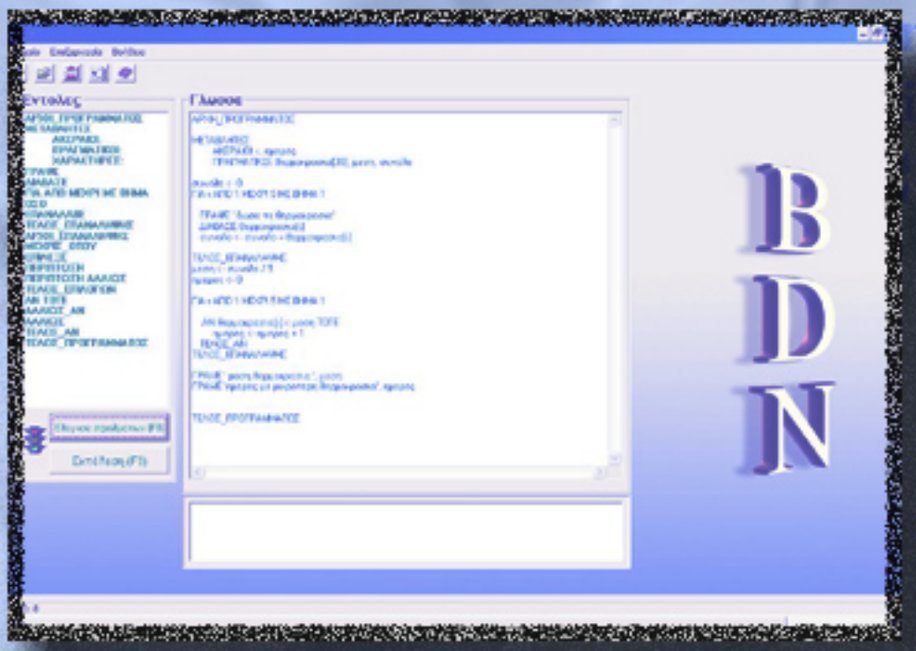
ΤΕΙ ΣΕΡΡΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ




```
mikos_metablitis=-1;
for(i;i<=b;i++)
if(A[i]!=' ')
 metafrasi(Sende
 onomametablitis
 c_metabl++;
 mhkos2++;
 mikos_metablitis
 }

fputs(")",pf);
fputs("\n{\n",pf);
c_metabl2=c_metabl;
c_metabl=0;
```

Ανάπτυξη ψευδογλώσσας για υλοποίηση εφαρμογών σε οπτικό περιβάλλον

Μπάρκογλου Νίκος **Δούλαλας Νίκος**

ΣΕΡΡΕΣ 2006

ΚΑΤΑΣΚΕΥΗ ΠΡΟΓΡΑΜΜΑΤΟΣ ΑΝΑΠΤΥΞΗΣ ΨΕΥΔΟΓΛΩΣΣΑΣ ΓΙΑ ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΩΝ ΣΕ ΟΠΤΙΚΟ ΠΕΡΙΒΑΛΛΟΝ.

των

Μπάρκογλου Νικόλαου
&
Δούλαλα Νικόλαου

Πτυχιακή εργασία που υποβάλλεται προς
μερική εκπλήρωση των απαιτήσεων για την
απόκτηση του πτυχίου.

Τεχνολογικό Εκπαιδευτικό Ίδρυμα Σερρών

Οκτώβριος 2006

Εγκρίθηκε από τον _____

_____
_____
_____

Ημερομηνία_____

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΛΗΨΗ

Σκοπός της πτυχιακής εργασίας είναι η ανάπτυξη προγράμματος σε οπτικό περιβάλλον για την ανάπτυξη και την υλοποίηση εφαρμογών. Η γλώσσα που θα χρησιμοποιηθεί για την υλοποίηση της εργασίας είναι η C++.

Σε πρώτη φάση δημιουργούμε ένα οπτικό περιβάλλον στο οποίο ο χρήστης της εφαρμογής θα έχει τη δυνατότητα να 'γράφει' τον κώδικα στην γλώσσα BDN, τηρώντας πάντα κάποιους βασικούς κανόνες για τον τρόπο γραφής της γλώσσας, τους οποίους και θα αναλύσουμε στη συνέχεια.

Σε δεύτερη φάση, δημιουργούμε έναν Μεταγλωττιστή, που έχει ως σκοπό τη μετάφραση του πηγαίου προγράμματος που ο χρήστης έχει γράψει. Έτσι παράγουμε το ισοδύναμο πρόγραμμα της ΓΛΩΣΣΑΣ BDN, στη γλώσσα στόχο που είναι η C++. Όλη αυτή η διαδικασία γίνεται με το πάτημα του κουμπιού Έλεγχος σφαλμάτων. Αφού ολοκληρωθεί η διαδικασία της μεταγλώττισης στη συνέχεια με το πάτημα του κουμπιού Εκτέλεση από το χρήστη, θα εκτελείται η εφαρμογή.

1. ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

1.1. ΣΚΟΠΟΣ ΥΠΑΡΞΗΣ ΤΩΝ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Από τη δημιουργία του πρώτου υπολογιστή μέχρι σήμερα έχουν αλλάξει πάρα πολλά πράγματα. Οι πρώτοι υπολογιστές, τεράστιοι σε μέγεθος αλλά με πάρα πολύ περιορισμένες δυνατότητες και μικρές ταχύτητας επεξεργασίας εξελίχθηκαν σε πολύ μικρούς σε μέγεθος υπολογιστές με τεράστιες όμως δυνατότητες και ταχύτητες επεξεργασίας.

Οι γλώσσες προγραμματισμού αναπτύχθηκαν με σκοπό να μπορούν οι άνθρωποι να χρησιμοποιούν τον ηλεκτρονικό υπολογιστή για χρήσιμα πράγματα. Επειδή οι άνθρωποι δεν έχουν την ίδια ικανότητα στη σχέση τους με τους Η/Υ και επειδή υπάρχει μια εξαιρετικά, μεγάλη ποικιλία πολύπλοκων εργασιών, που μπορούν να κάνουν οι Η/Υ, έχουν δημιουργηθεί πολλές γλώσσες προγραμματισμού. Κάθε γλώσσα δημιουργήθηκε για να καλύψει μια ειδική ανάγκη. Υπάρχουν γλώσσες για παιδιά και για αρχάριους, γλώσσες για επιστήμονες, γλώσσες για την διευκόλυνση της δακτυλογράφησης, γλώσσες για εργασίες που απαιτούν τη συνολική προσπάθεια κωδικοποίησης αρκετών προγραμματιστών, γλώσσες που εκμεταλλεύονται στο έπακρο την ταχύτητα του Η/Υ, γλώσσες που έχουν σχεδιαστεί για να εφαρμοστούν μόνο σε πολύ μικρούς Η/Υ, γλώσσες που λειτουργούν καλύτερα σε μεγάλους Η/Υ κ.λ.π. Σε κάθε περίπτωση σκοπός της γλώσσας προγραμματισμού είναι να επικοινωνεί ο άνθρωπος με τον υπολογιστή για την επίλυση προβλημάτων.

1.2. ΑΛΓΟΡΙΘΜΟΙ + ΔΕΔΟΜΕΝΑ=ΠΡΟΓΡΑΜΜΑΤΑ

Η επίλυση ενός προβλήματος με τον υπολογιστή περιλαμβάνει τρία εξίσου σημαντικά στάδια.

- Τον ακριβή προσδιορισμό του προβλήματος.
- Την ανάπτυξη του αντίστοιχου αλγορίθμου.
- Την διατύπωση του αλγόριθμου σε κατανοητή μορφή από τον υπολογιστή.

Δεδομένα:

Τα δεδομένα είναι ακατέργαστα γεγονότα, και κάθε φορά η επιλογή τους εξαρτάται από τον τύπο του προβλήματος. Η συλλογή των ακατέργαστων δεδομένων και ο συσχετισμός τους, δίνουν ως αποτέλεσμα την πληροφορία (information). Ο αλγόριθμος είναι το μέσο για την παραγωγή πληροφορίας από τα δεδομένα.

Αλγόριθμος:

Η θεωρία των αλγορίθμων έχει μεγάλη παράδοση και η ηλικία μερικών αλγορίθμων αριθμεί χιλιάδες χρόνια, όπως για παράδειγμα ο αλγόριθμος του Ευκλείδη για την εύρεση του μέγιστου κοινού διαιρέτη δυο αριθμών ή το λεγόμενο κόσκινο του Ερατοσθένη για την εύρεση των πρώτων αριθμών από 1 ως n. Σήμερα το πεδίο της Θεωρίας Αλγόριθμων είναι ένα ιδιαίτερα ευρύ και πλούσιο πεδίο. Πληθώρα συγγραμμάτων έχει εμφανισθεί στη βιβλιογραφία, ενώ συνεχίζεται η περαιτέρω εμβάθυνση σε νέα σύγχρονα προβλήματα. Οι περισσότεροι από τους αλγορίθμους που συνήθως εξετάζονται στα σχετικά βιβλία έχουν προταθεί τα τελευταία 25 χρόνια, όση περίπου είναι και η ηλικία της Πληροφορικής ως μίας νέας αυθύπαρκτης επιστήμης.

Ο όρος αλγόριθμος, λοιπόν χρησιμοποιείται για να δηλώσει μεθόδους που εμφανίζονται για την επίλυση προβλημάτων. Ωστόσο, ένας πιο αυστηρός ορισμός της έννοιας αυτής είναι ο εξής:

Ορισμός: Αλγόριθμος είναι μια πεπερασμένη σειρά ενεργειών, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο χρόνο, που στοχεύουν στην επίλυση ενός προβλήματος.

Κάθε αλγόριθμος απαραίτητα ικανοποιεί τα επόμενα κριτήρια

- **Είσοδος** (input). Καμία, μία ή περισσότερες τιμές δεδομένων πρέπει να δίνονται ως είσοδοι στον αλγόριθμο. Η περίπτωση που δεν δίνονται τιμές δεδομένων εμφανίζεται, όταν ο αλγόριθμος δημιουργεί και επεξεργάζεται κάποιες πρωτογενείς τιμές με τη βοήθεια συναρτήσεων παραγωγής τυχαίων αριθμών ή με τη βοήθεια άλλων απλών εντολών.
- **Έξοδος** (output). Ο αλγόριθμος πρέπει να δημιουργεί τουλάχιστον μία τιμή δεδομένων ως αποτέλεσμα προς το χρήστη ή προς έναν άλλο αλγόριθμο.
- **Καθοριστικότητα** (definiteness). Κάθε εντολή πρέπει να καθορίζεται χωρίς καμία αμφιβολία για τον τρόπο εκτέλεσης της. Λόγου χάριν, μία εντολή διαίρεσης πρέπει να θεωρεί και την περίπτωση, όπου ο διαιρέτης λαμβάνει μηδενική τιμή.
- **Περατότητα** (finiteness). Ο αλγόριθμος να τελειώνει μετά από πεπερασμένα βήματα εκτέλεσης των εντολών του. Μία διαδικασία που δεν τελειώνει μετά από ένα συγκεκριμένο αριθμό βημάτων δεν αποτελεί αλγόριθμο, αλλά λέγεται απλά υπολογιστική διαδικασία(computational procedure).
- **Αποτελεσματικότητα** (effectiveness). Κάθε μεμονωμένη εντολή του αλγορίθμου να είναι απλή. Αυτό σημαίνει ότι μία εντολή δεν αρκεί να έχει οριστεί, αλλά πρέπει να είναι και εκτελέσιμη.

Πρόγραμμα:

Ο προγραμματισμός είναι αυτός που ασχολείται με τη διατύπωση του αλγορίθμου σε κατανοητή για τον υπολογιστή μορφή, τη δημιουργία δηλαδή του προγράμματος που δεν είναι τίποτε άλλο από το σύνολο των εντολών που πρέπει να δοθούν στον υπολογιστή, ώστε να υλοποιηθεί ο αλγόριθμος για την επίλυση του προβλήματος. Το πρόγραμμα, το οποίο γράφεται σε κάποια γλώσσα προγραμματισμού, δεν είναι απλά η υλοποίηση του αλγορίθμου, αλλά βασικό στοιχείο του είναι τα δεδομένα επί των οποίων ενεργεί. Φτάνουμε λοιπόν στο συμπέρασμα ότι αλγόριθμοι και δεδομένα είναι μία αδιάσπαστη ενότητα. Ο προγραμματισμός είναι αυτός που δίνει την εντύπωση ότι, οι υπολογιστές είναι έξυπνες μηχανές που επιλύουν τα πολύπλοκα προβλήματα.

1.3. ΓΛΩΣΣΕΣ ΧΑΜΗΛΟΥ-ΥΨΗΛΟΥ ΕΠΙΠΕΔΟΥ

Συμβολικές γλώσσες ή γλώσσες χαμηλού επιπέδου:

Από τα πρώτα χρόνια άρχισαν να γίνονται προσπάθειες για τη δημιουργία μίας συμβολικής γλώσσας, η οποία ενώ θα έχει έννοια για τον άνθρωπο, θα μετατρέπεται εσωτερικά από τους υπολογιστές στις αντίστοιχες ακολουθίες από 0 και 1. Για παράδειγμα η λέξη ADD (πρόσθεσε) ακολουθούμενη από δύο αριθμούς, είναι κατανοητή από τον άνθρωπο και απομνημονεύεται σχετικά εύκολα. Η εντολή αυτή θα μεταφραστεί από τον υπολογιστή σε μία ακολουθία δυαδικών ψηφίων και στη συνέχεια μπορεί να εκτελεστεί. Το έργο της μετάφρασης το αναλαμβάνει ένα ειδικό πρόγραμμα, ο συμβολομεταφραστής (assembler).

Η χρήση των πρώτων αυτών συμβολικών γλωσσών, που συνεχίζουν να χρησιμοποιούνται για ειδικούς σκοπούς, ήταν σαφώς μια εξέλιξη από τις ακατανόητες ακολουθίες δυαδικών στοιχείων. Ωστόσο παρέμεναν στενά συνδεδεμένες με την αρχιτεκτονική του κάθε υπολογιστή. Επίσης δεν διέθεταν εντολές πιο σύνθετων λειτουργιών οδηγώντας έτσι σε μακροσκελή προγράμματα, που ήταν δύσκολο να γραφούν και κύρια να συντηρηθούν. Ακόμη τα προγράμματα δεν μπορούν να μεταφερθούν σε άλλον διαφορετικό υπολόγιστή, ακόμη και του ίδιου κατασκευαστή. Οι γλωσσες αυτές ονομάζονται γλώσσες χαμηλου επιπέδου, αφού εξαρτώνται από την αρχιτεκτονική του υπολογιστή.

Γλώσσες υψηλού επιπέδου:

Οι παραπάνω ανεπάρκειες των συμβολικών γλωσσών και η προσπάθεια για καλύτερη επικοινωνία ανθρώπου – μηχανής, οδήγησαν στα τέλη της δεκαετίας του 50 στην εμφάνιση των πρώτων γλωσσών υψηλού επιπέδου. Μερικές από αυτές τις γλώσσες είναι η Fortran, η Cobol (COmmon Business Oriented Language), η Algol (ALGOrithmic Language – Αλγοριθμική γλώσσα), η Lisp(LISt Processor – Επεξεργαστής Λίστας), η PASCAL, η Basic (Begginer's All Purpose Symbolic Instruction Code), η JAVA και C.

2. ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C

2.1. ΓΕΝΙΚΑ

Η C δημιουργήθηκε υπό την επίβλεψη των Bell Labs στις αρχές της δεκαετίας του 1970. Είναι μία γλώσσα υψηλού επιπέδου με πολλά από τα πλεονεκτήματα των συμβολικών γλωσσών. Ονομάστηκε έτσι επειδή υπήρξαν παλιότεροι τύποι αυτής της γλώσσας με την ονομασία A και B. Σήμερα, λόγω του ρόλου της στη δημιουργία των πακέτων software και της σχέσης της με το λειτουργικό σύστημα UNIX, είναι μία από τις γλώσσες που προτιμούν οι προγραμματιστές.

Κύρια Χαρακτηριστικά

Τα κύρια χαρακτηριστικά της C σχετίζονται με την δημιουργία των εμπορικών software και τη στενή σχέση της με το UNIX.

- *Ανάπτυξη Πακέτων Software:*

Για αρκετά χρόνια, η διάθεση των πακέτων software έχει γίνει εξαιρετικά ανταγωνιστική. Μόλις ένα προϊόν software – όπως ένας επεξεργαστής κειμένου – δημιουργείται για ένα σύστημα μικροϋπολογιστή, είναι απαραίτητο να δημιουργηθούν, το συντομότερο δυνατό, άλλες εκδόσεις του προϊόντος για τα άλλα δημοφιλή συστήματα μικροϋπολογιστών. Όσο γρηγορότερα ένα προιόν software διατίθεται στην αγορά, τόσο δυσκολότερο είναι να πετύχουν τα ανταγωνιστικά προιόντα. Γι' αυτό οι δημιουργοί πακέτων software χρειάζονται μία γλώσσα υψηλού επιπέδου με δυνατότητα μεταφοράς, όπως η Pascal, και με την αποτελεσματικότητα εκτέλεσης μιας χαμηλού επιπέδου συμβολικής γλώσσας. Η C, που μερικές φορές αναφέρεται ως « φορητή συμβολική γλώσσα », καλύπτει αυτή την ανάγκη. Είναι επίσης δομημένη, γεγονός που επιτρέπει ευκολότερες τροποποιήσεις και ελέγχους και ευκολότερη εκμάθηση σε σχέση με μια συμβολική γλώσσα.

- *Σχέση με το UNIX:*

Το UNIX που είναι ένα δημοφιλές λειτουργικό σύστημα, είναι στο μεγαλύτερο μέρος του γραμμένο σε C. Έτσι καθώς αυξάνεται η ζήτηση του UNIX, μεγαλώνει ανάλογα και η προτίμηση στη C.

Η C χρησιμοποιείται κυρίως από τους ειδικούς σε Η/Υ. Παρ' ότι οι κώδικες που γράφονται σε C μοιάζουν κάπως με εκείνους της Pascal, η γλώσσα C δεν ενδείκνυται για αρχάριους, επειδή είναι πολύπλοκη και πλούσια.

Πρόσφατα δημιουργήθηκε μία νέα έκδοση της C, η λεγόμενη C++, για να βοηθήσει τον καλύτερο σχεδιασμό και τις πρακτικές προγραμματισμού. Η C++ είναι μία αντικειμενικά προσανατολισμένη έκδοση της C. Η C ++ είναι επίσης ένα υπερσύνολο της C, που επιτρέπει την ανάγνωση όλων των προγραμμάτων C από τους μεταγλωττιστές της C++. Επιπλέον, η C++ εξακολουθεί να εξελίσσεται γρήγορα ως γλώσσα.

2.2. ΟΠΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Ο οπτικός προγραμματισμός με C++ Builder είναι ένα εργαλείο προγραμματισμού, το οποίο μοιάζει πολύ στη Visual Basic αλλά και στη Delfi.

Η ανάπτυξη εφαρμογών γίνεται μέσω του ολοκληρωμένου περιβάλλοντος (IDE) που διαθέτει. Βασίζεται στην αντικειμενοστραφή γλώσσα προγραμματισμού C++. Ο C++ Builder είναι ευέλικτος και έχει πολλά πλεονεκτήματα σε όλα τα στάδια ανάπτυξης εφαρμογών, καθώς και στην εγκατάσταση και εκτέλεση των εφαρμογών από τους χρήστες, αφού η εφαρμογή μπορεί να αποτελείται από ένα αυτόνομο εκτελέσιμο αρχείο (.exe), χωρίς να βασίζεται στην ύπαρξη άλλων αρχείων. Με τον C++ Builder μπορούμε να ξαναχρησιμοποιήσουμε τα στοιχεία και τον κώδικα καθώς και να ενσωματώσουμε βιβλιοθήκες. Έτσι ο προγραμματιστής συντομεύει κατά πολύ τη δουλειά του και γίνεται πιο εύχρηστη.

Γενικά, ο οπτικός προγραμματισμός με C++ Builder βασίζεται κυρίως σε φόρμες. Οι φόρμες φιλοξενούν μηχανισμούς – αντικείμενα. Οι πληροφορίες για τα στοιχεία μίας φόρμας καθώς και των μηχανισμών που περιέχει, αποθηκεύονται σε ένα αρχείο φόρμας (.dfm) και σε ένα αρχείο μονάδας (.cpp) με το ίδιο όνομα.

Ο C++ Builder περιλαμβάνει δικούς του μηχανισμούς που αποκαλούνται οπτικά συστατικά (Visual Components) και περιέχονται στη βιβλιοθήκη VCL (Visual Components Library).

Κάθε συστατικό – αντικείμενο του C++ Builder που χρησιμοποιούμε το χειριζόμαστε μέσω των Ιδιοτήτων του (Properties), μέσω των Μεθόδων του (Methods) και μέσω των Συμβάντων του (Events).

3. Η ΓΛΩΣΣΑ BDN

3.1. ΤΟ ΑΛΦΑΒΗΤΟ ΤΗΣ ΓΛΩΣΣΑΣ BDN

Το αλφάβητο της ΓΛΩΣΣΑΣ αποτελείται από τα γράμματα του ελληνικού και του λατινικού αλφαβήτου, τα ψηφία, καθώς και από ειδικά σύμβολα, που χρησιμοποιούνται για προκαθορισμένες ενέργειες.

Συγκεκριμένα

Γράμματα

Κεφαλαία ελληνικού αλφαβήτου (Α-Ω)
Πεζά ελληνικού αλφαβήτου (α-ω)
Κεφαλαία λατινικού αλφαβήτου (A-Z)
Πεζά λατινικού αλφαβήτου (a-z)

Ψηφία

0-9

Ειδικοί χαρακτήρες

+ - * < > [ ] / = ( ) , . κενός χαρακτήρας
Η υποδιαστολή συμβολίζεται με την τελεια.

3.2. ΕΝΤΟΛΕΣ ΠΟΥ ΥΠΟΣΤΗΡΙΖΕΙ Η BDN

Οι εντολές που υποστηρίζονται από την εφαρμογή είναι οι ακόλουθες:
Εντολές εισόδου εξόδου

- ΓΡΑΨΕ
- ΔΙΑΒΑΣΕ

Έντολή εκχώρισης

- ‘ <- ‘

Εντολές επιλογής

- Εντολή ΑΝ
- Εντολή ΕΠΙΛΕΞΕ

Εντολές επανάληψης

- Εντολή ΟΣΟ….ΕΠΑΝΕΛΑΒΕ
- Εντολή ΜΕΧΡΙΣ_ΟΤΟΥ
- Εντολή ΓΙΑ…ΑΠΟ…ΜΕΧΡΙ

Τη σύνταξη, τη λειτουργία καθώς και μερικά παραδείγματα για την κάθε μία θα δούμε παρακάτω.

3.3. ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

Οι υπολογιστές επεξεργάζονται δεδομένα διαφόρων τύπων, γι αυτό είναι σημαντικό να κατανοήσουμε τους διαφορετικούς τύπους δεδομένων που χειρίζεται η ΓΛΩΣΣΑ BDN.

Οι τύποι δεδομένων που υποστηρίζει η BDN είναι οι αριθμητικοί, που περιλαμβάνουν τους ακέραιους και τους πραγματικούς αριθμούς και τέλος οι χαρακτήρες.

Ακέραιος τύπος: Ο τύπος αυτός περιλαμβάνει τους ακέραιους που είναι γνωστοί από τα μαθηματικά. Οι ακέραιοι μπορούν να είναι θετικοί, αρνητικοί ή μηδέν. Παραδείγμα ακεραίων είναι οι αριθμοί 1, 3409, 0, -980.

Πραγματικός τύπος: Ο τύπος αυτός περιλαμβάνει τους πραγματικούς αριθμούς που γνωρίζουμε από τα μαθηματικά. Οι αριθμοί 3.14, 2.71, -112.45 , 0.45 είναι

πραγματικοί αριθμοί. Και οι πραγματικοί αριθμοί μπορούν να είναι θετικοί, αρνητικοί ή μηδέν.

Χαρακτήρας : Ο τύπος αυτός αναφέρεται τόσο σε ένα χαρακτήρα όσο και σε μία σειρά χαρακτήρων. Τα δεδομένα αυτού του τύπου μπορούν να περιέχουν οποιοδήποτε χαρακτήρα παράγεται από το πληκτρολόγιο.

3.4. ΜΕΤΑΒΛΗΤΕΣ

Η έννοια της μεταβλητής (variable) είναι γνωστή από τα μαθηματικά. Για παράδειγμα ο τύπος της γεωμετρίας

$$E=\alpha\beta$$

υπολογίζει το εμβαδόν (Ε) ενός ορθογωνίου με διαστάσεις που συμβολίζονται με α και β. Αν στο α και στο β δοθούν οι αντίστοιχες τιμές, τότε ο τύπος αυτός υπολογίζει το εμβαδόν του ορθογωνίου.

Μια μεταβλητή λοιπόν, παριστάνει μία ποσότητα που η τιμή της μπορεί να μεταβάλλεται.

Οι μεταβλητές που χρησιμοποιούνται σε ένα πρόγραμμα, αντιστοιχούνται από το μεταγλωττιστή σε συγκεκριμένες θέσεις μνήμης του υπολογιστή. Η τιμή της μεταβλητής είναι η τιμή που βρίσκεται στην αντίστοιχη θέση μνήμης και όπως αναφέρθηκε μπορεί να μεταβάλλεται κατά τη διάρκεια της εκτέλεσης του προγράμματος.

Ενώ η τιμή της μεταβλητής μπορεί να αλλάζει κατά την εκτέλεση του προγράμματος, αυτό που μένει υποχρεωτικά αναλλοίωτο είναι ο τύπος της μεταβλητής.

Η ΓΛΩΣΣΑ BDN επιτρέπει τη χρήση μεταβλητών των τριών τύπων που αναφέρθηκαν, δηλαδή ακεραίων, πραγματικών και χαρακτήρων ενώ η δήλωση του τύπου κάθε μεταβλητής γίνεται υποχρεωτικά στο τμήμα δήλωσης μεταβλητών.

Το όνομα κάθε μεταβλητής, ακολουθεί τους κανόνες δημιουργίας ονομάτων, δηλαδή αποτελείται από ελληνικούς χαρακτήρες (πεζούς και κεφαλαίους), ψηφία καθώς και τον χαρακτήρα _ ενώ το όνομα κάθε μεταβλητής είναι μοναδικό για κάθε πρόγραμμα.

Παραδείγματα ονομάτων που είναι αποδεκτά από τη BDN είναι: α, ονομα, τιμη, τυπικη_αποκλιση, α100, ΜΕΓΙΣΤΟ.

Παραδείγματα ονομάτων που δεν είναι αποδεκτά είναι: 100α , Μεση τιμη, κοστος$, onoma.

Σύνταξη
ΜΕΤΑΒΛΗΤΕΣ
τύπος-1: Λίστα-μεταβλητών-1
τύπος-2: Λίστα-μεταβλητών-2
..
τύπος-ν: Λίστα-μεταβλητών-ν
Παράδειγμα
ΜΕΤΑΒΛΗΤΕΣ
ΠΡΑΓΜΑΤΙΚΟΙ: εμβαδον, πλατος
ΑΚΕΡΑΙΟΙ: τιμη, ποσοτητα
ΧΑΡΑΚΤΗΡΕΣ: ονομα[10]
Λειτουργία
Δηλώνει τον τύπο όλων των μεταβλητών που χρησιμοποιούνται στο πρόγραμμα.

Αν και όπως αναφέρθηκε, το όνομα των μεταβλητών μπορεί να είναι οποιοσδήποτε συνδυασμός χαρακτήρων, είναι καλή πρακτική να χρησιμοποιούνται ονόματα, τα οποία να υπονοούν το περιεχόμενο τους, κάνοντας το πρόγραμμα ευκολότερο στην ανάγνωση του και στην κατανόηση του.

Για παράδειγμα στην περίπτωση του υπολογισμού του εμβαδού είναι προτιμότερη η χρήση του ονόματος *εμβαδο* για την αντίστοιχη μεταβλητή, από ένα όνομα που αποτελείται από ένα μόνο γράμμα όπως *ε* ή *α* ή ένα οποιοδήποτε τυχαίο όνομα που δεν ανάγει στο πραγματικό περιεχόμενο της μεταβλητής όπως τιμή.

3.5. ΠΙΝΑΚΕΣ

Μπορούμε να ορίσουμε τον πίνακα ως μια δομή που περιέχει στοιχεία του ίδιου τύπου (δηλαδή ακέραιους, πραγματικούς κ.λ.π). Η δήλωση των στοιχείων ενός πίνακα και η μέθοδος αναφοράς τους εξαρτάται από τη συγκεκριμένη γλώσσα υψηλού επιπέδου που χρησιμοποιείται. Όμως, γενικά η αναφορά στα στοιχεία ενός πίνακα γίνεται με τη χρήση του συμβολικού ονόματος του πίνακα ακολουθούμενου από την τιμή ενός δείκτη σε αγκύλη.

Ένας πίνακας μπορεί να είναι μονοδιάστατος, δισδιάστατος, τρισδιάστατος και γενικά ν-διάστατος πίνακας. Εμείς θα ασχοληθούμε με τους μονοδιάστατους πίνακες.

Για παράδειγμα ένα πρόγραμμα το οποίο διαβάζει τις θερμοκρασίες διαφόρων ημερών του μήνα, έστω 30, και υπολογίζει τη μέση θερμοκρασία, μπορεί πολύ απλά να γραφεί ως εξής:

```
……..
συνολο <- 0
ΓΙΑ ημερα ΑΠΟ 1 ΜΕΧΡΙ 30 ΜΕ ΒΗΜΑ 1
   ΔΙΑΒΑΣΕ θερμοκρασια
   Συνολο <- συνολο + θερμοκρασια
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
Μεση <- συνολο / 30
……...
```

Χρησιμοποιώντας λοιπόν μόνο μία μεταβλητή, τη μεταβλητή θερμοκρασία, το πρόβλημα λύνεται πολύ απλά και το αντίστοιχο πρόγραμμα είναι σύντομο και κατανοητό.

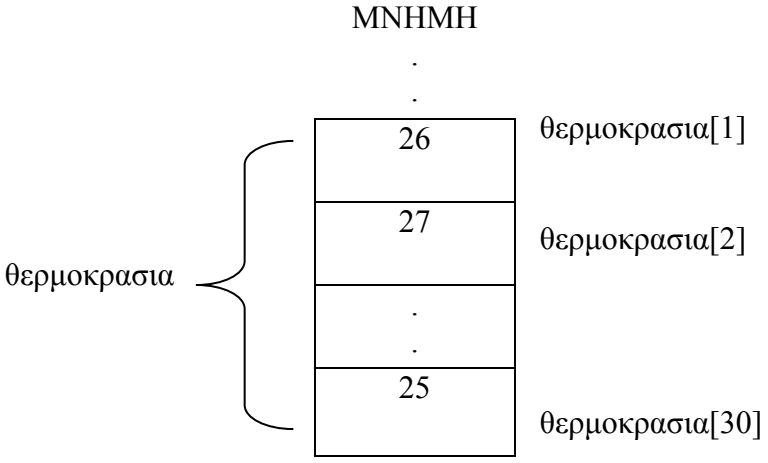
Αν όμως στο προηγούμενο πρόγραμμα ζητείται και ο αριθμός των ημερών που η θερμοκρασία ήταν κατώτερη της μέσης, τότε η σύγκριση αυτή πρέπει να γίνει μετά τον υπολογισμό της μέσης θερμοκρασίας. Αυτό σημαίνει ότι όλες οι θερμοκρασίες πρέπει να επαναεισαχθούν για να συγκριθούν με τη μέση.

Μία άλλη λύση είναι να καταχωρηθεί κάθε θερμοκρασία σε διαφορετική μεταβλητή, έτσι ώστε κάθε τιμή που εισάγεται να διατηρείται στη μνήμη και να μπορεί να συγκριθεί με τη μέση, αφού αυτή υπολογιστεί. Τότε όμως πρέπει να δημιουργηθούν 30 διαφορετικές μεταβλητές θερμοκρασια1, θερμοκρασια2,…. Θερμοκρασια30. Για να γραφτεί το πρόγραμμα χρειάζονται τριάντα εντολές ΔΙΑΒΑΣΕ και τριάντα εντολές ΑΝ.

Αν και αυτή η λύση είναι σωστή και πρακτική για μικρό αριθμό δεδομένων, προφανώς δεν εξυπηρετεί την επεξεργασία μεγάλου αριθμού δεδομένων.

Η καλύτερη λύση στο πρόβλημα αυτό είναι η χρήση μεταβλητής με δείκτες, και υλοποιείται στον προγραμματισμό με τη δομή δεδομένων του πίνακα. Χρησιμοποιείται λοιπόν μόνο ένα όνομα θερμοκρασία, που αναφέρεται στις τριάντα διαφορετικές θερμοκρασίες.

Το όνομα του πίνακα καθορίζει μία ομάδα διαδοχικών θέσεων στη μνήμη. Κάθε συγκεκριμένη θέση μνήμης καλείται στοιχείο του πίνακα και προσδιορίζεται από την τιμή ενός δείκτη, όπως φαίνεται και στο σχήμα.



Οι πίνακες που χρησιμοποιούν ένα μόνο δείκτη για την αναφορά των στοιχείων τους, ονομάζονται μονοδιάστατοι πίνακες.

Το όνομα του πίνακα μπορεί να είναι οποιοδήποτε δεκτό όνομα της ΓΛΩΣΣΑΣ και ο δείκτης είναι μία ακέραια έκφραση, μεταβλητή που περικλείεται μέσα στα σύμβολα [και]. Το στοιχείο θερμοκρασια[2], εκφράζει τη θερμοκρασία της δεύτερης ημέρας, αναφέρεται στο δεύτερο στοιχείο του πίνακα θερμοκρασία και έχει την τιμή 27.

Γενικότερα το στοιχείο θερμοκρασία[ι] αναφέρεται στο ι-οστό στοιχείο του πίνακα.

Κάθε πίνακας πρέπει υποχρεωτικά να περιέχει δεδομένα του ιδίου τύπου, δηλαδή ακέραια, πραγματικά η αλφαριθμητικά. Ο τύπος του πίνακα δηλώνεται μαζί με τις άλλες μεταβλητές του προγράμματος στο τμήμα δήλωσης μεταβλητών. Εκτός από τον τύπο του πίνακα πρέπει να δηλώνεται και ο αριθμός των στοιχείων που περιέχει ή καλύτερα ο μεγαλύτερος αριθμός στοιχείων που μπορεί να έχει ο συγκεκριμένος πίνακας.

Για παράδειγμα:

ΜΕΤΑΒΛΗΤΕΣ

ΠΡΑΓΜΑΤΙΚΕΣ: θερμοκρασια[30]

Πίνακας είναι ένα σύνολο αντικειμένων ίδιου τύπου, τα οποία αναφέρονται με ένα κοινό όνομα. Κάθε ένα από τα αντικείμενα που απαρτίζουν τον πίνακα λέγεται στοιχείο του πίνακα. Η αναφορά σε ατομικά στοιχεία του πίνακα γίνεται με το όνομα του πίνακα ακολουθούμενο από ένα δείκτη.

Παράδειγμα

Χρησιμοποιώντας μεταβλητές με δείκτες για το προηγούμενο παράδειγμα έχουμε το εξής πρόγραμμα.

```
ΑΡΧΗ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΜΕΤΑΒΛΗΤΕΣ
      ΑΚΕΡΑΙΟΙ: ι, ημερες
      ΠΡΑΓΜΑΤΙΚΟΙ: θερμοκρασια[30], μεση, συνολο

συνολο <- 0
ΓΙΑ ι ΑΠΟ 1 ΜΕΧΡΙ 30 ΜΕ ΒΗΜΑ 1

      ΓΡΑΨΕ ' δωσε τη θερμοκρασια'
      ΔΙΑΒΑΣΕ θερμοκρασια[ι]
      συνολο <- συνολο + θερμοκρασια[ι]

ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
μεση <- συνολο / 30
ημερες <- 0

ΓΙΑ ι ΑΠΟ 1 ΜΕΧΡΙ 30 ΜΕ ΒΗΜΑ 1
   ΑΝ θερμοκρασια[ι] < μεση ΤΟΤΕ
         ημερες <- ημερες + 1
  ΤΕΛΟΣ_ΑΝ

ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΓΡΑΨΕ ' μεση θερμοκρασια:', μεση
ΓΡΑΨΕ 'ημερες με μικροτερη θερμοκρασια', ημερες

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

3.6. ΑΡΙΘΜΗΤΙΚΟΙ ΤΕΛΕΣΤΕΣ

Οι αριθμητικοί τελεστές που υποστηρίζονται από τη ΓΛΩΣΣΑ BDN καλύπτουν τις βασικές πράξεις: πρόσθεση, αφαίρεση, πολλαπλασιασμό και διαίρεση ενώ υποστηρίζεται και η ύψωση σε δύναμη.
Οι τελεστές και οι αντίστοιχες πράξεις είναι:

Αριθμητικός τελεστής	*Πράξη*
+	Πρόσθεση
-	*Αφαίρεση*
*	*Πολλαπλασιασμός*
/	*Διαίρεση*

3.7. ΑΡΙΘΜΗΤΙΚΕΣ ΕΚΦΡΑΣΕΙΣ

Όταν μια τιμή προκύπτει από υπολογισμό, τότε αναφερόμαστε σε εκφράσεις (expressions). Για την σύνταξη μιας αριθμητικής έκφρασης χρησιμοποιούνται αριθμητικές σταθερές (αριθμοί), μεταβλητές, αριθμητικοί τελεστές και παρενθέσεις. Οι αριθμητικές εκφράσεις υλοποιούν απλές ή σύνθετες μαθηματικές πράξεις. Κάθε έκφραση παριστάνει μια συγκεκριμένη αριθμητική τιμή, η οποία βρίσκεται μετά την εκτέλεση των πράξεων. Γι' αυτό είναι καλό όλες οι μεταβλητές, που εμφανίζονται σε μία έκφραση να έχουν οριστεί προηγούμενα, δηλαδή να έχουν κάποια τιμή.

Οι πράξεις που παρουσιάζονται σε μία έκφραση, εκτελούνται σύμφωνα με την επόμενη ιεραρχία

1. Πολλαπλασιασμός και διαίρεση
2. πρόσθεση και αφαίρεση

Παραδείγματα

Μαθηματικά	ΓΛΩΣΣΑ
α+1	α+1
½ α	½*α

Όταν η ιεραρχία είναι ίδια, τότε οι πράξεις εκτελούνται από τ' αριστερά προς τα δεξιά. Σε πολλές όμως περιπτώσεις είναι απαραίτητο να προηγηθεί μια πράξη χαμηλότερης ιεραρχίας. Αυτό επιτυγχάνεται με την εισαγωγή των παρενθέσεων. Η πράξη που πρέπει να προηγηθεί περικλείεται σε ένα ζεύγος παρενθέσεων, οπότε και εκτελείται πρώτη. Π.χ. η έκφραση 2+3*4 δίδει ως αποτέλεσμα 14, ενώ η (2+3)*4 δίδει 20, διότι εκτελείται πρώτα η πρόσθεση και μετά ο πολλαπλασιασμός.

3.8. ΕΝΤΟΛΗ ΕΚΧΩΡΗΣΗΣ

Η εντολή εκχώρησης χρησιμοποιείται για την απόδοση τιμών στις μεταβλητές κατά τη διάρκεια εκτέλεσης του προγράμματος.

Σύνταξη

Όνομα-Μεταβλητής <-έκφραση

Παράδειγμα

```
Α <- 132
εμβαδον <- Α + Β
```

Λειτουργία

Υπολογίζεται η τιμή της έκφρασης στη δεξιά πλευρά και εκχωρείται η τιμή αυτή στη μεταβλητή, που αναφέρεται στην αριστερή μεριά.

Μια εντολή εκχώρησης σε καμιά περίπτωση δεν πρέπει να εκλαμβάνεται ως εξίσωση. Στην εξίσωση το αριστερό μέλος ισούται με το δεξιό, ενώ στην εντολή εκχώρησης η τιμή του δεξιού μέλους εκχωρείται, μεταβιβάζεται, αποδίδεται στη μεταβλητή του αριστερού μέλους. Για το λόγο αυτό ως τελεστής εκχώρησης χρησιμοποιείται το σύμβολο <- προκειμένου να διαφοροποιείται από το ίσον (=). Ωστόσο, ας σημειωθεί, ότι οι διάφορες γλώσσες προγραμματισμού χρησιμοποιούν διαφορετικά σύμβολα για το σκοπό αυτό.

3.9. ΕΝΤΟΛΕΣ ΕΙΣΟΔΟΥ ΕΞΟΔΟΥ

Σχεδόν όλα τα προγράμματα υπολογιστή δέχονται κάποια δεδομένα, τα επεξεργάζονται, υπολογίζουν τα αποτελέσματα και τέλος τα εμφανίζουν.

Τα δεδομένα εισάγονται κατά τη διάρκεια της εκτέλεσης του προγράμματος από μια μονάδα εισόδου, για παράδειγμα το πληκτρολόγιο και τα αποτελέσματα γράφονται σε μια μονάδα εξόδου, για παράδειγμα την οθόνη.

Η ΓΛΩΣΣΑ υποστηρίζει για την εισαγωγή δεδομένων από το πληκτρολόγιο την εντολή ΔΙΑΒΑΣΕ και για την εμφάνιση των αποτελεσμάτων την εντολή ΓΡΑΨΕ.

3.9.1. Η ΕΝΤΟΛΗ ΔΙΑΒΑΣΕ

Σύνταξη

ΔΙΑΒΑΣΕ λίστα-μεταβλητών

Παράδειγμα
ΔΙΑΒΑΣΕ ποσοτητα
    ΔΙΑΒΑΣΕ τιμη

Λειτουργία

Η εκτέλεση της εντολής οδηγεί στην είσοδο τιμής από το πληκτρολόγιο και την εκχώρησή της στη μεταβλητή που αναφέρεται.

Η εντολή ΔΙΑΒΑΣΕ ακολουθείται πάντοτε από ένα μόνο όνομα μεταβλητής. Κατά την εκτέλεση του προγράμματος η εντολή ΔΙΑΒΑΣΕ διακόπτει την εκτέλεσή του, και το πρόγραμμα περιμένει την εισαγωγή τιμής από το πληκτρολόγιο, που θα εκχωρηθεί στη μεταβλητή. Μετά την ολοκλήρωση της εντολής η εκτέλεση του προγράμματος συνεχίζεται με την επόμενη εντολή.

3.9.2. Η ΕΝΤΟΛΗ ΓΡΑΨΕ

Σύνταξη

ΓΡΑΨΕ λίστα-στοιχείων

Παράδειγμα
ΓΡΑΨΕ 'Η τετραγωνικη ριζα του', α, 'ειναι : ', ριζα

Λειτουργία
Χρησιμοποιείται για την εμφάνιση σταθερών τιμών καθώς και των τιμών των μεταβλητών που αναφέρονται στη λίστα .

Η εντολή ΓΡΑΨΕ έχει ως αποτέλεσμα την εμφάνιση των τιμών στη μονάδα εξόδου. Συσκευή εξόδου μπορεί να είναι η οθόνη του υπολογιστή, ο εκτυπωτής, βοηθητική μνήμη ή και γενικά οποιαδήποτε συσκευή εξόδου έχει οριστεί στο πρόγραμμα.

Η χρήση της εντολής ΓΡΑΨΕ είναι κυρίως η εμφάνιση μηνυμάτων από τον υπολογιστή στην οθόνη, καθώς και αποτελεσμάτων που περιέχονται στις μεταβλητές.

2.11. ΔΟΜΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

Όπως κάθε εντολή ακολουθεί αυστηρούς συντακτικούς κανόνες, έτσι και ολόκληρο το πρόγραμμα έχει αυστηρούς κανόνες για τον τρόπο που δομείται. Η πρώτη εντολή κάθε προγράμματος πρέπει να είναι η ΑΡΧΗ_ΠΡΟΓΡΑΜΜΑΤΟΣ. Στη συνέχεια ακολουθεί το τμήμα δήλωσης μεταβλητών, όπου δηλώνονται υποχρεωτικά τα ονόματα όλων των μεταβλητών καθώς και ο τύπος τους. Αυτό γίνεται με την εντολή ΜΕΤΑΒΛΗΤΕΣ και εν συνεχεία ακολουθεί ο τύπος των μεταβλητών που μπορεί να είναι είτε ΑΚΕΡΑΙΟΙ:, είτε ΠΡΑΓΜΑΤΙΚΟΙ: είτε ΧΑΡΑΚΤΗΡΕΣ:. Ακολουθεί το κύριο μέρος του προγράμματος, που περιλαμβάνει όλες τις εκτελέσιμες εντολές. Τέλος κάθε πρόγραμμα πρέπει να κλείνει με την εντολή ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ. Όλες οι εκτελέσιμες εντολές πρέπει να περιλαμβάνονται υποχρεωτικά ανάμεσα στις ΑΡΧΗ_ΠΡΟΓΡΑΜΜΑΤΟΣ και ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ. Τέλος να σημειώσουμε ότι κάθε εντολή γράφεται σε ξεχωριστή γραμμή. Στο παρακάτω παράδειγμα μπορούμε να δούμε τη Δομή που πρέπει να έχει το πρόγραμμα.

Παραδειγμα

Το επόμενο πρόγραμμα υπολογίζει το συνολικό κόστος παραγγελιών υπολογιστών. Το πρόγραμμα διαβάζει από το πληκτρολόγιο την ποσότητα της παραγγελίας και την τιμή του ενός υπολογιστή, υπολογίζει και γράφει το συνολικό κόστος καθώς και το αντίστοιχο κόστος του ΦΠΑ. Ο συντελεστής ΦΠΑ είναι 18% = 0,18.

ΑΡΧΗ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΜΕΤΑΒΛΗΤΕΣ

```
        ΑΚΕΡΑΙΟΙ:  ποσοτητα, τιμη_μοναδας, κοστος
        ΠΡΑΓΜΑΤΙΚΟΙ: φπα, αξια_φπα, συνολικο_κοστος

ΓΡΑΨΕ 'Δωσε την ποσοτητα της παραγγελιας '
ΔΙΑΒΑΣΕ ποσοτητα
ΓΡΑΨΕ 'Δωσε την τιμη του υπολογιστη'
ΔΙΑΒΑΣΕ τιμη_μοναδας

φπα <- 0.19
κοστος <- ποσοτητα * τιμη_μοναδας
αξια_φπα <- κοστος * φπα
συνολικο_κοστος <- κοστος + αξια_φπα

ΓΡΑΨΕ 'Το κοστος των', ποσοτητα, 'υπολογ. είναι ', κοστος
ΓΡΑΨΕ 'Το συνολικο κοστος ειναι ', συνολικο_κοστος

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

4. ΕΠΙΛΟΓΗ ΚΑΙ ΕΠΑΝΑΛΗΨΗ

4.1. ΕΝΤΟΛΕΣ ΕΠΙΛΟΓΗΣ

Μία από τις βασικότερες δομές που εμφανίζονται σε ένα πρόγραμμα, είναι η επιλογή. Σχεδόν σε όλα τα προβλήματα περιλαμβάνονται κάποιοι έλεγχοι και ανάλογα με το αποτέλεσμα αυτών των ελέγχων επιλέγονται οι ενέργειες που θα ακολουθήσουν.

Ας θεωρήσουμε το πολύ απλό πρόβλημα της καταμέτρησης των θετικών και των αρνητικών αριθμών. Πρέπει λοιπόν να γράψουμε ένα πρόγραμμα, το οποίο εισάγει αριθμούς και μετράει πόσοι από αυτούς είναι θετικοί και πόσοι αρνητικοί. Για να αποφασίσουμε αν ένας αριθμός είναι θετικός ή αρνητικός, πρέπει να τον συγκρίνουμε με το 0. το αποτέλεσμα αυτής της σύγκρισης καθορίζει το είδος του αριθμού, αν είναι μεγαλύτερος από το 0, τότε ο αριθμός είναι θετικός, ενώ αντίθετα αν είναι μικρότερος από το 0, είναι αρνητικός.

Λογική έκφραση

Για την σύνταξη μιας λογικής έκφρασης ή συνθήκης χρησιμοποιούνται μεταβλητές, αριθμητικές παραστάσεις, συγκριτικοί και λογικοί τελεστές, καθώς και παρενθέσεις. Στις λογικές εκφράσεις γίνεται σύγκριση της τιμής μιας έκφρασης, που βρίσκεται αριστερά από το συγκριτικό τελεστή με την τιμή μιας άλλης έκφρασης που βρίσκεται δεξιά. Το αποτέλεσμα είναι μια λογική τιμή ΑΛΗΘΗΣ ή ΨΕΥΔΗΣ.

Οι χρησιμοποιούμενοι τελεστές παρουσιάζονται στον επόμενο πίνακα.

ΣΥΓΚΡΙΤΙΚΟΙ ΤΕΛΕΣΤΕΣ		
Τελεστής	Ελεγχόμενη σχέση	Παράδειγμα
=	Ισότητα	Αριθμός=0
<>	Ανισότητα	Τιμή<>2
>	Μεγαλύτερο από	Τιμή>10000
>=	Μεγαλύτερο ή ίσο	Χ+Υ>=(Α+Β)/Γ
<	Μικρότερο από	(Α+Β)*Γ<0
<=	Μικρότερο ή ίσο	Βαρος<=500

Σύνθετες εκφράσεις

Σε πολλά προβλήματα οι επιλογές δεν αρκεί να γίνονται με απλές λογικές παραστάσεις όπως αυτές οι οποίες αναφέρθηκαν, αλλά χρειάζεται να συνδυαστούν μία ή περισσότερες λογικές παραστάσεις. Αυτό επιτυγχάνεται με τη χρήση των τριών βασικών λογικών τελεστών ΟΧΙ, ΚΑΙ, Η. Ο παρακάτω πίνακας δίνει τις τιμές των τριών αυτών λογικών πράξεων για όλους τους συνδυασμούς τιμών.

Πρόταση Α	**Πρόταση Β**	**Α ή Β**	**Α και Β**	**όχι Α**
Αληθής	Αληθής	Αληθής	Αληθής	Ψευδής
Αληθής	Ψευδής	Αληθής	Ψευδής	Ψευδής
Ψευδής	Αληθής	Αληθής	Ψευδής	Αληθής
Ψευδής	Ψευδής	Ψευδής	Ψευδής	Αληθής

Παραδείγματα

0<Χ<5 Χ > 0 ΚΑΙ Χ < 5
Χ=1 ή 2ή 3 Χ = 1 Η Χ = 2 Η Χ = 3
Η ιεραρχία των λογικών τελεστών είναι μικρότερη των αριθμητικών.

4.1.1. ΕΝΤΟΛΗ ΑΝ

Η δομή επιλογής υλοποιείται στη ΓΛΩΣΣΑ BDN με την εντολή ΑΝ. Η εντολή ΑΝ εμφανίζεται με τρεις διαφορετικές μορφές. Την απλή εντολή ΑΝ…ΤΟΤΕ, την εντολή ΑΝ…ΤΟΤΕ…ΑΛΛΙΩΣ και τέλος την εντολή ΑΝ…ΤΟΤΕ…ΑΛΛΙΩΣ _ΑΝ. Κάθε εντολή ΑΝ πρέπει να κλείνει με ΤΕΛΟΣ_ΑΝ.
Στην απλούστερη μορφή της η εντολή ΑΝ ελέγχει τη συνθήκη και αν αυτή ισχύει (είναι αληθής), τότε εκτελούνται οι εντολές που περιλαμβάνονται μεταξύ των λέξεων ΤΟΤΕ και ΤΕΛΟΣ_ΑΝ.

Αν για παράδειγμα θέλαμε να υπολογίσουμε το πηλίκο του αριθμού 10 με τους αριθμούς που διαβάζουμε από το πληκτρολόγιο, τότε το αντίστοιχο τμήμα προγράμματος είναι

```
ΔΙΑΒΑΣΕ α

ΑΝ α<>0 ΤΟΤΕ
        πηλικο <- 10 / α
ΤΕΛΟΣ_ΑΝ
```

Η γενική μορφή της ΑΝ είναι η εξής:

Σύνταξη
ΑΝ συνθήκη ΤΟΤΕ
    εντολή-1
    εντολή-2
    …
    εντολή-ν
ΤΕΛΟΣ_ΑΝ
Παράδειγμα
ΑΝ αριθμός > 0 ΤΟΤΕ
ΓΡΑΨΕ 'Ο αριθμός είναι θετικός'
    πλήθος_θετικων <- πληθος_θετικων + 1
ΤΕΛΟΣ_ΑΝ
Λειτουργία
    Αν η συνθήκη ισχύει, τότε εκτελούνται οι εντολές που βρίσκονται μεταξύ των λέξεων ΤΟΤΕ και ΤΕΛΟΣ_ΑΝ, σε αντίθετη περίπτωση οι εντολές αυτές αγνοούνται. Η εκτέλεση του προγράμματος συνεχίζεται με την εντολή που ακολουθεί τη δήλωση ΤΕΛΟΣ_ΑΝ

Συχνά η εντολή ΑΝ εκτός από το τμήμα των εντολών, που εκτελούνται όταν η λογική έκφραση είναι αληθής, περιέχει και το τμήμα των εντολών που εκτελούνται, αν δεν ισχύει η συνθήκη (είναι ψευδής).
Η μορφή της εντολής αυτής ονομάζεται ΑΝ…ΤΟΤΕ…ΑΛΛΙΩΣ.
Στο παράδειγμα υπολογισμού του πηλίκου έχουμε:

```
ΔΙΑΒΑΣΕ α
ΑΝ α<>0 ΤΟΤΕ
    πηλικο = 10 / α
ΑΛΛΙΩΣ
    ΓΡΑΨΕ 'Δεν γινεται διαιρεση με το 0'
ΤΕΛΟΣ_ΑΝ
```

Η γενική μορφή της εντολής ΑΝ…ΤΟΤΕ…ΑΛΛΙΩΣ έχει ως εξής:

Σύνταξη
ΑΝ συνθήκη ΤΟΤΕ
    εντολή-1
    εντολή-2
    …
    εντολή-ν
ΑΛΛΙΩΣ
    εντολή-1
    εντολή-2
    …
    εντολή-ν
ΤΕΛΟΣ_ΑΝ

Παράδειγμα

```
ΑΝ αριθμος > 0 ΤΟΤΕ
ΓΡΑΨΕ 'Ο αριθμος ειναι θετικος'
        πληθος_θετικων <- πληθος_θετικων + 1
ΑΛΛΙΩΣ
        ΓΡΑΨΕ 'Ο αριθμος ειναι αρνητικος'
        πληθος_μη_θετικων <- πληθος_μη_θετικων + 1
ΤΕΛΟΣ_ΑΝ
```

Λειτουργία

Αν η συνθήκη ισχύει, τότε εκτελούνται οι εντολές μεταξύ των λέξεων ΤΟΤΕ και ΑΛΛΙΩΣ, διαφορετικά εκτελούνται οι εντολές μεταξύ ΑΛΛΙΩΣ και ΤΕΛΟΣ_ΑΝ.

Η εκτέλεση του προγράμματος συνεχίζεται με την εντολή που ακολουθεί τη δήλωση ΤΕΛΟΣ_ΑΝ.

Η γενική μορφή της ΑΝ καλύπτει την επιλογή μιας από δυο εναλλακτικές περιπτώσεις.
Όταν οι εναλλακτικές περιπτώσεις είναι περισσότερες από τις δυο, τότε μπορούν να χρησιμοποιηθούν πολλές εντολές ΑΝ η μία μέσα στην άλλη, οι εμφωλευμένες όπως ονομάζονται.

Εμφωλευμένα ΑΝ ονομάζονται δυο ή περισσότερες εντολές μορφής
ΑΝ…ΤΟΤΕ…ΑΛΛΙΩΣ που περιέχονται η μία μέσα στην άλλη.
Για παράδειγμα οι παρακάτω εντολές προγράμματος

```
.....
ΔΙΑΒΑΣΕ βαρος, υψος
ΑΝ βαρος < 80 ΤΟΤΕ
        ΑΝ υψος < 1.70 ΤΟΤΕ
                ΓΡΑΨΕ 'ελαφρυς, κοντος'
        ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΑΝ
.....
```

Η χρήση εμφωλευμένων εντολών ΑΝ οδηγεί συνήθως σε πολύπλοκες δομές που αυξάνουν την πιθανότητα του λάθους καθώς και τη δυσκολία κατανόησης του προγράμματος.
Πολύ συχνά οι εντολές που έχουν γραφεί με εμφωλευμένα ΑΝ, μπορούν να γραφούν πιο απλά χρησιμοποιώντας σύνθετες εκφράσεις ή την εντολή επιλογής ΑΝ_ΑΛΛΙΩΣ_ΑΝ, που θα παρουσιαστεί στη συνέχεια.
Το προηγούμενο τμήμα προγράμματος μπορεί να γραφεί ως εξής:

```
ΔΙΑΒΑΣΕ βαρος, υψος
ΑΝ βαρος < 80 ΚΑΙ υψος < 1.70 ΤΟΤΕ
ΓΡΑΨΕ 'Ελαφρυς, κοντος'
ΤΕΛΟΣ_ΑΝ
```

Μία άλλη μορφή επιλογής είναι η εντολή ΑΝ…ΤΟΤΕ…ΑΛΛΙΩΣ_ΑΝ

Σύνταξη
ΑΝ συνθήκη-1 ΤΟΤΕ
    εντολή-1
    εντολή-2
    …
    εντολή-ν
ΑΛΛΙΩΣ_ΑΝ συνθήκη-2 ΤΟΤΕ
    εντολή-1
    εντολή-2
    …
    εντολή-ν
ΑΛΛΙΩΣ
    εντολή-1
    εντολή-2
    …
    εντολή-ν
ΤΕΛΟΣ_ΑΝ
Παράδειγμα
ΑΝ αριθμος > 0 ΤΟΤΕ
ΓΡΑΨΕ 'Ο αριθμος ειναι θετικος'
    πληθος_θετικων <- πληθος_θετικων + 1
ΑΛΛΙΩΣ_ΑΝ αριθμος < 0 ΤΟΤΕ
    ΓΡΑΨΕ 'Ο αριθμος ειναι αρνητικος'
    πληθος_μη_θετικων <- πληθος_μη_θετικων + 1
ΑΛΛΙΩΣ
    ΓΡΑΨΕ 'Ο αριθμος είναι 0'
    πληθος_0 <- πληθος_0 + 1
ΤΕΛΟΣ_ΑΝ
Λειτουργία
Εκτελούνται οι εντολές που βρίσκονται στο αντίστοιχο τμήμα, όταν η συνθήκη είναι αληθής
Η εκτέλεση του προγράμματος συνεχίζεται με την εντολή που ακολουθεί τη δήλωση ΤΕΛΟΣ_ΑΝ

Παράδειγμα

Στο προηγούμενο πρόγραμμα (πωλήσεις υπολογιστών) υποθέτουμε ότι η τιμή των υπολογιστών εξαρτάται από την ποσότητα παραγγελίας. Συγκεκριμένα ισχύουν οι παρακάτω τιμές αγοράς υπολογιστών.

ΠΟΣΟΤΗΤΑ	ΤΙΜΗ ΜΟΝΑΔΑΣ
1-50	580
51-100	520
101-200	470
Πάνω από 200	440

Ο υπολογισμός με τη χρήση της εντολής ΑΝ…ΤΟΤΕ…ΑΛΛΙΩΣ_ΑΝ έχει ως εξής:

```
ΑΡΧΗ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΜΕΤΑΒΛΗΤΕΣ
      ΑΚΕΡΑΙΟΙ: ποσοτητα, κοστος

ΓΡΑΨΕ 'δοσε ποσοτητα'
ΔΙΑΒΑΣΕ ποσοτητα
ΑΝ ποσοτητα <= 50 ΤΟΤΕ
        κοστος <- ποσοτητα * 580
ΑΛΛΙΩΣ_ΑΝ ποσοτητα <= 100 ΤΟΤΕ
        κοστος <- ποσοτητα * 520
ΑΛΛΙΩΣ_ΑΝ ποσοτητα <= 200 ΤΟΤΕ
                κοστος <- ποσοτητα * 470
ΑΛΛΙΩΣ
                κοστος <- ποσοτητα * 440
ΤΕΛΟΣ_ΑΝ
ΓΡΑΨΕ 'η ποσοτητα ειναι', ποσοτητα, ' και το κοστος', κοστος
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Ένα συχνό λάθος που παρατηρείται στα προγράμματα είναι ο έλεγχος περιττών συνθηκών. Οι επιπλέον έλεγχοι αυξάνουν την πολυπλοκότητα του προγράμματος.

Στο προηγούμενο παράδειγμα για το οποίο θεωρούμε ότι η ποσότητα είναι θετικός αριθμός, ένα παράδειγμα περιττών ελέγχων είναι το ακόλουθο:

```
ΑΝ ποσοτητα <= 50 ΤΟΤΕ
        κοστος <- ποσοτητα * 580
ΑΛΛΙΩΣ_ΑΝ ποσοτητα > 50 ΚΑΙ ποσοτητα <= 100 ΤΟΤΕ
        κοστος <- ποσοτητα * 520
ΑΛΛΙΩΣ_ΑΝ ποσοτητα > 100 ΚΑΙ ποσοτητα <= 200 ΤΟΤΕ
        κοστος <- ποσοτητα * 470
ΑΛΛΙΩΣ
        κοστος <- ποσοτητα * 470
ΤΕΛΟΣ_ΑΝ
```

Εντολή ΕΠΙΛΕΞΕ

Αν οι εναλλακτικές περιπτώσεις επιλογής είναι πολλές, μπορεί να χρησιμοποιηθεί η εντολή ΕΠΙΛΕΞΕ, η γενική μορφή της οποίας είναι:

Σύνταξη

```
ΕΠΙΛΕΞΕ έκφραση
        ΠΕΡΙΠΤΩΣΗ  λίστα_τιμών_1
                εντολές_1
        ΠΕΡΙΠΤΩΣΗ  λίστα_τιμών_2
                εντολές_2
        …….
        ΠΕΡΙΠΤΩΣΗ ΑΛΛΙΩΣ
                εντολές_αλλιώς
ΤΕΛΟΣ  ΕΠΙΛΟΓΩΝ
```

Παράδειγμα
ΔΙΑΒΑΣΕ αριθμος
ΕΠΙΛΕΞΕ αριθμος
    ΠΕΡΙΠΤΩΣΗ 0
        ΓΡΑΨΕ 'μηδεν'
    ΠΕΡΙΠΤΩΣΗ 1, 3, 5, 7, 9
        ΓΡΑΨΕ 'μονος αριθμος'
    ΠΕΡΙΠΤΩΣΗ 2, 4, 6, 8
        ΓΡΑΨΕ 'ζυγος'
    ΠΕΡΙΠΤΩΣΗ ΑΛΛΙΩΣ
        ΓΡΑΨΕ 'αριθμος < 0 η > 9 η οχι ακεραιος'
ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ

Λειτουργία
Υπολογίζεται η τιμή της έκφρασης και εκτελούνται οι εντολές που ανήκουν στην αντίστοιχη περίπτωση τιμών. Αν η τιμή της έκφρασης δεν αντιστοιχεί σε καμία περίπτωση, τότε εκτελούνται οι εντολές αλλιώς.

Στην εντολή αυτή οι λίστες τιμών που συνοδεύουν κάθε περίπτωση μπορούν να περιλάβουν μία ή περισσότερες τιμές.

4.2. ΕΝΤΟΛΕΣ ΕΠΑΝΑΛΗΨΗΣ

Η τρίτη βασική δομή είναι η δομή επανάληψης, ο βρόχος, η οποία επιτρέπει την εκτέλεση εντολών περισσότερες από μία φορά. Οι επαναλήψεις ελέγχονται πάντοτε από κάποια συνθήκη, η οποία καθορίζει την έξοδο από το βρόχο.

Η ΓΛΩΣΣΑ υποστηρίζει τρεις εντολές επανάληψης, την εντολή ΟΣΟ όπου η επανάληψη ελέγχεται από μία λογική έκφραση στην αρχή και εκτελείται συνεχώς όσο η συνθήκη είναι Αληθής, την εντολή ΜΕΧΡΙΣ_ΟΤΟΥ όπου η συνθήκη βρίσκεται στο τέλος του βρόχου και εκτελείται συνεχώς μέχρις ότου η συνθήκη αυτή γίνει Αληθής και τέλος την εντολή ΓΙΑ, με την οποία ο βρόχος επαναλαμβάνεται για προκαθορισμένο αριθμό φορών.

4.2.1. ΕΝΤΟΛΗ ΟΣΟ…ΕΠΑΝΑΛΑΒΕ

Η γενικότερη δομή επανάληψης υλοποιείται στη ΓΛΩΣΣΑ BDN με την εντολή ΟΣΟ…ΕΠΑΝΑΛΑΒΕ. Σε αυτή, η συνθήκη που ελέγχει την επανάληψη βρίσκεται στην αρχή της επανάληψης και ο βρόχος επαναλαμβάνεται συνεχώς, όσο η συνθήκη αυτή ισχύει. Χαρακτηριστικό της επανάληψης αυτής είναι ότι ο αριθμός των επαναλήψεων δεν είναι γνωστός, ούτε μπορεί να υπολογιστεί πριν από την εκτέλεση του προγράμματος.

Σύνταξη
ΟΣΟ συνθήκη ΕΠΑΝΑΛΑΒΕ
    εντολή-1
    εντολή-2
    …
    Εντολή-ν
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

Παράδειγμα
Αθροισμα <- 0
ΟΣΟ αθροισμα < 1000 ΕΠΑΝΑΛΑΒΕ
    ΔΙΑΒΑΣΕ α
    αθροισμα <- αθροισμα + α
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

Λειτουργία
Ελέγχεται η συνθήκη και αν είναι αληθής, εκτελούνται οι εντολές που βρίσκονται αναμεσα στις ΟΣΟ_ΕΠΑΝΑΛΑΒΕ και ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ. Στη συνέχεια ελέγχεται πάλι η συνθήκη και αν ισχύει, εκτελούνται πάλι οι ίδιες εντολές. Όταν η λογική έκφραση γίνει Ψευδής, τότε σταματάει η επανάληψη και εκτελείται η εντολή μετά το ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ.

Εφόσον μετά από κάθε επανάληψη ελέγχεται εκ νέου η συνθήκη, πρέπει υποχρεωτικά μέσα στο βρόχο να υπάρξει μια εντολή, η οποία να μεταβάλει την τιμή της μεταβλητής που ελέγχεται με την συνθήκη. Σε αντίθετη περίπτωση η επανάληψη δεν θα τερματίζεται και θα εκτελείται συνεχώς.

4.2.2. ΕΝΤΟΛΗ ΜΕΧΡΙΣ_ΟΤΟΥ

Η δεύτερη εντολή επανάληψης που χρησιμοποιεί η ΓΛΩΣΣΑ BDN είναι η εντολή ΜΕΧΡΙΣ_ΟΤΟΥ. Σε αυτή, οι εντολές του βρόχου εκτελούνται μέχρις ότου ικανοποιηθεί κάποια συνθήκη η οποία ελέγχεται στο τέλος της επανάληψης.

Σύνταξη
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
    εντολή-1
    εντολή-2
    …
    εντολή-ν
ΜΕΧΡΙΣ_ΟΤΟΥ λογική-έκφραση

Παράδειγμα

```
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
        ΔΙΑΒΑΣΕ α
        αθροισμα <- αθροισμα + α
ΜΕΧΡΙΣ_ΟΤΟΥ ( αθροισμα >= 1000 )
```

Λειτουργία

Εκτελούνται οι εντολές μεταξύ των ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ και ΜΕΧΡΙΣ_ΟΤΟΥ. Στη συνέχεια ελέγχεται η λογική έκφραση και αν δεν ισχύει (είναι Ψευδής), τότε οι εντολές που βρίσκονται ανάμεσα στις ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ και ΜΕΧΡΙΣ_ΟΤΟΥ, εκτελούνται πάλι. Ελέγχεται ξανά η λογική έκφραση και αν δεν ισχύει, επαναλαμβάνεται η εκτέλεση των ίδιων εντολών.

Όταν η λογική έκφραση γίνει Αληθής τότε σταματάει η επανάληψη και εκτελείται η εντολή μετά από την ΜΕΧΡΙΣ_ΟΤΟΥ.

Πολύ συχνά η ίδια επαναληπτική διαδικασία μπορεί να γραφεί εξίσου σωστά χρησιμοποιώντας είτε τη δομή ΟΣΟ…ΕΠΑΝΑΛΑΒΕ είτε τη δομή ΜΕΧΡΙΣ_ΟΤΟΥ και είναι προσωπική επιλογή του προγραμματιστή ποια από τις δυο θα χρησιμοποιήσει. Υπάρχουν όμως περιπτώσεις όπου η χρήση της εντολής ΜΕΧΡΙΣ_ΟΤΟΥ οδηγεί σε απλούστερα και πιο ευκολονόητα προγράμματα. Γενικά σε περιπτώσεις όπου η επανάληψη θα συμβεί υποχρεωτικά μία φορά, είναι προτιμότερη η χρήση της ΜΕΧΡΙΣ_ΟΤΟΥ.

4.2.3. ΕΝΤΟΛΗ ΓΙΑ…ΑΠΟ…ΜΕΧΡΙ…ΜΕ ΒΗΜΑ

Πολύ συχνά ο αριθμός των επαναλήψεων που πρέπει να εκτελεστούν, είναι γνωστός από την αρχή. Αν και αυτού του είδους οι επαναλήψεις μπορούν να αντιμετοπιστούν με τη χρήση των προηγούμενων εντολών επανάληψης, η ΓΛΩΣΣΑ BDN διαθέτει και την εντολή ΓΙΑ. Η εντολή αυτή χειρίζεται μια μεταβλητή, στην οποία αρχικά εκχωρείται η αρχική τιμή. Η τιμή της μεταβλητής συγκρίνεται με την τελική τιμή και εφόσον είναι μικρότερη από αυτή, τότε εκτελούνται οι εντολές που βρίσκονται στο βρόχο(ανάμεσα στις εντολές ΓΙΑ και ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ). Στη συνέχεια η μεταβλητή ελέγχου αυξάνεται κατά την τιμή που ορίζει το ΒΗΜΑ. Αν η νέα τιμή είναι μικρότερη της τελικής, τότε ο βρόχος εκτελείται ξανά. Η διαδικασία αυτή επαναλαμβάνεται συνεχώς, έως ότου η τιμή ελέγχου γίνει μεγαλύτερη της τελικής τιμής, οπότε τερματίζεται η επανάληψη και το πρόγραμμα συνεχίζει με την εντολή που ακολουθεί το ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ.

Φυσικά ο χρήστης έχει τη δυνατότητα να επιλέξει σαν τελική τιμή, δηλαδή τιμή σύγκρισης, κάποια μικρότερη από την αρχική με την προϋπόθεση όμως ότι το βήμα σε αυτή την περίπτωση θα είναι αρνητικό.

Π.χ: ΓΙΑ ι ΑΠΟ 20 ΜΕΧΡΙ 0 ΜΕ ΒΗΜΑ -2
Εντολή
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

Σύνταξη
ΓΙΑ μεταβλητή ΑΠΟ τιμή1 ΜΕΧΡΙ τιμή2 ΜΕ_ΒΗΜΑ τιμή3
εντολή-1
εντολή-2
…
εντολή-ν
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

Παράδειγμα
ΓΙΑ αριθμο ΑΠΟ 1 ΜΕΧΡΙ 100 ΜΕ_ΒΗΜΑ 2
αθροισμα <- αθροισμα + αριθμο
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

Λειτουργία
Οι εντολές του βρόγχου εκτελούνται για όλες τις τιμές της μεταβλητής από την αρχική τιμή την τελική τιμή, αυξανόμενες με την τιμή του βήματος.

Παράδειγμα
Το παρακάτω πρόγραμμα υπολογίζει το άθροισμα των περιττών αριθμών που είναι μικρότεροι από το 100.

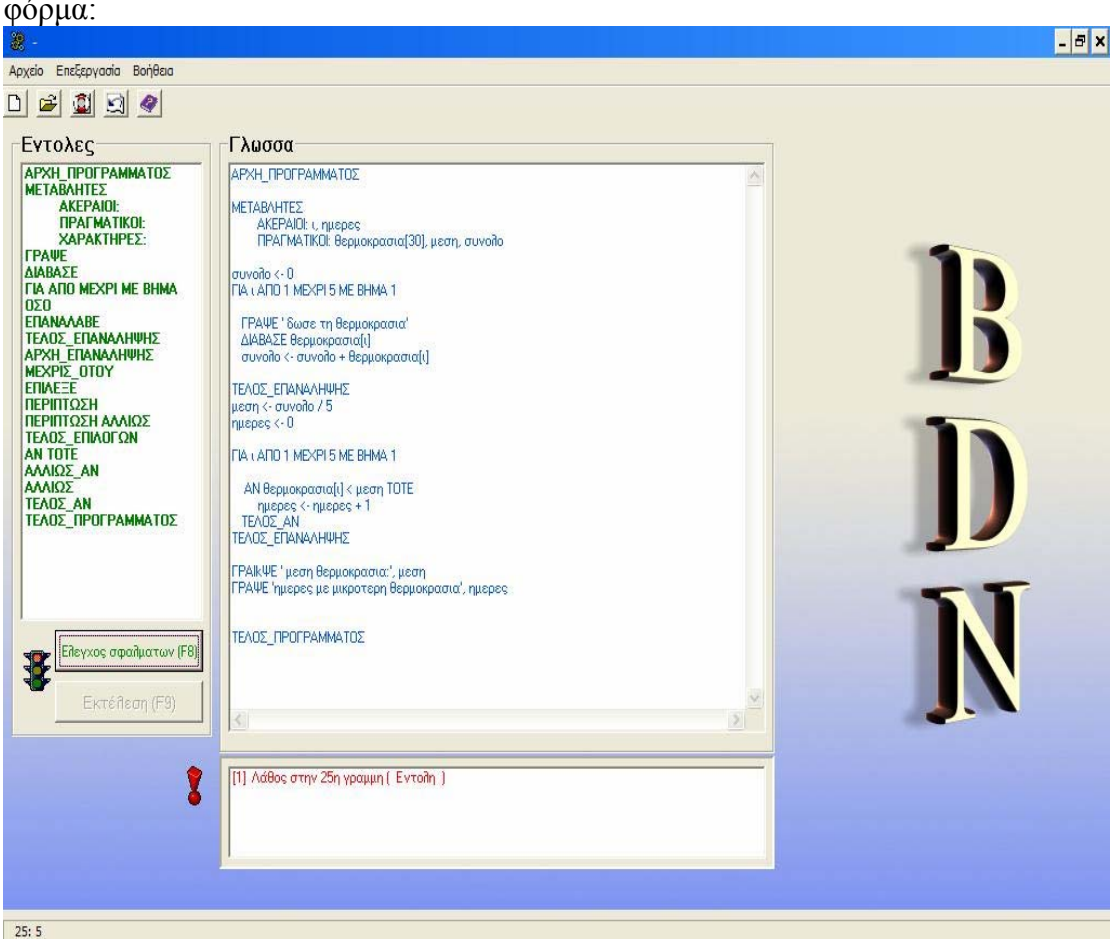
```
ΑΡΧΗ_ΠΡΟΓΡΑΜΜΑΤΟΣ
ΜΕΤΑΒΛΗΤΕΣ
        ΑΚΕΡΑΙΟΙ: αθροισμα, αριθμος
Αθροισμα <- 0
ΓΙΑ αριθμος ΑΠΟ 1 ΜΕΧΡΙ 100 ΜΕ ΒΗΜΑ 2
        αθροισμα <- αθροισμα + αριθμος
ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ
ΓΡΑΨΕ ‘Αθροισμα περιττων αριθμων είναι :’, αθροισμα
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

5. ΥΛΟΠΟΙΗΣΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

5.1. ΠΑΡΟΥΣΙΑΣΗ - ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

Η κατασκευή του προγράμματος μας, έγινε με τη βοήθεια της γλώσσας προγραμματισμού Borland C++ Builder 5.

Εργαστήκαμε σε οπτικό περιβάλλον και δημιουργήσαμε την παρακάτω φόρμα:



Για την δημιουργία της φόρμας χρησιμοποιήσαμε τα ακόλουθα αντικείμενα

- 1 memo,
- 2 ListBox,
- 2 GroupBox,
- 2 Button,
- 1 Panel,
- 4 Image,
- 1 ToolBar,
- 5 ToolButton,
- 1 MainMenu και
- 1 StatusBar.

των οποίων η χρησιμότητα αναλύεται παρακάτω.

Memo:

Σε αυτή την περιοχή ο χρήστης γράφει τον κώδικα του προγράμματος.

ListBox:

Το πρώτο ListBox βρίσκεται στο αριστερό μέρος της φόρμας, και περιέχει όλες τις εντολές που υποστηρίζει η γλώσσα BDN. Ο χρήστης έχει τη δυνατότητα να επιλέγει την εντολή που θέλει να χρησιμοποιήσει, κάνοντας κλίκ πάνω στην εντολή. Όταν γίνει αυτό η εντολή γράφεται στο memo στην σειρά που βρίσκεται ο κέρσορας.

Το δεύτερο ListBox είναι αυτό που βρίσκεται κάτω από τον κώδικα. Σε αυτό το αντικείμενο εμφανίζονται τα συντακτικά λάθη του προγράμματος, καθώς και η γραμμή του κώδικα στην οποία βρίσκονται.

Button:

Έλεγχος σφαλμάτων. Βρίσκεται κάτω από το ListBox των εντολών. Κάνοντας κλίκ στο κουμπί το πρόγραμμα ελέγχει για συντακτικά λάθη, και στην περίπτωση που βρεί, τα εμφανίζει στο ListBox κάτω από τον κώδικα, μαζί με τη γραμμή του κώδικα στην οποία υπάρχει το λάθος. Όταν το πρόγραμμα βρει συντακτικά λάθη δεν δίνει τη δυνατότητα στο χρήστη να 'τρέξει' το πρόγραμμα.

Εκτέλεση. Βρίσκεται κάτω από το Button Έλεγχος σφαλμάτων και με κλίκ πάνω του, το πρόγραμμα που ο χρήστης έχει γράψει εκτελείται. Όπως είπαμε και παραπάνω, βασική προϋπόθεση για να εκτελεστεί ο κώδικας είναι να γίνει ο έλεγχος σφαλμάτων, και να μην βρεθεί κάποιο συντακτικό λάθος. Σε αυτή την περίπτωση το Button 'Εκτέλεση' γίνεται ενεργό και δίνεται η δυνατότητα στο χρήστη να το χρησιμοποιήσει.

ToolBar & ToolButtons:

Η ToolBar βρίσκεται στο πάνω μέρος της φόρμας και αποτελείται από 5 ToolButton. Εδω ο χρήστης μπορεί να επιλέξει μια από τις κυριότερες λειτουργίες του Menu κάνοντας κλικ σε ένα από τα 5 κουμπιά συντόμευσης.

StatusBar:

Κατά την διάρκεια του προγράμματος, μας εμφανίζει την θέση του κέρσορα μέσα στο Memo. Δηλαδή την γραμμή και την στήλη.

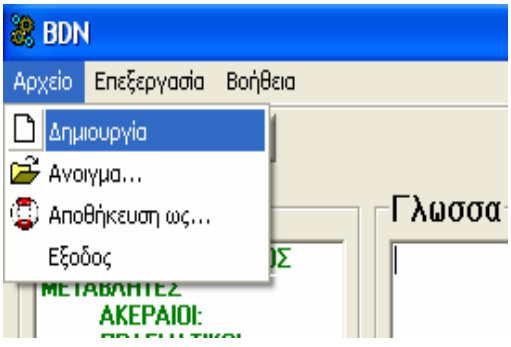
Panel:

Για λόγους αισθητικής το Panel περιέχει ένα από τα ListBox.

MainMenu:

Περιλαμβάνει 3 μενού: Αρχείο, Επεξεργασία και Βοήθεια.

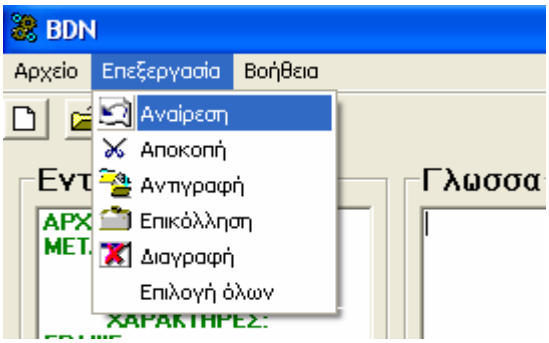
Το μενού Αρχείο αποτελείται από 4 επιλογές: Δημιουργία, Άνοιγμα, Αποθήκευση ως και Έξοδος:



Αρχείο->Δημιουργία: Δημιουργείται ένα καινούργιο αρχείο σβήνοντας οτιδήποτε υπάρχει μέσα στο Memo.
Αρχείο->Άνοιγμα: Ο χρήστης μπορεί να φορτώσει στο Memo ένα αρχείο (Κώδικας BDN) το οποίο βρίσκεται σε μορφή κειμένου (txt).

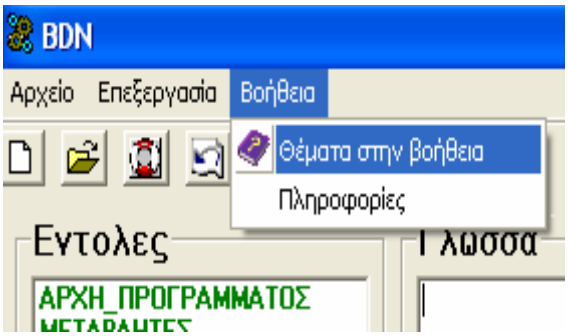
Αρχείο->Αποθήκευση ως: Ο χρήστης έχει την δυνατότητα να σώσει το περιεχόμενο του Memo σε ένα αρχείο κειμένου (txt) βάζοντας το όνομα και την κατάληξη του αρχείου. Πχ File.txt
Αρχείο->Έξοδος: Τερματίζεται το πρόγραμμα.

Το μενού Επεξεργασία περιλαμβάνει 6 επιλογές: Αναίρεση, Αποκοπή, Αντιγραφή, Επικόλληση, Διαγραφή, Επιλογή όλων:

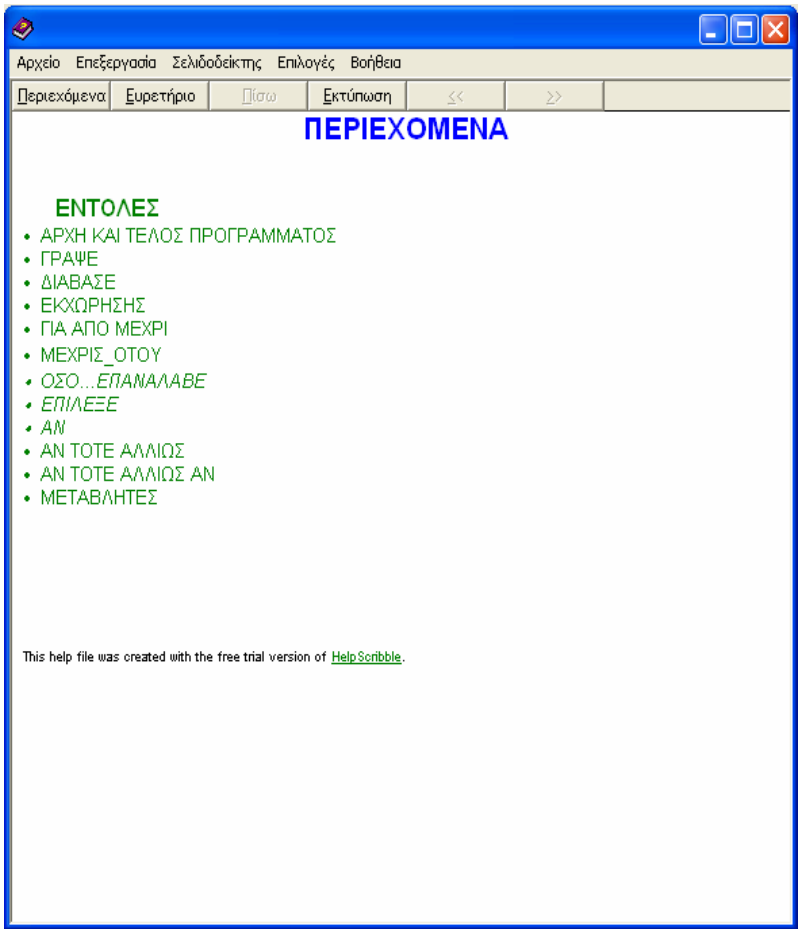


Επεξεργασία->Αναίρεση: Επαναφέρει την προηγούμενη ενέργεια του χρήστη στο Memo.
Επεξεργασία->Αποκοπή: Αποκόπτει το επιλεγμένο κείμενο του Memo.
Επεξεργασία->Αντιγραφή: Αντιγράφει το επιλεγμένο κείμενο του Memo.
Επεξεργασία->Επικόλληση: Εάν προηγουμένως έγινε μια αντιγραφή ή αποκοπή κειμένου τότε θα τοποθετηθεί στην θέση του κέρσορα.
Επεξεργασία->Διαγραφή: Διαγράφει το επιλεγμένο κείμενο του Memo.
Επεξεργασία->Επιλογή όλων: Μαρκάρει όλο το κείμενο του Memo.

Το μενού Βοήθεια αποτελείται από 2 επιλογές: Θέματα στην βοήθεια, Πληροφορίες.



Βοήθεια->Θέματα στην βοήθεια:



Εμφανίζεται το αρχείο βοήθειας του προγράμματος. Εδώ ο χρήστης μπορεί να βρει πολλές πληροφορίες σχετικά με τις εντολές του προγράμματος καθώς και πολλά παραδείγματα.
Βοήθεια->Πληροφορίες: Πληροφορίες για την εφαρμογή BDN Language.

Ας δούμε τώρα τον τρόπο με τον οποίο λειτουργεί το πρόγραμμα που δημιουργήσαμε:

Ο χρήστης γράφει τον κώδικα στο memo. Εμείς πρέπει τον κώδικα αυτό που έχει γραφτεί στη γλώσσα BDN, να τον μεταφράσουμε στην γλώσσα C. Αφου ο χρήστης γράψει τον κώδικα, όπως είπαμε κ προηγουμένως, για να μπορέσει να τον 'εκτελέσει' πρέπει πρώτα να κάνει έλεγχο για πιθανά σφάλματα, πατώντας το κουμπί 'Έλεγχος σφαλμάτων'. Μόλις ο χρήστης πατήσει αυτό το κουμπί το πρόγραμμα κάνει την μετάφραση του κώδικα αλλά ταυτόχρονα κάνει και έλεγχο για πιθανά συντακτικά λάθη. Στη συνέχεια και με την προϋπόθεση ότι δεν έχουν βρεθεί συντακτικά λάθη, δίνεται στο χρήστη η δυνατότητα να εκτελέσει τον κώδικά του πατώντας το κουμπί Εκτέλεση.

Ουσιαστικά όλα γίνονται με το πάτημα του κουμπιού Έλεγχος σφαλμάτων. Όταν λοιπόν το κουμπί αυτό πατηθεί, ανοίγουμε το αρχείο 'kodikas.c', μέσα στο οποίο αποθηκεύουμε τον μεταφρασμένο, σε γλώσσα C, κώδικά. Στη συνέχεια, παίρνουμε μία μία τις γραμμές του Memo. Για κάθε γραμμή αποθηκεύουμε την πρώτη λέξη σε μία μεταβλητή, τη word, και στη συνέχεια καλούμε τη συνάρτηση

compiler στην οποία και συγκρίνουμε το περιεχόμενο της μεταβλητής με όλες τις εντολές που υποστηρίζει η γλώσσα μας (BDN). Για παράδειγμα αν σε κάποια γραμμή του memo υπάρχει η εντολή : *ΓΡΑΨΕ 'το συνολο των επιτυχοντων ειναι:', επιτυχοντες* , τότε γίνεται το εξής: η μεταβλητή word παίρνει την πρώτη λέξη της γραμμής του memo, δηλαδή παίρνει την τιμή ΓΡΑΨΕ. Καλώντας στη συνέχεια τη συνάρτηση compiler ελέγχεται αν το περιεχόμενο της μεταβλητής αντιστοιχεί σε κάποια εντολή, αν δηλαδή η λέξη (στο συγκεκριμένο παράδειγμα η ΓΡΑΨΕ), είναι κάποια εντολή ή όχι.

Αν δεν αντιστοιχεί σε εντολή εμφανίζεται το κατάλληλο μήνυμα λάθους στο ListBox των λαθών. Αν είναι εντολή τότε ανάλογα με το ποια εντολή είναι καλούμε και την αντίστοιχη συνάρτηση. Στο παράδειγμά μας καλούμε τη συνάρτηση *synartisiGrapse.* Στη συνέχεια η *synartisiGrapse* αναλαμβάνει να κάνει τους απαραίτητους ελέγχους και τις κατάλληλες ενέργειες (π.χ. θα καλέσει και άλλες συναρτήσεις κ.α.), έως και τον τελευταίο χαρακτήρα της γραμμής του memo. Φυσικά οι ενέργειες που κάνει κάθε φορά, εξαρτώνται και από το περιεχόμενο της εντολής. Στο παράδειγμά μας θα κάνει πρώτα τη μετάφραση στο κείμενό μας, και στη συνέχεια θα ελέγξει και θα μεταφράσει τη μεταβλητή.(κάνοντας βέβαια και τους απαραίτητους ελέγχους, αν π.χ. η μεταβλητή έχει δηλωθεί από τον χρήστη κλπ). Αν όμως είχαμε την περίπτωση όπου η εντολή περιέχει απλά ένα κείμενο π.χ. ΓΡΑΨΕ 'hello' ή απλά μια μεταβλητή π.χ. ΓΡΑΨΕ ποσότητα, οι ενέργειες της συνάρτησης *synartisiGrapse* θα ήταν διαφορετικές. Πιο συγκεκριμένα στην πρώτη περίπτωση θα γινόταν απλά μετάφραση του κειμένου, κάτι που αντιστοιχεί στην εντολή της γλώσσας C : *printf("hello");* , που είναι και αυτό το οποίο πρέπει να αποθηκευτεί στο αρχείο kodikas.c. Στη δεύτερη περίπτωση που ακολουθεί μεταβλητή μετά την εντολή αυτό που πρέπει να γραφτεί στο αρχείο είναι : *printf("%i",posotita);* , αν υποθέσουμε βέβαια ότι η ποσότητα έχει δηλωθεί σαν ακέραια μεταβλητή. Συνεπώς οι δυο περιπτώσεις διαφέρουν μεταξύ τους ως προς τις ενέργειες και τους ελέγχους που πρέπει να κάνει η *synartisiGrapse*. Αυτό το «καταλαβαίνει» η συνάρτηση που έχει γραφτεί για κάθε εντολή, καταλαβαίνει δηλαδή αν η εντολή ΓΡΑΨΕ περιέχει κείμενο ή μεταβλητή ή το συνδιασμό και των δυο, και ανάλογα κάνει τους ελέγχους και τις ενέργειες που χρειάζονται.

Αφού τελειώσουμε και με τον τελευταίο χαρακτήρα του memo, παίρνουμε την επόμενη γραμμή του και επαναλαμβάνουμε την ίδια διαδικασία. Αυτό γίνεται ως και την τελευταία γραμμή του memo, όπου και τελειώνει ο κώδικας που έχουμε γράψει.

Στη συνέχεια αφού ολοκληρωθεί η μετάφραση και γίνει ο έλεγχος σφαλμάτων, αυτό που μένει είναι να εκτελέσουμε τον κώδικα. Όταν ο χρήστης επιλέξει να τρέξει το πρόγραμμα, τότε γίνεται compile στο αρχείο kodikas.C, στο οποίο έχει αποθηκευτεί ο κώδικας κατά την διάρκεια της μετάφρασης και στην συνέχεια τρέχει το εκτελέσιμο αρχείο που έχει δημιουργηθεί. Αυτό γίνεται με τις εντολές system("bcc32 kodikas.c"); και system("kodikas.exe"); που βρίσκονται στη συνάρτηση Button2Click. Με την εντολή system("bcc32 kodikas.c"); μεταγλωττίζουμε το αρχείο kodikas.c χρισιμοποιόντας τον compiler της C++ Builder, και στη συνέχεια με την εντολή system("kodikas.exe"); τρέχουμε το εκτελέσιμο αρχείο.

Παρακάτω ακολουθεί ένα παράδειγμα όπου φένεται ο τρόπος που λειτουργεί το προγραμμα:

1) ΑΡΧΗ_ΠΡΟΓΡΑΜΜΑΤΟΣ
2) ΜΕΤΑΒΛΗΤΕΣ

```
    ΑΚΕΡΑΙΟΙ: αμ[100], ι, συνεχεια, ζ
    ΠΡΑΓΜΑΤΙΚΟΙ: βαθμος1[100], βαθμος2[100], μο[100], ρ

3)  συνεχεια <- 1
4)  ι <- -1
5)  ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
6)  ι <- ι +1
7)  ΓΡΑΨΕ 'Δωσε ΑΜ'
8)  ΔΙΑΒΑΣΕ αμ[ι]
9)  ΓΡΑΨΕ 'Δωσε 1ο βαθμο'
10) ΔΙΑΒΑΣΕ βαθμος1[ι]
11) ΓΡΑΨΕ 'Δωσε 2ο βαθμο '
12) ΔΙΑΒΑΣΕ βαθμος2[ι]
13) μο[ι] <- ( βαθμος1[ι] + βαθμος2[ι] ) / 2
14) ΑΝ ( βαθμος1[ι] >10 || βαθμος1[ι]<0) || ( βαθμος2[ι]>10 || βαθμος2[ι]<0) ΤΟΤΕ
15)     ΓΡΑΨΕ 'Λανθασμενα στοιχεια'
16)     ι <- ι - 1
17) ΑΛΛΙΩΣ
18)     ΓΡΑΨΕ '1 για συνεχεια και 0 για τελος '
19)     ΔΙΑΒΑΣΕ συνεχεια
20) ΤΕΛΟΣ_ΑΝ
21) ΜΕΧΡΙΣ_ΟΤΟΥ ( συνεχεια = 0 )
22) ΓΡΑΨΕ 'Οι επιτυχοντες ειναι: '
23) ΓΙΑ ζ ΑΠΟ 0 ΜΕΧΡΙ ι ΜΕ ΒΗΜΑ 1
24) ΑΝ μο[ζ]>=5 ΤΟΤΕ
25)     ΓΡΑΨΕ 'ΑΜ:', αμ[ζ]
26)     ΓΡΑΨΕ 'Βαθμος 1: ', βαθμος1[ζ]
27)     ΓΡΑΨΕ 'Βαθμος 2:', βαθμος2[ζ]
28)     ΓΡΑΨΕ 'Μεσος ορος:', μο[ζ]
29) ΤΕΛΟΣ_ΑΝ
30) ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
31) ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Μόλις ο χρήστης πατήσει το κουμπί Έλεγχος σφαλμάτων, πηγαίνουμε στην πρώτη γραμμή του memo, οπου υπάρχει η εντολή ΑΡΧΗ_ΠΡΟΓΡΑΜΜΑΤΟΣ. Η μεταβλητή word παίρνει την τιμή ΑΡΧΗ_ΠΡΟΓΡΑΜΜΑΤΟΣ και στη συνέχεια καλείται η συνάρτηση compiler όπου και συγκρίνει το περιεχόμενο της μεταβλητής με όλες τις εντολές. Αφού βρει ότι πρόκειται για την εντολή ΑΡΧΗ_ΠΡΟΓΡΑΜΜΑΤΟΣ, τυπώνει στο αρχείο τα εξής :

```
fputs("#include <stdlib.h>\n",pf);
fputs("#include <conio.h>\n",pf);
fputs("#include <stdio.h>\n",pf);
fputs("int main(void)\n",pf);
fputs("{\n",pf);
```

Μόλις γίνουν αυτά μηδενίζουμε την μεταβλητή word. Αυτό γίνεται στη συνάρτηση Button1Click.

Στη συνέχεια αυξάνουμε τη γραμμή κατά ένα και η μεταβλητή παίρνει την τιμή ΜΕΤΑΒΛΗΤΕΣ. Γίνεται ο έλεγχος και αφού βρεθεί ότι περιέχει αυτή τη λέξη καλεί τη συνάρτηση synartisimetablites. Εκεί αναγνωρίζουμε, ελέγχουμε και τυπώνουμε τους τύπους που ακολουθούν, οι οποίοι στο παράδειγμα μας είναι

ΑΚΕΡΑΙΟΙ και ΠΡΑΓΜΑΤΙΚΟΙ. Μόλις το πρόγραμμα διαβάσει τον τύπο ΑΚΕΡΑΙΟΙ, τυπώνει στο kodikas.c τον αντίστοιχο τύπο στην C δηλαδή int, και στη συνέχεια καλεί την συνάρτηση compiler_2 όπου οι μεταβλητές που ακολουθούν μπαίνουν στις αντίστοιχες δομές που έχουμε δημιουργήσει για κάθε τύπο και μεταφράζονται, ελέγχονται και τυπώνονται στο αρχείο.

Στις επόμενες δυο γραμμές (3,4) εντοπίζεται η εντολή εκχώρησης. Ελέγχεται αν η μεταβλητή που βρίσκεται αριστερά της εντολής <- υπάρχει και είναι δηλωμένη, και μεταφράζεται στο αρχείο kodikas.c μέσω της συνάρτησης metafrasi. Αν όχι εμφανίζει το κατάλληλο μήνυμα λάθους. Το ίδιο κάνει και για τη μεταβλητή που βρίσκεται στα δεξιά.

Στη γραμμή 5 εντοπίζεται η λέξη ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ που αντιστοιχεί στην εντολή της C, do..while. Αποθηκεύεται στην μεταβλητή word, καλείται η συνάρτηση compiler όπου και τυπώνουμε στο αρχείο το: do{. Στη συνέχεια μεταφράζονται όλες οι εντολές που υπάρχουν μέσα στη δομή επανάληψης δηλαδή ανάμεσα στο ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ και ΜΕΧΡΙΣ_ΟΤΟΥ.
Στη γραμμή 6 γίνεται ότι και στις γραμμές 3 και 4. Στη γραμμή 7 η μεταβλητή word παίρνει την τιμή ΓΡΑΨΕ και καλώντας τη συνάρτηση compiler, εντοπίζεται ότι περιέχει την εντολή ΓΡΑΨΕ. Στη συνέχεια καλεί τη συνάρτηση synartisiGrapse, μέσα στην οποία γίνονται οι έλεγχοι για το αν έχουμε κείμενο ή μεταβλητή ή τον οποιοδήποτε συνδυασμό και των δυο, και μεταφράζονται καλώντας τη συνάρτηση metafrash_metablhton_grapse. Στο συγκεκριμένο παράδειγμα κάνει μετάφραση στο κείμενο που βρίσκεται μεσα στα εισαγωγικά (‘’). Ακολουθεί στη γραμμή 8 η εντολή ΔΙΑΒΑΣΕ, οπου αφού ελεγχθεί και διαπιστωθεί πια εντολή είναι, καλούμε τη συνάρτηση synartisiDiabase μεσο της οποίας ελέγχουμε αν ο πίνακας και ο δείκτης, έχει δηλωθεί και κάνουμε και τη μετάφραση τους καλώντας τη συνάρτηση metafrasi.

Στις γραμμές 9 ως 12 γίνονται τα ίδια πράγματα με τις γραμμές 7 και 8, ενώ στη γραμμή 13 έχουμε όπως και στις 3, 4 εντολή εκχώρησης.

Στη γραμμή 14 υπάρχει η εντολή ΑΝ. η μεταβλητή word παίρνει την τιμή ΑΝ και στην συνάρτηση compiler διαπιστώνεται για πια εντολή επρόκειτο. Μόλις γίνει αυτό τοποθετείται στο αρχείο μας το : (if( ) και στη συνέχεια καλείται η συνάρτηση synartisi_An. Εκεί μεταφράζονται οι συνθήκες της ΑΝ, που στο παράδειγμά μας είναι (βαθμος1[ι] >10 || βαθμος1[ι]<0) || (βαθμος2[ι]>10 || βαθμος2[ι]<0). Όταν βρεθεί η λέξη ΤΟΤΕ κλείνει η παρένθεση που περικλείει την ή τις συνθήκες, ανοίγει μια αγκύλη ‘{‘, μηδενίζεται η μεταβλητή word και πηγαίνουμε στην επόμενη γραμμή του memo. Στις γραμμές 15,16 και 18, 19 έχουμε εντολές για τις οποίες έχουμε αναλύσει παραπάνω πώς λειτουργεί το πρόγραμμά μας. Στη γραμμή 17 η word παίρνει την τιμή ΑΛΛΙΩΣ και πηγαίνοντας στη συνάρτηση compiler τυπώνει στο αρχείο kodikas.c :

```
}
else
{
```

Η πρώτη αγκύλη που κλείνει είναι αυτή που έχει ανοίξει με τη λέξη ΤΟΤΕ και περιέχει όλες τις εντολές που θα εκτελεστούν μέσα στην εντολή ΑΝ. Η δεύτερη αγκύλη θα περιέχει τις εντολές που πρέπει να εκτελεστούν μετά το ΑΛΛΙΩΣ, και θα κλείσει όταν βρεθεί το ΤΕΛΟΣ_ΑΝ. Όταν δηλαδή φτάσουμε στη γραμμή 20.

Στη συνέχεια στη γραμμή 21 υπάρχει η εντολή ΜΕΧΡΙΣ_ΟΤΟΥ. Με την εντολή αυτή κλείνει η εντολή που άρχισε στη γραμμή 5 (ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ….ΜΕΧΡΙΣ_ΟΤΟΥ), που μεταφράζεται στη γλώσσα C στην εντολή :

```
do
{
```

εντολές
}
while (συνθήκη)

Ελέγχεται το περιεχόμενο της word στην συνάρτηση compiler και αφού βρεθεί ότι περιέχει την εντολή ΜΕΧΡΙΣ_ΟΤΟΥ, κλείνει η αγκύλη που έχει ανοίξει με την εντολή ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ, τοποθετείται η while και στη συνέχεια καλείται η συνάρτηση synartisi_Mexris_Otou στην οποία γίνονται και οι υπόλοιποι έλεγχοι και η μετάφραση της συνθήκης.

Στις γραμμές 22, 25-28 ακολουθούν εντολές για τις οποίες έχουμε μιλήσει. Στη γραμμή 23 έχουμε την εντολή ΓΙΑ. Αποθηκεύεται και πάλι η λέξη στην μεταβλητή word, και στην compiler καταλαβαίνουμε ότι πρόκειται για την εντολή ΓΙΑ. Τυπώνεται στο αρχείο το ‘for(‘ και καλείται η συνάρτηση ΓΙΑ. Εκεί γίνονται οι απαραίτητοι έλεγχοι, και τυπώνεται η συνέχεια της εντολής. Όταν φτάσουμε στο τέλος της γραμμής του memo που έχει την εντολή ΓΙΑ, ανοίγουμε μία αγκύλη μέσα στην οποία θα γραφτούν οι εντολές που περιέχει η ΓΙΑ και η οποία θα κλείσει με την εντολή ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ, που ακολουθεί στη γραμμή 30 του παραδείγματος. Στο παράδειγμα μέσα στην εντολή επανάληψης ΓΙΑ, υπάρχει η εντολή ΑΝ, την οποία περιγράψαμε και παραπάνω στις γραμμές 14-20 του κώδικα.

Ακολουθεί η εντολή ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ που σημαίνει πως ο κώδικας έφτασε στο τέλος του. Όταν στη συνάρτηση compiler βρεθεί ότι η μεταβλητή word περιέχει τη λέξη ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ τυπώνουμε στο αρχείο τα εξής:

```
fputs("\n getch();\n",pf);
fputs("return 0;\n",pf);
fputs("}\n",pf);
```

Στη συνέχεια το μόνο που μένει είναι ο χρήστης να πατήσει το κουμπί Εκτέλεση για να εκτελεστεί ο κώδικας.

Παρακάτω φένεται ο κώδικας της γλώσσας BDN μεταφρασμένος σε γλώσσα C :

```
#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
int main(void)
{
int  am[100], i, sunexeia, z;
float  vathmos1[100], vathmos2[100], mo[100], r  ;
sunexeia =1;
i =-1;

do
{
        i =i+1;
        printf("Dwse AM");
        printf("\n");
        scanf("%i",&am[i] );
        printf(" Dwse 1o vathmo");
        printf("\n");
        scanf("%f",&vathmos1[i]);
        printf("Dwse 2o vathmo ");
```

```
        printf("\n");
        scanf("%f",& vathmos2[i]);
        mo[i] =(vathmos1[i]+vathmos2[i])/2;

        if( ( vathmos1[i] >10 || vathmos1[i]<0) || ( vathmos2[i]>10 || vathmos2[i]<0) )
        {
                printf(" Lanthasmena stoixeia");
                printf("\n");
                i =i-1;
        }
        else
        {
                printf(" 1 gia sunexeia kai 0 gia telos ");
                printf("\n");
                scanf("%i",&sunexeia);

        }
}
while ( ( sunexeia != 0 ) );printf(" Oi epituxontes einai: ");
printf("\n");
for( z =0; z<=i; z= z+1)
{
        if( mo[z]>=5 )
        {
                printf(" AM:%i",am[z]);
                printf("\n");
                printf(" Vathmos 1: %.2f",vathmos1[z]);
                printf("\n");
                printf(" Vathmos 2:%.2f",vathmos2[z]);
                printf("\n");
                printf(" Mesos oros:%.2f",mo[z]);
                printf("\n");

        }
}

getch();
return 0;
}
```

5.2. ΣΥΝΑΡΤΗΣΕΙΣ

Για τη κατασκευή του προγράμματος χρησιμοποιήσαμε τις παρακάτω συναρτήσεις:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
void __fastcall TForm1::compiler(TObject *Sender)
void __fastcall TForm1:: synartisiGrapse(TObject *Sender)
void __fastcall TForm1:: synartisiDiabase(TObject *Sender)
void __fastcall TForm1::compiler_2(TObject *Sender)
```

```
void __fastcall TForm1::synartisimetablites(TObject *Sender)
void __fastcall TForm1::sigkrisi_metabliton(TObject *Sender)
void __fastcall TForm1::metafrash_metablhton_grapse(TObject *Sender)
void __fastcall TForm1::metafrasi(TObject *Sender)
void __fastcall TForm1::synartisiOso(TObject *Sender)
void __fastcall TForm1::synartisi_Mexris_Otou(TObject *Sender)
void __fastcall TForm1::ListBox1Click(TObject *Sender)
void __fastcall TForm1::Memo1MouseDown(TObject *Sender,
    TMouseButton Button, TShiftState Shift, int X, int Y)
void __fastcall TForm1::Open1Click(TObject *Sender)
void __fastcall TForm1::Exit1Click(TObject *Sender)?
void __fastcall TForm1::New1Click(TObject *Sender)
void __fastcall TForm1::SaveAs1Click(TObject *Sender)
void __fastcall TForm1::Emfanise_error(TObject *Sender)
void __fastcall TForm1::Memo1KeyUp(TObject *Sender, WORD &Key,
    TShiftState Shift)
void __fastcall TForm1::sigkrisi_metabliton_2(TObject *Sender)
void __fastcall TForm1::Button2Click(TObject *Sender)
void __fastcall TForm1::Arxikes_times(TObject *Sender)
void __fastcall TForm1::entolh_ekxrorhshs(TObject *Sender)
void __fastcall TForm1::elegxos_pinakon(TObject *Sender)
void __fastcall TForm1::elegxos_pinakon2(TObject *Sender)
void __fastcall TForm1::sigkrisi_pinakon(TObject *Sender)
void __fastcall TForm1::About1Click(TObject *Sender)
void __fastcall TForm1::Help2Click(TObject *Sender)
void __fastcall TForm1::Memo1Change(TObject *Sender)
void __fastcall TForm1::sigkrisi_metabliton_3(TObject *Sender)
void __fastcall TForm1::Undo1Click(TObject *Sender)
void __fastcall TForm1::Cut1Click(TObject *Sender)
void __fastcall TForm1::Copy1Click(TObject *Sender)
void __fastcall TForm1::Paste1Click(TObject *Sender)
void __fastcall TForm1::SelectAll1Click(TObject *Sender)
void __fastcall TForm1::Delete1Click(TObject *Sender)
void _fastcall TForm1::synartisi_An(TObject *Sender)
void _fastcall TForm1::synartisi_Gia(TObject *Sender)
```

Παρακάτω αναφέρουμε δυο λόγια για την χρησιμότητα κάθε συνάρτησης

*void __fastcall TForm1::Button1Click(TObject *Sender) :*

Η συνάρτηση αυτή καλείται όταν πατηθεί το κουμπί *Εκτέλεση*. Διαβάζει το Memo γραμμή προς γραμμή και το μεταφράζει λέξη προς λέξη.

*void __fastcall TForm1::compiler(TObject *Sender) :*

Η συνάρτηση δέχεται κάθε λέξη του Memo και ελέγχει αν η λέξη αντιστοιχεί σε κάποια εντολή, και ανάλογα με την εντολή που αντιστοιχεί καλεί την κατάλληλη συνάρτηση.

*void __fastcall TForm1:: synartisiGrapse(TObject *Sender):*

Μεταφράζει την εντολή ΓΡΑΨΕ και ελέγχει το συντακτικό της (π.χ. ονόματα μεταβλητών κ.λ.π.). Αν υπάρχει λάθος, εμφανίζει το κατάλληλο μήνυμα λάθους στο LisBox των λαθών, καλώντας τη συνάρτηση
void __fastcall TForm1::Emfanise_ error(TObject *Sender).

*void __fastcall TForm1:: synartisiDiabase(TObject *Sender):*

Μεταφράζει την εντολή ΔΙΑΒΑΣΕ και ελέγχει το συντακτικό της (π.χ. ονόματα μεταβλητών κ.λ.π.). Αν υπάρχει λάθος, εμφανίζει το κατάλληλο μήνυμα λάθους στο LisBox των λαθών, καλώντας τη συνάρτηση
void __fastcall TForm1::Emfanise_ error(TObject *Sender).

*void __fastcall TForm1::compiler_2(TObject *Sender):*

Καλείται από τις συναρτήσεις void __fastcall TForm1::synartisimetablites (TObject *Sender) και void __fastcall TForm1::synartisiGrapse(TObject *Sender) και μεταφράζει το κείμενο της εντολής ΓΡΑΨΕ (καλώντας τη συνάρτηση void __fastcall TForm1::metafrasi(TObject *Sender)) και αποθηκεύει όλες τις μεταβλητές του προγράμματος στις κατάλληλες δομές τις οποίες από πρίν έχουμε δημιουργήσει.

*void __fastcall TForm1::synartisimetablites(TObject *Sender):*

Μεταφράζει τους τύπους και τα ονόματα των μεταβλητών στην αρχή του προγράμματος όπου και δηλώνονται οι μεταβλητές.

*void __fastcall TForm1::sigkrisi_metabliton(TObject *Sender):*

Καλείται από τις συναρτήσεις void __fastcall TForm1:: synartisiGrapse(TObject *Sender) και void __fastcall TForm1:: synartisiDiabase(TObject *Sender), και συγκρίνει τις μεταβλητές αυτών των εντολών με τις μεταβλητές που έχουν δηλωθεί στο πρόγραμμα.

*void __fastcall TForm1::metafrash_metablhton_grapse(TObject *Sender):*

Μεταφράζει τις μεταβλητές της εντολής ΓΡΑΨΕ.

*void __fastcall TForm1::metafrasi(TObject *Sender):*

Μεταφράζει το κείμενο της ΓΡΑΨΕ καλώντας τη συνάρτηση void __fastcall TForm1::compiler_2(TObject *Sender) και τις μεταβλητές που υπάρχουν στις διάφορες εντολές του προγράμματος πλην της ΓΡΑΨΕ.

*void __fastcall TForm1::synartisiOso(TObject *Sender):*

Μεταφράζει την εντολή ΟΣΟ..ΕΠΑΝΕΛΑΒΕ και ελέγχει το συντακτικό της (π.χ. ονόματα μεταβλητών κ.λ.π.) και αν υπάρχει λάθος, εμφανίζει το κατάλληλο μήνυμα λάθους, καλώντας τη συνάρτηση void __fastcall TForm1::Emfanise_ error(TObject *Sender).

*void __fastcall TForm1::synartisi_Mexris_Otou(TObject *Sender):*

Μεταφράζει την εντολή ΜΕΧΡΙΣ….ΟΤΟΥ και ελέγχει το συντακτικό της (π.χ. ονόματα μεταβλητών κ.λ.π.) και αν υπάρχει λάθος, εμφανίζει το κατάλληλο μήνυμα λάθους, καλώντας τη συνάρτηση void __fastcall TForm1::Emfanise_ error(TObject *Sender).

*void __fastcall TForm1::ListBox1Click(TObject *Sender):*

Γράφει την εντολή που θα επιλέξει ο χρήστης από το ListBox στο Memo, όπου γράφεται ο κώδικας.

*void __fastcall TForm1::Memo1MouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift, int X, int Y):*

Εμφανίζει τη γραμμή και τη στήλη του κέρσορα στο StatusBar.

*void __fastcall TForm1::Open1Click(TObject *Sender):*

Χρησιμοποιείται από το πρόγραμμα για άνοιγμα αρχείων .txt.

*void __fastcall TForm1::Exit1Click(TObject *Sender):*

Με αυτή τη συνάρτηση τερματίζεται η εφαρμογή.

*void __fastcall TForm1::New1Click(TObject *Sender):*

Επαναφέρει το πρόγραμμα στην αρχική του κατάσταση.

*void __fastcall TForm1::SaveAs1Click(TObject *Sender):*

Χρησιμοποιείται για την αποθήκευση του κώδικα DBN σε αρχείο .txt.

*void __fastcall TForm1::Emfanise_error(TObject *Sender):*

Καλείται από διάφορες συναρτήσεις του προγράμματος, όταν εντοπιστεί συντακτικό λάθος, για την εμφάνιση του κατάλληλου μηνύματος στο ListBox των μηνυμάτων λάθους.

*void __fastcall TForm1::Memo1KeyUp(TObject *Sender, WORD &Key, TShiftState Shift):*

Εμφανίζει τη γραμμή και τη στήλη που βρίσκεται ο κέρσορας στο StatusBar.

*void __fastcall TForm1::sigkrisi_metabliton_2(TObject *Sender):*

Συγκρίνει τις μεταβλητές των εντολών πλην της ΓΡΑΨΕ.

*void __fastcall TForm1::Button2Click(TObject *Sender):*

Μεταγλωττίζει και τρέχει τον κώδικα.

*void __fastcall TForm1::Arxikes_times(TObject *Sender):*

Η συνάρτηση αυτή δίνει σε όλες τις Global μεταβλητές, τις αρχικές τους τιμές.

*void __fastcall TForm1::entolh_ekxrorhshs(TObject *Sender):*

Μεταφράζει την εντολή εκχώρησης (<-) και ελέγχει το συντακτικό της (π.χ. ονόματα μεταβλητών κ.λ.π.) και αν υπάρχει λάθος, εμφανίζει το κατάλληλο μήνυμα λάθους, καλώντας τη συνάρτηση void __fastcall TForm1::Emfanise_ error(TObject *Sender).

*void __fastcall TForm1::elegxos_pinakon(TObject *Sender):*

Ελέγχει τα ονόματα των δηλωμένων μεταβλητών και εάν υπάρχει πίνακας ελέγχει τη σύνταξη του.

*void __fastcall TForm1::elegxos_pinakon2(TObject *Sender):*

Ελέγχει το συντακτικό των πινάκων που χρησιμοποιούν όλες οι εντολές του προγράμματος.

*void __fastcall TForm1::sigkrisi_pinakon(TObject *Sender):*

Συγκρίνει το όνομα του πίνακα της εντολής με αυτό που υπάρχει δηλωμένο. Επίσης συγκρίνει και τις θέσεις του πίνακα. Η θέση στην οποία αναφέρεται η εντολή δεν μπορεί να ξεπερνάει τις θέσεις του πίνακα.

*void __fastcall TForm1::About1Click(TObject *Sender):*

Εμφανίζει το AboutBox της φόρμας.

*void __fastcall TForm1::Help2Click(TObject *Sender):*

Εμφανίζει το αρχείο Βοήθειας του προγράμματος.

*void __fastcall TForm1::Memo1Change(TObject *Sender):*

Όταν γίνει μια αλλαγή στο Memo, τότε απενεργοποιείται το κουμπί Εκτέλεση.

*void __fastcall TForm1::sigkrisi_metabliton_3(TObject *Sender):*

Συγκρίνει τους δείκτες των πινάκων, σε περίπτωση που δεν είναι αριθμοί αλλά μεταβλητές, με τις δηλωμένες μεταβλητές. Αν για παράδειγμα έχουμε Pin[j] συγκρίνει το j με τις δηλωμένες μεταβλητές για να δει αν υπάρχει.

*void __fastcall TForm1::Undo1Click(TObject *Sender):*

Κάνει αναίρεση (Undo), και επαναφέρει το Memo στην προηγούμενη κατάσταση του.

*void __fastcall TForm1::Cut1Click(TObject *Sender):*

Κάνει αποκοπή του επιλεγμένου κειμένου.

*void __fastcall TForm1::Copy1Click(TObject *Sender):*

Κάνει αντιγραφή του επιλεγμένου κειμένου.

*void __fastcall TForm1::Paste1Click(TObject *Sender):*

Κάνει επικόλληση, εφόσον πρίν έχει ακολουθήσει αντιγραφή ή αποκοπή.

*void __fastcall TForm1::SelectAll1Click(TObject *Sender):*

Μαρκάρει όλο το κείμενο του Memo.

*void __fastcall TForm1::Delete1Click(TObject *Sender):*

Διαγράφει το επιλεγμένο κείμενο του Memo.

*void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key, TShiftState Shift):*

Καλεί τις συναρτήσεις ανάλογα με το πλήκτρο συντόμευσης που έχει πατηθει.F8 για Έλεγχο και F9 για Εκτέλεση.

*void _fastcall TForm1::synartisi_An(TObject *Sender):*

Μεταφράζει την εντολή ΑΝ και ελέγχει το συντακτικό της (π.χ. ονόματα μεταβλητών κ.λ.π.) και αν υπάρχει λάθος, εμφανίζει το κατάλληλο μήνυμα λάθους, καλώντας τη συνάρτηση void __fastcall TForm1::Emfanise_ error(TObject *Sender).

*void _fastcall TForm1::synartisi_Gia(TObject *Sender):*

Μεταφράζει την εντολή ΓΙΑ..ΑΠΟ..ΜΕΧΡΙ…ΜΕ ΒΗΜΑ, και ελέγχει το συντακτικό της (π.χ. ονόματα μεταβλητών κ.λ.π.) και αν υπάρχει λάθος, εμφανίζει το κατάλληλο μήνυμα λάθους, καλώντας τη συνάρτηση void __fastcall TForm1::Emfanise_ error(TObject *Sender).

5.2. Ο ΚΩΔΙΚΑΣ

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
#include "Unit1.h"
#include <conio.h>
#include <Printers.hpp>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "Unit2.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

//=============================================================
//          M E T A B L I T E S
//=============================================================

char    word[40]="0",
        onomametablitis[40],
        onoma_metabl_se_pinaka[40],
        onoma_metablhtis_dowhile[40];

int     typos_metablitis=0,
        a=0,
        p=0,
        ch=0,
        Synol_arith_metabl=0,
        mikos_metablitis=-1,
        flag_metablhth_meta_grapse=0,
        flag_sunartish_print_bug=0,
        found_tupos_metablt=0,
        flag_gia_epilekse=0,
        flag_gia_periptosh=0,
        metritis_gia_telos_epilogon=0,
        metritis_switch=0,
        metritis_oso_kai_gia=0,
```

```
        metritis_telos_epanalhpshs=0,
        mhkos_arithmon=0,
        sigkr_Grapse=0,
        sigkr_Epilekse=0,
        sigkr_Diabase=0,
        allios=0,
        sigkr_Mexris_Otou=0,
        Memo1_y=0,
        kena=0,
        me_kena=0,
        Found_metablhth=0,
        Found_metablhth_grapse=1,
        mhkos2=0,
        count=0,
        count_debug=0,
        pin_error[100],
        Testing=0,
        m1=0,
        c_metabl2=0,
        counter=0,
        arithmos_agiles=0,
        flag_error_pinakas=0,
        flag_error_metablhtes=0,
        flag_error_ekxorhs=0,
        found_pinakas_se_entolh=0,
        flag_error_thesh_pinaka=1,
        Error=0,
        temp_grammh=0,
        temp_grammh_error=0,
        temp_i,
        omp=0,
        flag_metablhth=0,
        metritis_Telos_An=1,
        metritis_An=1,
        neo_programma=1,
        end_programmatos=1,
        found_metablites=0,
        prothfora=0,
        metr_autakia=0,
        idio_onoma_pinaka=0;

struct akeraioi
{
        AnsiString Akeraios;
};
typedef struct akeraioi my_int;
struct pragmatikoi
{
        AnsiString Pragmatikos;
};
```

```
typedef struct pragmatikoi my_float;
struct xaraktires
{
        AnsiString Xaraktiras;
};
typedef struct xaraktires my_char;
struct Metablites
{
        AnsiString Metablith;
};
typedef struct Metablites my_metabl;

my_int Pinakas_Akeraiwn[30];
my_float Pinakas_Pragmatikwn[30];
my_char Pinakas_Xaraktirwn[30];
my_metabl Pinakas_metablhton[30],Pinakas_metablhton_grapse[30];

char    filename[20]="kodikas.C";
int     grammi=0,i=0,b,
        flag_giametabliti=0,
        lasti,flag_euresis_matablitis=1,
        pm=0,metablhtes_count=0,
        Arith_pin_xarakt=0,count_w=0;
char    pinakas_metabliton[200]="\",";
AnsiString A;
AnsiString Debug=" Λάθος στην ",Debug2="η γραμμη",count_errors;
AnsiString Expl=" ";
FILE *pf;



//---------------------------------------------------------------------

 void __fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------


//***********************************************************************
//    ΠΑΤΗΜΑ ΚΟΥΜΠΙΟΥ ΓΙΑ ΜΕΤΑΓΛΩΤΤΙΣΗ
//***********************************************************************
//Η συνάρτηση αυτή καλείται όταν πατηθεί το κουμπί Εκτέλεση. Διαβάζει το Memo
//γραμμή προς γραμμή και το μεταφράζει λέξη προς λέξη.

void __fastcall TForm1::Button1Click(TObject *Sender)
{
        int     synolikesgrammes,temp777;
```

```
                prothfora++;
                Arxikes_times(Sender);
                synolikesgrammes=Memo1->Lines->Count;
                pf=fopen(filename,"w");
        while(grammi<=synolikesgrammes)
        {
                A=Memo1->Lines->Strings[grammi];
                b=A.Length();
                for(i=1;i<=b;i++)
                {
                        if(A[i]!=' ')
                        {
                                word[count_w]=A[i];
                                if(A[i]=='\'')
                                {
                                        // SynarthrtisiPrint();??????????????????????????
                                }
                                count_w++;
                        }
                        if((A[i]==' ') || (i==b))
                                compiler(Sender);
                }
                flag_sunartish_print_bug=0;
                for(i=1;i<40;i++)
                word[i]=' ';
                grammi++;
                count_w=0;
                allios=0;
                kena=0;
                sigkr_Mexris_Otou=0;
        }
        if(metritis_gia_telos_epilogon!=metritis_switch)
        {
                Expl=" missing ΕΠΙΛΕΞΕ η ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ ";
                Emfanise_error(Sender);
        }
        if(metritis_oso_kai_gia!=metritis_telos_epanalhpshs)
        {
                Expl=" ΛΕΙΠΕΙ ΟΣΟ η ΓΙΑ η ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ ";
                Emfanise_error(Sender);
        }
        if(end_programmatos==1)
        {
                Expl="ΛΕΙΠΕΙ ΤΟ ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ Η ΔΕΝ
ΒΡΙΣΚΕΤΑΙ ΣΤΗ ΣΩΣΤΗ ΘΕΣΗ";
                Emfanise_error(Sender);
        }
        if(neo_programma!=2)
        {
```

```
                Expl=" ΛΕΙΠΕΙ ΤΟ ΑΡΧΗ_ΠΡΟΓΡΑΜΜΑΤΟΣ Η ΔΕΝ ΒΡΙΣΚΕΤΑΙ
ΣΤΗ ΣΩΣΤΗ ΘΕΣΗ";
                count_errors=count_debug+1;
                ListBox2->Items->Strings[count_debug]="["+count_errors+"] (
"+Expl+" )";
                count_debug++;
                Error=1;
        }
        if(Error==1)
        {
                Image3->Visible=true;
                Button2->Enabled=false;
        }
        else
        {
                Button2->Enabled=true;
                Image1->Visible=false;
        }
        Error=0;
        fclose(pf);
        Memo2->Lines->LoadFromFile("kodikas.c");
}
//---------------------------------------------------------------------------


//***************************************************************************
//      ΣΥΝΑΡΤΗΣΗ  C O M P I L E R
//***************************************************************************
// Η συνάρτηση δέχεται κάθε λέξη του Memo και ελέγχει αν η λέξη αντιστοιχεί σε
//κάποια εντολή, και ανάλογα με την εντολή που αντιστοιχεί καλεί την κατάλληλη
//συνάρτηση.


void __fastcall TForm1::compiler(TObject *Sender)
{
        int     ptr=1,c,t,telos_entolis=0,
                telos=0,j,c_metabl=0,
                temp_t,IsNum2,i2,
                autaki=0, x=0,x2=0,
                sigkr_Emfanise=0,
                sigkr_Telos=0,
                sigkr_Oso=0,
                sigkr_Telos_Epanal=0,
                sigkr_Metablites=0,
                sigkr_Arxh_epanal=0,
                sigkr_Entolh_ekxor=0,
                sigkr_Periptosh=0,
                sigkr_Telos_Epilogon=0,
                sigkr_An=0,
                sigkr_Alliws=0,
```

```
                sigkr_Alliws_An=0,
                sigkr_Telos_An=0,
                sigkr_Arxi_Programmatos=0,
                sigkr_Telos_Programmatos=0,
                sigkr_Gia=0,
                command_found=0;
        char    Diabase[10]="ΔΙΑΒΑΣΕ",
                Emfanise[10]="ΕΜΦΑΝΙΣΕ",
                Grapse[10]="ΓΡΑΨΕ",
                Telos[10]="ΤΕΛΟΣ",
                Metablites[20]="ΜΕΤΑΒΛΗΤΕΣ",
                Oso[7]="ΟΣΟ",
                An[2]="ΑΝ",
                Alliws[6]="ΑΛΛΙΩΣ",
                Alliws_An[9]="ΑΛΛΙΩΣ_ΑΝ",
                Telos_An[8]="ΤΕΛΟΣ_ΑΝ",
                Telos_Epanal[17]="ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ",
                Arxh_Epanalhpshs[20]="ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ",
                Mexris_Otou[15]="ΜΕΧΡΙΣ_ΟΤΟΥ",
                Epilekse[10]="ΕΠΙΛΕΞΕ",
                Periptosh[10]="ΠΕΡΙΠΤΩΣΗ",
                Telos_Epilogon[17]="ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ",
                Arxi_Programmatos[17]="ΑΡΧΗ_ΠΡΟΓΡΑΜΜΑΤΟΣ",
                Telos_Programmatos[18]="ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ",
                Gia[3]="ΓΙΑ";

        sigkr_Epilekse=0;
        if(count_w==0)
        {
                fputs(" ",pf);
                kena++;
        }
        telos_entolis=i;
//      Α Ρ Χ Η _ Π Ρ Ο Γ Ρ Α Μ Μ Α Τ Ο Σ
        for(c=0;c<18;c++)
        {
                if(Arxi_Programmatos[c]==word[c])
                        sigkr_Arxi_Programmatos++;
        }
//      Τ Ε Λ Ο Σ _ Π Ρ Ο Γ Ρ Α Μ Μ Α Τ Ο Σ
        for(c=0;c<19;c++)
        {
                if(Telos_Programmatos[c]==word[c])
                        sigkr_Telos_Programmatos++;
        }
//      Δ Ι Α Β Α Σ Ε
        for(c=0;c<7;c++)
        {
                if(Diabase[c]==word[c])
                        sigkr_Diabase++;
```

```
        }
//      Γ Ρ Α Ψ Ε
        for(c=0;c<5;c++)
        {
                if(Grapse[c]==word[c])
                        sigkr_Grapse++;
        }
//      Ε Μ Φ Α Ν Η Σ Ε
        for(c=0;c<8;c++)
        {
                if(Emfanise[c]==word[c])
                        sigkr_Emfanise++;
        }
//      Μ Ε Τ Α Β Λ Η Τ Ε Σ
        for(c=0;c<10;c++)
        {
                if(Metablites[c]==word[c])
                        sigkr_Metablites++;
        }
//      Γ Ι Α  Α Π Ο  Μ Ε Χ Ρ Ι  Μ Ε  Β Η Μ Α
        for(c=0;c<3;c++)
        {
                if(Gia[c]==word[c])
                        sigkr_Gia++;
        }
//      Α Ν
        for(c=0;c<2;c++)
        {
                if(An[c]==word[c])
                        sigkr_An++;
        }
//      Α Λ Λ Ι Ω Σ
        for(c=0;c<6;c++)
        {
                if(Alliws[c]==word[c])
                        sigkr_Alliws++;
        }
//      Α Λ Λ Ι Ω Σ _ Α Ν
        for(c=0;c<9;c++)
        {
                if(Alliws_An[c]==word[c])
                        sigkr_Alliws_An++;
        }
//      Τ Ε Λ Ο Σ _ Α Ν
        for(c=0;c<8;c++)
        {
                if(Telos_An[c]==word[c])
                        sigkr_Telos_An++;
        }
//      Τ Ε Λ Ο Σ
```

```
        for(c=0;c<5;c++)
        {
                if(Telos[c]==word[c])
                        sigkr_Telos++;
        }
//      Ο Σ Ο
        for(c=0;c<3;c++)
        {
                if(Oso[c]==word[c])
                        sigkr_Oso++;
        }
//      Τ Ε Λ Ο Σ  Ε Π Α Ν Α Λ Η Ψ Η Σ
        for(c=0;c<16;c++)
        {
                if(Telos_Epanal[c]==word[c])
                        sigkr_Telos_Epanal++;
        }
//      Α Ρ Χ Η _ Ε Π Α Ν Α Λ Η Ψ Η Σ
        for(c=0;c<15;c++)
        {
                if(Arxh_Epanalhpshs[c]==word[c])
                        sigkr_Arxh_epanal++;
        }
//      Μ Ε Χ Ρ Ι Σ _ Ο Τ Ο Υ
        for(c=0;c<11;c++)
        {
                if(Mexris_Otou[c]==word[c])
                        sigkr_Mexris_Otou++;
        }
//      Ε Π Ι Λ Ε Ξ Ε
        for(c=0;c<8;c++)
        {
                if(Epilekse[c]==word[c])
                        sigkr_Epilekse++;
        }
//      Π Ε Ρ Ι Π Τ Ω Σ Η
        for(c=0;c<9;c++)
        {
                if(Periptosh[c]==word[c])
                        sigkr_Periptosh++;
        }
//      Τ Ε Λ Ο Σ _ Ε Π Ι Λ Ο Γ Ω Ν
        for(c=0;c<14;c++)
        {
                if(Telos_Epilogon[c]==word[c])
                        sigkr_Telos_Epilogon++;
        }
//      ΜΕΤΑΦΡΑΣΗ ΓΙΑ Α Ρ Χ Η _ Π Ρ Ο Γ Ρ Α Μ Μ Α Τ Ο Σ
        if((sigkr_Arxi_Programmatos==17 ||(sigkr_Arxi_Programmatos==18 &&
prothfora==1)) && neo_programma==1)
```

```
        {
                neo_programma++;
                command_found=1;
                if(word[17]!=' ')
                        command_found=0;
                fputs("#include <stdlib.h>\n",pf);
                fputs("#include <conio.h>\n",pf);
                fputs("#include <stdio.h>\n",pf);
                fputs("int main(void)\n",pf);
                fputs("{\n",pf);
}
//      ΜΕΤΑΦΡΑΣΗ ΓΙΑ Τ Ε Λ Ο Σ _ Π Ρ Ο Γ Ρ Α Μ Μ Α Τ Ο Σ
        if(sigkr_Telos_Programmatos==18 && end_programmatos==1)
        {
                command_found=1;
                if(word[18]!=' ')
                        command_found=0;
                fputs("\n getch();\n",pf);
                fputs("return 0;\n",pf);
                fputs("}\n",pf);
                if(metritis_An!=metritis_Telos_An)
                {
                        x=(metritis_An)-(metritis_Telos_An);
                        for(x;x>0;x--)
                        {
                                Expl="ΛΕΙΠΕΙ ΤΟ ΤΕΛΟΣ ΑΝ ΤΗΣ IF";
                                count_errors=count_debug+1;
                                ListBox2->Items-
>Strings[count_debug]="["+count_errors+"] ( "+Expl+" )";
                                count_debug++;
                                Error=1;
                        }
                }
                end_programmatos++;
                metritis_An=1;
                metritis_Telos_An=1;
                i=b;
        }
//      ΜΕΤΑΦΡΑΣΗ ΓΙΑ  Τ Ε Λ Ο Σ _ Ε Π Ι Λ Ο Γ Ω Ν
        if(sigkr_Telos_Epilogon==14)
        {
                command_found=1;
                if(word[14]!=' ')
                        command_found=0;
                metritis_gia_telos_epilogon++;
                fputs("break;\n",pf);
                fputs("}\n",pf);
                if(flag_gia_periptosh==0)
                {
                        Expl=" ΠΕΡΙΠΤΩΣΗ ??? ";
```

```
                Emfanise_error(Sender);
            }
            if(flag_gia_epilekse==0)
            {
                Expl=" ΕΠΙΛΕΞΕ ??? ";
                Emfanise_error(Sender);
            }
            flag_gia_epilekse=0;
            flag_gia_periptosh=0;
        }
//      ΜΕΤΑΦΡΑΣΗ ΓΙΑ  Ε Π Ι Λ Ε Ξ Ε
        if(sigkr_Epilekse==7)
        {
            metritis_switch++;
            flag_gia_epilekse=1;
            c_metabl=0;
            command_found=1;
            if(word[7]!=' ')
                command_found=0;
            fputs("switch(",pf);
            mikos_metablitis=-1;
            for(i;i<=b;i++)
            {
                if(A[i]!=' ')
                {
                    metafrasi(Sender);
                    onomametablitis[c_metabl]=A[i];
                    c_metabl++;
                    mhkos2++;
                    mikos_metablitis++;
                }
            }
            fputs(")",pf);
            fputs("\n{\n",pf);
            c_metabl2=c_metabl;
            c_metabl=0;
            elegxos_pinakon2(Sender);
            if(found_pinakas_se_entolh==1)
            {
                sigkrisi_pinakon(Sender);
                Found_metablhth=1;
                found_pinakas_se_entolh=0;
            }
            else
                sigkrisi_metabliton_2(Sender);
            for(j=0;j<40;j++)
                onomametablitis[j]=' ';
            if(Found_metablhth==0 || flag_error_pinakas==1)
            {
                Expl=" Μεταβλητη ";
```

```
                        Emfanise_error(Sender);
                }
                Found_metablhth=1;
                sigkr_Epilekse=0;
                mikos_metablitis=-1;
        }
//      ΜΕΤΑΦΡΑΣΗ ΓΙΑ  Π Ε Ρ Ι Π Τ Ω Σ Η
        IsNum2=1;
        if(sigkr_Periptosh==9)
        {
                if(flag_gia_epilekse==0)
                {
                        Expl=" ΕΠΙΛΕΞΕ ??? ";
                        Emfanise_error(Sender);
                }
                if(flag_gia_periptosh==1 && allios!=1)
                        fputs("break;\n",pf);
                flag_gia_periptosh=1;
                command_found=1;
                c=i;
                for(c;c<=b;c++)
                {
                        if(A[c]=='Α' && A[c+1]=='Λ' && A[c+2]=='Λ' &&
A[c+3]=='Ι' && A[c+4]=='Ω' && A[c+5]=='Σ')
                        {
                                fputs("default: ",pf);
                                i==b;
                                allios=1;
                        }
                }
                if(allios==0)
                {
                        fputs("case ",pf);
                        autaki=0;
                        i2=i+1;
                        for(i2;i2<=b;i2++)
                        {
                                if(A[i2]=="")
                                        autaki=autaki+1;
                                if(A[i2]==',')
                                {
                                        autaki=0;
                                        IsNum2=1;
                                }
                                if(A[i2]!="" && A[i2]!=' ' && A[i2]!=',' && i2<=b)
                                {
                                        if(A[i2]>='0' && A[i2]<='9' && (A[i2]!=' '))
                                        {
                                                // ok
                                        }
```

```
                                        else
                                                IsNum2=0;
                                }
                        }
                        if(IsNum2==1 && autaki>0)
                        {
                                Expl=" χωρις \" ";
                                Emfanise_error(Sender);
                        }
                        if(autaki!=2 && IsNum2==0)
                        {
                                Expl=" με \" ";
                                Emfanise_error(Sender);
                        }
                        for(i;i<=b;i++)
                        {
                                if(A[i]=="")
                                {
                                        fputs("",pf);
                                }
                                if(A[i]!=',' && A[i]!="")
                                        metafrasi(Sender);
                                if(A[i]==',')
                                {
                                        fputs(" :\n",pf);
                                        fputs(" case ",pf);
                                        i++;
                                        metafrasi(Sender);
                                }
                                if(i==b)
                                        fputs(":\n",pf);
                        }
                }
                sigkr_Periptosh=0;
        }
sigkr_Periptosh=0;
//      ΜΕΤΑΦΡΑΣΗ ΓΙΑ  Γ Ρ Α Ψ Ε
        if(sigkr_Grapse==5)
        {
                command_found=1;
                if(word[5]!=' ')
                        command_found=0;
                t=6;
                temp_t=t;
                for(temp_t;temp_t<=b;temp_t++)
                {
                        if(A[temp_t]!=' ')
                                me_kena++;
                }
                while(t<b)
```

```
                {
                        if(A[t]=='\'')
                        {
                                t=b;
                        }
                        else
                                if(A[t]==' ')
                                        t++;??????????????????????????????????????
                                if(A[t]!=' ' && A[t]!='\'')
                                {
                                        flag_metablhth_meta_grapse=1;
                                        t=b;
                                }
                                temp_t++;
                }
                fputs("printf(\"",pf);
                if(me_kena==0)
                {
                        Expl=" Εντολη ";
                        Emfanise_error(Sender);
                }
                if(flag_metablhth_meta_grapse==1)
                {
                        i--;
                        Found_metablhth_grapse=0;
                        synartisiGrapse(Sender);
                }
                compiler_2(Sender);
                fputs("printf(\"\\n\");",pf);
                fputs("\n",pf);
                x2=metr_autakia%2;
                if(metr_autakia%2!=0 && metr_autakia!=0)
                {
                        Expl=" Σύνταξη  (')";
                        Emfanise_error(Sender);
                }
                metr_autakia=0;
                me_kena=0;
        }
        sigkr_Grapse=0;
//      ΜΕΤΑΦΡΑΣΗ ΓΙΑ  Γ Ρ Α Ψ Ε
        if(sigkr_Diabase==7)
        {
                temp_t=i;
                me_kena=0;
                for(temp_t;temp_t<=b;temp_t++)
                {
                        if(A[temp_t]!=' ')
                                me_kena++;
                }
```

```
                if(me_kena==0)
                {
                        Expl=" Εντολη ";
                        Emfanise_error(Sender);
                }
                command_found=1;
                if(word[7]!=' ')
                        command_found=0;
                fputs("scanf(\"",pf);
                synartisiDiabase(Sender);
                }
                sigkr_Diabase=0;
//      ΜΕΤΑΦΡΑΣΗ ΓΙΑ  Α Ν
        if(sigkr_An==2)
        {
                sigkr_An=0;
                command_found=1;
                if(word[2]!=' ')
                        command_found=0;
                fputs("\nif(",pf);
                synartisi_An(Sender);
                metritis_An++;
        }
//      ΜΕΤΑΦΡΑΣΗ ΓΙΑ Α Λ Λ Ι Ω Σ _ Α Ν
        if(sigkr_Alliws_An==9 || sigkr_Alliws_An==6)
        {
                if(sigkr_Alliws_An==9)
                {
                        sigkr_Alliws_An=0;
                        command_found=1;
                        if(word[9]!=' ')
                                command_found=0;
                        fputs("}\nelse\n",pf);
                        fputs("if(",pf);
                        i=i+1;
                        synartisi_An(Sender);
                }
                else
                        if(sigkr_Alliws_An==6)
                        {
                                sigkr_Alliws=0;
                                command_found=1;
                                if(word[6]!=' ')
                                        command_found=0;
                                        fputs("}\nelse\n{\n",pf);
                        }
        }
//      ΜΕΤΑΦΡΑΣΗ ΓΙΑ Τ Ε Λ Ο Σ _ Α Ν
        if(sigkr_Telos_An==8)
        {
```

```
                command_found=1;
                if(word[8]!=' ')
                        command_found=0;
                sigkr_Telos_An=0;
                fputs("\n}",pf);
                i=b;
                metritis_Telos_An++;
        }
//      ΜΕΤΑΦΡΑΣΗ Γ Ι Α
        if(sigkr_Gia==3)
        {
                sigkr_Gia=0;
                command_found=1;
                metritis_oso_kai_gia++;
                fputs("\nfor(",pf);
                synartisi_Gia(Sender);
        }
//      ΜΕΤΑΦΡΑΣΗ ΓΙΑ Μ Ε Τ Α Β Λ Η Τ Ε Σ
        if(sigkr_Metablites==10)
        {
                if(neo_programma!=2)
                {
                        Expl=" ΛΕΙΠΕΙ ΤΟ ΑΡΧΗ_ΠΡΟΓΡΑΜΜΑΤΟΣ Η ΔΕΝ
ΒΡΙΣΚΕΤΑΙ ΣΤΗ ΣΩΣΤΗ ΘΕΣΗ";
                        count_errors=count_debug+1;
                        ListBox2->Items->Strings[count_debug]="["+count_errors+"] (
"+Expl+" )";
                        count_debug++;
                        Error=1;
                }
                found_metablites=1;
                sigkr_Metablites=0;
                command_found=1;
                if(word[10]!=' ')
                        command_found=0;
                synartisimetablites(Sender);

        }
        sigkr_Metablites=0;
//      ΜΕΤΑΦΡΑΣΗ ΓΙΑ Ο Σ Ο
        if(sigkr_Oso==3)
        {
                sigkr_Oso=0;
                command_found=1;
                if(word[3]!=' ')
                        command_found=0;
                fputs("while ",pf);
                synartisiOso(Sender);
        }
        sigkr_Oso=0;
```

```
//      ΜΕΤΑΦΡΑΣΗ ΓΙΑ  Τ Ε Λ Ο Σ _ Ε Π Α Ν Α Λ Η Ψ Η Σ
        if(sigkr_Telos_Epanal==16)
        {
                fputs("}\n",pf);
                command_found=1;
                if(word[16]!=' ')
                        command_found=0;
                sigkr_Telos_Epanal=0;
                metritis_telos_epanalhpshs++;
        }
        sigkr_Telos_Epanal=0;
//      ΜΕΤΑΦΡΑΣΗ ΓΙ Α  Α Ρ Χ Η _ Ε Π Α Ν Α Λ Η Ψ Η Σ
        if(sigkr_Arxh_epanal==15)
        {
                fputs("\ndo\n",pf);
                command_found=1;
                if(word[15]!=' ')
                        command_found=0;
                for(count=1;count<=kena;count++)
                fputs(" ",pf);
                fputs("{\n",pf);
        }
        sigkr_Arxh_epanal=0;
//      ΜΕΤΑΡΑΣΗ ΓΙΑ  Μ Ε Χ Ρ Ι Σ _ Ο Τ Ο Υ
        if(sigkr_Mexris_Otou==11)
        {
                fputs("\n}\n",pf);
                command_found=1;
                if(word[11]!=' ')
                        command_found=0;
                for(count=0;count<=kena;count++)
                        fputs(" ",pf);
                fputs("while ",pf);
                synartisi_Mexris_Otou(Sender);
        }
        if(count_w==0)
                command_found=1;
//      ΜΕΤΑΦΡΑΣΗ ΓΙΑ Ε Ν Τ Ο Λ Η _ Ε Κ Χ Ο Ρ Η Σ Η Σ
        c=0;
        while(c<=b)
        {
                c++;
                if(c<b)
                        if(A[c]=='<' && A[c+1]=='-')
                        {
                                sigkr_Entolh_ekxor=1;
                                command_found=1;
                                sigkr_Entolh_ekxor=0;
                                entolh_ekxrorhshs(Sender);
                        }
```

```
        }
        if(command_found==0 && temp_grammh!=grammi)
        {
                Expl=" Εντολη ";
                Emfanise_error(Sender);
                command_found=1;
                temp_grammh=grammi;
        }
}
//=============================================================


//*****************************************************************
//            ΣΥΝΑΡΤΗΣΗ G R A P S E
//*****************************************************************
// Μεταφράζει την εντολή ΓΡΑΨΕ και ελέγχει το συντακτικό της (π.χ. ονόματα
//μεταβλητών κ.λ.π.). Αν υπάρχει λάθος, εμφανίζει το κατάλληλο  μήνυμα λάθους
//στο LisBox των λαθών.

void __fastcall TForm1::synartisiGrapse(TObject *Sender)
{
        char    kk[30];
        int     j=0,j2,v,
                flag_onomametablitis=0;
        for(i;i<=b;i++)
        {
                if(A[i]==' ' || i==b)
                {
                        if(mikos_metablitis==-1)
                        {
                                if(flag_sunartish_print_bug==0)
                                        fputs("\"",pf);
                                metafrash_metablhton_grapse(Sender);
                                if(flag_sunartish_print_bug==0)
                                {
                                        fputs(");\n",pf);
                                        flag_sunartish_print_bug=1;
                                }
                        }
                }
                if(A[i]==',' || flag_metablhth_meta_grapse==1)
                {
                        if(A[i]==',' && A[i-1]==' ')
                        {
                                Expl=" Κενο πριν το ,";
                                Emfanise_error(Sender);
                        }
                        flag_metablhth_meta_grapse=0;
                        i=i+2;
                        if(A[i]=='\")
```

```
                        compiler_2(Sender);
                    mikos_metablitis=-1;
                    for(j2=0;j2<40;j2++)
                    {
                        onomametablitis[j2]=' ';
                    }
                    for(v=i;v<=b;v++)
                        if(A[v]!=' ' && A[v]!=',' && A[v]!='\" &&
flag_onomametablitis==0)
                        {
                            onomametablitis[j]=A[v];
                            mikos_metablitis++;
                            j++;
                            if(A[v-1]==' ' && v==b)
                            {
                                Expl=" Κενο στο τελος ";
                                Emfanise_error(Sender);
                            }
                            if(v<b)
                                if(A[v+1]==' ' || A[v+1]==',' )
                                    flag_onomametablitis=1;
                        }
                    c_metabl2=j;
                    j=0;
                    elegxos_pinakon2(Sender);
                    if(found_pinakas_se_entolh==1)
                    {
                        sigkrisi_pinakon(Sender);
                        Found_metablhth_grapse=1;
                    }
                    sigkrisi_metabliton(Sender);
                    if(flag_error_pinakas==1 || flag_error_metablhtes==1)
                    {
                        Expl=" Πινακας ";
                        Emfanise_error(Sender);
                        flag_error_pinakas=0;
                        flag_error_metablhtes=0;
                    }
                    if(Found_metablhth_grapse==0 && onomametablitis[1]!=' ')
                    {
                        Expl=" Μεταβλητη ";
                        Emfanise_error(Sender);
                    }
                    Found_metablhth_grapse=0;
                    mikos_metablitis=-1;
                }
                for(j=0;j<40;j++)
                {
                    onomametablitis[j]=' ';
                }
```

```
                flag_onomametablitis=0;
                j=0;
                flag_metablhth=0;
        }
}
//=============================================================


//*****************************************************************
//      ΣΥΝΑΡΤΗΣΗ  D I A B A S E
//*****************************************************************
//Μεταφράζει την εντολή ΔΙΑΒΑΣΕ και ελέγχει το συντακτικό της (π.χ. ονόματα
//μεταβλητών κ.λ.π.). Αν υπάρχει λάθος, εμφανίζει το κατάλληλο  μήνυμα λάθους στο
//LisBox των λαθών.

void __fastcall TForm1::synartisiDiabase(TObject *Sender)
{
        int     v,j=0;
                mikos_metablitis=-1;
        for(i;i<=b;i++)
        {
                if(A[i]!=' ')
                {
                        if(mhkos2==0)
                                temp_i=i;
                        onomametablitis[j]=A[i];
                        mhkos2++;
                        mikos_metablitis++;
                        j++;
                }
                if(i==b)
                {
                        c_metabl2=j;
                        elegxos_pinakon2(Sender);
                        if(found_pinakas_se_entolh==1)
                        {
                                sigkrisi_pinakon(Sender);
                                Found_metablhth=1;
                                found_pinakas_se_entolh=0;
                        }
                        else
                                sigkrisi_metabliton_2(Sender);
                }
        }
        if(j>0)
        {
                sigkrisi_metabliton(Sender);
                Testing=0;
                fputs("\",&",pf);
                i=temp_i;
```

```
                for(i;i<=b;i++)
                {
                        metafrasi(Sender);
                }
                fputs(");\n",pf);
                for(j=0;j<40;j++)
                {
                        onomametablitis[j]=' ';
                }
                j=0;
                if(flag_error_pinakas==1 || flag_error_metablhtes==1)
                {
                        Expl=" Πινακας ";
                        Emfanise_error(Sender);
                        flag_error_pinakas=0;
                        flag_error_metablhtes=0;
                }
                if(Found_metablhth==0)
                {
                        Expl=" Μεταβλητη ";
                        Emfanise_error(Sender);
                }
                Found_metablhth=1;
                mikos_metablitis=-1;
        }
        flag_metablhth=0;
}
//=============================================================


//*****************************************************************
//      ΣΥΝΑΡΤΗΣΗ  C O M P I L E R  2
//*****************************************************************
//Καλείται από τις συναρτήσεις synartisimetablites και  synartisiGrapse και
//μεταφράζει το κείμενο της εντολής ΓΡΑΨΕ καλώντας τη συνάρτηση metafrasi και
//αποθηκεύει όλες τις μεταβλητές του προγράμματος στις κατάλληλες δομές τις
//οποίες από πρίν έχουμε δημιουργήσει.

void __fastcall TForm1::compiler_2(TObject *Sender)
{
        int     aytakia=0,
                k=0,m=0,test=1;
        char    temp[40],
                kl[40],
                dok[40]="geia",
                dok2[20]=" geia ";
        float x;
        AnsiString P;
        for(m=0;m<40;m++)
                temp[m]=' ';
```

```
        if(flag_giametabliti==1)
                i=lasti+1;
        for(i;i<=b;i++)
        {
                metafrasi(Sender);

//      αποθηκεύουμε τα ονόματα των μεταβλητών στις κατάλληλες διομές

                if((flag_giametabliti==1) && (typos_metablitis==1))
                {
                        if((A[i]!=' ') && (A[i]!=','))
                        {
                                if((A[i-1]==' ' || A[i-1]==',')&&(A[i]>='0'&&A[i]<='9'))
                                arithmos_agiles=1;
                                temp[k]=A[i];
                                k++;
                                if((A[i]>='α' && A[i]<='ω') || (A[i]>='A' && A[i]<='Ω')
|| (A[i]>='0' && A[i]<='9')|| A[i]=='[' || A[i]==']' || A[i]=='_')
                                {
                                        elegxos_pinakon(Sender);
                                }
                                else
                                        flag_error_metablhtes=1;
                        }
                        if(A[i]==','|| i==b)
                        {
                                Pinakas_Akeraiwn[a].Akeraios=temp;
                                Pinakas_metablhton[pm].Metablith=temp;
                                pm++;
                                a++;
                                k=0;
                                for(m=0;m<40;m++)
                                temp[m]=' ';
                        }
                }
                else if((flag_giametabliti==1) && (typos_metablitis==2))
                {
                        if((A[i]!=' ') && (A[i]!=','))
                        {
                                if((A[i-1]==' ' || A[i-1]==',')&&(A[i]>='0'&&A[i]<='9'))
                                        arithmos_agiles=1;
                                temp[k]=A[i];
                                k++;
                                if((A[i]>='α' && A[i]<='ω') || (A[i]>='A' && A[i]<='Ω')
|| (A[i]>='0' && A[i]<='9')|| A[i]=='[' || A[i]==']' || A[i]=='_')
                                {
                                        elegxos_pinakon(Sender);
                                }
                                else
                                        flag_error_metablhtes=1;
```

```
                        }
                        if(A[i]==','|| i==b)
                        {
                                Pinakas_Pragmatikwn[p].Pragmatikos=temp;
                                Pinakas_metablhton[pm].Metablith=temp;
                                pm++;
                                p++;
                                k=0;
                                for(m=0;m<40;m++)
                                        temp[m]=' ';
                        }
                }
                else if((flag_giametabliti==1) && (typos_metablitis==3))
                {
                        if( (A[i]!=' ') && (A[i]!=','))
                        {
                                if((A[i-1]==' ' || A[i-1]==',')&&(A[i]>='0'&&A[i]<='9'))
                                        arithmos_agiles=1;
                                temp[k]=A[i];
                                k++;
                                if((A[i]>='α' && A[i]<='ω') || (A[i]>='A' && A[i]<='Ω')
|| (A[i]>='0' && A[i]<='9')|| A[i]=='[' || A[i]==']' || A[i]=='_')
                                {
                                        elegxos_pinakon(Sender);
                                }
                                else
                                        flag_error_metablhtes=1;
                        }
                        if(A[i]==',' || i==b)
                        {
                                Pinakas_Xaraktirwn[ch].Xaraktiras=temp;
                                Pinakas_metablhton[pm].Metablith=temp;
                                pm++;
                                Arith_pin_xarakt++;
                                ch++;
                                k=0;
                                for(m=0;m<40;m++)
                                        temp[m]=' ';
                        }
                }
//============================================================
//ews edw  mono otan exw M E T A B L H T E S
//============================================================
                if(A[i]=='\" )
                {
                        metr_autakia++;
                        aytakia++;
                        x=aytakia%2;
                        if(x==0)
                                synartisiGrapse(Sender);
```

```
                }
                if(flag_giametabliti==1 && i==b)
                {
                        fputs(";",pf);
                        fputs("\n",pf);
                }
        }
        if(flag_error_metablhtes==1 || arithmos_agiles!=0)
        {
                Expl=" Δηλωση μεταβλητων ";
                Emfanise_error(Sender);
                arithmos_agiles=0;
        }
        Synol_arith_metabl=a+p+ch;
        flag_giametabliti=0;
        typos_metablitis=0;
        flag_euresis_matablitis=1;
}
//=============================================================


//*****************************************************************
//      ΣΥΝΑΡΤΗΣΗ  M E T A B L I T E S
//*****************************************************************
// Μεταφράζει τους τύπους και τα ονόματα των μεταβλητών στην αρχή του
//προγράμματος όπου και δηλώνονται οι μεταβλητές.


void __fastcall TForm1::synartisimetablites(TObject *Sender)
{
        int     found=0,
                metritis_anw_katw_teleias=0,
                arxiki_grammi=grammi;
        AnsiString G;
        A=Memo1->Lines->Strings[grammi+1];
        b=A.Length();
        for(i=1;i<=b;i++)
        {
                if(A[i]==':')
                {
                        grammi++;
                        A=Memo1->Lines->Strings[grammi+1];
                        b=A.Length();
                        i=1;
                }
        }
        arxiki_grammi++;
        A=Memo1->Lines->Strings[arxiki_grammi];
        b=A.Length();
        while(arxiki_grammi<=grammi)
```

```
{
        for(i=1;i<=b;i++)
        {
                if(A[i]==':')
                {
                        found=1;
                        lasti=i;
                }
        }
        temp_grammh_error=arxiki_grammi;
        while(found==1)
        {
                for(i=1;i<=b;i++)
                {
                        if(A[i]=='A' && A[i+1]=='K' && A[i+2]=='E' &&
A[i+3]=='P' && A[i+4]=='A' && A[i+5]=='I' && A[i+6]=='O' && A[i+7]=='I' &&
flag_euresis_matablitis==1)
                        {
                                fputs("int ",pf);
                                flag_giametabliti=1;
                                found_tupos_metablt=1;
                                typos_metablitis=1;
                                flag_euresis_matablitis=0;
                                compiler_2(Sender);
                        }
                        else
                                if(A[i]=='Π' && A[i+1]=='P' && A[i+2]=='A'
&& A[i+3]=='Γ' && A[i+4]=='M' && A[i+5]=='A' && A[i+6]=='T' && A[i+7]=='I'
&& A[i+8]=='K' && A[i+9]=='O' && A[i+10]=='I' && flag_euresis_matablitis==1)
                                {
                                        fputs("float ",pf);
                                        flag_giametabliti=1;
                                        found_tupos_metablt=1;
                                        typos_metablitis=2;
                                        flag_euresis_matablitis=0;
                                        compiler_2(Sender);
                                }
                        else
                                if(A[i]=='X' && A[i+1]=='A' && A[i+2]=='P'
&& A[i+3]=='A' && A[i+4]=='K' && A[i+5]=='T' && A[i+6]=='H' && A[i+7]=='P'
&& A[i+8]=='E' && A[i+9]=='Σ' && flag_euresis_matablitis==1)
                                {
                                        fputs("char ",pf);
                                        flag_giametabliti=1;
                                        found_tupos_metablt=1;
                                        typos_metablitis=3;
                                        flag_euresis_matablitis=0;
                                        compiler_2(Sender);
                                }
                        found=0;
```

```
                        }
                }
                arxiki_grammi++;
                A=Memo1->Lines->Strings[arxiki_grammi];
                b=A.Length();
        }
        if(found_tupos_metablt==0)
        {
                Expl=" Δηλωση Μεταβλητων ";
                Emfanise_error(Sender);
                found_tupos_metablt=0;
        }
}
//=============================================================

//*************************************************************
//        ΣΥΝΑΡΤΗΣΗ  S I G K R I S I _ M E T A B L I T W N
//*************************************************************
//Καλείται από τις συναρτήσεις synartisiGrapse και synartisiDiabase, και συγκρίνει
//τις μεταβλητές αυτών των εντολών με τις μεταβλητές που έχουν δηλωθεί στο
//πρόγραμμα.

void __fastcall TForm1::sigkrisi_metabliton(TObject *Sender)
{
        int     g=0,
                q=1,
                mikos=0,
                flag_sigkrisi_metabliton=0,
                j,j2=1,
                flag_pinakas=0,
                idio_onoma=0;

//          A K E P A I O I
//-----------------------------------------------------------------------------
                for(g=0;g<a;g++)
                {
                        mikos=0;
                        while(Pinakas_Akeraiwn[g].Akeraios[q]!=' ')
                        {
                                if(Pinakas_Akeraiwn[g].Akeraios[q]==
onomametablitis[q-1] && flag_sigkrisi_metabliton==0)
                                        mikos++;
                                q++;
                                if(Pinakas_Akeraiwn[g].Akeraios[q]=='[' &&
((mikos+1)==q) && onomametablitis[q-1]=='[')
                                        flag_pinakas=1;
                                if(Pinakas_Akeraiwn[g].Akeraios[q]=='[' &&
((mikos+1)==q) && onomametablitis[q-1]!='[')
                                        flag_error_pinakas=1;
```

```
                        if(Pinakas_Akeraiwn[g].Akeraios[q]!='[' &&
((mikos+1)==q) && onomametablitis[q-1]=='[')
                                flag_error_pinakas=1;
                        }
                        if(flag_pinakas==1)
                        {
                                if(sigkr_Epilekse!=7)
                                        fputs("%i",pf);
                                flag_sigkrisi_metabliton=1;
                                flag_pinakas=0;
                                if(sigkr_Grapse==5)
                                {
                                        Pinakas_metablhton_grapse[metablhtes_
                                        count].Metablith=onomametablitis;
                                        metablhtes_count++;
                                }
                        }
                        q=1;
                        if(flag_pinakas==0)
                                if((mikos==(mikos_metablitis+1)) &&
mikos_metablitis!=-1)
                                {
                                        if(sigkr_Epilekse!=7)
                                                fputs("%i",pf);
                                        flag_sigkrisi_metabliton=1;
                                        Testing=1;
                                        if(sigkr_Grapse==5)
                                        {
                                        Pinakas_metablhton_grapse[metablhtes_count].
                                        Metablith=Pinakas_Akeraiwn[g].Akeraios;
                                        metablhtes_count++;   }
                                        }
                                }


//              Π Ρ Α Γ Μ Α Τ Ι Κ Ο Ι
//---------------------------------------------------------------------

                        for(g=0;g<p;g++)
                        {
                                mikos=0;
                                while(Pinakas_Pragmatikwn[g].Pragmatikos[q]!=' ')
                                {
                                        if(Pinakas_Pragmatikwn[g].Pragmatikos[q]=
=onomametablitis[q-1] && flag_sigkrisi_metabliton==0)
                                                mikos++;
                                        q++;
                                        if(Pinakas_Pragmatikwn[g].Pragmatikos[q]=='['
&& ((mikos+1)==q))
                                                flag_pinakas=1;
```

```
                                        if(Pinakas_Pragmatikwn[g].Pragmatikos[q]=='['
&& ((mikos+1)==q) && onomametablitis[q-1]!='[')
                                                if(Pinakas_Pragmatikwn[g].
Pragmatikos[q]!='[' && ((mikos+1)==q) && onomametablitis[q-1]=='[')
                                                        flag_error_pinakas=1;
                                }
                                if(flag_pinakas==1)
                                {
                                        if(sigkr_Epilekse!=7)
                                                fputs("%f",pf);
                                        flag_sigkrisi_metabliton=1;
                                        flag_pinakas=0;
                                        if(sigkr_Epilekse==7)
                                        {
                                                Expl=" Μεταβλητη ";
                                                Emfanise_error(Sender);
                                        }
                                        if(sigkr_Grapse==5)
                                        {

        Pinakas_metablhton_grapse[metablhtes_count]. Metablith=onomametablitis;
                                                metablhtes_count++;
                                        }
                                }
                                q=1;
                                if(flag_pinakas==0)
                                        if((mikos==(mikos_metablitis+1)) &&
mikos_metablitis!=-1)
                                        {
                                                if(sigkr_Grapse==5)
                                                        fputs("%.2f",pf);
                                                else
                                                if(sigkr_Epilekse!=7)
                                                        fputs("%f",pf);
                                                flag_sigkrisi_metabliton=1;
                                                Testing=1;
                                                if(sigkr_Grapse==5)
                                                {
Pinakas_metablhton_grapse[metablhtes_count].Metablith=Pinakas_Pragmatikwn[g].P
ragmatikos;
                                                        metablhtes_count++;
                                                }
                                        }
                                }

//              Χ Α Ρ Α Κ Τ Η Ρ Ε Σ
//---------------------------------------------------------------------

                                for(g=0;g<ch;g++)
```

```
                        {
                                mikos=0;
                                while(Pinakas_Xaraktirwn[g].Xaraktiras[q]!=' ')
                                {
                                        if(Pinakas_Xaraktirwn[g].Xaraktiras[q]=
=onomametablitis[q-1] && flag_sigkrisi_metabliton==0)
                                                mikos++;
                                        q++;
                                        if(Pinakas_Xaraktirwn[g].Xaraktiras[q]=
='[' && ((mikos+1)==q))
                                                flag_pinakas=1;
                                        if(onomametablitis[q-1]=='[' &&
flag_pinakas==1)
                                                flag_error_pinakas=1;
                                }
                                q=1;
                                if((mikos==(mikos_metablitis+1)) &&
mikos_metablitis!=-1)
                                {
                                        if(flag_pinakas==1)
                                        {
                                                if(sigkr_Epilekse!=7)
                                                        fputs("%s",pf);
                                                flag_pinakas=0;
                                                Found_metablhth=1;

                                        }
                                        else
                                        if(sigkr_Epilekse!=7 &&
sigkr_Diabase==7)
                                                        fputs("%s",pf);
                                        else
                                                        fputs("%c",pf);

                                        flag_sigkrisi_metabliton=1;
                                        Testing=1;
                                        if(sigkr_Grapse==5)
                                        {


        Pinakas_metablhton_grapse[metablhtes_count].Metablith=onomametablitis;
                                                metablhtes_count++;
                                        }
                                }
                        }
                        if(flag_sigkrisi_metabliton==1 && sigkr_Epilekse!=7)
                                Found_metablhth_grapse=1;
                        flag_sigkrisi_metabliton=0;
}
//=======================================================
```

```
//*****************************************************************
//  ΣΥΝΑΡΤΗΣΗ M E T A F R A S H _ M E T A B L H T W N
//*****************************************************************
// Μεταφράζει τις μεταβλητές της εντολής ΓΡΑΨΕ.

void __fastcall TForm1::metafrash_metablhton_grapse(TObject *Sender)
{
        int     u=0,fg=1,p=1;
        AnsiString Me;
        AnsiString Exampl="geia ";
        for(u=0;u<metablhtes_count;u++)
        {
                fputs(",",pf);
                Me=Pinakas_metablhton[u].Metablith;
                Me=Pinakas_metablhton_grapse[u].Metablith;
                fg=1;
                while(Me[fg]!=' ')
                {
                        switch(Me[fg])
                        {
                                case 'A': fputs("A",pf); break;
                                case 'B': fputs("V",pf); break;
                                case 'Γ': fputs("G",pf); break;
                                case 'Δ': fputs("D",pf); break;
                                case 'E': fputs("E",pf); break;
                                case 'Z': fputs("Z",pf); break;
                                case 'H': fputs("I",pf); break;
                                case 'Θ': fputs("TH",pf); break;
                                case 'I': fputs("I",pf); break;
                                case 'K': fputs("K",pf); break;
                                case 'Λ': fputs("L",pf); break;
                                case 'M': fputs("M",pf); break;
                                case 'N': fputs("N",pf); break;
                                case 'Ξ': fputs("KS",pf); break;
                                case 'O': fputs("O",pf); break;
                                case 'Π': fputs("P",pf); break;
                                case 'P': fputs("R",pf); break;
                                case 'Σ': fputs("S",pf); break;
                                case 'T': fputs("T",pf); break;
                                case 'Y': fputs("Y",pf); break;
                                case 'Φ': fputs("F",pf); break;
                                case 'X': fputs("X",pf); break;
                                case 'Ψ': fputs("PS",pf); break;
                                case 'Ω': fputs("W",pf); break;
                                case 'α': fputs("a",pf); break;
                                case 'β': fputs("v",pf); break;
                                case 'γ': fputs("g",pf); break;
```

```
                                case 'δ': fputs("d",pf); break;
                                case 'ε': fputs("e",pf); break;
                                case 'ζ': fputs("z",pf); break;
                                case 'η': fputs("i",pf); break;
                                case 'θ': fputs("th",pf); break;
                                case 'ι': fputs("i",pf); break;
                                case 'κ': fputs("k",pf); break;
                                case 'λ': fputs("l",pf); break;
                                case 'μ': fputs("m",pf); break;
                                case 'ν': fputs("n",pf); break;
                                case 'ξ': fputs("ks",pf); break;
                                case 'ο': fputs("o",pf); break;
                                case 'π': fputs("p",pf); break;
                                case 'ρ': fputs("r",pf); break;
                                case 'σ': fputs("s",pf); break;
                                case 'ς': fputs("s",pf); break;
                                case '_': fputs("_",pf); break;
                                case 'τ': fputs("t",pf); break;
                                case 'υ': fputs("u",pf); break;
                                case 'φ': fputs("f",pf); break;
                                case 'χ': fputs("x",pf); break;
                                case 'ψ': fputs("ps",pf); break;
                                case 'ω': fputs("w",pf); break;
                                case '1': fputs("1",pf); break;
                                case '2': fputs("2",pf); break;
                                case '3': fputs("3",pf); break;
                                case '4': fputs("4",pf); break;
                                case '5': fputs("5",pf); break;
                                case '6': fputs("6",pf); break;
                                case '7': fputs("7",pf); break;
                                case '8': fputs("8",pf); break;
                                case '9': fputs("9",pf); break;
                                case '0': fputs("0",pf); break;
                                case '[': fputs("[",pf); break;
                                case ']': fputs("]",pf); break;
                        }
                        fg++;

                }
        }
        metablhtes_count=0;
}
//=============================================================


//*************************************************************
//      ΣΥΝΑΡΤΗΣΗ M E T A F R A S I
//*************************************************************
```

```
// Μεταφράζει το κείμενο της ΓΡΑΨΕ καλώντας τη συνάρτηση compiler_2 και τις
//μεταβλητές που υπάρχουν στις διάφορες εντολές του προγράμματος πλην της
//ΓΡΑΨΕ.

void __fastcall TForm1::metafrasi(TObject *Sender)
{
        switch(A[i])
        {
                case 'A': fputs("A",pf); break;
                case 'B': fputs("B",pf); break;
                case 'C': fputs("C",pf); break;
                case 'D': fputs("D",pf); break;
                case 'E': fputs("E",pf); break;
                case 'F': fputs("F",pf); break;
                case 'G': fputs("G",pf); break;
                case 'H': fputs("H",pf); break;
                case 'I': fputs("I",pf); break;
                case 'J': fputs("J",pf); break;
                case 'K': fputs("K",pf); break;
                case 'L': fputs("L",pf); break;
                case 'M': fputs("M",pf); break;
                case 'N': fputs("N",pf); break;
                case 'O': fputs("O",pf); break;
                case 'P': fputs("P",pf); break;
                case 'Q': fputs("Q",pf); break;
                case 'R': fputs("R",pf); break;
                case 'S': fputs("S",pf); break;
                case 'T': fputs("T",pf); break;
                case 'Y': fputs("Y",pf); break;
                case 'V': fputs("V",pf); break;
                case 'X': fputs("X",pf); break;
                case 'U': fputs("U",pf); break;
                case 'W': fputs("W",pf); break;
                case 'Z': fputs("Z",pf); break;
                case 'a': fputs("a",pf); break;
                case 'b': fputs("b",pf); break;
                case 'c': fputs("c",pf); break;
                case 'd': fputs("d",pf); break;
                case 'e': fputs("e",pf); break;
                case 'f': fputs("f",pf); break;
                case 'g': fputs("g",pf); break;
                case 'h': fputs("h",pf); break;
                case 'i': fputs("i",pf); break;
                case 'j': fputs("j",pf); break;
                case 'k': fputs("k",pf); break;
                case 'l': fputs("l",pf); break;
                case 'm': fputs("m",pf); break;
                case 'n': fputs("n",pf); break;
                case 'o': fputs("o",pf); break;
                case 'p': fputs("p",pf); break;
```

```
            case 'q': fputs("q",pf); break;
            case 'r': fputs("r",pf); break;
            case 's': fputs("s",pf); break;
            case 't': fputs("t",pf); break;
            case 'u': fputs("u",pf); break;
            case 'v': fputs("v",pf); break;
            case 'x': fputs("x",pf); break;
            case 'y': fputs("y",pf); break;
            case 'w': fputs("w",pf); break;
            case 'z': fputs("z",pf); break;
    }
    switch(A[i])
    {
            case 'A': fputs("A",pf); break;
            case 'B': fputs("V",pf); break;
            case 'Γ': fputs("G",pf); break;
            case 'Δ': fputs("D",pf); break;
            case 'E': fputs("E",pf); break;
            case 'Z': fputs("Z",pf); break;
            case 'H': fputs("I",pf); break;
            case 'Θ': fputs("TH",pf); break;
            case 'I': fputs("I",pf); break;
            case 'K': fputs("K",pf); break;
            case 'Λ': fputs("L",pf); break;
            case 'M': fputs("M",pf); break;
            case 'N': fputs("N",pf); break;
            case 'Ξ': fputs("KS",pf); break;
            case 'O': fputs("O",pf); break;
            case 'Π': fputs("P",pf); break;
            case 'P': fputs("R",pf); break;
            case 'Σ': fputs("S",pf); break;
            case 'T': fputs("T",pf); break;
            case 'Y': fputs("Y",pf); break;
            case 'Φ': fputs("F",pf); break;
            case 'X': fputs("X",pf); break;
            case 'Ψ': fputs("PS",pf); break;
            case 'Ω': fputs("W",pf); break;
            case 'α': fputs("a",pf); break;
            case 'β': fputs("v",pf); break;
            case 'γ': fputs("g",pf); break;
            case 'δ': fputs("d",pf); break;
            case 'ε': fputs("e",pf); break;
            case 'ζ': fputs("z",pf); break;
            case 'η': fputs("i",pf); break;
            case 'θ': fputs("th",pf); break;
            case 'ι': fputs("i",pf); break;
            case 'κ': fputs("k",pf); break;
            case 'λ': fputs("l",pf); break;
            case 'μ': fputs("m",pf); break;
            case 'ν': fputs("n",pf); break;
```

```
            case 'ξ': fputs("ks",pf); break;
            case 'ο': fputs("o",pf); break;
            case 'π': fputs("p",pf); break;
            case 'ρ': fputs("r",pf); break;
            case 'σ': fputs("s",pf); break;
            case 'ς': fputs("s",pf); break;
            case 'τ': fputs("t",pf); break;
            case 'υ': fputs("u",pf); break;
            case 'φ': fputs("f",pf); break;
            case 'χ': fputs("x",pf); break;
            case 'ψ': fputs("ps",pf); break;
            case 'ω': fputs("w",pf); break;
    }

    switch(A[i])
    {

            case '1': fputs("1",pf); break;
            case '2': fputs("2",pf); break;
            case '3': fputs("3",pf); break;
            case '4': fputs("4",pf); break;
            case '5': fputs("5",pf); break;
            case '6': fputs("6",pf); break;
            case '7': fputs("7",pf); break;
            case '8': fputs("8",pf); break;
            case '9': fputs("9",pf); break;
            case '0': fputs("0",pf); break;
            case ',': fputs(",",pf); break;
            case ' ': fputs(" ",pf); break;
            case '[': fputs("[",pf); break;
            case ']': fputs("]",pf); break;
            case '{': fputs("{",pf); break;
            case '}': fputs("}",pf); break;
            case '!': fputs("!",pf); break;
            case '#': fputs("#",pf); break;
            case '$': fputs("$",pf); break;
            case '%': fputs("%",pf); break;
            case '^': fputs("^",pf); break;
            case '&': fputs("&",pf); break;
            case '*': fputs("*",pf); break;
            case '(': fputs("(",pf); break;
            case ')': fputs(")",pf); break;
            case '_': fputs("_",pf); break;
            case '-': fputs("-",pf); break;
            case '+': fputs("+",pf); break;
            case '=': fputs("=",pf); break;
            case ':': fputs(":",pf); break;
            case ';': fputs(";",pf); break;
            case '|': fputs("|",pf); break;
            case '?': fputs("?",pf); break;
```

```
                case '>': fputs(">",pf); break;
                case '.': fputs(".",pf); break;
                case '<': fputs("<",pf); break;
                case '\\': fputs("\\",pf); break;
                case '/': fputs("/",pf); break;
        }

}
//=============================================================

//*************************************************************
//          ΣΥΝΑΡΤΗΣΗ   O S O
//*************************************************************
// Μεταφράζει την εντολή ΟΣΟ..ΕΠΑΝΕΛΑΒΕ και ελέγχει το συντακτικό της (π.χ.
//ονόματα μεταβλητών κ.λ.π.) και αν υπάρχει λάθος, εμφανίζει το κατάλληλο
//μήνυμα λάθους, καλώντας τη συνάρτηση Emfanise_ error.

void __fastcall TForm1::synartisiOso(TObject *Sender)
{
        int     flag_parenthesi=-1,
                flag_xwris_parenthesi=0,
                sigkr_Epanalabe=0,
                sigkr_Kai=0,
                sigkr_H=0,
                c_metabl=0,
                counter_arithmos=0,
                cl,
                c=0,
                z3=0,
                er=0,
                arithm_parenthes=0;
        char    Epanalabe[10]="ΕΠΑΝΑΛΑΒΕ",
                Kai[5]="ΚΑΙ",
                H[3]="Η";

        metritis_oso_kai_gia++;
        if(A[i+1]!='(')
                fputs("(",pf);
        if(A[i+1]=='(')
                fputs("(",pf);
        for(i;i<=b;i++)
        {
                if(A[i]=='(')
                {
                        flag_parenthesi=0;
                        flag_xwris_parenthesi=1;
                        c=0;
                        arithm_parenthes++;
                }
                if(A[i]==')')
```

```
                {
                        flag_parenthesi=1;
                        i++;
                        fputs(")",pf);
                        arithm_parenthes--;
                }
                if( A[i]==' ' || flag_parenthesi==0 || flag_xwris_parenthesi==0)
                {
                        if(A[i]=='E' && A[i+1]=='Π' && A[i+2]=='A' &&
A[i+3]=='N' && A[i+4]=='A' && A[i+5]=='Λ' && A[i+6]=='A' && A[i+7]=='B'
&& A[i+8]=='E')
                        {
                                sigkr_Epanalabe=9;
                                i=b;
                        }
                        else
                        {
                                if(A[i]=='=' && A[i-1]!='>' && A[i-1]!='<')
                                        fputs("=",pf);
                                if(A[i]=='<' && A[i+1]=='>')
                                {
                                        fputs("!=",pf);
                                        i=i+2;
                                }
                                else
                                        metafrasi(Sender);
                        }
                        if(A[i]!='(' && A[i]!=' ' && A[i]!='<' && A[i]!='>' &&
A[i]!='=' && sigkr_Epanalabe!=9)
                        {
                                onomametablitis[c_metabl]=A[i];
                                mhkos2++;
                                c_metabl++;
                                if(A[i]>='0' && A[i]<='9')
                                        counter_arithmos++;
                                if(A[i+1]==' ')
                                {
                                        c_metabl2=c_metabl;
                                        elegxos_pinakon2(Sender);
                                        if(found_pinakas_se_entolh==1)
                                        {
                                                sigkrisi_pinakon(Sender);
                                                Found_metablhth=1;
                                                found_pinakas_se_entolh=0;
                                        }
                                        else
                                                sigkrisi_metabliton_2(Sender);
                                        if(Found_metablhth==0 &&
(counter_arithmos!=c_metabl))
                                        {
```

```
                                        Expl=" Μεταβλητη ";
                                        Emfanise_error(Sender);
                                }
                                c_metabl=0;
                                mhkos2=0;
                                Found_metablhth=0;
                                counter_arithmos=0;
                                found_pinakas_se_entolh=0;
                                for(cl=0;cl<40;cl++)
                                {
                                        onomametablitis[cl]=' ';
                                }
                        }
                }
        }

        if(A[i]!=' ' && flag_parenthesi==1)
        {
                if(A[i]==Epanalabe[c])
                        sigkr_Epanalabe++;
                if(A[i]==Kai[c])
                        sigkr_Kai++;
                if(A[i]=='H')
                        fputs("||",pf);
                c++;
        }
        if(sigkr_Kai==3)
        {
                fputs("&&",pf);
                sigkr_Kai=0;
        }
}
if(flag_xwris_parenthesi==0)
        fputs(")",pf);
else
        fputs(")",pf);

if(sigkr_Epanalabe==9)
{
        fputs("\n",pf);
        for(count=0;count<=kena;count++)
                fputs(" ",pf);
        fputs("{\n",pf);
}
else
{
        Expl=" Συνταξη ";
        Emfanise_error(Sender);
}
if(arithm_parenthes!=0)
```

```
        {
                Expl=" παρενθεση ";
                Emfanise_error(Sender);
        }
        flag_metablhth=0;
}
//=============================================================
```

```
//*****************************************************************
//      ΣΥΝΑΡΤΗΣΗ  M E X R I S _ O  T  O U
//*****************************************************************
//Μεταφράζει την εντολή ΜΕΧΡΙΣ….ΟΤΟΥ και ελέγχει το συντακτικό της (π.χ.
//ονόματα μεταβλητών κ.λ.π.) και αν υπάρχει λάθος, εμφανίζει το κατάλληλο  μήνυμα
//λάθους, καλώντας τη συνάρτηση Emfanise_ error.

void __fastcall TForm1::synartisi_Mexris_Otou(TObject *Sender)
{
        int     flag_parenthesi=-1,
                flag_xwris_parenthesi=0,
                sigkr_Kai=0,
                sigkr_H=0,
                arithm_parenthes=0,
                c=0,
                c_c=0,
                c_s=1,
                cl,
                isa=0,
                counter_arithmos=0,
                c_metabl=0,
                found_metabl=0,
                telos=0,
                flag_double_kai=0,
                flag_telos_metabl=0;

        char    Kai[5]="KAI",
                H[3]="H";

        AnsiString s=" ",temp_onoma;
        k4=i;                   //++++
        found_keno=1;
        for(k4;k4<=b;k4++)
        {
                if((k4==b) && A[k4]!=' ')
                {
                        Expl=" Kåío óôï ôåëïò";
                        Emfanise_error(Sender);
                        found_keno=0;
                }
        }
```

```
if(found_keno==1)
{
	if(A[i+1]!='(')
		fputs("(",pf);

	if(A[i+1]=='(')
		fputs("(",pf);

	for(i;i<=b;i++)
	{
		if(A[i]=='(')
		{
			flag_parenthesi=0;
			flag_xwris_parenthesi=1;
			c=0;
			arithm_parenthes++;
		}
		if(A[i]==')')
		{
			flag_parenthesi=1;
			arithm_parenthes--;
			if((i+1)<b)
				i++;
			fputs(")",pf);
		}
		flag_telos_metabl=1;
		if( A[i]==' ' || flag_parenthesi==0 || flag_xwris_parenthesi==0)
		{
			if(A[i]=='=' && A[i-1]!='>' && A[i-1]!='<')
				fputs("!",pf);
			if(A[i]=='\"')
				fputs("'",pf);
			if(A[i]=='<' && A[i+1]=='>')
			{
				fputs("==",pf);
				i=i+2;
			}
			else
				if(A[i]=='>' && A[i+1]=='=')
					fputs("<",pf);
			else
				if(A[i]=='<' && A[i+1]=='=')
					fputs(">",pf);
			else
				if(A[i]=='>')
					fputs("<=",pf);
			else
				if(A[i]=='<')
					fputs(">=",pf);
			else
```

```
                                if(A[i]!='=')
                                        metafrasi(Sender);

                        if(A[i]!='(' && A[i]!=' ' && A[i]!='<' && A[i]!='>' &&
A[i]!='=' && A[i]!='\" && A[i-1]!='\")
                                {
                                        onomametablitis[c_metabl]=A[i];
                                        mhkos2++;
                                        c_metabl++;
                                        flag_telos_metabl=0;
                                        if(A[i]>='0' && A[i]<='9')
                                                counter_arithmos++;
                                }
                        }

                        if(A[i]!=' ' && flag_parenthesi==1)
                        {
                                if(A[i]==Kai[c])
                                        sigkr_Kai++;
                                else
                                        if(A[i]=='H')
                                                fputs("&&",pf);
                                else
                                        if(A[i]!=')')
                                        {
                                                Expl=" KAI , H ";
                                                if(flag_double_kai==0)
                                                        Emfanise_error(Sender);
                                                flag_double_kai=1;
                                        }
                                c++;
                        }

                        if(sigkr_Kai==3)
                        {
                                fputs("||",pf);
                                sigkr_Kai=0;
                        }
                        if(A[i]=='(' || A[i]==')')
                        {
                                if((i+1)<=b)
                                        if(A[i+1]!=' ')
                                        {
                                                Expl=" Κενά ανάμεσα στις παρενθέσεις ";
                                                Emfanise_error(Sender);
                                        }
                        }

                        if(flag_telos_metabl==0 && A[i+1]==' ')
                        {
```

```
                                Testing=0;
                                c_metabl2=c_metabl;
                                elegxos_pinakon2(Sender);
                                if(found_pinakas_se_entolh==1)
                                {
                                        sigkrisi_pinakon(Sender);
                                        Found_metablhth=1;
                                        found_pinakas_se_entolh=0;
                                }
                                else
                                        sigkrisi_metabliton_2(Sender);
                                mhkos2=0;

                                if(Found_metablhth==0 &&
(counter_arithmos!=c_metabl))
                                {
                                        Expl=" Μεταβλητη ";
                                        Emfanise_error(Sender);
                                }

                                Found_metablhth=0;
                                c_metabl=0;
                                counter_arithmos=0;

                                for(cl=0;cl<40;cl++)
                                {
                                        onomametablitis[cl]=' ';
                                }
                                temp_onoma=" ";
                        }
                }

                fputs(")",pf);
                fputs(";",pf);
                if(i==b)
                {
                        fputs("\n",pf);
                }
                if(arithm_parenthes!=0)
                {
                        Expl=" παρενθεση ";
                        Emfanise_error(Sender);
                }
                flag_double_kai=0;
                flag_metablhth=0;
        }
}
//=============================================================
```

```
//***************************************************************************
//        LISTBOX1CLICK
//***************************************************************************
//Γράφει την εντολή που θα επιλέξει ο χρήστης από το ListBox στο Memo, όπου
γράφεται ο κώδικας.

void __fastcall TForm1::ListBox1Click(TObject *Sender)
{
        int       i=ListBox1->ItemIndex;

        AnsiString s=ListBox1->Items->Strings[i];

        Memo1->Lines->Insert(Memo1_y,s);
}
//===========================================================


//***************************************************************************
//        MEMO1MOUSEDOWN
//***************************************************************************
//Εμφανίζει τη γραμμή και τη στήλη του κέρσορα στο StatusBar.

void __fastcall TForm1::Memo1MouseDown(TObject *Sender,
    TMouseButton Button, TShiftState Shift, int X, int Y)
{

        int       x=Memo1->CaretPos.x;
                  int y=Memo1->CaretPos.y;

        AnsiString lines,col;

        lines=(y+1);
        col=x;
        Memo1_y=y;
        StatusBar1->SimpleText="   "+lines+": "+ col;
}
//===========================================================


//***************************************************************************
//        OPEN1CLICK
//***************************************************************************
//Χρησιμοποιείται από το πρόγραμμα για άνοιγμα αρχείων .txt.


void __fastcall TForm1::Open1Click(TObject *Sender)
{
        if(OpenDialog1->Execute())
        {
                Memo1->Lines->LoadFromFile(OpenDialog1->FileName);
```

```
        }
}
//=========================================================================


//*************************************************************************
//      EXIT1CLICK
//*************************************************************************
//Με αυτή τη συνάρτηση τερματίζεται η εφαρμογή.

void __fastcall TForm1::Exit1Click(TObject *Sender)
{
        Application->Terminate();
}
//=========================================================================


//*************************************************************************
//      NEW1CLICK
//*************************************************************************
//Επαναφέρει το πρόγραμμα στην αρχική του κατάσταση.

void __fastcall TForm1::New1Click(TObject *Sender)
{
        Memo1->Clear();
        if (FileExists("kodikas.c"))
                DeleteFile("kodikas.c");
        if (FileExists("kodikas.exe"))
                DeleteFile("kodikas.exe");
        Image1->Visible=true;
        Image3->Visible=false;
        ListBox2->Items->Clear();
        Button2->Enabled=false;
}
//=========================================================================


//*************************************************************************
//      SAVEAS1CLICK
//*************************************************************************
// Χρησιμοποιείται για την αποθήκευση του κώδικα DBN σε αρχείο .txt.

void __fastcall TForm1::SaveAs1Click(TObject *Sender)
{
        if(SaveDialog1->Execute())
        {
                Memo1->Lines->SaveToFile(SaveDialog1->FileName);
        }

}
```

```
//=============================================================


//*************************************************************
//        ΣΥΝΑΡΤΗΣΗ E M F A N I S E _ E R R O R
//*************************************************************
//Καλείται από διάφορες συναρτήσεις του προγράμματος, όταν εντοπιστεί
//συντακτικό λάθος, για την εμφάνιση του κατάλληλου μηνύματος στο  ListBox των
//μηνυμάτων λάθους.

void __fastcall TForm1::Emfanise_error(TObject *Sender)
{
        count_errors=count_debug+1;
        if(flag_giametabliti==1)
                ListBox2->Items->Strings[count_debug]="["+count_errors+"] "+" (
"+Expl+" )";
        else
                ListBox2->Items->Strings[count_debug]="["+count_errors+"]
"+Debug+(grammi+1)+Debug2+" ( "+Expl+" )";
        Expl=" ";
        pin_error[count_debug]=grammi;
        count_debug++;
        Error=1;
}
//=============================================================


//*************************************************************
//        MEMO1KEYUP
//*************************************************************
//Εμφανίζει τη γραμμή και τη στήλη που βρίσκεται ο κέρσορας στο StatusBar.

void __fastcall TForm1::Memo1KeyUp(TObject *Sender, WORD &Key,
    TShiftState Shift)
{
        int y=Memo1->CaretPos.y;
        int x=Memo1->CaretPos.x;

        AnsiString lines,col;

        lines=(y+1);
        col=x;
        StatusBar1->SimpleText="   "+lines+": "+ col;
}
//=============================================================


//*************************************************************
//        ΣΥΝΑΡΤΗΣΗ S I G K R I S I _  M E T A B L I T W N _ 2
//*************************************************************
```

```
//Συγκρίνει τις μεταβλητές των εντολών πλην της ΓΡΑΨΕ.

void __fastcall TForm1::sigkrisi_metabliton_2(TObject *Sender)
{
        int     g=0,
                q=1,
                length=0,
                isa=1;

        Found_metablhth=0;
        while(g<Synol_arith_metabl)
        {
                while(Pinakas_metablhton[g].Metablith[q]!=' ')
                {
                        if(Pinakas_metablhton[g].Metablith[q]==onomametablitis[q-
1])
                                length++;
                        else
                                isa=0;
                        q++;
                }
                if(length==0 && mhkos2==0)
                        isa=1;

                if(length==mhkos2 && isa==1)
                {
                        Found_metablhth=1;
                }
                g++;
                q=1;
                length=0;
                isa=1;
        }
        mhkos2=0;
}
//=============================================================


//*************************************************************************
//      BUTTON2CLICK
//*************************************************************************
//Μεταγλωττίζει και τρέχει τον κώδικα.

void __fastcall TForm1::Button2Click(TObject *Sender)
{
        system("bcc32 kodikas.c");
        Form1->Visible=false;
        system("kodikas.exe");
        Form1->Visible=true;
}
```

```
//=============================================================

//*************************************************************
//        ΣΥΝΑΡΤΗΣΗ A R X I K E S _ T I M E S
//*************************************************************
//Η συνάρτηση αυτή δίνει σε όλες τις Global μεταβλητές, τις αρχικές τους τιμές.

void __fastcall TForm1::Arxikes_times(TObject *Sender)
{
        int      j;

        Memo2->Clear();
        if (FileExists("kodikas.c"))
                DeleteFile("kodikas.c");
        if (FileExists("kodikas.exe"))
                DeleteFile("kodikas.exe");
        Image1->Visible=true;
        Image3->Visible=false;
        ListBox2->Items->Clear();
        grammi=0;
        a=0;
        p=0;
        ch=0;
        word[40]='0';
        for(j=0;j<40;j++)
        {
                onomametablitis[j]=' ';
        }
        Synol_arith_metabl=0;
        mikos_metablitis=-1;
        flag_metablhth_meta_grapse=0;
        flag_sunartish_print_bug=0;
        found_tupos_metablt=0;
        flag_gia_epilekse=0;
        flag_gia_periptosh=0;
        metritis_gia_telos_epilogon=0;
        metritis_switch=0;
        metritis_oso_kai_gia=0;
        metritis_telos_epanalhpshs=0;
        mhkos_arithmon=0;
        sigkr_Grapse=0;
        sigkr_Epilekse=0;
        sigkr_Diabase=0;
        allios=0;
        sigkr_Mexris_Otou=0;
        Memo1_y=0;
        kena=0;
        me_kena=0;
        Found_metablhth=0;
```

```
        Found_metablhth_grapse=1;
        mhkos2=0;
        count=0;
        count_debug=0;
        metr_autakia=0;
        Testing=0;
        m1=0;
        c_metabl2=0;
        counter=0;
        arithmos_agiles=0;
        flag_error_pinakas=0;
        flag_error_metablhtes=0;
        flag_error_ekxorhs=0;
        found_pinakas_se_entolh=0;
        flag_error_thesh_pinaka=1;
        Error=0;
        temp_grammh=0;
        temp_grammh_error=0;
        temp_i;
        grammi=0;
        i=0;
        flag_giametabliti=0;
        flag_euresis_matablitis=1;
        pm=0;
        metablhtes_count=0;
        Arith_pin_xarakt=0;
        count_w=0;
        omp=0;
        flag_metablhth=0;
        idio_onoma_pinaka=0;
        metritis_Telos_An=1;
        metritis_An=1;
        neo_programma=1;
        end_programmatos=1;
        found_metablites=0;
}
//===========================================================

//*************************************************************
//      E N T O L H _ E K X R O R H S H S
//*************************************************************
//Μεταφράζει την εντολή εκχώρησης (<-) και ελέγχει το συντακτικό της (π.χ.
//ονόματα μεταβλητών κ.λ.π.) και αν υπάρχει λάθος, εμφανίζει το κατάλληλο  μήνυμα
//λάθους, καλώντας τη συνάρτηση Emfanise_ error.

void __fastcall TForm1::entolh_ekxrorhshs(TObject *Sender)
{
        int     c_metabl=0,
                stop=0,
```

```
                lathos=0,
                j=0,
                ison=0,
                arith_metabl_Symbol=0,
                arith_parenth=0,
                arith_agiles=0,
                sunol_arith_metabl_Symbol=0,
                itsok=0,
                flag_arithmos=0;

        AnsiString onoma;
        mhkos2=0;
        for(i=1;i<=b;i++)
        {
                if(ison==0)
                        if(A[i]=='<' && A[i+1]=='-')
                        {
                                ison=1;
                                stop=1;
                                fputs(" =",pf);
                                i=i+2;
                        }
                if( (A[i]>='A' && A[i]<='Ω') ||(A[i]>='0' && A[i]<='9') || (A[i]>='α'
&& A[i]<='ω') || (A[i]=='[' || A[i]==']') || (A[i]>='a' && A[i]<='z') || (A[i]=='.'||A[i]=='
'||A[i]=='^'|| A[i]=='+' || A[i]=='-' || A[i]=='_' || A[i]=='/'||A[i]=='*'||A[i]=='('||A[i]==')'))
                {
                        // ok
                }
                else
                        if(ison==1)
                        {
                                Expl=" Συνταξη ";
                                Emfanise_error(Sender);
                        }
                if((A[i]>='A' && A[i]<='Ω') ||(A[i]>='α' && A[i]<='ω') || (A[i]>='a'
&& A[i]<='z') || (A[i]=='[' || A[i]==']') || (A[i]>='0' && A[i]<='9') || (A[i]=='_'))
                {
                        onomametablitis[c_metabl]=A[i];
                        mhkos2++;
                        c_metabl++;
                        if(i==b)
                        stop=1;
                }
                else
                        stop=1;
                if(stop==1)
                {
                        for(j=0;j<c_metabl;j++)
                        {
```

```
                        if((onomametablitis[j]>='0' &&
onomametablitis[j]<='9') || onomametablitis[j]=='.')
                                flag_arithmos=1;
                        else
                        {
                                flag_arithmos=0;
                                j=c_metabl;
                        }
                }

                if(flag_arithmos==0)
                {
                        c_metabl2=c_metabl;
                        elegxos_pinakon2(Sender);
                        if(found_pinakas_se_entolh==1)
                        {
                                Found_metablhth=1;
                                itsok=1;
                                sigkrisi_pinakon(Sender);
                                Memo2->Lines->Strings[0]=
found_pinakas_se_entolh;
                                Memo2->Lines->Strings[0]=Found_metablhth;
                        }
                        else
                        {
                                sigkrisi_metabliton_2(Sender);
                        }
                }
                c_metabl2=0;
                c_metabl=0;
                mhkos2=0;
                stop=0;
                flag_arithmos=0;
                for(j=0;j<40;j++)
                {
                        onomametablitis[j]=' ';
                }
                if(itsok==1 && flag_error_ekxorhs!=1)
                {
                        itsok=0;
                        Found_metablhth=1;
                }
                if(Found_metablhth==0 || flag_error_ekxorhs==1)
                {
                        Memo2->Lines->Strings[0]=flag_error_ekxorhs;
                        Expl=" Μεταβλητη ";
                        Emfanise_error(Sender);
                        flag_error_ekxorhs=0;
                }
                found_pinakas_se_entolh=0;
```

```
                }
                if(A[i]=='(')
                        arith_parenth++;
                if(A[i]==')')
                        arith_parenth--;
                if(A[i]=='[')
                        arith_agiles++;
                if(A[i]==']')
                        arith_agiles--;
                if(A[i]!=' ')
                        metafrasi(Sender);
        }
        if(arith_agiles!=0)
                arith_metabl_Symbol=2;
        if(arith_parenth!=0)
        {
                Expl=" παρενθεση ";
                Emfanise_error(Sender);
        }
        fputs(";",pf);
        fputs("\n",pf);
        ison=0;
        arith_agiles=0;
        arith_metabl_Symbol=0;
        flag_error_thesh_pinaka=1;
        flag_error_pinakas=0;
        flag_metablhth=0;
}
//=================================================================


//*****************************************************************
//      ΣΥΝΑΡΤΗΣΗ E L E G X O S _ P I N A K O N _ 2
//*****************************************************************
//Ελέγχει το συντακτικό των πινάκων που χρησιμοποιούν όλες οι εντολές του
//προγράμματος.


void __fastcall TForm1::elegxos_pinakon2(TObject *Sender)
{
        flag_metablhth=0;
        found_pinakas_se_entolh=0;


        for(m1=0;m1<c_metabl2;m1++)
        {
                if(onomametablitis[m1]=='[')
                {
                        found_pinakas_se_entolh=1;
                        counter=m1;
```

```
                        arithmos_agiles++;
                        if(onomametablitis[m1+1]==']' || (onomametablitis[m1+1]=='0'
&& onomametablitis[m1+2]!=']'))
                        {
                                flag_error_pinakas=1;
                        }
                        while(counter<c_metabl2)
                        {
                                counter++;
                                if(onomametablitis[counter]==']')
                                        arithmos_agiles--;
                                if(onomametablitis[counter]=='[')
                                        arithmos_agiles++;
                                if(onomametablitis[counter]!=']')
                                {
                                        onoma_metabl_se_pinaka[omp]=
onomametablitis[counter];
                                        omp++;
                                        if((onomametablitis[counter]>='0' &&
onomametablitis[counter]<='9') || (onomametablitis[counter]>='α' &&
onomametablitis[counter]<='ω'))
                                        {
                                                if(onomametablitis[counter]>='α' &&
onomametablitis[counter]<='ω')
                                                        flag_metablhth=1;
                                        }
                                        else
                                        {
                                                flag_error_pinakas=1;
                                        }
                                }
                                else
                                {
                                        counter=c_metabl2;
                                        m1=c_metabl2;
                                        if(flag_error_pinakas==0)
                                                found_pinakas_se_entolh=1;
                                }
                        }
                }
        }
        if(flag_metablhth==1)
        {
                sigkrisi_metabliton_3(Sender);
        }
        for(omp=0;omp<40;omp++)
                onoma_metabl_se_pinaka[omp]=' ';
        omp=0;
}
//=======================================================
```

```
//*****************************************************************
//        ΣΥΝΑΡΤΗΣΗ  E L E G X O S _ P I N A K O N
//*****************************************************************
//Ελέγχει τα ονόματα των δηλωμένων μεταβλητών και εάν υπάρχει πίνακας ελέγχει
//τη σύνταξη του.

void __fastcall TForm1::elegxos_pinakon(TObject *Sender)
{
        if(A[i]=='[')
        {
                counter=i;
                arithmos_agiles++;
                if(A[i+1]==']' || (A[i+1]=='0' && A[i+2]!=']'))
                {
                        flag_error_metablhtes=1;
                }
                 while(counter<b && A[counter]!=',')
                 {
                        counter++;
                        if(A[counter]==']')
                                arithmos_agiles--;
                        if(A[counter]=='[')
                                arithmos_agiles++;
                        if(A[counter]!=']')
                        {
                                if(A[counter]>='0' && A[counter]<='9')
                                {
                                        //ok
                                }
                                else
                                {
                                        flag_error_metablhtes=1;
                                }
                        }
                        else
                                if(counter!=b)
                                        if(A[counter+1]==',')
                                                counter=b;
                    }
            }
}
//=================================================================


//*****************************************************************
//        ΣΥΝΑΡΤΗΣΗ S I G K R I S I _ P I N A K O N
//*****************************************************************
// Συγκρίνει το όνομα του πίνακα της εντολής με αυτό που υπάρχει δηλωμένο. Επίσης
```

```
//συγκρίνει και τις θέσεις του πίνακα. Η θέση στην οποία αναφέρεται η εντολή δεν
//μπορεί να ξεπερνάει τις θέσεις του πίνακα.

void __fastcall TForm1::sigkrisi_pinakon(TObject *Sender)
{
        int     g=0,
                q=1,
                length=0,
                telos=0,
                flag_sosto=0,
                isa=0,
                sosto=0,
                error;
        sosto=0;
        idio_onoma_pinaka=0;
        flag_error_thesh_pinaka=0;
        while(g<Synol_arith_metabl)
        {
                while(Pinakas_metablhton[g].Metablith[q]!=' ')
                {
                        if(Pinakas_metablhton[g].Metablith[q]==onomametablitis[q-
1])
                                length++;
                        if(onomametablitis[q-1]=='[' && (length==q))
                        {
                                idio_onoma_pinaka=1;
                                if(flag_metablhth==0)
                                {
                                        while(onomametablitis[q-1]!=']')
                                        {
                                                if(onomametablitis[q-1]>='0' &&
onomametablitis[q-1]<='9')
                                                {
                                                        if(Pinakas_metablhton[g].
Metablith[q]==']')
                                                        {
                                                                flag_error_thesh_pinaka=1;
                                                                error=1;
                                                        }
                                                        if(onomametablitis[q]==']' &&
Pinakas_metablhton[g].Metablith[q+1]!=']')
                                                        {
                                                                flag_sosto=1;
                                                        }
                                                        if(onomametablitis[q-
1]<Pinakas_metablhton[g].Metablith[q])
                                                        {
                                                                flag_sosto=1;

                                                                if(onomametablitis[q]==']')
```

```
                                                    if(onomametablitis[q-
1]>=Pinakas_metablhton[g].Metablith[q])
                                                    {
                                                        error=1;
                                                    }
                                                    else
                                                    sosto++;
                                                }
                                                else
                                                    if(onomametablitis[q-
1]==Pinakas_metablhton[g].Metablith[q] && onomametablitis[q]!=']')
                                                    {
                                                        //ok
                                                    }
                                                    else
                                                    {
                                                        error=1;
                                                    }
                                            }
                                            q++;
                                        }
                                    }
                                }
                                q++;
                            }
                            g++;
                            q=1;
                            length=0;
                        }
                        if(flag_sosto==0)
                        if(error==1)
                        {
                            Expl="Πινακας";
                            Emfanise_error(Sender);
                        }
                        if(flag_error_thesh_pinaka==1 || idio_onoma_pinaka==0)
                        {
                            Expl="Πινακας";
                            Emfanise_error(Sender);
                        }
                        error=0;
                        sosto=0;
}
//============================================================


//*************************************************************
//      ABOUT1CLICK
//*************************************************************
//Εμφανίζει το AboutBox της φόρμας.
```

```
void __fastcall TForm1::About1Click(TObject *Sender)
{
        AboutBox->ShowModal();
}
//=========================================================================


//*************************************************************************
//        HELP2CLICK
//*************************************************************************
//Εμφανίζει το αρχείο Βοήθειας του προγράμματος.

void __fastcall TForm1::Help2Click(TObject *Sender)
{
        Application->HelpFile = "HFile2.HLP";
        Application->HelpCommand(HELP_CONTENTS, 0);
}
//=========================================================================


//*************************************************************************
//        MEMO1CHANGE
//*************************************************************************
//Όταν γίνει μια αλλαγή στο Memo, τότε απενεργοποιείται το κουμπί Εκτέλεση.

void __fastcall TForm1::Memo1Change(TObject *Sender)
{
        Button2->Enabled=false;
        Image1->Visible=true;
}
//=========================================================================


//*************************************************************************
//        ΣΥΝΑΡΤΗΣΗ SIGKRISI_METABLITWN 3
//*************************************************************************
//Συγκρίνει τους δείκτες των πινάκων, σε περίπτωση που δεν είναι αριθμοί αλλά
//μεταβλητές, με τις δηλωμένες μεταβλητές. Αν για παράδειγμα έχουμε Pin[j]
//συγκρίνει το j με τις δηλωμένες μεταβλητές για να δει αν υπάρχει.

void __fastcall TForm1::sigkrisi_metabliton_3(TObject *Sender)
{
        int     g=0,
                q=1,
                length=0,
                isa=1;

        Found_metablhth=0;
        while(g<Synol_arith_metabl)
```

```
        {
                while(Pinakas_metablhton[g].Metablith[q]!=' ')
                {
                        if(Pinakas_metablhton[g].Metablith[q]=
=onoma_metabl_se_pinaka[q-1])
                                length++;
                        else
                                isa=0;
                        q++;
                }
                if(length==omp)
                {
                        Found_metablhth=1;
                }
                g++;
                q=1;
                length=0;
                isa=1;
        }
        if(Found_metablhth==0)
        {
                Expl="Πινακας - Μεταβλητη";
                Emfanise_error(Sender);
        }
        omp=0;
}
//---------------------------------------------------------------------------


//***************************************************************************
//      FORMKEYDOWN
//***************************************************************************
//Καλεί τις συναρτήσεις ανάλογα με το πλήκτρο συντόμευσης που έχει πατηθει.F8
//για Έλεγχο και F9 για Εκτέλεση.


void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key,
    TShiftState Shift)
{
        if(Key==VK_F8)
                Button1Click(Sender);
        if(Key==VK_F9 && Button2->Enabled==true)
                Button2Click(Sender);
}
//---------------------------------------------------------------------------


//***************************************************************************
//      UNDO1CLICK
//***************************************************************************
```

```
//Κάνει αναίρεση (Undo), και επαναφέρει το Memo στην προηγούμενη κατάσταση
//του.

void __fastcall TForm1::Undo1Click(TObject *Sender)
{
        Memo1->SetFocus();
        Memo1->Undo();
}
//=====================================================================


//**********************************************************************
//      CUT1CLICK
//**********************************************************************
//Κάνει αποκοπή του επιλεγμένου κειμένου.

void __fastcall TForm1::Cut1Click(TObject *Sender)
{
        Memo1->CutToClipboard();
}
//=====================================================================


//**********************************************************************
//      COPY1CLICK
//**********************************************************************
//Κάνει αντιγραφή του επιλεγμένου κειμένου.

void __fastcall TForm1::Copy1Click(TObject *Sender)
{
        Memo1->CopyToClipboard();
}
//=====================================================================


//**********************************************************************
//      PASTE1CLICK
//**********************************************************************
//Κάνει επικόλληση, εφόσον πρίν έχει ακολουθήσει αντιγραφή ή αποκοπή.

void __fastcall TForm1::Paste1Click(TObject *Sender)
{
        Memo1->PasteFromClipboard();
}
//=====================================================================


//**********************************************************************
//      SELECTALL1CLICK
//**********************************************************************
```

```
// Μαρκάρει όλο το κείμενο του Memo.

void __fastcall TForm1::SelectAll1Click(TObject *Sender)
{
        Memo1->SetFocus();
        Memo1->SelectAll();
}
//=============================================================


//*************************************************************************
//        DELETE1CLICK
//*************************************************************************
//Διαγράφει το επιλεγμένο κείμενο του Memo.

void __fastcall TForm1::Delete1Click(TObject *Sender)
{
        Memo1->ClearSelection();
}
//=============================================================


//*************************************************************************
//  ΣΥΝΑΡΤΗΣΗ A N
//*************************************************************************
//Μεταφράζει την εντολή AN και ελέγχει το συντακτικό της (π.χ. ονόματα
//μεταβλητών κ.λ.π.) και αν υπάρχει λάθος, εμφανίζει το κατάλληλο  μήνυμα λάθους,
//καλώντας τη συνάρτηση Emfanise_ error.

void _fastcall TForm1::synartisi_An(TObject *Sender)
{
        int     dok=0,
                kl=0,
                Num=0,
                prosorino_mhkos=0,
                found_pinakas=0;
        for(i;i<=b;i++)
        {
                if(A[i]!=' ' && A[i]!='(' && A[i]!=')' && (A[i]!='=' || A[i]!='<' ||
A[i]=='>'))
                        if(A[i]=='=')
                        {
                                fputs("==",pf);
                                for(kl=0;kl<=mhkos2;kl++)
                                {
                                        if(onomametablitis[kl]=='[')
                        found_pinakas=1;
                                                if(onomametablitis[kl]>='0' &&
onomametablitis[kl]<='9' || onomametablitis[kl]=='.')
                                                {
```

```
                                        prosorino_mhkos++;
                                }
                        }
                        if(prosorino_mhkos==mhkos2 || found_pinakas==1)
                                Num=0;
                        else
                                Num=1;
                        if(Num==1)
                        {
                                sigkrisi_metabliton_2(Sender);
                                if(Found_metablhth==0)
                                {
                                        Expl="αδυλωτη μεταβλητη";
                                        Emfanise_error(Sender);
                                }
                        }

                        for(kl=0;kl<40;kl++)
                        {
                                onomametablitis[kl]=' ';
                        }
                        mhkos2=0;
                        i++;
                        Num=0;
                        prosorino_mhkos=0;
                }

                if(A[i]=='<' && A[i+1]=='>')
                {
                        fputs("!=",pf);
                        for(kl=0;kl<=mhkos2;kl++)
                        {
                                if(onomametablitis[kl]=='[')
                                        found_pinakas=1;
                                if(onomametablitis[kl]>='0' &&
onomametablitis[kl]<='9' || onomametablitis[kl]=='.')
                                {
                                        prosorino_mhkos++;
                                }
                        }
                        if(prosorino_mhkos==mhkos2 || found_pinakas==1)
                                Num=0;
                        else
                                Num=1;
                        if(Num==1)
                        {
                                sigkrisi_metabliton_2(Sender);
                                if(Found_metablhth==0)
                                {
                                        Expl="αδυλωτη μεταβλητη";
```

```
                                        Emfanise_error(Sender);
                                }
                        }

                        for(kl=0;kl<40;kl++)
                        {
                                onomametablitis[kl]=' ';
                        }
                        mhkos2=0;
                        i=i+2;
                        Num=0;
                        prosorino_mhkos=0;
                }
                if(A[i]=='<'&& A[i+1]!='>')
                {
                        metafrasi(Sender);
                        for(kl=0;kl<=mhkos2;kl++)
                        {
                                if(onomametablitis[kl]=='[')
                                        found_pinakas=1;
                                if(onomametablitis[kl]>='0' &&
onomametablitis[kl]<='9' || onomametablitis[kl]=='.')
                                {
                                        prosorino_mhkos++;
                                }
                        }
                        if(prosorino_mhkos==mhkos2 || found_pinakas==1)
                                Num=0;
                        else
                                Num=1;
                        if(Num==1)
                        {
                                sigkrisi_metabliton_2(Sender);
                                if(Found_metablhth==0)
                                {
                                        Expl="αδυλωτη μεταβλητη";
                                        Emfanise_error(Sender);
                                }
                        }
                        for(kl=0;kl<40;kl++)
                        {
                                onomametablitis[kl]=' ';
                        }
                        mhkos2=0;
                        i++;
                        Num=0;
                        prosorino_mhkos=0;
                }
                if(A[i]=='>')
                {
```

```
                        metafrasi(Sender);
                        for(kl=0;kl<=mhkos2;kl++)
                        {
                                if(onomametablitis[kl]=='[')
                                        found_pinakas=1;
                                if(onomametablitis[kl]>='0' &&
onomametablitis[kl]<='9' || onomametablitis[kl]=='.')
                                {
                                        prosorino_mhkos++;
                                }
                        }
                        if(prosorino_mhkos==mhkos2 || found_pinakas==1)
                                Num=0;
                        else
                                Num=1;
                        if(Num==1)
                        {
                                sigkrisi_metabliton_2(Sender);
                                if(Found_metablhth==0)
                                {
                                        Expl="αδυλωτη μεταβλητη";
                                        Emfanise_error(Sender);
                                }
                        }
                        for(kl=0;kl<40;kl++)
                        {
                                onomametablitis[kl]=' ';
                        }
                        mhkos2=0;
                        i++;
                        Num=0;
                        prosorino_mhkos=0;
                }
                if(A[i]=='K' && A[i+1]=='A' && A[i+2]=='I' && A[i+3]==' ')
                {
                        fputs(" && ",pf);
                        for(kl=0;kl<=mhkos2;kl++)
                        {
                                if(onomametablitis[kl]=='[')
                                        found_pinakas=1;
                                if(onomametablitis[kl]>='0' &&
onomametablitis[kl]<='9' || onomametablitis[kl]=='.')
                                {
                                        prosorino_mhkos++;
                                }
                        }
                        if(prosorino_mhkos==mhkos2 || found_pinakas==1)
                                Num=0;
                        else
                                Num=1;
```

```
                        if(Num==1)
                        {
                                sigkrisi_metabliton_2(Sender);
                                if(Found_metablhth==0)
                                {
                                        Expl="αδυλωτη μεταβλητη";
                                        Emfanise_error(Sender);
                                }
                        }
                        for(kl=0;kl<40;kl++)
                        {
                                onomametablitis[kl]=' ';
                        }
                        mhkos2=0;
                        i=i+3;
                        Num=0;
                        prosorino_mhkos=0;
                }
                if(A[i]=='H' && A[i-1]==' ' && A[i+1]==' ')
                {
                        fputs(" || ",pf);
                        for(kl=0;kl<=mhkos2;kl++)
                        {
                                if(onomametablitis[kl]=='[')
                                        found_pinakas=1;
                                if(onomametablitis[kl]>='0' &&
onomametablitis[kl]<='9' || onomametablitis[kl]=='.')
                                {
                                        prosorino_mhkos++;
                                }
                        }
                        if(prosorino_mhkos==mhkos2 || found_pinakas==1)
                                Num=0;
                        else
                                Num=1;
                        if(Num==1)
                        {
                                sigkrisi_metabliton_2(Sender);
                                if(Found_metablhth==0)
                                {
                                        Expl="αδυλωτη μεταβλητη";
                                        Emfanise_error(Sender);
                                }
                        }
                        for(kl=0;kl<40;kl++)
                        {
                                onomametablitis[kl]=' ';
                        }
                        mhkos2=0;
                        i++;
```

```
                        Num=0;
                        prosorino_mhkos=0;
                }
                if(A[i]=='T' && A[i-1]==' ' && A[i+1]=='O' && A[i+2]=='T'
&& A[i+3]=='E')
                {
                        i=b;
                        fputs(")\n{\n",pf);
                        for(kl=0;kl<=mhkos2;kl++)
                        {
                                if(onomametablitis[kl]=='[')
                                        found_pinakas=1;
                                if(onomametablitis[kl]>='0' &&
onomametablitis[kl]<='9' || onomametablitis[kl]=='.')
                                {

                                        prosorino_mhkos++;
                                }
                        }
                        if(prosorino_mhkos==mhkos2 || found_pinakas==1)
                                Num=0;
                        else
                                Num=1;
                        if(Num==1)
                        {
                                sigkrisi_metabliton_2(Sender);
                                if(Found_metablhth==0)
                                {
                                        Expl="αδυλωτη μεταβλητη";
                                        Emfanise_error(Sender);
                                }
                        }
                        for(kl=0;kl<40;kl++)
                        {
                                onomametablitis[kl]=' ';
                        }
                        mhkos2=0;
                        Num=0;
                        prosorino_mhkos=0;
                }
                else
                {
                        metafrasi(Sender);
                        if(A[i]!='(' && A[i]!=')'&& A[i]!=' '&& A[i]!='=')
                        {
                                onomametablitis[mhkos2]=A[i];
                                mhkos2++;
                        }
                }
        }
```

```
}
//=============================================================

//*************************************************************
//        ΣΥΝΑΡΤΗΣΗ  G I A
//*************************************************************
//Μεταφράζει την εντολή ΓΙΑ..ΑΠΟ..ΜΕΧΡΙ…ΜΕ ΒΗΜΑ, και ελέγχει το
//συντακτικό της (π.χ. ονόματα μεταβλητών κ.λ.π.) και αν υπάρχει λάθος, εμφανίζει
//το κατάλληλο  μήνυμα λάθους, καλώντας τη συνάρτηση Emfanise_ error.

void _fastcall TForm1::synartisi_Gia(TObject *Sender)
{
        int     found1=0,
                found2=0,
                found3=0,
                d=0,
                arnitiko_bhma=0,
                c=0,
                x=0,
                arxi=0,
                arxi_metablitis=0,
                dok=0,
                telos_metablitis=0,
                ps=0,
                k=0,
                fpros=0,
                thesiM=0,
                megalytero=0,
                prosimo=0,
                l=0,
                ps1=0,
                megalytero=2,
                kl=0,
                metr=0,
                aritmos=0,
                metabl=0,
                prosorino_mhkos=0,
                Num=0,
                NotNum=0;

        char    metabliti1[20],
                metabliti2[20],
                metabliti3[20],
                metabliti4[20];

        dok=i;
        for(dok;dok<=b;dok++)
        {
```

```
                if(A[dok]=='M' && A[dok+1]=='E' && A[dok+2]==' ' &&
A[dok+3]=='B' && A[dok+4]== 'H' && A[dok+5]=='M' && A[dok+6]=='A')
                {
                        dok=dok+7;
                        for(dok;dok<=b;dok++)
                        {
                                if(A[dok]!=' ' && fmegalytero==0)
                                {
                                        if(A[dok]=='-')
                                        {
                                                fmegalytero=1;
                                                megalytero=1;
                                        }
                                        else
                                        {
                                                fmegalytero=1;
                                                megalytero=0;
                                        }
                                }
                        }
                }
        }
        for(i;i<=b;i++)
        {
                for(i;i<=b;i++)
                {
                        if(A[i]=='M' && found1==1 && l==1)
                        {
                                thesiM=i;
                        }
                        if((A[i]!=' ' && A[i]!='A' || A[i+1]!='Π' || A[i+2]!='O' ||
A[i+3]!=' ') && found1==0)
                        {
                                if(arxi==0)
                                {
                                        arxi=1;
                                        arxi_metablitis=i;
                                }
                                if(A[i]!=' ')
                                {
                                        onomametablitis[mhkos2]=A[i];
                                        mhkos2++;
                                }
                                metafrasi(Sender);
                        }
                        else
                        if(A[i]=='A' && A[i+1]=='Π' && A[i+2]=='O' && A[i+3]==' '
                        && found1==0)
                        {
                                sigkrisi_metabliton_2(Sender);
```

```
                        if(Found_metablhth==0)
                        {
                                Expl="αδυλωτη μεταβλητη";
                                Emfanise_error(Sender);
                        }
                        for(kl=0;kl<40;kl++)
                        {
                                onomametablitis[kl]=' ';
                        }
                        fputs("=",pf);
                        found1=1;
                        telos_metablitis=i-1;
                        i=i+3;
                }
                if(A[i]!=' ' && (A[i]!='M' && A[i+1]!='E' && A[i+2]!='X' &&
A[i+3]!='P' && A[i+4]!='I' ) && found2==0 && found1==1)
                {
                        metafrasi(Sender);
                        if(A[i]>='0' && A[i]<='9' || A[i]=='-')
                                Num=1;
                        else
                        NotNum=1;
                }
                else
                if(A[i]=='M' && A[i+1]=='E' && A[i+2]=='X' &&
            A[i+3]=='P' && A[i+4]=='I' && A[i+5]==' ' && found2==0)
                {
                        if(NotNum==1)
                        {
                                Expl="μονο νουμερο μετα το ΑΠΟ";
                                Emfanise_error(Sender);
                                NotNum=0;
                        }
                        fputs(";",pf);
                        for(c=arxi_metablitis;c<=
telos_metablitis-1;c++)
                        {
                                A[i]=A[c];
                                metafrasi(Sender);
                        }
                        found2=1;
                        l=1;
                        if(megalytero==1)
                                fputs(">=",pf);
                        else
                        if(megalytero==0)
                                fputs("<=",pf);
                                i=i+5;
                        }
```

```
                        if(A[i]!='M' && A[i+1]!='E' && A[i+2]!=' ' &&
A[i+3]!='B' && A[i+4]!= 'H' && A[i+5]!='M' && A[i+6]!='A' && found3==0 &&
found1==1 && found2==1)
                        {
                                metafrasi(Sender);
                                found2=0;
                                if(A[i]>='0' && A[i]<='9' || A[i]=='-' || A[i]==' ')
                                        Num=1;
                                else
                                        NotNum=1;
                        }
                        else
                        if(A[i]=='M' && i==thesiM)
                        {
                                if(A[i+1]!='E' || A[i+2]!=' ' || A[i+3]!='B' ||
A[i+4]!='H' || A[i+5]!='M' || A[i+6]!='A' )
                                {
                                        Expl="lathos sth sintaksi toy me bhma";
                                        Emfanise_error(Sender);
                                }
                                found3=1;
                                if(NotNum==1)
                                {
                                        Expl="μονο νουμερο μετα το MEXPI";
                                        Emfanise_error(Sender);
                                        NotNum=0;
                                }
                                fputs(";",pf);
                                for(c=arxi_metablitis;c<=
telos_metablitis-1;c++)
                                {
                                        A[i]=A[c];
                                        metafrasi(Sender);
                                }
                                fputs("=",pf);
                                for(c=arxi_metablitis;c<=
telos_metablitis-1;c++)
                                {
                                        A[i]=A[c];
                                        metafrasi(Sender);
                                }
                                k=i+7;
                                for(k;k<=b;k++)
                                {
                                        if(A[k]!=' ' && fpros==0)
                                        {
                                                if(A[k]=='-')
                                                {
                                                        prosimo=1;
                                                        fpros=1;
```

```
                                        }
                                else
                                {
                                        prosimo=0;
                                        fpros=1;
                                }
                        }
                }
                if(prosimo==0 && fpros==1)
                {
                        fputs("+",pf);
                        i=i+8;
                }
                else
                if(prosimo==1  && fpros==1)
                {
                        fputs("-",pf);
                        i=i+9;
                }
                for(i;i<=b;i++)
                {
                        if(A[i]>='0' && A[i]<='9' || A[i]=='-')
                                Num=1;
                        else
                                NotNum=1;
                        metafrasi(Sender);
                }
                if(NotNum==1)
                {
                        Expl="μονο νουμερο μετα το ΜΕ ΒΗΜΑ";
                        Emfanise_error(Sender);
                        NotNum=0;
                }
        }
    }
    fputs(")\n{",pf);
    if(found1==0)
    {
        Expl="ΛΑΘΟΣ ΣΥΝΤΑΞΗ ΣΤΗ ΛΕΞΗ ΑΠΟ";
        Emfanise_error(Sender);
    }
    if(l==0)
    {
        Expl="ΛΑΘΟΣ ΣΥΝΤΑΞΗ ΣΤΗ ΛΕΞΗ ΜΕΧΡΙ";
        Emfanise_error(Sender);
    }
  }
}
//=============================================================
```

6. ΒΙΒΛΙΟΓΡΑΦΙΑ – ΠΗΓΕΣ

1.Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον, Α. Βάκαλη, Η. Γιαννόπουλος, Ν. Ιωαννίδης, Χ. Κοίλιας, Κ. Μάλαμας, Ι. Μανωλόπουλος, Π. Πολίτης.
2.Πληροφορική το παρόν και το μέλλον, Charles S. Parker, Τόμος 2, εκδόσεις Ι.Φλώρος.
3.Μεταγλωττιστές Γλωσσών Προγραμματισμού, Κ. Λαζός, Π. Κατσαρός, Ζ. Καραΐσκος, Θεσσαλονίκη 2003.
4.Οπτικός Προγραμματισμός, Καζαρλής Σπύρος, Τ.Ε.Ι. Σερρών 2004.