρουσιάσει την τιμή 1 στην έξοδο Y. Για να πραγματοποιήσουμε την εξομοίωση θα χρειαστεί να τοποθετήσουμε από τη βιβλιοθήκη στοιχείων 2 γεννήτριες τυχαίων δυαδικών αριθμών και μία πύλη AND. Επιλέγουμε το παράθυρο Data Path. Από το μενού επιλέγουμε

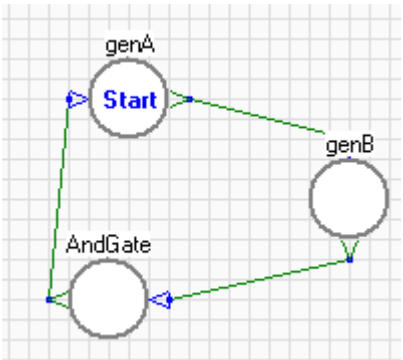
Element > Insert > Input Components > Boolean Generator

Εκτελούμε ξανά την ίδια διαδικασία έτσι ώστε να υπάρχουν 2 στοιχεία Boolean generators στο Data Path. Σέρνουμε τα στοιχεία ώστε να μην είναι το ένα πάνω στο άλλο. Από το μενού επιλέγουμε:

Element > Insert > Boolean Logic > AndGate

Επιλέγουμε το στοιχείο BooleanGenerator και στον property Inspector αλλάζουμε το όνομα από BooleanGenerator σε genA, κάνουμε δεξί κλικ πάνω του και επιλέγουμε την επιλογή Rotate Left από το μενού.
Επιλέγουμε το στοιχείο BooleanGenerator1 και στον property Inspector αλλάζουμε το όνομα από BooleanGenerator1 σε genB, κάνουμε δεξί κλικ πάνω του και επιλέγουμε την επιλογή Rotate Left από το μενού. Κάνουμε δεξί κλικ πάνω στο στοιχείο AndGate και επιλέγουμε Rotate Left. Συνδέουμε την έξοδο Y του genA με την εισοδο A του AndGate. Αντίστοιχα συνδέουμε την έξοδο Y του genB με την είσοδο B του AndGate. Στο σημείο αυτό το Data Path πρέπει να μοιάζει με αυτό στην εικόνα 7 (σελ 14)

Κάνουμε κλικ πάνω στην έξοδο Y του AndGate ώστε να το επιλέξουμε και από το Property Inspector θέτουμε τις Ιδιότητες LogOutput και MonitorOutput ίσο με true. Επαναλαμβάνουμε την ίδια διαδικασία για την έξοδο Y του genA και την έξοδο Y του genB. Εάν επιλέξουμε στο σημείο αυτό το κουμπί Graphs θα δούμε ότι έχουν δημιουργηθεί 3 κενά γραφήματα, ένα για κάθε στοιχείο επεξεργασίας που περιέχει εξόδους με την ιδιότητα MonitorOutput ίσο με True.
Στο σημείο αυτό πρέπει να ορίσουμε το Control Path. Επιλέγουμε το κουμπί control path και στο παράθυρο ροής ελέγχου εμφανίζεται ένα στοιχείο ελέγχου για κάθε στοιχείο επεξεργασίας. Κάνουμε δεξί κλικ πάνω στο στοιχείο ελέγχου AndGate και επιλέγουμε Rotate Left. Επαναλαμβάνουμε ξανά ώστε να αντιστραφεί η φορά του στοιχείου. Κάνουμε δεξί κλικ πάνω στο στοιχείο genB και επιλέγουμε Rotate Left. Συνδέουμε την έξοδο genA με την είσοδο του genB. Στο παράθυρο διαλόγου που εμφανίζεται πατούμε OK. Συνδέουμε την έξοδο genB με την είσοδο του AndGate. Πατούμε ξανά OK στο παράθυρο διαλόγου. Συνδέουμε την έξοδο του AndGate με την είσοδο του GenA. Αυτή τη φορά στο παράθυρο διαλόγου που εμφανίζεται πληκτρολογούμε την εντολή **`AndGate('Y')=0`**

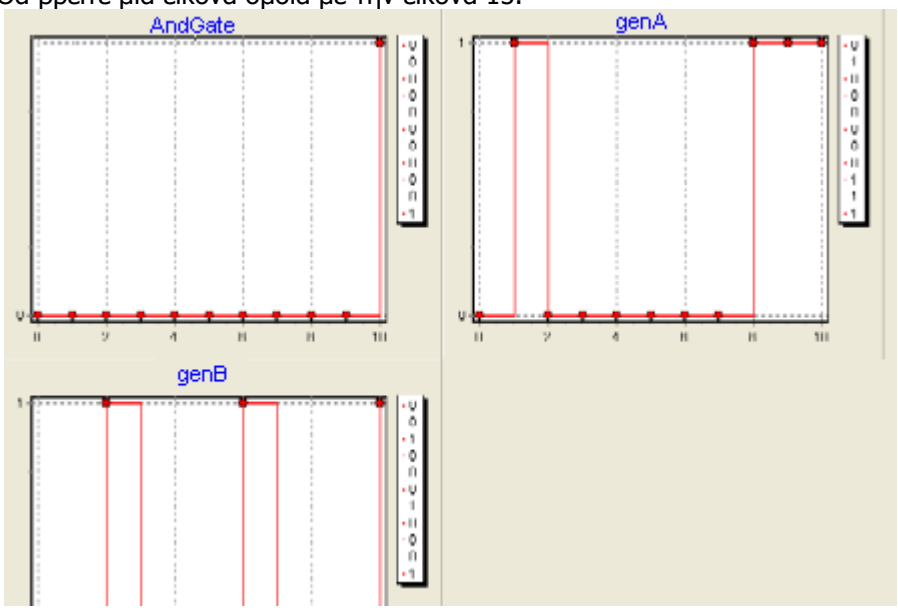


Εικόνα 14: Control Path

Όσο η έξοδος Y του στοιχείου AndGate είναι ίση με το μηδέν η παραπάνω συνθήκη θα είναι αληθής οπότε ο έλεγχος ροής του προγράμματος θα μεταφέρεται από το στοιχείο AndGate στο στοιχείο genA. Το control path σε αυτό το σημείο θα πρέπει να μοιάζει με αυτό στην εικόνα 14. Η εξομοίωση θα ξεκινήσει από το στοιχείο genA. Όταν ολοκληρωθεί η εκτέλεσή του (όταν δηλαδή στην έξοδό του οριστεί τιμή) θα εκτελεστεί το στοιχείο genB. Όταν εκτελεστεί και αυτό θα εκτελεστεί το AndGate που θα διαβάσει τις εξόδους των δυο στοιχείων θα εκτελέσει την πράξη AND και εφόσον η έξοδός του είναι ίση με 0 θα εκτελεστεί ξανά το

στοιχείο genA επαναλαμβάνοντας με αυτόν το τρόπο τη διαδικασία της εξομοίωσης.

Στο σημείο αυτό η εξομοίωση είναι έτοιμη να εκτελεστεί. Μπορούμε να πατήσουμε το πλήκτρο F9 ή να επιλέξουμε από το μενού Project > Run Simulation
Θα εμφανιστεί ένα νέο παράθυρο. Πρόκειται για τη μηχανή scripting που εκτελεί τον κώδικα.
Με την ολοκλήρωση της εκτέλεσης της εξομοίωσης είμαστε σε θέση να πραγματοποιήσουμε μια ανάλυση στα δεδομένα που αποφασίσαμε να παρακολουθήσουμε. Επιλέξτε το κουμπί Graphs. Θα βρείτε μια εικόνα όμοια με την εικόνα 15.



Εικόνα 15: Γραφήματα αποτέλεσμα εξομοίωσης

Το κείμενο που δημιουργήθηκε στο log είναι το ακόλουθο:

```
29/4/2006 12:10:26 πμ : genA('Y') = 0
29/4/2006 12:10:27 πμ : genB('Y') = 0
29/4/2006 12:10:27 πμ : AndGate('Y') = 0
29/4/2006 12:10:27 πμ : genA('Y') = 1
29/4/2006 12:10:27 πμ : genB('Y') = 0
29/4/2006 12:10:27 πμ : AndGate('Y') = 0
29/4/2006 12:10:27 πμ : genA('Y') = 0
29/4/2006 12:10:27 πμ : genB('Y') = 1
29/4/2006 12:10:27 πμ : AndGate('Y') = 0
29/4/2006 12:10:27 πμ : genA('Y') = 0
29/4/2006 12:10:27 πμ : genB('Y') = 0
29/4/2006 12:10:27 πμ : AndGate('Y') = 0
29/4/2006 12:10:27 πμ : genA('Y') = 0
29/4/2006 12:10:27 πμ : genB('Y') = 0
29/4/2006 12:10:27 πμ : AndGate('Y') = 0
29/4/2006 12:10:27 πμ : genA('Y') = 0
29/4/2006 12:10:27 πμ : genB('Y') = 0
29/4/2006 12:10:27 πμ : AndGate('Y') = 0
29/4/2006 12:10:27 πμ : genA('Y') = 0
29/4/2006 12:10:27 πμ : genB('Y') = 1
29/4/2006 12:10:27 πμ : AndGate('Y') = 0
29/4/2006 12:10:27 πμ : genA('Y') = 0
29/4/2006 12:10:28 πμ : genB('Y') = 0
29/4/2006 12:10:28 πμ : AndGate('Y') = 0
29/4/2006 12:10:28 πμ : genA('Y') = 1
29/4/2006 12:10:28 πμ : genB('Y') = 0
29/4/2006 12:10:28 πμ : AndGate('Y') = 0
29/4/2006 12:10:28 πμ : genA('Y') = 1
29/4/2006 12:10:28 πμ : genB('Y') = 0
29/4/2006 12:10:28 πμ : AndGate('Y') = 0
29/4/2006 12:10:28 πμ : genA('Y') = 1
29/4/2006 12:10:28 πμ : genB('Y') = 1
29/4/2006 12:10:28 πμ : AndGate('Y') = 1
```

Παρατηρούμε ότι κάθε τρεις καταχωρήσεις στο log αποτελούν ένα κύκλο εκτέλεσης. Όποτε το στοιχείο AndGate είχε έστω και μια είσοδο ίση με το μηδέν η έξοδός του ήταν μηδεν και συνεχίστηκε η ροή της εξομοίωσης. Όταν στις εισόδους του και οι δυο τιμές ήταν ίσες με 1 η έξοδος του AndGate έγινε ίση με 1 και σταμάτησε η διαδικασία της εξομοίωσης.

6.2 Εξομοίωση για συγκεκριμένο αριθμό εκτελέσεων.

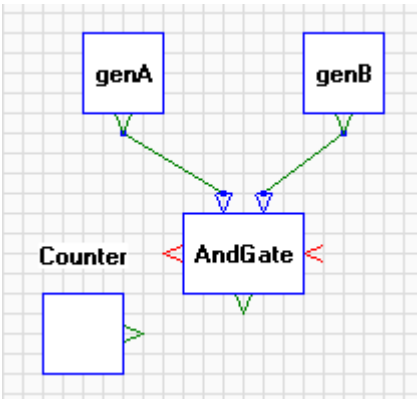
Σε μια παραλλαγή της παραπάνω άσκησης μπορεί να θελήσουμε να εκτελέσουμε την εξομοίωση για 20 κύκλους εκτέλεσης ανεξάρτητα από το αποτέλεσμα που εμφανίζει η έξοδος Y του AndGate. Συνεχίζοντας από το σημείο που είχαμε μείνει στο κεφάλαιο 6.1, πατούμε την επιλογή Data Path και επιλέγουμε από το Μενού

Element > Insert > Input Components > Counter

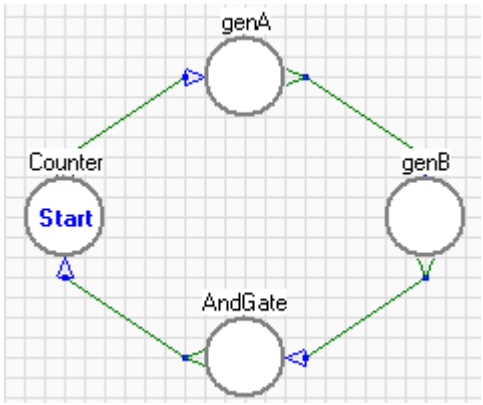
Το counter δε θα συνδεθεί με κανένα στοιχείο στο Data Path, Απλά θα χρησιμοποιήσουμε τις τιμές που παράγει στο Control Path ώστε να ελέγξουμε τη ροή εκτέλεσης. Επιλέγουμε Control Path. Παρατηρούμε ότι έχει εμφανιστεί ένα νέο στοιχείο ελέγχου που αντιστοιχεί στον Counter. Κάνουμε δεξί κλικ στη σύνδεση μεταξύ AndGate και genA και και επιλέγουμε delete ώστε να διαγραφεί η μεταξύ τους σύνδεση. Κάνουμε δεξί κλικ στο στοιχείο Counter και επιλέγουμε Rotate Right. Συνδέουμε την έξοδο του στοιχείου Counter με την είσοδο του στοιχείου genA και πατούμε OK στο πλαίσιο διαλόγου.

Συνδέουμε την έξοδο του στοιχείου AndGate στην είσοδο του στοιχείου Counter και στο πλαίσιο διαλόγου εισάγουμε ως condition: **`Counter('Y') < 20`**
Τέλος στο Control Path κάνουμε διπλό κλικ πάνω στο στοιχείο Counter ώστε η εξομοίωση να ξεκινήσει από αυτό το στοιχείο.

Επιλέγουμε Datapath. Φροντίζουμε οι έξοδοι Y των στοιχείων genA, genB και AndGate να έχουν την ιδιότητα logOutput ίση με False και θέτουμε την αντίστοιχη ιδιότητα ίση με true για την έξοδο Y του στοιχείου Counter από τον Property Inspector. Θέτουμε για την ίδια εξοδο την τιμή Monitor Output ίση με true ώστε να δούμε και το γράφημα του counter με το πέρας της εξομοίωσης. Το data path και το control path πρέπει να μοιάζουν με αυτά στις εικόνες

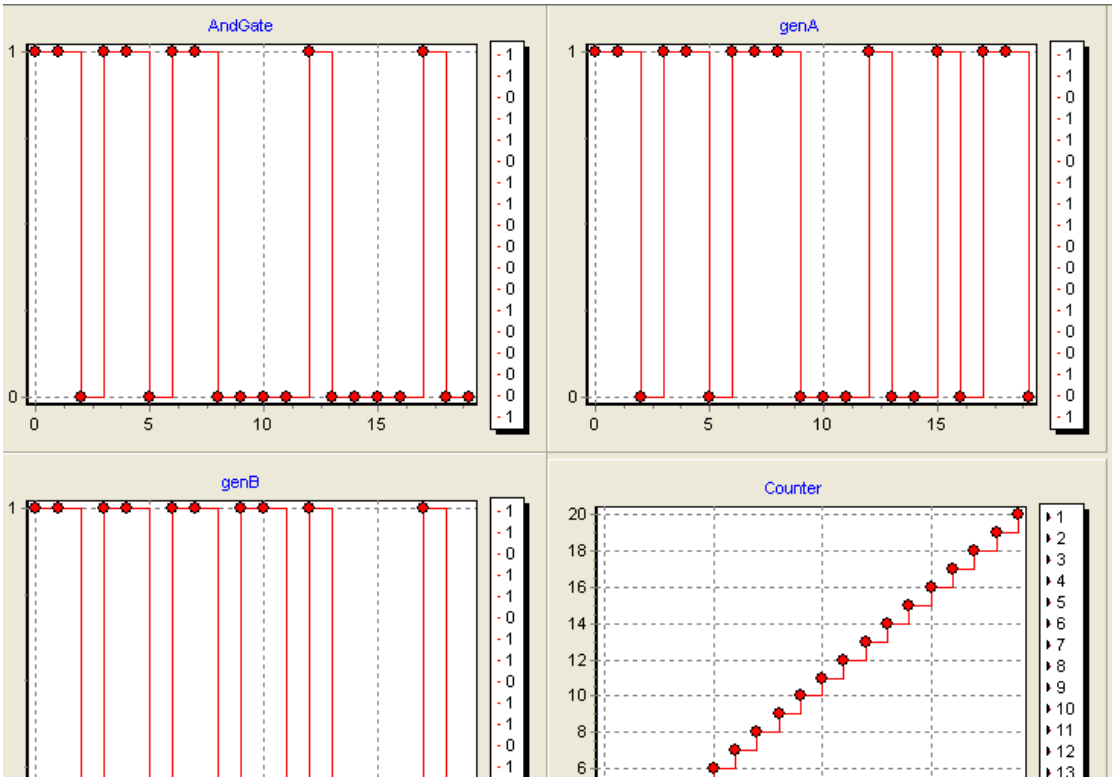


Εικόνα 16: Data path εξομοίωσης πύλης AND για πεπερασμένο αριθμό επαναλήψεων με χρήση counter



Εικόνα 17: Control path εξομοίωσης πύλης AND για πεπερασμένο αριθμό επαναλήψεων με χρήση counter

Η εξομοίωση είναι έτοιμη. Επιλέγουμε Log > Clear και στη συνέχεια
Project > Run Simulation. Με το πέρας της εξομοίωσης μπορούμε να αναλύσουμε τα δεδομένα που προέκυψαν. Το γράφημα παρουσιάζεται στην εικόνα 18 και από κάτω φαίνεται το κείμενο που δημιουργήθηκε στο Log.



Εικόνα 18: Γραφήματα εξομοίωσης πύλης AND για 20 επαναλήψεις

```
29/4/2006 1:08:30 πμ : Counter('Y') = 1
29/4/2006 1:08:31 πμ : Counter('Y') = 2
29/4/2006 1:08:31 πμ : Counter('Y') = 3
29/4/2006 1:08:31 πμ : Counter('Y') = 4
29/4/2006 1:08:32 πμ : Counter('Y') = 5
29/4/2006 1:08:32 πμ : Counter('Y') = 6
29/4/2006 1:08:32 πμ : Counter('Y') = 7
29/4/2006 1:08:32 πμ : Counter('Y') = 8
29/4/2006 1:08:33 πμ : Counter('Y') = 9
29/4/2006 1:08:33 πμ : Counter('Y') = 10
29/4/2006 1:08:33 πμ : Counter('Y') = 11
29/4/2006 1:08:34 πμ : Counter('Y') = 12
29/4/2006 1:08:34 πμ : Counter('Y') = 13
29/4/2006 1:08:34 πμ : Counter('Y') = 14
29/4/2006 1:08:34 πμ : Counter('Y') = 15
29/4/2006 1:08:35 πμ : Counter('Y') = 16
29/4/2006 1:08:35 πμ : Counter('Y') = 17
29/4/2006 1:08:35 πμ : Counter('Y') = 18
29/4/2006 1:08:36 πμ : Counter('Y') = 19
29/4/2006 1:08:36 πμ : Counter('Y') = 20
```

Αποτέλεσμα Log Αρχείου.

Από την καταχώρηση στο log αρχείο βλέπουμε ότι η διάρκεια της εξομοίωσης ήταν περίπου 6sec και επαληθεύεται ότι πραγματοποιήθηκαν 20 κύκλοι εκτέλεσης της εξομοίωσης.

6.3 Εξομοίωση Διαμόρφωσης AM

Έστω ότι θέλουμε να πραγματοποιήσουμε την AM διαμόρφωση του σήματος m που περιγράφεται από τη σχέση:

$$m(t) = A_m * \cos(f_m * t)$$

Έστω το φέρον σήμα: $c(t) = A_c * \cos(f_c * t)$

Το διαμορφωμένο σήμα δίνεται από τη σχέση: $s(t) = A_c * (1 + m(t)) * \cos(f_c * t)$

Για να εκτελέσουμε μια εξομοίωση διαμόρφωσης σήματος κατά AM τοποθετούμε στο παράθυρο Data Flow ένα στοιχείο Counter, ένα στοιχείο CosTrans και ένα στοιχείο AMModulation.

Συνδέουμε την έξοδο T του Counter με την είσοδο t του στοιχείου CosTrans και επαναλαμβάνουμε τη διαδικασία για να συνδέσουμε την έξοδο T του Counter με την είσοδο t του στοιχείου AMModulation. Επιλέγουμε το στοιχείο CosTrans και στον Property Inspector τροποποιούμε τα Attributes του σήματος.
Θέτουμε:

Frequency = 10
Amplitude = 1

Το σήμα που θα προκύψει από το στοιχείο CosTrans είναι το σήμα που θέλουμε να διαμορφώσουμε, συνεπώς το φέρων σήμα πρέπει να έχει μεγαλύτερη συχνότητα από το σήμα που θέλουμε να διαμορφώσουμε

Property Inpector:

Element: AMModulation	
BackgroundColor	clWhite
ExecutionColor	clGreen
ExecutionHost	127.0.0.1
ExecutionMode	cmLocal
Language	PascalScript
Left	110
Name	AMModulation
ShowCaptions	True
ShowGraph	False
ShowLabels	False
Top	220
Attributes	
CarrierFreq	100
CarrierAmp	2
B	1.0

Εικόνα 19: Ιδιότητες στοιχείου AM Modulation όπως φαίνονται στον Property Inspector

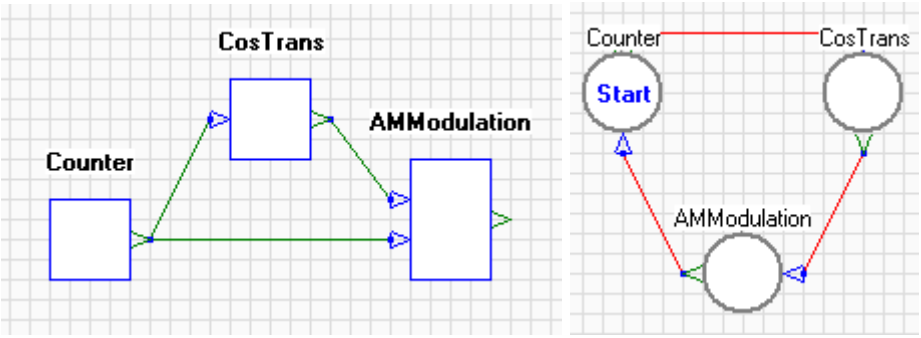
Θέτουμε στα Attributes του στοιχείου AMModulation τις ακόλουθες τιμές:

CarrierFreq = 100
CarrierAmp = 1

Η έξοδος T του counter είναι τα χρονικά σημεία στα οποία θα γίνει δειγματοληψία των σημάτων CosTrans και του φέροντος στο στοιχείο AMModulation. Σύμφωνα με το θεώρημα του Nyquist η συχνότητα δειγματοληψίας πρέπει να είναι τουλάχιστον το διπλάσιο της μέγιστης συχνότητας που επιθυμούμε να δειγματοληπτήσουμε.

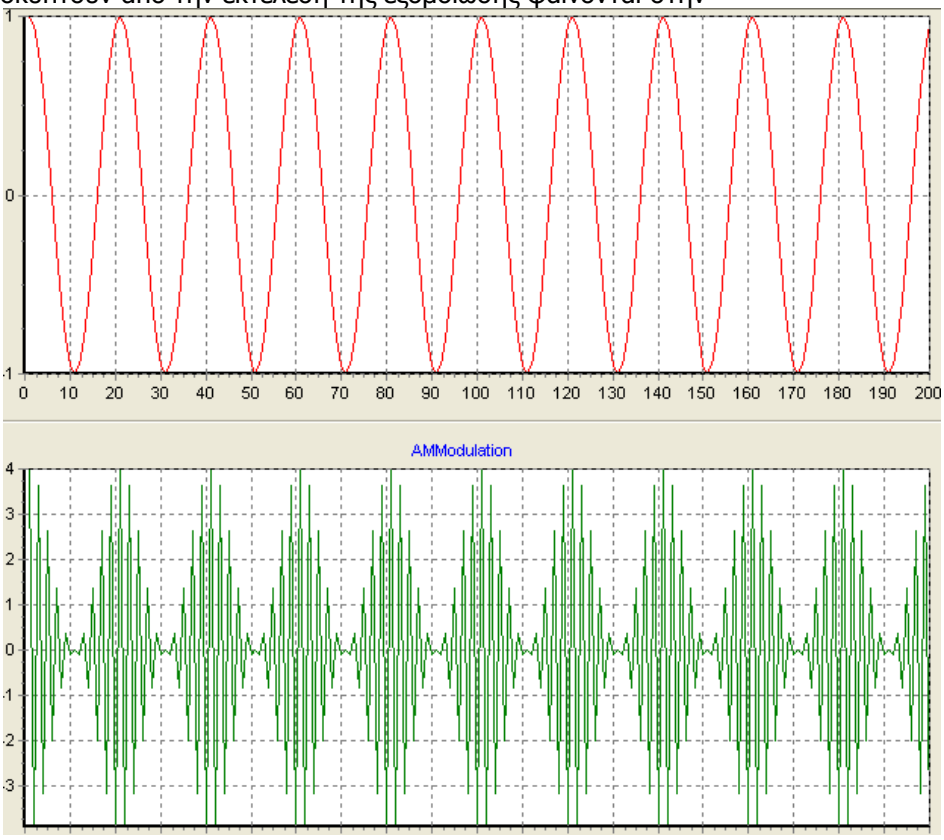
Συνεπώς για να δειγματοληπτήσουμε επαρκώς το φέρων σήμα των 100Hz πρέπει να δειγματοληπτούμε με συχνότητα τουλάχιστον 200Hz δηλ ένα δείγμα ανά (1/200)sec = 0.005. Συνεπώς στο στοιχείο counter θέτουμε στο Attribute step την τιμή 0.005.

Στο **Control path** συνδέουμε το στοιχείο Counter με το στοιχείο CosTrans και θέτουμε ως συνθήκη `Counter('T') < 1`
Η συνθήκη αυτή μπορεί να μεταφραστεί ως εκτέλεσε την εξομοίωση για ένα δευτερόλεπτο. Συνδέουμε το στοιχείο CosTrans με το στοιχείο AMModulation και πατούμε OK στο παράθυρο διαλόγου. Ομοίως συνδέουμε το AMModulation με το στοιχείο Counter. Τα γραφήματα του Data Path και του Control Path πρέπει να μοιάζουν με αυτά στην εικόνα 20.



Εικόνα 20: Data Path και Control Path για εκτέλεση εξομοίωσης διαμόρφωσης κατά ΑΜ

Επιλέγουμε να κάνουμε MonitorOutput τις εξόδους Y των στοιχείων CosTrans και AMModulation. Στις επιλογές των γραφημάτων φροντίζουμε ώστε το γράφημα να είναι τύπου γραμμής δίχως επισημάνσεις. Είμαστε σε θέση να εκτελέσουμε την εξομοίωση. Πατούμε F9 και κοιτούμε τα γραφήματα να γεμίζουν ενώ πραγματοποιείται η εξομοίωση. Τα γραφήματα που προκύπτουν από την εκτέλεση της εξομοίωσης φαίνονται στην



Εικόνα 21: Γράφημα διαμόρφωσης σήματος κατά ΑΜ με δείκτη διαμόρφωσης 1

7 Εσωτερική ανάλυση στοιχείων εφαρμογής

7.1 Κατάλογος αρχείων

Η εφαρμογή αποτελείται από τα ακόλουθα αρχεία:

- **Scriptexec.exe**: Πρόκειται για το Scripting Engine της εφαρμογής, που υποστηρίζει scripting σε τρεις γλώσσες προγραμματισμού. Αποτελεί COM αντικείμενο και μπορεί να χρησιμοποιηθεί και από εφαρμογές τρίτων.

- **abamos.exe**: Είναι το κύριο εκτελέσιμο αρχείο της εφαρμογής εξομοίωσης μαθηματικών μοντέλων (AbAMoS = Abstract Algorithm Model Simulator)

- **Component Library**: Πρόκειται για μια συλλογή από αρχεία στοιχείων επεξεργασίας οργανωμένα σε φακέλους, ανάλογα με τη θεματική ενότητα στην οποία ανήκει το στοιχείο επεξεργασίας.

7.2 Εγκατάσταση Εφαρμογής

Το μόνο βήμα που απαιτείται για την εγκατάσταση της εφαρμογής είναι η αντιγραφή των απαραίτητων αρχείων στον προοριζόμενο υπολογιστή και η εκχώρηση του COM αντικειμένου της scripting γλώσσας στο μητρώο συστήματος.

Για να πραγματοποιηθεί η εκχώριση του COM αντικειμένου εκτελούμε από το Command line την ακόλουθη εντολή

```
Scriptexec /regserver
```

Η κατανεμημένη εξομοίωση προϋποθέτει ότι το αρχείο που περιέχει το scripting engine (scriptexec.exe) είναι εγκατεστημένο στον απομακρυσμένο υπολογιστή.

7.3 Τύποι & Format αρχείων αποθήκευσης

Υπάρχουν τρεις τύποι αρχείων της εφαρμογής. Πρόκειται για τα αρχεία τύπου

- Processing Elements (*.pel): Σε αυτά τα αρχεία αποθηκεύεται ένα στοιχείο επεξεργασίας με τον κώδικά του και πληροφορίες σχετικές με τα Pin του.
- Component Library Files: Αποθηκεύονται στο φάκελο ονομασμένο με την κατηγορία του στοιχείου με κατάληξη .xml. Επίσης αποθηκεύεται ένα εικονίδιο που αντιστοιχεί στο στοιχείο επεξεργασίας με το ίδιο όνομα και κατάληξη .bmp
- Simulation Project Files (*.spr): Σε αυτά αποθηκεύεται ένα ολόκληρο project με συνδέσεις του datapath και Control path καθώς και τους κώδικες όλων των στοιχείων του project.

Και οι τρεις παραπάνω τύποι αρχείων είναι δομημένα ως XML αρχεία.

7.3.1 Αρχεία τύπου Processing Elements (*.pel)

```
<component>
  <code>Y := A and B;</code>
    <name>AndGate</name>
    <poTop />
    <poLeft>
        <pin>
          <datatype>1</datatype>
          <flowcontrol>0</flowcontrol>
          <pintype>0</pintype>
          <pinname>A</pinname>
        </pin>
        <pin>
          <datatype>1</datatype>
          <flowcontrol>0</flowcontrol>
          <pintype>0</pintype>
          <pinname>B</pinname>
        </pin>
    </poLeft>
    <poBottom />
    <poRight>
        <pin>
          <datatype>1</datatype>
          <flowcontrol>1</flowcontrol>
          <pintype>0</pintype>
          <pinname>Y</pinname>
        </pin>
    </poRight>
</component>
```

Τα αρχεία Processing Elements και τα αρχεία που μπαίνουν στο Component Library έχουν την ίδια ακριβώς δομή αλλά με διαφορετική κατάληξη. Η εφαρμογή δίνει τη δυνατότητα αποθήκευσης στοιχείων ξεχωριστά σε αρχεία .pel. Για να αποθηκεύσουμε ένα αρχείο .pel θα πρέπει να επιλέξουμε ένα στοιχείο επεξεργασίας και από το μενού επιλέγουμε

Element > Save As…

Για να φορτώσουμε ένα αποθηκευμένο στοιχείο επεξεργασίας επιλέγουμε από το μενού

Element > Load From File…

Τα αρχεία τύπου .pel έχουν xml δομή όπως φαίνεται στο διπλανό απόσπασμα αρχείου.

7.3.2 Αρχεία τύπου Simulation Project (*.spr)

Για να αποθηκεύσουμε ένα τέτοιο αρχείο επιλέγουμε από το μενού

Project > Save As…

και εισάγουμε το όνομα του αρχείου στο παράθυρο διαλόγου.

Για να φορτώσουμε ένα αποθηκευμένο αρχείο επιλέγουμε από το μενού

Project > Open…

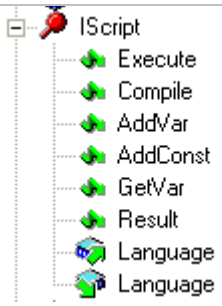
Και επιλέγουμε από το παράθυρο διαλόγου το αρχείο .spr που θέλουμε να ανοίξει η εφαρμογή.

8 Το Αντικείμενο COM του Scripting Engine

Το αντικείμενο COM του Scripting Engine βρίσκεται εντός του αρχείου ScriptExec.exe και η κύρια διεπαφή (Interface) είναι το IScript. Υπάρχουν δυο κλάσεις που υλοποιούν τις μεθόδους του IScript:

- DWSScript: Πρόκειται για την υλοποίηση του IScript σε pascal (η ονομασία προήλθε από τη χρησιμοποιούμενη βιβλιοθήκη Delphi Web Script, www.sf.org/projects/dws/)
- MSScript: Υλοποίηση του IScript χρησιμοποιώντας το scripting Engine της Microsoft.

8.1 Μεθόδοι και ιδιότητες της διεπαφής IScript



Εικόνα 22: Μεθόδοι της διεπαφής IScript

Οι μεθόδοι του IScript φαίνονται στην εικόνα 22. Αναλυτικά η διεπαφή έχει τις ακόλουθες μεθόδους και ιδιότητες:

Compile*(Code: String)*
Ελέγχει εάν ο κώδικας είναι δίχως συντακτικά λάθη και δημιουργεί σφάλμα στην περίπτωση λάθους.

AddVar*(Varname: String; Value: Olevariant)*
Ορίζει μια νέα μεταβλητή με όνομα Varname και αρχική τιμή Value

AddConst*(Constname: string; Value: Olevariant)*
Ορίζει μια νέα σταθερά με όνομα Constname και τιμή Value

GetVar*(Varname: string): Olevariant*
Επιστρέφει την τιμή της μεταβλητής με όνομα VarName. Η συνάρτηση αυτή χρησιμοποιείται μετά την εντολή execute. Εάν το script έχει τροποποιήσει την τιμή κάποιας μεταβλητής τότε αυτή μπορεί να ανακτηθεί με την παραπάνω συνάρτηση

Execute
Εκτελεί τον κώδικα που έχει προηγουμένως περαστεί στην παράμετερο Compile.

Result:string
Η συνάρτηση επιστρέφει τυχόν μηνύματα που έχουν επιστραφεί από το script μέσω της συνάρτησης println για PascalScript ή Application.println σε JScript/VBscript

Language
Ιδιότητα που επιστρέφει τη γλώσσα του Scripting Engine.
Η κλάση DWSScript επιστρέφει πάντα PascalScript δίχως να είναι εφικτό απόδοση τιμής στην ιδιότητα αυτή.
Στη κλάση MSScript μπορεί να τεθεί τιμή ίση με μια από τις εγκατεστημένες γλώσσες που υποστηρίζονται από το Microsoft Scripting Engine. Η Microsoft υποστηρίζει scripting με JScript και VBScript ωστόσο υπάρχουν και τρίτοι που έχουν δημιουργήσει

8.2 Κλήση του Com αντικειμένου

Το αντικείμενο COM μπορεί να κληθεί χρησιμόποιώντας μια από τις δυο υλοποιημένες κλάσεις που υπάρχουν μέσα σε αυτό. Παρακάτω φαίνονται ενδεικτικά παραδείγματα κλήσης του αντικειμένου σε VBScript μέσω του Windows Scripting Host.

```
Dim myscript
set myscript = CreateObject("ABAMOSLIB.MSSCRIPT")
myscript.language = "JScript" 'Μπορούσε να είναι VBScript
code = "Application.prinln('Hello ' + Pin('name')+ '…');"
myscript.AddVar('name','World');
myscript.Compile(code)
myscript.Execute
WScript.echo(myscript.result);
```

Η εκτέλεση του παραπάνω κώδικα σε ένα αρχείο .vbs των windows θα δημιουργήσει ένα παράθυρο διαλόγου που λέει 'Hello World…'. Η ουσία πίσω από το παραπάνω απόσπασμα

κώδικα είναι ο τρόπος με τον οποίο μπορούμε να περάσουμε και να ανακτήσουμε μεταβλητές σε ένα 3ο script που μπορεί να φιλοξενηθεί σε custom εφαρμογή.

Ο αντίστοιχος κώδικας για την κλήση της διεπαφής PascalScript θα ήταν:

```
Dim myscript
set myscript = CreateObject("ABAMOSLIB.DWSSCRIPT")
code = "prinln('Hello ' + name + '…');"
myscript.AddVar('name','World');
myscript.Compile(code)
myscript.Execute;
WScript.echo(myscript.result);
```

9 Ανάπτυξη της εφαρμογής

Η εφαρμογή έχει δημιουργηθεί σε Delphi χρησιμοποιώντας βιβλιοθήκες ανοιχτού κώδικα.
Για να μπορέσει να γίνει Compile ή εφαρμογή χρειάζονται να είναι εγκατεστημένα στη Delphi οι ακόλουθες βιβλιοθήκες:

- Delphi Web Script. (http://www.sf.net/projects/dws/) Περιέχει το scripting engine για pascal και τις προχωρημένες λειτουργίες του IDE για pascal
- Γίνεται χρήση του component TJVInspector της βιβλιοθήκης JVCL που μπορεί να βρεθεί στο http://jvcl.sf.net
- Το IDE προσφέρει Syntax Highlighting χάρη στο component SynEdit που μπορεί να βρεθεί στο http://synedit.sf.net
- Ο σχεδιασμός των αντικειμένων στο DataPath και στο Control Path πραγματοποιείται κάνοντας χρήση του αντικειμένου TImage32 που μπορεί να βρεθεί στο: http://graphics32.sf.net

Η ίδια εφαρμογή, ως εφαρμογή ανοικτού κώδικα, δε θα μπορούσε να λείπει από το Sourceforge. Για το λόγο αυτό δημιουργήθηκε project στο sourceforge για την εφαρμογή με τίτλο AAMS, και ένας διαδικτυακός τόπος που τη συνοδεύει http://aams.sf.net

10 Μελλοντική ανάπτυξη – Συμπεράσματα

10.1 Μελλοντική ανάπτυξη

Υπάρχουν ακόμη πολλές δυνατότητες που θα μπορούσαν να υλοποιηθούν στην εφαρμογή ώστε να την κάνουν πληρέστερη. Μερικές από τις πιο σημαντικές δυνατότητες είναι η δημιουργία Compound Elements, η υποστήριξη Arrays από τους ακροδέκτες (Pins), Η υποστήριξη χρήσης φανταστικών αριθμών από τα Scripting Engines καθώς και event based control logic.

10.1.1 Compound Elements

Ένα σύνολο από στοιχεία επεξεργασίας θα μπορούσαν όλα μαζί να απεικονίζονται ως ένα Compound Element. Η δυνατότητα αυτή υποστηρίζεται εν μέρει από την εφαρμογή αλλά δεν ολοκληρώθηκε. Στην εφαρμογή μπορεί να γίνει επιλογή πολλών elements ταυτόχρονα και κάνοντας δεξί κλικ πάνω σε ένα από αυτά επιλέγουμε από το μενού Create Execution Group και δημιουργείται ένα Execution Group. Το execution group απλά πραγματοποιεί μια νοερή ομαδοποίηση των στοιχείων μέσα σε ένα πλαίσιο. Θα μπορούσε το πλαίσιο αυτό να απεικονίζεται ως ένα στοιχείο επεξεργασίας με εισόδους και εξόδους όσα από τα pins έχουν την ιδιότητα Global σε True. Το compound Element θα έπρεπε επίσης να περιλαμβάνει το Control Path των στοιχείων που περιέχει.

10.1.2 Υποστήριξη Arrays από τα pins

Η υποστήριξη arrays από τα pins θα έδινε τη δυνατότητα στα στοιχεία επεξεργασίας να χειρίζονται με ένα κύκλο εκτέλεσης πολλά δεδομένα και αυτά να οπτικοποιούνται με γραφήματα εφόσον είναι monitored.

10.1.3 Υποστήριξη φανταστικών αριθμών

Η ενδογενής υποστήριξη των φανταστικών αριθμών θα αύξανε σημαντικά το πεδίο εφαρμογών του προγράμματος. Υπάρχει η δυνατότητα υποστήριξης πραγματικών αριθμών αλλά πρέπει ο χρήστης να δημιουργήσει σε script τους δικούς του τύπους δεδομένων ώστε να τους υποστηρίξει. Για παράδειγμα ένας πραγματικός αριθμός μπορεί να αναπαραστεί ως μια κλάση με δυο ιδιότητες, RealPart και ImaginaryPart.

10.1.4 Event Based Control Logic

Τα στοιχεία επεξεργασίας θα μπορούσαν να είναι γεννήτριες η καταναλωτές συμβάντων. Στο control path, ανάλογα το συμβάν που έχει δημιουργηθεί θα μπορούσε να δίνεται έλεγχος εκτέλεσης κάποιου στοιχείου επεξεργασίας. Αυτού του είδους η λογική κάνει εφικτό την εξομοίωση εξυπηρέτησης πελατών από ένα ταμία σε μια τράπεζα. Θα μπορούσαν να ορίζονται εκτός από αντικείμενα (π.χ. ταμίας, πελάτης, ουρά) και συμβάντα, (π.χ. Είσοδος πελάτη, Πέρας εξυπηρέτησης πελάτη). Το αποτέλεσμα μιας τέτοιας εξομοίωσης θα ήταν χαρακτηριστικά μεγέθη κάποιον αντικειμένων (π.χ. χρόνος αναμονής στην ουρά).

10.2 Συμπεράσματα

Αποτέλεσμα αυτής της πτυχιακής εργασίας ήταν η δημιουργία ενός λογισμικού που θα μπορούσε να χρησιμοποιηθεί στην εκπαιδευτική διαδικασία χάρη στις δυνατότητες που έχει για την εκτέλεση εξομοιώσεων όπως αυτές περιγράφηκαν μέσα από παραδείγματα και τις αναφορές σε προηγούμενα κεφάλαια. Ένα ακόμη χρήσιμο εργαλείο που προέκυψε από την εφαρμογή είναι η δημιουργία του scripting αντικειμένου com που μπορεί να χρησιμοποιηθεί από προγραμματιστές για προσθήκη δυνατότητας scripting στις εφαρμογές τους.

11 Βιβλιογραφία

11.1 Βιβλιογραφία

Η βιβλιογραφία περιλαμβάνει αναφορές κυρίως σε ιστοσελίδες που δώσαν κάποιες ιδέες για τη δημιουργία της πτυχιακής είτε περιέχουν περιεχόμενο σχετικό με το αντικείμενο και θα μπορούσαν να αποτελέσουν σημεία επιπρόσθετης μελέτης.

Simulation Modeling Handbook – A practical Approach
Christofer A. Chung., CRC Press

Ορισμός μαθηματικών μοντέλων. Πανεπιστήμιο Κολοράντο ΗΠΑ
http://snobear.colorado.edu/Markw/SnowHydro/Modeling/model_def.html

Συλλογή από εφαρμογές εξομοιωτών και προγραμματιστικά εργαλεία για πραγματοποίηση εξομοιώσεων
http://www.idsia.ch/~andrea/simtools.html

Μεγάλο πλήθος μαθηματικών συναρτήσεων για χρήση σε Pascal
http://www-rab.larc.nasa.gov/nmp/nmpCode.htm

Εξήγηση διαμόρφωσης AM
http://en.wikipedia.org/wiki/Amplitude_modulation

Scripting σε VBScript ή JScript
http://www.microsoft.com/scripting

Επικοινωνία της εφαρμογής με το Matlab μέσω COM (σελ 8-104)
http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/apiext.pdf

Μεγάλη συλλογή από sites και άρθρα σχετικές με την έννοια της εξομοίωσης:
http://home.ubalt.edu/ntsbarsh/Business-stat/RefSim.htm

12 Παραρτήματα

12.1 Ευρετήριο εικόνων

12.2 Ευρετήριο δειγμάτων κώδικα