ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΣΕΡΡΩΝ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

Τμήμα Πληροφορικής και Επικοινωνιών

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη και υλοποίηση εκπαιδευτικού λογισμικού για το γνωστικό αντικείμενο «Αριθμητικές Μέθοδοι σε Προγραμματιστικό Περιβάλλον» για λειτουργικό σύστημα Mac OS X

Παρασχούδης Συμεών

ΑΕΜ 1808

Επιβλέπων: Δρ. Δημήτριος Βαρσάμης

Καθηγητής Εφαρμογών

Σέρρες, 2013

ΠΕΡΙΕΧΟΜΕΝΑ

Περίληψη

Σκοπός της εργασίας αυτής ήταν να υλοποιήσουμε τις αριθμητικές μεθόδους σε περιβάλλον Mac OS X. Έτσι, δεν ασχοληθήκαμε τόσο με την δημιουργία των αλγορίθμων, αλλά επικεντρωθήκαμε στη μεταφορά των αλγορίθμων αυτών στη γλώσσα προγραμματισμού Objective-C αλλά και τη χρήση του Cocoa μαζί με το CorePlot Framework για να συνθέσουμε την εφαρμογή αυτήν.

Στο κεφάλαιο 1 κάνουμε μια ιστορική αναδρομή για την εταιρεία Apple.
Στο κεφάλαιο 2 περιγράφουμε το λειτουργικό σύστημα OS X και στο κεφάλαιο 3 κάνουμε μια εισαγωγή στην Objective-C.
Πρόσθετα, στο κεφάλαιο 4 δείχνουμε πως μπορείτε να χρησιμοποιήσετε την εφαρμογή και τέλος στο κεφάλαιο 5 εξηγούμε κάποια βασικά στοιχεία για την εφαρμογή.

1. Η εταιρεία Apple [1]

Η Apple Inc. (μέχρι τις 9 Ιανουαρίου 2007 Apple Computer Inc.) γνωστή και απλώς ως Apple είναι αμερικάνικη ιδιωτική εταιρεία τεχνολογίας υπολογιστών με εξάπλωση σε πάνω από 20 ακόμη χώρες. Εδρεύει στο Κουπερτίνο της Καλιφόρνια και τα πιο γνωστά από τα προϊόντα της είναι οι υπολογιστές της σειράς Macintosh, το λειτουργικό σύστημα Mac OS X, το φορητό jukebox iPod και τη πολυσυσκευή κινητό τηλέφωνο iPhone.

1.1 Ιστορία

Η Apple ιδρύθηκε από τον Στηβ Τζομπς (Steve Jobs) (1955-2011) και τον Στήβεν Βόζνιακ (Steven Wozniak) το 1976 σε ένα γκαράζ στην μικρή πόλη, το Λος Άλτος της Καλιφόρνιας των ΗΠΑ. Σκοπός της ίδρυσης και πρώτο της προϊόν ήταν ο υπολογιστής Apple I, δημιούργημα του Βόζνιακ, ο οποίος έγινε ευρέως αποδεκτός ως ο πρώτος ολοκληρωμένος προσωπικός υπολογιστής του κόσμου.
Έναν χρόνο αργότερα, τον Απρίλιο του 1977, η Apple ανακοίνωσε τον Apple II, τον διάδοχο του Apple I, ο οποίος για πολλά έτη παρέμεινε βασικός παράγοντας της οικονομικής ευημερίας της εταιρίας. Ο Apple II κατέκτησε εκατομμύρια χρηστών που μέχρι τότε δεν είχαν πρόσβαση σε ηλεκτρονικούς υπολογιστές με πρωτοποριακά για την εποχή προγράμματα όπως το VisiCalc του Νταν Μπρίκλιν (Dan Bricklin), το πρώτο πρόγραμμα υπολογιστικού φύλλου (spreadsheet), το οποίο έγινε και λόγος αγοράς του εν λόγω υπολογιστή.

Το 1983 η Apple παρουσίασε τον υπολογιστή Lisa, τον πρώτο εμπορικό υπολογιστή με γραφικό περιβάλλον εργασίας, του οποίου η τιμή πώλησης πλησίαζε τα $10.000 δολάρια, κάνοντάς τον ιδιαίτερα ακριβό ακόμη και για επιχειρηματικά περιβάλλοντα. Ένα χρόνο αργότερα, τον Ιανουάριο του 1984, και με τον Lisa να έχει αποτύχει εμπορικά, η Apple παρουσίασε τον Macintosh (και μετέπειτα σειρά φορητών και επιτραπέζιων υπολογιστών καθώς και

διακομιστών), τον υπολογιστή που θα έκανε την Apple γνωστή στο ευρύ κοινό και θα επαναπροσδιόριζε την διεπαφή ανθρώπου και μηχανής, εξασφαλίζοντας της σημαντική θέση στον τομέα των προσωπικών υπολογιστών μέχρι και τα μέσα της δεκαετίας του 1990. Το 1985 το διοικητικό συμβούλιο της Apple αποφάσισε να 'εξορίσει' τον συνιδρυτή της εταιρίας Steve Jobs, ο οποίος και αποχώρησε από αυτή και ίδρυσε την εταιρία NeXT Computer Inc.

Μεταξύ 1985 και 1992 η Apple κατείχε κυρίαρχη θέση στην αγορά προσωπικών υπολογιστών με γραφικό περιβάλλον. Οι υπολογιστές Macintosh αποτελούσαν μηχανήματα υψηλής ποιότητος και τιμής, και ιδιαίτερης ευκολίας χρήσης. Το 1992 η Apple μεταπήδησε στη πλατφόρμα επεξεργαστών PowerPC σε συνεργασία με τις εταιρίες Motorola και IBM. Η συνεχιζόμενη ανάπτυξη του κέλυφους (αρχικά) και λειτουργικού συστήματος (αργότερα) Windows από την Microsoft, καθώς και τα εσωτερικά προβλήματα της εταιρίας, τόσο τεχνολογικά και λειτουργικά (παρωχημένο λειτουργικό σύστημα, έλλειψη στρατηγικών στόχων, αυξημένος ανταγωνισμός και απώλεια εσόδων) όσο και διοικητικά/πολιτικά (διαφωνίες μεταξύ στελεχών, αδυναμία λήψης βασικών αποφάσεων) έφεραν την Apple στα πρόθυρα πτώχευσης το 1996.

Μετά από αρκετές διαπραγματεύσεις, τόσο με την NeXT, όσο και με την Be Inc., σχετικά με την αγορά λογισμικού ως το νέο λειτουργικό σύστημα της εταιρίας, η Apple αγόρασε την NeXT τον Φεβρουάριο του 1997. Λίγο αργότερα ο Στήβ Τζόμπς επέστρεψε στην εταιρία ως προσωρινός Διευθύνων Σύμβουλος (αργότερα μονιμοποιήθηκε).
Από το 1998 μέχρι σήμερα η Apple διανύει μια νέα περίοδο ευημερίας. Υπό τη καθοδήγηση τόσο του Στηβ Τζομπς, όσο και των ομάδων που δημιούργησε (και εν μέρει έφερε μαζί του από τη NeXT) η Apple προχώρησε σε αναδιάρθρωση τόσο των υπολογιστών της όσο και του λογισμικού της. Το 1998 παρουσίασε το iMac, ένα σχεδιαστικά πρωτότυπο μηχάνημα το οποίο της απέφερε σημαντικά έσοδα. Το 2001 η Apple παρουσίασε μια ριζικά βελτιωμένη έκδοση του λειτουργικού συστήματος για τους υπολογιστές της Macintosh, το MacOS X το οποίο βασιζόταν εν μέρει στο λογισμικό της NeXT που είχε αγοράσει μερικά

χρόνια νωρίτερα. Λίγους μήνες αργότερα, τον Οκτώβριο του 2001 ανακοίνωσε το πρώτο μη-σχετικό με Macintosh προϊόν της μετά το Apple Newton, το iPod, μια συσκευή αποθήκευσης και αναπαραγωγής μουσικής (αργότερα απέκτησε και δυνατότητα αναπαραγωγής φωτογραφιών και πρόσφατα video). Παρ'ότι δεν ήταν η πρώτη συσκευή του είδους, ο καλός σχεδιασμός της, η ευκολία χρήσης και η πετυχημένη προώθηση της εταιρίας στις ΗΠΑ και τη Δυτική Ευρώπη το κατέστησαν το δημοφιλέστερο προϊόν του είδους, κατακτώντας στην ακμή του άνω του 70% της αγοράς.

Στα μέσα του 2005 η Apple ανακοίνωσε τη πρόθεση της για χρήση επεξεργαστών Intel --- προς έκπληξη πολλών --- και την εγκατάλειψη της πλατφόρμας PowerPC με αιτιολογία την αργή πρόοδο εξέλιξης των επεξεργαστών απο την IBM. Λιγότερο από 7 μήνες αργότερα η Apple παρουσίασε τον πρώτο υπολογιστή της με επεξεργαστή Intel (Macbook Pro)· τον ακολούθησαν νεώτερες εκδόσεις για όλες τις σειρές προϊόντων υπολογιστών της εταιρίας και, τον Αύγουστο του 2006, όλοι οι υπολογιστές της Apple ήταν βασισμένοι σε επεξεργαστές της Intel, ξεκινώντας ένα νέο κεφάλαιο για την εταιρία.

Εικόνα 1-1. Διάφορες συσκευές από την Apple.

2. Το λειτουργικό σύστημα Mac OS X [2]

Το Mac OS X είναι μια σειρά γραφικών λειτουργικών συστημάτων που αναπτύσσεται, προωθείται και πωλείται από την Apple Inc. και συμπεριλαμβάνεται σε κάθε καινούριο υπολογιστή Macintosh (Mac). Το Mac OS X είναι η εξέλιξη του αρχικού Mac OS το οποίο ήταν το αρχικό λειτουργικό σύστημα της Apple την περίοδο 1984-1999. Σε αντίθεση με το αρχικό Mac OS, το Mac OS X είναι ένα UNIX λειτουργικό σύστημα το οποίο άρχισε να αναπτύσσεται στην εταιρία NeXT από τα τέλη του 1980 και μέχρι την εξαγορά της από την Apple το 1997.

Η πρώτη έκδοσή του κυκλοφόρησε το 1999 ως Mac OS X Server 1.0, ενώ η πρώτη desktop έκδοσή του, Mac OS X v10.0 "Cheetah" , ακολούθησε τον Μάρτιο του 2001. Έκτοτε, το Mac OS X έχει δεχτεί εννιά αναβαθμίσεις από τις οποίες οι πιο πρόσφατες είναι το Mac OS X v10.8 “Mountain Lion” που κυκλοφόρησε στις 25 Ιουλίου 2012, το Mac OS X v10.7 "Lion" (Λιοντάρι), που κυκλοφόρησε στις 20 Ιουλίου 2011, το Mac OS X v10.5 "Leopard" (Λεοπάρδαλη), που κυκλοφόρησε τον Οκτώβριο του 2007, ενώ η επόμενη έκδοση (10.6) με το όνομα Snow Leopard κυκλοφόρησε τον Σεπτέμβριο του 2009. Όλες οι εκδόσεις του Mac OS X φέρουν ονόματα αιλουροειδών. Το Mac OS X v10.4 αναφέρεται συνήθως ως Tiger (Τίγρης), το Mac OS X 10.3 ως Panther (Πάνθηρας), το Mac OS X 10.2 ως Jaguar (Ιαγουάρος), το Mac OS X 10.1 ως Puma (Πούμα) και το Mac OS X 10.0 ως Cheetah (Τσίτα).

Το Mac OS X δημιουργήθηκε για να λειτουργεί μόνο σε ηλεκτρονικούς υπολογιστές Macintosh, οι οποίοι κατασκευάζονται από την ίδια την Apple. Ωστόσο από το 2006 και μετά οι Macintosh χρησιμοποιούν σχεδόν το ίδιο hardware (υλικό) με αυτό των περισσότερων άλλων κατασκευαστών, δίνοντας τη δυνατότητα για μη εξουσιοδοτημένη παράνομη χρήση του Mac OS X σε μη-Apple υπολογιστές με τη χρήση hacks. Η Apple παράγει τροποποιημένες εκδόσεις του Mac OS X για χρήση σε άλλες τρεις συσκευές της, το AppleTV, το iPhone και το iPod Touch. Το τροποποιημένο λειτουργικό περιέχει θεωρητικά μόνο ό,τι χρειάζεται για τη συγκεκριμένη συσκευή, χωρίς drivers (οδηγούς) και άλλα υποσυστήματα του λειτουργικού που δεν χρειάζονται.

2.1 Ετυμολογία

Mac: Συντόμευση της λέξης Macintosh.
OS: Αρχικά γράμματα του όρου Operating System (Λειτουργικό σύστημα).
X: Το λατινικό νούμερο 10. Ιστορικά το Mac OS X είναι η εξέλιξη του Mac OS System 9.

2.2 Ιστορία

Το Mac OS X βασίζεται στον πυρήνα Mach. Συγκεκριμένα κομμάτια απο το FreeBSD και το NetBSD υιοθετήθηκαν στο Nextstep, που αποτέλεσε την βάση για το Mac OS X. Το Nextstep ήταν ένα αντικειμενοστραφές (object-oriented) λειτουργικό σύστημα αναπτυγμένο από τη εταιρία του Στηβ Τζομπς "NeXT", την οποία δημιούργησε αφού έφυγε από την Apple το 1985. Κατά την απουσία του Τζομπς, η Apple προσπάθησε να δημιουργήσει ένα "νέας-γενιάς" λειτουργικό με το Taligent και το Copland, αλλά χωρίς ιδιαίτερη επιτυχία.

Τελικά, το λειτουργικό της NeXT, που είχε πλέον μετονομαστεί σε OPENSTEP, επιλέχθηκε ως η βάση του επόμενου λειτουργικού της Apple, και η Apple εξαγόρασε την NeXT. Ο Στηβ Τζομπς επέστρεψε στην Apple ως προσωρινός CEO (διευθυντής) και πάλι, αναλαμβάνοντας το έργο της μετατροπής του φιλικού-για-προγραμματιστές Openstep σε ένα σύστημα που θα μπορούσε να χρησιμοποιήσει η βασική αγορά της Apple, δηλαδή οι οικιακοί χρήστες και οι επαγγελματίες στο δημιουργικό τομέα. Αρχικά το έργο ονομάστηκε Rhapsody (Ραψωδία) και στη συνέχεια μετονομάστηκε σε Mac OS X.
Συμβαδίζοντας με την πολιτική της Apple όπου ο υπολογιστής αποτελεί το "digital hub" (ψηφιακός κόμβος), με κάθε καινούρια έκδοση, το Mac OS X εξελίχθηκε δίνοντας λιγότερη βάση στη συμβατότητα με παλιότερο λογισμικό και περισσότερη στις εφαρμογές "digital lifestyle" (ψηφιακός τρόπος ζωής) όπως το πακέτο "iLife", επιχειρηματικές εφαρμογές όπως το "iWork" και ενσωματώνοντας το σύστημα οικιακής ψυχαγωγίας "Front Row media center". Κάθε νέα έκδοση περιείχε γενικές μετατροπές στη διασύνδεση χρήστη (interface)

όπως η brushed metal (γδαρμένο μέταλλο) εμφάνιση που προστέθηκε στην έκδοση 10.2 και η ενοποιημένη εμφάνιση στο 10.4.

2.3 Περιγραφή

Το Mac OS X είναι μια ριζική αναχώρηση από τα προηγούμενα λειτουργικά συστήματα των Macintosh. Ο θεμελιώδης κώδικας και η δομή του είναι εντελώς διαφορετική από της προηγούμενες εκδόσεις. Η βάση του, με το όνομα Darwin (Δαρβίνος) είναι ένα ελεύθερο και ανοιχτού κώδικα UNIX λειτουργικό σύστημα χτισμένο πάνω στο XNU kernel, με τις τυπικές UNIX ευκολίες διαθέσιμες μέσα από τη γραμμή εντολών (command line interface). Πάνω στο Darwin η Apple πρόσθεσε αρκετά συστατικά όπως ο "Finder" και το γραφικό περιβάλλον "Aqua", ολοκληρώνοντας το βασισμένο σε γραφικό περιβάλλον λειτουργικό σύστημα που είναι το Mac OS X.

Το Mac OS X περιλαμβάνει αρκετά χαρακτηριστικά με σκοπό να κάνουν το λειτουργικό σύστημα πιο σταθερό και αξιόπιστο. για παράδειγμα το pre-emptive multitasking και η προστασία μνήμης (memory protection) βελτίωσαν τη δυνατότητα του λειτουργικού να τρέχει πολλαπλές εφαρμογές ταυτόχρονα χωρίς κάποια εφαρμογή να επεμβαίνει ή να ρίχνει το σύστημα. Πολλές πτυχές του Mac OS X πηγάζουν από το Openstep, που σχεδιάστηκε να είναι "φορητό" - να διευκολύνει την μεταφορά του ανάμεσα σε διαφορετικές πλατφόρμες. Για παράδειγμα:

- Το Nextstep μεταφέρθηκε από το αρχικό NeXT workstation που βασιζόταν σε 68k επεξεργαστή, σε άλλες αρχιτεκτονικές πριν εξαγοραστεί η NeXT από την Apple,
- Το Openstep μεταφέρθηκε στην PowerPC αρχιτεκτονική ως μέρος του Rhapsody
- Το Mac OS X v10.4 μεταφέρθηκε στην intel αρχιτεκτονική το 2006 για τα νέα Macintosh με intel επεξεργαστή και το Mac OS X v10.5 μεταφέρθηκε στην αρχιτεκτονική ARM για την παραγωγή του iPhone και iPod Touch

Η πιο ορατή διαφορά ήταν το γραφικό περιβάλλον Aqua. Η χρήση κουμπιών σαν χρωματιστές σταγόνες, οι διαφάνειες και φωτορεαλιστικά εικονίδια έφεραν υφή και χρώμα σε σχέση με τα προηγούμενα λειτουργικά. Πολλοί χρήστες εξέφρασαν την αρνητική άποψη ότι ήταν πολύ "χαριτωμένο" χωρίς επαγγελματικό ερέθισμα. Άλλοι πίστεψαν ότι το Aqua ήταν ένα γενναίο και πρωτοποριακό βήμα σε μια εποχή που τα γραφικά περιβάλλοντα ήταν απλά βαρετά. Παρόλο το διχασμό, η εμφάνισή του ήταν άμεσα αναγνωρίσιμη, ακόμα και πριν την πρώτη έκδοσή του Mac OS X, άλλοι προγραμματιστές άρχισαν να προσπαθούν να αντιγράψουν την εμφάνιση τουAqua.

Το Mac OS X περιέχει δικό του πακέτο προγραμματισμού, με επίκεντρο το περιβάλλον προγραμματισμού το Xcode. Το Xcode παρέχει περιβάλλοντα (interfaces) και compilers που υποστηρίζουν πολλές γλώσσες προγραμματισμού όπως C, C++, Objective-C, Java, Applescipt, και πολλές άλλες με compilers τρίτων.

2.4 Συμβατότητα

Λογισμικό (Software)

Κατά τα τέλη της δεκαετίας του '90, για τη διευκόλυνση της μετατροπής των υφισταμένων εφαρμογών από το Mac OS 9 στο Mac OS X, ενσωματώθηκε στο Mac OS X το Carbon (Άνθρακας) API (Application Programming Interface - περιβάλλον προγραμματισμού εφαρμογών). Οι εφαρμογές γραμμένες με το Carbon τρέχανε κανονικά σε όλα τα συστήματα και οι εφαρμογές που είχαν σχεδιαστεί για το Mac OS 9 χρειάζονταν μόνο λίγες μετατροπές για να λειτουργούν με το Carbon API. Από την άλλη, τα πιο ισχυρά API του Mac OS X που προήλθαν από το Openstep δεν ήταν συμβατά με τις εκδόσεις που προηγήθηκαν του Mac OS X. Αυτά τα API αναφέρονται ως Cocoa (Κακάο). Αυτή η κληρονομιά είναι πολύ εμφανής στους προγραμματιστές με Cocoa, αφού τα περισσότερα Cocoa class ονόματα ξεκινούν με το συνθετικό "NS" από τη λέξη Nextstep.

Από το 2006 και μετά εγκαταλείφθηκε η Java ως το προτιμώμενο πακέτο λογισμικού. Στο Mac OS X η Java είχε πάντα ιδιαίτερη μεταχείριση. Εφαρμογές γραμμένες σε Java ενσωματώνονται όσο καλύτερα γίνεται στο σύστημα ενώ παραμένουν cross-platform (ανεξάρτητης-πλατφόρμας) και γραφικά περιβάλλοντα γραμμένα με Swing φαίνονται σχεδόν ακριβώς το ίδιο με τα κανονικά Cocoa περιβάλλοντα. Παραδοσιακά οι εφαρμογές για το Mac OS X γράφονται με Objective-C, με την Java μόνο ως εναλλακτική. Όμως στις 11 Ιουλίου 2005 η Apple ανακοίνωσε ότι "τα χαρακτηριστικά που θα προστεθούν στο Cocoa μετά το Mac OS X v10.4 δεν θα προστεθούν στο Cocoa-Java API"

Υλικό (Hardware)

Στις αρχές του Mac OS X, υποστηρίζονταν όλες οι αρχιτεκτονικές των τότε Macintosh υπολογιστών (φορητών, επιτραπέζιων και διακομιστών) που βασίζονταν σε επεξεργαστές PowerPC G3, G4 και G5. Μετέπειτα εκδόσεις διέκοψαν την υποστήριξη για το παλιότερο υλικό. Για παράδειγμα το v10.3 δεν υποστηρίζει τα "beige" (μπεζ) G3s, το v10.4 δεν υποστηρίζει τα συστήματα πριν την έναρξη χρήσης των θυρών FireWire από την Apple και το v10.5 δεν υποστηρίζει το συστήματα παλιότερα των G4 στα 867MHz, αν και κυκλοφόρησαν εργαλεία όπως το XPostFacto τρίτων που επιτρέπουν την αναβάθμιση χωρίς την υποστήριξη της Apple.

Το Mac OS X διατηρεί συμβατότητα με εφαρμογές γραμμένες για παλιότερες εκδόσεις του Mac OS παρέχοντας ένα περιβάλλον προσομοίωσης που λέγεται Classic, που επιτρέπει στους χρήστες να τρέχουν το Mac OS 9 ως κομμάτι του Mac OS X. Το Classic έπαψε να υποστηρίζεται 7 χρόνια μετά την έναρξη του Mac OS X, με την έλευση των βασισμένων σε intel Macintosh.

Εικόνα 2-1. Το λειτουργικό σύστημα Mac OS X Mountain Lion

3. Εισαγωγή στην Objective-C [3]

Η Objective-C, είναι η πρωτεύων γλώσσα η οποία χρησιμοποιείται για την συγγραφή λογισμικού Mac.

3.1 Καλώντας τις Μεθόδους (Calling Methods)

Η βασική σύνταξη για να κληθεί μια μέθοδος σε ένα αντικείμενο είναι η εξής:

```
[object method];
[object methodWithInput:input];
```

Οι μέθοδοι, μπορούν να επιστρέφουν μια τιμή:

```
output = [object methodWithOutput];
output = [object methodWithInputAndOutput:input];
```

Μπορείτε επιπλέον να καλέσετε τις μεθόδους και σε κλάσεις, όπου έτσι δημιουργείτε αντικείμενα. Στο παρακάτω παράδειγμα, καλούμε την μέθοδο *string*, σε μια κλάση τύπου NSString, η οποία επιστρέφει ένα καινούργιο NSString αντικείμενο:

```
id myObject = [NSString string];
```

Ο τύπος **id** σημαίνει ότι η μεταβλητή **myObject** μπορεί να απευθυνθεί σε οποιοδήποτε τύπου αντικειμένου, έτσι ώστε οι πραγματικές κλάσεις και μέθοδοι που υλοποιεί δεν είναι γνωστές όταν χτίζεται η εφαρμογή.

Σε αυτό το παράδειγμα, είναι προφανές ότι το αντικείμενο θα είναι τύπου **NSString**, έτσι μπορούμε να αλλάξουμε το αντικείμενο:

```
NSString* myString = [NSString string];
```

Τώρα η μεταβλητή μας είναι *NSString,* έτσι ώστε ο μεταγλωττιστής θα μας προειδοποιήσει αν προσπαθήσουμε να χρησιμοποιήσουμε μια μέθοδο σε αυτό το αντικείμενο την οποία το NSString δεν υποστηρίζει.

Προσέξτε ότι υπάρχει αστερίσκος δεξιά από τον τύπο αντικειμένου. Όλες οι μεταβλητές στην Objective-C είναι τύπου δείκτες (pointers). Ο τύπος *id* είναι προκαθορισμένος σαν ένας τύπος δείκτη, έτσι ώστε δεν χρειάζεται να βάζουμε τον αστερίσκο.

3.1.1 Ένθετα Μηνύματα (Nested Messages)

Σε πολλές γλώσσες, ένθετοι μέθοδοι ή κλήσεις συναρτήσεων μοιάζουν έτσι:

```
συνάρτηση1 ( συνάρτηση2() );
```

Το αποτέλεσμα της συνάρτησης2 μεταβιβάζεται σαν είσοδος στην συνάρτηση1. Στην Objective-C, τα ένθετα μηνύματα φαίνονται έτσι:

```
[NSString stringWithFormat:[prefs format]];
```

Αποφεύγετε να εμφωλιάζετε παραπάνω από δυο κλήσεις μηνυμάτων σε μια γραμμή, καθώς μπορεί πολύ εύκολα να γίνει δυσανάγνωστο.

3.1.2 Μέθοδοι Πολλαπλών Τιμών (Multi-Input Methods)

Κάποιοι μέθοδοι παίρνουν πολλαπλές τιμές εισόδων. Στην Objective-C, ένα όνομα μιας μεθόδου μπορεί να διαχωριστεί σε πολλά τμήματα. Η επικεφαλίδα (header) μιας μεθόδου πολλών εισόδων μοιάζει έτσι:

```
-(BOOL)writeToFile:(NSString *)path atomically:(BOOL)useAuxiliaryFile;
```

Μπορείτε να καλέσετε την συνάρτηση έτσι:

```
BOOL result = [myData writeToFile:@"/tmp/log.txt" atomically:NO];
```

Αυτά δεν είναι μόνο ονομαζόμενα ορίσματα. Το όνομα της μεθόδου είναι στην πραγματικότητα το `writeToFile:atomically:` στο σύστημα κατά την ώρα εκτέλεσης.

3.2 Μέθοδοι Ανάγνωσης (Accessors)

Όλες οι μεταβλητές αντιτύπων (instance variables) στην Objective-C είναι ιδιωτικές (private) εξ'ορισμού, γι'αυτο θα πρέπει να χρησιμοποιείτε τους accessors για να πάρετε και να ορίσετε τις τιμές τις περισσότερες φορές. Υπάρχουνε δυο συντάξεις. Η παρακάτω είναι η παραδοσιακή σύνταξη 1.x:

```
[photo setCaption:@"Day at the Beach"];
output = [photo caption];
```

Ο κώδικας στη δεύτερη γραμμή δεν διαβάζει απ'ευθείας μια μεταβλητή αντιτύπων. Ουσιαστικά καλεί μια μέθοδο ονομαζόμενη caption. Στις περισσότερες περιπτώσεις, δεν χρειάζεται να προσθέτετε το πρόθεμα "get" στις μεθόδους get στην Objective-C.

Όποτε βλέπετε κώδικα μέσα σε αγκύλες, στέλνετε ένα μήνυμα σε ένα αντικείμενο ή μια κλάση.

Σύνταξη της τελείας (Dot Syntax)

Η σύνταξη dot για τις μεθόδους εγγραφής και ανάγνωσης είναι καινούργια στην Objective-C 2.0, η οποία είναι μέρος του Mac OS X 10.5:

```
photo.caption = @"Day at the Beach";
output = photo.caption;
```

Μπορείτε να χρησιμοποιείτε οποιοδήποτε από τους δυο τρόπους, αλλά επιλέξτε μόνο έναν για κάθε project. Η σύνταξη dot θα πρέπει να χρησιμοποιείται μόνο για τις μεθόδους εγγραφής - ανάγνωσης και όχι για μεθόδους γενικού σκοπού.

3.3 Δημιουργώντας αντικείμενα

Υπάρχουν δυο κύριοι τρόποι για να δημιουργήσετε ένα αντικείμενο. Ο πρώτος είναι αυτός που είδατε προηγουμένως:

```
NSString* myString = [NSString string];
```

Αυτό είναι το περισσότερο αυτόματο βολικό στυλ. Σε αυτήν την περίπτωση, δημιουργείτε ενα **αυτοαπαλλασσόμενο** (autoreleased) αντικείμενο, το οποίο θα εξετάσουμε λεπτομερώς αργότερα. Σε πολλές περιπτώσεις, ωστόσο, πρέπει να δημιουργείτε ένα αντικείμενο χρησιμοποιώντας τον μη αυτόματο τρόπο:

```
NSString* myString = [[NSString alloc] init];
```

Αυτή είναι η εμφωλιασμένη κλήση μιας μεθόδου. Η πρώτη είναι η μέθοδος alloc η οποία καλείται από το ίδιο το NSString. Αυτή είναι μια σχετικά χαμηλού επιπέδου κλήση η οποία δεσμεύει μνήμη και συγκεκριμενοποιεί (instantiates) ενα αντικείμενο.

Το δεύτερο κομμάτι είναι μια κλήση στο **init** του καινούργιου αντικειμένου. Η εφαρμογή init κάνει συνήθως την βασική εγκατάσταση, όπως να δημιουργεί μεταβλητές αντιτύπων. Οι λεπτομέρειες αυτών είναι άγνωστες σε εσάς σαν πελάτη της κλάσης.

Σε μερικές περιπτώσεις, μπορείτε να χρησιμοποιήσετε διάφορες εκδόσεις του **init**, το οποίο παίρνει παραμέτρους:

```
NSNumber* value = [[NSNumber alloc] initWithFloat:1.0];
```

3.4 Βασική Διαχείριση Μνήμης

Αν γράφετε μια εφαρμογή για το Mac OS X, έχετε την επιλογή να ενεργοποιήσετε την συλλογή σκουπιδιών (garbage collection). Γενικά, αυτό σημαίνει ότι δεν χρειάζεται να σκέφτεστε για το πως θα διαχειρίζεστε την μνήμη έως ότου αντιμετωπίσετε περισσότερο πολύπλοκες περιπτώσεις.

Εντούτοις, μπορεί να μην δουλεύετε πάντα σε ένα περιβάλλον το οποίο υποστηρίζει την συλλογή σκουπιδιών. Σε αυτήν την περίπτωση, πρέπει να γνωρίζετε κάποιες βασικές έννοιες.

Αν δημιουργείτε ένα αντικείμενο χρησιμοποιώντας το χειροκίνητο alloc στυλ, πρέπει αργότερα να αποδεσμεύσετε το αντικείμενο. Δεν πρέπει μόνοι σας να αποδεσμεύετε ένα αυτοαπαλλασσόμενο αντικείμενο επειδή η εφαρμογή σας θα καταρρεύσει (crash) αν το κάνετε.

Εδώ είναι κάποια παραδείγματα:

```
// το string1 θα αποδεσμευτεί αυτόματα
NSString* string1 = [NSString string];

// πρέπει να αποδεσμευτεί αυτό μόλις τελειώσουμε
NSString* string2 = [[NSString alloc] init];
[string2 release];
```

3.5 Σχεδιάζοντας μια Διεπαφή Κλάσεων (Class Interface)

Η σύνταξη της Objective-C για τη δημιουργία μιας κλάσης είναι πολύ απλή. Συνήθως διατίθεται σε δυο μέρη.

Η διεπαφή κλάσης συνήθως αποθηκεύεται στο αρχείο ClassName.h, και ορίζει τις μεταβλητές αντιτύπων καθώς και τις public μεθόδους.

Η υλοποίηση (implementation) βρίσκεται στο αρχείο ClassName.m και περιέχει τον αληθινό κώδικα γι' αυτές τις μεθόδους. Επιπλέον ορίζει τις private μεθόδους, οι οποίες δεν είναι διαθέσιμες στους πελάτες της κλάσης.

Παρακάτω φαίνεται πως μοιάζει μια κλάση διεπαφών. Η κλάση ονομάζεται Photo και έτσι το αρχείο ονομάζεται Photo.m

```
#import <Cocoa/Cocoa.h>

@interface Photo : NSObject {
    NSString* caption;
    NSString* photographer;
}
@end
```

Αρχικά, εισάγουμε το αρχείο Cocoa.h, για να τραβήξουμε όλες τις βασικές κλάσεις για μια εφαρμογή Cocoa. Η οδηγία (directive) #import προφυλάσσει εναντίον της συμπερίληψης ενός μοναδικού αρχείου πολλαπλές φορές.

Το @interface δηλώνει ότι αυτός είναι ένας ορισμός τύπου κλάσης Photo. Το σημείο ':' καθορίζει την super κλάση (superclass), η οποία είναι το NSObject.
Μέσα στα άγκιστρα, υπάρχουν δυο μεταβλητές αντιτύπων: **caption** και **photographer**.

Και τα δυο είναι NSStrings, αλλά θα μπορούσανε να είναι οποιαδήποτε τύπου αντικείμενα, συμπεριλαμβάνοντας και το id.

Τέλος, το **@end** σύμβολο τελειώνει τον ορισμό της κλάσης.

3.5.1 Προσθήκη Μεθόδων

Ας προσθέσουμε κάποιες μεθόδους get για τις μεταβλητές αντιτύπων:

```
#import <Cocoa/Cocoa.h>

@interface Photo : NSObject {
    NSString* caption;
    NSString* photographer;
}

- caption;
- photographer;

@end
```

Θυμηθείτε, οι μέθοδοι στην Objective-C τυπικά παραλείπουνε το πρόθεμα “get”. Μια μονή παύλα (-) πριν από το όνομα μιας μεθόδου σημαίνει ότι είναι μια μέθοδος αντιτύπων (instance method). Αντίστοιχα ενα συν (+) σημαίνει οτι είναι μια μέθοδος κλάσης.

Εξ’ορισμού ο μεταγλωττιστής υποθέτει ότι μια μέθοδος επιστρέφει ένα αντικείμενο τύπου id και ότι όλες οι τιμές εισόδων είναι τύπου id. Ο παραπάνω κώδικας είναι τεχνικά σωστός, αλλά είναι ασυνήθιστος. Ας προσθέσουμε συγκεκριμένους τύπους για τις επιστρεφόμενες τιμές. (return values)

```
#import <Cocoa/Cocoa.h>

@interface Photo : NSObject {
    NSString* caption;
    NSString* photographer;
}

- (NSString*) caption;
- (NSString*) photographer;

@end
```

Τώρα ας προσθέσουμε τους μεθόδους εγγραφής (setters).

```
#import <Cocoa/Cocoa.h>

@interface Photo : NSObject {
    NSString* caption;
    NSString* photographer;
}
- (NSString*) caption;
- (NSString*) photographer;

- (void) setCaption: (NSString*)input;
- (void) setPhotographer: (NSString*)input;

@end
```

Οι μέθοδοι εγγραφής δεν χρειάζεται να επιστρέφουν μια τιμή, έτσι τις καθορίζουμε σαν void.

3.6 Υλοποίηση Κλάσεων (Class Implementation)

Ας δημιουργήσουμε την υλοποίηση, ξεκινώντας με τις μεθόδους ανάγνωσης:

```
#import "Photo.h"

@implementation Photo

- (NSString*) caption {
    return caption;
}

- (NSString*) photographer {
    return photographer;
}

@end
```

Αυτό το κομμάτι κώδικα ξεκινάει με το @implementation και το όνομα της κλάσης και έχει το @end ακριβώς όπως η διεπαφή. Όλες οι μέθοδοι πρέπει να εμφανίζονται μεταξύ αυτών των δύο εντολών.

Συνεχίζουμε με τους μεθόδους εγγραφής:

```
- (void) setCaption: (NSString*)input
{
    [caption autorelease];
    caption = [input retain];
}

- (void) setPhotographer: (NSString*)input
{
    [photographer autorelease];
    photographer = [input retain];
}
```

Η κάθε μέθοδος εγγραφής, ασχολείται με δύο μεταβλητές. Η πρώτη είναι μια αναφορά στο υπάρχον αντικείμενο. και η δεύτερη είναι το καινούργιο αντικείμενο εισόδου. Σε ένα περιβάλλον συλλογής σκουπιδιών (garbage collected environment), θα μπορούσαμε να ορίσουμε την καινούργια μεταβλητή απ'ευθείας:

```
- (void) setCaption: (NSString*)input {
    caption = input;
}
```

Αλλά αν δεν μπορείτε να χρησιμοποιήσετε την συλλογή σκουπιδιών, πρέπει να αποδεσμεύσετε το παλιό αντικείμενο και να διατηρήσετε το καινούργιο.

Στην πραγματικότητα υπάρχουν δυο τρόποι για να ελευθερώσετε μια αναφορά σε ένα αντικείμενο: **release** και **autorelease**. Το στάνταρ release θα ελευθερώσει την αναφορά αμέσως. Η μέθοδος autorelease θα την ελευθερώσει κάποια στιγμή στο μέλλον, αλλά σίγουρα θα παραμείνει μέχρι το τέλος της συγκεκριμένης συνάρτησης (εκτός και αν προσθέσετε κώδικα που συγκεκριμένα το αλλάζει αυτό).

Η μέθοδος autorelease είναι ασφαλέστερη σε μια μέθοδο set επειδή οι μεταβλητές για τις καινούργιες και τις παλιές τιμές μπορούν να δείχνουν στο ίδιο αντικείμενο. Δεν θα θέλατε αμέσως να απελευθερώσετε ένα αντικείμενο το οποίο πρόκειται να το διατηρήσετε.

Init

```
- (id) init
{
    if ( self = [super init] )
    {
        [self setCaption:@"Default Caption"];
        [self setPhotographer:@"Default Photographer"];
```

```
    }
    return self;
}
```

Το παραπάνω είναι αυτεπεξηγηματικό, ωστόσο ίσως η δεύτερη γραμμή φαίνεται λίγο ασυνήθιστη. Αυτό είναι ένα σημάδι ισότητας, το οποίο εκχωρεί το αποτέλεσμα του **[super init]** στο **self**.

Ουσιαστικά αυτό ρωτά την super κλάση (superclass) να κάνει την δική της αρχικοποίηση. Η συνθήκη **if** επαληθεύει ότι η αρχικοποίηση ήταν επιτυχής πριν προσπαθήσει να ορίσει τις προεπιλεγμένες τιμές.

Dealloc

Η μέθοδος **dealloc** καλείται σε ένα αντικείμενο όταν αφαιρείται από την μνήμη. Αυτή είναι συνήθως η κατάλληλη στιγμή για να απελευθερώνετε τις αναφορές:

```
- (void) dealloc
{
    [caption release];
    [photographer release];
    [super dealloc];
}
```

Στις δυο πρώτες γραμμές, απλά στέλνουμε το μήνυμα release σε κάθε μια από τις μεταβλητές αντιτύπων. Εδώ δεν χρειάζεται να χρησιμοποιήσουμε το autorealease καθώς το στάνταρτ release είναι λίγο γρηγορότερο.

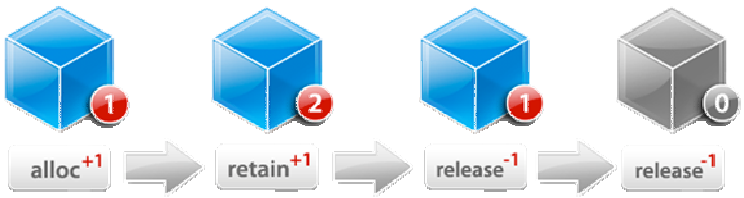
Η τελευταία γραμμή είναι πολύ σημαντική. Πρέπει να στείλουμε το μήνυμα **[super dealloc]** για να ζητήσουμε από την κλάση super να κάνει τον δικό της καθαρισμό. Αν δεν το κάνουμε αυτό, το αντικείμενο δεν θα αφαιρεθεί, το οποίο οδηγεί στην διαρροή μνήμης (memory leak).

Η μέθοδος dealloc δεν καλείται σε αντικείμενα αν η συλλογή σκουπιδιών είναι ενεργοποιημένη. Αντιθέτως, πρέπει να υλοποιήσετε την μέθοδο finalize.

3.7 Περισσότερα με την Διαχείριση Μνήμης

Το σύστημα διαχείρισης μνήμης της Objective-C ονομάζεται reference counting. Αυτό που πρέπει να κάνετε είναι να σημειώνετε τις αναφορές (references) και η ώρα εκτέλεσης (runtime) ουσιαστικά ελευθερώνει την μνήμη.

Με απλά λόγια, εσείς κατανέμετε (**alloc)** ένα αντικείμενο, ίσως κάποια στιγμή το διατηρήσετε (**retain),** μετά το απελευθερώνετε (**release**) για κάθε alloc/retain που στείλατε. Έτσι, αν έχετε χρησιμοποιήσει το alloc μια φορά και άλλη μια το retain, τότε πρέπει να το απελευθερώσετε (release) δυο φορές.



Αυτή είναι η θεωρία του reference counting. Πρακτικά όμως, συνήθως υπάρχουν δυο λόγοι για δημιουργήσετε ένα αντικείμενο:

1. Να το κρατήσετε σαν μεταβλητή αντιτύπων (instance variable)
2. Να το χρησιμοποιήσετε προσωρινά για απλή χρήση μέσα σε μια συνάρτηση

Τις περισσότερες φορές, η μέθοδος εγγραφής (setter) για μια μεταβλητή αντιτύπων θα πρέπει απλά να αυτοαπελευθερώσει (**autorelease)** το παλιό αντικείμενο, και να διατηρήσει (**retain**) το καινούργιο. Τότε μπορείτε να σιγουρευτείτε ότι θα το απελευθερώσετε (release) επίσης στη μέθοδο **dealloc**.

Επομένως η μόνη πραγματική δουλειά είναι να διαχειρίζεστε τις τοπικές αναφορές (local references) μέσα σε μια συνάρτηση˙ και υπάρχει ένας κανόνας: αν δημιουργείτε ένα αντικείμενο με την **alloc** ή **copy** , στέλνετε ένα **release** ή **autorelease** μήνυμα στο τέλος της συνάρτησης. Αν δημιουργείτε ένα αντικείμενο με οποιαδήποτε άλλο τρόπο, δεν κάνετε τίποτα.

Εδώ είναι η πρώτη περίπτωση, διαχειρίζοντας μια μεταβλητή αντιτύπων:

```
- (void) setTotalAmount: (NSNumber*)input
{
    [totalAmount autorelease];
    totalAmount = [input retain];
}

- (void) dealloc
{
    [totalAmount release];
    [super dealloc];
}
```

Εδώ είναι η δεύτερη περίπτωση, αυτή των τοπικών αναφορών (local references). Το μόνο που χρειάζεται είναι να απελευθερώσουμε το αντικείμενο το οποίο δημιουργήθηκε με την **alloc**:

```
NSNumber* value1 = [[NSNumber alloc] initWithFloat:8.75];
NSNumber* value2 = [NSNumber numberWithFloat:14.78];

// απελευθέρωσε μονο την μεταβλητή value1, οχι την value2
[value1 release];
```

Πρόσθετα, παραθέτουμε έναν συνδυασμό: χρησιμοποιώντας τοπικές αναφορές για να ορίσουμε ένα αντικείμενο σαν μια μεταβλητή αντιτύπων.

```
NSNumber* value1 = [[NSNumber alloc] initWithFloat:8.75];
[self setTotal:value1];
```

```
NSNumber* value2 = [NSNumber numberWithFloat:14.78];
[self setTotal:value2];

[value1 release];
```

Προσέξτε πως οι κανόνες για την διαχείριση τοπικών αναφορών είναι ακριβώς οι ίδιοι,
ανεξαρτήτως αν τις ορίζετε σαν μεταβλητές αντιτύπων ή όχι. Δεν χρειάζεται να σκέφτεστε πως υλοποιούνται οι μέθοδοι εγγραφής.

3.8 Διαχείριση Μνήμης μέσω ARC (Automatic Reference Counting) [4]

Η καινούργια λύση η οποία εισήχθη στο Mac OS 10.7 και iOS 5, είναι η αυτόματη μέτρηση αναφορών (automatic reference counter), περισσότερο γνωστό σαν ARC. [5]

Το ARC είναι ένα χαρακτηριστικό του καινούργιου LLVM 3.0 μεταγλωττιστή το οποίο καταργεί εντελώς την χειροκίνητη διαχείριση μνήμης.

Χρησιμοποιώντας το ARC στις δικές σας εργασίες είναι εξαιρετικά απλό. Συνεχίζετε να προγραμματίζετε όπως συνήθως, εκτός ότι δεν καλείτε πλέον το retain, release και autorelease.

Με το Automatic Reference Counting ενεργοποιημένο, ο μεταγλωττιστής θα εισάγει αυτόματα τα retains, release και autorelease στα σωστά σημεία στο πρόγραμμά σας. Δεν χρειάζεται πλέον να ανησυχείτε για αυτά, επειδή ο μεταγλωττιστής το κάνει αυτόματα για εσάς. Χρησιμοποιώντας το ARC είναι τόσο απλό.

3.8.1 Πως δουλεύει

Όπως αναφέρθηκε πιο πάνω η χειροκίνητη διαχείριση μνήμης δουλεύει κάπως έτσι:

- Αν χρειάζεστε ένα αντικείμενο πρέπει να το διατηρήσετε, εκτός και αν ήδη διατηρήθηκε για εσάς.
- Αν θέλετε να σταματήσετε να χρησιμοποιείτε ένα αντικείμενο πρέπει να το απελευθερώσετε, εκτός και αν απελευθερώθηκε ήδη για εσάς (μέσω του autorelease).

Οι αρχές της χειροκίνητης διαχείρισης μνήμης δεν είναι δύσκολες αλλά είναι πολύ εύκολο να κάνετε ένα λάθος. Και αυτά τα μικρά λάθη μπορούν να έχουν τρομερές συνέπειες. Είτε η εφαρμογή σας θα καταρρεύσει κάποια στιγμή επειδή ελευθερώσατε ένα αντικείμενο νωρίς και οι μεταβλητές σας δείχνουν σε δεδομένα τα οποία δεν είναι πλέον έγκυρα, ή θα ξεμείνετε από μνήμη επειδή δεν έχετε ελευθερώσει τα αντικείμενα επαρκώς και παραμένουν τριγύρω για πάντα.

Το εργαλείο στατικής ανάλυσης (static analyzer) από το Xcode είναι μια καλή βοήθεια για να βρείτε αυτού του είδους προβλήματα αλλά το ARC πηγαίνει ένα βήμα περαιτέρω. Αποφεύγει εντελώς τα προβλήματα διαχείρισης μνήμης εισάγοντας αυτόματα τα κατάλληλα retain και release για εσάς.

Είναι σημαντικό να καταλάβετε ότι το ARC είναι ένα χαρακτηριστικό του Objective-C μεταγλωττιστή και ως εκ τούτου όλα τα πράγματα σχετικά με το ARC γίνονται καθώς χτίζετε την εφαρμογή σας. Το ARC δεν είναι ένα χαρακτηριστικό χρόνου εκτέλεσης, (εκτός ενός πολύ μικρού κομματιού, το weak pointer system), ούτε είναι ένας συλλέκτης σκουπιδιών (garbage collection) που ίσως γνωρίζετε από άλλες γλώσσες.

Αυτό που κάνει το ARC είναι να εισάγει τα retains και τα releases στον κώδικά σας όταν το μεταγλωττίζει ακριβώς όπου - ή τουλάχιστον θα έπρεπε να τα έχετε τοποθετήσει οι ίδιοι. Αυτό το κάνει το ARC τόσο γρήγορο όσο η χειροκίνητη διαχείριση του κώδικα και μερικές φορές ακόμη γρηγορότερα επειδή μπορεί να πραγματοποιήσει κάποιες σίγουρες βελτιστοποιήσεις.

3.8.2 Οι δείκτες κρατούνε τα αντικείμενα ζωντανά.

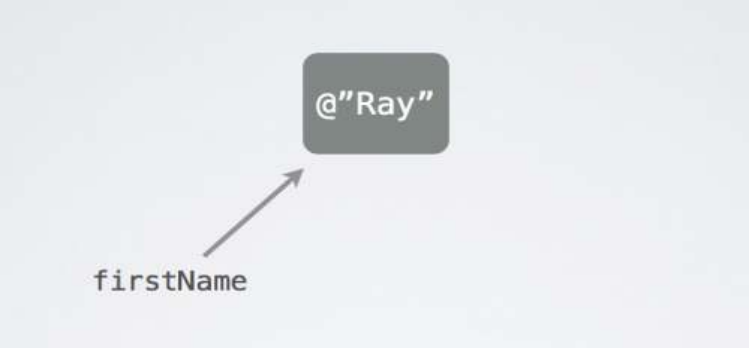
Οι καινούργιοι κανόνες που έχετε να μάθετε για το ARC είναι σχετικά απλοί. Με τη χειροκίνητη διαχείριση μνήμης έπρεπε να διατηρείτε ένα αντικείμενο για να το κρατήσετε ζωντανό. Αυτό πλέον δεν είναι απαραίτητο, το μόνο που πρέπει να κάνετε είναι να έχετε έναν δείκτη στο αντικείμενο. Όσο υπάρχει η μεταβλητή η οποία δείχνει σε ένα αντικείμενο, το αντικείμενο αυτό παραμένει στη μνήμη.

Όταν ο δείκτης παίρνει μια καινούργια τιμή ή διακόπτει την υπάρχουσα, το συσχετισμένο αντικείμενο απελευθερώνεται. Αυτό συμβαίνει για όλες τις μεταβλητές: μεταβλητές αντιτύπων, παραγόμενες ιδιότητες (synthesized properties) και ακόμη και τις τοπικές μεταβλητές.

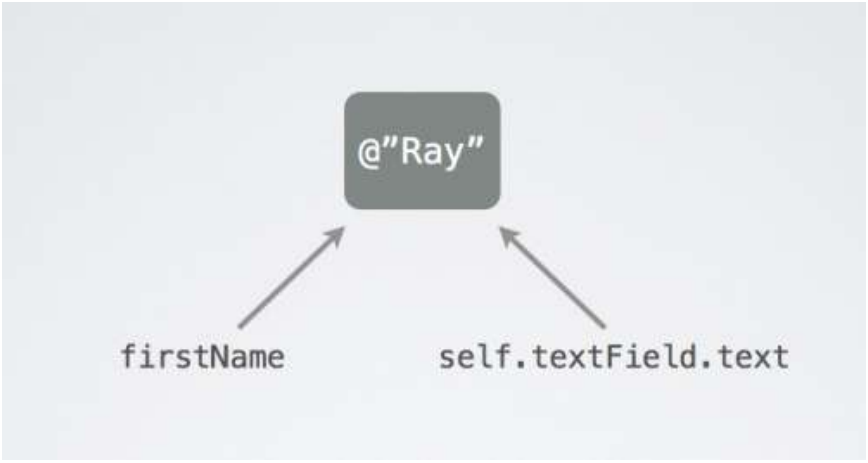
Μπορείτε να το σκέφτεστε σε αντιστοιχία ιδιοκτησίας. Όταν κάνετε το παρακάτω,

```
NSString *firstName = self.textField.text;
```

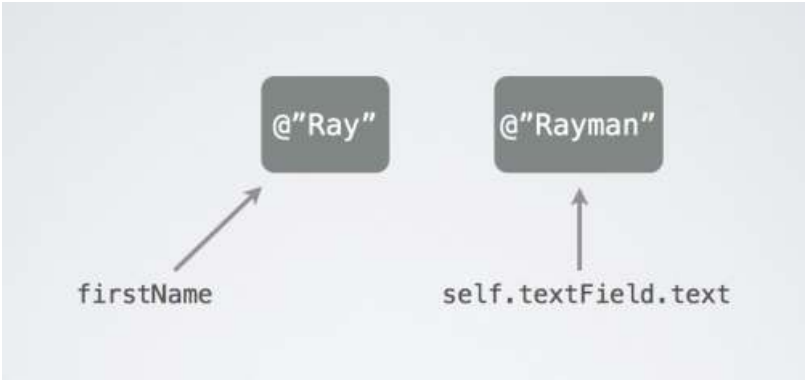
η μεταβλητή firstName γίνεται ένας δείκτης στο αντικείμενο NSString το οποίο κρατάει τα περιεχόμενα του πεδίου κειμένου. Η μεταβλητή firstName είναι τώρα ο ιδιοκτήτης αυτού του αντικειμένου string.



Ένα αντικείμενο μπορεί να έχει περισσότερους από έναν ιδιοκτήτη. Μέχρι ο χρήστης να αλλάξει τα περιεχόμενα του UITextField, η ιδιότητα text είναι επίσης ένας ιδιοκτήτης του αντικειμένου string. Υπάρχουνε δυο δείκτες που κρατούνε το ίδιο string αντικείμενο ζωντανό:



Κάποια στιγμή αργότερα ο χρήστης θα πληκτρολογήσει κάτι καινούργιο στο πεδίο κειμένου και η ιδιότητα text τώρα δείχνει σε ένα καινούργιο string αντικείμενο. Αλλά το αρχικό string αντικείμενο έχει ακόμη έναν ιδιοκτήτη (τη μεταβλητή firstName) και έτσι παραμένει στη μνήμη.

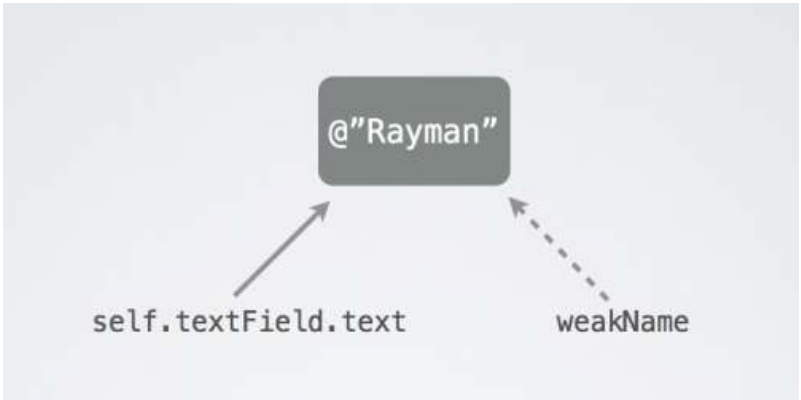


Μόνο όταν το firstName παίρνει μια καινούργια τιμή ή βγαίνει εκτός των δραστηριοτήτων του - επειδή είναι μια τοπική μεταβλητή και η μέθοδος τελειώνει, ή επειδή είναι μια μεταβλητή αντιτύπων και το αντικείμενο που του ανήκει έχει αποδεσμευτεί - λήγει η ιδιοκτησία. Το string αντικείμενο πλέον δεν

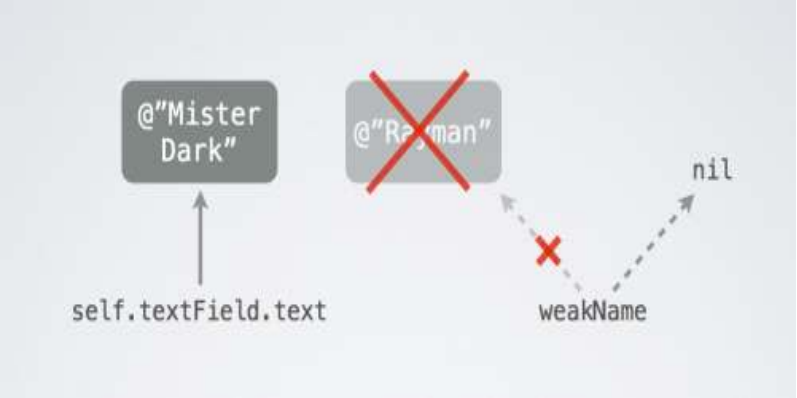
έχει άλλους ιδιοκτήτες, το μέτρημα των διατηρήσεων του (retain count) πέφτει στο 0 και το αντικείμενο αποδεσμεύεται.

Αποκαλούμε του δείκτες όπως firstName και textField.text “δυνατούς” (strong) επειδή κρατούνε τα αντικείμενα ζωντανά. Εξ’ ορισμού όλες οι μεταβλητές αντιτύπων και οι τοπικές μεταβλητές είναι δυνατοί δείκτες. Υπάρχει επίσης και ο “αδύναμος” (weak) δείκτης. Μεταβλητές οι οποίες είναι αδύναμες μπορούνε επίσης να δείχνουνε σε αντικείμενα αλλά δεν γίνονται ιδιοκτήτες.

```
__weak NSString *weakName = self.textField.text;
```



Η μεταβλητή weakName δείχνει στο ίδιο string αντικείμενο που δείχνει και η ιδιότητα textField.text, αλλά δεν είναι ο ιδιοκτήτης. Αν τα περιεχόμενα του πεδίου κειμένου αλλάξουν, τότε το αντικείμενο string δεν θα έχει πλέον ιδιοκτήτες και θα αποδεσμευτεί:



Όταν συμβεί αυτό, η τιμή του weakName αυτόματα θα γίνει nil. Λέγεται ότι είναι ένας αδύναμος δείκτης που 'μηδενίζεται' ('zeroing' weak pointer)

Προσέξτε ότι είναι υπερβολικά εξυπηρετικό επειδή αποτρέπει τους αδύναμους pointers να δείχνουν σε απελευθερωμένη μνήμη. Αυτή η κατάσταση δημιουργούσε πάρα πολλά bugs - ίσως να έχετε ακούσει τον όρο "dangling pointers" (αιωρούμενοι δείκτες) ή "zombies" - αλλά χάρη στον μηδενισμό των αδύναμων δεικτών αυτό πλέον δεν είναι πρόβλημα.

Πιθανόν να μη χρησιμοποιείτε συχνά τους αδύναμους δείκτες. Είναι κατά κύριο λόγο χρήσιμοι όταν τα αντικείμενα έχουν σχέση γονέα-παιδιού. Ο γονέας θα έχει έναν δυνατό δείκτη στο παιδί - και έτσι "κατέχει" το παιδί - αλλά για την αποφυγή των κύκλων ιδιοκτησίας, μόνο το παιδί θα έχει έναν αδύναμο δείκτη πίσω στον γονέα.

Προσέξτε ότι τα παρακάτω δεν είναι και πολύ χρήσιμα:

```
__weak NSString *str = [[NSString alloc] initWithFormat:...];
NSLog(@"%@",str) // θα έχει έξοδο "(null)"
```

Δεν υπάρχει ιδιοκτήτης του αντικειμένου string (επειδή το str είναι αδύναμο) και το αντικείμενο θα απελευθερωθεί αμέσως μόλις δημιουργηθεί.

Μπορείτε να χρησιμοποιήσετε τη λέξη κλειδί __strong για να υποδηλώσετε ότι μια μεταβλητή είναι ένας δυνατός δείκτης:

```
__strong NSString *firstName = self.textField.text;
```

Αλλά επειδή οι μεταβλητές είναι εξ' ορισμού δυνατές αυτό είναι περιττό.

Οι ιδιότητες (properties) μπορούν επίσης να είναι δυνατές και αδύναμες. Ο συμβολισμός για τις ιδιότητες είναι:

```
@property (nonatomic, strong) NSString *firstName;
@property (nonatomic, weak) id <MyDelegate> delegate;
```

Το ARC είναι σπουδαίο και θα αφαιρέσει μεγάλη αταξία απ' τον κώδικά σας. Δεν χρειάζεται πλέον να σκέφτεστε πότε πρέπει να διατηρείτε και πότε να απελευθερώνετε. Η ερώτηση που πρέπει να κάνετε είναι: ποιος κατέχει τι;

Για παράδειγμα, ήταν αδύνατο παλιότερα να γράψετε κώδικα όπως αυτόν:

```
id obj = [array objectAtIndex:0];
[array removeObjectAtIndex:0];
NSLog(@"%@", obj);
```

Χρησιμοποιώντας τη χειροκίνητη διαχείριση μνήμης, η αφαίρεση του αντικειμένου από τον πίνακα θα ακύρωνε τα περιεχόμενα του αντικειμένου obj.

Το αντικείμενο θα απελευθερωνότανε αμέσως μόλις δεν ήταν μέρος του πίνακα. Εμφανίζοντας το αντικείμενο με την NSLog() θα οδηγούσε στην κατάρρευση της εφαρμογής. Με το ARC, ο παραπάνω κώδικας θα δούλευε όπως επιδιώκετε. Επειδή τοποθετούμε το αντικείμενο μέσα στη μεταβλητή obj, ο οποίος είναι ένας δυνατός δείκτης, ο πίνακας δεν είναι πλέον ο μοναδικός ιδιοκτήτης του αντικειμένου. Ακόμα και αν αφαιρέσουμε το αντικείμενο από τον πίνακα, το αντικείμενο θα είναι ακόμη ζωντανό επειδή το obj συνεχίζει να δείχνει σε αυτό.

Το Automatic Reference Counting έχει κάποιους περιορισμούς. Για αρχή, το ARC δουλεύει μόνο με Objective-C αντικείμενα. Αν η εφαρμογή σας χρησιμοποιεί το Core Foundation ή την malloc() και free(), τότε είστε εσείς υπεύθυνοι για να κάνετε την διαχείριση μνήμης. Επιπλέον, ορισμένοι κανόνες της γλώσσας έχουν γίνει αυστηρότεροι έτσι ώστε σίγουρα το ARC θα κάνει πάντα σωστά την δουλειά του.

Επειδή το ARC αναλαμβάνει τα retain και releases για εσάς στα σωστά σημεία, δεν σημαίνει ότι πρέπει να ξεχάσετε εντελώς την διαχείριση μνήμης. Επειδή οι δυνατοί δείκτες κρατούνε τα αντικείμενα ζωντανά, υπάρχουν μερικές καταστάσεις όπου πρέπει να ορίσετε αυτούς τους δείκτες σε nil μόνοι σας, ή η εφαρμογή σας μπορεί να ξεμείνει από διαθέσιμη μνήμη. Αν εξακολουθείτε να κρατάτε τα αντικείμενα που έχετε δημιουργήσει, τότε το ARC δεν θα είναι σε θέση να τα απελευθερώσει. Ως εκ τούτου, οποτεδήποτε δημιουργείτε ένα καινούργιο αντικείμενο, θα πρέπει ακόμη να σκέφτεστε ποιος το κατέχει και για πόσο καιρό το αντικείμενο θα υπάρχει.

3.9 Καταγραφή Μηνυμάτων (Logging)

Η καταγραφή μηνυμάτων (logging messages) στη κονσόλα (console) στην Objective-C είναι πολύ απλή. Στην πραγματικότητα, η συνάρτηση NSLog() είναι πανομοιότυπη με την συνάρτηση της C **printf()**, εκτός ότι υπάρχει μια επιπλέον ένδειξη %@ για αντικείμενα.

```
NSLog ( @"The current date and time is: %@", [NSDate date] );
```

Μπορείτε να καταγράψετε ένα αντικείμενο στην κονσόλα. Η συνάρτηση NSLog καλεί τη μέθοδο description (περιγραφή) στο αντικείμενο, και εκτυπώνει το NSString το οποίο επιστρέφεται. Μπορείτε να παραμερίσετε τη μέθοδο description στις κλάσεις σας και να επιστρέψετε ένα δικό σας string.

3.10 Ιδιότητες (Properties)

Προηγουμένως όταν γράψαμε την accessor μέθοδο για τη μεταβλητή 'caption' και 'photographer', μπορεί ίσως να προσέξατε ότι ο κώδικας είναι απλός και πιθανώς μπορεί να γενικευτεί.

Οι ιδιότητες είναι ένα χαρακτηριστικό της Objective-C το οποίο μας επιτρέπει να παράγουμε αυτόματα accessors, και να έχουμε επιπλέον δευτερεύον οφέλη. Ας μετατρέψουμε την κλάση Photo και να χρησιμοποιήσουμε τις ιδιότητες:

Εδώ φαίνεται πως ήτανε προηγουμένως:

```
#import <Cocoa/Cocoa.h>

@interface Photo : NSObject {
    NSString* caption;
    NSString* photographer;
}
- (NSString*) caption;
- (NSString*) photographer;

- (void) setCaption: (NSString*)input;
- (void) setPhotographer: (NSString*)input;

@end
```

Παρακάτω φαίνεται ξανά όπως έχει μετατραπεί σε properties:

```
#import <Cocoa/Cocoa.h>

@interface Photo : NSObject {
    NSString* caption;
    NSString* photographer;
}
@property (retain) NSString* caption;
@property (retain) NSString* photographer;
```

```
@end
```

Το σύμβολο @property είναι μια οδηγία (directive) η οποία δηλώνει την ιδιότητα (property). Το “retain”
μέσα στις παρενθέσεις προσδιορίζει ότι η μέθοδος εγγραφής (setter) πρέπει να **διατηρεί** (retain) την τιμή εισόδου και η υπόλοιπη γραμμή απλά προσδιορίζει τον τύπο και το όνομα της ιδιότητας.

Τώρα ας ρίξουμε μια ματιά στην υλοποίηση αυτή της κλάσης:

```
#import "Photo.h"

@implementation Photo

@synthesize caption;
@synthesize photographer;

- (void) dealloc
{
    [caption release];
    [photographer release];
    [super dealloc];
}

@end
```

Το σύμβολο **@synthesize** αυτόματα δημιουργεί τους setters και τους getters για εμάς, έτσι ώστε το μόνο που πρέπει να υλοποιήσουμε εμείς για αυτήν την κλάση, είναι η μέθοδος dealloc.

Οι μέθοδοι ανάγνωσης (accessors) θα δημιουργηθούν μόνο αν δεν υπάρχουν ήδη, γι’ αυτό είσαστε ελεύθεροι να προσδιορίσετε το @synthesize για μια ιδιότητα, μετά να την υλοποιήσετε σύμφωνα με την δική σας μέθοδο εγγραφής ή ανάγνωσης εάν θέλετε. Ο μεταγλωττιστής θα συμπληρώσει οποιαδήποτε μέθοδο λείπει.

3.11 Κλήση μεθόδων σε Nil

Στην Objective-C, το αντικείμενο **nil** είναι λειτουργικά ισοδύναμο με τον **NULL** δείκτη όπως σε πολλές διαφορετικές γλώσσες. Η διαφορά είναι ότι μπορείτε να καλέσετε μεθόδους στο nil χωρίς να καταρρεύσει ή να εμφανιστεί μια εξαίρεση (exception).

Αυτή η τεχνική χρησιμοποιείται από τα frameworks σε μια πληθώρα τρόπων, αλλά αυτό που έχει σημασία για εσάς αυτήν την στιγμή είναι ότι συνήθως δεν χρειάζεται να ελέγχετε για το nil προτού να καλέσετε μια μέθοδο σε ένα αντικείμενο. Αν καλέσετε μια μέθοδο στο nil η οποία επιστρέφει ένα αντικείμενο, τότε θα πάρετε nil σαν επιστρεφόμενη τιμή.

Μπορούμε ακόμη να χρησιμοποιήσουμε αυτό για να βελτιώσουμε ελαφρώς τη μέθοδο dealloc:

```
- (void) dealloc
{
    self.caption = nil;
    self.photographer = nil;
    [super dealloc];
}
```

Αυτό δουλεύει επειδή όταν ορίσουμε μια μεταβλητή αντιτύπων σε **nil**, ο μέθοδος εγγραφής (setter) απλά διατηρεί το nil (το οποίο δεν κάνει τίποτα) και απελευθερώνει την παλιά τιμή. Αυτή η προσέγγιση είναι συχνά καλύτερη για αποδέσμευση καθώς δεν υπάρχει πιθανότητα της μεταβλητής να δείχνει σε τυχαία δομένα όπου ένα αντικείμενο βρισκότανε παλαιότερα.

Προσέξτε ότι εδώ χρησιμοποιούμε τη σύνταξη **self.<var>**,η οποία σημαίνει ότι χρησιμοποιούμε τον μέθοδο εγγραφής. Αν ορίζαμε απλώς τις τιμές όπως φαίνεται παρακάτω, τότε θα υπήρχε διαρροή μνήμης (memory leak).

```
// λανθασμένο. δημιουργεί διαρροή μνήμης
// χρησιμοποιήστε το self.caption για να το ορίσετε μέσω του μεθόδου εγγραφής
caption = nil;
```

3.12 Κατηγορίες (Categories)

Οι κατηγορίες είναι ένα από τα πιο χρήσιμα χαρακτηριστικά της Objective-C. Ουσιαστικά, μια κατηγορία σας επιτρέπει να προσθέσετε μεθόδους σε ήδη υπάρχουσες κλάσεις χωρίς να χρειάζεται να το βάλετε σε μια υποκλάση (subclass) ή να χρειάζεται να ξέρετε τις λεπτομέρειες της υλοποίησής της.

Αυτό είναι ιδιαίτερα χρήσιμο επειδή μπορείτε να προσθέσετε μεθόδους σε ενσωματωμένα αντικείμενα (built-in objects). Αν θέλετε να προσθέσετε μια μέθοδο σε όλες τις μεταβλητές (instances) του NSString στην εφαρμογή σας, απλά προσθέτετε μια κατηγορία.

Για παράδειγμα, αν θέλετε να προσθέσετε μια μέθοδο NSString για να αποφασίσετε αν τα περιεχόμενα είναι ένα URL, θα γινότανε κάπως έτσι:

```
#import <Cocoa/Cocoa.h>

@interface NSString (Utilities)
- (BOOL) isURL;
@end
```

Αυτό είναι παρόμοιο με τη δήλωση μιας κλάσης. Οι διαφορές είναι ότι δεν απαριθμείται η super κλάση και ότι υπάρχει ένα όνομα της κατηγορίας στις παρενθέσεις. Το όνομα μπορεί να είναι οτιδήποτε θέλετε, ωστόσο θα πρέπει να ανακοινώνει τι κάνει η εσωτερική μέθοδος.

Ορίστε η υλοποίηση. Κρατήστε στο μυαλό σας ότι αυτή δεν είναι μια καλή υλοποίηση για ανίχνευση ενός URL. Απλά προσπαθούμε να αποκτήσουμε την έννοια των κατηγοριών εγκαρσίως:

```
#import "NSString-Utilities.h"
```

```
@implementation NSString (Utilities)

- (BOOL) isURL
{
    if ( [self hasPrefix:@"http://"] )
        return YES;
    else
        return NO;
}

@end
```

Τώρα μπορείτε να χρησιμοποιήσετε αυτή τη μέθοδο σε οποιοδήποτε NSString. Ο παρακάτω κώδικας θα εκτυπώσει "string1 is a URL" στη κονσόλα:

```
NSString* string1 = @"http://pixar.com/";
NSString* string2 = @"Pixar";

if ( [string1 isURL] )
    NSLog (@"string1 is a URL");

if ( [string2 isURL] )
    NSLog (@"string2 is a URL");
```

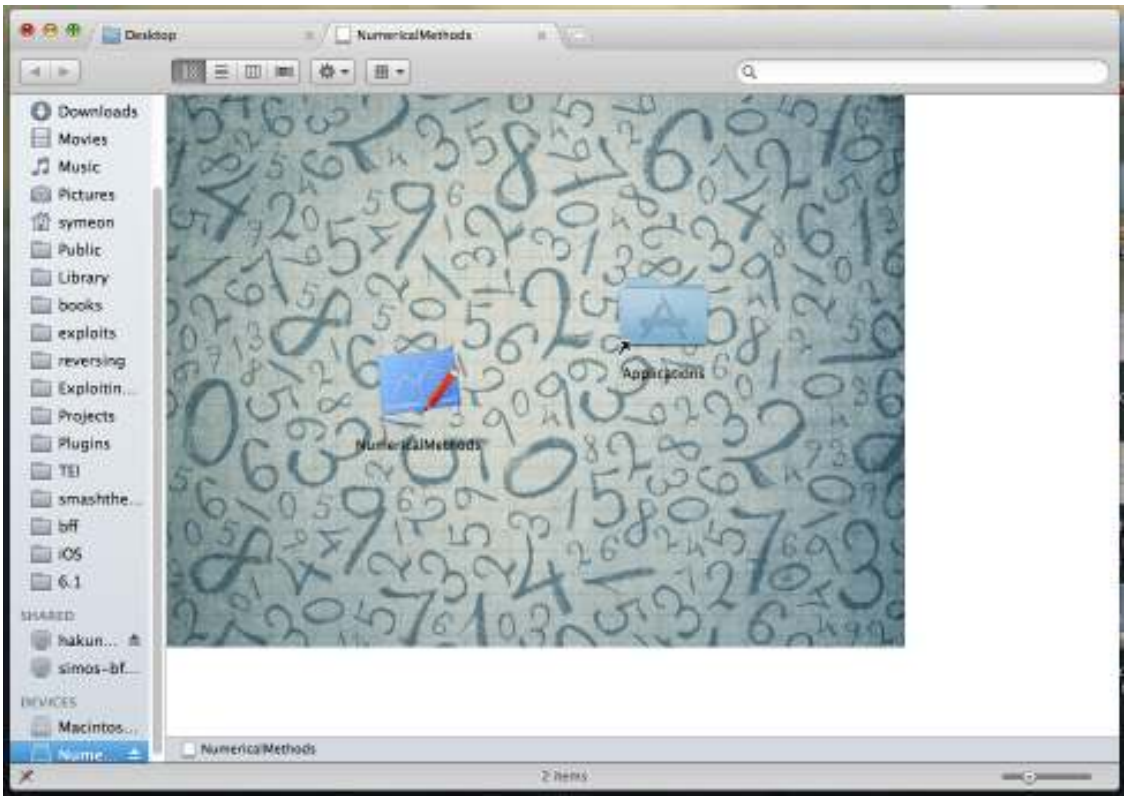
Αντιθέτως με τις υποκλάσεις, οι κατηγορίες δεν μπορούν να προσθέσουν μεταβλητές αντιτύπων. Μπορείτε, ωστόσο, να χρησιμοποιήσετε τις κατηγορίες για να παρακάμψετε υπάρχουσες μεθόδους στις κλάσεις, αλλά να το κάνετε πολύ προσεκτικά.

Θυμηθείτε, όταν κάνετε αλλαγές σε μια κλάση χρησιμοποιώντας μια κατηγορία, επηρεάζονται όλες οι μεταβλητές της κλάσης αυτής σε όλη την εφαρμογή.

4. Οδηγός της εφαρμογής

4.1 Εγκατάσταση της εφαρμογής

Για να εγκαταστήσετε την εφαρμογή πρέπει να τρέξετε το αρχείο NumericalMethods.dmg και στην συνέχεια πρέπει απλά να σύρετε την εφαρμογή στο 'Applications'.

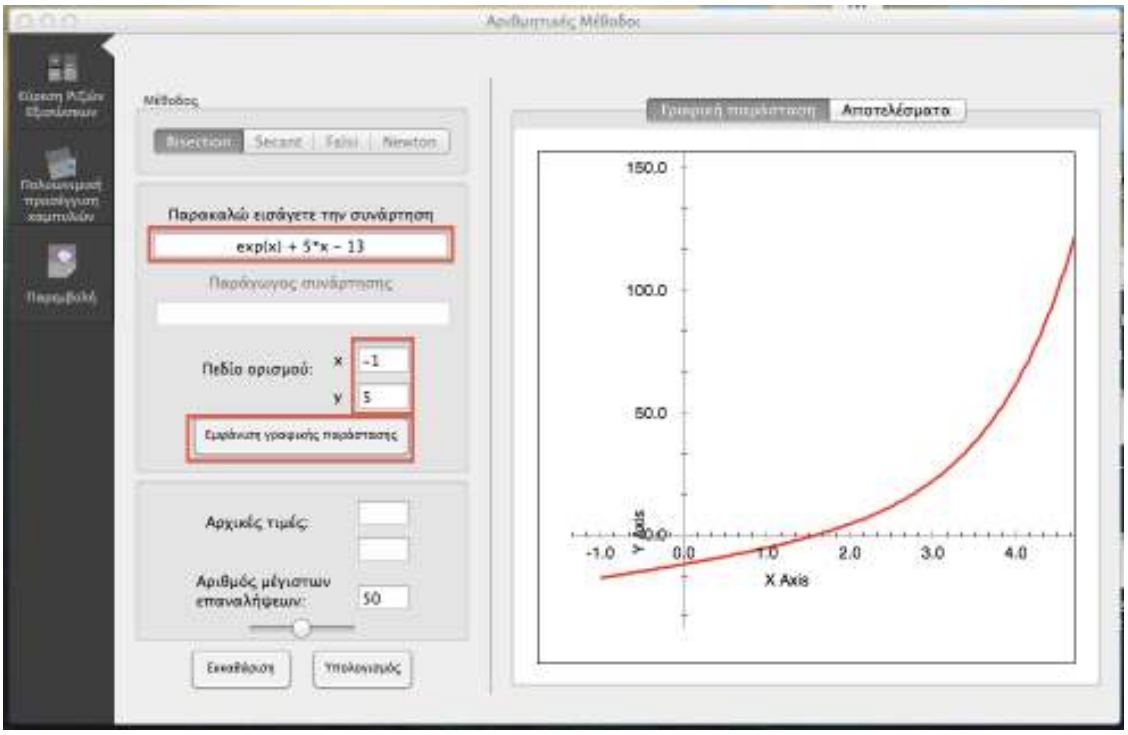


Εικόνα 4.1.1.Το αρχείο NumericalMethods.dmg το οποίο περιέχει την εφαρμογή.

4.2 Χρήση της εφαρμογής

4.2.1 Σχεδιασμός γραφικής παράστασης

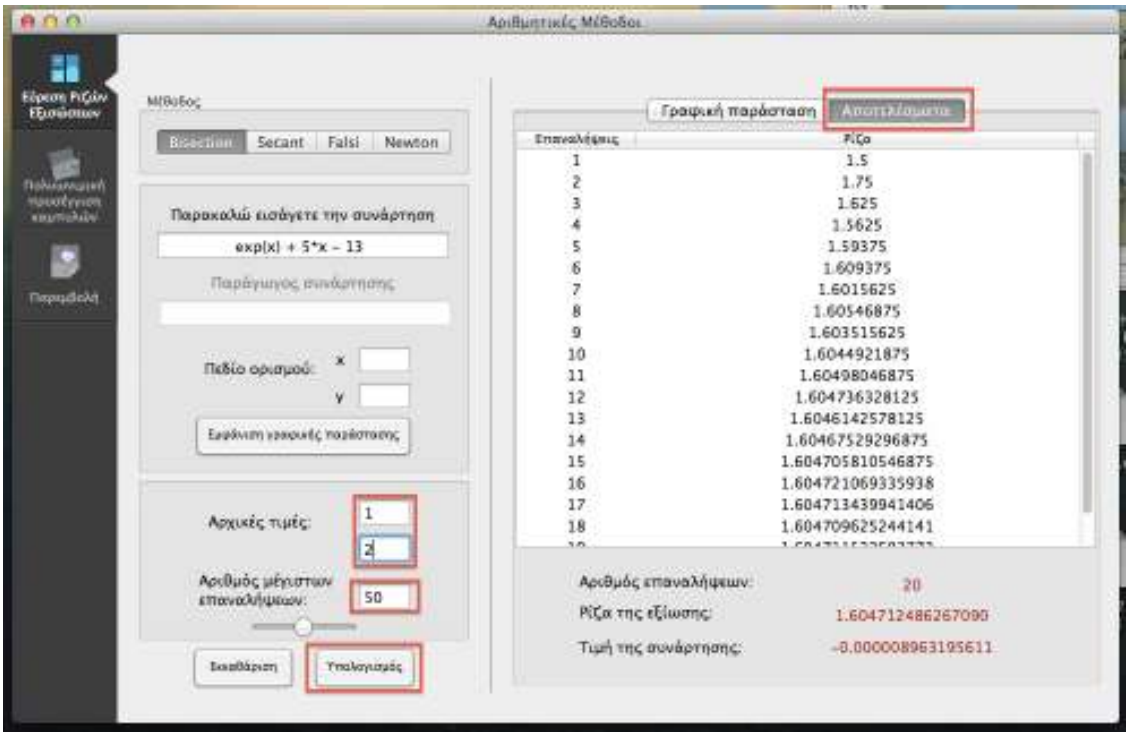
Για να σχεδιάσετε μια γραφική παράσταση, πρέπει να εισάγετε την συνάρτηση, να εισάγετε τις τιμές για x και y και στη συνέχεια πατάτε το κουμπί 'Εμφάνιση γραφικής παράστασης'.



Εικόνα 4.2.1.Η γραφική παράσταση e^{x}*5x-13 όπως σχεδιάζεται από την εφαρμογή.

4.2.2 Υπολογισμός βέλτιστης ρίζας

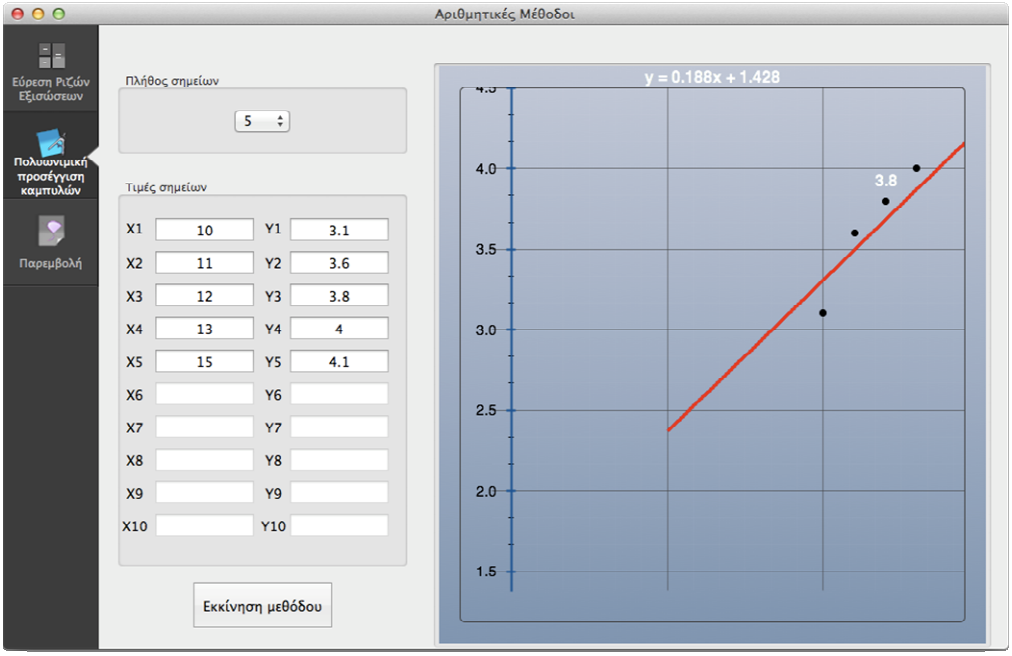
Για να βρείτε την βέλτιστη ρίζα της συνάρτησης αρχικά διαλέγετε μία από τις μεθόδους (Bisection, Secant, Falsi, Newton), στη συνέχεια εισάγετε αρχικές τιμές της συνάρτησης, επιλέγετε τον αριθμό μέγιστων επαναλήψεων και τέλος αφού διαλέξετε την καρτέλα 'Αποτελέσματα', επιλέγετε το κουμπί 'Υπολογισμός'.



Εικόνα 4.2.2.Εύρεση βέλτιστων ριζών χρησιμοποιώντας την μέθοδο Bisection.

4.2.3 Υπολογισμός ευθείας ελαχίστων τετραγώνων

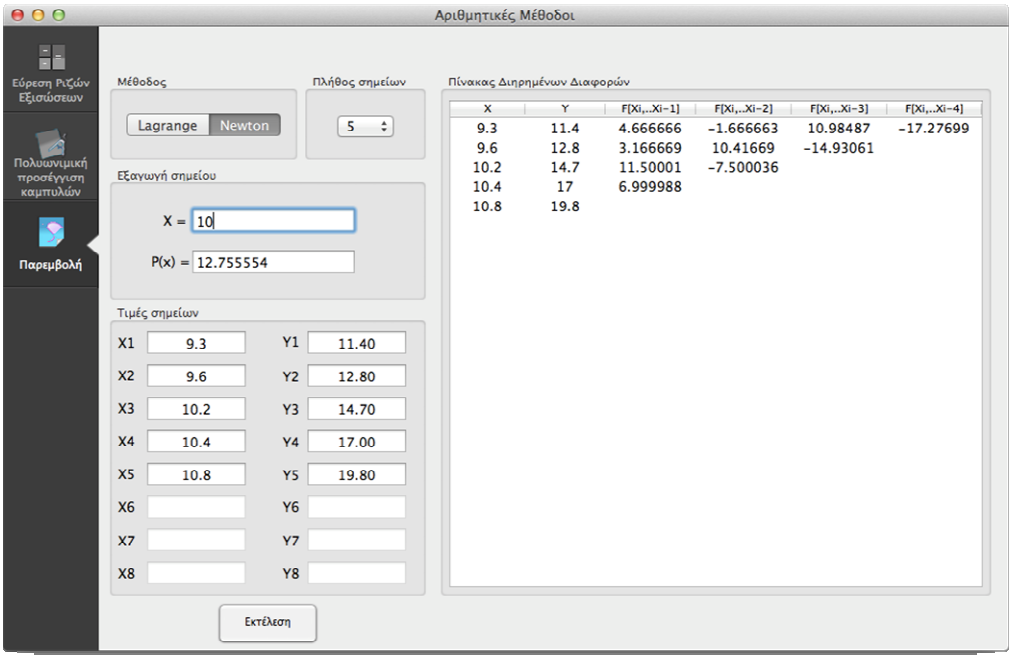
Για να υπολογίσετε την ευθεία ελαχίστων τετραγώνων, διαλέγετε το πλήθος των σημείων που θέλετε να εισάγετε, εισάγετε τις τιμές για x,y και επιλέγετε 'Εκκίνηση μεθόδου'.



Εικόνα 4.2.3. Υπολογισμός ευθείας ελαχίστων τετραγώνων.

4.2.4 Υπολογισμός παρεμβολής

Για υπολογίσετε την παρεμβολή μίας συνάρτησης διαλέγετε μία από τις μεθόδους (Lagrange, Newton), στη συνέχεια εισάγετε το πλήθος των σημείων και αφού εισάγετε τις τιμές τους, επιλέγετε το κουμπί 'Εκτέλεση'.



Εικόνα 4.2.4. Υπολογισμός με τη μέθοδο Newton και εύρεση του πίνακα διηρημένων διαφορών.

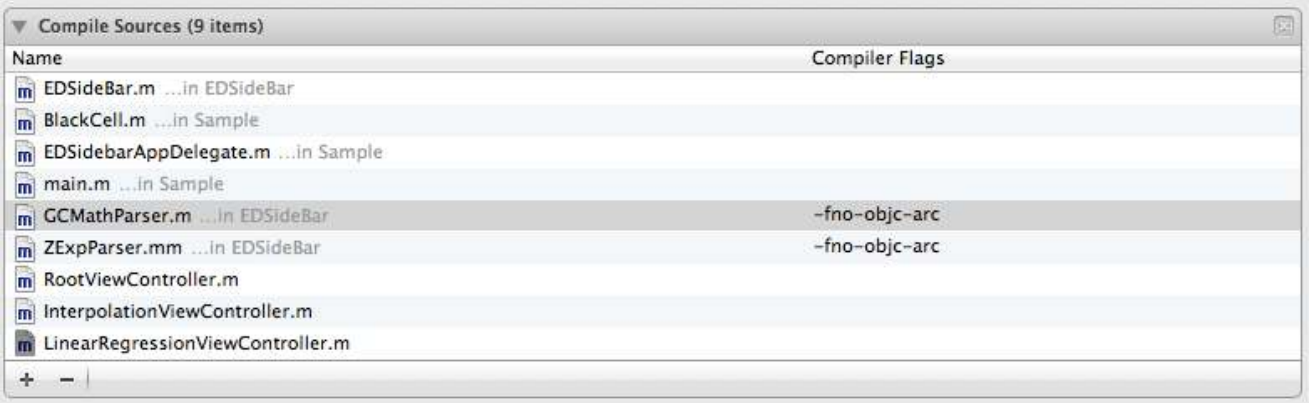
5. Ανάπτυξη της εφαρμογής

5.1 Γενικές πληροφορίες & εργαλεία που χρησιμοποιήθηκαν

Για να δημιουργήσουμε την εφαρμογή αυτήν χρησιμοποιήσαμε το Xcode [6] έκδοση 4.3.2 σε λειτουργικό σύστημα Mac OS X Lion (10.7.5).

Επιπλέον, για την δημιουργία των γραφικών παραστάσεων χρησιμοποιήθηκε το Core-Plot Framework v.1.1 [4], ένα δωρεάν και ανοιχτού κώδικα framework το οποίο μας επιτρέπει να σχεδιάσουμε γραφικές παραστάσεις σε υπολογιστές Mac και φορητές συσκευές με λειτουργικό σύστημα iOS.

Πρόσθετα, για τη διαχείριση μνήμης χρησιμοποιήσαμε την τεχνολογία ARC, εκτός κάποιων αρχείων καθώς παρουσίαζαν προβλήματα συμβατότητας.

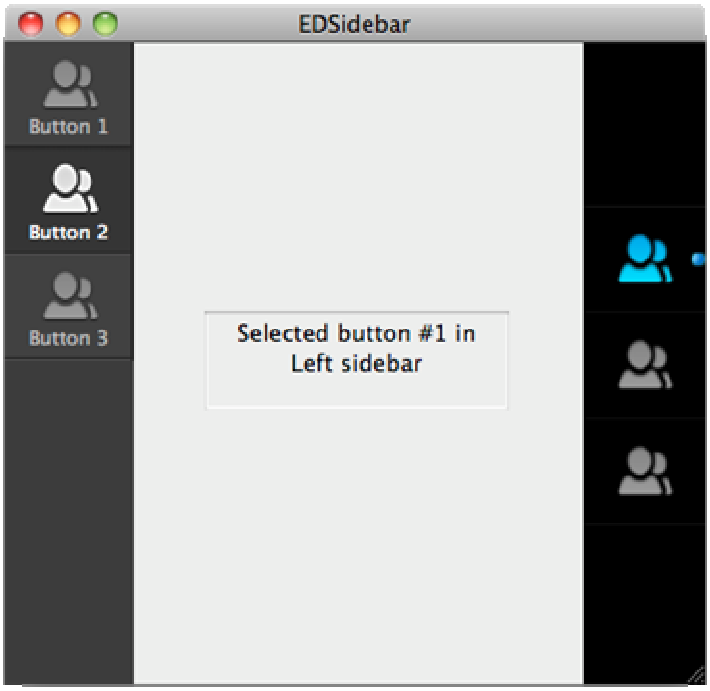


Εικόνα 5.1.1. Μπορείτε να βρείτε το υπομενού 'Compile Sources' στην καρτέλα "Build Phases". Αφού επιλέξετε το 'target' του project και προσθέσετε διάφορα flags ,αυτά θα εφαρμοστούνε κατά τη μεταγλώττιση της εφαρμογής.

Παρατηρήστε ότι σε κάποια από τα αρχεία μας, έχουμε προσθέσει το flag **–f**orce**no**-**obj**ective**c**-**arc** το οποίο απενεργοποιεί το ARC και επομένως η διαχείριση μνήμης σε αυτά τα αρχεία γίνεται χειροκίνητα.

5.2 Χρήση του EDSideBar Control

Αρχικά, χρησιμοποιούμε ένα custom control, το EDSideBar[5] το οποίο και έχουμε τροποποιήσει για τις ανάγκες της εργασίας.

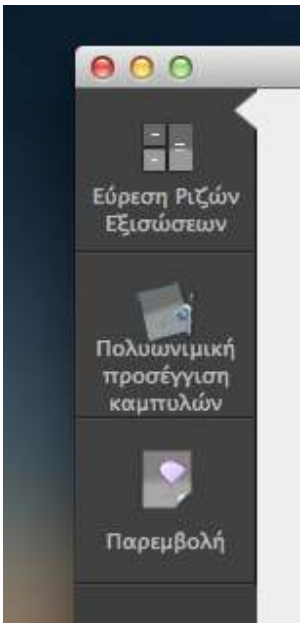


Εικόνα 5.2.1. Το EDSidebar στην πρωτότυπη έκδοσή του.

5.3 Δημιουργία κουμπιών & αντίστοιχων εικονιδίων

Η δημιουργία κουμπιών γίνεται στο αρχείο EDSidebarAppDelegate.m στη συνάρτηση
awakeFromNib():

```
[sideBarDefault addButtonWithTitle:@"Εύρεση Ριζών \nΕξισώσεων" image:[NSImage
imageNamed:@"roots_blue.png"] alternateImage:[NSImage
imageNamed:@"roots_gray.png"]];

[sideBarDefault addButtonWithTitle:@"Πολυωνιμική \nπροσέγγιση
\nκαμπυλών" image:[NSImage imageNamed:@"math_blue.png"]
alternateImage:[NSImage imageNamed:@"math_gray.png"]];

[sideBarDefault addButtonWithTitle:@"Παρεμβολή́" image:[NSImage
imageNamed:@"interpolation_blue.png"] alternateImage:[NSImage
imageNamed:@"interpolation_gray.png"]];
```



Εικόνα 5.3.1 Η εφαρμογή με τα κουμπιά & τα αντίστοιχα εικονίδιά τους.

5.4 Σύνδεση Views και εναλλαγή μεταξύ τους

Η εφαρμογή αποτελείται από έναν βασικό delegate (EDSidebarAppDelegate.m) – τον κορμό της εφαρμογής ο οποίος είναι υπεύθυνος για την σύνδεση και την εναλλαγή των Views (RootViewController, LinearRegressionViewController και InterpolationViewController).

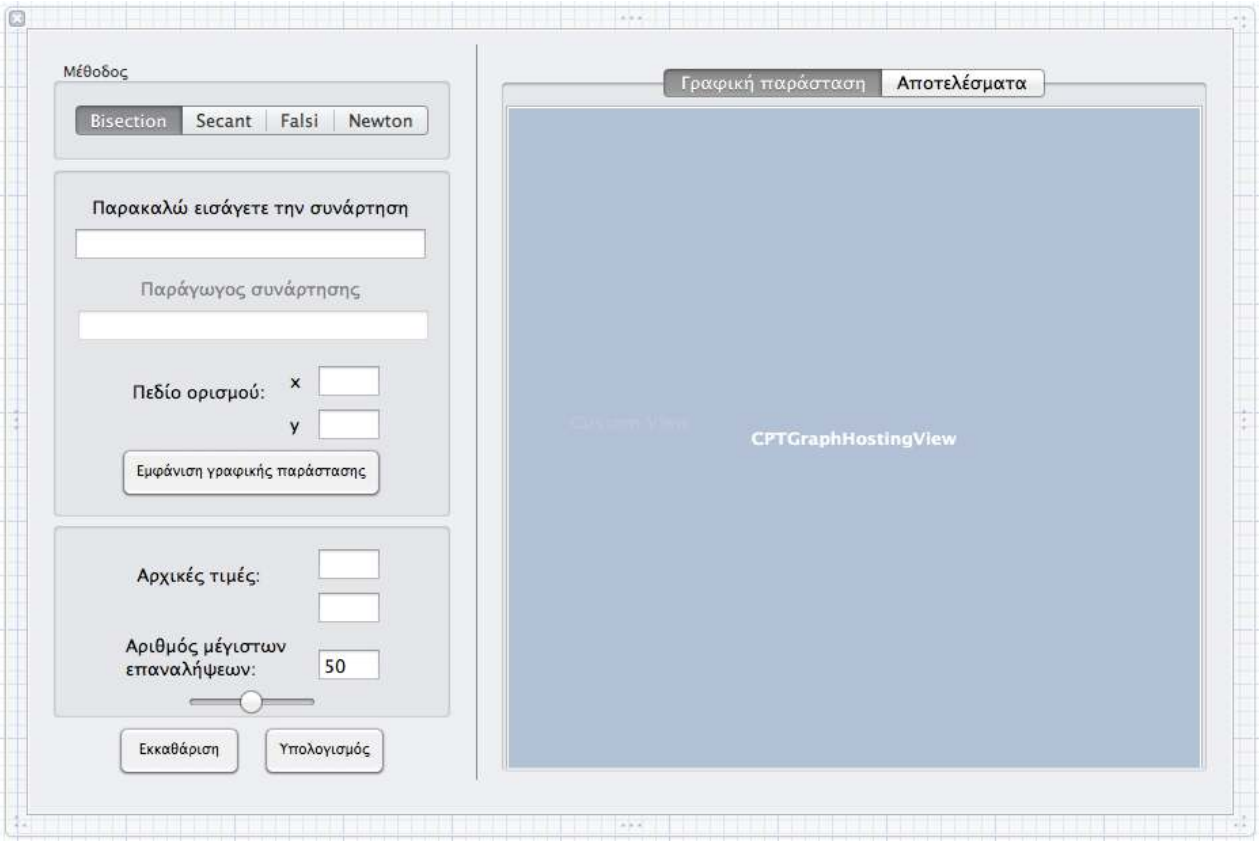
Ως εκ τούτου, αφού έχουμε κάνει τις κατάλληλες συνδέσεις (αρχείο MainMenu.xib) στον Interface Builder ο οποίος είναι ενσωματωμένος με το περιβάλλον εργασίας του Xcode η επιλογή και φόρτωση του κατάλληλου view γίνεται σε αυτό το σημείο:

```
-(void)sideBar:(EDSideBar*)tabBar didSelectButton:(NSInteger)button
{
   switch (button) {
        // Insert code here to initialize your application
      case 0:
        self.rootViewController = [[RootViewController alloc] initWithNibName:@"RootViewController" bundle:nil];

        for (NSView *allViews in [self.currentView subviews])
        {
           [allViews removeFromSuperview];
        }
        [self.currentView addSubview:rootViewController.view];
        break;

      case 1:
        self.linearRegressionViewController = [[LinearRegressionViewController alloc] initWithNibName:@"LinearRegressionViewController" bundle:nil];
```

```
        for (NSView *allViews in [self.currentView subviews])
        {
            [allViews removeFromSuperview];
        }

        [self.currentView addSubview:linearRegressionViewController.view];
        break;

    case 2:
        self.interpolationViewController = [[InterpolationViewController alloc]
initWithNibName:@"InterpolationViewController" bundle:nil];
        for (NSView *allViews in [self.currentView subviews])
        {
            [allViews removeFromSuperview];
        }

        [self.currentView addSubview:interpolationViewController.view];
        break;

    default:
        NSLog(@"Something went bad.");

        break;
  }

}
```

Έτσι, όταν ο χρήστης επιλέξει το 3ο κουμπί (case 2), τότε αφαιρούνται τα υπάρχοντα Views και στη συνέχεια φορτώνεται το View της παρεμβολής (interpolationViewController.view).

5.5 Επεξήγηση αρχείων RootViewController.h/ RootViewController.m



Εικόνα 5.5.1. Το RootViewController.xib. Αποτελείται από NSTextFields, NSSegmentedControl, NSTableViews και ένα CPTXYGraph για τη σχεδίαση της γραφικής παράστασης.

Στο αρχείο header δηλώνουμε όλα τα απαραίτητα αντικείμενα τα οποία συνθέτουν το RootViewController όπως φαίνεται παρακάτω:

```
@interface RootViewController : NSViewController
<NSApplicationDelegate,NSTextFieldDelegate,CPTPlotSpaceDelegate,CPTPlotDataSou
rce,CPTScatterPlotDelegate>
{
   NSMutableArray *iterations;
   NSMutableArray *rootArray;
   NSMutableArray *samples;
   GCMathParser* parser;
   int startPoint,endPoint;
}
@property (unsafe_unretained) IBOutlet NSTextField *startPointTextField;
@property (unsafe_unretained) IBOutlet NSTextField *endPointTextField;

@property (unsafe_unretained) IBOutlet NSTextField *rootFunctionLabel;
@property (unsafe_unretained) IBOutlet NSTextField *functionField;
@property (unsafe_unretained) IBOutlet NSTextField *totalIterationsValueLabel;
@property (unsafe_unretained) IBOutlet NSTextField *valueFunctionLabel;
@property (unsafe_unretained) IBOutlet NSTextField *startValTextField;
@property (unsafe_unretained) IBOutlet NSTextField *endValTextField;
@property (unsafe_unretained) IBOutlet NSTextField *derivativeTextField;
@property (unsafe_unretained) IBOutlet NSTextField *maxIterations;
@property (unsafe_unretained) IBOutlet NSTableView *iterationsAndRootsView;
@property (unsafe_unretained) IBOutlet CPTGraphHostingView
*defaultLayerHostingView;
@property (unsafe_unretained) IBOutlet NSSegmentedControl *selectedMethod;
@property (unsafe_unretained) IBOutlet NSTextField *derivativeFunctionLabel;
@property (nonatomic,assign) NSWindow *window;

// - binded value to NSSlider
@property (readwrite,assign) int max; // Maximum number of iterations

-(IBAction)fetchSelectedMethod:(id)sender;
-(IBAction)clearResults:(id)sender;
-(IBAction)displayGraph:(id)sender;
-(IBAction)methodChanged:(id)sender;

-(BOOL)checkIfFuncIsEntered;

-(void)alertValuesMissing;
```

```
-(void)alertSameSign;

// Root Calculation Methods
-(void)calculateFalsiMethod;
-(void)calculateBisectionMethod;
-(void)calculateSecantMethod;
-(void)calculateNewtonMethod;
-(void)clearData;

-(BOOL) checkForSameSign: (double) x1 and: (double) x2;
-(void) printIterations: (int) x withRoot: (double) y andDiffer: (double) k andValue:
(double) z;
-(void) parseFunction;

@end
```

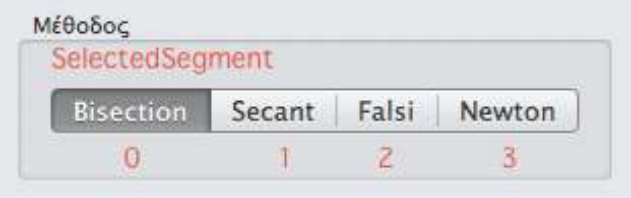
Η λέξη κλειδί 'IBOutlet' και 'IBAction' είναι ένα macro το οποίο επιτρέπει μεταβλητές και μεθόδους να αναφέρονται στον Interface Builder για να συνδέσουν στοιχεία User Interface με τον κώδικα.

Σημαντικές συναρτήσεις:

```
-(IBAction)fetchSelectedMethod:(id)sender
{
   if ([selectedMethod selectedSegment] == 3)
   {
      // Check only if we have entered a function and the first start point
      if ([[functionField stringValue] length] == 0 || ([[startValTextField stringValue]
length] == 0) || ([[derivativeTextField stringValue] length] == 0) )
      {
         [self alertValuesMissing];
         return;

      }
      else
      {
         [self calculateNewtonMethod];
         return;
      }
   }

   if ( [self checkIfFuncIsEntered] == YES )
   {
```

```
        if ([selectedMethod selectedSegment] == 0)
        {
            [self calculateBisectionMethod];
        }
        else if ([selectedMethod selectedSegment] == 1)
        {
            [self calculateSecantMethod];
        }
        else if ([selectedMethod selectedSegment] == 2)
        {
            [self calculateFalsiMethod];
        }
    }
    else
    {
        [self alertValuesMissing];
    }
}
```

Σε αυτήν την συνάρτηση ελέγχουμε ποιο από τα τέσσερα segments έχει επιλέξει ο χρήστης, στη συνέχεια ελέγχουμε αν ο χρήστης έχει εισάγει μια συνάρτηση (και παράγωγο συνάρτησης αν πρόκειται για μέθοδο Newton), καθώς και αν έχει εισάγει αρχικές τιμές. Τέλος, αν όλα έχουν εισαχθεί σωστά, καλούμε τις αντίστοιχες συναρτήσεις και υπολογίζουμε τις ρίζες.



Εικόνα 5.5.2 Περιγραφή των τιμών του Segmented Control

```
-(void) parseFunction
{
   double length = (endPoint - startPoint);
   double delta = length / (NUM_SAMPLES - 1);

   samples = [[NSMutableArray alloc] initWithCapacity:NUM_SAMPLES];
   parser = [GCMathParser parser];

   NSString *fx = [functionField stringValue];

   for (int i = 0; i < NUM_SAMPLES; i++)
   {
      double x = startPoint + (delta * i);
      [parser setSymbolValue:x forKey:@"x"];
      double y = [parser evaluate:fx];

      NSDictionary *pointsDictionary = [NSDictionary dictionaryWithObjectsAndKeys:
                        [NSNumber numberWithDouble:x],X_VAL,
                        [NSNumber numberWithDouble:y],Y_VAL,
                        nil];

      [samples addObject:pointsDictionary];
   }

}
```

Σε αυτήν την συνάρτηση αρχικοποιούμε τον parser και τον πίνακα samples με χωρητικότητα 200 αντικειμένων. Πρόσθετα, εκχωρούμε τη συνάρτηση που έχει δώσει ο χρήστης σε ένα αλφαριθμητικό και σε ένα βρόχο 200 επαναλήψεων, αποθηκεύουμε σε ένα λεξικό (NSDictionary) τις τιμές x,y οι οποίες προκύπτουνε από τον πολλαπλασιασμό της συνάρτησης με το εκάστοτε i. Τέλος, οι τιμές αυτές χρησιμοποιούνται αργότερα για τον σχεδιασμό της γραφικής παράστασης.

5.6 Επεξήγηση αρχείων LinearRegressionViewController .h/ LinearRegressionViewController .m

```
-(void)plotDistinctPoints
{
   NSMutableArray *tempArray = [NSMutableArray arrayWithCapacity:[maxDataPoints
intValue]];
   // Fetch values from textfield.
   for ( NSUInteger i = 1; i <= [maxDataPoints intValue]; i++ ) {
      float x = [[self.xValues cellWithTag:i] floatValue];
      // Find min and max numbers and use them later for the graph.
      minNumber = [[self.xValues cellWithTag:1] floatValue];
      maxNumber = [[self.xValues cellWithTag:1] floatValue];
      if (x < minNumber) {
         minNumber = x;
      }
      if (x > maxNumber) {
         maxNumber = x;
      }
      float y = [[self.yValues cellWithTag:i] floatValue];
      [tempArray addObject:[NSMutableDictionary
dictionaryWithObjectsAndKeys:[NSNumber numberWithFloat:x], @"x", [NSNumber
numberWithFloat:y], @"y", nil]];
   }

   distinctMarksData= [NSArray arrayWithArray:tempArray];

}
```

Η παραπάνω συνάρτηση αποθηκεύει τα σημεία x,y τα οποία ο χρήστης έχει εισάγει σε έναν προσωρινό πίνακα (tempArray). Στη συνέχεια, αποθηκεύουμε αυτόν τον πίνακα στον πίνακα distinctMarksData, όπου και ανακτούμε αργότερα αυτά τα δεδομένα μέσω του dataSource, για να σχεδιάσουμε αυτά τα σημεία.

```
-(void)plotLeastSquareLine
{
   // Parse the function.
   parser = [GCMathParser parser];

   NSMutableArray *tmpArray = [NSMutableArray new];

   for (float i=(minNumber-5) ; i <(maxNumber+5); i+= 0.05)
   {
      [parser setSymbolValue:i forKey:@"x"];
```

```
        float y = [parser evaluate:[NSString stringWithFormat:@"%0.3f*x + %0.3f",m,b]];
        [tmpArray addObject:[NSMutableDictionary
dictionaryWithObjectsAndKeys:[NSNumber numberWithFloat:i], @"x", [NSNumber
numberWithFloat:y], @"y", nil]];
    }
    leastSquareData= [NSArray arrayWithArray:tmpArray];
}
```

Σε αυτήν την συνάρτηση υπολογίζουμε την ευθεία ελαχίστων τετραγώνων η οποία αποθηκεύεται στον πίνακα leastSquareData. Χρησιμοποιώντας τον GCMathParser η ευθεία αυτή διέρχεται από τα σημεία τα οποία όπως προαναφέρθηκε είναι αποθηκευμένα στον πίνακα distinctMarksData.

5.7 Επεξήγηση αρχείων InterpolationViewController .h/InterpolationViewController.m

```
#pragma mark - TableView Data Source.
- (NSInteger)numberOfRowsInTableView:(NSTableView *)aTableView
{
    return [xValuesMutArray count];
}

-(id)tableView:(NSTableView *)tableView
objectValueForTableColumn:(NSTableColumn *)tableColumn row:(NSInteger)row
{
    if ([[tableColumn identifier] isEqualToString:@"x"]) {
        return [xValuesMutArray objectAtIndex:row];
    }
    if ([[tableColumn identifier] isEqualToString:@"y"]) {
        return [yValuesMutArray objectAtIndex:row];
    }
    if ([[tableColumn identifier] isEqualToString:@"y1"]) {
        return [y1MutArray objectAtIndex:row];
    }
    if ([[tableColumn identifier] isEqualToString:@"y2"]) {
        return [y2MutArray objectAtIndex:row];
    }
    if ([[tableColumn identifier] isEqualToString:@"y3"]) {
        return [y3MutArray objectAtIndex:row];
    }
    if ([[tableColumn identifier] isEqualToString:@"y4"]) {
        return [y4MutArray objectAtIndex:row];
    }

    return nil;
}
```

Εδώ χρησιμοποιώντας το data source από το TableView εμφανίζουμε αναλόγως το αναγνωριστικό της στήλης τις αντίστοιχες τιμές του εκάστοτε πίνακα στην κατάλληλη γραμμή (row) του TableView.

6. Βιβλιογραφία

1 Wikipedia, the free encyclopedia. Διαθέσιμο στη διεύθυνση http:// http://el.wikipedia.org/wiki/Apple. [πρόσβαση 14-5-2012]

2 Wikipedia, the free encyclopedia. Διαθέσιμο στη διεύθυνση http://el.wikipedia.org/wiki/Mac_OS_X [πρόσβαση 14-5-2012]

3 Learn Objective-C, Cocoa Dev Central. Διαθέσιμο στη διεύθυνση http://cocoadevcentral.com/d/learn_objectivec/ . [πρόσβαση 4-5-2012]

4 Beginning ARC in iOS 5 Tutorial Part 1. Διαθέσιμο στη διεύθυνση http://www.raywenderlich.com/5677/beginning-arc-in-ios-5-part-1 [πρόσβαση 10-7-2012]

5 Aaron Hillegass, Adam Preble. *Cocoa Programming for Mac OS X*, 4th edition, 2011

6 Apple Xcode IDE, Διαθέσιμο στη διεύθυνση https://developer.apple.com/xcode/

7 core-plot, Cocoa plotting framework for OS X and iOS , Διαθέσιμο στη διεύθυνση http://code.google.com/p/core-plot/, [πρόσβαση 10-6-2012]

8 EDSideBar Control, Διαθέσιμο στη διεύθυνση https://github.com/erndev/EDSidebar [πρόσβαση 7-11-2011]

7. Παράρτημα

Πηγαίος κώδικας:

```
//
//  RootViewController.h
//
//
//  Created by Symeon Paraschoudis on 8/28/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <Cocoa/Cocoa.h>
#import <CorePlot/CorePlot.h>

@class GCMathParser;
@class EDSidebarAppDelegate;

@interface RootViewController : NSViewController
<NSApplicationDelegate,NSTextFieldDelegate,CPTPlotSpaceDelegate,CPTPlotDataSource,CPT
ScatterPlotDelegate>
{
    NSMutableArray *iterations;
    NSMutableArray *rootArray;
    NSMutableArray *samples;

    GCMathParser* parser;

    int startPoint,endPoint;
}
@property (unsafe_unretained) IBOutlet NSTextField *startPointTextField;
@property (unsafe_unretained) IBOutlet NSTextField *endPointTextField;

@property (unsafe_unretained) IBOutlet NSTextField *rootFunctionLabel;
@property (unsafe_unretained) IBOutlet NSTextField *functionField;
@property (unsafe_unretained) IBOutlet NSTextField *totalIterationsValueLabel;
@property (unsafe_unretained) IBOutlet NSTextField *valueFunctionLabel;
@property (unsafe_unretained) IBOutlet NSTextField *startValTextField;
@property (unsafe_unretained) IBOutlet NSTextField *endValTextField;
@property (unsafe_unretained) IBOutlet NSTextField *derivativeTextField;
@property (unsafe_unretained) IBOutlet NSTextField *maxIterations;
@property (unsafe_unretained) IBOutlet NSTableView *iterationsAndRootsView;
@property (unsafe_unretained) IBOutlet CPTGraphHostingView *defaultLayerHostingView;
@property (unsafe_unretained) IBOutlet NSSegmentedControl *selectedMethod;
@property (unsafe_unretained) IBOutlet NSTextField *derivativeFunctionLabel;
@property (nonatomic,assign) NSWindow *window;

// - binded value to NSSlider
@property (readwrite,assign) int max; // Maximum number of iterations
```

```
-(IBAction)fetchSelectedMethod:(id)sender;
-(IBAction)clearResults:(id)sender;
-(IBAction)displayGraph:(id)sender;
-(IBAction)methodChanged:(id)sender;

-(BOOL)checkIfFuncIsEntered;

-(void)alertValuesMissing;
-(void)alertSameSign;

// Root Calculation Methods
-(void)calculateFalsiMethod;
-(void)calculateBisectionMethod;
-(void)calculateSecantMethod;
-(void)calculateNewtonMethod;
-(void)clearData;

-(BOOL) checkForSameSign: (double) x1 and: (double) x2;
-(void) printIterations: (int) x withRoot: (double) y andDiffer: (double) k andValue: (double) z;

-(void) parseFunction;


@end
//
//  RootViewController.m
//
//
//  Created by Symeon Paraschoudis on 8/28/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "RootViewController.h"
#import "GCMathParser.h"
#import "EDSidebarAppDelegate.h"

#define NUM_SAMPLES 200

#define X_VAL @"X_VAL"
#define Y_VAL @"Y_VAL"

@implementation RootViewController

@synthesize window = _window;
@synthesize defaultLayerHostingView,max;
@synthesize startPointTextField,endPointTextField;
@synthesize startValTextField,endValTextField;
@synthesize rootFunctionLabel,functionField,totalIterationsValueLabel;
@synthesize derivativeTextField,maxIterations,iterationsAndRootsView;
@synthesize selectedMethod,derivativeFunctionLabel,valueFunctionLabel;
```

```
- (id)init
{
    // call your superclass's designated initializer
    return [self initWithNibName:@"RootViewController" bundle:nil];
}

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Initialization code here.
        iterations = [[NSMutableArray alloc] init];
        rootArray = [[NSMutableArray alloc] init];
        // Initialize our slider..
        [self setValue:[NSNumber numberWithInt:50] forKey:@"max"];

    }
    return self;
}

//- (void)awakeFromNib
//{
//    [functionField setStringValue:@"exp(x) + 5*x - 13 "];
//    [derivativeTextField setStringValue:@"exp(x)+5"];
//}

- (void)dealloc
{
    iterations = nil;
    rootArray = nil;
    samples = nil;
}

-(IBAction)fetchSelectedMethod:(id)sender
{
    if ([selectedMethod selectedSegment] == 3)
    {
        // Check only if we have entered a function and the first start point
        if ([[functionField stringValue] length] == 0 || ([[startValTextField stringValue] length] ==
0) || ([[derivativeTextField stringValue] length] == 0) )
        {
            [self alertValuesMissing];
            return;

        }
        else
        {
            [self calculateNewtonMethod];
            return;
        }
    }
```

```
    if ( [self checkIfFuncIsEntered] == YES )
    {
        if ([selectedMethod selectedSegment] == 0)
        {
            [self calculateBisectionMethod];
        }
        else if ([selectedMethod selectedSegment] == 1)
        {
            [self calculateSecantMethod];
        }
        else if ([selectedMethod selectedSegment] == 2)
        {
            [self calculateFalsiMethod];
        }
    }
    else
    {
        [self alertValuesMissing];
    }
}

-(BOOL)checkIfFuncIsEntered
{
    // Has the user entered something?
    if ([[functionField stringValue]length] == 0 || ([[startValTextField stringValue]length] == 0)||
([[endValTextField stringValue]length] == 0) )
    {
        return NO;
    }

    return YES;

}

-(void)clearData
{
    // Wipe out the existing array contents
    [iterations removeAllObjects];
    [rootArray removeAllObjects];
    [iterationsAndRootsView reloadData];

    // Clear results
    [totalIterationsValueLabel setStringValue:@"0"];
    [rootFunctionLabel setStringValue:@"0"];
    [valueFunctionLabel setStringValue:@"0"];
}


-(void)displayGraph:(id)sender
{
```

```
    self.view = defaultLayerHostingView;
    [self initPlot];
}

-(IBAction)methodChanged:(id)sender
{
    switch ([sender selectedSegment]) {
        case 0:
            [derivativeFunctionLabel setTextColor:[NSColor disabledControlTextColor]];
            [derivativeTextField setEnabled:NO];
            [endValTextField setEnabled:YES];
            break;
        case 1:
            [derivativeFunctionLabel setTextColor:[NSColor disabledControlTextColor]];
            [derivativeTextField setEnabled:NO];
            [endValTextField setEnabled:YES];
            break;
        case 2:
            [derivativeFunctionLabel setTextColor:[NSColor disabledControlTextColor]];
            [derivativeTextField setEnabled:NO];
            [endValTextField setEnabled:YES];
            break;
        case 3:
            [derivativeFunctionLabel setTextColor:[NSColor selectedTextColor]];
            [derivativeTextField setEnabled:YES];
            [endValTextField setStringValue:@""];
            [endValTextField setEnabled:NO];
            break;

        default:
            break;
    }
}

- (IBAction)clearResults:(id)sender
{
    NSBeep();

    [functionField setStringValue:@""];
    [startValTextField setStringValue:@""];
    [endValTextField   setStringValue:@""];
    [iterations removeAllObjects];
    [iterationsAndRootsView reloadData];
}

#pragma mark - Alerts
-(void)alertValuesMissing
{
    NSAlert *alert = [NSAlert  alertWithMessageText:@"Σφάλμα"
                                      defaultButton:@"OK"
                                    alternateButton:nil
```

```
                                otherButton:nil
                  informativeTextWithFormat:@"Παρακαλώ εισάγετε την συνάρτηση και αρχικές
τιμές!"];

    [alert beginSheetModalForWindow:[self window]
                      modalDelegate:self
                     didEndSelector:nil contextInfo:NULL];

}

-(void)alertSameSign
{
    NSAlert *alert = [NSAlert alertWithMessageText:@"Σφάλμα χρησιμοποιώντας την μέθοδο
Falsi"
                                     defaultButton:@"Προσπαθήστε ξανά"
                                   alternateButton:nil
                                      otherButton:nil
                        informativeTextWithFormat:@"Η συνάρτηση έχει ίδια πρόσημα"];

    [alert beginSheetModalForWindow:[self window]
                      modalDelegate:self
                     didEndSelector:nil contextInfo:NULL];
}

#pragma mark - Calculate Numerical Methods
-(void)calculateFalsiMethod
{
    /*

     Original Source Code:
http://math.fullerton.edu/mathews/n2003/regulafalsi/RegulaFalsiProg/Links/RegulaFalsiProg_lnk
_4.html
     ------------------------------------------------------------------------
     Algo2-3.c   C program for implementing Algorithm 2.3
     Algorithm translated to  C  by: Dr. Norman Fahrer
     IBM and Macintosh verification by: Daniel Mathews

     NUMERICAL METHODS: C Programs, (c) John H. Mathews 1995
     To accompany the text:
     NUMERICAL METHODS for Mathematics, Science and Engineering, 2nd Ed, 1992
     Prentice Hall, Englewood Cliffs, New Jersey, 07632, U.S.A.
     Prentice Hall, Inc.; USA, Canada, Mexico ISBN 0-13-624990-6
     Prentice Hall, International Editions:   ISBN 0-13-625047-5
     This free software is compliments of the author.
     E-mail address:       in%"mathews@fullerton.edu"

     Algorithm 2.3 (False position or Regula Falsi Method).
     Section   2.2, Bracketing Methods for Locating a Root, Page 62
     ------------------------------------------------------------------------
     */
    /*
```

```
-------------------------------------------------------------------------

Algorithm 2.3 (False Position or Regula Method). To find a
root of the equation  f(x) = 0  in the interval [a,b].
Proceed with the method only if  f(x)  is continuous and  f(a)
and  f(b)  have opposite signs.

-------------------------------------------------------------------------
*/

double DX;          // change in iterate
double Delta = 1E-6;      // Closeness for consecutive iterates
double Epsilon = 1E-6;     // Tolerance for the size of f(C)
BOOL satisfied;

double YA, YB ;        // Function values at the interval-borders
int K;              // Loop Counter
double A, B;
double C = 0, YC = 0;        // new iterate and function value there

// User input - get initial values

A = [startValTextField doubleValue];
B = [endValTextField doubleValue];

// Parse the function and compute initial values
parser = [GCMathParser parser];

NSString *expression = [functionField stringValue];

[parser setSymbolValue:A forKey:@"x"];
YA = [parser evaluate:expression];

[parser setSymbolValue:B forKey:@"x"];
YB = [parser evaluate:expression];

// Check to see if YA and YB have same SIGN

if ([self checkForSameSign:YA and:YB] == YES) // Does not satisfy Bolzano theorem
{
   [self alertSameSign];
   [self clearData];
}
else // Different sign at end points, satisfies Bolzano.
{

   // Wipe out the existing array contents
   [rootArray removeAllObjects];
   [iterations removeAllObjects];

   for(K = 1; K <= max ; K++)
```

```
        {
            if( satisfied == TRUE)
                break;

            DX = YB * (B - A)/(YB -YA);      /* Change in iterate */
            C = B - DX;                      /* New iterate       */

            /* Function value of new iterate */
            [parser setSymbolValue:C forKey:@"x"];
            YC = [parser evaluate:expression];

            if( YC == 0) {          /* first 'if'          */
                satisfied = TRUE; /* Exact root is found */
            }
            else if( ( (YB >= 0) && (YC >=0) ) || ( (YB < 0) && (YC < 0) )   ) {
                B = C;      /* Squeeze from the right */
                YB = YC;
            }

            else {
                A = C;      /* Squeeze from the left */
                YA = YC;
            }

            [iterations addObject:[NSNumber numberWithInt:K]];
            [rootArray addObject:[NSNumber numberWithDouble:C]];
            // [NSString stringWithFormat:@"%0.15f",C]];


            if( (fabs(DX) < Delta) && (fabs(YC) < Epsilon) )
            {
                satisfied = TRUE;
            }
        }  /* end of 'for'-loop */

        [self printIterations:K-1 withRoot:C andDiffer:DX andValue:YC];

        // Update GUI

        [totalIterationsValueLabel setStringValue:[NSString stringWithFormat:@"%d", K-1]];
        [rootFunctionLabel setStringValue:[NSString stringWithFormat:@"%0.15lf",C]];
        [valueFunctionLabel setStringValue:[NSString stringWithFormat:@"%0.15lf",YC]];

        [iterationsAndRootsView reloadData];
    }

}

-(void)calculateBisectionMethod
{
    /*
```

```
--------------------------------------------------------------------------
Algo2-2.c   C program for implementing Algorithm 2.2
Algorithm translated to  C  by: Dr. Norman Fahrer
IBM and Macintosh verification by: Daniel Mathews

NUMERICAL METHODS: C Programs, (c) John H. Mathews 1995
To accompany the text:
NUMERICAL METHODS for Mathematics, Science and Engineering, 2nd Ed, 1992
Prentice Hall, Englewood Cliffs, New Jersey, 07632, U.S.A.
Prentice Hall, Inc.; USA, Canada, Mexico ISBN 0-13-624990-6
Prentice Hall, International Editions:   ISBN 0-13-625047-5
This free software is compliments of the author.
E-mail address:       in%"mathews@fullerton.edu"

Algorithm 2.2 (Bisection Method).
Section   2.2, Bracketing Methods for Locating a Root, Page 61
--------------------------------------------------------------------------
*/
/*
--------------------------------------------------------------------------

Algorithm 2.2 (Bisection Method). To find a root of the
equation  f(x) = 0  in the interval [a,b]. Proceed with the
method only if  f(x)  is continuous and  f(a)  and  f(b)  have
opposite signs.

--------------------------------------------------------------------------
*/

double Delta = 1E-6;         /* Tolerance for width of interval */
int Satisfied = 0;           /* Condition for loop termination  */
double A, B;                 /* Endpoints of the interval [A,B] */
double YA, YB;       /* Function values at the interval-borders */
//int Max=100;      /* Calculation of the maximum number of iterations */
int K;                                  /* Loop Counter */
double C, YC;   /* Midpoint of interval and function value there */

// User input - get initial values

A = [startValTextField doubleValue]; /* compute function values */
B = [endValTextField doubleValue];
max = [maxIterations intValue];

//Max = (int) ( 1 + floor( ( log(B-A) - log(Delta) ) / log(2) ) );
printf("Max = %d\n",max);

// Parse the function and compute initial values

parser = [GCMathParser parser];

NSString *expression = [functionField stringValue];
```

```
    [parser setSymbolValue:A forKey:@"x"];
    YA = [parser evaluate:expression];

    [parser setSymbolValue:B forKey:@"x"];
    YB = [parser evaluate:expression];

//    NSLog(@"YA = %.15lf", YA);
//    NSLog(@"YB = %.15lf", YB);

    /* Check to see if the bisection method applies */

    if( ( (YA >= 0) && (YB >=0) ) || ( (YA < 0) && (YB < 0) )   ) {
        printf("The values ffunction(A) and ffunction(B)\n");
        printf("do not differ in sign.\n");
        return;
    }
    else {
        // Clear existing objects
        [iterations removeAllObjects];
        [rootArray removeAllObjects];

        for(K = 1; K <= max ; K++) {

            if(Satisfied == 1)  break;

            C = (A + B) / 2;     /* Midpoint of interval */

            [parser setSymbolValue:C forKey:@"x"]; /* Function value at midpoint */
            YC = [parser evaluate:expression];


            if( YC == 0) {         /* first 'if' */
                A = C;       /* Exact root is found */
                B = C;
            }
            else if( ( (YB >= 0) && (YC >=0) ) || ( (YB < 0) && (YC < 0) )   ) {
                B = C;      /* Squeeze from the right */
                YB = YC;
            }
            else {
                A = C;      /* Squeeze from the left */
                YA = YC;
            }

            [iterations addObject:[NSNumber numberWithInt:K]];
            [rootArray addObject:[NSNumber numberWithDouble:C]];;

            if( (B-A) < Delta ) Satisfied = 1; /* check for early convergence */

        }  /* end of 'for'-loop */
```

```
        // Update GUI

        [totalIterationsValueLabel setStringValue:[NSString stringWithFormat:@"%d", K-1]];
        [rootFunctionLabel setStringValue:[NSString stringWithFormat:@"%0.15lf",C]];
        [valueFunctionLabel setStringValue:[NSString stringWithFormat:@"%0.15lf",YC]];

        [iterationsAndRootsView reloadData];

        NSLog(@"-----------------------------------------------");
        NSLog(@"The maximum number of iterations is : %d",max);
        NSLog(@"The number of performed iterations is : %d",K - 1);
        NSLog(@"-----------------------------------------------");
        NSLog(@"The computed root of f(x) = 0 is : %0.15lf ",C);
        NSLog(@"-----------------------------------------------");
        NSLog(@"The accuracy is +- %lf", B-A);
        NSLog(@"-----------------------------------------------");
        NSLog(@"The value of the function f(C) is %0.15lf\n",YC);
    }
}

-(void)calculateSecantMethod
{
    int n,m;
    double d ;

    double A, B;                /* Endpoints of the interval [A,B] */
    double YA, YB;      /* Function values at the interval-borders */

    m = [maxIterations intValue];

    A = [startValTextField doubleValue]; /* compute function values */
    B = [endValTextField doubleValue];

    parser = [GCMathParser parser];

    NSString *expression = [functionField stringValue];

    // Wipe out the existing array contents

    [iterations removeAllObjects];
    [rootArray removeAllObjects];

    for (n = 1; n <= m; n++)
    {
        [parser setSymbolValue:A forKey:@"x"];
        YA = [parser evaluate:expression];

        [parser setSymbolValue:B forKey:@"x"];
        YB = [parser evaluate:expression];
```

```
        d = (B - A) / ( (YB - YA) ) * YB;
        if (fabs(d) < 5E-11)
        {
            break;
        }

        A = B;
        B = B - d;
        printf("\nIteration %d:   root: %0.15lf",n, B );

        [iterations addObject:[NSNumber numberWithInt:n]];
        [rootArray addObject:[NSNumber numberWithDouble:B]];
    }


    // Update GUI

    [totalIterationsValueLabel setStringValue:[NSString stringWithFormat:@"%d", n-1]];
    [rootFunctionLabel setStringValue:[NSString stringWithFormat:@"%0.15lf",B]];
    [valueFunctionLabel setStringValue:@"-"];

    [iterationsAndRootsView reloadData];

}

-(void)calculateNewtonMethod
{
    /*

     Source:
http://math.fullerton.edu/mathews/n2003/newtonsmethod/Newton'sMethodProg/Links/Newton's
MethodProg_lnk_4.html

     --------------------------------------------------------------------
     Algo2-5.c   C program for implementing Algorithm 2.5
     Algorithm translated to  C  by: Dr. Norman Fahrer
     IBM and Macintosh verification by: Daniel Mathews

     NUMERICAL METHODS: C Programs, (c) John H. Mathews 1995
     To accompany the text:
     NUMERICAL METHODS for Mathematics, Science and Engineering, 2nd Ed, 1992
     Prentice Hall, Englewood Cliffs, New Jersey, 07632, U.S.A.
     Prentice Hall, Inc.; USA, Canada, Mexico ISBN 0-13-624990-6
     Prentice Hall, International Editions:   ISBN 0-13-625047-5
     This free software is compliments of the author.
     E-mail address:       "mathews@fullerton.edu"

     Algorithm 2.5 (Newton-Raphson Iteration).
     Section   2.4, Newton-Raphson and Secant Methods, Page 84
     --------------------------------------------------------------------
     */
```

```
/*
 ---------------------------------------------------------------------------

 Algorithm 2.5 (Newton-Raphson Iteration). To find a root
 f(x) = 0 given one initial approximation  p_0  and using the iteration

 f(p_(k-1))
 p_k  =  p_(k-1)  -  -------------     for k = 1, 2, ...
 f'(p_(k-1))

 ---------------------------------------------------------------------------
 */

double Delta = 1E-6;       /* Tolerance  */
double Epsilon = 1E-6;     /* Tolerance  */
double Small = 1E-6;       /* Tolerance  */

int Max = [maxIterations intValue];  /* Maximum number of iterations  */
int Cond = 0;  /* Condition fo loop termination */
int K;         /* Counter for loop              */

double P0;    /* INPUT : Must be close to the root */
double P1;    /* New iterate    */
double Y0;    /* Function value */
double Y1;    /* Function value */
double Df;    /* Derivative     */
double Dp;
double RelErr;

/*
printf("-----------------------------------------------\n");
printf("Please enter initial approximation of root !\n");
scanf("%lf",&P0);
printf("-----------------------------------------------\n");
printf("Initial value for root: %lf\n",P0); */

//iterations = [[NSMutableArray alloc] init];

P0 = [startValTextField doubleValue];

parser = [GCMathParser parser];

NSString *expression = [functionField stringValue];
NSString *derivative = [derivativeTextField stringValue];

[parser setSymbolValue:P0 forKey:@"x"];
Y0 = [parser evaluate:expression];

// Wipe out the existing array contents

[iterations removeAllObjects];
```

```
[rootArray removeAllObjects];

for ( K = 1; K <= Max ; K++) {

   if(Cond) break;

   [parser setSymbolValue:P0 forKey:@"x"]; /* Compute the derivative  */
   Df = [parser evaluate:derivative];

   if( Df == 0) {       /* Check division by zero */
      Cond = 1;
      Dp   = 0;
   }

   else Dp = Y0/Df;

   P1 = P0 - Dp;       /* New iterate */


   [parser setSymbolValue:P1 forKey:@"x"]; /* New function value */
   Y1 = [parser evaluate:expression];

   RelErr = 2 * fabs(Dp) / ( fabs(P1) + Small );  /* Relative error */

   if( (RelErr < Delta)  && (fabs(Y1) < Epsilon) ) { /* Check for   */

      if( Cond != 1) Cond = 2;                  /* convergence */

   }

   [iterations addObject:[NSNumber numberWithInt:K]];
   [rootArray addObject:[NSNumber numberWithDouble:P1]];

   printf("\nstep: %d ,  root: %0.15lf",K,P1);

   P0 = P1;
   Y0 = Y1;
}

// Update GUI
[totalIterationsValueLabel setStringValue:[NSString stringWithFormat:@"%d", K-1]];
[rootFunctionLabel setStringValue:[NSString stringWithFormat:@"%0.15lf",P1]];
[valueFunctionLabel setStringValue:[NSString stringWithFormat:@"%0.15lf",Y1]];

[iterationsAndRootsView reloadData];

printf("\n---------------------------------------------\n");
printf("The current %d -th iterate is %0.15lf\n",K-1, P1);
printf("Consecutive iterates differ by %lf\n",Dp);
printf("The value of f(x) is %0.15lf\n",Y1);
printf("---------------------------------------------\n");
```

```
    if (Cond == 0)
    {
        NSAlert *alert = [NSAlert  alertWithMessageText:@"Σφάλμα"
                                          defaultButton:@"OK"
                                        alternateButton:nil
                                           otherButton:nil
                             informativeTextWithFormat:@"Ο αριθμός μέγιστων επαναλήψεων έχει
ξεπεραστεί"];

        [alert beginSheetModalForWindow:[self window]
                          modalDelegate:self
                         didEndSelector:nil contextInfo:NULL];

    }


    if (Cond == 1)
    {
        NSAlert *alert = [NSAlert  alertWithMessageText:@"Σφάλμα"
                                          defaultButton:@"OK"
                                        alternateButton:nil
                                           otherButton:nil
                             informativeTextWithFormat:@"Αδύνατη διαίρεση με το μηδέν."];

        [alert beginSheetModalForWindow:[self window]
                          modalDelegate:self
                         didEndSelector:nil contextInfo:NULL];

    }

    if(Cond == 2) printf("The root was found with the desired tolerance !\n");

    printf("-------------------------------------------\n");
}


-(BOOL) checkForSameSign: (double) x1 and: (double) x2
{
    // x1,x2: Function values at the interval-borders
    // Check to see if x1 and x2 have same SIGN

    if( ( (x1 >= 0) && (x2 >=0) ) || ( (x1 < 0) && (x2 < 0) )   ) {
        NSLog(@"The values %lf and %lf",x1,x2);
        NSLog(@"do not differ in sign.");
        return YES;     // Display the NSAlert window
    }
    return NO;
}

-(void) printIterations: (int) x withRoot: (double) y andDiffer: (double) k andValue: (double) z
```

```
{
    NSLog(@"-----------------------------------------------");
    NSLog(@"The number of performed iterations is : %d",x);
    NSLog(@"-----------------------------------------------");
    NSLog(@"The computed root of f(x) = 0 is : %0.15lf",y);
    NSLog(@"-----------------------------------------------");
    NSLog(@"Consecutive iterates differ by %lf", k);
    NSLog(@"-----------------------------------------------");
    NSLog(@"The value of the function f(C) is %lf",z);
}

#pragma mark NSTable Data Source

- (NSInteger)numberOfRowsInTableView:(NSTableView *)tableView
{
    return [iterations count];
}

- (id)tableView:(NSTableView *)tableView
objectValueForTableColumn:(NSTableColumn *)tableColumn
            row:(int)row
{
    id value = nil;

    if ([[tableColumn identifier] isEqualToString:@"iterations"])
    {
        value = [iterations objectAtIndex:row];
    }
    else
    {
        value = [rootArray objectAtIndex:row];
    }

    return value;
}

#pragma mark - Core Plot Functions

-(void) parseFunction
{
    double length = (endPoint - startPoint);
    double delta = length / (NUM_SAMPLES - 1);

    samples = [[NSMutableArray alloc] initWithCapacity:NUM_SAMPLES];
    parser = [GCMathParser parser];

    NSString *fx = [functionField stringValue];

    for (int i = 0; i < NUM_SAMPLES; i++)
    {
        double x = startPoint + (delta * i);
```

```
        [parser setSymbolValue:x forKey:@"x"];
        double y = [parser evaluate:fx];

        NSDictionary *pointsDictionary = [NSDictionary dictionaryWithObjectsAndKeys:
                        [NSNumber numberWithDouble:x],X_VAL,
                        [NSNumber numberWithDouble:y],Y_VAL,
                        nil];

        [samples addObject:pointsDictionary];
    }

}

#pragma mark - Chart behavior
-(void)initPlot
{
    startPoint = [startPointTextField intValue];
    endPoint = [endPointTextField intValue];

    if (startPoint > endPoint || (startPoint == 0 && endPoint == 0) || [[functionField stringValue]
isEqualToString:@""])
    {
        NSAlert *alert = [NSAlert  alertWithMessageText:@"Σφάλμα σχεδιασμού γραφικής
παράστασης"
                                  defaultButton:@"OK"
                                alternateButton:nil
                                   otherButton:nil
                     informativeTextWithFormat:@"Μη έγκυρες τιμές!"];

        [alert beginSheetModalForWindow:[self window]
                          modalDelegate:self
                         didEndSelector:nil contextInfo:NULL];

        return;
    }
    else
    {
        [self parseFunction];
        [self configureGraph];
        [self configurePlots];
        [self configureAxes];
    }

}

-(void)configureGraph
{
    // 1 - Create the graph
    CPTGraph *graph = [[CPTXYGraph alloc]
initWithFrame:self.defaultLayerHostingView.bounds];
    [graph applyTheme:[CPTTheme themeNamed:kCPTPlainWhiteTheme]];
```

```
    self.defaultLayerHostingView.hostedGraph = graph;
    // 2 - Set graph title
    NSString *title = [functionField stringValue];
    graph.title = title;
    // 3 - Create and set text style
    CPTMutableTextStyle *titleStyle = [CPTMutableTextStyle textStyle];
    titleStyle.color = [CPTColor whiteColor];
    titleStyle.fontName = @"Helvetica-Bold";
    titleStyle.fontSize = 16.0f;
    graph.titleTextStyle = titleStyle;
    graph.titlePlotAreaFrameAnchor = CPTRectAnchorTop;
    graph.titleDisplacement = CGPointMake(0.0f, 10.0f);
    // 4 - Set padding for plot area
    [graph.plotAreaFrame setPaddingLeft:30.0f];
    [graph.plotAreaFrame setPaddingBottom:30.0f];
    // 5 - Enable user interactions for plot space
    CPTXYPlotSpace *plotSpace = (CPTXYPlotSpace *) graph.defaultPlotSpace;
    plotSpace.allowsUserInteraction = YES;
}

-(void)configurePlots
{
    // 1 - Get graph and plot space
    CPTGraph *graph = self.defaultLayerHostingView.hostedGraph;
    CPTXYPlotSpace *plotSpace = (CPTXYPlotSpace *) graph.defaultPlotSpace;
    // 2 - Create the three plots
    CPTScatterPlot *functionPlot = [[CPTScatterPlot alloc] init];
    functionPlot.dataSource = self;
    functionPlot.identifier = @"DefaultPlot";
    CPTColor *functionColor = [CPTColor redColor];
    [graph addPlot:functionPlot toPlotSpace:plotSpace];
    // 3 - Set up plot space
    [plotSpace scaleToFitPlots:[NSArray arrayWithObjects:functionPlot,nil]];
    CPTMutablePlotRange *xRange = [plotSpace.xRange mutableCopy];
    [xRange expandRangeByFactor:CPTDecimalFromCGFloat(1.1f)];
    plotSpace.xRange = xRange;
    CPTMutablePlotRange *yRange = [plotSpace.yRange mutableCopy];
    [yRange expandRangeByFactor:CPTDecimalFromCGFloat(1.2f)];
    plotSpace.yRange = yRange;
    // 4 - Create styles and symbols
    CPTMutableLineStyle *aaplLineStyle = [functionPlot.dataLineStyle mutableCopy];
    aaplLineStyle.lineWidth = 2.5;
    aaplLineStyle.lineColor = functionColor;
    functionPlot.dataLineStyle = aaplLineStyle;
    CPTMutableLineStyle *aaplSymbolLineStyle = [CPTMutableLineStyle lineStyle];
    aaplSymbolLineStyle.lineColor = functionColor;
    CPTPlotSymbol *aaplSymbol = [CPTPlotSymbol plotSymbol];
    aaplSymbol.fill = [CPTFill fillWithColor:functionColor];
    aaplSymbol.lineStyle = aaplSymbolLineStyle;
    aaplSymbol.size = CGSizeMake(6.0f, 6.0f);
    functionPlot.plotSymbol = aaplSymbol;
```

```
}

-(void)configureAxes
{
    // Axes
    // Label x axis with a fixed interval policy
    CPTXYAxisSet *axisSet = (CPTXYAxisSet *)defaultLayerHostingView.hostedGraph.axisSet;
    CPTXYAxis *x          = axisSet.xAxis;
    x.majorIntervalLength         = CPTDecimalFromString(@"1.0");
    x.orthogonalCoordinateDecimal = CPTDecimalFromString(@"0.0");
    x.minorTicksPerInterval       = 5;

    x.title        = @"X Axis";
    x.titleOffset  = 30.0;
    x.titleLocation = CPTDecimalFromString(@"1.25");

    // Label y with an automatic label policy.
    CPTXYAxis *y = axisSet.yAxis;
    y.labelingPolicy              = CPTAxisLabelingPolicyAutomatic;
    y.orthogonalCoordinateDecimal = CPTDecimalFromString(@"0.0");
    y.minorTicksPerInterval       = 2.0;
    y.preferredNumberOfMajorTicks = 8;
    y.labelOffset                 = 10.0;

    y.title        = @"Y Axis";
    y.titleOffset  = 30.0;
    y.titleLocation = CPTDecimalFromString(@"1.0");
}

#pragma mark -
#pragma mark Plot Data Source Methods

-(NSUInteger)numberOfRecordsForPlot:(CPTPlot *)plot;
{
    return NUM_SAMPLES;
}

-(NSNumber *)numberForPlot:(CPTPlot *)plot field:(NSUInteger)fieldEnum
             recordIndex:(NSUInteger)index;
{
    NSDictionary *sample = [samples objectAtIndex:index];

    if (fieldEnum == CPTScatterPlotFieldX)
        return [sample valueForKey:X_VAL];
    else
        return [sample valueForKey:Y_VAL];

}

#pragma mark - NSTextField delegates
```

```
- (BOOL)control:(NSControl *)control textView:(NSTextView *)fieldEditor
doCommandBySelector:(SEL)commandSelector
{
    BOOL retval = NO;

    if (commandSelector == @selector(insertNewline:)) { // did the user press enter?
        retval = YES; // causes Apple to NOT fire the default enter action
        [self displayGraph:nil];
    }
    //NSLog(@"Selector = %@", NSStringFromSelector( commandSelector ) );
    return retval;
}

@end
```

```
//
//  linearRegressionView.h
//  NumericalMethods
//
//  Created by Συμεών Π. on 9/10/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <Foundation/Foundation.h>
#import <math.h>
#import <CorePlot/CorePlot.h>
#import "EDSidebarAppDelegate.h"
#import "GCMathParser.h"

@interface LinearRegressionViewController : NSViewController
<NSApplicationDelegate,CPTPlotSpaceDelegate,CPTPlotDataSource,
CPTScatterPlotDelegate>
{
    IBOutlet NSPopUpButton *numsSelected;

    IBOutlet NSMatrix *xValues;
    IBOutlet NSMatrix *yValues;
    IBOutlet CPTGraphHostingView *defaultLayerHostingView;
    CPTPlotSpaceAnnotation *symbolTextAnnotation;

    NSMutableArray *graphs;
    NSArray *distinctMarksData;
    NSArray *leastSquareData;

    NSString *title;
    NSString *currentThemeName;

    GCMathParser* parser;

    double   b;                    /* y-intercept of best fit line  */
    double   m;                     /* slope of best fit line       */

    float minNumber,maxNumber;
}

@property (nonatomic,retain) NSMutableArray *iterationsArray;
@property (nonatomic,retain) NSMutableArray *contentArray;

@property (strong) IBOutlet NSMatrix *xValues;
@property (strong) IBOutlet NSMatrix *yValues;

@property  (readwrite, assign) id maxDataPoints; // Maximum number of iterations

@property (nonatomic,strong)    CPTGraphHostingView *defaultLayerHostingView;
@property (nonatomic,strong)    NSMutableArray *graphs;
@property (nonatomic,strong)    NSString *title;
```

```
@property (nonatomic,strong)    CPTPlotSpaceAnnotation *symbolTextAnnotation;
@property (nonatomic, copy)     NSString *currentThemeName;

-(IBAction)markChanged:(id)sender;
-(IBAction)calculateRegression:(id)sender;
-(double) sqr:(double)x;

-(void)plotDistinctPoints;
-(void)plotLeastSquareLine;
-(void)clearPreviousGraph;

@end
```

```
//
//  linearRegressionView.m
//  NumericalMethods
//
//  Created by Συμεών Π. on 9/10/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "LinearRegressionViewController.h"
#import "EDSidebarAppDelegate.h"

@implementation LinearRegressionViewController

@class EDSidebarAppDelegate;

NSString *  const CPDTickerSymbolMarks      = @"Marks";
NSString *  const CPDTickerSymbolLine       = @"Line";

@synthesize iterationsArray;
@synthesize xValues,yValues,maxDataPoints;
@synthesize defaultLayerHostingView,graphs,title,symbolTextAnnotation,currentThemeName;
@synthesize contentArray;

-(id)init
{
   return [self initWithNibName:@"LinearRegressionViewController" bundle:nil];
}

-(id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
   self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
   if (self)
   {
      // Initialization code here.
      self.contentArray = [[NSMutableArray alloc] init];
      self.graphs = [[NSMutableArray alloc] init];
      self.iterationsArray = [[NSMutableArray alloc] init];

      for (int i = 1; i <= 10; i++)
      {
         [iterationsArray addObject:[NSNumber numberWithInt:i]];
      }
      // Set initial value
      maxDataPoints = [iterationsArray objectAtIndex:0];
   }

   return self;
}

-(IBAction)calculateRegression:(id)sender
{
```

```
    [self clearPreviousGraph];

    NSArray * xCells = [xValues cells];
    NSArray *yCells = [yValues cells];

/*****************************************************************************/
    /*                                                           */
    /*                      L I N R E G                          */
    /*                                                           */
    /*  Program:      LINREG                                     */
    /*                                                           */
    /*  Programmer:   Dr. David G. Simpson                       */
    /*                Department of Physical Science             */
    /*                Prince George's Community College          */
    /*                Largo, Maryland  20774                     */
    /*                                                           */
    /*  Date:         January 21, 2002                           */
    /*                                                           */
    /*  Language:     C                                          */
    /*                                                           */
    /*  Description:  This program performs a linear regression analysis for a   */
    /*                set of data given as (x,y) pairs.  The output from the     */
    /*                program is the slope and y-intercept of the least-squares  */
    /*                best fit straight line through the data points.            */
    /*                                                           */
/*****************************************************************************/

    // Source: http://www.pgccphy.net/Linreg/linreg.html

/*****************************************************************************/
    /*  function prototypes                                      */
/*****************************************************************************/

    //void   chop (char *str);              /* remove \n from end of string  */
    //double sqr (double x);                /* compute the square of a number*/



/*****************************************************************************/
    /*  global variables                                         */
/*****************************************************************************/

    double   n = 0.0;                   /* number of data points        */
    double   r;                       /* correlation coefficient      */
    double   sumx = 0.0;                  /* sum of x                   */
```

```
double   sumx2 = 0.0;                /* sum of x**2              */
double   sumxy = 0.0;                /* sum of x * y             */
double   sumy = 0.0;                 /* sum of y                 */
double   sumy2 = 0.0;                /* sum of y**2              */
double   x;                       /* input x data             */
double   y;                       /* input y data             */

/*--------------------------------------------------------------------*/
/*  Print introductory message.                                       */
/*--------------------------------------------------------------------*/

NSLog(@"LINREG - Perform linear regression");

/*--------------------------------------------------------------------*/
/*  Enter data and accumulate sums.                                   */
/*--------------------------------------------------------------------*/

for (int i =0; i < [maxDataPoints intValue]; i++)      /* loop for all data points      */
{
   x = [[xCells objectAtIndex:i] doubleValue];
   y = [[yCells objectAtIndex:i] doubleValue];

   NSDecimalNumber *xNumber = [[NSDecimalNumber alloc] initWithDouble:x];
   NSDecimalNumber *yNumber = [[NSDecimalNumber alloc] initWithDouble:y];
   [contentArray addObject:[NSMutableDictionary dictionaryWithObjectsAndKeys:xNumber,
@"x", yNumber, @"y", nil]];

   n += 1.0;                      /* increment num of data points  */
   sumx  += x;                    /* compute sum of x              */
   sumx2 += x * x;                /* compute sum of x**2           */
   sumxy += x * y;                /* compute sum of x * y          */
   sumy  += y;                    /* compute sum of y              */
   sumy2 += y * y;                /* compute sum of y**2           */
}                              /* loop again for more data      */

/*--------------------------------------------------------------------*/
/*  Compute least-squares best fit straight line.                     */
/*--------------------------------------------------------------------*/

double sumx_square = [self sqr:sumx];
double sumy_square = [self sqr:sumy];


m = (n * sumxy  -  sumx * sumy) /       /* compute slope                */
(n * sumx2 - sumx_square);

b = (sumy * sumx2  -  sumx * sumxy) /   /* compute y-intercept          */
(n * sumx2  -  sumx_square);

r = (sumxy - sumx * sumy / n) /         /* compute correlation coeff    */
sqrt((sumx2 - sumx_square/n) *
```

```
        (sumy2 - sumy_square/n));

    /*------------------------------------------------------------------------*/
    /*  Print results and return to operating system.                         */
    /*------------------------------------------------------------------------*/

    NSLog(@"nSlope       m = %0.8f",m);
    NSLog(@"y-intercept  b = %0.8f",b);
    NSLog(@"Correlation  r = %0.8f",r);

    self.title = [NSString stringWithFormat:@"y = %0.3fx + %0.3f",m,b];

    [self plotDistinctPoints];
    [self plotLeastSquareLine];
    [self initPlot];
}

-(double) sqr:(double)x
{
    return (x * x);                     /* compute square of argument    */
}

-(IBAction)markChanged:(id)sender
{
    // Interpolation: 2-10 max marks.
    maxDataPoints = [iterationsArray objectAtIndex:[sender selectedTag]];

    [self.xValues setEnabled:NO];
    [self.yValues setEnabled:NO];

    // Enable the selected textfields.
    for (int i = 1; i <= [maxDataPoints intValue]; i++) {
        [[self.xValues cellWithTag:i] setEnabled:YES];
        [[self.yValues cellWithTag:i] setEnabled:YES];
    }

}

-(void)plotDistinctPoints
{
    NSMutableArray *tempArray = [NSMutableArray arrayWithCapacity:[maxDataPoints
intValue]];
    // Fetch values from textfield.
    for ( NSUInteger i = 1; i <= [maxDataPoints intValue]; i++ ) {
        float x = [[self.xValues cellWithTag:i] floatValue];
        // Find min and max numbers and use them later for the graph.
        minNumber = [[self.xValues cellWithTag:1] floatValue];
        maxNumber = [[self.xValues cellWithTag:1] floatValue];
        if (x < minNumber) {
            minNumber = x;
        }
```

```
        if (x > maxNumber) {
            maxNumber = x;
        }
        float y = [[self.yValues cellWithTag:i] floatValue];
        [tempArray addObject:[NSMutableDictionary dictionaryWithObjectsAndKeys:[NSNumber
numberWithFloat:x], @"x", [NSNumber numberWithFloat:y], @"y", nil]];
    }

    distinctMarksData = [NSArray arrayWithArray:tempArray];

}

-(void)plotLeastSquareLine
{
    // Parse the function.
    parser = [GCMathParser parser];

    NSMutableArray *tmpArray = [NSMutableArray new];

    for (float i=(minNumber-5) ; i <(maxNumber+5); i+= 0.05)
    {
        [parser setSymbolValue:i forKey:@"x"];
        float y = [parser evaluate:[NSString stringWithFormat:@"%0.3f*x + %0.3f",m,b]];
        [tmpArray addObject:[NSMutableDictionary dictionaryWithObjectsAndKeys:[NSNumber
numberWithFloat:i], @"x", [NSNumber numberWithFloat:y], @"y", nil]];
    }
    leastSquareData = [NSArray arrayWithArray:tmpArray];
}

-(void)clearPreviousGraph
{
    if ([graphs count])
    {
        [graphs removeAllObjects];
        if ( symbolTextAnnotation )
        {
            symbolTextAnnotation = nil;
        }
    }
}

-(void)dealloc
{
    distinctMarksData = nil;
    leastSquareData = nil;
    iterationsArray = nil;
    contentArray= nil;
}

#pragma mark - Chart behavior
-(void)initPlot
```

```
{
    [self configureHost];
    [self configureGraph];
    [self configurePlots];
    [self configureAxes];
}

-(void)configureHost
{
    [self.view addSubview:self.defaultLayerHostingView];
}

-(void)configureGraph
{
    // 1 - Create the graph
    CPTGraph *graph = [[CPTXYGraph alloc]
initWithFrame:self.defaultLayerHostingView.bounds];
    [graph applyTheme:[CPTTheme themeNamed:kCPTSlateTheme]];
    self.defaultLayerHostingView.hostedGraph = graph;
    // 2 - Set graph title
    graph.title = self.title;
    // 3 - Create and set text style
    CPTMutableTextStyle *titleStyle = [CPTMutableTextStyle textStyle];
    titleStyle.color = [CPTColor whiteColor];
    titleStyle.fontName = @"Helvetica-Bold";
    titleStyle.fontSize = 16.0f;
    graph.titleTextStyle = titleStyle;
    graph.titlePlotAreaFrameAnchor = CPTRectAnchorTop;
    graph.titleDisplacement = CGPointMake(0.0f, 10.0f);
    // 4 - Set padding for plot area
    [graph.plotAreaFrame setPaddingLeft:30.0f];
    [graph.plotAreaFrame setPaddingBottom:30.0f];
    // 5 - Enable user interactions for plot space
    CPTXYPlotSpace *plotSpace = (CPTXYPlotSpace *) graph.defaultPlotSpace;
    plotSpace.allowsUserInteraction = YES;

    // Add the graph
    [graphs addObject:graph];
}

-(void)configurePlots
{
    // 1 - Get graph and plot space
    CPTGraph *graph = self.defaultLayerHostingView.hostedGraph;
    CPTXYPlotSpace *plotSpace = (CPTXYPlotSpace *) graph.defaultPlotSpace;
    // 2 - Create the three plots
    CPTScatterPlot *marksPlot = [[CPTScatterPlot alloc] init];
    marksPlot.dataSource = self;
    marksPlot.delegate = self;
    marksPlot.identifier = CPDTickerSymbolMarks;
    CPTColor *aaplColor = [CPTColor blackColor];
```

```
    [graph addPlot:marksPlot toPlotSpace:plotSpace];
    CPTScatterPlot *linePlot = [[CPTScatterPlot alloc] init];
    linePlot.dataSource = self;
    linePlot.identifier = CPDTickerSymbolLine;
    CPTColor *googColor = [CPTColor redColor];
    [graph addPlot:linePlot toPlotSpace:plotSpace];
    // 3 - Set up plot space
    [plotSpace scaleToFitPlots:[NSArray arrayWithObjects:marksPlot, linePlot, nil]];
    CPTMutablePlotRange *xRange = [plotSpace.xRange mutableCopy];
    [xRange expandRangeByFactor:CPTDecimalFromCGFloat(1.1f)];
    plotSpace.xRange = xRange;
    CPTMutablePlotRange *yRange = [plotSpace.yRange mutableCopy];
    [yRange expandRangeByFactor:CPTDecimalFromCGFloat(1.2f)];
    plotSpace.yRange = yRange;
    // 4 - Create styles and symbols
    CPTMutableLineStyle *aaplLineStyle = [marksPlot.dataLineStyle mutableCopy];
    aaplLineStyle.lineWidth = 3.0;
    aaplLineStyle.lineColor = [CPTColor clearColor];
    marksPlot.dataLineStyle = aaplLineStyle;
    CPTMutableLineStyle *aaplSymbolLineStyle = [CPTMutableLineStyle lineStyle];
    aaplSymbolLineStyle.lineColor = aaplColor;
    CPTPlotSymbol *aaplSymbol = [CPTPlotSymbol ellipsePlotSymbol];
    aaplSymbol.fill = [CPTFill fillWithColor:aaplColor];
    aaplSymbol.lineStyle = aaplSymbolLineStyle;
    aaplSymbol.size = CGSizeMake(6.0f, 6.0f);
    marksPlot.plotSymbol = aaplSymbol;
    CPTMutableLineStyle *googLineStyle = [linePlot.dataLineStyle mutableCopy];
    googLineStyle.lineWidth = 3.0;
    googLineStyle.lineColor = googColor;
    linePlot.dataLineStyle = googLineStyle;
    CPTMutableLineStyle *googSymbolLineStyle = [CPTMutableLineStyle lineStyle];
    googSymbolLineStyle.lineColor = [CPTColor redColor];
}

-(void)configureAxes
{
    // Grid line styles
    CPTMutableLineStyle *majorGridLineStyle = [CPTMutableLineStyle lineStyle];
    majorGridLineStyle.lineWidth = 0.75;
    majorGridLineStyle.lineColor = [[CPTColor colorWithGenericGray:0.2]
colorWithAlphaComponent:0.75];

    CPTMutableLineStyle *minorGridLineStyle = [CPTMutableLineStyle lineStyle];
    minorGridLineStyle.lineWidth = 0.25;
    minorGridLineStyle.lineColor = [[CPTColor whiteColor] colorWithAlphaComponent:0.1];

    // Axes
    // Label x axis with a fixed interval policy
    CPTXYAxisSet *axisSet = (CPTXYAxisSet *)defaultLayerHostingView.hostedGraph.axisSet;
    CPTXYAxis *x          = axisSet.xAxis;
    x.majorIntervalLength         = CPTDecimalFromString(@"5.0");
```

```
    x.orthogonalCoordinateDecimal = CPTDecimalFromString(@"0.0");
    x.minorTicksPerInterval       = 5;
    x.majorGridLineStyle          = majorGridLineStyle;
    x.minorGridLineStyle          = minorGridLineStyle;

    x.title        = @"X Axis";
    x.titleOffset  = 30.0;
    x.titleLocation = CPTDecimalFromString(@"1.25");

    // Label y with an automatic label policy.
    CPTXYAxis *y = axisSet.yAxis;
    y.labelingPolicy              = CPTAxisLabelingPolicyAutomatic;
    y.orthogonalCoordinateDecimal = CPTDecimalFromString(@"0.0");
    y.minorTicksPerInterval       = 2.0;
    y.preferredNumberOfMajorTicks = 8;
    y.majorGridLineStyle          = majorGridLineStyle;
    y.minorGridLineStyle          = minorGridLineStyle;
    y.labelOffset                 = 10.0;

    y.title        = @"Y Axis";
    y.titleOffset  = 30.0;
    y.titleLocation = CPTDecimalFromString(@"1.0");

}

#pragma mark - CPTPlotDataSource methods
-(NSUInteger)numberOfRecordsForPlot:(CPTPlot *)plot {
    if ([plot.identifier isEqual:CPDTickerSymbolMarks] == YES)
    {
        return [distinctMarksData count];
    }
    else if ([plot.identifier isEqual:CPDTickerSymbolLine] == YES)
    {
        return [leastSquareData count];
    }
    return nil;
}

-(NSNumber *)numberForPlot:(CPTPlot *)plot field:(NSUInteger)fieldEnum
recordIndex:(NSUInteger)index
{
    switch (fieldEnum)
    {
        case CPTScatterPlotFieldX:
            if ([plot.identifier isEqual:CPDTickerSymbolMarks] == YES)
            {
                return [[distinctMarksData objectAtIndex:index] valueForKey:@"x"];
            }
            else if ([plot.identifier isEqual:CPDTickerSymbolLine] == YES)
            {
                return [[leastSquareData objectAtIndex:index] valueForKey:@"x"];
```

```
            }
            break;

        case CPTScatterPlotFieldY:
            if ([plot.identifier isEqual:CPDTickerSymbolMarks] == YES)
            {
                return [[distinctMarksData objectAtIndex:index] valueForKey:@"y"];
            }
            else if ([plot.identifier isEqual:CPDTickerSymbolLine] == YES)
            {
                return [[leastSquareData objectAtIndex:index] valueForKey:@"y"];
            }
            break;
    }

    return nil;
}

#pragma mark CPTScatterPlot delegate method

-(void)scatterPlot:(CPTScatterPlot *)plot
plotSymbolWasSelectedAtRecordIndex:(NSUInteger)index
{
    CPTGraph *graph = [graphs objectAtIndex:0];

    if ( symbolTextAnnotation ) {
        [graph.plotAreaFrame.plotArea removeAnnotation:symbolTextAnnotation];
        symbolTextAnnotation = nil;
    }

    // Setup a style for the annotation
    CPTMutableTextStyle *hitAnnotationTextStyle = [CPTMutableTextStyle textStyle];
    hitAnnotationTextStyle.color    = [CPTColor whiteColor];
    hitAnnotationTextStyle.fontSize = 16.0f;
    hitAnnotationTextStyle.fontName = @"Helvetica-Bold";

    // Determine point of symbol in plot coordinates
    NSNumber *x         = [[distinctMarksData objectAtIndex:index] valueForKey:@"x"];
    NSNumber *y         = [[distinctMarksData objectAtIndex:index] valueForKey:@"y"];
    NSArray *anchorPoint = [NSArray arrayWithObjects:x, y, nil];

    // Add annotation
    // First make a string for the y value
    NSNumberFormatter *formatter = [[NSNumberFormatter alloc] init];
    [formatter setMaximumFractionDigits:2];
    NSString *yString = [formatter stringFromNumber:y];

    // Now add the annotation to the plot area
    CPTTextLayer *textLayer = [[CPTTextLayer alloc] initWithText:yString
style:hitAnnotationTextStyle];
```

```
    symbolTextAnnotation             = [[CPTPlotSpaceAnnotation alloc]
initWithPlotSpace:graph.defaultPlotSpace anchorPlotPoint:anchorPoint];
    symbolTextAnnotation.contentLayer = textLayer;
    symbolTextAnnotation.displacement = CGPointMake(0.0f, 20.0f);
    [graph.plotAreaFrame.plotArea addAnnotation:symbolTextAnnotation];
}

@end
```

```
//
//  interpolationView.h
//  NumericalMethods
//
//  Created by Symeon Paraschoudis on 9/5/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <Foundation/Foundation.h>
#import <CorePlot/CorePlot.h>

@interface InterpolationViewController : NSViewController
<NSApplicationDelegate,NSTableViewDelegate,NSTableViewDataSource>
{
    NSNumber *selectedTextFields;

    IBOutlet NSSegmentedControl *selectedMethod;
    IBOutlet NSPopUpButton *numsSelected;

    // Text Fields
    IBOutlet NSTextField *xValueInterpolatedTextField;
    IBOutlet NSTextField *yValueInterpolatedTextField;

    // Newton's divided dif. box
    IBOutlet NSBox *newtonDivDifBox;
    IBOutlet NSTableView *newtonsMethodTbView;
}
@property (strong) IBOutlet NSTextField *xTextField;
@property (strong) IBOutlet NSPopUpButton *numsSelected;

@property (strong) IBOutlet NSMatrix *xValues;
@property (strong) IBOutlet NSMatrix *yValues;

@property (strong)  NSMutableArray *xValuesMutArray;
@property (strong)  NSMutableArray *yValuesMutArray;

@property (strong)  NSMutableArray *y1MutArray;
@property (strong)  NSMutableArray *y2MutArray;
@property (strong)  NSMutableArray *y3MutArray;
@property (strong)  NSMutableArray *y4MutArray;

@property  (nonatomic,assign) id maxDataPoints; // Maximum number of iterations

@property (nonatomic, copy) NSString *selectedMarks;
@property (nonatomic, readwrite, strong) NSMutableArray *iterationsArray;

@property (assign) IBOutlet NSWindow *window;

-(id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil;

-(IBAction)markChanged:(id)sender;
```

```
-(IBAction)calculateMethodPressed:(id)sender;

-(void)calculateLagrange;
-(void)calculateNewton;

@end
```

```
//
//  interpolationView.m
//  NumericalMethods
//
//  Created by Symeon Paraschoudis on 9/5/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "InterpolationViewController.h"

@implementation InterpolationViewController

@synthesize numsSelected,maxDataPoints,iterationsArray;
@synthesize selectedMarks,xValues,yValues;
@synthesize xValuesMutArray,yValuesMutArray;
@synthesize y1MutArray,y2MutArray,y3MutArray,y4MutArray;
@synthesize window;

- (id)init
{
    return [self initWithNibName:@"InterpolationViewController" bundle:nil];
}

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];

    if (self)
    {
        // then do your own initializing
        xValuesMutArray = [[NSMutableArray alloc] init];
        yValuesMutArray = [[NSMutableArray alloc] init];
        y1MutArray = [[NSMutableArray alloc] initWithCapacity:[yValuesMutArray count]];
        y2MutArray = [[NSMutableArray alloc] init];
        y3MutArray = [[NSMutableArray alloc] init];
        y4MutArray = [[NSMutableArray alloc] init];
        iterationsArray = [[NSMutableArray alloc] init];

        for (int i = 1; i <= 8; i++) {

            [iterationsArray addObject:[NSNumber numberWithInteger:i]];
        }

        // Set initial value
        maxDataPoints = [iterationsArray objectAtIndex:1];
    }

    return self;
}
```

```
-(void)dealloc
{
    self.xValuesMutArray = nil;
    self.yValuesMutArray = nil;
    self.y1MutArray = nil;
    self.y2MutArray = nil;
    self.y3MutArray = nil;
    self.y4MutArray = nil;
}

-(void)awakeFromNib
{
}


-(IBAction)markChanged:(id)sender
{
    [self.xValues setEnabled:NO];
    [self.yValues setEnabled:NO];

    // Interpolation: 2-10 max marks.
    maxDataPoints = [iterationsArray objectAtIndex:[sender selectedTag]];

    // Enable the selected textfields.
    for (int i = 1; i <= [maxDataPoints intValue]; i++) {
        [[self.xValues cellWithTag:i] setEnabled:YES];
        [[self.yValues cellWithTag:i] setEnabled:YES];
    }
}

-(IBAction)calculateMethodPressed:(id)sender
{
    if ([selectedMethod selectedSegment] == 0)
    {
        [self calculateLagrange];

    }
    else if ([selectedMethod selectedSegment] == 1)
    {
        [self calculateNewton];
    }
}

#pragma mark - Lagrange,Newton,Splines implemention

-(void)calculateLagrange
{
    /* Download this program from  www.vtubooks.com                    */
    /* File name : lagrangs.cpp                                        */

    /*----------------- LAGRANGE'S INTERPOLATION  METHOD ----------------------*/
```

```
/*    THE PROGRAM CALCULATES THE VALUE OF f(x) AT GIVEN VALUE OF x
 USING LAGRANGE'S INTERPOLATION METHOD.

 INPUTS :  1) Number of entries of the data.

 2) Values of 'x' & corresponding y = f(x).

 3) Value of 'xr' at which y = f(x) to be calculated.

 OUTPUTS : Interpolated value f(x) at x = xr.                  */

/*-----------------------------    PROGRAM  -----------------------------*/

NSArray * xCells = [xValues cells];
NSArray *yCells = [yValues cells];

double x[10],y[10],xr,fy,num,den;
int i,j;

NSLog(@"LAGRANGE'S INTERPOLATION TECHNIQUE");

for(i = 0; i < [maxDataPoints intValue]; i++)
{
   /*  LOOP TO GET x AND y = f(x) IN THE ARRAY */
   x[i] = [[xCells objectAtIndex:i] doubleValue];
   y[i] = [[yCells objectAtIndex:i] doubleValue];

}

xr = [xValueInterpolatedTextField doubleValue];

fy = 0;
for(j = 0; j < [maxDataPoints intValue]; j++)
{
   /*  LOOP TO CALCULATE LAGRANGE'S INTERPOLATION  */
   num = den = 1;
   for(i = 0; i < [maxDataPoints intValue]; i++)
   {
      if(i == j) continue;
      num = num * (xr - x[i]);
      den = den * (x[j] - x[i]);
   }
   fy = fy + ((num/den) * y[j]);
}

NSLog(@"The value of y = f(x) at xr = %lf is yr = %lf", xr,fy);

[yValueInterpolatedTextField setStringValue:[NSString stringWithFormat:@"%lf",fy]];
}
```

```
-(void)calculateNewton
{
   // Author: Matthew Evans
   // Original Source Code: http://www.dailyfreecode.com/code/newtons-divided-difference-
method-2381.aspx

   // Check if we already have objects.
   if ([xValuesMutArray count] > 0)
   {
      [xValuesMutArray removeAllObjects];
      [yValuesMutArray removeAllObjects];
      [y1MutArray removeAllObjects];
      [y2MutArray removeAllObjects];
      [y3MutArray removeAllObjects];
      [y4MutArray removeAllObjects];
   }

   NSArray * xCells = [xValues cells];
   NSArray *yCells = [yValues cells];

   float x[10],y[10][10],sum,p,temp;
   int i,j,k=0,f;
   float fact(int);

   for(i=0; i< [maxDataPoints intValue]; i++)
   {
      // Read the values from our matrix.

      x[i] = [[xCells objectAtIndex:i] floatValue];
      [xValuesMutArray addObject:[NSNumber numberWithFloat:x[i]]];

      y[k][i] = [[yCells objectAtIndex:i] floatValue];
      [yValuesMutArray addObject:[NSNumber numberWithFloat:y[k][i]]];
   }

   p = [xValueInterpolatedTextField floatValue];

   if (p == 0)
   {
      [self alertValuesMissing];
      return;
   }

   for(i=1;i<[maxDataPoints intValue];i++)
   {
      k=i;
      for(j=0;j<[maxDataPoints intValue]-i;j++)
      {
         y[i][j]=(y[i-1][j+1]-y[i-1][j])/(x[k]-x[j]);
         k++;
      }
```

```
}
printf("\n_____\n");
printf("\n  x(i)\t   y(i)\t   y1(i)   y2(i)   y3(i)  y4(i)");
printf("\n_____\n");
for(i=0;i<[maxDataPoints intValue];i++)
{
   printf("\n %.3f",x[i]);
   for(j=0;j<[maxDataPoints intValue]-i;j++)
   {
      if (j == 1) {
         [y1MutArray addObject:[NSNumber numberWithFloat:y[j][i]] ];
      }
      else if (j ==2) {
         [y2MutArray addObject:[NSNumber numberWithFloat:y[j][i]] ];
      }
      else if (j ==3) {
         [y3MutArray addObject:[NSNumber numberWithFloat:y[j][i]] ];
      }
      else if (j ==4) {
         [y4MutArray addObject:[NSNumber numberWithFloat:y[j][i]] ];
      }

      printf("    ");
      printf(" %.3f",y[j][i]);
   }
   printf("\n");
}

i=0;
do
{
   if(x[i]<p && p<x[i+1])
      k=1;
   else
      i++;
}while(k != 1);
f=i;

sum=0;
for(i=0;i<[maxDataPoints intValue]-1;i++)
{
   k=f;
   temp=1;
   for(j=0;j<i;j++)
   {
      temp = temp * (p - x[k]);
      k++;
   }
   sum = sum + temp*(y[i][f]);
}
```

```
    printf("\n\n f(%.2f) = %f ",p,sum);

    [yValueInterpolatedTextField setStringValue:[NSString stringWithFormat:@"%lf",sum]];

    // Add nil values so as to match the yValuesMutArray capacity.

    for (int i = [y1MutArray count]; i < [yValuesMutArray count]; i++)
    {
        [y1MutArray addObject:[NSString stringWithFormat:@""]];
    }

    for (int i = [y2MutArray count]; i < [yValuesMutArray count]; i++)
    {
        [y2MutArray addObject:[NSString stringWithFormat:@""]];
    }

    for (int i = [y3MutArray count]; i < [yValuesMutArray count]; i++)
    {
         [y3MutArray addObject:[NSString stringWithFormat:@""]];
    }

    for (int i = [y4MutArray count]; i < [yValuesMutArray count]; i++)
    {
        [y4MutArray addObject:[NSString stringWithFormat:@""]];
    }

    [newtonsMethodTbView reloadData];

}

#pragma mark - Alerts
-(void)alertValuesMissing
{
    NSAlert *alert = [NSAlert  alertWithMessageText:@"Σφάλμα"

                                      defaultButton:@"OK"
                                    alternateButton:nil
                                       otherButton:nil
                         informativeTextWithFormat:@"Παρακαλώ εισάγετε σημείο εξαγωγής"];

    [alert beginSheetModalForWindow:[self window]
                      modalDelegate:self
                     didEndSelector:nil contextInfo:NULL];

}

#pragma mark - TableView Data Source.

- (NSInteger)numberOfRowsInTableView:(NSTableView *)aTableView
{
    return [xValuesMutArray count];
```

```
}

-(id)tableView:(NSTableView *)tableView objectValueForTableColumn:(NSTableColumn
*)tableColumn row:(NSInteger)row
{
    if ([[tableColumn identifier] isEqualToString:@"x"]) {
        return [xValuesMutArray objectAtIndex:row];
    }
    if ([[tableColumn identifier] isEqualToString:@"y"]) {
        return [yValuesMutArray objectAtIndex:row];
    }
    if ([[tableColumn identifier] isEqualToString:@"y1"]) {
        return [y1MutArray objectAtIndex:row];
    }
    if ([[tableColumn identifier] isEqualToString:@"y2"]) {
        return [y2MutArray objectAtIndex:row];
    }
    if ([[tableColumn identifier] isEqualToString:@"y3"]) {
        return [y3MutArray objectAtIndex:row];
    }
    if ([[tableColumn identifier] isEqualToString:@"y4"]) {
        return [y4MutArray objectAtIndex:row];
    }

    return nil;
}

@end
```

```
//
//  AboutAppViewController.h
//  NumericalMethods
//
//  Created by Symeon Paraschoudis on 6/26/13.
//
//

#import <Cocoa/Cocoa.h>

@interface AboutAppViewController : NSViewController

@end
```

```
//
//  AboutAppViewController.m
//  NumericalMethods
//
//  Created by Symeon Paraschoudis on 6/26/13.
//
//

#import "AboutAppViewController.h"

@interface AboutAppViewController ()

@end

@implementation AboutAppViewController

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Initialization code here.
    }

    return self;
}

@end
```