Ανάπτυξη πρωτότυπου πακέτου λογισμικού για σύγκριση αλγορίθμων ως προς τον υπολογιστικό τους φόρτο.

ΓΙΑΝΝΗΣ ΗΛΙΑΔΗΣ

Πτυχιακή εργασία που υποβλήθηκε για μερική εκπλήρωση των απαιτήσεων για την απόκτηση του πτυχίου του

Μηχανικού Πληροφορικής

Τμήμα Πληροφορικής
Σχολή Τεχνολογικών Εφαρμογών
ΤΕΙ Αθήνας

Ιούνιος 1996

ΠΕΡΙΕΧΟΜΕΝΑ

1. Βασικές Έννοιες

1.1 Το πρόβλημα

Η αλματώδης πρόοδος στον τομέα της βιομηχανικής παραγωγής μονάδων κεντρικής επεξεργασίας (CPU) αλλά και η κάθετη πτώση στις τιμές των τσιπ σιλικόνης δημιούργησε μια νέα αγορά στον χώρο της Πληροφορικής. Μια αγορά που απαιτεί την ανάπτυξη λογισμικού που να εκμεταλλεύεται τις ολοένα και αυξανόμενες δυνατότητες και ταχύτητες των υπολογιστών που παράγονται. Πρέπει να συνειδητοποιήσουμε ότι η χρήση των δομημένων αλγορίθμων δεν αρκεί πια. Το αγοραστικό κοινό ζητά κάτι περισσότερο. Απαιτεί την υλοποίηση εφαρμογών εύκολα συντηρήσιμων αλλά και με τη μέγιστη δυνατή απόδοση. Πρέπει λοιπόν να επανεξετάσουμε ή μάλλον να επανεξετάζουμε ανά περίπτωση, την δομή των αλγορίθμων που χρησιμοποιούμε στις εφαρμογές που αναπτύσσουμε. Αυτό βέβαια απαιτεί περισσότερη έρευνα στον τομέα της ανάπτυξης λογισμικού. Έρευνα η οποία απαιτεί, με τη σειρά της, τη χρήση σύγχρονων και εξειδικευμένων εργαλείων προκειμένου να ολοκληρωθεί σε ένα λογικό χρονικό διάστημα αλλά και να αποδώσει τους αναμενόμενους καρπούς. Εργαλείων που να έχουν την δυνατότητα να αναλύουν αλγορίθμους και να παρέχουν μέτρα σύγκρισης για αυτούς, με την βοήθεια των οποίων να μπορούμε πια να επιλέγουμε τους βέλτιστους ανά περίπτωση αλγόριθμους, και στον ελάχιστο δυνατό χρόνο.

1.1.1 Ανάλυση αλγορίθμου

Η ανάλυση αλγορίθμων είναι ένας τομέας της επιστήμης της Πληροφορικής στον οποίο έχουν ήδη γίνει αρκετές μελέτες. Με τη βοήθεια της ανάλυσης αλγορίθμων μπορούμε, μεταξύ άλλων, να αξιολογούμε και να βελτιστοποιούμε τους αλγόριθμους που χρησιμοποιούμε κατά την ανάπτυξη των εφαρμογών μας. Αυτό επιτυγχάνεται διερευνώντας την πολυπλοκότητα (complexity) τους και υπολογίζοντας το φόρτο τους ανά περίπτωση. Μέχρι τώρα όμως αυτή η εργασία ήταν τις περισσότερες φορές χρονοβόρα και πολλές φορές περίπλοκη. Για αυτόν τον λόγο κυρίως δεν συνηθιζόταν στην ανάπτυξη εφαρμογών μικρής ή μεσαίας κλίμακας (utilities ή και εφαρμογές περιορισμένου ενδιαφέροντος που αναπτύσσονταν είτε από ιδιώτες είτε από μικρές ή μεσαίες εταιρίες λογισμικού). Τώρα πια, με τη βοήθεια του ACT, αυτό είναι δυνατό.

1.1.2 Πολυπλοκότητα

Η πολυπλοκότητα (complexity) ενός αλγορίθμου εκφράζεται από μία παραμετρική εξίσωση, με παραμέτρους τις διαστάσεις των μεταβλητών που λαμβάνουν μέρος στον αλγόριθμο. Βάσει αυτής μπορούμε να εκτιμήσουμε την απόδοση του αλγορίθμου αυτού κάτω από ορισμένες συνθήκες. Ο φόρτος του αλγορίθμου υπολογίζεται από την παραμετρική εξίσωση πολυπλοκότητας, αν αντικαταστήσουμε τις παραμέτρους με τις ισχύουσες ανά περίπτωση αριθμητικές διαστάσεις των μεταβλητών. Ο φόρτος λοιπόν , σε συνδυασμό με την πολυπλοκότητα ενός αλγορίθμου, αποτελούν δύο στοιχεία με τα οποία μπορούμε να συγκρίνουμε με ασφάλεια αλγορίθμους και να καταλήξουμε σε συμπεράσματα για την αποδοτικότητά τους.

1.2 Λύση

1.2.1 Algorithm Comparison Tool (ACT)

Η πρότυπη (prototype) αυτή εφαρμογή απευθύνεται σε επιστήμονες οποιουδήποτε τομέα, οι οποίοι επιθυμούν να αξιολογήσουν τους αλγορίθμους ή μεθόδους που χρησιμοποιούν στην επιστήμη τους. Αναπτύχθηκε για να καλύψει την έλλειψη βοηθητικών εργαλείων έρευνας στον τομέα της ανάλυσης αλγορίθμων. Μέσω αυτής μπορεί ο χρήστης να επιλέξει από μια ομάδα αλγορίθμων που εισάγει ο ίδιος και που επιτελούν το ίδιο έργο, τον καταλληλότερο για την δική του εργασία. Επίσης έχει την δυνατότητα να διορθώσει τυχόν συντακτικά λάθη στον αλγόριθμο του, για τα οποία πληροφορείται εκτενώς από το πρόγραμμα. Ας δούμε λίγο πιο αναλυτικά τις δυνατότητες που προσφέρει η εφαρμογή αυτή.

- **Δυνατότητα αποθήκευσης και ανάκτησης αλγορίθμων.** Ο χρήστης μπορεί να εισάγει αλγορίθμους μέσα στην εφαρμογή αυτή και να τους αποθηκεύει ή να τους ανακτά. Σε αυτό το επίπεδο του προγράμματος προτιμήθηκαν τα ASCII αρχεία προκειμένου να διατηρηθεί η δυνατότητα ενσωμάτωσης των αλγορίθμων σε άλλα προγράμματα ή πιθανώς και σε περιβάλλοντα ανάπτυξης εφαρμογών.

- **Εύρεση πολυπλοκότητας (complexity) αλγορίθμου.** Η εύρεση της παραμετρικής εξίσωσης που εκφράζει την πολυπλοκότητα του αλγορίθμου γίνεται κατά παραγγελία του χρήστη και χωρίς καμία περαιτέρω επέμβαση από μέρους του, αν ο αλγόριθμος είναι συντακτικά σωστός. Αν δεν ισχύει το τελευταίο, ζητείται η επέμβαση του χρήστη για την συντακτική διόρθωση του αλγορίθμου, όποτε αυτό κριθεί αναγκαίο.

- **Εύρεση φόρτου (burden) αλγορίθμου.** Η εύρεση του φόρτου του αλγορίθμου πραγματοποιείται αφού ο χρήστης εισάγει συγκεκριμένα αριθμητικά δεδομένα για τις διαστάσεις των μεταβλητών που λαμβάνουν μέρος στον αλγόριθμο. Ο φόρτος ενός αλγορίθμου εκφράζει την πολυπλοκότητά του, όταν οι μεταβλητές που περιέχει λαμβάνουν τις προαναφερθείσες, αριθμητικές διαστάσεις..

- **Βάση δεδομένων για την επεξεργασία των αλγορίθμων, των πολυπλοκοτήτων και των φόρτων τους.** Ενσωματωμένη στην εφαρμογή προκειμένου να μπορεί ο χρήστης να διαχειριστεί, στο σύνολό τους, τους αλγόριθμους που έχει κατά καιρούς εισάγει, να τους αναθεωρήσει, να τους συγκρίνει κατά ομάδες και να επιλέξει τον βέλτιστο ανά περίπτωση.

- **Διαγράμματα (charts).** Ο χρήστης έχει την δυνατότητα να δημιουργήσει διαγράμματα συγκριτικής παρουσίασης των αλγορίθμων που έχει ήδη εισάγει στην βάση, προκειμένου να έχει μια γραφική σύγκριση των φόρτων των αλγορίθμων αυτών. Είδη διαγραμμάτων που διατίθενται : Bar charts, Pie charts, Linear charts. Και τα τρία είδη αυτά προσφέρονται στο χρήστη σε δυσδιάστατη ή τρισδιάστατη μορφή. Τα διαγράμματα αυτά μπορούν να εξαχθούν και να χρησιμοποιηθούν από οποιοδήποτε άλλο πρόγραμμα που λειτουργεί στο περιβάλλον MS-Windows.

- **Context-sensitive Help.** Έχουν ενσωματωθεί κείμενα βοήθειας για τον χρήστη που του παρέχουν πληροφορίες για την εργασία που εκτελεί ανά πάσα στιγμή.

1.2.2 Υλοποίηση

Θα αναφερθούμε σε γενικές γραμμές εδώ στο πως υλοποιήθηκε αυτή η εφαρμογή. Για περισσότερες πληροφορίες μπορείτε να δείτε τα κεφάλαια *Ανάπτυξη* και *Προγραμματιστικές Τεχνικές*.

Η έκδοση της εφαρμογής που λειτουργείτε αυτή τη στιγμή λειτουργεί σε υπολογιστές IBM PC ή συμβατούς, και κάτω από το λειτουργικό σύστημα MS-Windows. Έχει κατασκευασθεί με τη βοήθεια των ακόλουθων περιβαλλόντων ανάπτυξης εφαρμογών :

- Borland Delphi v1.0

- Borland Pascal with Objects v7.0

- Borland Paradox v7.0

- Microsoft Fortran v5.1

Η ανάπτυξή της έγινε τμηματικά. Συγκεκριμένα, τα τμήματα που αναπτύχθηκαν και ενοποιήθηκαν στην πορεία είναι τα εξής :

- **I f διεπικοινωνία με τον χρήστη** . Το interface σχεδιάστηκε εξ' ολοκλήρου σε Borland Delphi v1.0. Παρέχει στον χρήστη τις τυπικές δυνατότητες ενός συντάκτη κειμένου (text editor) προκειμένου να απλοποιηθεί η εισαγωγή του αλγορίθμου και η συντήρησή του. Προτιμήθηκε η χρήση της συγκεκριμένης πλατφόρμας ανάπτυξης εφαρμογών γιατί αποτελεί ένα από τα πιο εξελιγμένα εργαλεία στον τομέα της υλοποίησης στις ημέρες μας. Εργαλεία τα οποία έχουν χαρακτηριστεί ως Rapid Application Development tools, επειδή επιτρέπουν στον προγραμματιστή να αναπτύξει σε σχετικά μικρό χρονικό διάστημα το σκελετό και το interface μιας εφαρμογής, δίνοντας του παράλληλα την δυνατότητα για εύκολη και γρήγορη μετέπειτα συντήρηση.

- **P /S συντακτικός/λεκτικός αναλυτής αλγορίθμου** . Υλοποιήθηκαν σε Borland Pascal with Objects v7.0 . Χρησιμοποιείται από το κυρίως πρόγραμμα προκειμένου να ελέγξει τη συντακτική και λεκτική ορθότητα του αλγορίθμου. Αν βρεθούν λάθη στον αλγόριθμο στην πορεία της ανάλυσης, και δεν μπορούν να διορθωθούν αυτόματα από τον parser/scanner καλείται ο χρήστης να τα διορθώσει, προκειμένου να προχωρήσει η ανάλυση.

- **Βιβλιοθήκες εύρεσης πολυπλοκότητας**. Έχουν δημιουργηθεί σε Microsoft Fortran v5.1 και έχουν μετατραπεί σε μορφή DLL. Με αυτόν τον τρόπο καλούνται από το κυρίως πρόγραμμα προκειμένου να υπολογίσουν την παραμετρική εξίσωση πολυπλοκότητας ενός αλγορίθμου, ο οποίος έχει ήδη αναλυθεί και είναι σωστός ως προς το συντακτικό και το λεκτικό του τμήμα.

- **Βάση δεδομένων**. Αναπτύχθηκε με σκοπό να αποθηκεύονται, ανακτώνται και επεξεργάζονται κατά ομάδες οι αλγόριθμοι που έχει εισάγει στο πρόγραμμα κατά καιρούς ο χρήστης. Αρχικά υλοποιήθηκαν οι πίνακες που την απαρτίζουν σε Borland Paradox και η διαχείρισή τους έγινε από το κυρίως πρόγραμμα και με την βοήθεια της Borland Database Engine (BDE).

- **Δημιουργία γραφημάτων**. Η ανάπτυξη γραφημάτων δημιουργήθηκε με τη βοήθεια του Borland Delphi v1.0 και των βιβλιοθηκών γραφικών που αυτό παρέχει. Ενσωματώθηκε μετέπειτα ως λειτουργία στο κυρίως πρόγραμμα.

- **Βιβλιοθήκες ενσωμάτωσης ρουτινών από DLL άλλων κατασκευαστών**. Βιβλιοθήκες που αναπτύχθηκαν με την βοήθεια του Borland Delphi v1.0 και οι οποίες προσφέρουν την δυνατότητα στο πρόγραμμα που τις χρησιμοποιεί να ενσωματώνει τις ρουτίνες βιβλιοθηκών DLL που έχουν κατασκευάσει άλλοι προγραμματιστές, χωρίς αυτοί να γνωρίζουν την εσωτερική αρχιτεκτονική του εν λόγω προγράμματος. Η ενσωμάτωση αυτή των ρουτινών γίνεται κατά την εκτέλεση του προγράμματος.

2. Ανάπτυξη πακέτου λογισμικού

2.1 Περιβάλλον Λειτουργίας

Η εφαρμογή αναπτύχθηκε αρχικά για τα λειτουργικά συστήματα **MS DOS** και **P DOS.** Την εποχή εκείνη όμως τα τεράστια τεχνολογικά άλματα που έγιναν στην βιομηχανία παραγωγής μονάδων κεντρικής επεξεργασίας (CPU) και η κάθετη πτώση στις τιμές των τσιπ υπολογιστών όσο και η εμφάνιση εργαλείων ταχείας ανάπτυξης εφαρμογών (Rapid Application Tools), όπως το Delphi, γέννησαν ένα νέο, πολύ πιο απαιτητικό κοινό χρηστών υπολογιστών. Έχοντας υπόψη όλα αυτά και προσπαθώντας να αποτελέσουμε ένα μέρος αυτής της ριζικής μεταμόρφωσης στον κόσμο των υπολογιστών, αποφασίσαμε να αλλάξουμε την πορεία μας. Και έτσι έγινε ! Η εφαρμογή μετατράπηκε ούτως ώστε **να λειτουργεί στο περιβάλλον MS W w** και νέες, βελτιωμένες εκδόσεις συνεχώς αναπτυσσόντουσαν...Μια ελαφρώς βελτιωμένη 32-bit έκδοση για τα λειτουργικά συστήματα Win95 και WinNT είναι μία από τις βραχυπρόθεσμες προοπτικές ανάπτυξης.
Η έκδοση της εφαρμογής που έχετε στα χέρια σας λειτουργεί σε υπολογιστές IBM PC ή συμβατούς, κάτω από το περιβάλλον MS-Windows 3.x. Βέβαια, μπορεί να λειτουργήσει και στα περιβάλλοντα Win95 και WinNT σαν μια 16-bit εφαρμογή, προς το παρόν.

2.2 Εργαλεία

Τα εργαλεία ανάπτυξης εφαρμογών που χρησιμοποιήθηκαν για την σε περιβάλλον MS-DOS ήταν τα εξής:

- **B T P** 7.0 (parser engine)
- **B T V** 2.0 for Turbo Pascal (Borland-like interface)
- **Mf F** 5.1 (βιβλιοθήκες για την εύρεση του φόρτου αλγορίθμου)
- **Mf A** 5.0 (μικρές ρουτίνες που γράφτηκαν σε Assembly για να αποφευχθεί το overhead και κάποια TSR όπως calculator, calendar ενσωματωμένα στην εφαρμογή)

Τα εργαλεία ανάπτυξης εφαρμογών που χρησιμοποιήθηκαν για την σε περιβάλλον MS-Windows ήταν τα εξής:

- **B D.** Interface
- **B T P 7.0 w O j** (parser engine). Στην πραγματικότητα, ολόκληρη η parser engine δημιουργήθηκε εξ' αρχής, σε αντικειμενοστραφή μορφή.
- **B P.** Δημιουργία της βάσης δεδομένων.
- **M f F** 5.1 Δημιουργία DLL βιβλιοθηκών για την εύρεση του φόρτου αλγορίθμου
- **Sfw If Mg** 2.6. Δημιουργία Help Files.

2.3 Προηγούμενες εκδόσεις

Όπως ήδη ίσως υποπτευθήκατε η εφαρμογή αυτή πέρασε από πολλά alpha-development στάδια προτού εξελιχθεί στην εφαρμογή που έχετε αυτή τη στιγμή στο υπολογιστικό σας σύστημα. Οι διάφορες εκδόσεις που δημιουργήθηκαν είναι οι εξής :

MSDOS / PDOS εκδόσεις

Έκδοση 1.0 : Μικρή parser engine, χωρίς να ελέγχει για λάθη χρήστη. Κανενός είδους παραθυρική διεπικοινωνία με τον χρήστη. Ίσως θα έπρεπε να έχει αριθμηθεί ως έκδοση 0.x ...

Έκδοση 1.1 : Προστέθηκαν κάποια TSR για να διευκολύνουν τυχόν αριθμητικούς υπολογισμούς που θα ήθελε να κάνει ο χρήστης

Έκδοση 2.0 : Δημιουργήθηκε απλή αλλά αποτελεσματική παραθυρική επικοινωνία με τον χρήστη. Χωρίς μενού ή πλαίσια διαλόγων. Μόνο μερικά παράθυρα εισόδου/εξόδου δεδομένων. Βελτιώθηκε η parser engine.

Έκδοση 3.0 : Όλη η εφαρμογή τμηματοποιείται σε πλήθος βιβλιοθηκών προκειμένου να απλοποιηθεί η αποσφαλμάτωση. Η διεπικοινωνία με τον χρήστη γίνεται λίγο πιο φιλική. Τώρα υπάρχει και κάποιο παράθυρο στο οποίο αναφέρονται βασικές οδηγίες χρήσης.

Έκδοση 3.1 έως 3.3 : Η parser engine βελτιώνεται κατά πολύ. Ένας αριθμός σφαλμάτων στην εισαγωγή του αλγόριθμου από τον χρήστη μπορεί τώρα πια να διορθωθεί με ευριστικό τρόπο, χωρίς την παρέμβαση του χρήστη. Δυστυχώς, όμως, η εφαρμογή αρχίζει να μην νιώθει τόσο άνετα με το φράγμα των 640K. "640K πρέπει να είναι αρκετά για τον οποιονδήποτε". Ποιος ήταν άραγε αυτός που είχε κάνει αυτήν την δήλωση το 1981 ;

Έκδοση 4.0 : Μεγάλες αλλαγές ! Η διεπικοινωνία με τον χρήστη τώρα διαχωρίζεται από το υπόλοιπο πρόγραμμα και αναπτύσσεται χωριστά σε Turbo Vision 2.0! Όλη η parser engine κατασκευάζεται εξ' αρχής, σε αντικειμενοστραφή μορφή αυτήν τη φορά, και με τέτοιο τρόπο που να εκμεταλλεύεται την Protected Memory. Το φράγμα των 640K ξεπερνιέται !

Έκδοση 4.0 έως 4.5 : Διάφορες αλλαγές στην parser engine. Ο χρήστης πια πληροφορείται με πλήρη μηνύματα λάθους οποτεδήποτε ανιχνευθεί ένα σφάλμα στον δοθέντα αλγόριθμο με τον οποίο τροφοδοτεί το πρόγραμμα. Ο parser αναγνωρίζει περισσότερες μεταβλητές και τώρα πια και συναρτήσεις !

Μετά από όλα αυτά ... Αποφασίσαμε ότι το πρόγραμμα έπρεπε να μετατραπεί ούτως ώστε να λειτουργεί κάτω από το περιβάλλον **MS-Windows** *!! Και όντως έτσι έγινε ! Οι διάφορες εκδόσεις που δημιουργήθηκαν ήταν οι εξής* **:**

Έκδοση 1.0 : Η πρώτη απόπειρα μεταφοράς της εφαρμογής σε περιβάλλον MS-Windows. Όλη η είσοδος/έξοδος γίνεται μέσα από ένα απλό παράθυρο εισόδου/εξόδου, με διαδοχικές κλήσεις ρουτινών εισόδου/εξόδου ορμαθών χαρακτήρων σε παράθυρο

Έκδοση 2.0 : Δημιουργείται ένας απλός συντάκτης κειμένου για να εισάγεται ο αλγόριθμος, και κατόπιν καλείται από τον χρήστη ο parser.

Έκδοση 3.0 : Δημιουργία ολοκληρωμένης διεπικοινωνίας με τον χρήστη. Ο συντάκτης κειμένου τώρα πια υποστηρίζει λειτουργίες cut/copy/paste , λειτουργίες αναζήτησης και αντικατάστασης κειμένου, ανάκτηση και αποθήκευση σε αρχείο ascii, εκτύπωση. Ενσωματώνεται context-sensitive help.

Εκδόσεις 3.1 έως 3.2 : Δημιουργία πλαισίου διαλόγου "About" με βασικές πληροφορίες για την εφαρμογή. Μπάρα εργαλείων για γρηγορότερη πρόσβαση στις εντολές των μενού. Μπάρα βοήθειας όπου εμφανίζεται η περιγραφή της κάθε λειτουργίας πλήκτρου της μπάρας εργαλείων ή εντολής από τα μενού όποτε ο δείκτης του ποντικιού βρίσκεται πάνω από αυτό.

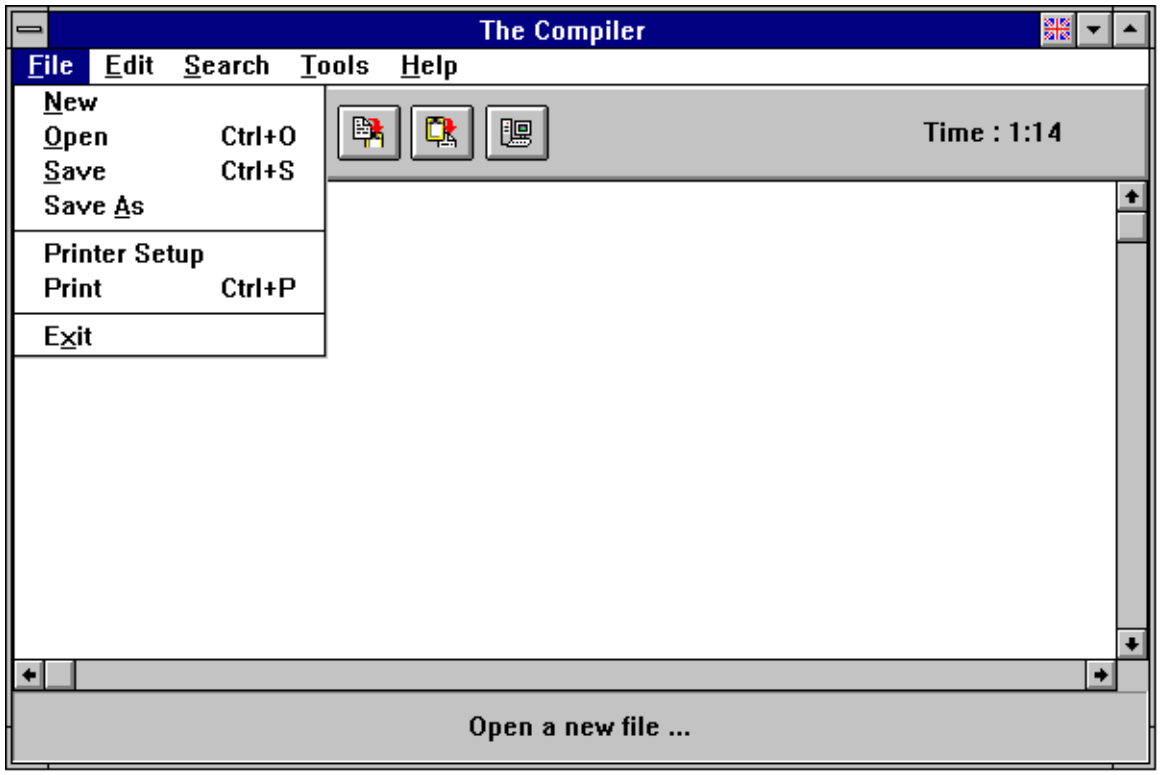
Έκδοση 3.3 : Ενσωματώνεται βάση δεδομένων για την αποθήκευση των αλγορίθμων και συνόλου στοιχείων σχετικών με την εργασία του χρήστη πάνω στον εκάστοτε αλγόριθμο (ημερομηνία και ώρα καταχώρησης στην βάση, υπολογισμένος φόρτος αλγορίθμου, είδος εργασίας που επιτελεί ο αλγόριθμος κτλ). Ενσωματώνεται δυνατότητα δημιουργίας δισδιάστατων και τρισδιάστατων γραφημάτων σύγκρισης φόρτων αλγορίθμων.

Έκδοση 4.0 : Βελτιώσεις πάνω στην βάση δεδομένων και στην δημιουργία γραφημάτων. Δυνατότητα να αντιγραφούν τα γραφήματα στο clipboard προκειμένου να μεταφερθούν σε μια οποιαδήποτε εφαρμογή που λειτουργεί κάτω από το περιβάλλον MS-Windows και η οποία μπορεί να δεχτεί δεδομένα από το clipboard .

Έκδοση 5.0 :Τώρα πια ο δρόμος για την ανάπτυξη βιβλιοθηκών-επεκτάσεων για το πρόγραμμα αυτό είναι ανοιχτός ! Προγραμματιστές τρίτων εταιριών (3rd party developers) έχουν πια την δυνατότητα να αναπτύσσουν (υπό μορφή DLL βιβλιοθηκών) τις βελτιώσεις/προσθήκες που κατά τη γνώμη τους λείπουν από το πρόγραμμα. Η εφαρμογή έχει την δυνατότητα να ανιχνεύει αυτά τα DLL κατά την διάρκεια της εκτέλεσής της και να ενσωματώνει τις λειτουργίες τους στο κεντρικό menu !

2.4 Ανάπτυξη τμημάτων

2.4.1 Interface



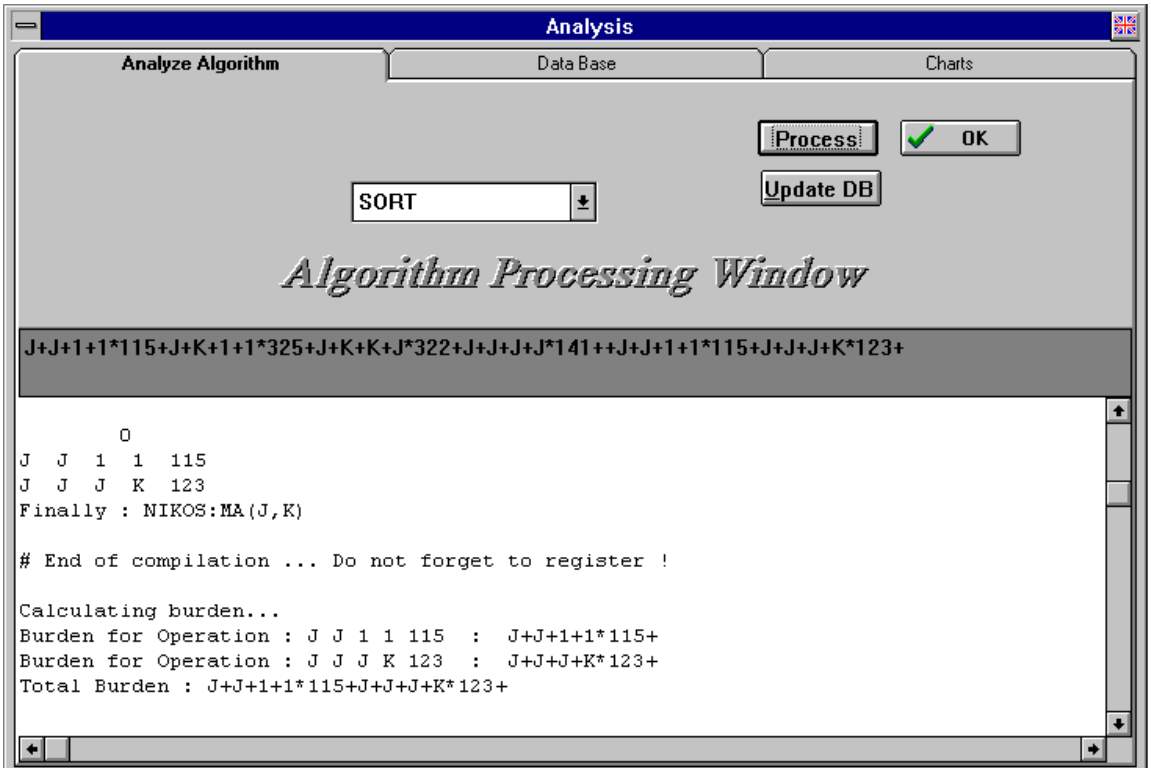
οθόνη συντάκτη κειμένου (βασική οθόνη)

Το πρόγραμμα έχει την δυνατότητα να λαμβάνει ως είσοδο έναν αλγόριθμο, ο οποίος θα πρέπει να τηρεί κάποιους κανόνες σύνταξης, οι οποίοι βέβαια περιγράφονται αναλυτικά στην τεκμηρίωση του προγράμματος (documentation) . Βέβαια, εκτός από την αναλυτική αναφορά στην σύνταξη του αλγορίθμου μέσα στην τεκμηρίωση του προγράμματος, υπάρχει και αναφορά στον τρόπο σύνταξης του αλγορίθμου στο "on-line help" για να διευκολύνεται ακόμα και ένας χρήστης ο οποίος δεν έχει ιδιαίτερη ευχέρεια στους αλγόριθμους, χωρίς να χρειάζεται να ανατρέχει κάθε φορά στο έντυπο documentation για να διορθώσει μια πιθανή λανθασμένη σύνταξή του. Θα μπορεί συνεπώς να χρησιμοποιηθεί με σχετική ευκολία και από άτομα που δεν ανήκουν στον χώρο της Πληροφορικής και δεν έχουν εξοικειωθεί με τον τομέα αυτό, αλλά επιθυμούν να συγκρίνουν αλγόριθμους που έχουν σχέση με την δική τους επιστήμη (π.χ. Μαθηματικοί, Φυσικοί, Χημικοί κτλ) . Ο αλγόριθμος που κατασκευάζει ο χρήστης μπορεί να αποθηκευθεί με την μορφή ascii αρχείου. Το interface (η διεπικοινωνία με τον χρήστη) έχει σχεδιαστεί ούτως ώστε να προσδίδει στο όλο πρόγραμμα μια φιλικότητα (user-friendly) προς τον χρήστη. Όλες οι ενέργειες που επιθυμεί ο χρήστης απέχουν μερικά πλήκτρα μόνο και μπορούν φυσικά να γίνουν και εξ' ολοκλήρου με τη χρήση του ποντικιού (mouse).

Η βασική οθόνη η οποία παρουσιάζεται στην αρχή χρησιμοποιείται για την εισαγωγή του αλγορίθμου και θυμίζει έναν τυπικό συντάκτη κειμένου (text editor). Υπάρχουν άλλες, χωρισμένες, οθόνες για την συντακτική/λεκτική ανάλυση , για την εύρεση της πολυπλοκότητας και του φόρτου του αλγορίθμου καθώς και για την διαχείριση της βάσης και την δημιουργία γραφημάτων. Όλες έχουν σχεδιαστεί με τέτοιο τρόπο ούτως ώστε να μην προκαλούν σύγχυση ακόμα και στον πιο άπειρο χρήστη. Εξ' άλλου υπάρχει σε κάθε σημείο context-sensitive help.

Το interface αναπτύχθηκε εξ' ολοκλήρου στο περιβάλλον Ταχείας Ανάπτυξης Εφαρμογών (Rapid Application Development) Borland Delphi v1.0

2.4.2 Parser



οθόνη Ανάλυσης Αλγορίθμων

Κατά την διάρκεια της συντακτικής/λεκτικής ανάλυσης του αλγορίθμου (πλήκτρο ενέργειας **P***rocess*) , ο χρήστης ενημερώνεται συνεχώς για την εξέλιξή της, προκειμένου να είναι σε θέση να διορθώσει τμήματα του αλγόριθμου, σε περίπτωση που ο parser δεν μπορέσει να τα διορθώσει (heuristic correction, βλ. *Προγραμματιστικές Τεχνικές*) και κρίνει αναγκαία την παρέμβαση του χρήστη. Αν χρειαστούν περαιτέρω δεδομένα από το χρήστη, ή η διόρθωση ορισμένων εκ των δεδομένων που έχει ήδη εισάγει, εμφανίζεται η κατάλληλη ερώτηση στο λευκό παράθυρο **Alg***orithm* **P***rocessin***g W***indo***w** και το πρόγραμμα περιμένει τον χρήστη να κάνει την διόρθωση που απαιτείται για να προχωρήσει η ανάλυση του αλγορίθμου.

Αφού αναλυθεί ο εκάστοτε αλγόριθμος ο χρήστης μπορεί, αν επιθυμεί, να καταχωρήσει τον αλγόριθμο καθώς και τα στοιχεία που προέκυψαν από την ανάλυσή του στην ενσωματωμένη βάση δεδομένων (συγκεκριμένα αποθηκεύεται ο αλγόριθμος, το όνομα του ascii αρχείου στο οποίο βρισκόταν ο αλγόριθμος, η παραμετρική εξίσωση πολυπλοκότητας του αλγορίθμου, το είδος του αλγορίθμου (π.χ. sort, search etc. Αν ο χρήστης δεν κατηγοριοποιήσει τον αλγόριθμο το πρόγραμμα, έπειτα από σχετική προειδοποίηση και από λήψη σχετικής άδειας από τον χρήστη τον κατατάσσει στην κατηγορία UNKNOWN, δηλαδή "αγνώστου είδους"), η ημερομηνία και η ώρα καταχώρησης

στην βάση καθώς και ένας σειριακός αριθμός ο οποίος από εκείνη την στιγμή και πέρα θα χαρακτηρίζει εκείνον τον αλγόριθμο μέσα στην βάση (index). Η καταχώρηση στη βάση γίνεται με το πλήκτρο ενέργειας *Update* **DB**.

Ο parser/scanner σχεδιάστηκε και υλοποιήθηκε σε γλώσσα Borland Pascal with Objects 7.0, σε αντικειμενοστραφή μορφή. Υπέστη μικρές αλλαγές και βελτιώσεις, όταν η εφαρμογή αυτή μετατράπηκε ούτως ώστε να λειτουργεί κάτω από το λειτουργικό σύστημα MS-Windows, προκειμένου να ενσωματωθεί στο interface που σχεδιάστηκε σε Borland Delphi v1.0

2.4.3 Πολυπλοκότητα

Η πολυπλοκότητα ενός αλγορίθμου είναι μια συνάρτηση των διαστάσεων των μεταβλητών που λαμβάνουν μέρος σε αυτόν. Αποτελεί ένα αντικειμενικό και επιστημονικό κριτήριο σύγκρισης αλγορίθμων. Αφού μπορούμε να συγκρίνουμε αλγορίθμους με τη βοήθεια της πολυπλοκότητάς τους, μπορούμε λοιπόν και να τους κατηγοριοποιούμε ανάλογα με το χρόνο και την δυσκολία εκτέλεσης τους. Αυτές οι κατηγορίες στις οποίες εντάσσονται οι αλγόριθμοι, με κριτήριο την πολυπλοκότητά τους ονομάζονται επίσημα "τάξεις μεγέθους" (order of growth).

Η κατάταξη ενός αλγορίθμου σε μια τέτοια κατηγορία γίνεται με τη χρήση συγκεκριμένων μαθηματικών κανόνων. Δεν αποτελεί όμως σκοπό αυτού του συγγράμματος η επεξήγηση αυτών. Θα τους παρουσιάσουμε απλώς.

Υπάρχουν τρία είδη πολυπλοκότητας αλγορίθμων. Η πολυπλοκότητα "χειρότερης περίπτωσης", η πολυπλοκότητα "μέσης περίπτωσης" και η πολυπλοκότητα "καλύτερης περίπτωσης". Το πως κατηγοριοποιούνται οι αλγόριθμοι ανάλογα με τις πολυπλοκότητες που αναφέραμε, θα το καταλάβουμε καλύτερα μέσα από μερικά παραδείγματα.

Η τάξη μεγέθους ενός αλγορίθμου προκύπτει σύμφωνα με απόλυτα ορισμένους μαθηματικούς κανόνες από την εξίσωση εκείνη των μεταβλητών του, που εκφράζει την πολυπλοκότητά του. Οι αλγόριθμοι κατατάσσονται τελικά σε κατηγορίες, ως προς την ταχύτητά τους κυρίως, συγκρίνοντας τις τάξεις μεγέθους τους.

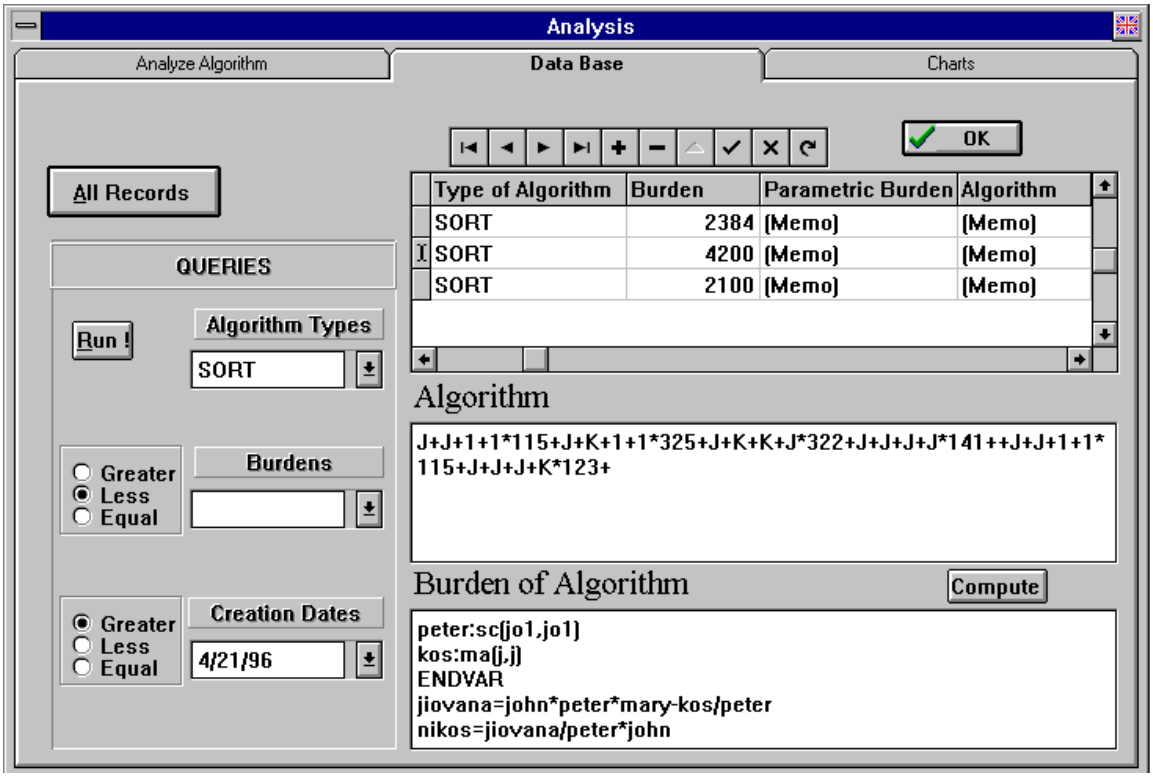
Υπάρχουν τρία είδη τάξεων μεγέθους, τα εξής :

- Τάξη μεγέθους χειρότερης περίπτωσης (O)
- Τάξη μεγέθους βέλτιστης περίπτωσης (Ω)
- Τάξη μεγέθους μέσου όρου (Θ)

Στην εφαρμογή (ACT), η παραμετρική εξίσωση που εκφράζει την πολυπλοκότητα του αλγορίθμου κατασκευάζεται ταυτόχρονα με τη συντακτική/λεκτική ανάλυση, χρησιμοποιώντας τον αναλυμένο μέχρι εκείνη τη στιγμή (parsed) αλγόριθμο. Η κατασκευή της εξίσωσης αυτής γίνεται με την βοήθεια μιας βιβλιοθήκης που κατασκευάστηκε σε μορφή dll ειδικά για την εύρεση της πολυπλοκότητας στοιχειωδών εντολών και πράξεων. Αυτή η εξίσωση παρουσιάζεται στο γκρι παράθυρο **A**/g*orithm* **P***rocessin***g W***indo***w**

Οι ρουτίνες εύρεσης της εξίσωσης αυτής δημιουργήθηκαν σε Microsoft Fortran v5.1

2.4.4 Βάση δεδομένων



Η βάση περιέχει όλους τους αλγορίθμους που έχει κατά καιρούς καταχωρήσει ο χρήστης. Με γραφικό τρόπο και χωρίς να χρειαστεί η πληκτρολόγηση καμίας απολύτως εντολής μπορεί ο χρήστης να διαλέξει την παρουσίαση στο πλέγμα παρουσίασης εγγραφών ενός συγκεκριμένου συνόλου αλγορίθμων μόνο.

Τα κριτήρια επιλογής είναι :

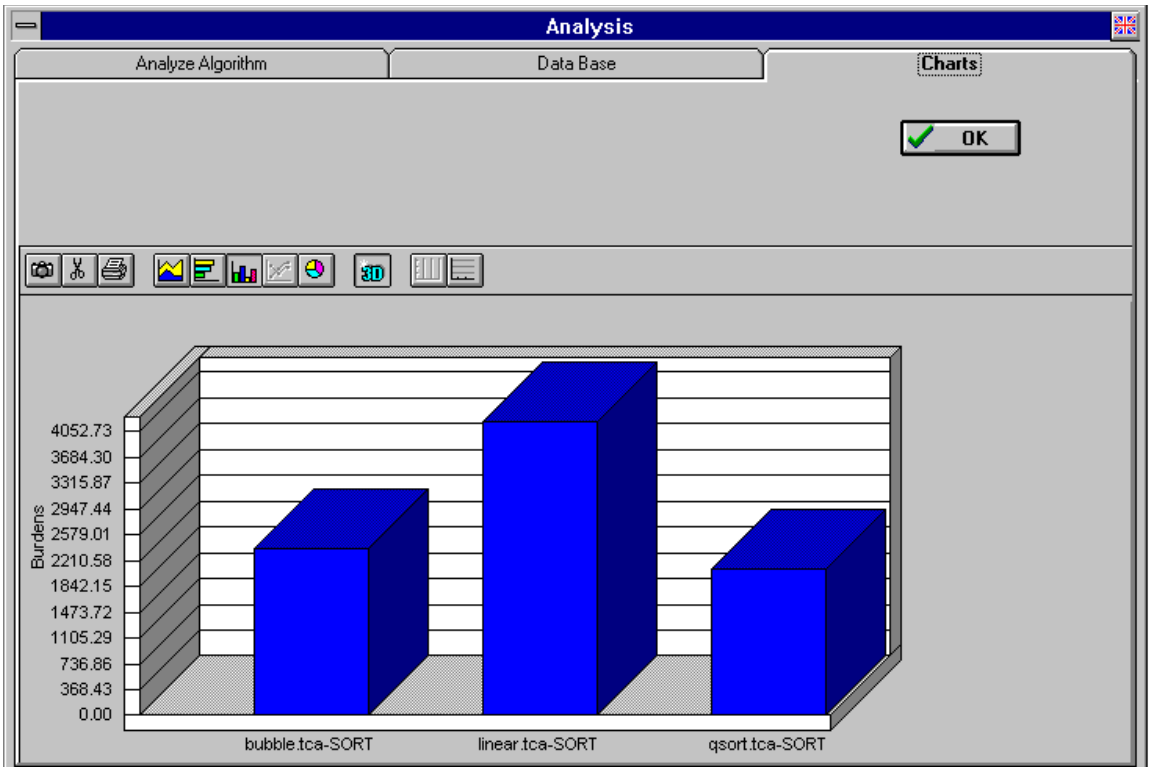
- **A***lgorithm* **T** *pes* **:** είδος του αλγορίθμου (sort , search etc.). Μπορεί ο χρήστης να επιλέξει να εμφανιστούν μόνο οι αλγόριθμοι "ταξινόμησης", ή μπορεί φυσικά να επιλέξει να εμφανιστούν οι αλγόριθμοι όλων των ειδών. Φυσικά, όλα τα είδη αλγορίθμων που έχουν εισαχθεί μέχρι εκείνη τη στιγμή είναι διαθέσιμα προς επιλογή στο χρήστη. Δεν είναι δηλαδή απαραίτητο να θυμάται ο ίδιος τι είδους αλγορίθμους έχει ήδη εισάγει.

- **B** *rdens* **:** πολυπλοκότητα του αλγορίθμου. Μπορεί ο χρήστης να επιλέξει να εμφανιστούν οι αλγόριθμοι που έχουν πολυπλοκότητα (για δεδομένες διαστάσεις των μεταβλητών που μετέχουν σε αυτούς και τις οποίες θα πρέπει να έχει ήδη εισάγει ο χρήστης) μικρότερη, ίση ή μεγαλύτερη από την πολυπλοκότητα ενός εκ των δοθέντων αλγορίθμων. Φυσικά, όλες οι πολυπλοκότητες αλγορίθμων που έχουν υπολογιστεί είναι διαθέσιμες στον χρήστη προς επιλογή . Δεν είναι δηλαδή απαραίτητο να θυμάται ο ίδιος τις πολυπλοκότητες των αλγορίθμων που έχει ήδη εισάγει.

- *reation* **D***ates* **:** Ημερομηνία καταχώρησης στην βάση. Μπορεί ο χρήστης να επιλέξει να εμφανιστούν όλοι οι αλγόριθμοι που έχει εισάγει πριν, μετά η σε μία ακριβώς συγκεκριμένη ημερομηνία. Και πάλι, όλες οι ημερομηνίες που έγιναν καταχωρήσεις αλγορίθμων στην βάση διατίθενται στον χρήστη προς επιλογή.

Η βάση δεδομένων δημιουργήθηκε αρχικά χρησιμοποιώντας την Borland Paradox, και η μετέπειτα διαχείρισή της έγινε μέσα από το εργαλείο ανάπτυξης εφαρμογών Borland Delphi v1.0 και με την βοήθεια της Borland Database Engine.

2.4.5 *Γραφήματα (charts)*



οθόνη γραφημάτων

Η εφαρμογή διαθέτει την δυνατότητα σχεδίασης δισδιάστατων και τρισδιάστατων γραφημάτων. Πρέπει πρώτα βέβαια ο χρήστης να διαλέξει για ποιό σύνολο ήδη καταχωρημένων αλγορίθμων να δημιουργηθεί γράφημα (βλ. *Βάση*) . Κατόπιν, μεταβαίνει στην σελίδα παρουσίασης γραφημάτων όπου το γράφημα θα έχει ήδη δημιουργηθεί αυτόματα. Υπάρχει η δυνατότητα να αλλάξει εκεί (μέσω της μπάρας επιλογών) το είδος του γραφήματος. Διαθέσιμα είδη γραφημάτων : horizontal, vertical pie charts, bar charts, linear και όλα αυτά σε δισδιάστατη ή τρισδιάστατη μορφή. Υπάρχει επίσης η δυνατότητα να αντιγράψει στο clipboard το τρέχον γράφημα προκειμένου να το μεταφέρει εκείνη την στιγμή σε κάποια άλλη εφαρμογή η οποία έχει την δυνατότητα εισαγωγής αντικειμένων από το clipboard !

Η δημιουργία γραφημάτων πραγματοποιήθηκε χρησιμοποιώντας τις βιβλιοθήκες γραφικών που παρέχει το περιβάλλον ανάπτυξης εφαρμογών Borland Delphi v1.0

2.4.6 *On-line Help*

Προσφέρεται η δυνατότητα στο χρήστη, σε οποιοδήποτε σημείο του προγράμματος, να καλέσει κείμενα βοήθειας (Help files) για τις συγκεκριμένες ενέργειες που προσπαθεί να εκτελέσει στην δεδομένη στιγμή. Αυτό γίνεται με τη βοήθεια του πλήκτρου F1.

Η κατασκευή του context sensitive help πραγματοποιήθηκε με τη βοήθεια του εργαλείου δημιουργίας HelpFiles HelpMagician.

3. Προγραμματιστικές Τεχνικές

Πολλοί θα αναρωτηθούν ίσως, ποιος ο λόγος για τον οποίο χρησιμοποιήθηκαν τόσα διαφορετικά περιβάλλοντα προγραμματισμού. Δυο κυρίως ήταν οι λόγοι :

(i) Κάθε γλώσσα προγραμματισμού παρουσιάζει ιδιαίτερες δυνατότητες, περισσότερο εξελιγμένες ίσως, σε έναν συγκεκριμένο τομέα σε σχέση με μια άλλη. π.χ. η Fortran χειρίζεται πολλές φορές καλύτερα τα αριθμητικά δεδομένα σε σύγκριση με την Pascal. Μάλιστα διαθέτει (στις τελευταίες εκδόσεις της) και περισσότερους βασικούς τύπους μεταβλητών. Έχει ενσωματωθεί ακόμα και δυνατότητα χειρισμού μιγαδικών αριθμών! Για αυτό το λόγο χρησιμοποιήθηκε Fortran προκειμένου να υπολογιστεί η πολυπλοκότητα του αλγορίθμου. Η Pascal, (πάλι στις τελευταίες της εκδόσεις) υποστηρίζει τον object oriented προγραμματισμό, ο οποίος ως γνωστό προσφέρει πολλά πλεονεκτήματα έναντι του "διαδικαστικού" (procedural), ένα εκ των οποίων είναι η εύκολη συντήρηση και βελτίωση του προγράμματος. Επειδή λοιπόν η φάση συντήρησης και βελτίωσης στην ανάπτυξη ενός parser διαρκεί πολύ περισσότερο από οποιαδήποτε άλλη φάση ανάπτυξης του, χρησιμοποιήθηκε η Pascal για την κατασκευή του parser του αλγορίθμου. Παρόμοιοι λόγοι μας ώθησαν στο να επιλέξουμε και τα υπόλοιπα περιβάλλοντα προγραμματισμού και εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του προγράμματος και των οδηγιών χρήσης αυτού. Τα εργαλεία αυτά παρουσιάζονται στον παρακάτω πίνακα.

Borland Delphi v1.0 / Turbo Vision v2.0	Δημιουργία Interface Δημιουργία γραφημάτων (charts) Δημιουργία βιβλιοθηκών σύνδεσης εξωτερικών DLL
Borland Pascal 7.0 with Objects	Δημιουργία συντακτικού/λεκτικού αναλυτή (parser/scanner)
Borland Paradox v7.0	Δημιουργία βάσης δεδομένων
Microsoft Fortran v5.1	Δημιουργία βιβλιοθηκών DLL για την εύρεση φόρτου στοιχειωδών πράξεων
Software Interphase Help Magician v2.6	Δημιουργία Help Files

(ii) Θελήσαμε να παρουσιάσουμε στο κοινό αυτόν τον νέο τρόπο ανάπτυξης εφαρμογών, στην προσπάθειά μας να αναδείξουμε τα πλεονεκτήματα της χρήσης πληθώρας περιβαλλόντων ανάπτυξης εφαρμογών προκειμένου να εκμεταλλεύονται τα προγραμματιστικά επιτελεία τις καλύτερες δυνατότητες από το καθένα από αυτά στο μέλλον, και να αναπτύσσεται υψηλότερης ποιότητας λογισμικό. Αυτός ο λόγος λειτούργησε συμπληρωματικά, ως προς τον πρώτο. Θα λέγαμε μάλλον ότι επιβεβαίωσε το ορθό της απόφασης μας να αναπτύξουμε την εφαρμογή χρησιμοποιώντας τα προαναφερθέντα περιβάλλοντα ανάπτυξης εφαρμογών.

3.1 Interface

Στην κατασκευή του interface δεν χρησιμοποιήθηκε καμία ιδιαίτερη προγραμματιστική τεχνική. Όπως ήδη είπαμε, έχει δημιουργηθεί εξ' ολοκλήρου στο περιβάλλον ανάπτυξης εφαρμογών Borland Delphi v1.0 Περιλαμβάνει έναν τυπικό συντάκτη κειμένου (text editor) για την σύνταξη των αλγορίθμων, και 3 οθόνες (φόρμες) στις οποίες γίνεται η ανάλυση του αλγορίθμου, η διαχείριση της βάσης δεδομένων και η δημιουργία των γραφημάτων αντίστοιχα. (βλ. Κεφ. 2, *Ανάπτυξη πακέτου λογισμικού*)

3.2 Parser

Πριν αρχίσει η ανάλυση του κυρίως τμήματος του αλγόριθμου, αναλύεται το τμήμα δηλώσεων αυτού και δημιουργείται ένας πίνακας συμβόλων (symbol table) ο οποίος χρησιμοποιείται καθ' όλη την πορεία της ανάλυσης του αλγορίθμου προκειμένου να πληροφορήσει σχετικά με το είδος και τις (αόριστες) διστάσεις των μεταβλητών που λαμβάνουν μέρος στον αλγόριθμο. Κατόπιν, το interface του προγράμματος τροφοδοτεί την μηχανή του parser (parser engine) με τον αλγόριθμο, μία προς μία γραμμή. Κάθε φορά που ο parser δέχεται και μια νέα γραμμή την αναλύει ως προς το λεκτικό και

συντακτικό της περιεχόμενο και αν δεν βρει λάθη καλεί τις ρουτίνες ανάλυσης πολυπλοκότητας (γραμμένες σε Fortran, μεταφρασμένες σε DLL) προκειμένου να βρεθεί η πολυπλοκότητα για την γραμμή εκείνη του αλγορίθμου που μόλις αναλύθηκε. Η ευρεθείσα εξίσωση πολυπλοκότητας συγχωνεύεται με αυτήν που έχει ήδη βρεθεί για όλες τις προηγούμενες γραμμές αλγορίθμου. Βέβαια, αν βρεθεί λεκτικό ή συντακτικό σφάλμα στον αλγόριθμο, προτού καλέσει ο parser τις ρουτίνες ανάλυσης πολυπλοκότητας προτρέπει τον χρήστη να διορθώσει το σφάλμα αυτό, αν δεν μπορεί να το διορθώσει από μόνος του ο parser (heuristic correction). Οι αλλαγές που ζητά ο parser από τον χρήστη στον αλγόριθμο κατά την διάρκεια της ανάλυσής του, προκειμένου να αφαιρεθούν ορισμένα κρίσιμα λάθη, δεν καταχωρούνται στον αλγόριθμο -σκόπιμα-, για να έχει τη δυνατότητα ο χρήστης να διορθώσει από μόνος του τον αλγόριθμο έπειτα, όπως αυτός επιθυμεί. Ο σκοπός που εξυπηρετούν είναι να ολοκληρωθεί η τρέχουσα ανάλυση του αλγορίθμου ανεμπόδιστα, για να σχηματίσει ο χρήστης μια γενική ιδέα για τον τρέχον αλγόριθμο. Κατόπιν ο χρήστης (και σύμφωνα βέβαια) με τις υποδείξεις του parser μπορεί να επιφέρει τις επιθυμητές αλλαγές ή βελτιώσεις στο σύνολο του αλγορίθμου, είτε στον κώδικα είτε στο τμήμα δηλώσεων.

3.3 Πολυπλοκότητα

Η πολυπλοκότητα ενός αλγορίθμου (βλ. Κεφ. 2, *Ανάπτυξη πακέτου λογισμικού*) εκφράζεται από μια παραμετρική συνάρτηση των διαστάσεων των μεταβλητών που λαμβάνουν μέρος στον αλγόριθμο. Οι παράμετροι της συνάρτησης αυτής (πλήθος παραμέτρων : 3) εκφράζουν το λόγο της πολυπλοκότητας των βασικών (τερματικών) πράξεων, στις οποίες αναλύεται τελικά κάθε αλγόριθμος. Αν δώσουμε συγκεκριμένα αριθμητικά δεδομένα στις διαστάσεις της συνάρτησης αυτής, τότε λαμβάνουμε έναν αριθμό[i] ο οποίος ονομάζεται **φόρτος** του αλγορίθμου και είναι ανάλογος του χρόνου εκτέλεσης (cpu time) του συγκεκριμένου αλγόριθμου, με τις συγκεκριμένες, αριθμητικές διαστάσεις μεταβλητών.

Κάποιοι που έχουν ασχοληθεί σε επαγγελματικό επίπεδο με την ανάλυση αλγορίθμων ίσως να αναρωτηθούν σε αυτό το σημείο αν ο τρόπος ανάλυσης που χρησιμοποιήθηκε στην συγκεκριμένη εφαρμογή υπολογίζει με ορθό όντως τρόπο την πολυπλοκότητα του δοθέντος αλγόριθμου.

Οι μέθοδοι υπολογισμού πολυπλοκότητας αλγορίθμου που έχουν χρησιμοποιηθεί στην έκδοση αυτή της εφαρμογής υπολογίζουν προσεγγιστικά (συγκεκριμένα, υπολογίζουν την "μέση πολυπλοκότητα") και με τον καλύτερο δυνατό τρόπο, κατά τη γνώμη μας, την πολυπλοκότητα ενός αλγόριθμου. Βέβαια, υπάρχουν επιστήμονες οι οποίοι ίσως στο μέλλον ή ακόμα και σήμερα να υποστηρίζουν διαφορετικούς τρόπους ανάλυσης. Αυτό όμως δεν αφορά στον τελικό χρήστη αφού δύο ή περισσότεροι αλγόριθμοι στην εφαρμογή αυτή υπόκεινται σε "σχετική σύγκριση". Συγκρίνονται δηλαδή και αντιπαραβάλλονται μεταξύ τους, οπότε ακόμα και αν κάποιος θεωρεί ότι η "απόλυτη" πολυπλοκότητα ενός αλγορίθμου είναι κατάτι διαφορετική από αυτήν που παρουσιάζουμε εμείς, η "σχετική σύγκριση" των αλγορίθμων αποφέρει αποτελέσματα τα οποία είναι ίδια , όποια μέθοδος σύγκρισης και αν χρησιμοποιηθεί.

Επίσης, και όπως είναι ίσως γνωστό, η αλματώδης εξέλιξη στην τεχνολογία της Πληροφορικής έχει επιφέρει ριζικές αλλαγές στην ταχύτητα των αλγορίθμων. Παλιότερα, ο λόγος της πολυπλοκότητας στον πολλαπλασιασμό δύο γραμμικών (scalar) μεταβλητών και της πρόσθεσής τους ήταν 4.0. Τώρα, με την πρόοδο της τεχνολογίας, έχει μειωθεί στο 1.1 ! Όπως ήδη είπαμε όμως, η σχετική σύγκριση αλγορίθμων δίνει πάντα τα ίδια "σχετικά" αποτελέσματα. Μία από τις βελτιώσεις που θα γίνουν στην επόμενη έκδοση της εφαρμογής αυτής θα είναι ο χρήστης να μπορεί να επιλέξει τις τρεις παραμέτρους εκείνες που καθορίζουν την αναλογία πολυπλοκότητας βασικών (τερματικών) "πράξεων", στις οποίες τελικά αναλύεται ο οποιοσδήποτε αλγόριθμος. Με αυτόν τον τρόπο η εφαρμογή αυτή καθίσταται διαχρονική, αφού θα υπολογίζει σωστά το φόρτο, σε οποιονδήποτε νέο επεξεργαστή δημιουργηθεί !

3.4 Βάση Δεδομένων

Η ενσωματωμένη βάση δεδομένων δημιουργήθηκε αρχικά χρησιμοποιώντας το Borland Paradox for Windows. Η διαχείριση των πινάκων που την απαρτίζουν πραγματοποιείται μέσα από την εφαρμογή με κλήσεις προς τις βιβλιοθήκες Borland Database Engine (BDE), της Borland.

---

[i] Ο αριθμός αυτός δεν έχει μονάδες μέτρησης

Αποτελείται κυρίως από δύο πίνακες : ο ένας χρησιμοποιείται στην οθόνη *Ανάλυσης Αλγορίθμου* προκειμένου να διευκολύνει τον χρήστη να κατατάξει τον αλγόριθμο που επεξεργάζεται εκείνη τη στιγμή σε μία από τις ήδη υπάρχουσες κατηγορίες. Βέβαια, ο χρήστης μπορεί να δημιουργήσει μια νέα κατηγορία αλγορίθμων εκείνη τη στιγμή, χωρίς καμία ιδιαίτερη δυσκολία, και να εντάξει τον υπό επεξεργασία αλγόριθμο σε αυτήν την κατηγορία. Αυτός ο πίνακας λοιπόν αποτελείται από ένα μόνο αλφαριθμητικό πεδίο μήκους 40 χαρακτήρων, στο οποίο αποθηκεύονται οι διάφορα κατηγορίες των αλγορίθμων.

Ο δεύτερος πίνακας της βάσης είναι αυτός που χρησιμοποιείται κυρίως από την εφαρμογή και αποτελείται από τα εξής πεδία :

1. **Index.** Αυτό το πεδίο περιέχει έναν μοναδικό για κάθε εγγραφή δείκτη, ο οποίος χρησιμεύει στην επεξεργασία των δεδομένων που περιέχονται στη βάση. Αναλυτική αναφορά πάνω στον τρόπο με το οποίο η εφαρμογή χρησιμοποιεί εσωτερικά αυτό το πεδίο είναι πέραν από τους σκοπούς του συγγράμματος αυτού.

2. **Type of Algorithm.** Αυτό το πεδίο περιέχει την κατηγορία στην οποία εντάσσεται ο αλγόριθμος που αντιστοιχεί σε αυτήν την εγγραφή. Έχουν υλοποιηθεί κατάλληλοι περιορισμοί ακεραιότητας αναφοράς προκειμένου οι κατηγορίες αλγορίθμων που περιέχονται στα πεδία αυτού του πίνακα να είναι οι ίδιες με αυτές που περιέχονται στα πεδία του πρώτου πίνακα. Είναι αλφαριθμητικό πεδίο 15 χαρακτήρων.

3. **Burden.** Σε αυτό το πεδίο αποθηκεύεται ο φόρτος του αλγορίθμου της εγγραφής αυτής, αν και όταν ο χρήστης εφοδιάσει τον αλγόριθμό του με συγκεκριμένες αριθμητικές τιμές για τις διαστάσεις των μεταβλητών που λαμβάνουν μέρος στον αλγόριθμο. Είναι αριθμητικό πεδίο ακεραίων.

4. **Parametric Burden.** Σε αυτό το πεδίο αποθηκεύεται η εξίσωση εκείνη που εξάγεται από την ανάλυση του αλγορίθμου και η οποία εκφράζει την πολυπλοκότητά του (complexity). Είναι αλφαριθμητικό πεδίο θεωρητικά άπειρων χαρακτήρων. Αν το πλήθος των χαρακτήρων που προσπαθούμε να αποθηκεύσουμε σε αυτό είναι πάνω από 40 τότε οι θα αποθηκευθεί σε ένα εξωτερικό αρχείο με κατάληξη MB.

5. **Algorithm.** Σε αυτό το πεδίο αποθηκεύεται ο αλγόριθμος, όπως τον εισήγαγε ο χρήστης. Το είδος του πεδίου είναι το ίδιο με το πεδίο Parametric Burden.

6. **Date of Analysis.** Σε αυτό το πεδίο αποθηκεύεται η ημερομηνία καταχώρησης του αλγορίθμου στη βάση. Είναι πεδίο καταχώρησης ημερομηνίας.

7. **Time of Analysis.** Σε αυτό το πεδίο αποθηκεύεται η ώρα καταχώρησης του αλγορίθμου στη βάση. Είναι πεδίο καταχώρησης ώρας.

8. **Filename of Algorithm.** Σε αυτό το πεδίο αποθηκεύεται το όνομα (full pathname) του αρχείου στο οποίο βρίσκεται (ή τουλάχιστον βρισκόταν κατά την ανάλυσή του) ο αλγόριθμος της συγκεκριμένης εγγραφής.. Είναι αλφαριθμητικό πεδίο 40 χαρακτήρων.

3.5 Γραφήματα

Δεν έχουν χρησιμοποιηθεί ιδιαίτερες προγραμματιστικές τεχνικές για την δημιουργία των γραφημάτων. Η Borland παρέχει τώρα πια στον προγραμματιστή τις αναγκαίες βιβλιοθήκες εκείνες (components) που απαιτούνται για την δημιουργία γραφικών και γραφημάτων, με αποτέλεσμα ο τελευταίος να έχει τη δυνατότητα να συγκεντρωθεί στην ανάπτυξη και βελτίωση του κυρίως μέρους του προγράμματος του.

3.6 Context sensitive Help

Παλιότερα, η κατασκευή context sensitive help files ήταν κάτι ιδιαίτερα δύσκολο αφού ο προγραμματιστής έπρεπε να μάθει μια ξεχωριστή γλώσσα μόνο και μόνο για να δημιουργήσει οδηγίες χρήσης του προγράμματός του, οι οποίες να μπορούν βέβαια να κληθούν δυναμικά από το πρόγραμμα (help files) ! Τώρα πια, υπάρχουν εργαλεία ανάπτυξης των help files. Αυτά τα εργαλεία βοηθούν στην δημιουργία των help files, χωρίς ο προγραμματιστής να έχει πλήρη γνώση του κώδικα και των

μακροεντολών της γλώσσας που απαιτείται για την δημιουργία αυτών. Μέσα από κατάλληλα γραφικά περιβάλλοντα μπορεί να επιλέξει ο προγραμματιστής ποιες μακροεντολές επιθυμεί να εκτελεστούν και έχει την δυνατότητα να δει τα αποτελέσματα στην οθόνη του προτού μεταφράσει το όλο πρόγραμμα σε Help file... Αυτά λοιπόν τα εργαλεία αποτελούν ανεκτίμητη βοήθεια σε έναν προγραμματιστή και τείνουν να εδραιωθούν στις ημέρες μας ως βασικά εργαλεία ανάπτυξης , τουλάχιστον του βασικού μέρους, των help files.

Το εργαλείο που χρησιμοποιήσαμε εμείς είναι το Help Magician v2.6 της εταιρίας Software Interphase.

3.7 Open Architecture

Όπως είναι γνωστό, το μεγαλύτερο μέρος του κύκλου ζωής ενός προγράμματος αποτελείται από τη φάση συντήρησης. Λαμβάνοντας υπόψη αυτό, το κυρίως μέρος του προγράμματος, ο parser/scanner , σχεδιάστηκε σε αντικειμενοστραφή μορφή προκειμένου να είναι δυνατή η ταχεία και επιτυχής συντήρηση του προγράμματος.

Πολλές φορές, "τρίτοι" προγραμματιστές αντιλαμβάνονται προοπτικές για την βελτίωση ενός προγράμματος, τις οποίες δεν αντιλαμβάνεται ο αρχικός προγραμματιστής. Για προφανείς λόγους δεν δημοσιεύονται οι πηγαίοι κώδικες των εφαρμογών, με αποτέλεσμα να μην είναι δυνατόν να αναπτυχθούν ή να εξελιχθούν τα προγράμματα και από "τρίτους" προγραμματιστές. Εμείς, θέλοντας λοιπόν να δώσουμε την ευκαιρία τόσο στην εφαρμογή αυτή να αναπτυχθεί προς οποιοδήποτε τομέα αλλά και σε όποιον προγραμματιστή θελήσει, να προσθέσει τμήματα που αυτός θεωρεί οτι θα έπρεπε να υπήρχαν, την σχεδιάσαμε λαμβάνοντας υπόψη τα νέα πρότυπα κατασκευής εφαρμογών που αρχικά έθεσαν οι εταιρίες λογισμικού Adobe Systems και Quark με τα αντίστοιχα προγράμματά τους Photoshop και Xpress.

Η εφαρμογή αυτή λοιπόν υποστηρίζει την ανοικτή αρχιτεκτονική (Open Architecture) η οποία παρουσιάστηκε σε πρακτική εφαρμογή για πρώτη φορά στα προαναφερθέντα προγράμματα. Παρέχει την δυνατότητα δηλαδή σε "τρίτους" προγραμματιστές να προσθέσουν τμήματα σε αυτή, χωρίς να έχουν τον πηγαίο κώδικα !! Αυτό επιτυγχάνεται κατασκευάζοντας τις επιθυμητές προσθήκες υπό μορφή DLL και χρησιμοποιώντας την κατάλληλη σύνταξη (βλ. Παράρτημα). Κάθε φορά που ο χρήστης εκτελεί την εφαρμογή, αυτή ανιχνεύει τον κατάλογο (directory) της για τέτοια dll. Αν βρει, ενωματώνει με δυναμικό τρόπο τις λειτουργίες τους στα menu της

3.8 Μελλοντική συντήρηση Software

Μερικές από τις βελτιώσεις που θα γίνουν στις επόμενες εκδόσεις του προγράμματος είναι οι εξής :

- Βελτίωση του συντακτικού/λεκτικού αναλυτή (parser optimisation)
- Παραγωγή κώδικα (code generation)
- Επέκταση πράξεων

4. Οδηγίες χρήσης

4.1 Σύνταξη αλγορίθμων

Οι αλγόριθμοι που μπορεί να εισάγει στο πρόγραμμα ο χρήστης και να αναλύσει είναι προς το παρόν περιορισμένης εμβέλειας. Μία από τις βελτιώσεις των μελλοντικών εκδόσεων του προγράμματος είναι και η αύξηση των tokens, λειτουργιών και συναρτήσεων που μπορεί να αναγνωρίσει ο parser.

Οι αλγόριθμοι που αναγνωρίζονται είναι κυρίως αλγόριθμοι διαχείρισης πινάκων (matrix manipulation). Αναγνωρίσιμες πράξεις μεταξύ πινάκων είναι η πρόσθεση, αφαίρεση και πολλαπλασιασμός. Βέβαια γίνεται έλεγχος κατά το parsing για το αν επιτρέπεται η εκάστοτε πράξη και ειδοποιείται ο χρήστης με κατάλληλα μηνύματα για να επιφέρει διορθώσεις στον αλγόριθμο, αν οι διαστάσεις των πινάκων είναι τέτοιες που να μην επιτρέπεται μια πράξη. Επίσης, επιτρέπεται ο πολλαπλασιασμός και η διαίρεση ενός πίνακα με scalar μεταβλητή. Τέλος, επιτρέπεται η αντιστροφή και αναστροφή πινάκων.

Όσον αφορά στην αντιστροφή, ο αλγόριθμος που θεωρούμε ότι θα χρησιμοποιήσει ο χρήστης για να την πραγματοποιήσει είναι αυτός του Cholesky

Τυπικά δείγματα αλγορίθμων που είναι σε θέση να αναγνωρίσει ο parser στην παρούσα μορφή του είναι τα εξής :

```
    beginvar
a:mas(n,n)
b:ma(n,n)
d:sc(1,1)
endvar

c=a+b/d
```

```
    beginvar
y1:ma(n,n)
y2:ma(n,n)
a:ma(n,n)
num:sc(1,1)
endvar

y2=num*(y1+inv(y1)*a)
```

(ο δεύτερος αλγόριθμος αντιστρέφει έναν πίνακα)

Για περισσότερα παραδείγματα αλγορίθμων μπορείτε να ανατρέξετε στην ίδια την εφαρμογή. Υπάρχουν έτοιμοι αλγόριθμοι είτε με τη μορφή εξωτερικών (ASCII) αρχείων, είτε καταχωρημένοι μέσα στη βάση δεδομένων.

Τα tokens που δέχεται η πρωτότυπη (prototype) αυτή έκδοση της εφαρμογής είναι τα εξής :

MA : Τυχαίος πίνακας δύο διαστάσεων, χωρίς καμία ιδιαίτερη ιδιότητα

MAT : Τυχαίος πίνακας δύο διαστάσεων, τριγωνικός

MAS : Τυχαίος πίνακας δύο διαστάσεων, συμμετρικός

VE : Τυχαίος πίνακας-γραμμή ή πίνακας-στήλη. Προφανώς οι διαστάσεις του είναι (1,x) ή (x,1) αντίστοιχα.

SC : Scalar (γραμμική) μεταβλητή. Προφανώς οι διαστάσεις της είναι (1,1). Αν γίνει λάθος στην δήλωση των διαστάσεών της διορθώνεται αυτόματα.

INV : Αντιστροφή πίνακα. Οι διαστάσεις του πίνακα παραμένουν ίδιες

TR : Αναστροφή πίνακα. Οι διαστάσεις του πίνακα εναλλάσσονται. π.χ. ένας πίνακας διαστάσεων (x,z) επιστρέφει έναν πίνακα διαστάσεων (z,x)

Ο αλγόριθμος που εισάγει ο χρήστης χωρίζεται στα εξής τμήματα :

- **τμήμα δηλώσεων.** Σε αυτό το τμήμα ο χρήστης πρέπει να εισάγει τις μεταβλητές που θα χρησιμοποιήσει, τους τύπους δεδομένων και τις (αόριστες) διαστάσεις αυτών . Ένα παράδειγμα τμήματος δηλώσεων είναι το εξής :

  Beginvar
  john:mat(j,k)
  mary:mas(k,j)
  peter:sc(1,1)
  Endvar

  Προσέξτε ότι το τμήμα δηλώσεων περικλείεται από τις κρατημένες λέξεις (reserved words) **Begin**var, **End**var. Με το παραπάνω τμήμα δηλώνουμε στο πρόγραμμα ότι ο αλγόριθμος που θα χρησιμοποιήσουμε περιέχει τη μεταβλητή john που είναι δυσδιάστατος πίνακας διαστάσεων j,k , τη μεταβλητή mary που είναι δυσδιάστατος συμμετρικός πίνακας διαστάσεων k,j και τη μεταβλητή peter που είναι γραμμική μεταβλητή.

τμήμα αλγορίθμου. Σε αυτό το τμήμα γράφουμε τον αλγόριθμό μας. Έπεται του τμήματος δηλώσεων και δεν περικλείεται από καμία δεσμευμένη λέξη. Ένα παράδειγμα αλγορίθμου είναι το εξής :

jiovana=int(john*peter*mary)/peter
nikos=jiovana/peter*john

Συνολικά λοιπόν ο συγκεκριμένος αλγόριθμος θα είναι ο εξής

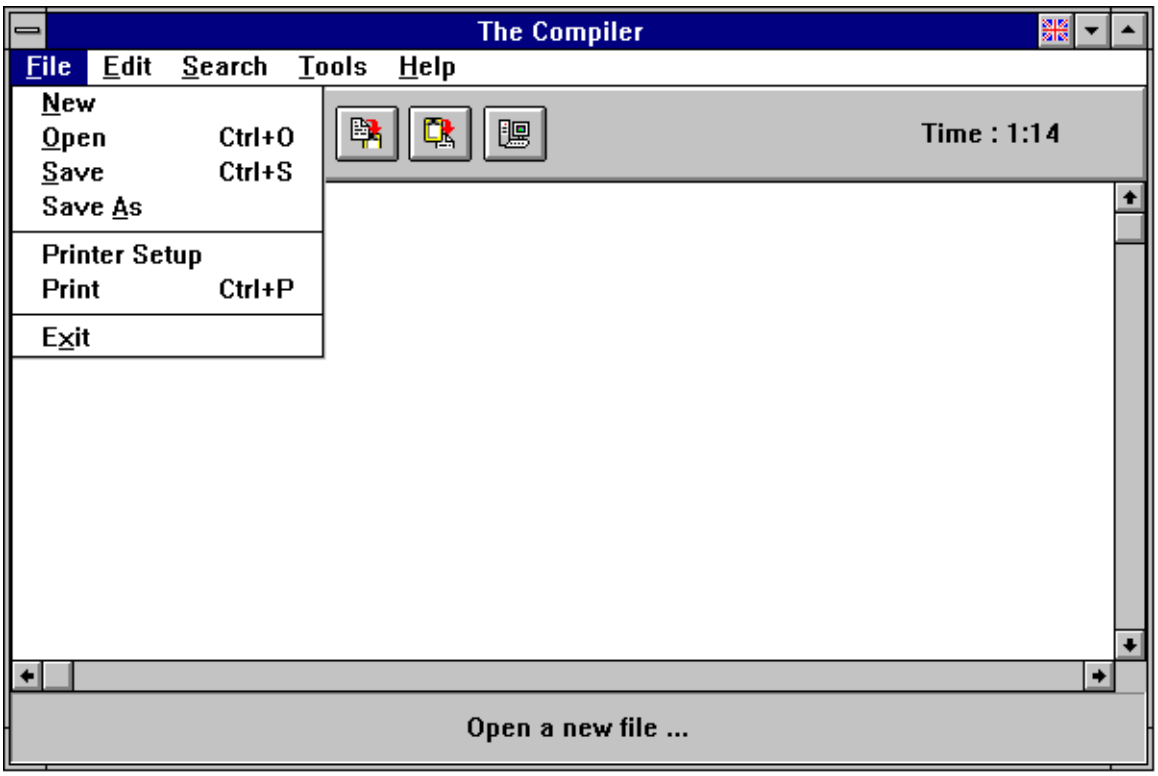
Beginvar
john:mat(j,k)
mary:mas(k,j)
peter:sc(1,1)
Endvar
jiovana=int(john*peter*mary)/peter
nikos=jiovana/peter*john

4.2 Παράδειγμα

Ένα ολοκληρωμένο παράδειγμα χρήσης θα βοηθούσε ακόμα περισσότερο. Ας το δούμε :

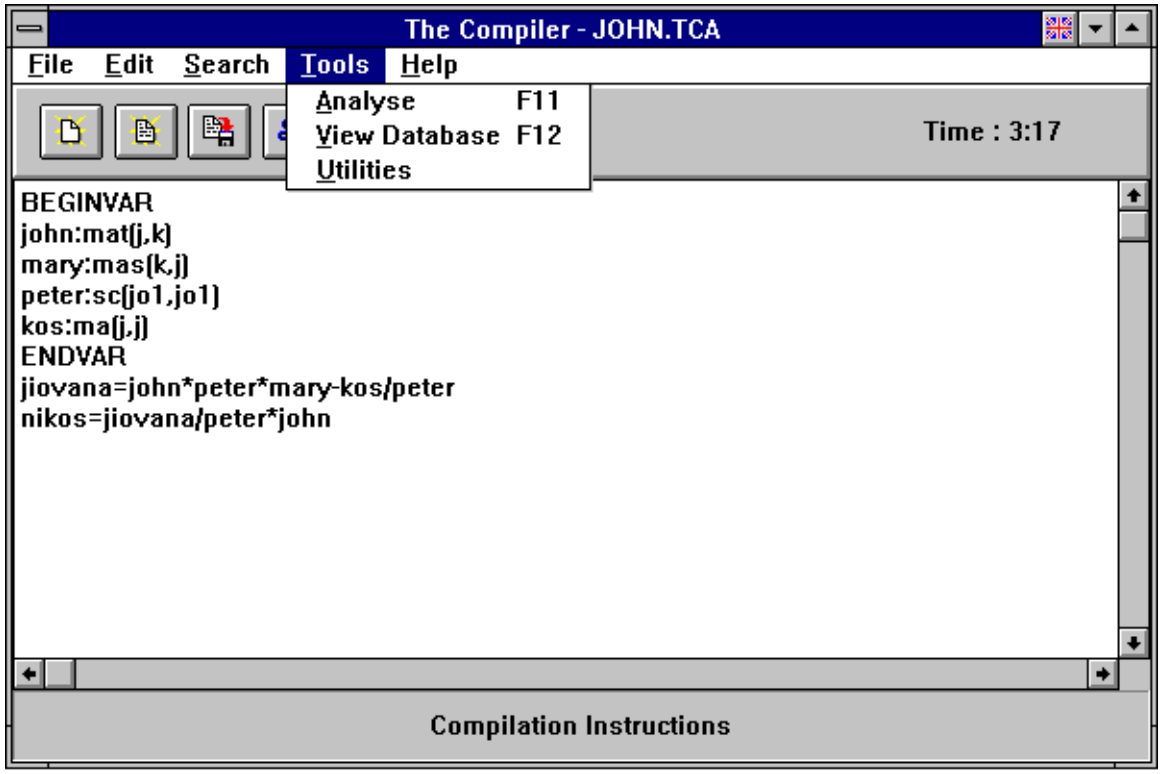
Έστω ότι ένας μαθηματικός θέλει να υπολογίσει την πολυπλοκότητα του αλγορίθμου με τον οποίο θα μπορούσε να υπολογίσει την πρόσθεση σε ένα δυσδιάστατο πίνακα (έστω A(κ,λ)) του εαυτού του, δηλαδή να υπολογίσει τον πολυπλοκότητα του αλγορίθμου με τον οποίο μπορεί να γίνει η πράξη A+A. Ο εν λόγω αλγόριθμος είναι ο εξής :

```
αλγόριθμος :
  Beginvar
  A,X : ma(k,l)
  Endvar
  X=A+A
```



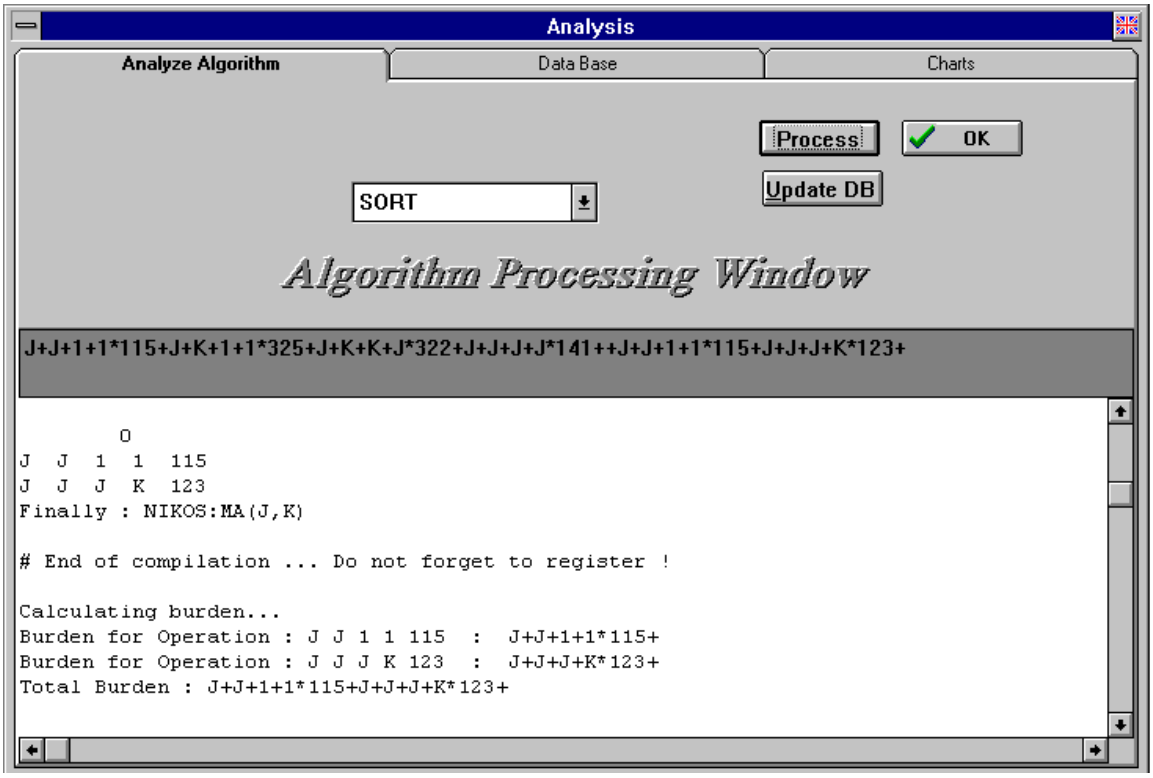
οθόνη συντάκτη κειμένου (βασική οθόνη)

Ο χρήστης λοιπόν θα πρέπει να ξεκινήσει μια νέα εργασία (επιλογή 'New' από το menu 'File'), και έπειτα να γράψει τον συγκεκριμένο κώδικα στο παράθυρο εισαγωγής αλγόριθμου (το πρώτο λευκό παράθυρο που εμφανίζεται στην εφαρμογή και φέρει τον τίτλο του αρχείου στο οποίο βρίσκεται αποθηκευμένος ο αλγόριθμος ή φέρει τον τίτλο "Untitled" αν δεν έχει ακόμα αποθηκευθεί ο αλγόριθμος σε αρχείο). Τέλος θα πρέπει να αποθηκεύσει τον αλγόριθμο σε ένα αρχείο (επιλογή 'Save As' από το menu 'File').



οθόνη συντάκτη κειμένου (βασική οθόνη)

Η ανάλυση του αλγορίθμου (γίνεται αυτόματα λεκτική, συντακτική ανάλυση και αν αυτή επιτευχθεί, ολοκληρώνεται η ανάλυση του αλγορίθμου και με την εύρεση της πολυπλοκότητάς του) πραγματοποιείται στην οθόνη που προκύπτει όταν ο χρήστης επιλέξει το 'Analyze' από το menu 'Tools'.



οθόνη ανάλυσης αλγορίθμου

Στην παραπάνω οθόνη (πολλαπλές φόρμες) πρέπει ο χρήστης να μεταβεί στην σελίδα 'Analysis' αν δεν βρίσκεται ήδη εκεί. Τώρα μπορεί να πατήσει το πλήκτρο 'Process' και ακολουθεί στο παρουσιαζόμενο παράθυρο I/O μια συνεχής ροή πληροφοριών που αφορούν στην ανάλυση του αλγορίθμου ο οποίος υπόκειται σε επεξεργασία εκείνη την στιγμή. Αν προκύψει ένα οποιοδήποτε λάθος οι περιπτώσεις χειρισμού είναι οι εξής δύο :

(i) Αν το λάθος μπορεί να διορθωθεί "αυτόματα" από τον parser, ο χρήστης απλά ειδοποιείται για την διόρθωση που έλαβε χώρα. Ο parser είναι εξοπλισμένος με την δυνατότητα να διορθώνει ορισμένα λάθη από μόνος του, χρησιμοποιώντας ευριστικές τεχνικές ανάλυσης αλγορίθμου και ανάνηψης από λάθη. Αυτό σημαίνει ότι όταν ο parser "πιστεύει" πως η διόρθωση ενός λάθους είναι αυτονόητη την εφαρμόζει από μόνος του.

(ii) Αν το λάθος δεν μπορεί να διορθωθεί από τον parser χωρίς την παρέμβαση του χρήστη τότε ο τελευταίος ειδοποιείται για το σφάλμα που συνέβη κατά το scanning/parsing και καλείται να διορθώσει τον αλγόριθμο, στο ή στα σημεία όπου χρειάζεται διόρθωση ή αλλαγή. Αυτή η αλλαγή σκόπιμα δεν μεταβάλλει τον αλγόριθμο, όπως αυτός εμφανίζεται και στο παράθυρο εισαγωγής αλγορίθμου, προκειμένου να μπορέσει ο χρήστης να διορθώσει ολόκληρο τον αλγόριθμο έπειτα, όπως επιθυμεί αυτός, έχοντας πια υπόψη του ότι ένα τμήμα του ήταν λανθασμένο.

Τελικά, αφού ολοκληρωθεί η ανάλυση του αλγορίθμου εξάγεται η παραμετρική εκείνη εξίσωση (με παραμέτρους τις διαστάσεις των στοιχείων που έλαβαν μέρος στον αλγόριθμο) η οποία εκφράζει την πολυπλοκότητα του αλγορίθμου και φυσικά παρουσιάζεται στο χρήστη. Έπειτα ο χρήστης έχει την δυνατότητα, αν επιθυμεί, να ενημερώσει τη βάση δεδομένων με τον αλγόριθμο που μόλις επεξεργάστηκε με την βοήθεια του προγράμματος. Αυτή η ενέργεια γίνεται με το πλήκτρο 'Update DB'. Αν ο χρήστης προσπαθήσει να χρησιμοποιήσει το πλήκτρο αυτό πριν να ορίσει (στο διπλανό από το πλήκτρο, πλαίσιο επιλογών) το "είδος" του αλγορίθμου* η εφαρμογή τον προτρέπει είτε να ορίσει το είδος του συγκεκριμένου αλγορίθμου είτε να τον καταχωρήσει ως "αλγόριθμο άγνωστου είδους" ("unknown")

Στην βάση αποθηκεύονται :

1) ο αλγόριθμος
2) η παραμετρική εξίσωση που εκφράζει την πολυπλοκότητά του

3) ο φόρτος που υπολογίσθηκε τελευταία φορά για συγκεκριμένες, αριθμητικές διαστάσεις των μεταβλητών του αλγορίθμου

4) η ημερομηνία και ώρα καταχώρησης στην βάση

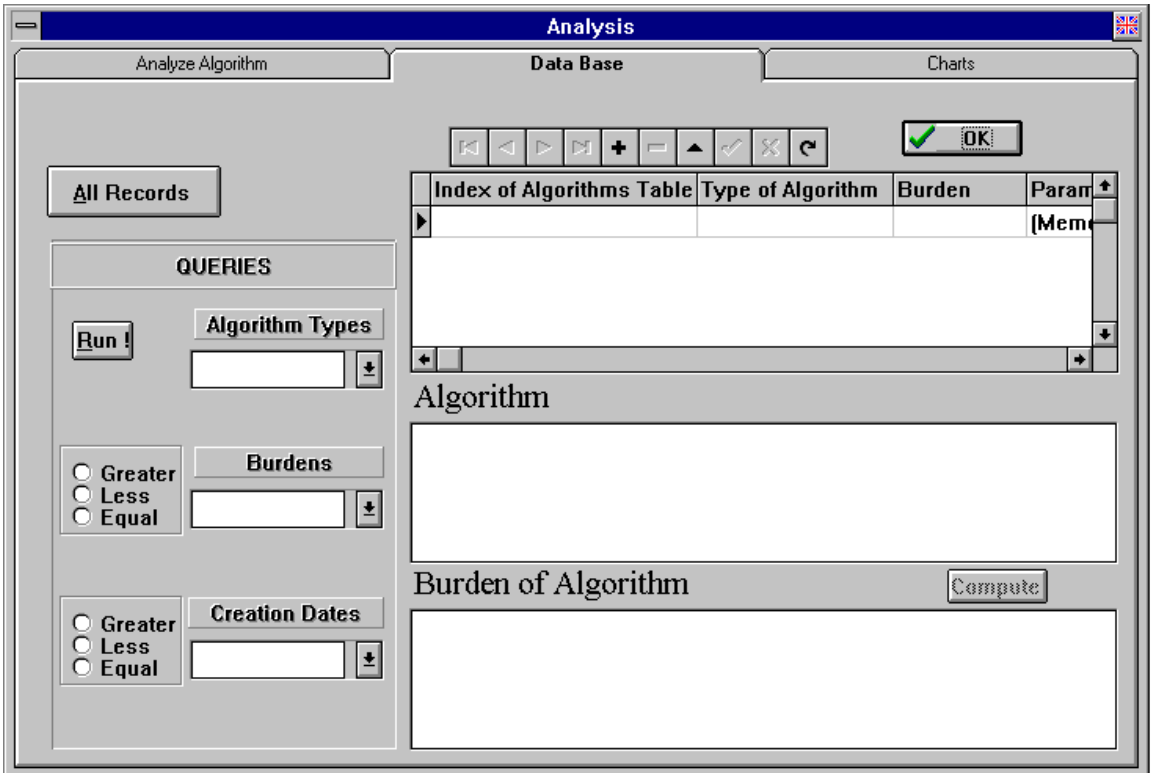
4) το όνομα του ascii αρχείου στο οποίο βρίσκεται ή τουλάχιστον βρισκόταν ο αλγόριθμος όταν αναλύθηκε για τελευταία φορά

Βέβαια, το να εξάγει κανείς την πολυπλοκότητα για ένα αλγόριθμο μόνο ίσως να μην παρουσιάζει τόσο μεγάλο επιστημονικό ενδιαφέρον. Η χρησιμότητα της εφαρμογής αυτής θα αναδειχθεί περισσότερο αν προχωρήσουμε στην εκμάθηση του προγράμματος χρησιμοποιώντας από εδώ και εις το εξής περισσότερους του ενός αλγορίθμους! Ας υποθέσουμε λοιπόν ότι ο χρήστης εισήγαγε στην βάση, με τον τρόπο που περιγράψαμε παραπάνω, άλλους τρεις αλγορίθμους που επιτελούν το ίδιο ακριβώς έργο, δηλαδή την πρόσθεση σε έναν πίνακα του εαυτού του. Τώρα πια θα αναφερόμαστε στους εξής 4 αλγόριθμους :

X=A+AX=2*A

X=8(A+A+A)/12X=4(A+5*A-3*A)/6

Είναι προφανές βέβαια οτι αν θέλουμε να αναζητήσουμε τον βέλτιστο αλγόριθμο για αυτήν την δουλειά πρέπει να τον αναζητήσουμε ανάμεσα στους δύο πρώτους. Οι άλλοι δύο εισήχθησαν στο προκείμενο παράδειγμα για λόγους εξάσκησης και εξοικείωσης με το πρόγραμμα.



οθόνη βάσης δεδομένων

Ας μεταφερθούμε λοιπόν στη σελίδα εκείνη που βρίσκεται δίπλα στη σελίδα “Analysis”, στην οποία εργαζόμασταν μέχρι τώρα. Δηλαδή στην σελίδα η οποία φέρει τον τίτλο “Database”.

Το πλέγμα που παρουσιάζεται άνω δεξιά εμφανίζει τους αλγόριθμους που έχουμε ή θα έχουμε επιλέξει από την βάση. Υπάρχει η δυνατότητα να εμφανιστούν όλοι οι καταχωρηθέντες αλγόριθμοι (με το πλήκτρο ενέργειας "All Records", ή μπορούμε να διαλέξουμε ένα υποσύνολο αυτών με τη βοήθεια των κριτηρίων που παρουσιάζονται στο κάτω αριστερό τμήμα της σελίδας.

Μπορούμε λοιπόν να επιλέξουμε αλγορίθμους οι οποίοι :

(I) να ανήκουν σε μια συγκεκριμένη κατηγορία (πλαίσιο επιλογής "Algorithm Types")

(ii) ο φόρτος τους να είναι μεγαλύτερος, μικρότερος ή ίσος με τον φόρτο που έχουμε εισάγει στο πλαίσιο επιλογής "Burden"

(iii) η ημερομηνία καταχώρησής τους στην βάση να είναι μικρότερη, ίση ή μεγαλύτερη αυτής που έχουμε επιλέξει στο πλαίσιο επιλογής "Date of Algorithm".

Έπειτα, το μόνο που έχουμε να κάνουμε για να εμφανιστεί το επιλεχθέν υποσύνολο των αλγορίθμων που έχουμε καταχωρήσει στην βάση είναι να χρησιμοποιήσουμε το πλήκτρο ενέργειας "Run Query". Οι αλγόριθμοι που έχουμε επιλέξει θα εμφανιστούν στο προαναφερθέν πλέγμα !

Βέβαια, αν δεν συμπληρώσουμε ορισμένα από αυτά τα κριτήρια, αυτά δεν θα λάβουν μέρος στην επιλογή των αλγορίθμων. Μερικά παραδείγματα ίσως να διασαφηνίσουν τις ενέργειες που αναφέραμε ως τώρα :

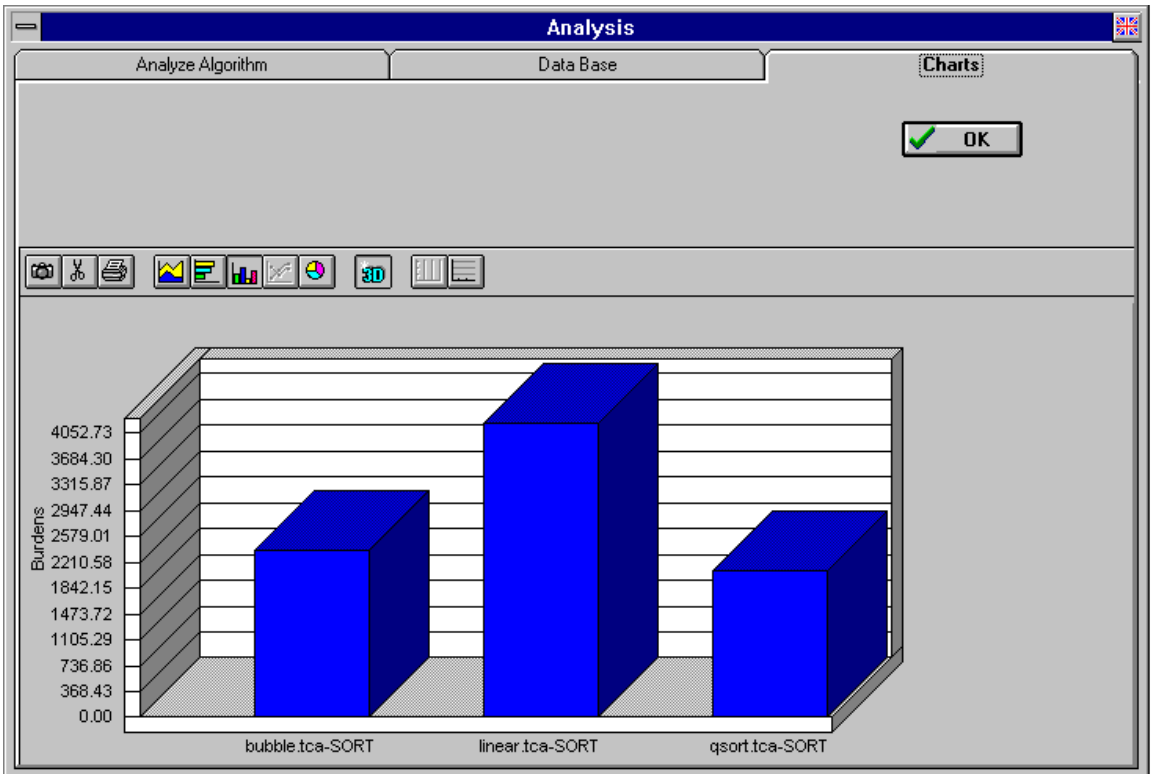
a)Έστω ότι θέλουμε να εμφανίσουμε όλους τους αλγορίθμους ταξινόμησης (SORT), των οποίων ο φόρτος να είναι μικρότερος του 1250. Θα επιλέξουμε στο πλαίσιο επιλογής "Algorithm Types" τους αλγόριθμους SORT. Στο πλαίσιο επιλογής "Burden" θα επιλέξουμε 1250 και στο διπλανό πλαίσιο επιλογής ως προς τη μορφή του κριτηρίου σύγκρισης θα διαλέξουμε "Less"(μικρότεροι). Αυτό θα έχει ως αποτέλεσμα να εμφανιστούν στο πλέγμα όλοι οι αλγόριθμοι ταξινόμησης των οποίων ο φόρτος είναι μικρότερος από 1250.

b) Έστω τώρα ότι θέλουμε να εμφανίσουμε όλους τους αλγορίθμους αναζήτησης (SEARCH), των οποίων ο φόρτος να είναι μικρότερος του 500, και έχουν καταχωρηθεί στην βάση μετά την 4η Ιανουαρίου 1995. Θα επιλέξουμε στο πλαίσιο επιλογής "Algorithm Types" τους αλγόριθμους SEARCH. Στο πλαίσιο επιλογής "Burden" θα επιλέξουμε 500 και στο διπλανό πλαίσιο επιλογής ως προς τη μορφή του κριτηρίου σύγκρισης θα διαλέξουμε "Less"(μικρότεροι). Στο πλαίσιο επιλογής "Date of Algorithm" θα επιλέξουμε 4/1/95 και στο διπλανό πλαίσιο επιλογής ως προς τη μορφή του κριτηρίου σύγκρισης θα διαλέξουμε "Greater"(μεγαλύτερο). Αυτό θα έχει ως αποτέλεσμα να εμφανιστούν στο πλέγμα όλοι οι αλγόριθμοι αναζήτησης των οποίων ο φόρτος είναι μικρότερος από 500, και έχουν καταχωρηθεί στην βάση μετά την 4η Ιανουαρίου 1995.

Σε αυτό το σημείο θα πρέπει να διεκρινήσουμε ότι όταν γίνεται η ανάλυση του αλγορίθμου και η ενημέρωση της βάσης δεδομένων, ΔΕΝ εισάγεται στην βάση ο φόρτος του αλγορίθμου, αλλά μόνο η παραμετρική εξίσωση εκείνη που εκφράζει την πολυπλοκότητά του. Αυτό γίνεται γιατί ο χρήσης δεν έχει ακόμα εφοδιάσει στο σημείο εκείνο τον αλγόριθμο με αντίστοιχα αριθμητικά δεδομένα, μέσω των οποίων θα μπορούσε να υπολογιστεί αλγεβρικά η τιμή της προαναφερθείσας εξίσωσης. Ο σχεδιασμός του προγράμματος έγινε σκόπιμα με τέτοιων τρόπο, ούτως ώστε να έχει την δυνατότητα ο χρήστης , από την σελίδα "Database" και αμέσως πριν ετοιμάσει ένα συγκριτικό γράφημα φόρτου αλγορίθμων, να εισάγει αριθμητικά δεδομένα για τις διαστάσεις των μεταβλητών που μετέχουν στον εκάστοτε αλγόριθμο. Έτσι, μπορεί να φροντίσει να συμφωνούν τα αριθμητικά δεδομένα μεταξύ των διαφόρων αλγορίθμων. Η εισαγωγή αριθμητικών διαστάσεων των μεταβλητών πραγματοποιείται από την σελίδα "Database" και με τη χρήση του πλήκτρου ενέργειας "Compute". π.χ. αν ο χρήστης επιθυμεί να εμφανίσει στο πλέγμα όλους τους αλγορίθμους ταξινόμησης μονοδιάστατων πινάκων, έχει την δυνατότητα (μετά την εμφάνισή τους στο πλέγμα)να εισάγει τις διαστάσεις των πινάκων που λαμβάνουν μέρος στους συγκεκριμένους αλγόριθμους, και στην εφαρμογή στην οποία εργάζεται εκείνη την χρονική περίοδο, και κατόπιν να εμφανίσει (όπως θα δούμε παρακάτω) ένα γράφημα συγκριτικής παρουσίασης αυτών των συγκεκριμένων αλγορίθμων με τις δοθείσες διαστάσεις.

Τα δύο πλαίσια κειμένου που βρίσκονται κάτω από το πλέγμα εμφάνισης των εγγραφών του πίνακα της βάσης που περιέχει τους αλγόριθμους αφορούν στον αλγόριθμο και στην παραμετρική εξίσωση που συμβολίζει την πολυπλοκότητα αυτού, της τρέχουσας επιλεγμένης εγγραφής (οι επιλεγμένες εγγραφές στο πλέγμα διακρίνονται από ένα βέλος στην αρχή της εγγραφής)

Αφού λοιπόν ο χρήστης έχει πια επιλέξει ένα υποσύνολο των αλγορίθμων που έχει κατά καιρούς εισάγει στη βάση και αυτοί έχουν εμφανισθεί στο προαναφερθέν πλέγμα, έχει την δυνατότητα να μεταβεί στην σελίδα "Charts" όπου θα είναι ήδη έτοιμο το συγκριτικό γράφημα παρουσίασης των αλγορίθμων που θα έχει προεπιλέξει.



οθόνη γραφημάτων

Οι επιλογές όσον αφορά στο είδος του γραφήματος και στην διαχείρισή του συμβάλλουν στην λειτουργικότητα της όλης εφαρμογής αλλά και στην διασυνδεσιμότητα της με άλλες εφαρμογές. (π.χ. ο χρήστης μπορεί να ετοιμάσει μερικούς αλγόριθμους, να τους συγκρίνει και να ζητήσει τον σχεδιασμό του συγκριτικού γραφήματος και κατόπιν να εισάγει το γράφημα (μέσα από την εφαρμογή που παρουσιάζουμε) σε κάποια άλλη εφαρμογή ! Ας δούμε όμως λίγο πιο αναλυτικά τι σημαίνουν οι διάφορες επιλογές που έχει στην διάθεσή του ο χρήστης :

Το **πρώτο σύνολο πλήκτρων** (δύο πλήκτρα) φέρουν τις εξής λειτουργίες :

το πρώτο πλήκτρο χρησιμοποιείται για την μεταφορά όλου του γραφήματος (ως εικόνας) σε μια άλλη εφαρμογή. Αν πατήσουμε αυτό το πλήκτρο και μετά σε μια οποιαδήποτε άλλη εφαρμογή χρησιμοποιήσουμε την λειτουργία 'Paste', τότε το γράφημά μας μεταφέρεται σε εκείνη την εφαρμογή, είτε είναι επεξεργαστής κειμένου, είτε στατιστικό πακέτο , είτε οτιδήποτε άλλο ! Αρκεί η άλλη εφαρμογή να υποστηρίζει λήψη δεδομένων από το clipboard. Όσοι δεν γνωρίζετε τι είναι αυτό δεν πρέπει να ανησυχήσετε γιατί οι περισσότερες , αν όχι όλες, εφαρμογές που λειτουργούν κάτω από το περιβάλλον MS Windows στις ημέρες μας υποστηρίζουν αυτήν την δυνατότητα.

Το δεύτερο πλήκτρο εκτυπώνει το γράφημά μας στον εκτυπωτή που έχουμε εμείς επιλέξει (είτε μέσα από τα Windows είτε από την λειτουργία 'Printer Setup' που βρίσκεται στον βασική φόρμα της εφαρμογής αυτής.

Το **δεύτερο σύνολο πλήκτρων** χρησιμοποιούνται για να τροποποιήσουν το είδος του γραφήματος που εμφανίζεται. Πιο συγκεκριμένα έχουμε την επιλογή να εμφανίσουμε γράφημα με μπάρες (bar

chart), γραμμικό, ή γράφημα πίτας (pie chart). Και για τα τρία έχουμε την επιλογή να τα εμφανίσουμε είτε τρισδιάστατα είτε δυσδιάστατα !

Το **τρίτο σύνολο πλήκτρων** χρησιμοποιείται για να ενεργοποιήσουν ή να απενεργοποιήσουν την εμφάνιση οριζόντιου ή και κατακόρυφου πλέγματος πίσω από το γράφημα.

Αυτή ήταν μια σύντομη παρουσίαση των δυνατοτήτων που προσφέρει η εφαρμογή αυτή. Περισσότερες και πιο εξειδικευμένες πληροφορίες μπορεί να αναζητήσει κανείς στο context-sensitive help.

5. Παραρτήματα

5.1 Hardware/Software requirements

Hardware Requirements

Η **απαιτούμενη σύνθεση για την λειτουργία** της εφαρμογής είναι η εξής :
386 SX-33 Mhz, 4MB RAM, 5MB σκληρού δίσκου διαθέσιμα

Η **προτεινόμενη σύνθεση για την λειτουργία** της εφαρμογής είναι η εξής :
486 DX-80 Mhz, 8MB RAM, 15MB σκληρού δίσκου διαθέσιμα

Ας σημειώσουμε σε αυτό το σημείο ότι για την πλήρη **εγκατάσταση** της εφαρμογής, θα χρειαστούν
περίπου 20MB διαθέσιμου χώρου στον σκληρό δίσκο.

Software Requirements

Θα πρέπει να έχει εγκατασταθεί στον υπολογιστή ένα τουλάχιστο παραθυρικό λειτουργικό σύστημα της Microsoft. Δηλαδή ένα εκ των ακόλουθων :

1. MS-DOS και MS-Windows 3.x
2. MS-Win95
3. MS-WinNT

Επίσης, θα πρέπει να έχει εγκατασταθεί και η Borland Database Engine. Την redistributable έκδοση της BDE διανέμουμε μαζί με την υπόλοιπη εφαρμογή, σε περίπτωση που δεν την έχετε.

5.2 Installation Instructions

Οι δισκέτες εγκατάστασης είναι οι εξής :

- Δύο δισκέτες εγκατάστασης του Algorithm Comparison Tool (ACT). Σε αυτές βρίσκεται το κυρίως τμήμα της εφαρμογής. Η επιλογή των τμημάτων που θέλετε να εγκαταστήσετε γίνεται με απλό τρόπο κατά την διάρκεια της εγκατάστασης. Η εγκατάσταση γίνεται εκτελώντας το πρόγραμμα setup.exe, το οποίο βρίσκεται στην πρώτη δισκέτα.

- Δύο δισκέττες εγκατάστασης της Redistributable Borland Database Engine (BDE). Είναι ένα εργαλείο που προσφέρει η Borland σε όσους χρησιμοποιούν τα περιβάλλοντα ανάπτυξης εφαρμογών της, το οποίο προσφέρει κάποιου είδους διασύνδεση ανάμεσα στις διάφορες βάσεις δεδομένων. Αν δεν έχετε εγκατεστημένο το BDE στον υπολογιστή σας, πρέπει να το εγκαταστήσετε **πριν από το ACT.** Η εγκατάσταση γίνεται εκτελώντας το πρόγραμμα setup.exe, το οποίο βρίσκεται στην πρώτη δισκέτα.

5.3 Δημιουργία DLL

Παραγωγή DLL

Ας δούμε λοιπόν λίγο πιο αναλυτικά πως μπορεί να δημιουργήσει κανείς βιβλιοθήκες DLL, σε διάφορες γλώσσες, και κατόπιν να τις ενσωματώσει σε κάποιο περιβάλλον ανάπτυξης εφαρμογών για τα MS-Windows. Πιο συγκεκριμένα θα μιλήσουμε για την παραγωγή DLL σε Borland/Microsoft C/C++ ,Borland Pascal,Borland Delphi, Microsoft Fortran και το πως αυτά μπορούν να χρησιμοποιηθούν από τις : Borland Delphi, Borland Pascal for Windows, Microsoft/Borland C/C++.Για την περίπτωση παραγωγής dll από MS Fortran,Borland Pascal και χρήσης αυτών από το Borland Delphi θα δώσουμε και συγκεκριμένα παραδείγματα πηγαίου κώδικα (source code).

δηλώσεις υποπρογραμμάτων

Όταν καλείται μια υπορουτίνα σε ένα πρόγραμμα (είτε είναι υπορουτίνα που δεν επιστρέφει στην ίδια της τη διεύθυνση κάποιο αποτέλεσμα που προκύπτει από την επεξεργασία των δεδομένων που λαμβάνει [procedure, subroutine] είτε επιστρέφει [function]) ακριβώς πριν να αρχίσει η εκτέλεσή της αποθηκεύονται στο stack το code segment (αν ήταν far) και το ip στο οποίο βρισκόταν το πρόγραμμα που την κάλεσε, καθώς και οι διάφορες παράμετροι που πιθανώς να έχει εισάγει ο προγραμματιστής στην υπορουτίνα αυτή. Όλα αυτά τα δεδομένα βέβαια αποθηκεύονται με μια δεδομένη (για κάθε compiler) σειρά. Υπάρχουν δύο σειρές αποθήκευσης αυτών των δεδομένων στο stack ιδιαίτερα διαδεδομένες στους compilers που κυκλοφορούν στις ημέρες μας. Αυτό λοιπόν είναι ένα σημείο το οποίο θα πρέπει να προσέχουμε κατά την δημιουργία βιβλιοθηκών dll. π.χ. αν αναπτύσσουμε μια βιβλιοθήκη dll σε C/C++ και θελήσουμε να την χρησιμοποιήσουμε σε περιβάλλον Delphi θα πρέπει να προσέξουμε ιδιαίτερα στην δήλωση των υπορουτινών μας στο dll. Θα πρέπει να δηλώσουμε οτι θέλουμε να μεταφραστούν (compiled)σαν να επρόκειτο για υπορουτίνες Delphi, γιατί οι C/C++ χρησιμοποιούν τον ένα από τους δύο τρόπους αποθήκευσης στο stack κατά την κλήση υποπρογραμμάτων ενώ το Delphi χρησιμοποιεί τον άλλο. Στη συγκεκριμένη περίπτωση αυτό επιτυγχάνεται με τη χρήση της "κρατημένης λέξης-κλειδί" (reserved keyword)'PASCAL' της C/C++. Κάτι άλλο που θα πρέπει να μας απασχολήσει είναι το πώς "περνάει" η κάθε γλώσσα τα ορίσματα μιας υπορουτίνας , στην υπορουτίνα την ίδια αν εμείς βέβαια δεν δηλώσουμε με κάποιον συγκεκριμένο τρόπο ότι επιθυμούμε 'by value' ή 'by reference parameter passing'. Για παράδειγμα, η Fortran θεωρεί ως default το 'by reference parameter passing' ενώ η Delphi το by 'value' !

εξαγώγιμα δεδομένα από βιβλιοθήκη dll

Άμεσα, μόνο υποπρογράμματα (και φυσικά τα ορίσματα εξόδου που συνοδεύουν αυτές τα υποπρογράμματα) μπορούμε να εξάγουμε από μία dll, για χρήση σε ένα οποιοδήποτε περιβάλλον προγραμματισμού των MS-Windows. Πρακτικά, μπορούμε να εξάγουμε και σταθερές, σχεδόν οποιουδήποτε τύπου, ορίζοντας τις ως τιμή επιστροφής μιας κενής κατά τα άλλα συνάρτησης βιβλιοθήκης, π.χ. στην κατασκευή ενός dll σε Pascal μπορούμε να εξάγουμε την σταθερά 32768 ως εξής:

```
function WhatIsTheMaxInteger:integer;
begin
WhatIsTheMaxInteger:=32768;
end;
```

Λεπτομερέστερα παραδείγματα θα δοθούν αργότερα. Δυστυχώς, η εξαγωγή τύπων δεδομένων είναι πρακτικά μάλλον αδύνατη. Το σημείο στο οποίο θα πρέπει όμως να δώσουμε ιδιαίτερη προσοχή είναι η συμβατότητα των διαφόρων τύπων δεδομένων. Για παράδειγμα η δομή δεδομένων integer στην Fortran αντιστοιχεί σε έναν ακέραιο 4 bytes, ενώ στην Pascal σε ακέραιο των δύο bytes! Αν λοιπόν δημιουργούμε dll Σε Fortran, για την Pascal, θα πρέπει να δηλώσουμε τους ακέραιους αριθμούς που θα εξάγουμε μέσω των υπορουτινών μας ως integer*2.

Επίσης, τα τελευταία χρόνια η γνωστή σε όλους μας δομή string ή οποία αντιστοιχεί σε μία σειρά 255 χαρακτήρων, ο πρώτος από τους οποίους είναι ο αριθμός των χαρακτήρων που έχουν αποθηκευθεί στο string, αντικαθίσταται αργά αλλά σταδιακά (στις περισσότερες γλώσσες) με μια άλλη δομή δεδομένων που χρησιμοποιούσε από παλιότερα η C/C++ και η οποία αντιστοιχεί σε μια σειρά άπειρων θεωρητικά χαρακτήρων (αυτό εξαρτάται βέβαια από την διαθέσιμη μνήμη). Το πέρας του ορμαθού αυτού χαρακτήρων εντοπίζεται από τον τελευταίο σε σειρά χαρακτήρα ο οποίος οφείλει

να είναι πάντα ο #0. Καλό θα ήταν λοιπόν να χρησιμοποιούμε τον δεύτερο τύπο δεδομένων στην διαχείριση και δημιουργία dll, για λόγους συμβατότητας. Αν δεν το κάνουμε... τις περισσότερες φορές θα βρεθούμε προ δυσάρεστων εκπλήξεων σχετικά με την αποσφαλμάτωση της δυναμικής βιβλιοθήκης που κατασκευάζουμε.

εφαρμογή / κατασκευή dll σε MS-Fortran, Borland Pascal

Θα κατασκευάσουμε ένα dll σε MS-Fortran και σε Borland Pascal, το οποίο απλά θα περιέχει μια υπορουτίνα η οποία θα προσθέτει δύο ακέραιους και θα επιστρέφει το αποτέλεσμα της πρόσθεσης. Σκόπιμα επιλέχθηκε μια τόσο απλή ρουτίνα προκειμένου να εστιάσουμε περισσότερο πάνω στην κατασκευή dll.

source code in Borland Turbo Pascal

```
library addnums;

procedure AddIntegers(a,b integer ; var IntResult :integer); export;
begin
IntResult := A + b ;
end;

exports AddIntegers index 1;

end.
```

Όπως παρατηρούμε και από την εντολή exports, πρέπει να δοθεί και ένα index σε κάθε υπορουτίνα που εξάγουμε. Αυτό γίνεται για την ταχύτερη αλλά και ευκολότερη ανεύρεσή της. Καλό θα ήταν να το συμπεριλαμβάνουμε πάντα.

Ας δούμε το ίδιο dll γραμμένο σε MS-Fortran ver5.1. Εδώ θα χρειαστούν δύο αρχεία. Το πρώτο θα είναι αυτό που θα περιέχει την υπορουτίνα και το δεύτερο θα είναι ένα def αρχείο στο οποίο θα πρέπει να ορίσουμε μερικές από τις βασικές παραμέτρους -runtime κυρίως- συμπεριφοράς του dll που θα κατασκευάσουμε. Το αρχείο που περιέχει την υπορουτίνα είναι λοιπόν το εξής :

κώδικας σε MS-Fortran ver5.1

```
subroutine AddIntegers(a, b, IntResult)
integer *2 a, b, IntResult
IntResult = a + b
end
```

και το αρχείο def :

κώδικας σε MS-Fortran ver5.1

```
library      AddNums
description 'sample dll, with subroutine adding two integers'
apploader   '__mslangload'
exetype     Windows 3.0
code         preload moveable discardable
data preload moveable single
heapsize     1024
exports      AddIntegers,
        WEP
```

Το να εξηγήσουμε τι σημαίνει η κάθε μία γραμμή κώδικα που μόλις είδαμε είναι πέρα από τους στόχους αυτής της τεχνικής αναφοράς. Αυτό θα απαιτούσε άλλωστε μεγάλη κάλυψη, και εξ' άλλου πληροφορίες για αυτά μπορείτε να βρείτε σε όλα τα βιβλία για MS-Fortran ver 5.1. Θα περιοριστούμε στο να πούμε ότι δεν θα ήταν φρόνιμο να αλλάξετε οτιδήποτε άλλο εκτός βέβαια από το όνομα της εκάστοτε βιβλιοθήκης που αναπτύσσεται (εδώ : addnums) και το όνομα της υπορουτίνας (εδώ : AddIntegers), αν δεν γνωρίζετε ακριβώς για ποιό λόγο επιθυμείτε να τα αλλάξετε. Τα υπόλοιπα αποτελούν τις πιο καθιερωμένες επιλογές που θα μπορούσε να κάνει κανείς και λειτουργούν στη μεγίστη πλειοψηφία των περιπτώσεων. Τέλος, να πούμε ότι το αρχείο στο οποίο βρίσκεται ο

παραπάνω κώδικας θα πρέπει να φέρει το όνομα του αρχείου όπου βρίσκεται ο πηγαίος κώδικας της υπορουτίνας και επέκταση def. Αν δηλαδή η υπορουτίνα είναι αποθηκευμένη στο addnums.for , αυτό το αρχείο θα πρέπει να ονομάζεται addnums.def.

5.4 Σύνδεση εξωτερικών DLL

Η κατασκευή βιβλιοθηκών DLL είναι πολλές φορές μια περίπλοκη διαδικασία. Εμείς θα προσπαθήσουμε να εξηγήσουμε με απλά λόγια πως πρέπει να κατασκευαστεί ένα DLL προκειμένου να ενσωματωθεί στην εφαρμογή αυτή.

Κάθε Dynamic Link Library πρέπει να αποτελεί ένα ολοκληρωμένο module. Πρέπει να έχει την δυνατότητα να εγκαινιάσει τις μεταβλητές που χρησιμοποιεί και να εκτελεί και τους κατάλληλους ελέγχους λαθών. Επίσης, πρέπει να έχει την δυνατότητα να απελευθερώνει τους πόρους που καταλαμβάνουν οι ρουτίνες που περιέχει κατά το χρονικό διάστημα στο οποίο αυτές εκτελούνται. Σε αυτό το σημείο θα πρέπει να διασαφηνίσουμε το εξής : η απελευθέρωση της μνήμης που καταλαμβάνει η βιβλιοθήκη αυτή καθεαυτή δεν πρέπει να γίνεται από τον προγραμματιστή. Γίνεται αυτόματα από τα MS-Windows, όταν κριθεί απαραίτητο (θεωρούμε ότι δεν είναι σκόπιμο να εξηγήσουμε σε αυτό το σημείο πότε το περιβάλλον MS-Windows απελευθερώνει τη μνήμη που καταλάμβανε μέχρι εκείνη τη στιγμή μια βιβλιοθήκη DLL). Οποιαδήποτε προσπάθεια του προγραμματιστή να απελευθερώσει ο ίδιος τη μνήμη αυτή θα οδηγήσει σε απρόβλεπτες καταστάσεις σφαλμάτων.

Η κατασκευή DLL που θα ενσωματωθούν στην εφαρμογή αυτή παρουσιάζει την εξής διαφορά: Δεν πρέπει να πραγματοποιείται οποιοσδήποτε έλεγχος έχει σχέση με το stack.. Το stack που χρησιμοποιούν οι βιβλιοθήκες DLL που καλούνται από εφαρμογές σχεδιασμένες και μεταφρασμένες από το Delphi είναι αυτό που χρησιμοποιούν αυτές οι ίδιες εφαρμογές, συνεπώς ο έλεγχος του εναπόκειται σε αυτές. Δεν πρέπει λοιπόν να υλοποιήσουμε κανένα έλεγχο για το stack και επίσης θα πρέπει να απαγορεύσουμε στον μεταφραστή (compiler) της γλώσσας που χρησιμοποιούμε να προσθέσει ρουτίνες αυτόματου ελέγχου stack στην βιβλιοθήκη.

Τέλος, θα πρέπει να υπάρχουν οπωσδήποτε οι ακόλουθες δύο υπορουτίνες :

Μια συνάρτηση (function) η οποία να επιστρέφει τύπο δεδομένων pchar (pointer to characters). Η συγκεκριμένη αλφαριθμητική τιμή που θα επιστρέφει η συνάρτηση αυτή θα είναι ο τίτλος της επιλογής στο μενού, από την οποία θα μπορούμε να εκτελέσουμε τον κώδικα που περιέχεται στην βιβλιοθήκη.

Μια διαδικασία, ή αλλιώς υπορουτίνα, (procedure, subroutine) η οποία θα περιέχει τον αναγκαίο κώδικα εγκαινίασης (αρχικοποίηση των μεταβλητών της βιβλιοθήκης, δέσμευση χώρου στη μνήμη για δείκτες κτλ), θα εκτελεί την "προσθήκη" και τέλος για τον τερματισμό αυτής και την αποδέσμευση του αντίστοιχου χώρου μνήμης.

Ας δούμε ένα παράδειγμα για να καταλάβουμε καλύτερα τα παραπάνω. Έστω ότι ο χρήστης της εφαρμογής αυτής επιθυμεί να δημιουργήσει ένα ημερολόγιο και να το ενσωματώσει στις λειτουργίες της. Θα πρέπει να δημιουργήσει όλον τον απαραίτητο κώδικα σε μορφή Dynamic Link Library. ΈΣτω ότι την βιβλιοθήκη την ονομάζει MyCalendar και ότι επιθυμεί να εμφανίζεται σαν επιλογή στα μενού της εφαρμογής με το όνομα "Full Calendar". Οι ενέργειες που θα πρέπει να κάνει, αφού κατασκευάσει το προαναφερθέν DLL είναι οι εξής :

Υλοποίηση μιας συνάρτησης, μέσα στο DLL, η οποία να επιστρέφει τύπο δεδομένων pchar και τιμή "Full Calendar". Το όνομα αυτής της συνάρτησης θα **πρέπει** να είναι

MenuNameofAddOnMyCalendar.

Πρέπει επίσης να υλοποιήσει μια υπορουτίνα η οποία θα καλεί το ημερολόγιο προς εκτέλεση. Ο κώδικας για το ημερολόγιο αυτός καθαυτός είναι δυνατόν να περιέχεται σε μια η παραπάνω υπορουτίνες του DLL αυτού. Η εκτέλεσή του όμως θα πρέπει να γίνεται μέσα από την υπορουτίνα στην οποία αναφερόμαστε τώρα. Το όνομα αυτής της υπορουτίνας θα **πρέπει** να είναι

ExecAddOnMyCalendar

Όταν μεταφράσουμε αυτό το DLL πρέπει να το τοποθετήσουμε στον υποκατάλογο AddOn (AddOn subdirectory) της εφαρμογής που παρουσιάζουμε. Κάθε φορά που εκτελούμε την εφαρμογή, θα γίνεται σχετικός έλεγχος στον υποκατάλογο AddOn και αυτό το DLL "προσθήκης" (καθώς και όσα ακόμα υπάρχουν σε αυτόν τον κατάλογο) , θα ενσωματώνονται στα μενού επιλογών.

Η χρησιμότητα αυτής της δυνατότητας θα φανεί περισσότερο αν αποφασίσει κάποιος χρήστης να δημιουργήσει μια προσθήκη η οποία να αντλεί δεδομένα από την βάση της εφαρμογής και να εκτελεί περαιτέρω επεξεργασία πάνω στους αλγορίθμους του, χωρίς να χρειάζεται να φύγει από το συγκεκριμένο πρόγραμμα.

6. ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Aho, S. Ravi, J. D. Ullman, “Compilers, Principles, Techniques and Tools“, Addison-Wesley, 1985

2. Cormen, C. E. Leiserson, R. L. Rivest, “Introduction to Algorithms”, MIT Press / McGraw-Hill Book Company, 1990

3. Ασημάκης, “Αλγόριθμοι για την επίλυση της εξίσωσης Riccati”, Διδακτορική Διατριβή στο τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών & Πληροφορικής της Πολυτεχνικής σχολής του Πανεπιστημίου Πατρών

4. Lainiotis, N. D. Assimakis, S. K. Katsikas, “Fast and Stable Algorithm for Computing the Principal Sqaure Root of a Complex Matrix”, Neural, Parallel and Scientific Computations, 1993

5. C. Calvert, ”Delphi Unleashed”, SAMS Publishing, 1995

6. O’ Brien, S. Nameroff, “Turbo Pascal 7 , the complete reference”, McGraw-Hill, 1993

7. Microsoft Fortran v5.1 Programming Manuals

8. Watson, “Programming with Turbo Vision”, MIS Press, 1994

9. Borland Paradox v 7.0 Programming Manuals

10. Microsoft Assembler v5.0 Programming Manuals