ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Paschool

Εκπαιδευτική Γλώσσα Προγραμματισμού για το Γυμνάσιο

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ιωάννης Ν. Κορκόντζελος

Επιβλέπων : Ευστάθιος Ζάχος
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2005

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Paschool

Εκπαιδευτική Γλώσσα Προγραμματισμού για το Γυμνάσιο

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ιωάννης Ν. Κορκόντζελος

Επιβλέπων : Ευστάθιος Ζάχος
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 18η Ιουλίου 2005.

.....	.....	.....
Ευστάθιος Ζάχος, Καθηγητής Ε.Μ.Π.	Νικόλαος Παπασπύρου, Λέκτορας Ε.Μ.Π.	Τιμολέων Σελλής, Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2005

.....

Ιωάννης Ν. Κορκόντζελος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Πρόλογος

Η παρούσα διπλωματική εργασία αποτελεί την πρώτη προσπάθεια συγγραφής ενός διδακτικού βοηθήματος, που εισάγει την εκπαιδευτική γλώσσα προγραμματισμού Paschool I στους μαθητές της Α΄ τάξης του Γυμνασίου και την προτείνει ως χρήσιμο εργαλείο για την κατανόηση και εμπέδωση των μαθηματικών εννοιών.

Η γλώσσα προγραμματισμού Paschool κατασκευάστηκε από την ομάδα μελέτης της πληροφορικής στην εκπαίδευση που δημιουργήθηκε στο Εθνικό Μετσόβιο Πολυτεχνείο, στη σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών κατά την ακαδημαϊκή περίοδο 2003 – 2004, στα πλαίσια του εργαστηρίου λογισμικού (Softlab) και του εργαστηρίου Λογικής και Επιστήμης Υπολογισμών (CoReLab). Έχουν κατασκευαστεί τρεις εκδόσεις της Paschool. Η πρώτη από αυτές, η Paschool I, έχει μικρές δυνατότητες, έτσι ώστε να είναι απλή και κατανοητή. Σε κάθε επόμενη έκδοση της γλώσσας προστίθενται βαθμιαία επιπλέον χαρακτηριστικά.

Το διδακτικό βοήθημα διατηρεί στενή σχέση με τη ροή της διδακτέας ύλης του μαθήματος των μαθηματικών. Παράλληλα, στην αρχή κάθε κεφάλαιου και όπου αλλού κρίνεται απαραίτητο εισάγονται και επεξηγούνται όλα τα χαρακτηριστικά της Paschool I. Βασική αρχή του βοηθήματος είναι η παρουσίαση πολλών ολοκληρωμένων παραδειγμάτων σε συνδυασμό με έγχρωμα πλαίσια και σκίτσα για τον εκτελέσιμο κώδικα, τις ασκήσεις και τα βασικότερα τμήματα της θεωρίας.

Τέλος, θα ήθελα να ευχαριστήσω όλους όσους συνέβαλαν στην εκπόνηση της διπλωματικής μου εργασίας, παρέχοντας πολύτιμη βοήθεια, συμβουλές και εποικοδομητική κριτική.

Ιωάννης Κορκόντζελος,

Ιούλιος 2005

Preface

The present dissertation constitutes the first effort of writing a school book, that introduces the educational programming language Paschool to the students of A class of junior high school and proposes it as a useful tool for the comprehension and strengthening of mathematics.

The programming language Paschool was manufactured by the team of study of Educational Informatics that was created in the National Technical University of Athens, in the faculty of Electrical and Computer Engineering, during the academic period 2003 – 2004, in the frames of the software laboratory (Softlab) and the computation and reasoning laboratory (CoReLab). There have been created three versions of Paschool. Paschool I has limited functionality, so as to be simple and comprehensible. In each next publication, more characteristics are added gradually.

The school book maintains narrow relation with the book of mathematics. At the same time, in the beginning of each chapter and wherever it is judged essential all the characteristics of Paschool are imported and are explained. Basic principle of the book is the presentation of many complete examples in combination with coloured frames and sketches for the executable code, the exercises and the fundamental parts of theory.

Finally, I would want to thank everyone that contributed to the development of my dissertation, providing precious help, pieces of advice and constructive criticism.

Ioannis Korkontzelos,

July 2005

Περιεχόμενα

Εισαγωγή

Πληροφορική και Εκπαίδευση.

Ο 21ος αιώνας χαρακτηρίζεται ως «Η Εποχή των Υπολογιστών», λόγω της ευρείας διάδοσης τους. Οι υπολογιστές χρησιμοποιούνται όχι μόνο ως επαγγελματικό εργαλείο αλλά και ως κέντρο επικοινωνίας και ψυχαγωγίας. Οι γνώσεις χειρισμού βασικών εφαρμογών ηλεκτρονικών υπολογιστών θεωρούνται απαραίτητο προσόν για τους περισσότερους εργαζόμενους.

Αντίθετα με τις παραπάνω τάσεις, η μέση εκπαίδευση, σε γενικές γραμμές, αποτυγχάνει να εφοδιάσει τους νέους με χρήσιμες γνώσεις σχετικά με την τεχνολογία και των τρόπο λειτουργίας των ηλεκτρονικών υπολογιστών. Στα περισσότερα σχολεία είτε δεν υπάρχουν υπολογιστές διαθέσιμοι στους μαθητές, είτε τα εργαστήρια υπολογιστών υπολειτουργούν λόγω παλαιότητας του εξοπλισμού τους. Επιπλέον, σε πολύ λίγα σχολεία εργάζονται εξειδικευμένοι καθηγητές, ενώ κατά κανόνα τη διδασκαλία του μαθήματος της πληροφορικής αναλαμβάνουν καθηγητές άλλων ειδικοτήτων, έχοντας παρακολουθήσει ολιγοήμερα εκπαιδευτικά σεμινάρια. Εξαιτίας αυτής της κατάστασης, συνήθως οι καθηγητές του μαθήματος της πληροφορικής αδυνατούν να διδάξουν το μάθημα, με αποτέλεσμα οι μαθητές να ασχολούνται με ηλεκτρονικά παιχνίδια και περιήγηση στο διαδίκτυο, κατά τη διάρκεια του μαθήματος και το μάθημα να υποβαθμίζεται.

Κατά την ακαδημαϊκή περίοδο 2003 – 2004, με πρωτοβουλία του καθηγητή του Εθνικού Μετσόβιου Πολυτεχνείου κ. Ζάχου και ουσιαστική συμβολή του λέκτορα κ. Παπασπύρου, δημιουργήθηκε μία ομάδα μελέτης της πληροφορικής στην εκπαίδευση. Μελέτησε τον τρόπο με τον οποίο διδάσκεται η πληροφορική στο εξωτερικό καθώς και τις εξής εκπαιδευτικές γλώσσες προγραμματισμού: Pascal, ABC, Blue, Logo, Python, Squeak και Turing. Επίσης, μελέτησε τα χαρακτηριστικά των γλωσσών Ada, BETA, C++, C-sharp, Component_Pascal, Delphi, Java, Modula-2, Modula-3 και Smalltalk, με σκοπό να επιλέξει ποια από αυτά θα ήταν χρήσιμο να διδαχθούν σε μαθητές της μέσης εκπαίδευσης.

Κατέληξε στο συμπέρασμα ότι οι μαθητές θα έπρεπε να διδάσκονται από τις πρώτες τάξεις του Γυμνασίου βασικές αρχές του προγραμματισμού. Λόγω της στενής σχέσης του προγραμματισμού με τα μαθηματικά, ένα τέτοιο μάθημα θα τους έδινε τη δυνατότητα να κατανοήσουν ευκολότερα τις μαθηματικές έννοιες. Πιο συγκεκριμένα, κατασκευάζοντας απλά προγράμματα σχετικά με κάθε μαθηματική έννοια, οι μαθητές θα μπορέσουν να αντιληφθούν καλύτερα την πρακτική διάστασή της.

Για αυτό το σκοπό, η ομάδα ανέπτυξε μία απλή εκπαιδευτική γλώσσα προγραμματισμού, που ονομάστηκε Paschool. Η γλώσσα αυτή διέπεται από τα χαρακτηριστικά που κρίθηκαν απαραίτητα για τη διδασκαλία της σε μαθητές της μέσης εκπαίδευσης.

Η Γλώσσα Προγραμματισμού Paschool.

Υποστηρίζει την τεχνική της βαθμιαίας συγκεκριμενοποίησης (stepwise refinement), δηλαδή παρέχει στον προγραμματιστή τη δυνατότητα να σχεδιάσει αρχικά το πρόγραμμά του, αναλύοντας το σε μεγάλες ενότητες και στη συνέχεια να επεξηγήσει τη λειτουργία καθεμιάς σε μικρότερες και απλούστερες. Συνεχίζοντας βαθμιαία αυτή τη διαδικασία, κάθε ενότητα αναλύεται σε επίπεδο εκτελέσιμων εντολών της γλώσσας. Πρακτικά, η βαθμιαία συγκεκριμενοποίηση υλοποιείται με διαδικασίες και συναρτήσεις.

Λόγω της μικρής ηλικίας των μαθητών που πρόκειται να μελετήσουν τη γλώσσα, έχουν κατασκευαστεί τρεις εκδόσεις της. Η πρώτη από αυτές, η Paschool I, έχει μικρές δυνατότητες, έτσι ώστε να είναι απλή και κατανοητή. Σε κάθε επόμενη έκδοση της γλώσσας προστίθενται βαθμιαία επιπλέον χαρακτηριστικά.

Χαρακτηριστικά της Paschool I.

- Όσον αφορά τις μεταβλητές, υποστηρίζονται μόνο οι τύποι number, boolean και string. Δεν υποστηρίζεται δήλωση μεταβλητών.
- Η Paschool I υποστηρίζει διαδικασίες και συναρτήσεις. Οι διαδικασίες δε δέχονται ορίσματα. Οι συναρτήσεις δέχονται κανένα, ένα ή περισσότερα ορίσματα και επιστρέφουν ακριβώς μία τιμή.
- Οι εντολές που υποστηρίζονται είναι η ανάθεση τιμής, η κλήση διαδικασίας, η εντολή διακλάδωσης if-then-else, η for loop, η εντολή return, που χρησιμοποιείται για επιστροφή τιμής συνάρτησης, εντολές εμφάνισης στην οθόνη και αλλαγής γραμμής καθώς και εντολή ανάγνωσης δεδομένων κατά την εκτέλεση.
- Οι εντολές επανάληψης while και repeat αντικαθίστανται με την εντολή loop, η οποία επαναλαμβάνει μια ακολουθία εντολών και σταματά τις επαναλήψεις όταν εκτελεστεί η εντολή exit.
- Η εντολή εμφάνισης στην οθόνη και η for loop, δέχονται ως ορίσματα απλές παραστάσεις δηλαδή σταθερές ή μεταβλητές.
- Δεν υποστηρίζονται δυαδικοί τελεστές.

Συντακτική περιγραφή της γλώσσας Paschool I:

```
<prog>        ::= 'program' <id> ';' <block> '.'

<compound>    ::= 'begin' <stmt> ( ';' <stmt> )* 'end'
                | '{' <stmt> ( ';' <stmt> )* '}'

<block>       ::= <compound> ( ';' <refine> )*

<refine>      ::= 'procedure' <id> ';' <compound>
                | 'function' <id> <param-list> [ ':' <result> ]
```

```
                                ( ';' <compound> | '=' <expr> )

<result>        ::= 'number' | 'string' | 'boolean'

<param-list>    ::= '(' <id> ( ',' <id> )* ')'

<cond>          ::= <fcall> | <expr> <relop> <expr>
<relop>         ::= '=' | '<>' | '>' | '<' | '<=' | '>='
<expr>          ::= <const> | <id> | <fcall> | '(' <expr> ')' | <unop> <expr>
                  | <expr> <binop> <expr>

<sexpr>         ::= <const> | <id>

<unop>          ::= '+' | '-'

<binop>         ::= '+' | '-' | '*' | '/' | 'div' | 'mod' | '**'

<stmt>          ::= ''
                  | <id> ':=' <expr>
                  | <pcall>
                  | <compound>
                  | 'if' <cond> 'then' <stmt> [ 'else' <stmt> ]
                  | 'loop' <stmt>
                  | 'for' <iter> 'do' <stmt>
                  | 'exit' | 'return' <sexpr>
                  | 'write' '(' <sexpr> ( ',' <sexpr> )* ')'
                  | 'writeln'
                  | 'readln' '(' <id> ')'

<iter>          ::= <expr> 'times'
                  | <id> ':=' <sexpr> ( 'to' | 'downto' ) <sexpr> 'do'

<pcall>         ::= <id>

<fcall>         ::= <id> '(' <sexpr> ( ',' <sexpr> )* ')'

<id>            ::= L(L|D)*

<const>         ::= D+[.D+] | ''P*''
```

συναρτήσεις: odd, abs, sqrt, length, ...

Επιπλέον χαρακτηριστικά της Paschool II.

- Δεν υποστηρίζονται οι παραστάσεις. Η εντολή εμφάνισης στην οθόνη και η for loop, δέχονται ως ορίσματα οποιαδήποτε παράσταση.
- Εισάγονται δυαδικοί τελεστές.
- Υποστηρίζεται η δομή πίνακα (array).
- Μορφοποίηση εξόδου (output formatting).

Συντακτική περιγραφή της γλώσσας Paschool II:

```
<prog>        ::= 'program' <id> ';' <block> '.'

<block>       ::= <compound> ( ';' <refine> )*

<compound>    ::= 'begin' <stmt> ( ';' <stmt> )* 'end'
                | '{' <stmt> ( ';' <stmt> )* '}'

<refine>      ::= 'procedure' <id> ';' <compound>
                | 'function' <id> <param-list> [ ':' <result> ]
                    ( ';' <compound> | '=' <expr> )

<result>      ::= 'number' | 'string' | 'boolean'

<param-list>  ::= '(' <id> ( ',' <id> )* ')'

<relop>       ::= '=' | '<>' | '>' | '<' | '<=' | '>='

<var>         ::= <id> [ '[' <expr> ( ',' <expr> )* ']' ]

<expr>        ::= <const> | <var> | <fcall> | '(' <expr> ')'
                | <unop> <expr> | <expr> <binop> <expr>

<unop>        ::= '+' | '-' | 'not'

<binop>       ::= '+' | '-' | '*' | '/' | 'div' | 'mod' | '**' | 'and' | 'or'

<stmt>        ::= '' | <var> ':=' <expr> | <pcall> | <compound>
                | 'if' <expr> 'then' <stmt> [ 'else' <stmt> ]
                | 'loop' <stmt> | 'for' <iter> 'do' <stmt> | 'exit'
                | 'write' '(' <wexpr> ( ',' <wexpr> )* ')'
                | 'writeln' [ '(' <wexpr> ( ',' <wexpr> )* ')' ]
                | 'readln' '(' <var> ')' | 'return' <expr>

<iter>        ::= <expr> 'times' | <id> ':=' <expr> ( 'to' | 'downto' ) <expr>

<pcall>       ::= <id>
```

<fcall> ::= <id> '(' <expr> (',' <expr>)* ')'

<wexpr> ::= <expr> [<format>]

<format> ::= 'format' '(' <format-param> (',' <format-param>)* ')'

<format-param> ::= <expr> 'places' | <expr> 'decimal' | 'sign' | 'expon'
| 'left' | 'right' | 'center'

<id> ::= L(L|D)*

<const> ::= D+[.D+] | "P*"

συναρτήσεις: odd, abs, sqrt, length, ...

Επιπλέον χαρακτηριστικά της Paschool III.

- Υποστηρίζονται δηλώσεις μεταβλητών. Τύποι: integer, real, char, boolean, strings, arrays.
- Δεν υποστηρίζονται οι απλές παραστάσεις. Η εντολή εμφάνισης στην οθόνη και η for loop, δέχονται ως ορίσματα οποιαδήποτε παράσταση.
- Επιτρέπεται η ανάδρομη.
- Ορίζεται εμβέλεια μεταβλητών.
- Οι διαδικασίες και οι συναρτήσεις δέχονται κανένα, ένα ή περισσότερα ορίσματα.

Συντακτική περιγραφή της γλώσσας Paschool III:

<prog> ::= 'program' <id> ';' <prg-block> '.'

<prg-block> ::= <block> (';' <refine>)*

<block> ::= (<v-decl>)* (<pf-decl>)* <compound>

<v-decl> ::= ('variable' | 'variables')
(<id> (',' <id>)* ':' <type> ';')*

<type> ::= 'integer' | 'real' | 'char' | 'boolean' | 'string'
| 'array' '[' <const> '..' <const> ']' 'of' <type>

<decl> ::= 'procedure' <id> <p-param-list>
| 'function' <id> <f-param-list> ':' <type>

```
<compound>      ::= 'begin' <stmt> ( ';' <stmt> )* 'end'
                  | '{' <stmt> ( ';' <stmt> )* '}'

<refine>        ::= 'procedure' ( <id> '::' )* <id> <p-param-list> <block>
                  | 'function' ( <id> '::' )* <id> <f-param-list> ':' <type>
                        ( ';' <block> | '=' <expr> )

<p-param-list>  ::= '(' <param> ( ';' <param> )* ')'

<p-param>       ::= <id> ( ',' <id> ) ':' <type> [ 'reference' ]

<f-param-list>  ::= '(' <param> ( ';' <param> )* ')'

<f-param>       ::= <id> ( ',' <id> ) ':' <type>

<relop>         ::= '=' | '<>' | '>' | '<' | '<=' | '>='

<var>           ::= <id> [ '[' <expr> ( ',' <expr> )* ']' ]

<expr>          ::= <const> | <var> | <call> | '(' <expr> ')'
                  | <unop> <expr> | <expr> <binop> <expr>

<unop>          ::= '+' | '-' | 'not'

<binop>         ::= '+' | '-' | '*' | '/' | 'div' | 'mod' | '**' | 'and' | 'or'

<stmt>          ::= '' | <var> ':=' <expr> | <call> | <compound>
                  | 'if' <expr> 'then' <stmt> [ 'else' <stmt> ]
                  | 'loop' <stmt> | 'for' <iter> 'do' <stmt> | 'exit'
                  | 'write' '(' <wexpr> ( ',' <wexpr> )* ')'
                  | 'writeln' [ '(' <wexpr> ( ',' <wexpr> )* ')' ]
                  | 'readln' '(' <var> ')' | 'return' <expr>

<iter>          ::= <expr> 'times' | <id> ':=' <expr> ( 'to' | 'downto' ) <expr>

<call>          ::= <id>

<wexpr>         ::= <expr> [ <format> ]

<format>        ::= 'format' '(' <format-param> ( ',' <format-param> )* ')'

<format-param>  ::= <expr> 'places' | <expr> 'decimal' | 'sign' | 'expon'
                  | 'left' | 'right' | 'center'

<id>            ::= L(L|D)*
```

<const> ::= D+[.D+] | "P*"

συναρτήσεις: odd, abs, sqrt, length, ...

Το διδακτικό βοήθημα

Η παρούσα διπλωματική εργασία αποτελεί την πρώτη προσπάθεια συγγραφής διδακτικού βοηθήματος για τη διδασκαλία της γλώσσας Paschool I σε μαθητές της Α΄ τάξης Γυμνασίου. Το βοήθημα αναπτύσσεται σε στενή σχέση με το βιβλίο των μαθηματικών. Στις αρχές των περισσότερων κεφαλαίων, εισάγονται σταδιακά όλα τα χαρακτηριστικά της γλώσσας και συνοδεύονται από πολλά, απλά παραδείγματα, που αποσαφηνίζουν πιθανές απορίες των μαθητών.

Κάθε ενότητα του παρόντος βοηθήματος αντιστοιχεί σε μία ενότητα του βιβλίου των μαθηματικών και περιέχει προγράμματα σχετικά με την ύλη της αντίστοιχης ενότητας των μαθηματικών. Στο τέλος των ενοτήτων, υπάρχουν ασκήσεις προς λύση για να διασφαλιστεί η εμπέδωση τόσο των μαθηματικών εννοιών, όσο και των προγραμματιστικών τεχνικών.

Στο τέλος του βοηθήματος παρατίθενται όλα τα συντακτικά διαγράμματα της γλώσσας. Επίσης, έχει υλοποιηθεί και compiler για την γλώσσα Paschool I. Όλα τα προγράμματα που περιλαμβάνονται στα κεφάλαια, έχουν εκτελεστεί σε αυτόν τον compiler.

Το βιβλίο της Paschool I εντάσσεται στα πλαίσια ενός ευρύτερου προγράμματος με ορίζοντα πενταετίας, με απώτερο σκοπό την αντικατάσταση του μαθήματος των μαθηματικών της μέσης εκπαίδευσης με ένα μάθημα που θα συνδυάζει μαθηματικά και προγραμματισμό. Για το σκοπό αυτό, στο μέλλον θα συγγραφούν αντίστοιχα βιβλία και compilers που θα εισάγουν τη γλώσσα Paschool II στη Β΄ τάξη Γυμνασίου και την Paschool III στην Γ΄ τάξη Γυμνασίου. Όσον αφορά το λύκειο, θα αναπτυχθούν επόμενες εκδόσεις της Paschool, με αντικειμενοστραφή προσανατολισμό (object oriented).

Σε δεύτερο επίπεδο, θα πρέπει να γίνει δοκιμαστική διδασκαλία του μαθήματος σε περιορισμένο αριθμό σχολείων, να ακολουθήσει έρευνα στους μαθητές όσον αφορά το μάθημα και βάσει των συμπερασμάτων, να γίνουν οι απαραίτητες διορθώσεις και βελτιώσεις στο βιβλίο. Αρκετά ενδιαφέρουσα θα ήταν και μια προσπάθεια δημιουργίας ενός ενιαίου βιβλίου για το μάθημα μαθηματικών και πληροφορικής.

Paschool

Εκπαιδευτική Γλώσσα Προγραμματισμού για το Γυμνάσιο

Κεφάλαιο 1

Οι Φυσικοί Αριθμοί

ΕΙΣΑΓΩΓΗ

Συχνά, για την επίλυση διάφορων μαθηματικών προβλημάτων χρησιμοποιούνται οι ηλεκτρονικοί υπολογιστές. Οι λόγοι που χρησιμοποιούνται οι ηλεκτρονικοί υπολογιστές είναι:

- ✓ η ταχύτητά τους στην εκτέλεση των πράξεων και
- ✓ η ακρίβειάς τους.

Οι ηλεκτρονικοί υπολογιστές έχουν την ικανότητα να εκτελούν πολλές αριθμητικές πράξεις σε μικρό χρονικό διάστημα. Για παράδειγμα, ένας τυπικός ηλεκτρονικός υπολογιστής σημερινής τεχνολογίας μπορεί να εκτελέσει δισεκατομμύρια πράξεις το δευτερόλεπτο! Επομένως, οι ηλεκτρονικοί υπολογιστές επιλύουν γρήγορα πολύπλοκα προβλήματα.

Όμως παρά τις δυνατότητες που έχει ο ηλεκτρονικός υπολογιστής, δεν είναι τίποτα περισσότερο από μια περίπλοκη μηχανή. Δεν έχει λογική και κρίση. Έχει όμως τη δυνατότητα να εκτελεί **εντολές**, δηλαδή οδηγίες που δίνονται από άνθρωπο.

Για την επικοινωνία ανθρώπου και υπολογιστή χρησιμοποιούνται ειδικές γλώσσες με συγκεκριμένα σύμβολα και συντακτική δομή.

Η περιγραφή των εργασιών που πρέπει να εκτελέσει ο ηλεκτρονικός υπολογιστής για να λύσει ένα πρόβλημα ονομάζεται **πρόγραμμα**. Κάθε πρόγραμμα είναι γραμμένο σε μια **γλώσσα προγραμματισμού**, δηλαδή μια γλώσσα η οποία είναι αντιληπτή από τον άνθρωπο και από τον ηλεκτρονικό υπολογιστή.

Με άλλα λόγια, ο προγραμματιστής γνωρίζοντας τον τρόπο με τον οποίο λύνεται ένα πρόβλημα, σχεδιάζει εκ των προτέρων (γι' αυτό και πρόγραμμα) και περιγράφει στον υπολογιστή ποιες εργασίες πρέπει να γίνουν για να λυθεί.

Μετά το τέλος της **συγγραφής** του προγράμματος, δίνεται στον ηλεκτρονικό υπολογιστή κατάλληλη εντολή ώστε να αρχίσει να εκτελεί το πρόγραμμα, δηλαδή να διεκπεραιώσει όλες τις εργασίες που περιγράφονται σε αυτό με τη σωστή σειρά. Η διαδικασία αυτή ονομάζεται **εκτέλεση προγράμματος**.

Κατά τη διάρκεια της εκτέλεσης του προγράμματος, εμφανίζονται στην οθόνη τα αποτελέσματα. Ένα πρόγραμμα μπορεί να εκτελεστεί πολλές φορές ώστε να επιλυθεί το ίδιο πρόβλημα για διαφορετικά αριθμητικά δεδομένα.

Ανακεφαλαιώνοντας, διακρίνονται δύο διαφορετικές λειτουργίες. Η **συγγραφή** ενός προγράμματος και η **εκτέλεσή** του.

ΔΟΜΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

Τα προγράμματα έχουν συγκεκριμένη δομή. Η δομή ενός προγράμματος είναι η ακόλουθη:

```
program όνομα προγράμματος;                    (*Επικεφαλίδα*)
begin
        εντολές προγράμματος                    (*Κυρίως σώμα προγράμματος*)
end.
```

1. Το όνομα του προγράμματος είναι μια συμβολοσειρά (**string**), δηλαδή μια σειρά από γράμματα και αριθμούς, η οποία:

   - Πρέπει να ξεκινά με γράμμα του λατινικού αλφαβήτου και να μην περιλαμβάνει κενά.
   - Δεν πρέπει να είναι κάποια από τις δεσμευμένες λέξεις, δηλαδή λέξεις με προκαθορισμένη σημασία, όπως οι λέξεις **program**, **begin** και **end** (στη συνέχεια θα αναφερθούν και άλλες)

   Για παράδειγμα, ένα πρόγραμμα θα μπορούσε να είχε ένα από τα ονόματα:

   agymnasiou
   t123ert45
   d1a2b3c
   program1

   αλλά κανένα από τα παρακάτω (γιατί;):

   1gymnasiou
   kl23 ert45
   la2b3c
   program
   end

2. Οι εντολές (statements, commands) του προγράμματος περιγράφουν διάφορες απλές εργασίες που μπορεί να εκτελέσει ο ηλεκτρονικός υπολογιστής. Με τη χρήση πολλών εντολών και συνδυασμό αυτών σε ένα πρόγραμμα μπορούν να εκτελεστούν συνθετότερες εργασίες (περισσότερα θα δούμε στη συνέχεια).

3. Οι χαρακτήρες που βρίσκονται μεταξύ των συμβόλων (* και *) αποτελούν **σχόλια** και δεν επηρεάζουν την εκτέλεση του προγράμματος. Διευκολύνουν την κατανόηση των προγραμμάτων, δηλαδή είναι για τους ανθρώπους - αναγνώστες. Σχόλια μπορούν να συμπεριληφθούν σε οποιοδήποτε σημείο του προγράμματος.

ΕΝΤΟΛΗ **WRITE** (πρώτο μέρος)

Η εντολή **write** εμφανίζει μηνύματα στην οθόνη.

Για παράδειγμα

```
                                        (*Εμφάνιση μηνύματος στην οθόνη*)
program Hello1;
begin
        write(“Καλημέρα !!!”)
end.
```

Το παραπάνω πρόγραμμα εμφανίζει στην οθόνη το εξής μήνυμα:

Καλημέρα !!!

Δηλαδή η εντολή **write** εμφανίζει στην οθόνη τη συμβολοσειρά (string) που βρίσκεται μεταξύ παρενθέσεων και αριστερών αποστρόφων, δηλαδή μεταξύ των συμβόλων (“ και “).

Μεταξύ δύο διαδοχικών εντολών πρέπει να τοποθετείται το σύμβολο **; (semicolon)**.

Για παράδειγμα:

```
                                        (*Εμφάνιση 5 μηνυμάτων στην οθόνη*)
program Hello2;
begin
        write(“Καλημέρα !!!”);
        write(“Καλημέρα !!!”);
        write(“Καλημέρα !!!”);
        write(“Καλημέρα !!!”);
        write(“Καλημέρα !!!”)
end.
```

Το παραπάνω πρόγραμμα εμφανίζει στην οθόνη το εξής μήνυμα:

Καλημέρα !!! Καλημέρα !!! Καλημέρα !!! Καλημέρα !!! Καλημέρα !!!

ΠΡΟΣΟΧΗ!!

Τονίζεται ότι οι λέξεις **begin** και **end** δεν αποτελούν εντολές. Είναι απλώς δεσμευμένες λέξεις. Επομένως μετά το **begin** δε χρειάζεται semicolon. Το ίδιο ισχύει και πριν το **end**.

Η εντολή **writeln** μετακινεί το **δρομέα (cursor)**, δηλαδή το σημάδι που αναβοσβήνει στην οθόνη, στην αρχή της επόμενης γραμμής.

Για παράδειγμα:

```
                                        (*Εμφάνιση 5 μηνυμάτων σε 5 γραμμές*)
program Hello3;
begin
        write(“Καλημέρα !!!”);   writeln;
        write(“Καλημέρα !!!”);   writeln;
        write(“Καλημέρα !!!”);   writeln;
        write(“Καλημέρα !!!”);   writeln;
        write(“Καλημέρα !!!”);   writeln
end.
```

Το παραπάνω πρόγραμμα εμφανίζει στην οθόνη το εξής μήνυμα:

Καλημέρα !!!
Καλημέρα !!!
Καλημέρα !!!
Καλημέρα !!!
Καλημέρα !!!

Ασκήσεις – Εφαρμογές

1. Να γράψετε ένα πρόγραμμα το οποίο να εμφανίζει στην οθόνη το όνομά σας.
2. Να γράψετε ένα πρόγραμμα το οποίο να εμφανίζει στην πρώτη σειρά το όνομά σας και στην δεύτερη το επίθετό σας.

ΜΕΤΑΒΛΗΤΕΣ – ΑΝΑΘΕΣΗ ΤΙΜΗΣ

Στα περισσότερα προγράμματα είναι χρήσιμο να αποθηκεύονται στη μνήμη του ηλεκτρονικού υπολογιστή συγκεκριμένα δεδομένα που είναι απαραίτητα για τον υπολογισμό των αποτελεσμάτων. Για το λόγο αυτό, χρησιμοποιούνται μεταβλητές.

Οι μεταβλητές είναι ονόματα περιοχών της μνήμης του υπολογιστή (κελιών μνήμης) καθεμιά από τις οποίες μπορεί να αποθηκεύει έναν αριθμό.

Μια μεταβλητή, όπως και το όνομα του προγράμματος, είναι μια συμβολοσειρά, δηλαδή μια σειρά από γράμματα και αριθμούς, η οποία πρέπει να ξεκινά με γράμμα και να μην περιλαμβάνει κενά.

Συνήθως, τα ονόματα των μεταβλητών που χρησιμοποιούνται σε ένα πρόγραμμα περιγράφουν το περιεχόμενο της μεταβλητής, σε σχέση με τη χρησιμότητά της στο πρόγραμμα. Αυτό γίνεται για να είναι πιο ευανάγνωστο το πρόγραμμα.

Για παράδειγμα, σε ένα πρόγραμμα που κάνει πρόσθεση δύο αριθμών, μπορούν να χρησιμοποιηθούν τα ονόματα number1 και number2 για τους δύο αριθμούς (ή για συντομία num1 και num2) και το όνομα result ή sum για το άθροισμά τους.

Η εκχώρηση τιμής σε μια μεταβλητή γίνεται με το σύμβολο := , που διαβάζεται ως εξής: «παίρνει την τιμή».

Για παράδειγμα:

```
num := 43          (*Η μεταβλητή num παίρνει την τιμή 43*)
sum := 78
result := 0
```

Η εντολή που αποθηκεύει τιμή σε μεταβλητή ονομάζεται **ανάθεση τιμής**.

Η ανάθεση τιμής σε μεταβλητές φαίνεται στο παρακάτω πρόγραμμα:

```
                                        (*Αναθέσεις τιμών*)
program Variables;
begin
        big := 4;   small := 3;
        result := big + small + 10
end.
```

Στο πρόγραμμα αυτό:

- Η μεταβλητή big παίρνει την τιμή 4.
- Η μεταβλητή small παίρνει την τιμή 3.
- Πρώτα, ο υπολογιστής αποτιμά την αριθμητική παράσταση big+small+10 (δηλαδή 4+3+10) και μετά η μεταβλητή result παίρνει την τιμή της παράστασης.

ΠΡΟΣΟΧΗ!!!

Τονίζεται ότι το σύμβολο := έχει διαφορετική σημασία από το ίσο (=) που χρησιμοποιείται στα μαθηματικά. Το ίσο (=) σημαίνει ότι η παράσταση δεξιά του έχει ίση τιμή με την παράσταση αριστερά του. Το σύμβολο := σημαίνει ότι η τιμή της παράστασης ή της μεταβλητής δεξιά του αποθηκεύεται στη μεταβλητή αριστερά του.

Λόγω αυτής της διαφοράς, η εντολή i:=i+1 είναι σωστή, και σημαίνει ότι στην μεταβλητή i αποθηκεύεται η τιμή που είχε η μεταβλητή i μέχρι τώρα, αυξημένη κατά μια μονάδα. Δηλαδή, η τιμή της μεταβλητής i αυξάνεται κατά ένα.

Τι τιμή θα έχει η μεταβλητή i μετά την εκτέλεση του παρακάτω προγράμματος;

```
                                        (*Αναθέσεις τιμών*)
program Variables2;
begin
        i:=4;   i:=3;
        i:=i+10;
        i:=i-9
end.
```

Τα προηγούμενα δύο προγράμματα δεν έχουν νόημα όμως, διότι δεν εμφανίζουν τα αποτελέσματα στην οθόνη.

ΕΝΤΟΛΗ **WRITE** (δεύτερο μέρος)

Η εντολή **write** χρησιμοποιείται και για την εμφάνιση της τιμής μιας μεταβλητής στην οθόνη. Εντός των παρενθέσεων του **write** αναφέρεται το όνομα που αντιστοιχεί

στη μεταβλητή (χωρίς το ζεύγος αριστερών αποστρόφων), η τιμή της οποίας θα εμφανιστεί στην οθόνη όταν εκτελεστεί το πρόγραμμα.

Για παράδειγμα:

```
                                                    (*Εμφάνιση τιμής μεταβλητής στην οθόνη*)
program VariableWrite1;
begin
        num := 34;                                                  (*Ανάθεση τιμής*)
        write(“num”);   write(num);   writeln; (*Εμφάνιση μηνύματος στην οθόνη*)
end.
```

Το πρόγραμμα εμφανίζει στην οθόνη:

num 34

Δηλαδή η πρώτη εντολή **write** του προγράμματος εμφανίζει τη λέξη num και η δεύτερη εντολή **write** εμφανίζει την τιμή που έχει η μεταβλητή num στο πρόγραμμα όταν εκτελείται το **write**, δηλαδή την τιμή 34. (Παρατήρηση: στην οθόνη δεσμεύονται συνολικά 8 θέσεις για την τιμή μιας μεταβλητής. Άρα, εδώ έχουμε 6 κενά σύν δύο θέσεις για τα ψηφία 3 4.)

Επίσης, η εντολή **write** μπορεί να χρησιμοποιηθεί όπως παρακάτω:

```
program VariableWrite2;
begin
        num := 34;                                          (*Ανάθεση τιμής*)
        write(“num”,num);                    (*Εμφάνιση μηνύματος στην οθόνη*)
        num := 43;                                          (*Ανάθεση τιμής*)
        write(“num”,num);                    (*Εμφάνιση μηνύματος στην οθόνη*)
        writeln                                             (*Αλλαγή γραμμής*)
end.
```

Το πρόγραμμα εμφανίζει στην οθόνη:

num 34num 43

Για κάθε εντολή **write** του παραπάνω προγράμματος, εμφανίζονται στην οθόνη ακριβώς όπως αναγράφονται εντός της παρένθεσης οι λέξεις που περικλείονται από ζεύγος αριστερών αποστρόφων (“num”). Οι λέξεις εκτός αποστρόφων είναι ονόματα

μεταβλητών. Κατά την εκτέλεση εμφανίζεται η τιμή που είναι αποθηκευμένη σε καθεμιά.

ΕΝΤΟΛΕΣ ΕΠΑΝΑΛΗΨΗΣ

Σε αυτό το κεφάλαιο θα δούμε τρεις εντολές επανάληψης. Αυτές οι εντολές επιτρέπουν την εκτέλεση άλλων εντολών καμία, μία ή περισσότερες φορές.
Εντολές επανάληψης, παραδείγματος χάρην:

- **for** 9 **times do** *σώμα*
- **for** i := 1 **to** 9 **do** *σώμα*
- **loop** *σώμα*

Σώμα ονομάζεται μία εντολή απλή ή σύνθετη.

Σε αυτό το κεφάλαιο θα μελετήσουμε τις δύο πρώτες εντολές.

- Εντολή: **for** *αριθμός* **times do** *σώμα*
  - ✓ Ο αριθμός δείχνει πόσες φορές θα εκτελεστούν οι εντολές του σώματος.
  - ✓ Το σώμα περιέχει εντολές. Αν περιέχει περισσότερες από μία εντολές τότε οι εντολές πρέπει να περιέχονται μέσα σε **begin – end**.

Σύνθετη εντολή ονομάζεται μια ακολουθία εντολών που περικλείεται από τις λέξεις **begin** και **end**.

Παράδειγμα σύνθετης εντολής: **begin** i := 4; j := 3; i := j + 10; j := i - 9; **end**

```
                                    (*Εμφάνιση 5 όμοιων μηνυμάτων σε μια γραμμή της οθόνης*)
program Hello5;
begin
        for 5 times do write(“Καλημέρα !!!”)
end.
```

Η εντολή σώματος θα εκτελεστεί 5 φορές, οπότε το πρόγραμμα εμφανίζει στην οθόνη:

Καλημέρα !!! Καλημέρα !!! Καλημέρα !!! Καλημέρα !!! Καλημέρα !!!

```
                    (*Εμφάνιση 10 όμοιων μηνυμάτων σε διαδοχικές γραμμές της οθόνης*)
program Hello9;
begin
        for 9 times do
                begin   write(“Καλημέρα !!!”);   writeln   end
end.
```

Η σύνθετη εντολή σώματος θα εκτελεστεί 9 φορές, οπότε το πρόγραμμα εμφανίζει στην οθόνη:

Καλημέρα !!!
Καλημέρα !!!
Καλημέρα !!!
Καλημέρα !!!
Καλημέρα !!!
Καλημέρα !!!
Καλημέρα !!!
Καλημέρα !!!
Καλημέρα !!!

- Ένας άλλος τρόπος διατύπωσης της εντολής αυτής είναι ο εξής:

for i:=1 **to** 9 **do** *σώμα*

Αυτή η διατύπωση χρησιμοποιεί τη μεταβλητή i, η οποία κατά την πρώτη επανάληψη έχει την τιμή 1 και σε κάθε επόμενη επανάληψη η τιμή της αυξάνεται κατά 1. Όταν η τιμή της γίνει 10, τότε το σώμα του **for loop** εκτελείται για τελευταία φορά και έπειτα οι επαναλήψεις σταματούν.

Αυτή η διατύπωση της εντολής επανάληψης μπορεί να είναι λίγο πιο περίπλοκη από την προηγούμενη, όμως έχει το πλεονέκτημα ότι μπορούμε να χρησιμοποιήσουμε στις εντολές που επαναλαμβάνονται την τιμή της μεταβλητής i (που ονομάζεται μεταβλητή ελέγχου). Παράδειγμα:

```
                    (*Εμφάνιση 9 διαφορετικών μηνυμάτων σε διαδοχικές γραμμές της οθόνης*)
program Hello9;
begin
        for i:=1 to 9 do   begin   write(“Μήνυμα “,i,” :  Καλημέρα !!!”);   writeln   end
end.
```

Η σύνθετη εντολή σώματος θα εκτελεστεί 9 φορές, οπότε το πρόγραμμα εμφανίζει στην οθόνη:

Μήνυμα 1 : Καλημέρα !!!
Μήνυμα 2 : Καλημέρα !!!
Μήνυμα 3 : Καλημέρα !!!
Μήνυμα 4 : Καλημέρα !!!
Μήνυμα 5 : Καλημέρα !!!
Μήνυμα 6 : Καλημέρα !!!
Μήνυμα 7 : Καλημέρα !!!
Μήνυμα 8 : Καλημέρα !!!
Μήνυμα 9 : Καλημέρα !!!

Γενίκευση: **for** i := *αριθμός1* **to** *αριθμός2* **do** *σώμα*
Η μεταβλητή i κατά την πρώτη επανάληψη έχει την τιμή του *αριθμός1* και σε κάθε επόμενη επανάληψη η τιμή της αυξάνεται κατά 1. Όταν η τιμή της γίνει ίση με τον *αριθμό2*, τότε το *σώμα* εκτελείται για τελευταία φορά και έπειτα οι επαναλήψεις σταματούν. Συνολικά, οι εντολές σώματος εκτελούνται: (αριθμό2 - αριθμός1) + 1 φορές.

- Θα εξηγήσουμε την εντολή **loop** *σώμα* σε επόμενο κεφάλαιο.

Ασκήσεις – Εφαρμογές

1. Εντοπίστε ομοιότητες και διαφορές στα παρακάτω προγράμματα. Τι θα εμφανιστεί στην οθόνη μετά την εκτέλεση καθενός;

```
program HelloExercise1a;
begin
        for 5 times do write(“Καλημέρα !!!”)
end.

program HelloExercise1b;
begin
        for 5 times do   begin   write(“Καλημέρα !!!”)   end
end.

program HelloExercise1c;
begin
        for 5 times do    begin   write(“Καλη”);   write(“μέρα !!!”)   end
end.


program HelloExercise1d;
begin
        for 5 times do write(“Καλη”);   write(“μέρα !!!”)
end.
```

2. Τι θα εμφανιστεί στην οθόνη μετά την εκτέλεση του προγράμματος;

```
program WriteCheck;
begin
        one := 25;   two := 52;    three := one;   one := two;
        write(“one”,two,”three”);   writeln;
        write(one,”two”,three);   writeln;
        write(one,two,three);   writeln
end.
```

3. Εντοπίστε ομοιότητες και διαφορές στα παρακάτω προγράμματα. Τι θα εμφανιστεί στην οθόνη μετά την εκτέλεση καθενός;

```
program HelloExercise2a;
begin   write(“Καλημέρα !!!”);   writeln
end.
```

```
program HelloExercise2b;
begin   write(“Καλη”,” μέρα !!!”);   writeln
end.
```

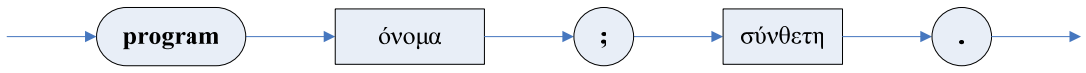
```
program HelloExercise2c;
begin   write(“Καλη”);   write(“μέρα !!!”);   writeln
end.
```

Συντακτικά Διαγράμματα

Τα **συντακτικά διαγράμματα** αποτελούν έναν αυστηρό τρόπο περιγραφής συντακτικά ορθού προγράμματος ή τμήματος προγράμματος.

Για παράδειγμα, το συντακτικό διάγραμμα **πρόγραμμα** (συντακτικό διάγραμμα 1) περιγράφει τον συντακτικά ορθό τρόπο συγγραφής ενός προγράμματος:

- Στην αρχή, ο προγραμματιστής γράφει τη δεσμευμένη λέξη **program**.
- Στη συνέχεια γράφει ένα όνομα, μετά semicolon, μετά μια σύνθετη εντολή και τέλος μια τελεία.

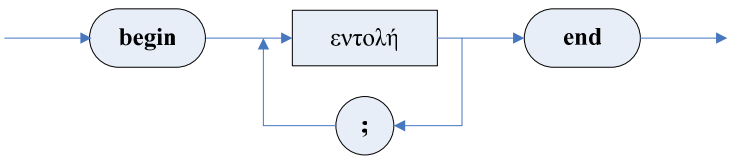


Συντακτικό διάγραμμα 1: ***πρόγραμμα***

Τα σύμβολα που εμφανίζονται στα συντακτικά διαγράμματα επεξηγούνται παρακάτω:

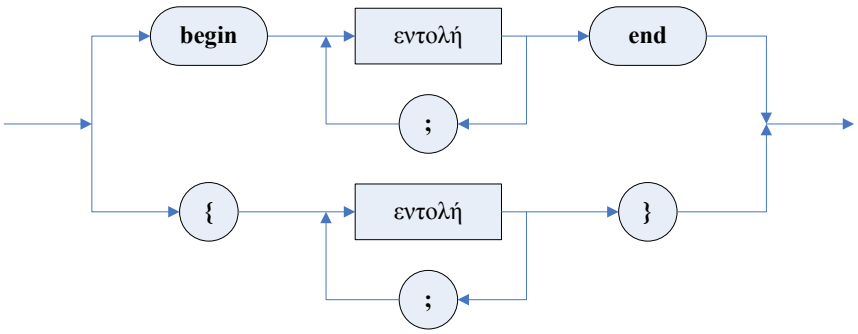
- *Οβάλ ή κυκλικό πλαίσιο με έντονα λατινικά γράμματα ή σύμβολα:* Οι λέξεις αυτές γράφονται όπως έχουν στο πρόγραμμα.
- *Ορθογώνιο πλαίσιο με ελληνικά ή λατινικά γράμματα:* Έννοιες που επεξηγούνται σε άλλο συντακτικό διάγραμμα.
- *Βέλη:* Δείχνουν την διαδρομή που ακολουθείται στο συντακτικό διάγραμμα, για να παράχθει ένα συντακτικά ορθό πρόγραμμα ή τμήμα προγράμματος.
- *Διασταυρώσεις βελών:* Επιλέγουμε όποια διαδρομή επιθυμούμε αρκεί να διατηρείται η φορά των βελών.

Το τμήμα προγράμματος *σύνθετη*, που εμφανίζεται στο συντακτικό διάγραμμα 1, επεξηγείται στο συντακτικό διάγραμμα2 :



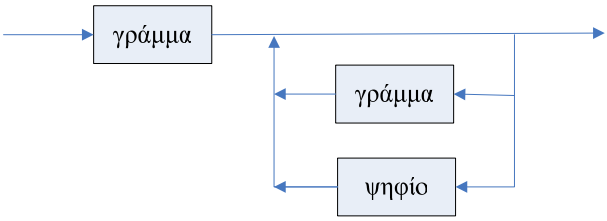
Συντακτικό διάγραμμα 2: ***σύνθετη (εντολή)***

Εναλλακτικά, αντί για begin και end, μπορεί να χρησιμοποιηθεί το ζεύγος άγκιστρων. Οπότε για τη σύνθετη εντολή προκύπτει το συντακτικό διάγραμμα 3.



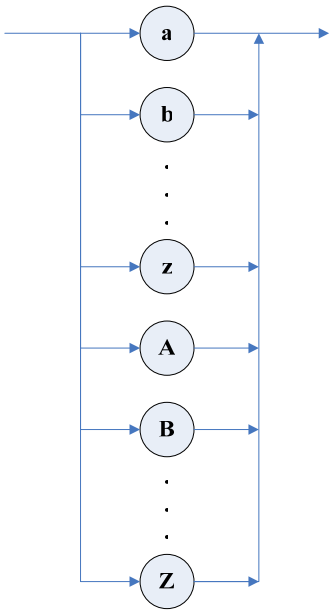
Συντακτικό διάγραμμα 3: ***σύνθετη (εντολή)***

Με τους κανόνες που παρουσιάστηκαν παραπάνω, επεξηγούνται όλα τα παρακάτω συντακτικά διαγράμματα, που παρουσιάζουν το τμήμα της γλώσσας προγραμματισμού Paschool που έχουμε παρουσιάσει μέχρι τώρα.

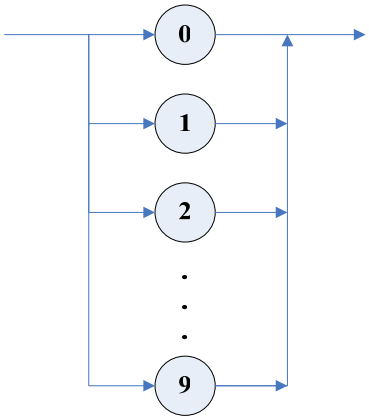


Συντακτικό διάγραμμα 4: ***όνομα***

Δηλαδή, ένα όνομα αρχίζει πάντα με γράμμα και περιέχει γράμματα (μικρά ή κεφαλαία του λατινικού αλφαβήτου, όπως φαίνεται στο συντακτικό διάγραμμα 5) ή ψηφία (0 . . 9, συντακτικό διάγραμμα 6).

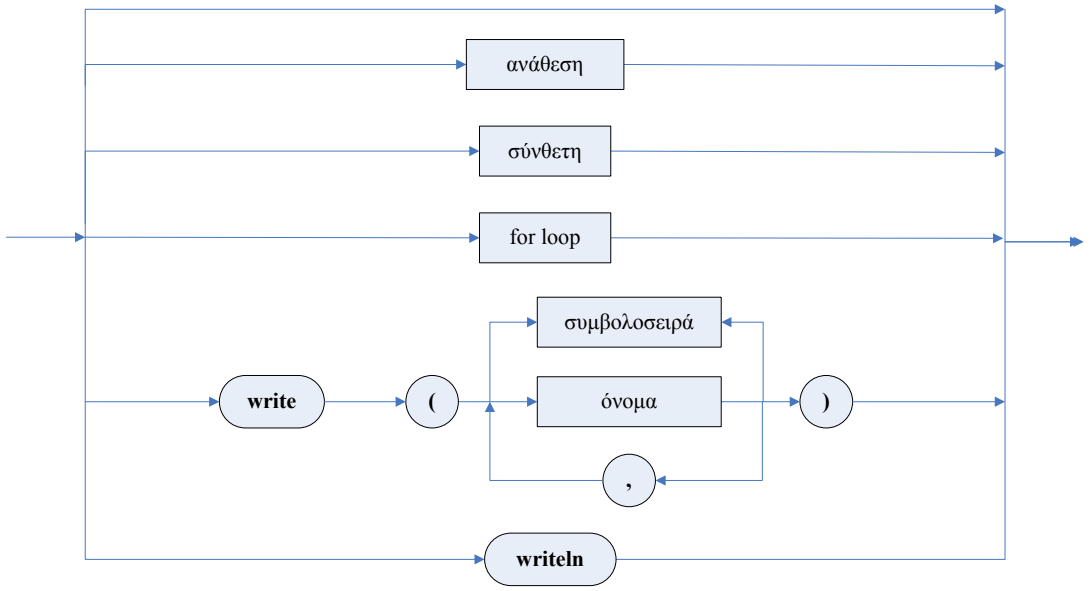


Συντακτικό διάγραμμα 5: ***γράμμα***



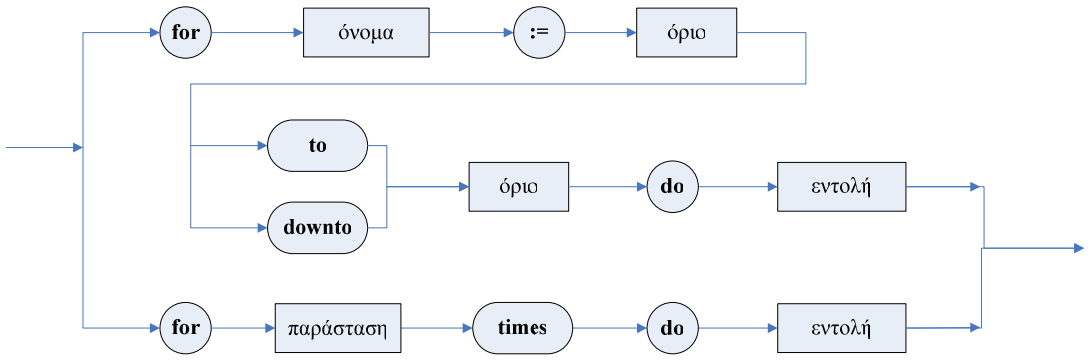
Συντακτικό διάγραμμα 6: ***ψηφίο***

Στο παρακάτω συντακτικό διάγραμμα, εμφανίζονται όλες οι εντολές της γλώσσας που έχουν παρουσιαστεί μέχρι τώρα. Αυτές είναι, η κενή εντολή, η εντολή ανάθεσης, η σύνθετη εντολή, η **for loop** με δύο μορφές, η **write** και η **writeln**.

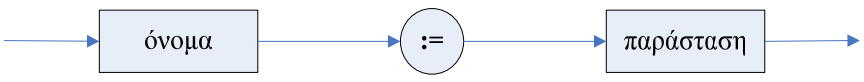


Συντακτικό διάγραμμα 7: ***εντολή***

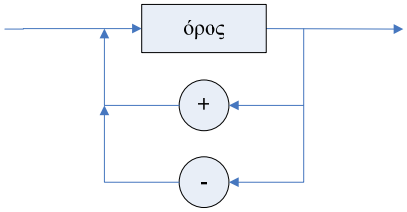
Στα συντακτικά διαγράμματα που ακολουθούν, εμφανίζεται η for loop, η εντολή ανάθεσης του διαγράμματος 7, καθώς και τα τμήματα προγράμματος που τη συνθέτουν.



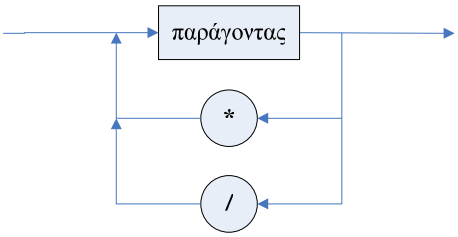
Συντακτικό διάγραμμα 8: ***for loop***



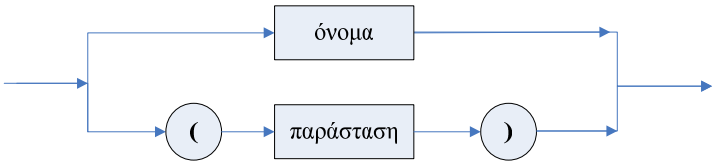
Συντακτικό διάγραμμα 9: ***ανάθεση***



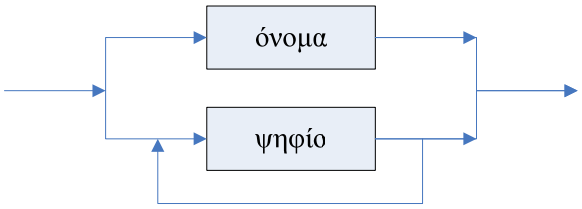
Συντακτικό διάγραμμα 10: ***παράσταση***



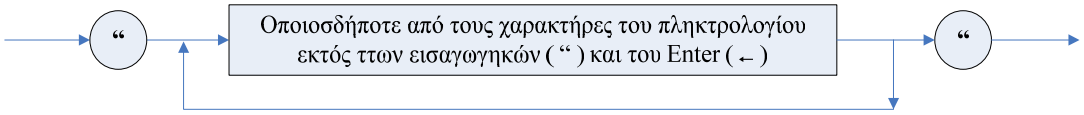
Συντακτικό διάγραμμα 11: ***όρος***



Συντακτικό διάγραμμα 12: ***παράγοντας***



Συντακτικό διάγραμμα 13: ***όριο***



Συντακτικό διάγραμμα 14: ***συμβολοσειρά***

Κεφάλαιο 1.1 – 1.2 – Οι Φυσικοί Αριθμοί

1.1.1. Να γραφτούν δύο προγράμματα τα οποία να εμφανίζουν στην οθόνη όλους τους φυσικούς αριθμούς από το 1 μέχρι το 100, με δύο τρόπους, ένα για κάθε τρόπο διατύπωσης της εντολής επανάληψης **for**.

Α΄ τρόπος

```
program Natural1;
begin
        i:=0;                                          (*Αρχικοποίηση μεταβλητής*)
        for 100 times do          (*Υπολογισμός αριθμού, εμφάνιση στην οθόνη,*)
        begin  i:=i+1;  write(i);  writeln  end                (*αλλαγή γραμμής*)
end.
```

Β΄ τρόπος

```
program Natural2;
begin
        for i:=1 to 100 do  begin  write(i);  writeln  end
end.
```

1.1.2. Να γραφτούν δύο προγράμματα τα οποία να εμφανίζουν στην οθόνη όλους τους άρτιους αριθμούς από το 2 μέχρι το 100, με δύο τρόπους, ένα για κάθε τρόπο διατύπωσης της εντολής επανάληψης **for**.

Α΄ τρόπος

```
program Even1;
begin
        i:=0;
        for 50 times do
        begin              (*Ο αριθμός που εμφανίζεται στην οθόνη, αυξάνεται κατά 2*)
                i:=i+2;  write(i);  writeln
        end
end.
```

Β΄ τρόπος

```
program Even2;
begin                                          (*Με χρήση βοηθητικής μεταβλητής j*)
        for i:=1 to 50 do  begin  j:=2*i;  write(j);  writeln  end
end.
```

ΕΝΤΟΛΗ **READLN**

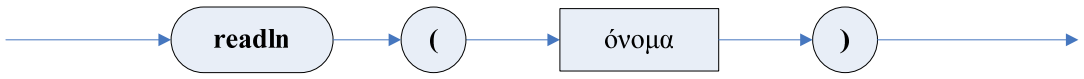
Σε σχέση με τον υπολογιστή, ο άνθρωπος έχει δύο ρόλους:

- Το ρόλο του προγραμματιστή, ο οποίος κατασκευάζει ένα πρόγραμμα
- Το ρόλο του χρήστη, ο οποίος χρησιμοποιεί το πρόγραμμα και βλέπει τα αποτελέσματά του στην οθόνη.

Τα προγράμματα που αναπτύχθηκαν μέχρι τώρα διαθέτουν όλες τις απαραίτητες πληροφορίες πριν εκτελεστούν.

Ιδιαίτερα βολικό θα ήταν να έχει ο χρήστης τη δυνατότητα να αναθέσει τιμές σε μεταβλητές κατά τη διάρκεια της εκτέλεσης ενός προγράμματος. Αυτό μπορεί να γίνει με την εντολή **readln**.

Όταν ο ηλεκτρονικός υπολογιστής εκτελεί την εντολή **readln**(num) τότε ο δρομέας αναβοσβήνει στην οθόνη και ο υπολογιστής περιμένει από τον χρήστη να πληκτρολογήσει μία τιμή, που θα αποθηκευτεί στη μεταβλητή num.



*Συντακτικό διάγραμμα της εντολής **readln***

Ας δούμε τη λειτουργία της εντολής **readln** σε ένα παράδειγμα:

```
program Variables1;
begin
                                                (*Εισαγωγή δεδομένων*)
        write(“Δώστε τον πρώτο αριθμό: “);   readln(num1);
        write(“ Δώστε τον δεύτερο αριθμό: “);   readln(num2);
        sum:=num1+num2;                          (*Υπολογισμός αποτελέσματος*)
        write(“Το άθροισμά τους είναι: “,sum);   writeln
end.
```

Ας δούμε αναλυτικά τι κάνει το παραπάνω πρόγραμμα:

- Το πρόγραμμα εμφανίζει στην οθόνη το μήνυμα «Δώστε τον πρώτο αριθμό :»
- Με την εντολή **readln**(num1) ο υπολογιστής περιμένει μέχρι ο χρήστης να δώσει την τιμή για τον πρώτο αριθμό, που αποθηκεύεται στη μεταβλητή num1.
- Στη συνέχεια το πρόγραμμα εμφανίζει στην οθόνη το μήνυμα «Δώστε τον δεύτερο αριθμό :»
- Με την εντολή **readln**(num2) ο υπολογιστής περιμένει μέχρι ο χρήστης να δώσει την τιμή για το δεύτερο αριθμό , που αποθηκεύεται στη μεταβλητή num2.
- Με την εντολή sum := num1 + num2 ο υπολογιστής υπολογίζει το άθροισμα των δύο αριθμών που έδωσε ο χρήστης και το αποθηκεύει στη μεταβλητή sum.
- Τέλος, ο υπολογιστής, με την εντολή **write**("Το άθροισμά τους είναι: ",sum), εμφανίζει στην οθόνη το άθροισμα που υπολογίστηκε, δηλαδή την τιμή της sum.

Το παραπάνω πρόγραμμα μπορεί να τροποποιηθεί έτσι ώστε να δίνει στο χρήστη δυνατότητα να εκτελέσει όσες προσθέσεις επιθυμεί:

```
program Variables2;
begin
        write("Πόσες προσθέσεις επιθυμείτε να εκτελέσετε ; "); readln(i);
        for i times do
        begin
                write("Δώστε τον πρώτο αριθμό : "); readln(num1);
                write(" Δώστε το δεύτερο αριθμό : "); readln(num2);
                sum:=num1+num2;
                write("Το άθροισμά τους είναι : ",sum); writeln
        end
end.
```

ΕΝΤΟΛΗ **IF – THEN – ELSE**

Πολλές φορές, αν ικανοποιείται κάποια συνθήκη ενεργούμε με έναν συγκεκριμένο τρόπο, ειδάλλως ενεργούμε διαφορετικά.
Για παράδειγμα:

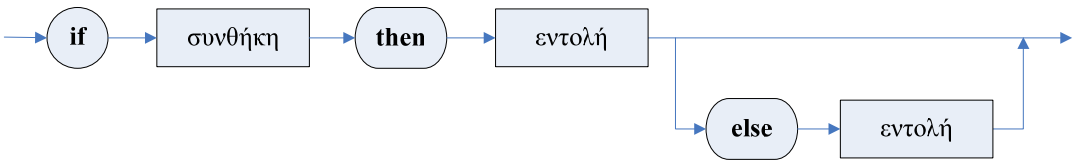
- Αν βρέχει θα πάρω ομπρέλα. Αν δεν βρέχει δεν θα πάρω.
- Αν κάνει κρύο θα πάρω το παλτό μου αλλιώς δεν θα το πάρω.
- Αν ο παρονομαστής του κλάσματος είναι μηδέν τότε έχω κάνει λάθος.

Η ίδια λογική υπάρχει και στον προγραμματισμό. Έτσι λοιπόν μπορούμε να κατασκευάζουμε προγράμματα τα οποία εκτελούν συγκεκριμένες εντολές αφού ελέγξουν μία η περισσότερες συνθήκες

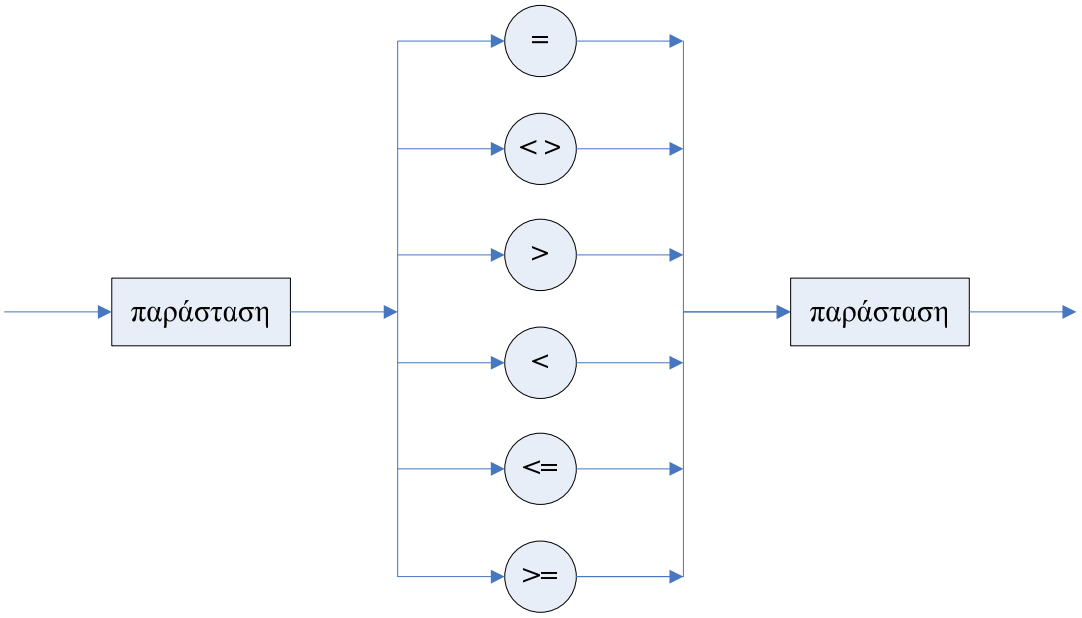
Ο έλεγχος αυτός γίνεται από την εντολή **if – then – else**.

Η εντολή **if – then – else** συντάσσεται ως εξής:

- Μετά το **if** ακολουθεί μία συνθήκη ελέγχου, δηλαδή μια μαθηματική παράσταση, την οποία ο υπολογιστής ελέγχει αν είναι αληθής ή ψευδής.
  - Αν η συνθήκη ελέγχου είναι αληθής τότε εκτελείται η εντολή που ακολουθεί το **then**.
  - Αν η συνθήκη ελέγχου είναι ψευδής τότε εκτελείται η εντολή που ακολουθεί το **else**.



*Συντακτικό διάγραμμα της εντολής **if – then – else***



Συντακτικό διάγραμμα: ***συνθήκη***

Για παράδειγμα:

```
program MinMax20;
begin
        write(“Δώστε έναν αριθμό : “);   readln(num);
        if num>20 then write(“Μεγαλύτερος από 20”)
                  else write(“Μικρότερος ή ίσος του 20”);
        writeln
end.
```

Πολλές φορές, είναι επιθυμητό να εκτελείται μια εντολή αν ισχύει μια συνθήκη ελέγχου, ενώ όταν δεν ισχύει να μην εκτελείται τίποτε. Σε αυτή την περίπτωση από την εντολή απουσιάζει το **else**.

Για παράδειγμα, αν θέλουμε στο παραπάνω πρόγραμμα να εμφανίζεται μήνυμα στην οθόνη μόνο αν η μεταβλητή num έχει τιμή μεγαλύτερη του 20, τότε η εντολή **if – then – else** θα πρέπει να αντικατασταθεί με την παρακάτω:

```
if num>20 then write(“Μεγαλύτερος από 20”)
```

ΠΡΟΣΟΧΗ!!!

Το παραπάνω πρόγραμμα εμφανίζει μόνο ένα από τα δύο μηνύματα.

Στην περίπτωση που ο έλεγχος είναι ψευδής και δεν υπάρχει **else** τότε το πρόγραμμα συνεχίζει να εκτελεί τις εντολές που έπονται της **if – then – else**.

ΕΝΤΟΛΗ **loop**

Η δομή της εντολής είναι:

```
loop begin
        εντολές
        if συνθήκη then exit;
        εντολές
end
```

Η εντολή **loop** εκτελεί τις εντολές που βρίσκονται στο σώμα, έως ότου εκτελεστεί η εντολή **exit**. Η διαφορά της με τις άλλες δύο εντολές επανάληψης είναι ότι δεν είναι εκ των προτέρων γνωστό πόσες φορές θα εκτελεστούν οι εντολές που βρίσκονται στο σώμα.

Συντακτικό διάγραμμα της εντολής ***loop***

ΠΡΟΣΟΧΗ!!!

Στο σώμα της εντολής **loop** πρέπει πάντα να υπάρχει η εντολή **exit** διαφορετικά οι εντολές θα εκτελούνται επ' άπειρο (endless loop). Η δημιουργία endless loop αποτελεί συχνό προγραμματιστικό λάθος.

Ασκήσεις εξοικείωσης

1. Να γραφεί πρόγραμμα το οποίο δέχεται την ταχύτητα ενός αυτοκινήτου σε μέτρα/δευτερόλεπτο και την μετατρέπει σε χιλιόμετρα/ώρα.

```
program Velocity;
begin
        write("Δώσε την ταχύτητα του αυτοκινήτου : "); readln(velocity1);
        velocity2 := velocity1 * 3600 / 1000;                    (*Μετατροπή*)
        write("Η ταχύτητα σε χιλιόμετρα/ώρα είναι : ",velocity2); writeln
end.
```

2. Να γραφεί πρόγραμμα που θα βρίσκει όλους τους τριψήφιους xyz από το 100 μέχρι το 999 για τους οποίους να ισχύει x<y<z.

   Το πρόγραμμα θα αναπτυχθεί με τρεις τρόπους. Στον πρώτο από αυτούς δημιουργούνται όλοι οι πιθανοί τριψήφιοι αριθμοί και έπειτα ελέγχονται, έτσι ώστε να εμφανιστούν στην οθόνη μόνο εκείνοι που πληρούν την ιδιότητα: x<y<z. Αυτός ο τρόπος δεν είναι αποδοτικός.

   Στο δεύτερο τρόπο, εξαιρούνται οι αριθμοί 8 και 9 για το αριστερότερο ψηφίο του αριθμού, οι αριθμοί 1 και 9 για το μεσαίο ψηφίο του αριθμού και οι αριθμοί 1 και 2 για το δεξιότερο ψηφίο του αριθμού. Αυτή είναι μια βελτίωση, διότι είναι εκ των προτέρων γνωστό ότι κανένας τριψήφιος αριθμός με τα παραπάνω ψηφία στις αντίστοιχες θέσεις δεν πληροί την ιδιότητα x<y<z.

   Στον τρίτο, τα ψηφία που μπορεί να έχει ένας αριθμός στη δεύτερη θέση εξαρτώνται από το ψηφίο που έχει ήδη επιλεγεί για την πρώτη θέση. Συγκεκριμένα, για τη δεύτερη θέση επιλέγονται ψηφία μεγαλύτερα από το ψηφίο της πρώτης θέσης. Όμοια, για την τρίτη θέση επιλέγονται ψηφία μεγαλύτερα από το ψηφίο της δεύτερης. Με αυτό τον τρόπο χρειάζεται να χρησιμοποιήσουμε εντολές **if – then – else**. Αυτός είναι ο πιο αποδοτικός τρόπος.

   Πρώτος τρόπος

```
program ThreeDigitNums1;
begin
        for x:=1 to 9 do                                    (*Κατάλληλα ψηφία στη θέση x*)
          for y:=0 to 9 do                                  (*Κατάλληλα ψηφία στη θέση y*)
            for z:=0 to 9 do                                (*Κατάλληλα ψηφία στη θέση z*)
              if x<y then if y<z then                       (*Έλεγχοι ζητούμενης ιδιότητας*)
                                                  (*Υπολογισμός αριθμού από τα ψηφία του*)
                  begin  num := x * 100 + y * 10 + z;  write(num);  writeln  end
end.
```

   Δεύτερος τρόπος

```
program ThreeDigitNums2;
begin
        for x:=1 to 7 do                                    (*Κατάλληλα ψηφία στη θέση x*)
          for y:=2 to 8 do                                  (*Κατάλληλα ψηφία στη θέση y*)
            for z:=3 to 9 do                                (*Κατάλληλα ψηφία στη θέση z*)
              if x<y then if y<z then                       (*Έλεγχοι ζητούμενης ιδιότητας*)
                                                  (*Υπολογισμός αριθμού από τα ψηφία του*)
                  begin  num := x * 100 + y * 10 + z;  write(num);  writeln  end
end.
```

Τρίτος τρόπος

```
program ThreeDigitNums3;
begin
        for x:=1 to 7 do  begin                          (*Κατάλληλα ψηφία στη θέση x*)
            x1:=x+1;
            for y:=x1 to 8 do   begin                    (*Κατάλληλα ψηφία στη θέση y*)
                y1:=y+1;
                for z:=y1 to 9 do                        (*Κατάλληλα ψηφία στη θέση z*)
                                                  (*Υπολογισμός αριθμού από τα ψηφία του*)
                begin   num := x * 100 + y * 10 + z;   write(num);   writeln   end
            end
        end
end.
```

3. Να γραφεί πρόγραμμα το οποίο θα υπολογίζει το άθροισμα των αριθμών μέχρι το 50, δηλαδή:

$$sum = 1 + 2 + 3 + 4 + ... + 47 + 48 + 49 + 50$$

```
program NaturalSum1;
begin
        sum := 0;                                        (*Αρχικοποίηση*)
        for i:=1 to 50 do   sum := sum + i;
        write(sum);   writeln
end.
```

Παρατήρηση του Gauss: 1 + 50 = 51, 2 + 49 = 51, 3 + 48 = 51, . . .
δηλαδή 25 φορές το 51.

Άρα ισοδύναμο πρόγραμμα:

```
program NaturalSum2;
begin   sum := 25 * 51;   write(sum);   writeln   end.
```

4. Να γραφεί πρόγραμμα το οποίο θα υπολογίζει το άθροισμα των περιττών αριθμών μέχρι το 49, δηλαδή:

$$sum = 1 + 3 + 5 + 7 + ... + 43 + 45 + 47 + 49$$

```
program OddSum1;
begin
        sum := 0;                                              (*Αρχικοποίηση*)
        for i:=1 to 25 do   sum := sum + 2 * i - 1;
        write(sum);   writeln
end.
```

Η μεταβλητή i παίρνει τιμές 1, 2, 3 . . . ,23, 24, 25. Σε κάθε επανάληψη, προστίθεται στο άθροισμα sum η τιμή 2 * i + 1. Η τιμή αυτή για κάθε i ισούται με τον αντίστοιχο περιττό αριθμό.

Παρατήρηση: 1 + 3 = 2 * 2, 1 + 3 + 5 = 3 * 3, 1 + 3 + 5 + 7 = 4 * 4, . . .
δηλαδή sum := 25 * 25

Άρα ισοδύναμο πρόγραμμα:

```
program OddSum2;
begin   sum := 25 * 25;   write(sum);   writeln   end.
```

5. Να γραφεί πρόγραμμα που να υπολογίζει το άθροισμα: $sum = 1 + 3 + 5 + 7 + 9 + 11 + ...$ και σταματά όταν το άθροισμα γίνει μεγαλύτερο από 10000.

```
program OddSum10000a;
begin
        sum := 0;   counter := 0;                              (*Αρχικοποιήσεις*)
        loop begin
                counter := counter + 1;                        (*Ενημέρωση μετρητή*)
                sum := sum + (2 * counter - 1);                (*Πρόσθεση στο άθροισμα*)
                if   sum > 10000  then exit                    (*Έλεγχος για έξοδο*)
        end;
        write("Προστέθηκαν συνολικά ",counter," περιττοί αριθμοί."));   writeln;
        write("Άθροισμα : ",sum);   writeln
end.
```

Για την έξοδο, από την επανάληψη **loop**, στο παραπάνω πρόγραμμα, γίνεται έλεγχος της τιμής της μεταβλητής sum. Αν αυτή είναι μεγαλύτερη από 10000 τότε οι επαναλήψεις σταματούν.
Με βάση την παρατήρηση της προηγούμενης άσκησης, προκύπτει το εξής ισοδύναμο πρόγραμμα:

```
program OddSum10000b;
begin
        counter := 0;                                          (*Αρχικοποιήσεις*)
        loop begin
                counter := counter + 1;                        (*Ενημέρωση μετρητή*)
                sum := counter * counter;                      (*Πρόσθεση στο άθροισμα*)
                if   sum > 10000  then exit                    (*Έλεγχος για έξοδο*)
        end;
        write("Προστέθηκαν συνολικά ",counter," περιττοί αριθμοί."); writeln;
        write("Άθροισμα : ",sum);   writeln
end.
```

6. Να γραφεί πρόγραμμα που να υπολογίζει το άθροισμα:

$$sum = 5^2 + 10^2 + 15^2 + ... + 40^2 + 45^2 + 50^2$$

```
program SquareSum;
begin
        sum := 0;                                              (*Αρχικοποίηση*)
        for i:=1 to 10 do   sum := sum + 25 * i * i;           (*  (5*i) * (5*i) = 25*i*i  *)
        write(sum);   writeln
end.
```

Και για αυτό το άπροισμα υπάρχει απλούστερος τρόπος υπολογισμού.

7. Να γραφεί πρόγραμμα που να υπολογίζει το άθροισμα: $sum = 5^2 + 10^2 + 15^2 + ...$ και σταματά όταν το άθροισμα γίνει μεγαλύτερο από 10000.

```
program SquareSum10000;
begin
        sum := 0;   counter := 0;                              (*Αρχικοποιήσεις*)
        loop begin
                counter := counter + 5;                        (*Ενημέρωση μετρητή*)
                sum := sum + (counter * counter);              (*Πρόσθεση στο άθροισμα*)
                if  sum > 10000  then exit                     (*Έλεγχος για έξοδο*)
        end;
        write(sum);   writeln
end.
```

8. Να γραφτεί πρόγραμμα που να υπολογίζει το άθροισμα: $Sum = 1+2+4+7+11+16+22+29...$ για τους πρώτους 20 όρους.
Υπόδειξη: Προσπαθήστε να κατανοήσετε τον τρόπο με τον οποίο προκύπτουν οι όροι του αθροίσματος.

```
program Sum;
begin
        term := 0;   sum:=0;
        for  i := 1  to 20 do   begin   term := term + i;   sum := sum + term   end;
        write(sum);   writeln
end.
```

Ασκήσεις – Εφαρμογές

1. Να γραφεί πρόγραμμα που να υπολογίζει το γινόμενο των περιττών αριθμών μέχρι και το 9, δηλαδή: $p = 1*3...7*9$

Κεφάλαιο 1.3 – Σύγκριση Δύο Αριθμών

1.3.1 Να γραφτεί πρόγραμμα το οποίο να δέχεται δύο αριθμούς και να τους εμφανίζει στην οθόνη με το κατάλληλο σύμβολο ανισότητας ανάμεσά τους.

```
program Compare;
begin

        write("Δώστε τον πρώτο αριθμό : "); readln(num1);
        write("Δώστε το δεύτερο αριθμό : "); readln(num2);
        if num1>num2 then write(num1," > ",num2)
                     else if num1<num2 then write(num1," < ",num2)
                                       else write(num1," = ",num2);
        writeln
end.
```

Αξίζει να σημειωθεί ότι το παραπάνω πρόγραμμα περιέχει δύο εντολές **if – then – else**. Η δεύτερη από αυτές είναι τοποθετημένη μετά το **else** της πρώτης, πράγμα που σημαίνει ότι θα εκτελεστεί μόνο αν δεν ισχύει η συνθήκη της πρώτης, δηλαδή αν num1<=num2.

Όταν εντολές **if – then – else** είναι τοποθετημένες κατά αυτό τον τρόπο τότε λέμε ότι η μία είναι **φωλιασμένη** μέσα στην άλλη.

Ασκήσεις – Εφαρμογές

1. Να γράψετε πρόγραμμα το οποίο να δέχεται δύο αριθμούς και να τυπώνει τη διαφορά τους.

2. Να τροποποιήσετε το πρόγραμμα της άσκησης 1, χρησιμοποιώντας επανάληψη, έτσι ώστε ο χρήστης να επιλέγει πόσες φορές θα εκτελεστεί.

3. Να γράψετε πρόγραμμα το οποίο να δέχεται τρεις αριθμούς και να τυπώνει το άθροισμα των δύο πρώτων αν ο τρίτος είναι μεγαλύτερος του 100 και τη διαφορά των δύο πρώτων αν ο τρίτος είναι μικρότερος ή ίσος του 100.

4. Να τροποποιήσετε το πρόγραμμα της άσκησης 3 χρησιμοποιώντας επανάληψη, έτσι ώστε ο χρήστης να επιλέγει πόσες φορές θα εκτελεστεί.

5. Να γραφτούν προγράμματα για τις παρακάτω μετατροπές μονάδων μέτρησης:
   - Μέτρα σε εκατοστά
   - Μέτρα σε χιλιοστά
   - Χιλιοστά σε εκατοστά
   - Δραχμές σε Ευρώ
   - Κιλά σε γραμμάρια
   - Γραμμάρια σε κιλά

6. Να γράψετε πρόγραμμα το οποίο να δέχεται δύο αριθμούς και να τυπώνει το string ’Mathematics’ τόσες φορές όση η διαφορά τους και έπειτα το string ’Computer Programming’ τόσες φορές όσο το άθροισμά τους.

7. Να γράψετε πρόγραμμα το οποίο να δέχεται έναν αριθμό και να τυπώνει το μήνυμα:
   a. Zero, αν ο αριθμός ίσος με μηδέν.
   b. One, αν ο αριθμός ίσος με ένα.
   c. Two, αν ο αριθμός ίσος με δύο.
   d. Greater than two, αν ο αριθμός είναι μεγαλύτερος του δύο.

Κεφάλαιο 1.4 – Στρογγυλοποίηση των Αριθμών

1.4.1 Να γραφτεί πρόγραμμα το οποίο να δέχεται έναν αριθμό και να τον εμφανίζει στην οθόνη στρογγυλοποιημένο στη δεκάδα.

```
program DecimalRounding10a;
begin
        write(“ Δώστε έναν αριθμό :”);   readln(num);
        lastdigit := num mod 10;                                  (*Last decimal digit*)
        nwoutld := num – lastdigit;                  (*Number without last decimal digit*)
        if lastdigit>4 then nwoutld := nwoutld + 10;               (*Στρογγυλοποίηση*)
        write(“Αποτέλεσμα : “, nwoutld);   writeln
end.
```

Το παραπάνω πρόγραμμα χρησιμοποιεί την πράξη **mod**. Η πράξη αυτή δίνει το υπόλοιπο της διαίρεσης δύο ακέραιων αριθμών, για παράδειγμα:

12 **mod** 7 = 5, 33 **mod** 6 = 3, 47 **mod** 5 = 2

Η πράξη **mod**, όπως χρησιμοποιείται στο πρόγραμμα έχει σαν αποτέλεσμα να αποθηκευθεί στη μεταβλητή lastdigit το τελευταίο ψηφίο του αριθμού που περιέχει η μεταβλητή num. Έπειτα, η μεταβλητή nwoutld (number without last digit) παίρνει την τιμή της διαφοράς των μεταβλητών num και lastdigit. Τέλος, γίνεται έλεγχος αν το τελευταίο ψηφίο του αριθμού που έδωσε ο χρήστης είναι μεγαλύτερος του 4. Τότε, ο αριθμός στρογγυλοποιείται στην επόμενη δεκάδα, γι’ αυτό και το αποτέλεσμα αυξάνεται κατά 10.

Με όμοιο τρόπο μπορούν να κατασκευαστούν προγράμματα που στρογγυλοποιούν έναν αριθμό στη μονάδα, στην εκατοντάδα και ούτω καθεξής.

Υπάρχουν και οι έτοιμες συναρτήσεις **round(x)** και **trunc(x)**, οι οποίες χρησιμοποιούνται για στρογγυλοποίηση στη μονάδα και αποκοπή στη μονάδα αντίστοιχα.

Κατά την αποκοπή στη μονάδα, διαγράφονται τα ψηφία του αριθμού δεξιά της υποδιαστολής και το ψηφίο των μονάδων δε μεταβάλλεται.

Παραδείγματα:

round(4.33) = 4 **round(4.53) = 5**

trunc(4.33) = 4 **trunc(4.53) = 4**

1.4.2 Να γραφτεί πρόγραμμα το οποίο να δέχεται έναν αριθμό και να τον εμφανίζει στην οθόνη στρογγυλοποιημένο στο εκατοστό.

Το ζητούμενο πρόγραμμα υλοποιείται με παρόμοιο τρόπο με το προηγούμενο. Η μεταβλητή στην οποία αποθηκεύτηκε ο αριθμός πολλαπλασιάζεται επί χίλια, έτσι ώστε η υποδιαστολή να μετατοπιστεί τρεις θέσεις δεξιά. Έπειτα γίνεται στρογγυλοποίηση στη δεκάδα. Τέλος, το αποτέλεσμα διαιρείται με χίλια ώστε να προκύψει ο αρχικός αριθμός στρογγυλοποιημένος στο εκατοστό.

```
program DecimalRounding01a;
begin
        write(“ Δώστε έναν αριθμό :”);   readln(num);
        num := num *1000;   num := trunc(num);
        lastdigit := num mod 10;   nwoutld := num – lastdigit;
        if lastdigit>4 then nwoutld := nwoutld + 10;
        nwoutld := nwoutld /1000;
        write(“Αποτέλεσμα : “, nwoutld);   writeln
end.
```

Το ίδιο πρόγραμμα μπορεί να υλοποιηθεί και με χρήση της έτοιμης συνάρτησης **round( ):**

```
program DecimalRounding01b;
begin
        write(“ Δώστε έναν αριθμό :”);   readln(num);
        num := num * 100;   num := round(num);   num := num / 100;
        write(“Αποτέλεσμα : “, num);   writeln
end.
```

Ασκήσεις.

1. Να γραφτεί πρόγραμμα το οποίο να δέχεται έναν αριθμό και να τον τυπώνει στρογγυλοποιημένο:
   - στη μονάδα
   - στην εκατοντάδα
   - στη χιλιάδα

Κεφάλαιο 1.5 – Η Έννοια της Μεταβλητής

1.5.1 Το παρακάτω πρόγραμμα δέχεται ως είσοδο την τιμή που κοστίζει μια τηλεφωνική μονάδα. Έπειτα, δέχεται το πλήθος των μονάδων μιας τηλεφωνικής συνδιάλεξης και υπολογίζει το κόστος της. Κατόπιν, επιστρέφει στην κατάσταση αναμονής του πλήθους των μονάδων τηλεφωνικής συνδιάλεξης και η διαδικασία συνεχίζεται έτσι, μέχρι ο χρήστης να εισάγει τον αριθμό 0.

```
program Telephone;
begin
        write("Τιμή μονάδας (σε λεπτά) : ");   readln(unitcost);
        loop begin
                writeln;
                write("Πλήθος μονάδων : ");   readln(units);
                if units=0 then exit;
                cost := unitcost * units;
                write("Κόστος συνδιάλεξης : ",cost," λεπτά")
        end
end.
```

Για την έξοδο, από την επανάληψη **loop**, στο παραπάνω πρόγραμμα, γίνεται έλεγχος της τιμής που εισάγει ο χρήστης. Αν αυτή είναι 0 τότε η επανάληψη σταματά.

1.5.2 Να γραφτεί πρόγραμμα το οποίο να υπολογίζει την αξία διαδρομών με ταξί, αν υποθέσουμε ότι η αξία της σημαίας είναι 90 λεπτά και πληρώνουμε 20 λεπτά για κάθε χιλιόμετρο απλής ταρίφας και 36 λεπτά για κάθε χιλιόμετρο διπλής ταρίφας. Το πρόγραμμα να σταματά όταν δίνονται ως είσοδος 0 χιλιόμετρα απλής ταρίφας και 0 χιλιόμετρα διπλής ταρίφας.

```
program Taxi;
begin
        loop begin
                write("Χιλιόμετρα απλής ταρίφας : ");   readln(simpletariffkm);
                write("Χιλιόμετρα διπλής ταρίφας : ");   readln(doubletariffkm);
                if simpletariffkm =0 then if doubletariffkm =0 then exit;
                cost := 90 + simpletariffkm * 20 + doubletariffkm * 36;
                write("Κόστος διαδρομής : ",cost," λεπτά");   writeln
        end
end.
```

Ασκήσεις.

1. Να γραφτεί πρόγραμμα το οποίο να δέχεται αριθμούς και να υπολογίζει το πενταπλάσιο τους, έως ότου δοθεί ο αριθμός 7654321.

2. Να γραφτεί πρόγραμμα το οποίο να δέχεται ως είσοδο το χρόνο που κινήθηκε ένα αυτοκίνητο και την ταχύτητά του και να υπολογίζει την απόσταση που διένυσε.

3. Ένα εστιατόριο έχει τις εξής τιμές:
   - Κουβέρ : 2 ευρώ
   - Τιμή Μενού 1 : 15 ευρώ
   - Τιμή Μενού 2 : 20 ευρώ
   - Τιμή Φιάλης Κρασιού : 10 ευρώ

   Να γραφτεί πρόγραμμα το οποίο να δέχεται ως είσοδο το πλήθος των ατόμων που δείπνησαν, των ατόμων που επέλεξαν το Μενού 1, των ατόμων που επέλεξαν το Μενού 2 και το πλήθος των φιαλών κρασιού που κατανάλωσαν και να υπολογίζει το λογαριασμό που θα πρέπει να πληρώσουν.

Κεφάλαιο 1.6 – Η Έννοια της Εξίσωσης

1.6.1 Το παρακάτω πρόγραμμα δέχεται ως είσοδο έναν αριθμό και ελέγχει αν αυτός ο αριθμός είναι λύση της εξίσωσης x + 4 = 16.

```
program Equation;
begin
                                        (*Εξήγηση λειτουργίας προγράμματος *)
    write("Αυτό το πρόγραμμα ελέγχει αν η είσοδος είναι λύση"); writeln;
    write("της εξίσωσης  x + 4 = 16.  Δώστε είσοδο : ");   readln(x);
    if x+4=16 then write("Το ",x," είναι λύση της εξίσωσης.")
              else write("Το ",x," δεν είναι λύση της εξίσωσης.");
    writeln
end.
```

1.6.2 Το παρακάτω πρόγραμμα λύνει την εξίσωση x + a = b. Δέχεται ως είσοδο την τιμή το a και του b και υπολογίζει την τιμή του x.

```
program Equation1;
begin
        write("Αυτό το πρόγραμμα λύνει την εξίσωση  x + a = b."); writeln;
        write("Δώστε την τιμή του a :"); readln(a);
        write("Δώστε την τιμή του b :"); readln(b);
        x := b – a;  write("Αποτέλεσμα :  x = ",x);  writeln
end.
```

1.6.3 Το παρακάτω πρόγραμμα λύνει την εξίσωση $\frac{x}{a} = b$. Δέχεται ως είσοδο την τιμή το a και του b και υπολογίζει την τιμή του x.

```
program Equation2;
begin
        write(" Αυτό το πρόγραμμα λύνει την εξίσωση  x / a = b. "); writeln;
        loop begin                                   (*Ελεγχόμενη εισαγωγή δεδομένων*)
                write("Δώστε τιμή για το a, όχι όμως μηδέν :"); readln(a);
                if a<>0 then exit
        end;
        write("Δώστε την τιμή του b :"); readln(b);
        x := a * b;  write("Αποτέλεσμα :  x = ",x);  writeln
end.
```

Ως γνωστό, το a δεν επιτρέπεται να έχει την τιμή 0, γιατί είναι παρονομαστής σε κλάσμα. Στην περίπτωση που δοθεί στο a η τιμή 0, τότε το πρόγραμμα ζητά νέα τιμή για το a. Βγαίνει από το **loop** μόνο όταν δοθεί για το a τιμή διαφορετική του μηδενός.

Η μέθοδος που εφαρμόζεται σε αυτό το πρόγραμμα για την αποφυγή της τιμής 0, εφαρμόζεται ευρύτερα όταν θέλουμε να αποφύγουμε να δοθεί μία συγκεκριμένη τιμή σε μια μεταβλητή.

Ασκήσεις.

1. Να γραφτεί πρόγραμμα για τη λύση της εξίσωσης x - a = b. Να δέχεται ως είσοδο τις τιμές του a και του b και να υπολογίζει την τιμή του x.

2. Να γραφτεί πρόγραμμα για τη λύση της εξίσωσης x * a = b. Να δέχεται ως είσοδο τις τιμές του a και του b και να υπολογίζει την τιμή του x.

3. Να γραφτεί πρόγραμμα για τη λύση της εξίσωσης $\frac{a*x}{b} = c$. Να δέχεται ως είσοδο τις τιμές του a, του b και του c και να υπολογίζει την τιμή του x.

4. Να γραφτεί πρόγραμμα για τη λύση της εξίσωσης a * x + b = c. Να δέχεται ως είσοδο τις τιμές του a, του b και του c και να υπολογίζει την τιμή του x.

Κεφάλαιο 1.7 – Πρόσθεση Φυσικών Αριθμών

1.7.1 Το παρακάτω πρόγραμμα προσθέτει τους αριθμούς που δέχεται ως είσοδο. Η πρόσθεση σταματά και το αποτέλεσμα εμφανίζεται στην οθόνη, όταν δοθεί ως είσοδος ο αριθμός 0.

```
program Addition;
begin
        write("Αυτό το πρόγραμμα προσθέτει τους αριθμούς που δίνονται."); writeln;
        i := 1;   sum := 0;
        loop begin
                write(i,"ος προσθετέος :");   readln(s);
                if s=0 then exit;
                sum := sum + s;   i := i + 1
        end;
        i := i – 1;
        write("Προστέθηκαν ",i," αριθμοί."); writeln;
        write("Το αποτέλεσμα είναι : ",sum); writeln
end.
```

Το παραπάνω πρόγραμμα λειτουργεί ως εξής: Πριν την είσοδο στο **loop**, δίνεται αρχική τιμή στις μεταβλητές i και sum, ή αλλιώς οι μεταβλητές αυτές αρχικοποιούνται. Στην μεταβλητή sum αποθηκεύεται το άθροισμα των αριθμών που πληκτρολογεί ο χρήστης και στη μεταβλητή i αποθηκεύεται ο αύξων αριθμός του προσθετέου που πληκτρολογείται κάθε φορά. Μέσα στο **loop**, κάθε φορά που ο

χρήστης πληκτρολογεί έναν αριθμό, ελέγχεται αν είναι 0. Αν είναι, τότε ενεργοποιείται το **exit** και η εκτέλεση συνεχίζεται μετά το σώμα της **loop**. Αν δεν είναι μηδέν, τότε ο αριθμός προστίθεται στο άθροισμα και ο αύξων αριθμός μεγαλώνει κατά 1.

1.7.2 Το παρακάτω πρόγραμμα δέχεται ως είσοδο το μήκος της πλευράς ενός τετραγώνου και υπολογίζει την περίμετρό του.

```
program SquarePerimeter;
begin
        write(“Μήκος πλευράς τετραγώνου : “);   readln(a);
        per := 4 * a;
        write(“Περίμετρος τετραγώνου : “,per);   writeln
end.
```

Ασκήση.

1. Να γραφτεί πρόγραμμα για τον υπολογισμό της περιμέτρου ορθογωνίου παραλληλογράμμου. Να δέχεται ως είσοδο το μήκος και το πλάτος του.

ΠΡΟΓΡΑΜΜΑ ΓΙΑ ΤΟ ΙΣΤΟΡΙΚΟ ΣΗΜΕΙΩΜΑ «ΟΙ ΤΡΙΓΩΝΟΙ ΑΡΙΘΜΟΙ»

Το παρακάτω πρόγραμμα δέχεται ως είσοδο έναν αριθμό και υπολογίζει τον αντίστοιχο τρίγωνο αριθμό. Για παράδειγμα, με είσοδο 3 υπολογίζει τον τρίτο τρίγωνο αριθμό δηλαδή έχει έξοδο 6.

```
program TriangleNumbers;                                    (*Gauss*)
begin
        write(“Δώστε έναν αριθμό : “);   readln(a);
        sum := 0;
        for i:=1 to a do sum := sum + i;
        write(a,”ος τρίγωνος αριθμός : “,sum);   writeln
end.
```

Κεφάλαιο 1.8 – Αφαίρεση Φυσικών Αριθμών

1.8.1 Το παρακάτω πρόγραμμα εκτελεί αφαίρεση. Δέχεται ως είσοδο τον μειωτέο και τον αφαιρετέο και τυπώνει στην οθόνη τη διαφορά. Στην περίπτωση που ο αφαιρετέος είναι μεγαλύτερος από το μειωτέο, το πρόγραμμα μπαίνει σε **loop** και ζητά νέο αφαιρετέο, μέχρι να δοθεί αφαιρετέος μικρότερος ή ίσος του μειωτέου.

```
program Subtraction;
begin
        write(“Αυτό το πρόγραμμα εκτελεί αφαίρεση.”);   writeln;
        write(“Μειωτέος : “);   readln(num1);
        loop begin
                write(“Αφαιρετέος όχι όμως μεγαλύτερος από το μειωτέο: “);
                readln(num2);
                if  num2 <= num1  then exit
        end;
        dif := num1 – num2;
        write(“Διαφορά :”,dif);   writeln
end.
```

1.8.2 Το παρακάτω πρόγραμμα δέχεται ως είσοδο ένα σύμβολο (+ ή –) και δύο αριθμούς και τους προσθέτει ή τους αφαιρεί, ανάλογα με το σύμβολο. Στην περίπτωση της αφαίρεσης, γίνεται έλεγχος, αν ο μειωτέος είναι πράγματι μεγαλύτερος ή ίσος με του αφαιρετέου. Το πρόγραμμα αυτό είναι συνδυασμός των προγραμμάτων 1.8.1 και 1.7.1.

```
program AdditionSubtraction;
begin
        write(“Αυτό το πρόγραμμα εκτελεί πρόσθεση ή αφαίρεση.”);   writeln;
        loop begin
                write(“Δώστε 1 για + ή 2 για – : “);   readln(symbol);
                if  symbol = 1  then exit;   if  symbol = 2  then exit
        end;
        write(“Αριθμός 1 : “);   readln(num1);
        loop begin
              write(“Αριθμός 2 : “);   readln(num2);
              if  symbol = 1  then exit;
              if num2 <= num1 then exit
              else write(“Ο μειωτέος, δεν μπορεί να είναι μικρότερος από τον αφαιρετέο. ”)
        end;
        if symbol=1 then result:=num1+num2 else result:=num1-num2;
        write(“Αποτέλεσμα :”,result);   writeln
end.
```

Κεφάλαιο 1.9 – Πολλαπλασιασμός Φυσικών Αριθμών

1.9.1 Το παρακάτω πρόγραμμα επαληθεύει τον ορισμό του πολλαπλασιασμού, δηλαδή ότι ισχύει:

$$1,5 + 1,5 + 1,5 + 1,5 + 1,5 + 1,5 + 1,5 = 7 * 1,5$$

```
program Multiplication;
begin
        write(“Το πρόγραμμα επαληθεύει τον ορισμό του”);   writeln;
        write(“ πολλαπλασιασμού, δηλαδή οτι διαδοχικές προσθέσεις”);   writeln;
        write(“ισοδυναμούν με πολλαπλασιασμό.”);   writeln;
        write(“Δώστε ακέραιο αριθμό a : ”);  readln(a);
        write(“Δώστε αριθμό b : ”);  readln(b);
        write(“Υπολογισμός “,b,” + . . . +”,b,” , “,a,” φορές.”);  writeln;
        total := 0;
        for a times do total := total + b;
        write(“Αποτέλεσμα : “, total);   writeln;
        write(“Υπολογισμός “,a,” * ”,b);   writeln;
        product := a * b;
        write(“Αποτέλεσμα : “, product);   writeln;
        if total= product then write(“Τα αποτελέσματα είναι ίσα!”)
                                else write(“Τα αποτελέσματα δεν είναι ίσα!”);
        writeln
end.
```

1.9.2 Το παρακάτω πρόγραμμα εκτελεί πολλαπλασιασμό. Δέχεται ως είσοδο τους παράγοντες και εμφανίζει στην οθόνη το γινόμενο.

```
program Multiplication1;
begin
        write(“Αυτό το πρόγραμμα εκτελεί πολλαπλασιασμό.”);   writeln;
        write(“Παράγοντας 1 : “);   readln(factor1);
        write(“ Παράγοντας 2 : “);   readln(factor2);
        result := factor1 * factor2;
        write(“Γινόμενο : “, result);   writeln
end.
```

1.9.3 Το παρακάτω πρόγραμμα έχει λειτουργία παρόμοια με αυτή μιας ταμειακής μηχανής ενός supermarket. Δέχεται ως δεδομένα τις ποσότητες και τις τιμές των προϊόντων μέχρι να δοθεί ποσότητα 0 για κάποιο προϊόν, οπότε η λειτουργία σταματάει. Έπειτα, τυπώνεται το άθροισμα στην οθόνη και εισάγεται από το χρήστη το ποσό που έδωσε ο πελάτης. Ύστερα, υπολογίζονται τα ρέστα και το πρόγραμμα είναι έτοιμο να εξυπηρετήσει τον επόμενο πελάτη. Το πρόγραμμα επαναλαμβάνεται επ' άπειρο.

```
program Supermarket;
begin
        loop begin
                write("Λογαριασμός πελάτη.");   writeln;
                sum := 0;   i := 0;
                loop begin
                        i := i + 1;
                        write("Προϊόν ",i,", ποσότητα : ");   readln(quantity);
                        if quantity=0 then exit;
                        write("Προϊόν ",i,", τιμή : ");   readln(price);
                        sum := sum + quantity*price
                end;
                write("Αριθμός διαφορετικών προϊόντων : ",items);   writeln;
                write("Πληρωτέο ποσό : ",sum);   writeln;
                loop begin
                        write("Πληρωμή πελάτη (μεγαλύτερη του πληρωτέου ποσού) : ");
                        readln(money);
                        if sum<=money then exit
                end;
                change := money – sum;
                write("Ρέστα : ",change);   writeln;   writeln
        end
end.
```

Ασκήσεις.

1. Να γραφτεί πρόγραμμα το οποίο να πολλαπλασιάζει τους αριθμούς που δέχεται ως είσοδο. Ο πολλαπλασιασμός να σταματά και το αποτέλεσμα να τυπώνεται στην οθόνη, όταν δοθεί ως είσοδος ο αριθμός 0.

ΠΡΟΓΡΑΜΜΑ ΓΙΑ ΤΟ ΙΣΤΟΡΙΚΟ ΣΗΜΕΙΩΜΑ «ΜΙΑ ΜΙΚΡΗ ΙΔΙΟΦΥΪΑ - GAUSS»

Σύμφωνα με τη μέθοδο του Gauss, για να υπολογίσουμε το άθροισμα διαδοχικών ακέραιων αριθμών, προσθέτουμε τον πρώτο και τον τελευταίο από αυτούς και έπειτα πολλαπλασιάζουμε αυτό το άθροισμα με το ήμισυ της διαφοράς τους, αυξημένης κατά 1.

$$k+(k+1)+(k+2)+.....+(l-2)+(l-1)+l=\frac{(l+k)(l-k+1)}{2}$$

Το παρακάτω πρόγραμμα υπολογίζει το άθροισμα διαδοχικών ακέραιων αριθμών με τη μέθοδο του Gauss. Δέχεται ως είσοδο δύο αριθμούς, το μεγαλύτερο και το μικρότερο αριθμό που συμμετέχουν στο άθροισμα και εξασφαλίζει ότι ο πρώτος από αυτούς είναι μικρότερος ή ίσος με το δεύτερο.

```
program Gauss;
begin
        write(“Άθροισμα διαδοχικών ακεραίων.”);   writeln;
        write(“Δώστε τον πρώτο αριθμό : “);   readln(num1);
        loop begin
                write(“Δώστε τον τελευταίο (και μεγαλύτερο) αριθμό : “);   readln(num2);
                if  num2 >= num1  then exit
        end;
        total := (num2 + num1) * (num2 – num1 + 1) / 2;
        write(“Άθροισμα : “, total);   writeln
end.
```

Κεφάλαιο 1.10 – Πολλαπλάσια Φυσικού Αριθμού

1.10.1 Το παρακάτω πρόγραμμα δέχεται έναν αριθμό και τυπώνει τον πίνακα προπαίδειας του αριθμού αυτού.

```
program MultiplicationTable;
begin
        write("Πίνακας προπαίδειας."); writeln;
        write("Δώστε αριθμό : "); readln(n); writeln;
        for i:=1 to 10 do begin res := i * n; write(i," * ",n," = ",res); writeln end
end.
```

Παρατήρηση: Αντί για τον υπολογισμό res := i * n σε κάθε βήμα, μπορούμε να προσθέτουμε τον αριθμό n στο προηγούμενο αποτέλεσμα, δηλαδή res := res + n, με την προυπόθεση ότι πριν τη **for loop** έχουμε αρχικοποιήσει το res (res := 0).

1.10.2 Το πρόγραμμα που ακολουθεί δέχεται δύο αριθμούς και υπολογίζει το ελάχιστο κοινό πολλαπλάσιό τους. Το πρόγραμμα ακολουθεί την εξής μέθοδο:

Βήμα 1^{ο}: Αντιγράφουμε τους δύο αριθμούς (n1 και n2) σε δύο νέες μεταβλητές (num1 και num2 αντιστοίχως).

Βήμα 2^{ο}: Συγκρίνουμε τις μεταβλητές num1 και num2.

Βήμα 3^{ο}: Αν είναι ίσες, τότε η τιμή τους είναι το ελάχιστο κοινό πολλαπλάσιο.

Βήμα 4^{ο}: Αν δεν είναι ίσες, τότε αν η μικρότερη είναι η num1, την αυξάνουμε κατά n1. Διαφορετικά, δηλαδή αν η μικρότερη είναι η num2, την αυξάνουμε κατά n2.

Βήμα 5^{ο}: Επιστρέφουμε στο 2^{ο} βήμα.

Η παραπάνω μέθοδος, είναι όμοια με την καταγραφή των πολλαπλάσιων των δύο αριθμών, τη διαλογή των κοινών και την επιλογή του μικρότερου από αυτά.

```
program Lcm;
begin
        write("Εύρεση Ε.Κ.Π."); writeln;
        write("Δώστε τον πρώτο αριθμό : "); readln(n1);
        write("Δώστε τον δεύτερο αριθμό : "); readln(n2);
        num1 := n1; num2 := n2;
        loop begin
                if num1 = num2 then exit;
                if num1 < num2 then num1 := num1 + n1
                               else num2 := num2 + n2
        end;
        lcm := num1;
        write("Ελάχιστο κοινό πολλαπλάσιο : ", lcm); writeln
end.
```

1.10.3 Το παρακάτω πρόγραμμα δέχεται δύο αριθμούς. Τυπώνει όλα τα πολλαπλάσια του πρώτου αριθμού που είναι μικρότερα από τον δεύτερο αριθμό.

```
program Mult;
begin
        write("Πολλαπλάσια του αριθμού a μικρότερα του b."); writeln;
        write("Δώστε τον αριθμό a : "); readln(a);
        write("Δώστε τον αριθμό b : "); readln(b);
        num1 := a;
        loop begin
                if num1 > b then exit;
                write(num1); writeln;
                num1 := num1 + a
        end
end.
```

Ασκήσεις.

1. Να γραφτεί πρόγραμμα το οποίο να δέχεται τρεις αριθμούς και να υπολογίζει το ελάχιστο κοινό πολλαπλάσιό τους.

2. Να επαναληφθεί η άσκηση 1, για τέσσερεις αριθμούς.

3. Να γραφτεί πρόγραμμα το οποίο να δέχεται δύο αριθμούς και να τυπώνει όλα τα πολλαπλάσια του πρώτου αριθμού που είναι μεγαλύτερα από το δεύτερο και μικρότερα από τον τρίτο αριθμό.

4. Να γραφτεί πρόγραμμα το οποίο να δέχεται δύο αριθμούς και να υπολογίζει τα πέντε μικρότερα κοινά πολλαπλάσια των αριθμών αυτών.

Κεφάλαιο 1.11 – Δυνάμεις Αριθμών

1.11.1 Το παρακάτω πρόγραμμα δέχεται έναν αριθμό και τυπώνει το τετράγωνό του.

```
program Square;
begin
        write("Δώστε αριθμό: "); readln(n);
        nsquare := n * n;
        write("Το τετράγωνό του είναι : ", nsquare); writeln
end.
```

1.11.2 Το πρόγραμμα που ακολουθεί δέχεται δύο αριθμούς. Ο πρώτος αποτελεί τη βάση και ο δεύτερος τον εκθέτη. Το πρόγραμμα υπολογίζει το αποτέλεσμα της ύψωσης της βάσης στον εκθέτη. Σημειώνεται ότι ο συμβολισμός a^b ισοδυναμεί με τον συμβολισμό a**b. Ο συμβολισμός αυτός χρησιμοποιείται αντί του a^b διότι ο τελευταίος δεν μπορεί να εισαχθεί εύκολα από το πληκτρολόγιο.

```
program Power1;
begin
        write(“Δώστε τη βάση : “);   readln(base);
        write(“Δώστε τον εκθέτη : “);   readln(exponent);
        result := 1;
        for i:=1 to exponent do result:= result * base;
        write(base,” ^ “, exponent,” = “,result);   writeln
end.
```

Υπάρχει η πράξη **, η οποία εκτελεί ύψωση σε δύναμη. Για παράδειγμα: 4**2 = 16
Οπότε, προκύπτει το ισοδύναμο πρόγραμμα:

```
program Power2;
begin
        write(“Δώστε τη βάση : “);   readln(base);
        write(“Δώστε τον εκθέτη : “);   readln(exponent);
        result := base ** exponent;
        write(base,” ^ “, exponent,” = “,result);   writeln
end.
```

1.11.3 Το πρόγραμμα που ακολουθεί δέχεται έναν αριθμό και τον τυπώνει σε αναπτυγμένη μορφή. Για παράδειγμα:

$$2345 = 5 * 1 + 4 * 10 + 3 * 100 + 2 * 1000$$

Σημειώνεται ότι η παραπάνω κανονική μορφή έχει αντίστροφη φορά από αυτήν της θεωρίας. Προτιμάται, όμως, γιατί το πρόγραμμα που την παράγει είναι ευκολότερο.

```
program DecimalRepresentation;
begin
        write("Δώστε ακέραιο αριθμό : ");   readln(num);
        write(num," = ");
        order := 1;
        loop begin
                digit := num mod 10;
                write(digit," * ", order);
                num := num div 10;   order := order * 10;
                if num=0 then exit else write(" + ")
        end
end.
```

Το παραπάνω πρόγραμμα λειτουργεί ως εξής: μετά την εισαγωγή των δεδομένων, για κάθε επανάληψη του **loop**, υπολογίζεται το τελευταίο ψηφίο του αριθμού με την πράξη **mod**. Η πράξη αυτή δίνει το υπόλοιπο της διαίρεσης δύο αριθμών, για παράδειγμα:

12 **mod** 7 = 5, 33 **mod** 6 = 3, 47 **mod** 5 = 2

Στη μεταβλητή order βρίσκεται αποθηκευμένη η τάξη του ψηφίου. Η μεταβλητή rank δεκαπλασιάζεται για να χρησιμοποιηθεί στην επόμενη επανάληψη του σώματος της εντολής **loop**. Ο αριθμός num που εισάγει ο χρήστης μεταβάλλεται σε κάθε επανάληψη ώστε στην επόμενη να υπολογιστεί το ψηφίο της επόμενης τάξης.

Ασκήσεις.

1. Να γραφτεί πρόγραμμα το οποίο να δέχεται έναν αριθμό και να υπολογίζει τον κύβο του.

2. Να γραφτεί πρόγραμμα το οποίο να δέχεται έναν αριθμό a και να τυπώνει τον πίνακα δυνάμεων, παρόμοιο με αυτόν της προπαίδειας. Δηλαδή, με είσοδο 2, το πρόγραμμα πρέπει να τυπώνει:

   2 ^ 1 = 2
   2 ^ 2 = 4
   2 ^ 3 = 8
   2 ^ 4 = 16
   2 ^ 5 = 32
   2 ^ 6 = 64
   2 ^ 7 = 128
   2 ^ 8 = 256
   2 ^ 9 = 512
   2 ^ 10 = 1024

3. Να εξηγήσετε πως συμπεριφέρεται το πρόγραμμα 1.11.2 στην περίπτωση που εισάγουμε εκθέτη 0. Το αποτέλεσμα είναι σωστό;

Κεφάλαιο 1.12 – Επιμεριστική Ιδιότητα

1.12.1 Το παρακάτω πρόγραμμα δέχεται τρεις αριθμούς a, b και c, και επαληθεύει ότι πράγματι ισχύει η επιμεριστική ιδιότητα, δηλαδή: a (b + c) = ab + ac

```
program Distributive;
begin
        write(“Δώστε αριθμό a : “);   readln(a);
        write(“Δώστε αριθμό b : “);   readln(b);
        write(“Δώστε αριθμό c : “);   readln(c);
        result1 := a * ( b + c );
        write(a,” (“,b,” + “,c,”) = “, result1);   writeln;
        result2 := ( a * b ) + ( a * c );
        write(a,” * “,b,” + “,a,” * “,c,” = “, result2);   writeln;
        if result1 = result2  then write(“Ισχύει η επιμεριστική ιδιότητα.”)
                              else write(“Δεν ισχύει η επιμεριστική ιδιότητα.”);
        writeln
end.
```

Ασκήσεις.

1. Να γραφτεί πρόγραμμα το οποίο να επαληθεύει ότι ισχύει:
   (b – c) a = ba – ca

2. Να γραφτεί πρόγραμμα το οποίο να επαληθεύει ότι ισχύει:
   a (b – c) = ab – ac

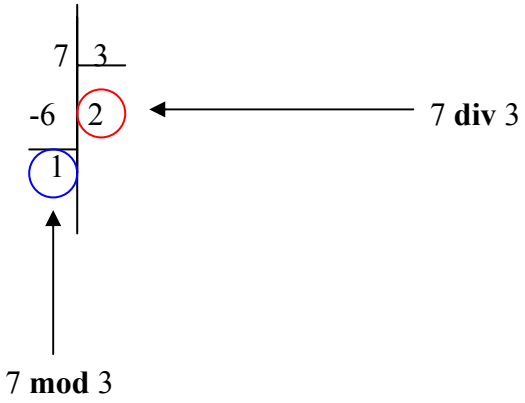
3. Να γραφτεί πρόγραμμα το οποίο να επαληθεύει ότι ισχύει:
   a (b + c + d) = ab + ac + ad

4. Να γραφτεί πρόγραμμα το οποίο να επαληθεύει ότι ισχύει:
   a (b + c + d + e) = ab + ac + ad + ae

5. Να γραφτεί πρόγραμμα το οποίο να επαληθεύει ότι δεν ισχύει:
   a + (b * c) = (a + b) * (a + c)

Κεφάλαιο 1.13 – Η Τέλεια Διαίρεση

Η διαίρεση υλοποιείται με τις πράξεις **div** και /. Η διαφορά μεταξύ τους είναι ότι η πράξη **div** εφαρμόζεται μεταξύ ακέραιων αριθμών ή μεταβλητών και το πηλίκο είναι ακέραιος αριθμός, ενώ η πράξη / εφαρμόζεται μεταξύ πραγματικών αριθμών και το πηλίκο είναι πραγματικός αριθμός. Επειδή οι ακέραιοι αριθμοί είναι και πραγματικοί, η πράξη / μπορεί να εφαρμοστεί μεταξύ ακεραίων αριθμών.
Επομένως, στην περίπτωση που ασχολούμαστε με την ακέραια διαίρεση θα χρησιμοποιήσουμε την πράξη **div.** Όπως έχουμε ήδη αναφέρει, το υπόλοιπο μιας διαίρεσης υπολογίζεται με την πράξη **mod** (δείτε το σχήμα). Για να εξασφαλίσουμε ότι μια ακέραια διαίρεση είναι τέλεια, είναι αρκετό να ελέγξουμε ότι έχει υπόλοιπο ίσο με το μηδέν.



1.13.1 Το παρακάτω πρόγραμμα δέχεται δύο αριθμούς, τον διαιρετέο και το διαιρέτη και υπολογίζει το ακέραιο πηλίκο της διαίρεσης. Επίσης, ελέγχει αν η διαίρεση είναι τέλεια. Σημειώνεται ότι πρέπει οπωσδήποτε να περιλαμβάνει έλεγχο για την περίπτωση που ο διαιρέτης, που πληκτρολογεί ο χρήστης, ισούται με το μηδέν, οπότε η διαίρεση δεν μπορεί να γίνει.

```
program PerfectDivision;
begin
        write(“Δώστε ακέραιο διαιρετέο : “);   readln(num);
        loop begin
                write(“Δώστε το διαιρέτη (όμως όχι μηδέν): “);   readln(divider);
                if divider <>0 then exit
        end;
        quotient := num div divider;   remainder := num mod divider;
        write(“Το πηλίκο είναι : “,quotient);   writeln;
        if remainder = 0  then write(“Η διαίρεση είναι τέλεια.”)
                          else write(“Η διαίρεση είναι ατελής.”);
        writeln
end.
```

1.13.2 Ακολουθεί μια παραλλαγή του προγράμματος στρογγυλοποίησης στην δεκάδα (1.4.1), η οποία χρησιμοποιεί την πράξη **div**.

```
program DecimalRounding10b;
begin
        write(" Δώστε έναν αριθμό :");   readln(num);
        lastdigit := num mod 10;   nwoutld := num div 10;
        if lastdigit>4 then nwoutld := nwoutld + 10;
        write("Αποτέλεσμα : ", nwoutld);   writeln
end.
```

1.13.3 Να γραφεί πρόγραμμα το οποίο θα βρίσκει το άθροισμα των ψηφίων ενός τριψήφιου αριθμού.

```
program AddThreeDigits
begin
        loop begin
                write("Δώσε το πολύ έναν τριψήφιο αριθμό");   readln(a);
                if  a < 1000  then exit
        end;
        firstdigit:=a div 100;   a := a mod 100;            (*Υπολογισμός των τριών ψηφίων*)
        seconddigit := a div 10;   thirddigit := a mod 10;
        sum := firstdigit + seconddigit + thirddigit;          (*Πρόσθεση των τριών ψηφίων*)
        write("Το άθροισμα των ψηφίων του αριθμού είναι:",sum);   writeln
end.
```

Κεφάλαιο 1.14 – Διαιρέτες Φυσικού Αριθμού

Γνήσιοι διαιρέτες ενός αριθμού n είναι όλοι οι διαιρέτες του εκτός από τον ίδιο τον αριθμό και τη μονάδα.

Γενικά, **υποψήφιοι γνήσιοι διαιρέτες** ενός αριθμού n είναι όλοι οι αριθμοί που είναι μεγαλύτεροι από τη μονάδα και μικρότεροι από τον αριθμό n.

Αν ένας αριθμός n έχει ένα γνήσιο διαιρέτη a, τότε υπάρχει και ένας δεύτερος b, (όχι απαραίτητα a<>b), τέτοιος ώστε a * b = n. Για παράδειγμα, για n = 24 και a = 3 έχουμε b = 8.

Επίσης, για κάθε αριθμό n, υπάρχει ένας αριθμός x, τέτοιος ώστε $x^2 <= n$ και $(x+1)^2 > n$. Για παράδειγμα για n = 24, προκύπτει x = 4, επειδή: $4^2 <= 24$ και $5^2 > 24$. Δηλαδή, το 5 είναι ο μικρότερος αριθμός που το τετράγωνό του είναι μεγαλύτερο του 24

Ένας από τους a, b είναι μικρότερος ή ίσος του x, διότι αν ήταν και οι δύο μεγαλύτεροι του x τότε το γινόμενο τους θα ήταν τουλάχιστον $(x+1)^2$ άρα μεγαλύτερο του n, σύμφωνα με τα παραπάνω. Στο παράδειγμα, πράγματι 3 <= 4 (a=3 και b=4).

Άρα, για να βρούμε τουλάχιστον ένα γνήσιο διαιρέτη του αριθμού (αν υπάρχει), δε χρειάζεται να ελέγξουμε όλους τους αριθμούς που είναι μικρότεροι από αυτόν. Είναι αρκετό να ελέγξουμε τους αριθμούς από 2 έως και τον μικρότερο x, για τον οποίο ισχύει $x^2 > n$.

1.14.1 Το παρακάτω πρόγραμμα υπολογίζει και τυπώνει τους διαιρέτες ενός αριθμού, τον οποίο εισάγει ο χρήστης.

```
program Divisors;
begin
        write("Δώστε έναν αριθμό : "); readln(num);
        divider := 2;
        loop begin
                if divider * divider > num then exit;
                if num mod divider = 0 then begin
                        write (divider); writeln;
                        temp := num div divider;  write (temp); writeln
                end;
                divider := divider +1
        end
end.
```

1.14.2 Για τον υπολογισμό του μέγιστου κοινού διαιρέτη δύο αριθμών έχουν διατυπωθεί διάφορες μέθοδοι. Η πιο απλή από αυτές είναι η εξής:

Βήμα 1ο: Ξεκινάμε από τον μικρότερο από τους δύο αριθμούς.

Βήμα 2ο: Ελέγχουμε αν ο αριθμός είναι κοινός διαιρέτης, δηλαδή αν η διαίρεση καθενός από τους δύο αριθμούς με αυτόν είναι τέλεια.

Βήμα 3ο: Αν ναι, το πρόγραμμα τελειώνει. Αν όχι, μειώνουμε τον αριθμό κατά μία μονάδα και επιστρέφουμε στο 2ο βήμα.

Το παρακάτω πρόγραμμα ακολουθεί την παραπάνω μέθοδο:

```
program Gcd1;
begin
        write("Υπολογισμός του μέγιστου κοινού διαιρέτη."); writeln;
        write("Δώστε τον αριθμό a : "); readln(num1);
        write("Δώστε τον αριθμό b : "); readln(num2);
        if num1 > num2 then min := num2 else min := num1;
        loop begin
                if num1 mod min = 0 then if num2 mod min = 0 then exit;
                min := min – 1
        end;
        write("Μέγιστος κοινός διαιρέτης : ",min); writeln
end.
```

1.14.3 Η παραπάνω μέθοδος και το αντίστοιχο πρόγραμμα μπορούν να βελτιωθούν με την εξής ιδέα: Έχει αποδειχθεί ότι ο Μ.Κ.Δ. δύο αριθμών ισούται με το Μ.Κ.Δ. του μικρότερου από αυτούς και της διαφοράς τους. Δηλαδή:

- Αν a>b τότε Μ.Κ.Δ.(a, b) = Μ.Κ.Δ.(a – b, b)
- Αν a<b τότε Μ.Κ.Δ.(a, b) = Μ.Κ.Δ.(a, b – a)

Η μέθοδος μοιάζει πολύ με τη μέθοδο που ακολουθείται για τον προσδιορισμό του ελάχιστου κοινού πολλαπλάσιου στο πρόγραμμα 1.10.2. Με βάση τα παραπάνω, μπορούμε να κατασκευάσουμε την παρακάτω μέθοδο:

Βήμα 1ο: Αν οι δύο αριθμοί είναι ίσοι τότε είναι ίσοι και με το Μ.Κ.Δ.

Βήμα 2ο: Αν οι δύο αριθμοί δεν είναι ίσοι τότε αντικαθιστούμε τον μεγαλύτερο από αυτούς με τη διαφορά τους και επαναλαμβάνουμε το 1ο βήμα.

Ακολουθεί το πρόγραμμα:

```
program Gcd2;
begin
        write("Υπολογισμός του μέγιστου κοινού διαιρέτη."); writeln;
        write("Δώστε τον αριθμό a : "); readln(a);
        write("Δώστε τον αριθμό b : "); readln(b);
        loop begin
                if a = b then exit;        (*Αντικατάσταση του μεγαλύτερου με τη διαφορά*)
                if a > b then a := a – b else b := b – a
        end;
        write("Μέγιστος κοινός διαιρέτης : ",a); writeln
end.
```

1.14.4 Η μέθοδος που ακολουθείται για την εύρεση του μέγιστου κοινού διαιρέτη επιδέχεται μία ακόμη βελτίωση (αλγόριθμος του Ευκλείδη). Έχει αποδειχθεί ότι ο Μ.Κ.Δ. δύο αριθμών ισούται με το Μ.Κ.Δ. του μικρότερου από αυτούς και του υπολοίπου της διαίρεσής τους. Αντί δηλαδή για διαδοχικές αφαιρέσεις, κάνουμε διαίρεση και χρησιμοποιούμε το υπόλοιπο. Δηλαδή:

- Αν a>b τότε Μ.Κ.Δ.(a, b) = Μ.Κ.Δ.(a **mod** b, b)
- Αν a<b τότε Μ.Κ.Δ.(a, b) = Μ.Κ.Δ.(a, b **mod** a)

Με βάση τα παραπάνω, κατασκευάζουμε την παρακάτω μέθοδο:

Βήμα 1^{ο}: Αν ένας από τους δύο αριθμούς είναι μηδενικός, τότε ο μέγιστος κοινός διαιρέτης είναι ο άλλος.

Βήμα 2^{ο}: Αν και οι δύο αριθμοί είναι μη μηδενικοί τότε αντικαθιστούμε τον μεγαλύτερο από αυτούς με το υπόλοιπο της ευκλείδειας διαίρεσης τους και επαναλαμβάνουμε το 1^{ο} βήμα.

Ακολουθεί το πρόγραμμα:

```
program Gcd3;
begin
        write("Υπολογισμός του μέγιστου κοινού διαιρέτη."); writeln;
        write("Δώστε τον αριθμό a : "); readln(a);
        write("Δώστε τον αριθμό b : "); readln(b);
        loop begin
                if a = 0 then exit;  if b = 0 then exit;
                                (*Αντικατάσταση του μεγαλύτερου με το πηλίκο της διαίρεσης*)
                if a > b  then  a := a mod b  else  b := b mod a
        end;                    (*Ενας εκ των a, b ισούται με μηδέν, άρα το άθροισμα ισούται*)
        gcd := a + b;                                        (*με τον άλλο, τον μη μηδενικό*)
        write("Μέγιστος κοινός διαιρέτης : ", gcd);  writeln
end.
```

1.14.5 Στο πρόγραμμα 1.10.2, για την εύρεση του ελάχιστου κοινού πολλαπλάσιου, ακολουθήσαμε μια μέθοδο που ξεκινά από τους δύο αριθμούς και ελέγχει διαδοχικά όλα τα πολλαπλάσια τους εως ότου βρει ένα κοινό πολλαπλάσιο. Η μέθοδος αυτή πράγματι προσδιορίζει σωστά το ελάχιστο κοινό πολλαπλάσιο, παρόλα αυτά δεν είναι πολύ αποδοτική όσον αφορά το χρόνο που χρειάζεται.
Μια πιο αποδοτική μέθοδος βασίζεται στην παρακάτω ισότητα:

$$\text{Ε.Κ.Π.}(a,b) * \text{Μ.Κ.Δ.}(a,b) = a * b$$

Ισχύει, δηλαδή, οτι το γινόμενο του ελάχιστου κοινού πολλαπλάσιου και του μέγιστου κοινού διαιρέτη δύο αριθμών ισούται με το γινόμενο των δύο αριθμών. Από αυτή τη σχέση προκύπτει οτι:

$$\text{Ε.Κ.Π.}(a,b) = \frac{a * b}{\text{Μ.Κ.Δ.}(a,b)}$$

Η παραπάνω ισότητα σημαίνει ότι αν γνωρίζουμε τους δύο αριθμούς και τον μέγιστο κοινό διαιρέτη τους, τότε μπορούμε να προσδιορίσουμε και το ελάχιστο κοινό πολλαπλάσιό τους. Συνεπώς, μπορούμε να τροποποιήσουμε το πρόγραμμα 1.14.4, έτσι ώστε να υπολογίζει και το ελάχιστο κοινό πολλαπλάσιο των δύο αριθμών. Το τροποποιημένο πρόγραμμα ακολουθεί. Τονίζεται ότι η μέθοδος που ακολουθεί είναι η αποδοτικότερη από κάθε άλλη που θα μπορούσε να εφαρμοστεί, για τον προσδιορισμό του μέγιστου κοινού διαιρέτη και του ελάχιστου κοινού πολλαπλάσιου.

```
program Gcd4;
begin
        write("Υπολογισμός του μέγιστου κοινού διαιρέτη."); writeln;
        write("Δώστε τον αριθμό a : "); readln(a);
        write(" Δώστε τον αριθμό b : "); readln(b);
        loop begin
                if a = 0 then exit; if b = 0 then exit;
                if a > b then a := a mod b else b := b mod a
        end;
        gcd := a + b;  lcm := a * b div gcd;
        write("Μέγιστος κοινός διαιρέτης : ", gcd); writeln;
        write("Ελάχιστο κοινό πολλαπλάσιο : ", lcm); writeln
end.
```

Ασκήσεις.

1. Να δικαιολογήσετε γιατί το πρόγραμμα 1.14.3 είναι αποδοτικότερο από το πρόγραμμα 1.14.2 και γιατί το πρόγραμμα 1.14.4 είναι αποδοτικότερο από το πρόγραμμα 1.14.3.

2. Να γραφτεί πρόγραμμα το οποίο να υπολογίζει το μέγιστο κοινό διαιρέτη και το ελάχιστο κοινό πολλαπλάσιο τριών αριθμών.

3. Να γραφτεί πρόγραμμα το οποίο να τυπώνει τους αριθμούς που είναι μεγαλύτεροι της μονάδας και μικρότεροι ή ίσοι του 100, οι οποίοι δεν είναι πολλαπλάσια του 2, του 3, του 5 και του 7. Ζητείται να κατασκευαστεί, δηλαδή, το κόσκινο του Ερατοσθένη.

Κεφάλαιο 1.15 – Χαρακτήρες Διαιρετότητας

1.15.1 Το παρακάτω πρόγραμμα δέχεται έναν αριθμό και αποφαίνεται αν αυτός είναι πολλαπλάσιο του 2. Υλοποιείται με βάση το κριτήριο διαιρετότητας, σύμφωνα με το οποίο ένας αριθμός είναι πολλαπλάσιο του 2 αν λήγει σε 0, 2, 4, 6 ή 8.

```
program Division2a;
begin
        write(“Δώστε έναν αριθμό : “);   readln(num);
        lastdigit := num mod 10;   k := 0;
        if lastdigit=0 then k:=1
                else if lastdigit=2 then k:=1
                        else if lastdigit=4 then k:=1
                                else if lastdigit=6 then k:=1
                                        else if lastdigit=8 then k:=1;
        if k=1 then write(“Ο αριθμός είναι πολλαπλάσιο του 2”)
              else write(“ Ο αριθμός δεν είναι πολλαπλάσιο του 2”);
        writeln
end.
```

1.15.2 Αν δε θεωρηθεί απαραίτητη η χρήση του κριτηρίου διαιρετότητας, τότε το πρόγραμμα μπορεί να γραφτεί ως εξής:

```
program Division2b;
begin
        write(“Δώστε έναν αριθμό : “);   readln(num);
        if  num mod 2 = 0  then write(“Ο αριθμός είναι πολλαπλάσιο του 2”)
                        else write(“ Ο αριθμός δεν είναι πολλαπλάσιο του 2”);
        writeln
end.
```

Εναλλακτικά, μπορεί να χρησιμοποιηθεί η έτοιμη συνάρτηση odd, η οποία ισούται με 0, αν ο αριθμός είναι άρτιος και με 1 αν ο αριθμός είναι περιττός. Προκύπτει το ισοδύναμο πρόγραμμα:

```
program Division2c;
begin
        write(“Δώστε έναν αριθμό : “);   readln(num);
        if  odd(num) = 0  then write(“Ο αριθμός είναι πολλαπλάσιο του 2”)
                        else write(“ Ο αριθμός δεν είναι πολλαπλάσιο του 2”);
        writeln
end.
```

1.15.3 Το παρακάτω πρόγραμμα δέχεται έναν αριθμό και αποφαίνεται αν αυτός είναι πολλαπλάσιος του 3. Υλοποιείται με βάση το κριτήριο διαιρετότητας, σύμφωνα με το οποίο ένας αριθμός είναι πολλαπλάσιος του 3 αν το άθροισμα των ψηφίων του διαιρείται με το 3.

Ακολουθείται η παρακάτω μέθοδος:

Βήμα 1^{ο}: Είσοδος ενός αριθμού a από το χρήστη.

Βήμα 2^{ο}: Υπολογισμός του αθροίσματος των ψηφίων του. Δηλαδή, υπολογισμός των ψηφίων του αριθμού a και άθροισή τους σε διαδοχικές εκτελέσεις του σώματος μιας εντολής **loop**, όπως στο πρόγραμμα 1.11.3.

Βήμα 3^{ο}: Έλεγχος αν το άθροισμα των ψηφίων του a αποτελεί μονοψήφιο πολλαπλάσιο του 3 και αντίστοιχη έξοδος.

Βήμα 4^{ο}: Τέλος του προγράμματος.

```
program Division3;
begin
        write(“Δώστε έναν αριθμό : “);   readln(num);
        sum := 0;
        loop begin
                digit := num mod 10;
                sum := sum + digit;
                num := num div 10;
                if num=0 then exit
        end;
        k:=0;
        if sum = 3 then k:=1
                    else if sum = 6 then k:=1
                                        else if sum = 9 then k:=1;
        if k=1 then write(“Ο αριθμός είναι πολλαπλάσιο του 3”)
               else write(“ Ο αριθμός δεν είναι πολλαπλάσιο του 3”);
        writeln
end.
```

Εναλλακτικά, οι τρεις εντολές:

```
        if sum = 3 then k:=1;
        if sum = 6 then k:=1;
        if sum = 9 then k:=1;
```

μπορούν να αντικατασταθούν με την εντολή:

```
        if  sum mod 3 =0  then k:=1;
```

η οποία θα έχει ακριβώς το ίδιο αποτέλεσμα.

1.15.4 Αν δε θεωρηθεί απαραίτητη η χρήση του κριτηρίου διαιρετότητας, τότε το πρόγραμμα μπορεί να γραφτεί ως εξής:

```
program Division2b;
begin
        write(“Δώστε έναν αριθμό : “);   readln(num);
        if  num mod 3 = 0  then write(“Ο αριθμός είναι πολλαπλάσιο του 3”)
                           else write(“ Ο αριθμός δεν είναι πολλαπλάσιο του 3”);
        writeln
end.
```

Ασκήσεις.

1. Να γραφτεί πρόγραμμα, το οποίο να δέχεται έναν αριθμό και να αποφαίνεται αν αυτός είναι πολλαπλάσιος του 5, με βάση το αντίστοιχο κριτήριο διαιρετότητας.

2. Να γραφτεί πρόγραμμα, το οποίο να δέχεται έναν αριθμό και να αποφαίνεται αν αυτός είναι πολλαπλάσιος του 9, με βάση το αντίστοιχο κριτήριο διαιρετότητας.

Κεφάλαιο 1.16 – Ανάλυση Αριθμού σε Γινόμενο Πρώτων Παραγόντων

1.16.1 Να γραφεί πρόγραμμα το οποίο θα διαβάζει έναν αριθμό και θα ελέγχει αν αυτός είναι πρώτος ή όχι.

Ένας αριθμός είναι πρώτος, αν δεν έχει γνήσιους διαιρέτες. Χρησιμοποιούμε μια παραλλαγή του προγράμματος 1.14.1

```
program PrimeChecker;
begin
        write(“Δώστε έναν αριθμό : “);   readln(num);
        loop begin
                if  divider * divider > num  then exit;
                if  num mod divider = 0  then begin   flag := 1;   exit   end;
                divider := divider +1
        end;
        if flag = 0 then write(“πρώτος”) else write(“σύνθετος”);   writeln
end.
```

1.16.2 Το παρακάτω πρόγραμμα δέχεται έναν αριθμό και τον αναλύει σε πρώτους παράγοντες, οι οποίοι έπειτα εμφανίζονται στην οθόνη.

```
program Primes;
begin
        write("Δώστε έναν αριθμό : ");   readln(num);
        flag := 0;
        loop begin                                          (*Ελεγχος διαιρετότητας με το 2*)
                if  num mod 2 = 0  then
                begin   flag := 1;   write("2");   writeln;   num := num div 2   end
                        else exit
        end;
        divider := 3;
        loop begin                          (*Ελεγχος διαιρετότητας με περιττούς αριθμούς > 2*)
                if   divider * divider > num  then exit;
                if  num mod divider = 0  then
                begin
                        flag := 1;   write(divider);   writeln;
                        num := num div divider
                end
                        else divider := divider +2
        end;
        if flag = 0 then  begin   write("Ο αριθμός ",num," είναι πρώτος.");   writeln   end
end.
```

Όπως αναφέρθηκε στο κεφάλαιο 1.14, οι διαιρέτες ενός αριθμού num είναι κάποιοι από τους τους αριθμούς από 2 έως και τον μικρότερο x, για τον οποίο ισχύει $x^2 > n$.

Γι' αυτό, η αναζήτηση πρώτων παραγόντων στο παραπάνω πρόγραμμα ξεκινά από το 2 (πρώτο loop), και σταματά όταν divider * divider > num (δεύτερο loop).

Ακόμη, από αυτούς τους αριθμούς είναι γνωστό ότι οι άρτιοι αριθμοί μεγαλύτεροι του 2 δεν είναι πρώτοι. Γι'αυτό το λόγο, το divider παίρνει τις τιμής 2, 3 και έπειτα αυξάνεται κατά 2, με αποτέλεσμα να έχει στη συνέχεια πάντα περιττή τιμή, με εξαίρεση το 2.

Βέβαια, όλοι οι περιττοί αριθμοί δεν είναι πρώτοι, παράδειγμα 15 = 5 * 3. Διότι δεν ισχύει ποτέ num **mod** divider = 0, για κάποιον divider που δεν είναι πρώτος, διότι αν δεν είναι πρώτος και διαιρεί τον num, τότε ο num διαιρείται και από τον πιό μικρό από τους παράγοντες του divider. Άρα θα έχει ήδη διαιρεθεί από αυτούς και έτσι δε θα διαιρείται πλέον με τον divider.

Για παράδειγμα: num =45

- divider = 2 δεν είναι παράγοντας
- divider = 3 είναι παράγοντας, num = 45 **div** 3 = 15
- divider = 3 είναι παράγοντας, num = 15 **div** 3 = 5

- divider = 3 δεν είναι παράγοντας
- divider = 5 είναι παράγοντας, num = 5 **div** 5 = 1

Η αναζήτηση πρώτων παραγόντων τελειώνει όταν έχουν ελεγχθεί όλοι οι υποψήφιοι αριθμοί divider. Αν δεν έχει βρεθεί κανένας παράγοντας τότε ο αριθμός, που δόθηκε από τον χρήστη, είναι πρώτος.

Η μέθοδος αυτή δεν είναι ιδιαίτερα αποδοτική. Όμως, δεν έχει βτρεθεί μια ουσιαστικά καλύτερη και εικάζεται ότι δεν υπάρχει τέτοια.

Κεφάλαιο 1.17 – Η Ευκλείδεια Διαίρεση

1.17.1 Το πρόγραμμα που ακολουθεί δέχεται τέσσερις αριθμούς, το διαιρετέο, το διαιρέτη, το πηλίκο και το υπόλοιπο και αποφαίνεται αν πρόκειται για τέλεια διαίρεση.

```
program EuclidChecker;
begin
        write(“Δώστε το διαιρετέο : “);   readln(num);
        write(“Δώστε το διαιρέτη : “);   readln(divider);
        write(“Δώστε το πηλίκο : “);   readln(quotient);
        write(“Δώστε το υπόλοιπο : “);   readln(remainder);
        r := divider * quotient + remainder;                    (*Υπολογισμός διαιρετέου*)
                                                (*Ελεγχος προϋποθέσεων ευκλείδειας διαίρεσης*)
        if num = r then if  remainder < divider  then write(“Ευκλείδεια διαίρεση”)
                                                 else write(“Όχι ευκλείδεια διαίρεση”);
        writeln
end.
```

1.17.2 Να γραφεί πρόγραμμα που να δέχεται δύο αριθμούς και να βρίσκει το πηλίκο και το υπόλοιπο της διαίρεσής στους χωρίς την χρήση των **div**, **mod**.

```
program EuclidWithoutDivMod
begin
        write(“Δώσε τον πρώτο αριθμό:”);   readln(a);
        write(“Δώσε τον δεύτερο αριθμό”);   readln(b);
                                                (*Υπολογισμός πηλίκου και υπόλοιπου*)
        quotient := a / b;   quotient := trunc(quotient);   remainder := a - (b * quotient);
        write(“Πηλίκο : “,quotient,”Υπόλοιπο : “,remainder);   writeln
end.
```

1.17.3 Το ακόλουθο πρόγραμμα δέχεται το διαιρετέο και το διαιρέτη και εκτελεί ευκλείδεια διαίρεση. Υπενθυμίζεται ότι, για τον υπολογισμό του πηλίκου χρησιμοποιείται η πράξη **div** και για τον υπολογισμό του υπόλοιπου η πράξη **mod**.

```
program Eukleidis;
begin
        write("Δώστε το διαιρετέο : ");   readln(num);
        write("Δώστε το διαιρέτη : ");   readln(divider);
                                                (*Υπολογισμός πηλίκου και υπόλοιπου*)
        quotient := num div divider;   remainder := num mod divider;
        write("Ευκλείδεια διαίρεση : ",num," = ", divider," * ", quotient," + ",remainder);
        writeln
end.
```

Ασκήσεις.

1. Να γραφτεί πρόγραμμα το οποίο να δέχεται το διαιρέτη divider και το πηλίκο quotient μιας ευκλείδειας διαίρεσης και να υπολογίζει όλους τους πιθανούς διαιρετέους.

2. Ένα πρόγραμμα δέχεται τους αριθμούς a, b και y. Υπολογίζει το μικρότερο ακέραιο αριθμό που όταν διαιρείται με καθέναν από τους αριθμούς:

   a, a+1, a+2, ..., b-2, b-1, b

   αφήνει πηλίκο y. Να κατασκευαστεί αυτό το πρόγραμμα με χρήση επανάληψης, εφόσον επιθυμεί ο χρήστης. Η επανάληψη τελειώνει με την είσοδο : a = b = y = 0.

Κεφάλαιο 1.21 – Τυποποιημένη Μορφή Μεγάλων αριθμών

1.21.1 Το πρόγραμμα που ακολουθεί δέχεται την τυποποιημένη μορφή ενός αριθμού και τον υπολογίζει. Ο χρήστης εισάγει τους συντελεστές και τους αντίστοιχους εκθέτες. Η εισαγωγή τελειώνει όταν δοθεί συντελεστής 0. Υποθέτουμε επίσης οτι οι εκθέτες είναι πάντοτε μεγαλύτεροι ή ίσοι με το 0. Για παράδειγμα: $6*10^9 + 2*10^6 + 7*10^2 + 1*10^0 = 6.002.000.701$

```
program StandardForm1;
begin
        sum := 0;
        loop begin
                loop begin
                        write("Δώστε το συντελεστή (0 έως 9) : ");
                        readln(digit);
                        if  digit >= 0  then if  digit < 10  then exit
                end;
                if  digit = 0  then exit;
                write("Δώστε τον εκθέτη : ");   readln(exp);
                sum := sum + digit * (10 ** exp);                    (*Υπολογισμός αριθμού*)
        end;
        write("Ο αριθμός είναι : ",sum);   writeln
end.
```

1.21.2 Χρήσιμο είναι και το αντίστροφο του παραπάνω προγράμματος. Το ακόλουθο πρόγραμμα δέχεται έναν αριθμό και τον εμφανίζει στην οθόνη σε τυποποιημένη μορφή.

```
program StandardForm2;
begin
        write("Δώστε έναν αριθμό : ");   readln(num);
        write(num," = ");
        rank := 0;
        loop begin                                          (*Ανάλυση αριθμού στα ψηφία του*)
                digit := num mod 10;
                write(digit," * 10 ^ ",rank);
                num := num div 10;
                if num=0 then exit else write(" + ");
                rank := rank + 1
        end;
        writeln
end.
```

Γενικές Ασκήσεις - Ασκήσεις Επανάληψης 1ου Κεφαλαίου.

1. Να γραφτεί πρόγραμμα το οποίο να υπολογίζει τον τετραψήφιο φυσικό αριθμό, που τα ψηφία του είναι όλα ίσα και έχουν άθροισμα 36.

2. Να γραφτεί πρόγραμμα το οποίο να υπολογίζει τον τετραψήφιο φυσικό αριθμό, που το γινόμενο των ψηφίων του ισούται με το άθροισμα των ψηφίων του.

3. Να γραφτεί πρόγραμμα το οποίο να υπολογίζει και να εμφανίζει στην οθόνη όλους τους τριψήφιους αριθμούς οι οποίοι να έχουν ψηφία ίσα με 0, 3, 6 ή 9.

4. Να γραφτεί πρόγραμμα το οποίο να υπολογίζει και να εμφανίζει στην οθόνη όλους τους τριψήφιους αριθμούς, που είναι ίσοι με το άθροισμα των ψηφίων τους υψωμένα στην τρίτη δύναμη. Παράδειγμα: $153 = 1^3 + 5^3 + 3^3$

5. Να γραφτεί πρόγραμμα το οποίο να υπολογίζει και να εμφανίζει στην οθόνη όλους τους τετραψήφιους αριθμούς, που είναι ίσοι με το άθροισμα των ψηφίων τους υψωμένα στην τέταρτη δύναμη. Παράδειγμα: $1634 = 1^4 + 6^4 + 3^4 + 4^4$

6. Υποθέστε ότι ένας δρόμος έχει x σπίτια. Να γραφτεί πρόγραμμα το οποίο να δέχεται ως είσοδο το x και να υπολογίζει πόσα ψηφία χρειάζεται να γραφούν για να αριθμηθούν όλα τα σπίτια του δρόμου.

Κεφάλαιο 2

Μετρήσεις Μεγεθών

ΔΙΑΔΙΚΑΣΙΕΣ

Από τα προγράμματα που παρουσιάστηκαν μέχρι τώρα, γίνεται κατανοητό ότι συχνά ένα πρόγραμμα είναι τόσο μακροσκελές που η κατανόησή είναι αρκετά δύσκολη. Το πρόβλημα εντείνεται όλο και περισσότερο, όσο αυξάνονται οι απαιτήσεις του προγραμματιστή από το πρόγραμμα και το πλήθος των εντολών του προγράμματος.

Παρόλα αυτά, μπορεί να αμβλυνθεί με τη χρήση των διαδικασιών (**procedures**).

Οι **διαδικασίες (procedures)** είναι υποπρογράμματα. Χρησιμοποιούνται ως δομικά στοιχεία για την κατασκευή του κυρίως προγράμματος.

Επεξηγώντας τον παραπάνω ορισμό, κάθε διαδικασία έχει:

- ένα όνομα (ισχύουν οι γνωστοί κανόνες)
- και το σώμα της αποτελείται από μία **σύνθετη εντολή.** Υπενθυμίζεται ότι **σύνθετη εντολή** ονομάζεται μια ακολουθία εντολών που περικλείεται από τις δεσμευμένες λέξεις **begin** και **end**.

Ουσιαστικά οι διαδικασίες είναι μικρά, αυτόνομα κομμάτια κώδικα τα οποία εκτελούν συγκεκριμένες λειτουργίες και η χρήση τους διευκολύνει πολύ τη δημιουργία ενός προγράμματος.

Η εμφάνιση του ονόματος μιας διαδικασίας στο κυρίως πρόγραμμα ονομάζεται **κλήση διαδικασίας**, επειδή με αυτό τον τρόπο το κυρίως πρόγραμμα καλεί προς εκτέλεση τις εντολές που περιλαμβάνει η διαδικασία.

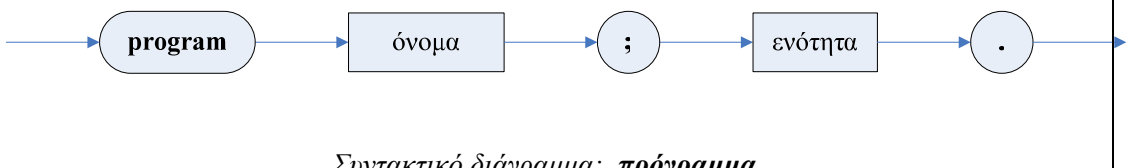
Η κλήση μιας διαδικασίας διαφέρει από την περιγραφή της διαδικασίας.

Περιγραφή διαδικασίας ονομάζεται το τμήμα του προγράμματος που αρχίζει με τη δεσμευμένη λέξη **procedure**, συνεχίζει με το όνομα της διαδικασίας και το σύμβολο semicolon (;) και τελειώνει με μία σύνθετη εντολή. Οι διαδικασίες γράφονται μετά από το κυρίως πρόγραμμα.

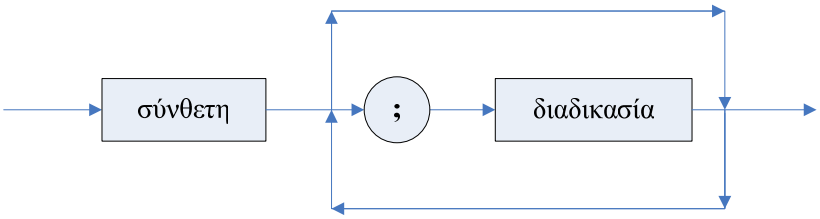
Η περιγραφή μιας διαδικασίας έχει την ακόλουθη δομή:

```
procedure όνομα διαδικασίας;                    (*Επικεφαλίδα*)
begin
        εντολές διαδικασίας                     (*Κυρίως σώμα διαδικασίας *)
end.
```

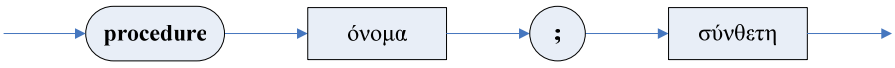
Τα συντακτικά διαγράμματα της γλώσσας αναδιαμορφώνονται, έτσι ώστε να συμπεριλαμβάνουν και τις διαδικασίες:



Συντακτικό διάγραμμα: ***πρόγραμμα***



Συντακτικό διάγραμμα: ***ενότητα***



Συντακτικό διάγραμμα: ***διαδικασία***

Συντακτικό διάγραμμα: ***κλήση διαδικασίας***

Ένα απλό παράδειγμα:

```
program HelloProcedures;
        begin   for 5 times do Hello   end;
```

```
procedure Hello;
        begin   write(“Hello World !!!”);   writeln   end.
```

Ακολουθεί το πρόγραμμα 1.6.1 με χρήση διαδικασιών:

```
program Equation1b;
begin
        TitleMessage;
        InputData;
        DisplayResults
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα λύνει την εξίσωση  x + a = b.”);
        writeln
end;
```

```
procedure InputData;
begin
        write(“Δώστε την τιμή του a :”);   readln(a);
        write(“Δώστε την τιμή του b :”);   readln(b)
end;
```

```
procedure DisplayResults;
begin
        x := b – a;
        write(“Αποτέλεσμα :  x = “,x);   writeln
end.
```

Αναλυτικά οι εργασίες του παραπάνω προγράμματος είναι:

- Αρχικά, το κυρίως πρόγραμμα Equation1b καλεί την διαδικασία TitleMessage.
- Η διαδικασία TitleMessage εμφανίζει στην οθόνη το μήνυμα: «Αυτό το πρόγραμμα λύνει την εξίσωση x + a = b»
- Στη συνέχεια το κυρίως πρόγραμμα καλεί την διαδικασία InputData.
- Η διαδικασία InputData ζητά από τον χρήστη τις τιμές για τις μεταβλητές a,b.
- Η τελευταία διαδικασία που καλεί το κυρίως πρόγραμμα είναι η DisplayResults.
- Η διαδικασία DisplayResults αποθηκεύει στη μεταβλητή x το αποτέλεσμα της αφαίρεσης b-a και το εμφανίζει στην οθόνη.

Ο προγραμματισμός με τη χρήση διαδικασιών επιτρέπει στον προγραμματιστή να δημιουργεί **μία διαδικασία για κάθε ανεξάρτητη λογική ενότητα ή λειτουργία** του προγράμματος. Στο παραπάνω παράδειγμα, οι λογικές ενότητες είναι η εισαγωγή, στην οποία περιγράφεται η λειτουργία του προγράμματος, η είσοδος των δεδομένων και η εμφάνιση του αποτελέσματος στην οθόνη.

Για καθεμιά από αυτές δημιουργήθηκε η αντίστοιχη διαδικασία. Η ανάπτυξη μιας διαδικασίας για κάθε συγκεκριμένη λειτουργία του προγράμματος αποτελεί ένα χρήσιμο τρόπο σκέψης για την ανάπτυξη προγραμμάτων.

Από εδώ και πέρα τα προγράμματα θα είναι δομημένα και θα έχουν την εξής γενική μορφή:

```
program ΟνομαΠρογράμματος;
begin
    TitleMessage;          (*Διαδικασία που εμφανίζει τι κάνει το πρόγραμμα*)
    loop begin
        InputData;         (*Διαδικασία που δέχεται δεδομένα από το χρήστη*)
        ComputeResults;       (*Διαδικασία που υπολογίζει τα αποτελέσματα*)
        DisplayResults;        (*Διαδικασία που εμφανίζει τα αποτελέσματα*)
                  (*Διαδικασία που ερωτά το χρήστη αν επιθυμεί να συνεχίσει *)
        AskUserWToContinue
    end;
end;
```

Τονίζεται, ότι δεν είναι απαραίτητο ένα πρόγραμμα να περιλαμβάνει όλες τις διαδικασίες του παραπάνω υποδείγματος. Αναλόγως με τις απαιτήσεις του προγράμματος, ο προγραμματιστής επιλέγει ποιες διαδικασίες χρειάζονται.

Ένα ακόμη πλεονέκτημα του προγραμματισμού με χρήση διαδικασιών είναι η δυνατότητα του προγραμματιστή να καλέσει την ίδια διαδικασία σε δύο ή περισσότερα σημεία του προγράμματος. Όταν ένα τμήμα ενός προγράμματος εμφανίζεται σε δύο ή περισσότερα διαφορετικά σημεία, τότε ο προγραμματιστής δημιουργεί μία διαδικασία που περιέχει αυτό το τμήμα προγράμματος και την καλεί όπου χρειάζεται στο κυρίως πρόγραμμα. Αυτό οδηγεί σε μικρότερα και κομψότερα προγράμματα, που γίνονται ευκολότερα κατανοητά, ενώ παράλληλα διευκολύνει την πληκτρολόγηση του προγράμματος στον υπολογιστή.

Οι διαδικασίες από μόνες τους (αν δεν τις καλεί το κυρίως πρόγραμμα) δεν εκτελούνται.

Κεφάλαιο 2.4 – Μέτρηση Μήκους Τμήματος

2.4.1 Να γραφτεί πρόγραμμα το οποίο να δέχεται έναν αριθμό, που αντιπροσωπεύει τη μονάδα μέτρησης μηκών και στη συνέχεια να "μετράει" με βάση αυτή τη μονάδα μέτρησης όλα τα επόμενα μήκη που εισάγονται από το χρήστη. Το πρόγραμμα σταματάει όταν δοθεί μήκος ίσο με το μηδέν.

```
program LengthUnit;
begin
        write("Μήκος μονάδας μέτρησης : "); readln(unitlength);
        loop begin
                write("Δώστε μήκος : "); readln(length);
                if length = 0 then exit;
                count := length / unitlength;
                write("Μέτρηση : ",count); writeln
        end
end.
```

Ασκήσεις.

1. Να γραφτεί το παραπάνω πρόγραμμα με χρήση διαδικασιών.

Κεφάλαιο 2.5 – Οι Κυριότερες Μονάδες Μήκους

2.5.1 Να γραφτούν, με χρήση επανάληψης, δύο προγράμματα τα οποία να δέχονται το μήκος ενός ευθύγραμμου τμήματος σε μέτρα και να το μετατρέπουν σε εκατοστόμετρα. Τα προγράμματα σταματούν όταν δοθεί μήκος τμήματος ίσο με το μηδέν. Για εξοικείωση με τις διαδικασίες, το πρώτο εξ αυτών δεν χρησιμοποιεί διαδικασίες ενώ το δεύτερο αντιθέτως χρησιμοποιεί.

Πρώτο πρόγραμμα:

```
program LengthUnitA;
begin
        write(“Αυτό το πρόγραμμα μετατρέπει μήκη”);   writeln;
        write(“μετρημένα σε μέτρα (m), σε μήκη”);   writeln;
        write(“μετρημένα σε εκατοστόμετρα (cm).”);   writeln;
        loop begin
                write(“Δώστε μήκος σε μέτρα (m) : “);   readln(mlength);
                if  mlength = 0  then exit;
                cmlength := mlength * 100;
                write(“Μήκος σε εκατοστόμετρα (cm) : “, cmlength);   writeln
        end
end.
```

Δεύτερο πρόγραμμα:

```
program LengthUnitB;
begin
        TitleMessage;
        loop begin
            InputData;   AskUserWToContinue;
            DisplayResults
    end
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα μετατρέπει μήκη”);   writeln;
        write(“μετρημένα σε μέτρα (m), σε μήκη”);   writeln;
        write(“μετρημένα σε εκατοστόμετρα (cm).”);   writeln
end;
```

```
procedure InputData;
begin   write(“Δώστε μήκος σε μέτρα (m) : “);   readln(mlength)   end;
```

```
procedure AskUserWToContinue;
begin   if  mlength = 0  then exit   end;
```

```
procedure DisplayResults;
begin
        cmlength := mlength * 100;
        write(“Μήκος σε εκατοστόμετρα (cm) : “, cmlength);   writeln
end.
```

2.5.2 Να γραφτούν με χρήση επανάληψης δύο προγράμματα τα οποία να δέχονται το μήκος ενός ευθύγραμμου τμήματος, τη μονάδα μέτρησης του και τη μονάδα στην οποία ζητείται να γίνει η μετατροπή. Τα προγράμματα κάνουν τη μετατροπή και εμφανίζουν το αποτέλεσμα στην οθόνη. Η επανάληψη τελειώνει και τα προγράμματα σταματούν όταν δοθεί μήκος τμήματος ίσο με το μηδέν. Για εξοικείωση με τις διαδικασίες, το πρώτο εξ αυτών δεν χρησιμοποιεί διαδικασίες ενώ το δεύτερο αντιθέτως χρησιμοποιεί. Πρώτο πρόγραμμα:

```
program LengthTransformerA;
begin
        write(“Αυτό το πρόγραμμα κάνει μετατροπές μονάδων μήκους”);   writeln;
        loop begin
                write(“Δώστε μήκος : “);   readln(length);
                if  length = 0  then exit;
                write(“7 για χιλιομετρα (km), 6 για εκατόμετρα (hm), “);  writeln;
                write(“5 για δεκάμετρα (km), 4 για μέτρα (m), “);   writeln;
                write(“3 για δεκατόμετρα (dm), 2 για εκατοστό-”);   writeln;
                write(“μετρα (cm), 1 για χιλιοστόμετρα (dm) : “);   writeln;
                loop begin
                        write(“Δώστε μονάδα μέτρησης (1 έως 7) : “);   readln(unitA);
                        if unitA>0 then if unitA<8 then exit
                end;
                loop begin
                        write(“Δώστε μονάδα μέτρησης αποτελέσματος (1 έως 7): “);
                        readln(unitB);
                        if unitB>0 then if unitB<8 then exit
                end;
                if  unitA > unitB  then begin                (*Μετατροπή σε υποδιαίρεση*)
                        steps := unitA – unitB;
                        for steps times length := length * 10
                end;
                else begin                                   (*Μετατροπή σε πολλαπλάσιο*)
                        steps := unitB – unitA;
                        for steps times length := length / 10
                end;
                write(“Αποτέλεσμα : “,length);   writeln
        end
end.
```

Δεύτερο πρόγραμμα:

```
program LengthTransformerB;
begin
        TitleMessage;
        loop begin
                GetLength;   AskUserWToContinue;
                GetUnitA;   GetUnitB;
                Transform;   DisplayResults
        end
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα κάνει μετατροπές μονάδων μήκους”);   writeln
end;
```

```
procedure GetLength;
        begin   write(“Δώστε μήκος : “);   readln(length)   end;
```

```
procedure AskUserWToContinue;
        begin   if  length = 0  then exit   end;
```

```
procedure GetUnitA;
begin
        write(“7 για χιλιομετρα (km), 6 για εκατόμετρα (hm), “);   writeln;
        write(“5 για δεκάμετρα (km), 4 για μέτρα (m), “);   writeln;
        write(“3 για δεκατόμετρα (dm), 2 για εκατοστό-”);   writeln;
        write(“μετρα (cm), 1 για χιλιοστόμετρα (dm) : “);   writeln;
        loop begin
                write(“Δώστε μονάδα μέτρησης (1 έως 7) : “);   readln(unitA);
                if unitA>0 then if unitA<8 then exit
        end
end;
```

```
procedure GetUnitB;
begin
        loop begin
                write("Δώστε μονάδα μέτρησης αποτελέσματος (1 έως 7): ");
                readln(unitB);   if unitB>0 then if unitB<8 then exit
        end
end;
```

```
procedure Transform;
begin
        if  unitA > unitB  then begin                    (*Μετατροπή σε υποδιαίρεση*)
                steps := unitA – unitB;
                for steps times length := length * 10
        end
        else begin                                       (*Μετατροπή σε πολλαπλάσιο*)
                steps := unitB – unitA;
                for steps times length := length / 10
        end
end.
```

```
procedure DisplayResults;
        begin   write("Αποτέλεσμα : ",length);   writeln   end.
```

Ασκήσεις.

1. Να γραφτεί το πρόγραμμα με χρήση διαδικασιών το οποίο δέχεται ένα μήκος μετρημένο σε υάρδες, πόδια ή ίντσες και εκτελεί όλες τις δυνατές μετατροπές μεταξύ των παραπάνω μονάδων.

2. Να γραφτεί το πρόγραμμα με χρήση διαδικασιών το οποίο δέχεται ένα μήκος μετρημένο σε μίλια και να το μετατρέπει σε ναυτικά μίλια και το αντίστροφο.

Κεφάλαιο 2.7 – Το Μέτρο Ως Μονάδα Μήκους

2.7.1 Να γραφτούν, με χρήση επανάληψης, δύο προγράμματα τα οποία να δέχονται το μήκος ενός ευθύγραμμου τμήματος σε μορφή συμμιγούς αριθμού και να το μετατρέπουν σε δεκαδική παράσταση. Τα προγράμματα σταματούν όταν δοθεί μήκος τμήματος ίσο με το μηδέν. Για εξοικείωση με τις διαδικασίες, το πρώτο εξ αυτών δεν χρησιμοποιεί διαδικασίες ενώ το δεύτερο αντιθέτως χρησιμοποιεί.

Πρώτο πρόγραμμα:

```
program LengthDecA;
begin
        write(“Αυτό το πρόγραμμα μετατρέπει μήκη σε μορφή”);   writeln;
        write(“συμμιγούς αριθμού, σε μήκη δεκαδικής παράστασης”); writeln;
        loop begin
                write(“Δώστε πλήθος χιλιόμετρων (km) : “);   readln(km);
                write(“Δώστε πλήθος εκατόμετρων (hm) : “);   readln(hm);
                write(“Δώστε πλήθος δεκάμετρων (dam) : “);   readln(dam);
                write(“Δώστε πλήθος μέτρων (m) : “);   readln(m);
                write(“Δώστε πλήθος δεκατόμετρων (dm) : “);   readln(dm);
                write(“Δώστε πλήθος εκατοστόμετρων (cm) : “);   readln(cm);
                write(“Δώστε πλήθος χιλιοστόμετρων (mm) : “);   readln(mm);
                sum:=1000*km+100*hm+10*dam+m+0.1*dm+0.01*cm+0.001*mm;
                if  sum = 0  then exit;
                write(“Συνολικό μήκος : “,sum);   writeln
        end
end.
```

Δεύτερο πρόγραμμα:

```
program LengthDecB;
begin
        TitleMessage;
        loop begin
                InputData;   Compute;
                AskUserWToContinue;
                DisplayResults
        end
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα μετατρέπει μήκη σε μορφή”);   writeln;
        write(“συμμιγούς αριθμού, σε μήκη δεκαδικής παράστασης”); writeln
end;
```

```
procedure InputData;
begin
        write(“Δώστε πλήθος χιλιόμετρων (km) : “);   readln(km);
        write(“Δώστε πλήθος εκατόμετρων (hm) : “);   readln(hm);
        write(“Δώστε πλήθος δεκάμετρων (dam) : “);   readln(dam);
        write(“Δώστε πλήθος μέτρων (m) : “);   readln(m);
        write(“Δώστε πλήθος δεκατόμετρων (dm) : “);   readln(dm);
        write(“Δώστε πλήθος εκατοστόμετρων (cm) : “);   readln(cm);
        write(“Δώστε πλήθος χιλιοστόμετρων (mm) : “);   readln(mm)
end;
```

```
procedure Compute;
begin   sum:=1000*km+100*hm+10*dam+m+0.1*dm+0.01*cm+0.001*mm   end;
```

```
procedure AskUserWToContinue;
begin   if  sum = 0  then exit   end;
```

```
procedure DisplayResults;
        begin   write(“Συνολικό μήκος : “,sum);   writeln   end.
```

Κεφάλαιο 2.8 – Εμβαδό Επίπεδων Επιφανειών

2.8.1 Να γραφτεί πρόγραμμα το οποίο να δέχεται έναν αριθμό, που αντιπροσωπεύει τη μονάδα μέτρησης εμβαδών και στη συνέχεια να "μετράει" με βάση αυτή τη μονάδα μέτρησης όλα τα επόμενα εμβαδά που εισάγονται από το χρήστη. Το πρόγραμμα σταματάει όταν δοθεί εμβαδό ίσο με το μηδέν.

```
program AreaUnit;
begin
        write("Εμβαδό μονάδας μέτρησης : "); readln(unitarea);
        loop begin
                write("Δώστε εμβαδό : "); readln(area);
                if area = 0 then exit;
                count := length / unitlength;
                write("Μέτρηση : ",count); writeln
        end
end.
```

Άσκηση.

1. Να γραφτεί το πρόγραμμα 2.8.1 με χρήση διαδικασιών.

Κεφάλαιο 2.10 – Σχέσεις Τετραγωνικού Μέτρου Με Υποδιαιρέσεις και Πολλαπλάσια

2.10.1 Να γραφτούν, με χρήση επανάληψης, δύο προγράμματα τα οποία να δέχονται το εμβαδό μιας επιφάνειας σε τετραγωνικά μέτρα και να το μετατρέπουν σε τετραγωνικά εκατοστόμετρα. Τα προγράμματα σταματούν όταν δοθεί εμβαδόν επιφάνειας ίσο με το μηδέν. Για εξοικείωση με τις διαδικασίες, το πρώτο εξ αυτών δεν χρησιμοποιεί διαδικασίες ενώ το δεύτερο αντιθέτως χρησιμοποιεί.

Πρώτο πρόγραμμα:

```
program AreaUnitA;
begin
        write(“Αυτό το πρόγραμμα μετατρέπει εμβαδά”);   writeln;
        write(“μετρημένα σε τετραγωνικά μέτρα (m^2), σε εμβαδά”);  writeln;
        write(“μετρημένα σε τετραγωνικά εκατοστόμετρα (cm^2).”);   writeln;
        loop begin
                write(“Δώστε εμβαδό σε τετραγωνικά μέτρα (m^2) : “);   readln(marea);
                if  marea = 0  then exit;
                cmarea := marea * 10000;
                write(“Εμβαδό σε τετραγωνικά εκατοστόμετρα (cm^2) : “,cmarea);
                writeln
        end
end.
```

Δεύτερο πρόγραμμα:

```
program AreaUnitB;
        begin
                TitleMessage;
                loop begin
                InputData;  AskUserWToContinue;   DisplayResults
                end
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα μετατρέπει εμβαδά”);   writeln;
        write(“μετρημένα σε τετραγωνικά μέτρα (m^2), σε εμβαδά”);  writeln;
        write(“μετρημένα σε τετραγωνικά εκατοστόμετρα (cm^2).”);   writeln
end;
```

```
procedure InputData;
begin
        write(“Δώστε εμβαδό σε τετραγωνικά μέτρα (m^2) : “);   readln(marea);
        if  marea = 0  then exit
end;
```

```
procedure AskUserToContinue;
        begin   if  marea = 0  then exit   end;
```

```
procedure DisplayResults;
begin
        cmarea := marea * 10000;
        write(“Εμβαδόν σε τετραγωνικά εκατοστόμετρα (cm^2): “,cmarea);   writeln
end.
```

2.10.2 Να γραφτούν με χρήση επανάληψης δύο προγράμματα τα οποία να δέχονται το εμβαδό μιας επιφάνειας, τη μονάδα μέτρησης του και τη μονάδα στην οποία ζητείται να γίνει η μετατροπή. Τα προγράμματα κάνουν τη μετατροπή και εμφανίζουν το αποτέλεσμα στην οθόνη. Η επανάληψη τελειώνει και τα προγράμματα σταματούν όταν δοθεί εμβαδό επιφάνειας ίσο με το μηδέν. Για εξοικείωση με τις διαδικασίες, το πρώτο εξ αυτών δεν χρησιμοποιεί διαδικασίες ενώ το δεύτερο αντιθέτως χρησιμοποιεί.

Πρώτο πρόγραμμα:

```
program LengthTransformerA;
begin
        write(“Αυτό το πρόγραμμα κάνει μετατροπές μονάδων εμβαδών”);  writeln;
        loop begin
                write(“Δώστε μήκος : “);   readln(area);
                if  area = 0  then exit;
                write(“7 για τ. χιλιόμετρα (km^2), 6 για τ. εκατόμετρα (hm^2), “);
                writeln;
                write(“5 για τ. δεκάμετρα (km^2), 4 για τ. μέτρα (m^2), “);   writeln;
                write(“3 για τ. δεκατόμετρα (dm^2), 2 για τ. εκατοστό-”);   writeln;
                write(“μετρα (cm^2), 1 για τ. χιλιοστόμετρα (dm^2) : “);   writeln;
                loop begin
                        write(“Δώστε μονάδα μέτρησης (1 έως 7) : “);   readln(unitA);
                        if unitA>0 then if unitA<8 then exit
                end;
                loop begin
                        write(“Δώστε μονάδα μέτρησης αποτελέσματος (1 έως 7): “);
                        readln(unitB);   if unitB>0 then if unitB<8 then exit
                end;
                if  unitA > unitB  then begin                (*Μετατροπή σε υποδιαίρεση*)
                        steps := unitA – unitB;
                        for steps times area := area * 100
                end;
                if  unitA < unitB  then begin                (*Μετατροπή σε πολλαπλάσιο*)
                        steps := unitB – unitA;
                        for steps times area := area / 100
                end;
                write(“Αποτέλεσμα : “, area);   writeln
        end
end.
```

Δεύτερο πρόγραμμα:

```
program AreaTransformerB;
begin
        TitleMessage;
        loop begin
                GetArea;   AskUserWToContinue;
                GetUnitA;   GetUnitB;
                Transform;   DisplayResults
        end
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα κάνει μετατροπές μονάδων εμβαδού”);   writeln
end;
```

```
procedure GetArea;
        begin   write(“Δώστε εμβαδό : “);   readln(area)   end;
```

```
procedure AskUserWToContinue;
        begin   if  area = 0  then exit   end;
```

```
procedure GetUnitA;
begin
        write(“7 για τ. χιλιομετρα (km^2), 6 για τ. εκατόμετρα (hm^2), “);   writeln;
        write(“5 για τ. δεκάμετρα (km^2), 4 για τ. μέτρα (m^2), “);   writeln;
        write(“3 για τ. δεκατόμετρα (dm^2), 2 για τ. εκατοστό-”);   writeln;
        write(“μετρα (cm^2), 1 για τ. χιλιοστόμετρα (dm^2) : “);   writeln;
        loop begin
                write(“Δώστε μονάδα μέτρησης (1 έως 7) : “);   readln(unitA);
                if unitA>0 then if unitA<8 then exit
        end
end;
```

```
procedure GetUnitB;
begin
        loop begin
        write(“Δώστε μονάδα μέτρησης αποτελέσματος (1 έως 7): “);   readln(unitB);
        if unitB>0 then if unitB<8 then exit
        end
end;
```

```
procedure Transform;
begin
        if  unitA > unitB  then begin                    (*Μετατροπή σε υποδιαίρεση*)
                steps := unitA – unitB;
                for steps times  area := area * 100
        end
        else begin                                       (*Μετατροπή σε πολλαπλάσιο*)
                steps := unitB – unitA;
                for steps times area := area / 100
        end
end;
```

```
procedure DisplayResults;
        begin   write(“Αποτέλεσμα : “, area);   writeln   end.
```

Ασκήσεις.

1. Να γραφτεί το πρόγραμμα με χρήση διαδικασιών το οποίο δέχεται ένα εμβαδό μετρημένο σε στρέμματα, τετραγωνικά χιλιόμετρα ή τετραγωνικά μέτρα και εκτελεί όλες τις δυνατές μετατροπές μεταξύ των παραπάνω μονάδων.

Κεφάλαιο 2.11 – Εμβαδό Ορθογωνίου και Τετραγώνου

2.11.1 Να γραφτεί με χρήση επανάληψης και διαδικασιών ένα πρόγραμμα το οποίο να δέχεται τα μήκη των πλευρών ενός ορθογώνιου παραλληλόγραμμου και να υπολογίζει την περίμετρο και το εμβαδό και του. Το πρόγραμμα σταματάει όταν δοθεί μηδενικές πλευρές ορθογώνιου παραλληλόγραμμου.

```
program AreaAB;
begin
        TitleMessage;
        loop begin
                GetSideA;   GetSideB;
                AskUserWToContinue;   DisplayResults;
        end
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα υπολογίζει το εμβαδό”);   writeln;
        write(“ορθογωνίου παραλληλογράμμου”);   writeln
end;
```

```
procedure GetSideA;
        begin   write(“Δώστε μήκος πλευράς a : “);   readln(side1)   end;
```

```
procedure GetSideB;
        begin   write(“Δώστε μήκος πλευράς b : “);   readln(side2)   end;
```

```
procedure AskUserWToContinue;
        begin   if  side1 = 0  then if  side2 = 0  then exit   end;
```

```
procedure DisplayResults;
begin
        per := 2 * a + 2 * b;   area := a * b;
        write(“Περίμετρος ορθογώνιου παραλληλόγραμμου : “, per);   writeln;
        write(“Εμβαδό ορθογώνιου παραλληλόγραμμου : “, area);   writeln
end.
```

2.11.2 Να γραφτεί με χρήση επανάληψης και διαδικασιών ένα πρόγραμμα το οποίο να δέχεται το μήκος πλευράς τετραγώνου και να υπολογίζει την περίμετρο και το εμβαδό του. Το πρόγραμμα σταματάει όταν δοθεί μηδενικό μήκος πλευράς τετραγώνου.

```
program AreaSquare;
begin
        TitleMessage;
        loop begin
                GetSide;   AskUserWToContinue;
                DisplayResults;
        end
end;
```

```
procedure TitleMessage;
begin   write(“Αυτό το πρόγραμμα υπολογίζει το εμβαδό τετραγώνου”);   writeln   end;
```

```
procedure GetSide;
        begin   write(“Δώστε μήκος πλευράς : “);   readln(side)   end;
```

```
procedure AskUserWToContinue;
        begin   if  side = 0  then exit   end;
```

```
procedure DisplayResults;
begin
        per := 4 * a;   area := a * a;
        write(“Περίμετρος τετραγώνου: “, area);   writeln;
        write(“Εμβαδό τετραγώνου: “, area);   writeln
end.
```

Ασκήσεις.

1. Να γραφτεί το πρόγραμμα με χρήση επανάληψης και διαδικασιών το οποίο δέχεται το εμβαδό ορθογωνίου παραλληλογράμμου και το μήκος της μιας πλευράς του και υπολογίζει την άλλη. Το πρόγραμμα σταματάει όταν δοθεί μηδενικό εμβαδό και μήκος πλευράς.

Κεφάλαιο 2.12 – Όγκος Στερεών

2.12.1 Να γραφτεί πρόγραμμα το οποίο να δέχεται έναν αριθμό, που αντιπροσωπεύει τη μονάδα μέτρησης όγκων και στη συνέχεια να "μετράει" με βάση αυτή τη μονάδα μέτρησης όλα τους επόμενους όγκους που εισάγονται από το χρήστη. Το πρόγραμμα σταματάει όταν δοθεί όγκος ίσο με το μηδέν.

```
program AreaUnit;
begin
        write("Όγκος μονάδας μέτρησης : "); readln(unitvolume);
        loop begin
                write("Δώστε όγκο : "); readln(volume);
                if volume = 0 then exit;
                count := volume / unitvolume;
                write("Μέτρηση : ",count); writeln
        end
end.
```

Ασκήσεις.

1. Να γραφτεί το πρόγραμμα 2.12.1 με χρήση διαδικασιών.

Κεφάλαιο 2.13 – Οι Κυριότερες Μονάδες Όγκου

2.13.1 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, ένα πρόγραμμα το οποίο να δέχεται τον όγκο ενός στερεού σε κυβικά μέτρα και να τον μετατρέπουν σε κυβικά εκατοστόμετρα. Το πρόγραμμα σταματάει όταν δοθεί όγκος στερεού ίσος με το μηδέν.

```
program VolumeUnit;
begin
        TitleMessage;
        loop begin
                Inputdata;  AskUserWToContinue;
                DisplayResults
        end
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα μετατρέπει όγκους”);   writeln;
        write(“μετρημένους σε κυβικά μέτρα (m^3), σε όγκους”);   writeln;
        write(“μετρημένους σε κυβικά εκατοστόμετρα (cm^3).”);   writeln
end;
```

```
procedure Inputdata;
begin
        write(“Δώστε όγκο σε κυβικά μέτρα (m^3) : “);   readln(mvolume)
end;
```

```
procedure AskUserWToContinue;
        begin   if  mvolume = 0  then exit   end;
```

```
procedure DisplayResults;
begin
        cmvolume := mvolume * 1000000;
        write(“Όγκος σε κυβικά εκατοστόμετρα (cm^3): “,cvolume);   writeln
end.
```

2.13.2 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, ένα πρόγραμμα το οποίο να δέχεται τον όγκο ενός στερεού, τη μονάδα μέτρησης του και τη μονάδα στην οποία ζητείται να γίνει η μετατροπή. Το πρόγραμμα κάνει τη μετατροπή και εμφανίζει το αποτέλεσμα στην οθόνη. Η επανάληψη τελειώνει και το πρόγραμμα σταματάει όταν δοθεί όγκος στερεού ίσος με το μηδέν.

```
program VolumeTransformer;
begin
        TitleMessage;
        loop begin
                GetVolume;   AskUserWToContinue;
                GetUnitA;   GetUnitB;
                Transform;   DisplayResults
        end
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα κάνει μετατροπές μονάδων όγκου”);   writeln
end;
```

```
procedure GetVolume;
        begin   write(“Δώστε όγκο : “);   readln(volume);   end
```

```
procedure AskUserWToContinue;
        begin   if  volume = 0  then exit   end;
```

```
procedure GetUnitA;
begin
        write(“7 για κ. χιλιομετρα (km^3), 6 για κ. εκατόμετρα (hm^3), “);   writeln;
        write(“5 για κ. δεκάμετρα (km^3), 4 για κ. μέτρα (m^3), “);   writeln;
        write(“3 για κ. δεκατόμετρα (dm^3), 2 για κ. εκατοστό-”);   writeln;
        write(“μετρα (cm^3), 1 για κ. χιλιοστόμετρα (dm^3) : “);   writeln;
        loop begin
                write(“Δώστε μονάδα μέτρησης (1 έως 7) : “);   readln(unitA);
                if unitA>0 then if unitA<8 then exit
        end
end;
```

```
procedure GetUnitB;
begin
        loop begin
                write(“Δώστε μονάδα μέτρησης αποτελέσματος (1 έως 7): “);
                readln(unitB);
                if unitB>0 then if unitB<8 then exit
        end
end;
```

```
procedure Transform;
begin
        if  unitA > unitB  then begin                    (*Μετατροπή σε υποδιαίρεση*)
                steps := unitA – unitB;
                for steps times  volume := volume * 1000
        end
        else begin                                       (*Μετατροπή σε πολλαπλάσιο*)
                steps := unitB – unitA;
                for steps times volume := volume / 1000
        end
end;
```

```
procedure DisplayResults;
        begin   write(“Αποτέλεσμα : “,volume);   writeln   end.
```

Ασκήσεις.

1. Να γραφτεί το πρόγραμμα με χρήση διαδικασιών το οποίο δέχεται έναν όγκο μετρημένο σε λίτρα (l), κυβικά μέτρα (m^3) ή κυβικά πόδια (ft^3) και εκτελεί όλες τις δυνατές μετατροπές μεταξύ των παραπάνω μονάδων.

Κεφάλαιο 2.14 – Όγκος Ορθογωνίου Παραλληλεπιπέδου

2.14.1 Να γραφτεί με χρήση επανάληψης και διαδικασιών ένα πρόγραμμα το οποίο να δέχεται τα μήκη των πλευρών ορθογώνιου παραλληλεπιπέδου και να υπολογίζει τα εξής:

- τον όγκο του
- το εμβαδό των βάσεων
- το εμβαδό της παράπλευρης επιφάνειας
- το συνολικό μήκος των ακμών του.

Το πρόγραμμα σταματάει όταν δοθεί μηδενικές και οι τρεις διαστάσεις του ορθογώνιου παραλληλεπιπέδου.

```
program VolumeABC;
begin
        TitleMessage;
        loop begin
                GetSideA;   GetSideB;   GetSideC;
                AskUserWToContinue;   DisplayResults;
        end
end;
```

```
procedure TitleMessage;
begin
        write("Αυτό το πρόγραμμα υπολογίζει τον όγκο, το");   writeln;
        write("εμβαδό των βάσεων και της παράπλευρης");   writeln;
        write("επιφάνειας και το συνολικό μήκος των ακμών");   writeln;
        write(" ορθογώνιου παραλληλεπιπέδου.");   writeln
end;
```

```
procedure GetSideA;
        begin   write("Δώστε μήκος ορθογωνίου : ");   readln(a)   end;
```

```
procedure GetSideB;
        begin   write("Δώστε πλάτος ορθογωνίου : ");   readln(b)   end;
```

```
procedure GetSideC;
        begin   write("Δώστε ύψος ορθογωνίου : ");   readln(c)   end;
```

```
procedure AskUserWToContinue;
        begin   if  a = 0  then if  b = 0  then if  c = 0  then exit   end;
```

```
procedure DisplayResults;
begin
        volume := a * b * c;   basesarea := 2 * a * b;
        sidearea := 2 * c * (a + b);   edgessum := 4 (a + b + c);
        write(“Όγκος ορθογώνιου : “,volume); writeln;
        write(“Εμβαδό βάσεων : “,basearea); writeln;
        write(“Εμβαδό παράπλευρης επιφάνειας : “,sidearea); writeln;
        write(“Συνολικό μήκος ακμών : “,edgesum); writeln
end.
```

2.14.2 Να γραφτεί με χρήση επανάληψης και διαδικασιών ένα πρόγραμμα το οποίο να δέχεται τα μήκη των πλευρών ορθογώνιου παραλληλεπιπέδου και να υπολογίζει τα εξής:

- τον όγκο του
- το εμβαδό των βάσεων
- το εμβαδό της παράπλευρης επιφάνειας
- το συνολικό μήκος των ακμών του.

Το πρόγραμμα σταματάει όταν δοθεί μηδενικές και οι τρεις διαστάσεις του κύβου.

```
program VolumeAAA;
begin
        TitleMessage;
        loop begin
                GetSide;   AskUserWToContinue;
                DisplayResults;
        end
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα υπολογίζει τον όγκο, το”);   writeln;
        write(“εμβαδό των βάσεων και της παράπλευρης”);   writeln;
        write(“επιφάνειας και το συνολικό μήκος των ακμών”);   writeln;
        write(“ κύβου πλευράς a.”);   writeln
end;
```

```
procedure GetSide;
        begin   write(“Δώστε πλευρά κύβου: “);   readln(a)   end;
```

```
procedure AskUserWToContinue;
        begin  if  a = 0  then exit  end;
```

```
procedure DisplayResults;
begin
        volume := a * a * a;   basesarea := 2 * a * a;
        sidearea := 4 * a * a;   edgessum := 12 * a;
        write(“Όγκος κύβου : “,volume); writeln;
        write(“Εμβαδό βάσεων : “,basearea); writeln;
        write(“Εμβαδό παράπλευρης επιφάνειας : “,sidearea); writeln;
        write(“Συνολικό μήκος ακμών : “,edgesum); writeln
end.
```

Ασκήσεις.

1. Να γραφτεί το πρόγραμμα με χρήση επανάληψης και διαδικασιών το οποίο δέχεται το μήκος, το πλάτος και το ύψος ενός ορθογώνιου παραλληλόγραμμου μετρημένα σε διάφορες μονάδες μέτρησης (μέτρα, δεκατόμετρα, εκατοστόμετρα ή χιλιοστόμετρα). Μετατρέπει τις μονάδες μέτρησης σε μετρά και υπολογίζει τον όγκο του ορθογώνιου παραλληλόγραμμου. Το πρόγραμμα σταματάει όταν δοθεί μηδενικό μήκος, πλάτος και ύψος.

Κεφάλαιο 2.15 – Μονάδες Μέτρησης του Χρόνου

2.15.1 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, ένα πρόγραμμα το οποίο να δέχεται τη διάρκεια ενός χρονικού διαστήματος σε δευτερόλεπτα και να τη μετατρέπει σε συμμιγή αριθμό, δηλαδή υπολογίζει πόσες ώρες, πόσα λεπτά και πόσα δευτερόλεπτα περιέχει η διάρκεια αυτή. Το πρόγραμμα σταματάει όταν δοθεί χρονικό διάστημα ίσο με το μηδέν.

```
program VolumeUnit;
begin
        TitleMessage;
        loop begin
                InputData;   AskUserWToContinue;
                DisplayResults
        end
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα μετατρέπει δευτερόλεπτα”);   writeln;
        write(“σε ώρες, λεπτά και δευτερόλεπτα.”);   writeln
end;
```

```
procedure InputData;
begin   write(“Δώστε χρονική διάρκεια σε δευτερόλεπτα : “); readln(seconds)   end;
```

```
procedure AskUserWToContinue;
        begin   if  seconds = 0  then exit   end;
```

```
procedure DisplayResults;
begin
        hours := seconds div 3600;   seconds := seconds mod 3600;
        minutes := seconds div 60;   seconds := seconds mod 60;
        write(hours,” ώρες, “,minutes,” λεπτά, “,seconds,” δευτερόλεπτα);   writeln
end.
```

Άσκηση.

1. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, ένα πρόγραμμα το οποίο να δέχεται τη διάρκεια ενός χρονικού διαστήματος σε δευτερόλεπτα, λεπτά ή ώρες και κάνει μετατροπή σε οποιαδήποτε από τις παραπάνω χρονικές μονάδες μέτρησης. Το πρόγραμμα σταματάει όταν δοθεί χρονικό διάστημα ίσο με το μηδέν.

Κεφάλαιο 2.16 – Μονάδες Μάζας

2.16.1 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, ένα πρόγραμμα το οποίο να δέχεται τη μάζα ενός σώματος σε τόνους, κιλά ή γραμμάρια και κάνει μετατροπή σε οποιαδήποτε από τις παραπάνω μονάδες μέτρησης μάζας. Το πρόγραμμα σταματάει όταν δοθεί χρονικό διάστημα ίσο με το μηδέν.

```
program MassTransformer;
begin
        TitleMessage;
        loop begin
                GetMass;   AskUserWToContinue;
                GetUnitA;   GetUnitB;
                Transform;   DisplayResults
        end
end;
```

```
procedure TitleMessage;
begin   write("Αυτό το πρόγραμμα κάνει μετατροπές μονάδων μάζας.");   writeln  end;
```

```
procedure GetMass;
        begin   write("Δώστε μάζα : ");   readln(volume)   end;
```

```
procedure AskUserWToContinue;
        begin   if  mass = 0  then exit   end;
```

```
procedure GetUnitA;
begin
        write("3 για τόνους, 2 για κιλά, 1 για γραμμάρια.");   writeln;
        loop begin
                write("Δώστε μονάδα μέτρησης (1 έως 3) : ");   readln(unitA);
                if unitA>0 then if unitA<4 then exit
        end
end;
```

```
procedure GetUnitB;
begin
        loop begin
                write(“Δώστε μονάδα μέτρησης αποτελέσματος (1 έως 3): “);
                readln(unitB);
                if unitB>0 then if unitB<4 then exit
        end
end;
```

```
procedure TransformA;
begin
        if  unitA > unitB  then begin                    (*Μετατροπή σε υποδιαίρεση*)
                steps := unitA – unitB;
                for steps times  mass := mass * 1000
        end
        else begin                                       (*Μετατροπή σε πολλαπλάσιο*)
                steps := unitB – unitA;
                for steps times mass := mass / 1000
        end
end;
```

```
procedure DisplayResults;
        begin   write(“Αποτέλεσμα : “,mass);   writeln   end.
```

Άσκηση.

1. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, ένα πρόγραμμα το οποίο να δέχεται τη μάζα ενός σώματος σε γραμμάρια και να τη μετατρέπει σε συμμιγή αριθμό, δηλαδή υπολογίζει πόσους τόνους, πόσα κιλά και πόσα γραμμάρια περιέχει η μάζα αυτή. Το πρόγραμμα σταματάει όταν δοθεί μάζα ίση με το μηδέν.

Κεφάλαιο 2.17 – Νομισματικές Μονάδες

2.17.1 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, ένα πρόγραμμα το οποίο να κάνει όλες τις δυνατές μετατροπές μεταξύ των παρακάτω νομισματικών μονάδων: Ευρώ, Δολάριο Η.Π.Α., Λίρα Αγγλίας και Γιέν Ιάπωνιας. Το πρόγραμμα σταματάει όταν δοθεί ποσό ίσο με το μηδέν. Το πρόγραμμα χρησιμοποιεί τον πίνακα ισοτιμιών και την τεχνική μετατροπής όλων των νομισμάτων σε Ευρώ.

```
program CurrencyTransformer;
begin
        TitleMessage;
        loop begin
                GetAmount;   AskUserWToContinue;
                GetUnitA;   GetUnitB;   Transform;   DisplayResults
        end
end;
```

```
procedure TitleMessage;
begin
        write("Αυτό το πρόγραμμα κάνει όλες τις δυνατές μετατροπές");  writeln;
        write("μεταξύ των παρακάτω νομισματικών μονάδων: Ευρώ,");  writeln;
        write("Δολάριο Η.Π.Α., Λίρα Αγγλίας και Γιέν Ιάπωνιας."); writeln
end;
```

```
procedure GetAmount;
        begin   write("Δώστε χρηματικό ποσό : ");   readln(amount)   end;
```

```
procedure AskUserWToContinue;
        begin   if  amount = 0  then exit   end;
```

```
procedure GetUnitA;
begin
        write("1 για Ευρώ, 2 για Δολάρια Η.Π.Α.,");   writeln;
        write("3 για Λίρα Αγγλίας, 4 για Γιέν Ιάπωνας (σε εκατοντάδες).");   writeln;
        loop begin
                write("Δώστε νόμισμα εισόδου (1 έως 4) : ");   readln(unitA);
                if unitA>0 then if unitA<5 then exit
        end
end;
```

```
procedure GetUnitB;
begin
        loop begin
                write("Δώστε νόμισμα αποτελέσματος (1 έως 4) : ");   readln(unitB);
                if unitA<>unitB then if unitB>0 then if unitB<5 then exit
        end
end;
```

```
procedure Transform;
begin
        if  unitA=2 then amount := amount / 0.9053;
        if  unitA=3 then amount := amount / 0.6272;
        if  unitA=4 then amount := amount / 119.587;
        (*Σε αυτό το σημείο, οποιοδήποτε ποσό έχει μετατραπεί σε Ευρώ*)
        if  unitB=2 then amount := amount * 0.8812;
        if  unitB=3 then amount := amount * 0.6105;
        if  unitB=4 then amount := amount * 116.406
end;
```

```
procedure DisplayResults;
        begin   write("Αποτέλεσμα : ",amount);   writeln   end.
```

Ασκήσεις.

1. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, ένα πρόγραμμα το οποίο να δέχεται ένα χρηματικό ποσό σε δραχμές και να το μετατρέπει σε ευρώ. Το πρόγραμμα σταματάει όταν δοθεί ποσό ίσο με το μηδέν.

2. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, ένα πρόγραμμα το οποίο να δέχεται ένα χρηματικό ποσό σε ευρώ και να το μετατρέπει σε δραχμές. Το πρόγραμμα σταματάει όταν δοθεί ποσό ίσο με το μηδέν.

3. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, ένα πρόγραμμα το οποίο να δέχεται ένα ποσό σε ευρώ και να υπολογίζει τα χαρτονομίσματα και τα κέρματα ελάχιστου πλήθους που το αποτελούν. Το πρόγραμμα σταματάει όταν δοθεί ποσό ίσο με το μηδέν.

Κεφάλαιο 3
Τα Κλάσματα

Κεφάλαιο 3.1 – Η Έννοια του Κλάσματος

3.1.1 Να γραφτεί, με χρήση επανάληψης και διαδικασιών,πρόγραμμα το οποίο να δέχεται έναν ακέραιο αριθμό a και ένα κλάσμα $\frac{b}{c}$ και να υπολογίζει έναν ακέραιο αριθμό d, τέτοιο ώστε να είναι ίσος με το μέρος $\frac{b}{c}$ του αριθμού a. Για παράδειγμα, αν a=150 και $\frac{b}{c}=\frac{1}{3}$ τότε d=50, διότι το $\frac{1}{3}$ του αριθμού 150 ισούται με 50. Αν το αποτέλεσμα δεν είναι ακέραιος αριθμός τότε το πρόγραμμα τον εμφανίζει στην οθόνη στρογυλοποιημένο στη μονάδα. Το πρόγραμμα σταματάει όταν δοθούν a=0, b=0 και c=1.

```
program QuantityAndRatio;
begin
        loop begin
                QuantityEntry;   FractionEntry;
                AskUserWToContinue;
                Computations;   QuantityAndRatioOutput
        end
end;
```

```
procedure QuantityEntry;
        begin   write("Δώσε την συνολική ποσότητα");   readln(a)   end;
```

```
procedure FractionEntry;
begin
        write("Δώσε τον αριθμητή του κλάσματος"); readln(b);
        loop begin
                write("Δώσε τον παρονομαστή του κλάσματος"); readln(c);
                if  c<>0  then exit;
        end
end;
```

```
procedure AskUserWToContinue;
        begin   if  a = 0  then if  b = 0  then if  c = 1  exit   end;
```

```
procedure QuantityEntry;
        begin   d := a * (b / c)   end;
```

```
procedure QuantityAndRatioOutput;
        begin   write(b,"/",c," του ",a," είναι ",d);   writeln   end.
```

3.1.2 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται δύο κλάσματα, να εξετάζει αν είναι ομώνυμα ή ετερώνυμα και να εμφανίζει το αντίστοιχο μήνυμα στην οθόνη. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program SameDenominator;
begin
        loop begin
                TwoFractionsEntry;   AskUserWToContinue;
                SameDenominatorOutput;
        end
end;
```

```
procedure TwoFractionsEntry;
begin
        write("Δώσε τον αριθμητή του πρώτου κλάσματος");   readln(num1);
        loop begin
                write("Δώσε τον παρονομαστή του πρώτου κλάσματος");   readln(den1);
                if den1<>0 then exit
        end;
        write("Δώσε τον αριθμητή του δεύτερου κλάσματος");   readln(num2);
        loop begin
                write("Δώσε τον παρονομαστή του δεύτερου κλάσματος");   readln(den2);
                if den2<>0 then exit
        end
end;
```

```
procedure AskUserWToContinue;
        begin   if num1=0 then if num2=0 then if den1=1 then if den2=1 exit   end;
```

```
procedure SameDenominatorOutput;
begin
        if  den1 <> den2  then write("Είναι ετερώνυμα.") else write("Είναι ομώνυμα.");
        writeln
end.
```

3.1.3 Να γραφτεί πρόγραμμα το οποίο να δέχεται σαν είσοδο μια συνολική ποσότητα και το πλήθος των τμημάτων στα οποία αυτή χωρίζεται και να υπολογίζει την ποσότητα του κάθε τμήματος.

```
program Fragments;
begin
        loop begin
                write("Δώστε το πλήθος των τμημάτων : ");   readln(fragments);
                if  fragments <> 0  then exit
        end;
        write("Δώστε τη συνολική ποσότητα : ");   readln(total);
        piece := total / fragments;
        write("το 1/",fragments," του ",total," είναι ",piece);   writeln
end.
```

3.1.4 Να γραφτεί, με χρήση διαδικασιών, πρόγραμμα το οποίο επιλύει προβλήματα με τη μέθοδο της **αναγωγής στη μονάδα**. Δέχεται ως είσοδο ένα κλάσμα $\frac{a}{b}$ και έναν αριθμό c. Οι δύο αυτές είσοδοι αποτελούν τα δεδομένα της μεθόδου, δηλαδή ισχύει ότι το μέρος $\frac{a}{b}$ μιας ποσότητας αντιστοιχεί στον αριθμό c. Για παράδειγμα τα $\frac{3}{5}$ ενός τραπεζικού λογαριασμού είναι 450 Ευρώ. Επίσης, το πρόγραμμα δέχεται ένα ακόμη κλάσμα $\frac{d}{b}$, ομώνυμο με το κλάσμα $\frac{a}{b}$, το οποίο αντιπροσωπεύει το μέρος της ποσότητας που ζητείται να υπολογιστεί. Για παράδειγμα, ζητούνται να υπολογιστούν τα $\frac{4}{5}$ του τραπεζικού λογαριασμού. Το πρόγραμμα, αν δοθούν οι είσοδοι του παραπάνω παραδείγματος, εμφανίζει στην οθόνη:

3/5	αντιστοιχούν σε	150
1/5	αντιστοιχεί σε	50
4/5	αντιστοιχούν σε	200

```
program UnitReduction;
begin
        FractionEntry;                                  (*Εισαγωγή του κλάσματος a / b*)
        QuantityEntry;   NumeratorEntry;   ReductionComputationsOutput
end;
```

```
procedure QuantityEntry;
begin
    write("Δώσε την ποσότητα στην οποία αντιστοιχεί το παραπάνω κλάσμα : "); readln(c)
end;
```

```
procedure NumeratorEntry;
begin
        write("Δώσε τον αριθμητή του ζητούμενου κλάσματος : ");   readln(d);
        write("Ο παρονομαστής του ζητούμενου κλάσματος είναι : ",b);   writeln
end;
```

```
procedure ReductionComputationsOutput;
begin
        writeln;   write(a,"/",b," αντιστοιχούν σε ",c);   writeln;
        unit := c/a;
        write(1,"/",b," αντιστοιχούν σε ",unit);   writeln;
        result := d * unit;
        write(d,"/",b," αντιστοιχούν σε ",result);   writeln
end.
```

Παρατηρεί κανείς ότι στο παραπάνω πρόγραμμα απουσιάζει η περιγραφή της διαδικασίας FranctionEntry. Αυτό συμβαίνει γιατί η διαδικασία αυτή έχει περιγραφεί στα πλαίσια προηγούμενου προγράμματος. Λόγω συντομίας, αυτό θα συμβεί και σε άλλα προγράμματα παρακάτω. Παρόλα αυτά, κατά τη συγγραφή ενός προγράμματος στον υπολογιστή είναι απαραίτητο να περιγραφούν όλες οι διαδικασίες που καλούνται από το κυρίως πρόγραμμα.

3.1.5 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται δύο ποσότητες, quantity1 και quantity2, και να υπολογίζει τι μέρος της πρώτης είναι η δεύτερη ποσότητα. Παράδειγμα:

quantity1=20, quantity 2=5, result=0.25

Το πρόγραμμα σταματάει όταν η πρώτη ποσότητα δοθεί ίση με το μηδέν.

```
program Quantities;
begin
        loop begin
                QuantityEntry1;   (* Εισαγωγή ποσότητας 1*)
                AskUserWToContinue;   QuantityEntry2;   (* Εισαγωγή ποσότητας 2*)
                QuantityComputationsOutput
        end
end;
```

```
procedure QuantityEntry1;
        begin   write("Δώσε την πρώτη ποσότητα : ");   readln(quantity1)   end;
```

```
procedure AskUserWToContinue;
        begin   if  quantity1=0  then exit   end;
```

```
procedure QuantityComputationsOutput;
begin
        result := quantity2 / quantity1;
        write("το ", quantity2," είναι το μέρος ",result," του ",quantity1);   writeln
end.
```

Ασκήσεις

1. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα που να δέχεται ένα μήκος Α σε cm και ένα μήκος Β σε mm και να υπολογίζει τι μέρος του Α είναι το Β.

2. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα που να δέχεται σαν είσοδο το πλήθος των τμημάτων στα οποία χωρίζεται η συνολική ποσότητα και την αξία κάθε τμήματος και να υπολογίζει τη συνολική ποσότητα.

3. Να γραφτεί το ίδιο πρόγραμμα για:
   - m και cm
   - kg και γραμμάρια
   - cm και dm

4. Να γραφτεί το πρόγραμμα 3.1.3 με χρήση διαδικασιών.

5. Να γραφτεί πρόγραμμα που να δέχεται την αρχική τιμή ενός προϊόντος και το ποσό της έκπτωσης και να εμφανίζει τι μέρος της αρχικής τιμής είναι η έκπτωση και τι μέρος της αρχικής τιμής πληρώσαμε τελικά.

6. Να γραφτεί πρόγραμμα που να δέχεται την αρχική τιμή ενός προϊόντος και το μέρος της έκπτωσης και να υπολογίζει την τελική τιμή που θα πληρώσουμε.

7. Να γραφεί πρόγραμμα που να δέχεται την τιμή των ¾ του κιλού και να εμφανίζει την τιμή των ⅝ του κιλού.

Κεφάλαιο 3.2 – Το Κλάσμα Ως Πηλίκο Δύο Φυσικών Αριθμών

3.2.1 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται δύο ακέραιους αριθμούς, διαιρετέο και διαιρέτη, και να εμφανίζει στην οθόνη το κλάσμα με το οποίο ισοδυναμεί η διαίρεση αυτή (σε μορφή $\frac{a}{b}$). Επίσης, να υπολογίζει το πηλίκο της διαίρεσης που ορίζεται από το κλάσμα. Το πρόγραμμα σταματάει όταν δοθεί αριθμητής ίσος με το μηδέν και παρονομαστής ίσος με τη μονάδα.

```
program IntegerDivision;
begin
        loop begin
                NumDividerEntry;   AskUserWToContinue;
                DivisionComputationsOutput
        end
end;
```

```
procedure NumDividerEntry;
begin
        write(“Δώσε τον διαιρετέο”); readln(num);
        loop begin
                write(“Δώσε τον διαιρέτη”);   readln(divider);   if divider<>0 then exit
        end
end;
```

```
procedure AskUserWToContinue;
        begin   if  num = 0  then if  divider = 1  then exit   end;
```

```
procedure DivisionComputationsOutput;
begin
        write(num);   writeln;
        write(“-----”);   writeln;
        write(divider);   writeln;   writeln;
        quotient := num / divider;   write(“Πηλίκο διαίρεσης : “, quotient);   writeln
end.
```

3.2.2 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται τον αριθμητή και τον παρονομαστή ενός κλάσματος και να ελέγχει αν το κλάσμα είναι ίσο με 0 ή 1.

```
program CheckZeroOne;
begin
        loop begin
                FractionEntry;   (*Εισαγωγή κλάσματος*)   ZeroOneOutput;
                AskUserWToContinue;
        end
end;
```

```
procedure ZeroOneOutput;
begin
        if  num = 0  then write (“Το κλάσμα ισούται με 0.”)
                     else if  num = den  then write(“Το κλάσμα ισούται με 1.”)
                                         else write(“Το κλάσμα δεν ισούται με 0 ή 1.”);
        writeln
end;
```

```
procedure AskUserWToContinue;
begin
        loop begin
                write (“Δώστε 1 για να επαναλάβετε, 0 για τέλος : “);   readln(flag);
                if  flag = 0  then exit;   if  flag = 1  then exit;
        end;
        if  flag = 0  then exit
end.
```

3.2.3 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται τον αριθμητή και τον παρονομαστή ενός κλάσματος και να ελέγχει αν το πηλίκο της διαίρεσης που αυτό ορίζει είναι ακέραιος αριθμός.

```
program IntegerCheck;
begin
        loop begin
                FractionEntry;                                  (*Εισαγωγή κλάσματος*)
                IntegerCheckComputationsOutput;
                AskUserWToContinue
        end
end;
```

```
procedure IntegerCheckComputationsOutput;
begin
        remainder := den mod num;
        if  remainder = 0  then
        begin
                quotient := ar div par;                          (*Υπολογισμός πηλίκου*)
                write(“Το κλάσμα ισούται με “, quotient)
        end;
        else write(“Το κλάσμα δεν ισούται με ακέραιο αριθμό.”);   writeln
end.
```

3.2.4 Να γραφτεί πρόγραμμα, με χρήση επανάληψης και διαδικασιών, το οποίο να επιλύει την εξίσωση $\frac{x-a}{b} = 0$, όπου τα a, b δίνονται από το χρήστη.

```
program SolveEq;
begin
        loop begin
                ParametersEntry;   SolveEqComputationsOutput;
                AskUserWToContinue
        end
end;
```

```
procedure ParametersEntry;
begin
        write(“Δώστε την παράμετρο a : “);   readln(a);
        loop begin
                write(“Δώστε την παράμετρο b : “);   readln(b);   if b<>0 then exit
        end
end;
```

```
procedure SolveEqComputationsOutput;
        begin   write(“x = “,a);   writeln   end.
```

Ασκήσεις

1. Να βρείτε ποια διαίρεση παριστάνει το καθένα από τα κλάσματα και να γράψετε πρόγραμμα που να την τυπώνει: $\frac{3}{7}$, $\frac{1}{19}$, $\frac{35}{19}$, $\frac{12}{3}$.

2. Να γραφτεί πρόγραμμα που να υπολογίζει σε κιλά το κλάσμα μιας αρχικής ποσότητας (σε κιλά) που θα δίνεται από τον χρήστη.

3. Να λυθούν εξισώσεις της μορφής: $\frac{x+a}{b}=0$, $\frac{x+a}{b}=1$, $\frac{a-x}{b}=1$, $\frac{x+a}{b}=-1$

Κεφάλαιο 3.3 – Ισοδύναμα Κλάσματα

3.3.1 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται ένα κλάσμα $\frac{a}{b}$ και ένα ακέραιο αριθμό n, να κατασκευάζει και να εμφανίζει στην οθόνη n ισοδύναμα κλάσματα με αυτό. Το πρόγραμμα σταματάει όταν δοθούν a=0 και b=1.

```
program EquivalentFractions;
begin
        loop begin
                FractionEntry;   AskUserWToContinue;
                EquivalentFractionsEntry;   i := 2;
                for n times do begin
                        EquivalentFractionsComputation;
                        EquivalentFractionsOutput;   i:=i+1
                end
        end
end;
```

```
procedure EquivalentFractionsEntry;
begin
        loop begin
        write("Δώσε αριθμό ισοδύναμων κλασμάτων");   readln(n);   if n>0 then exit
        end
end;
```

```
procedure AskUserWToContinue;
        begin   if  num = 0  then if  den = 1  then exit   end;
```

```
procedure EquivalentFractionsComputation;
        begin    newnum := num * i;   newden:= den * i   end;
```

```
procedure EquivalentFractionsOutput;
        begin    write(num," / ",den," = ",newnum," / ",newden);   writeln   end.
```

3.3.2 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται δύο κλάσματα, να εξετάζει αν είναι ισοδύναμα και να εμφανίζει το αντίστοιχο μήνυμα στην οθόνη. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program CheckEquivalent;
begin
        TwoFractionsEntry;
        AskUserWToContinue; CheckEquivalentComputationOutput
end.
```

```
procedure AskUserWToContinue;
        begin   if num1=0 then if den1=1 then if num2=0 then if den2=1 then exit   end;
```

```
procedure CheckEquivalentComputationOutput;
begin
        flag := num1 * den2 – num2 * den1;
        if flag = 0 then  write("Τα κλάσματα είναι ισοδύναμα.")
                        else  write("Τα κλάσματα δεν είναι ισοδύναμα.");
        writeln
end.
```

3.3.3 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται ένα κλάσμα $\frac{a}{b}$, εξετάζει αν το κλάσμα είναι ανάγωγο και εμφανίζει στην οθόνη κατάλληλο μήνυμα. Αν όχι, υπολογίζει το ισοδύναμο ανάγωγο κλάσμα. Το πρόγραμμα σταματάει όταν δοθούν a=0 και b=1.

```
program FractionSimplification;
begin
        loop begin
                FractionEntry;   AskUserWToContinue;   FindMkd;
                if mkd=1 then SimplificationOutput1
                        else begin  SimplificationComputation;  SimplificationOutput2  end
        end
end;
```

```
procedure AskUserWToContinue;
        begin   if  num = 0  then if  den = 1  then exit   end;
```

```
procedure FindMkd;
begin
        a:=num;   b:=den;
        loop begin
                if  a = 0 then exit;   if  b = 0 then exit;
                if  a > b  then  a := a mod b  else  b := b mod a
        end;
        mkd := a + b
end;
```

```
procedure SimplificationOutput1;
        begin   write("Το κλάσμα είναι ανάγωγο.");   writeln   end;
```

```
procedure SimplificationComputation;
        begin   newnum := num div mkd;   newden := den div mkd   end;
```

```
procedure SimplificationOutput2;
begin
        write(“Το κλάσμα δεν είναι ανάγωγο.”);   writeln;
        write(num,“ / “,den,” = “,newnum,” / “,newden);   writeln
end.
```

3.3.4 Έστω δύο κλάσματα $\frac{x}{a}$ και $\frac{b}{c}$. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται τους αριθμούς a, b και c και υπολογίζει τον αριθμό x, ώστε τα δύο κλάσματα να είναι ισοδύναμα. Το πρόγραμμα σταματάει όταν δοθούν a=0, b=1 και c=0.

```
program SolveEq1;
begin
        loop begin
                ABCEntry;   AskUserWToContinue;
                SolveEqComputation;   SolveEqOutput
        end
end;
```

```
procedure ABCEntry;
begin
        loop begin
                write(“Δώστε την παράμετρο a : “);   readln(den1);   if  den1 <> 0  then exit
        end
        write(“Δώστε την παράμετρο b : “);   readln(num2);
        loop begin
                write(“Δώστε την παράμετρο c : “);   readln(den2);   if  den2 <> 0  then exit
        end
end;
```

```
procedure AskUserWToContinue;
        begin   if  den1 = 1  then if  den2 = 1  then if  num2 = 0  then exit   end;
```

```
procedure SolveEqComputation;
        begin   if  den1 = den2  then   x := num2   else   x:= num1 * num2 / den2   end;
```

```
procedure SolveEqOutput;
        begin   write(“Η λύση είναι x = “,x);   writeln   end.
```

Ασκήσεις

1. Να γράψετε πρόγραμμα που να μετατρέπει τους αριθμούς 4, 8, 1, 13 σε κλάσματα με παρονομαστή το 13.

2. Να γράψετε πρόγραμμα που να μετατρέπει το κλάσμα ⅜ σε ισοδύναμο με παρονομαστή 16, 24, 64, 132.

3. Να γράψετε πρόγραμμα που να μετατρέπει το κλάσμα ⅜ σε ισοδύναμο με αριθμητή 6, 12, 21, 42.

4. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται δύο ακέραιους αριθμούς a και b, και εμφανίζει στην οθόνη τον αριθμό a γραμμένο υπό μορφή κλάσματος με παρονομαστή ίσο με b. Για παράδειγμα αν a=4 και b=7, τότε το πρόγραμμα εμφανίζει:
   4 = 28 / 7
   Το πρόγραμμα σταματάει όταν δοθούν a=0 και b=1.

5. Να γράψετε πρόγραμμα που να λύνει εξισώσεις της μορφής $\frac{\alpha}{x}=\frac{\beta}{\gamma}$. Να εξετάσετε χωριστά την περίπτωση β = 1.

6. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται ένα κλάσμα $\frac{a}{b}$ και έναν ακέραιο αριθμό c, και να εμφανίζει στην οθόνη, αν είναι δυνατό, ένα κλάσμα ισοδύναμο με το $\frac{a}{b}$ με παρονομαστή ίσο με c. Αν δεν είναι δυνατό, τότε το πρόγραμμα εμφανίζει στην οθόνη κατάλληλο μήνυμα. Το πρόγραμμα σταματάει όταν δοθούν a=0, b=1 και c=0.

Κεφάλαιο 3.4 – Σύγκριση Κλασμάτων

3.4.1 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται δύο ομώνυμα κλάσματα, να τα συγκρίνει και τα εμφανίζει στην οθόνη με το κατάλληλο σύμβολο ανάμεσά τους (<, > ή =). Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program EquivalentFractionsComparison;
begin
        loop begin
                EquivalentFractionsEntry;
                AskUserWToContinue;   EquivalentComparisonOutput
        end
end;
```

```
procedure EquivalentFractionsEntry;
begin
        write("Δώσε αριθμητή του πρώτου κλάσματος : ");   readln(num1);
        write("Δώσε αριθμητή του δεύτερου κλάσματος : ");   readln(num2);
        loop begin
                write("Δώσε τον κοινό παρονομαστή : "); readln(den);
                if den<>0 then exit;
        end
end;
```

```
procedure AskUserWToContinue;
        begin   if num1= 0 then if num2=0 then if den=1 then exit   end;
```

```
procedure EquivalentComparisonOutput;
begin
        if  num1 = num2  then write(num1," / ",den," = ",num2," / ",den)
                        else if ar1> ar2 then write(num1," / ",den," > ",num2," / ",den)
                                        else write(num1," / ",den," < ",num2," / ",den)
end.
```

3.4.2 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται δύο κλάσματα με τον ίδιο αριθμητή, να τα συγκρίνει και τα εμφανίζει στην οθόνη με το κατάλληλο σύμβολο ανάμεσά τους (<, > ή =). Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program SameNumeratorFractionsComparison;
begin
        loop begin
                SameNumeratorFractionsEntry;
                AskUserWToContinue;   SameNumeratorComparisonOutput;
        end
end;
```

```
procedure SameNumeratorFractionsEntry;
begin
        write("Δώσε τον κοινό αριθμητή : "); readln(num);
        loop begin
                write("Δώσε τον παρονομαστή του πρώτου κλάσματος"); readln(den1);
                if den1<>0 then exit
        end;
        loop begin
                write("Δώσε τον παρονομαστή του δεύτερου κλάσματος"); readln(den2);
                if den2<>0 then exit
        end
end;
```

```
procedure AskUserWToContinue;
        begin   if num = 0 then if  den1 = 1  then if  den2 = 1  then exit   end;
```

```
procedure SameNumeratorComparisonOutput;
begin
        if den1 = den2 then write(num," / ",den1," = ",num," / ", den2)
                        else if  den1 > den2  then write(num," / ",den1," < ",num," / ",den2)
                                              else write(num," / ",den1," > ",num," / ",den2);
        writeln
end.
```

3.4.3 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται δύο κλάσματα, και να τα τρέπει σε ομώνυμα. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές ίσοι με τη μονάδα.

```
program FractionsToEquivalent;
begin
        loop begin
                TwoFractionsEntry;   AskUserWToContinue;
                if den1 <> den2 then
                begin
                        FindLcm;                                (*Μετατροπή σε ομώνυμα*)
                        FractionsToEquivalentComputation
                end;
                FractionsToEquivalentOutput
        end
end;
```

```
procedure AskUserWToContinue;
        begin   if num1=0 then if num2=0 then if den1=1 then if den2=1 then exit   end;
```

```
procedure FindLcm;
begin
        a:=den1;   b:=den2;
        loop begin
                if  a = 0 then exit;   if  b = 0 then exit;
                if  a > b  then  a := a mod b  else  b := b mod a
        end;
        mkd := a + b;                                           (*Υπολογισμός ΜΚΔ*)
        lcm:= a* b div mkd                                      (*Υπολογισμός ΕΚΠ*)
end;
```

```
procedure FractionsToEquivalentComputation;
begin
        temp:= lcm div den1;   num1:=num1 * temp;   den1:= lcm;
        temp:= lcm div den2;   num2:=num2 * temp;   den2:= lcm
end;
```

```
procedure FractionsToEquivalentOutput;
begin
        write(“Πρώτο κλάσμα : “,num1,” / ,” den1);   writeln;
        write(“Δεύτερο κλάσμα : “,num2,” / ,” den2);   writeln
end.
```

3.4.4 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται δύο κλάσματα, να τα συγκρίνει και τα εμφανίζει στην οθόνη με το κατάλληλο σύμβολο ανάμεσά τους (<, > ή =). Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program FractionsComparison;
begin
        loop begin
                TwoFractionsEntry;   AskUserWToContinue;
                if den1<>den2 then
                begin  FindLcm;  FractionsToEquivalentComputation  end;
                EquivalentComparisonOutput
        end
end.
```

3.4.5 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται δύο ακέραιους αριθμούς a και b και να υπολογίζει ένα κλάσμα μεγαλύτερο από το $\frac{a}{b}$ και μικρότερο από το $\frac{a+1}{b}$. Το πρόγραμμα σταματάει όταν δοθούν α=0 και b=1.

```
program IntermediaryFraction;
begin
        loop begin
                ParametersEntry;   AskUserWToContinue;
                IntermediaryFractionComputationOutput
        end
end;
```

```
procedure AskUserWToContinue;
        begin  if  a = 0  then if  b = 1  then  exit   end;
```

```
procedure IntermediaryFractionComputationOutput;
begin
        par := b * 2;   ar := a * 2 + 1;
        write(a, ' / ',b,' < ',ar,' / ',par,' < ',a+1,' / ',b);   writeln
end.
```

Ασκήσεις

1. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται ένα κλάσμα, να το συγκρίνει μα τη μονάδα και να εμφανίζει στην οθόνη κατάλληλο μήνυμα. Το πρόγραμμα σταματάει όταν δοθεί αριθμητής ίσος με το μηδέν και παρονομαστής ίσος με τη μονάδα.

2. Να γραφεί πρόγραμμα που να δέχεται ένα κλάσμα και να υπολογίζει:
    a. ένα μεγαλύτερο κλάσμα με
        i. διαφορετικό παρονομαστή
        ii. διαφορετικό αριθμητή
    b. ένα μικρότερο κλάσμα με
        i. διαφορετικό παρονομαστή
        ii. διαφορετικό αριθμητή

3. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα που να προσθέτει μία σταθερά στον αριθμητή και τον παρονομαστή ενός κλάσματος και να υπολογίζει αν το νέο κλάσμα είναι μικρότερο από το αρχικό.

Κεφάλαιο 3.5 – Πρόσθεση Κλασμάτων

3.5.1 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται δύο ομώνυμα κλάσματα, τα προσθέτει και εμφανίζει το άθροισμά τους στη οθόνη. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program EquivalentSum;
begin
        loop begin
                EquivalentFractionsEntry;   AskUserWToContinue;
                EquivalentFractionsSum;   FractionsSumOutput
        end
end;
```

```
procedure AskUserWToContinue;
        begin   if num1=0 then if num2=0 then if den=1 then  exit   end;
```

```
procedure EquivalentFractionsSum;
        begin   num3 := num1 + num2;   den3:= den   end;
```

```
procedure FractionsSumOutput;
begin
        write(num1,"/ ",den," + ",num2,"/ ",den," = ",num3,"/ ",den3);   writeln
end.
```

3.5.2 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται δύο όχι απαραίτητα ομώνυμα κλάσματα, τα προσθέτει και εμφανίζει το άθροισμά τους στη οθόνη. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program FractionsSum;
begin
        loop begin
            TwoFractionsEntry;   AskUserWToContinue;
            if den1<>den2 then begin  FindLcm;  FractionsToEquivalentComputation  end;
            EquivalentFractionsSum;   FractionsSumOutput
        end
end.
```

```
procedure AskUserWToContinue;
        begin   if num1=0 if num2=0 then if den1=1 then if den2=1 then exit   end;
```

Ασκήσεις

1. Να γράψετε ένα πρόγραμμα που να εκτελεί πρόσθεση μεταξύ δύο κλασμάτων και το οποίο να μη χρησιμοποιεί το ελάχιστο κοινό πολλαπλάσιο των δύο παρονομαστών για να τα κάνει ομώνυμα. Το αποτέλεσμα θα εμφανίζεται υπό μορφή κλάσματος.

2. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται ένα κλάσμα και έναν ακέραιο αριθμό, τα προσθέτει και εμφανίζει το άθροισμά τους στη οθόνη. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

3. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται ένα κλάσμα, να το μετατρέπει σε μεικτό αριθμό και να το εμφανίζει στη οθόνη. Το πρόγραμμα σταματάει όταν δοθεί αριθμητής ίσος με το μηδέν και παρονομαστής ίσος με τη μονάδα.

4. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται έναν μεικτό αριθμό , να τον μετατρέπει σε κλάσμα και να το εμφανίζει στη οθόνη. Το πρόγραμμα σταματάει όταν δοθεί ο μεικτός αριθμός $0\frac{0}{1}$.

5. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται δύο μεικτούς αριθμούς, να τους προσθέτει και να εμφανίζει το άθροισμα στην οθόνη σε μορφή μικτού αριθμού. Το πρόγραμμα σταματάει όταν δοθεί ο μεικτοί αριθμοί ίσοι με $0\frac{0}{1}$.

Κεφάλαιο 3.6 – Αφαίρεση Κλασμάτων

3.6.1 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται δύο ομώνυμα κλάσματα, τα αφαιρεί και εμφανίζει τη διαφορά τους στη οθόνη. Σημειώστε ότι το πρόγραμμα πρέπει να ελέγχει να είναι δυνατό να γίνει η αφαίρεση. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program EquivalentSub;
begin
        loop begin
            EquivalentFractionsEntry;   AskUserWToContinue;
            if num1>num2 then begin   EquivalentFractionsSub;   FractionsSubOutput   end
                                else SubErrorOutput
        end
end;
```

```
procedure AskUserWToContinue;
        begin   if num1=0 then if num2=0 then if den=1 then  exit   end;
```

```
procedure EquivalentFractionsSub;
        begin   num3 := num1 + num2;   den3 := den   end;
```

```
procedure FractionsSubOutput;
begin
        write(num1,"/ ",den,"  -  ",num2,"/ ",den,"  =  ",num3,"/ ",den3);   writeln
end;
```

```
procedure SubErrorOutput;
        begin   write("Δεν είναι δυνατή η αφαίρεση.");   writeln   end.
```

3.6.2 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται δύο κλάσματα, τα αφαιρεί και εμφανίζει τη διαφορά τους στη οθόνη. Σημειώστε ότι το πρόγραμμα πρέπει να ελέγχει να είναι δυνατό να γίνει η αφαίρεση. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program FractionsSub;
begin
        loop begin
                TwoFractionsEntry;   AskUserWToContinue;
                if den1<>den2 then
                        begin   FindLcm;   FractionsToEquivalentComputation   end;
                if num1>num2 then
                        begin   EquivalentFractionsSub;   FractionsSubOutput   end
                                else SubErrorOutput
        end
end;
```

```
procedure AskUserWToContinue;
        begin   if num1=0 if num2=0 then if den1=1 then if den2=1 then exit   end.
```

Ασκήσεις

1. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα που να εκτελεί αφαίρεση μεταξύ δύο κλασμάτων χωρίς να χρησιμοποιεί το ελάχιστο κοινό πολλαπλάσιο των δύο παρονομαστών για να τα κάνει ομώνυμα. Το αποτέλεσμα θα εμφανίζεται υπό μορφή κλάσματος. Θα πρέπει να γίνονται όλοι οι απαραίτητοι έλεγχοι.

2. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται δύο μεικτούς αριθμούς, να τους προσθέτει και να εμφανίζει το άθροισμά στην οθόνη σε μορφή μικτού αριθμού. Το πρόγραμμα τερματίζει όταν δοθεί ο μεικτοί αριθμοί ίσοι με $0\frac{0}{1}$.

Κεφάλαιο 3.7 – Πολλαπλασιασμός Κλασμάτων

3.7.1. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται δύο κλάσματα, τα πολλαπλασιάζει και εμφανίζει τη διαφορά τους στη οθόνη. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program Multiplication;
begin
        loop begin
                TwoFractionsEntry;   AskUserWToContinue;
                FractionsMultiplication;   FractionsMultOutput
        end
end;
```

```
procedure FractionsMultiplication;
        begin   num3 := num1 * num2;   den3 := den1 * den2   end;
```

```
procedure FractionsMultOutput;
begin
        write(num1,"/ ",den," * ",num2,"/ ",den," = ",num3,"/ ",den3);   writeln
end.
```

Άσκηση

1. Ένα μαγαζί κάνει έκπτωση σε όλα του τα είδη ίση με $\frac{10}{115}$ της αξίας κάθε προϊόντος. Να γραφεί πρόγραμμα το οποίο να δέχεται τιμές προϊόντων (μία κάθε φορά), να υπολογίζει και να εμφανίζει την τιμή του προϊόντος μετά την έκπτωση. Το πρόγραμμα να σταματάει όταν δοθεί τιμή προϊόντος ίση με το μηδέν.

Κεφάλαιο 3.8 – Αντίστροφοι Αριθμοί

3.8.1 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται έναν ακέραιο αριθμό και να υπολογίζει τον αντίστροφό του. Το πρόγραμμα σταματάει όταν δοθεί μηδέν.

```
program Reverse;
begin
    loop begin
        write("Δώσε τον αριθμό : ");   readln(num);
        if num=0 then   exit   else begin
            rev:= 1 / num;
            write("Ο αντίστροφος του ", num," είναι ο 1/",num," = ",rev);   writeln
        end
    end
end.
```

Ασκήσεις

1. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται ένα κλάσμα και να υπολογίζει το αντίστροφό του. Το πρόγραμμα τερματίζει όταν δοθεί αριθμητής ίσος με το μηδέν και παρονομαστής ίσος με τη μονάδα.

2. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται δύο κλάσματα και να εξετάζει αν το πρώτο είναι αντίστροφο του δεύτερου. Το πρόγραμμα τερματίζει όταν δοθούν αριθμητές ίσοι με το μηδέν και παρονομαστές ίσοι με τη μονάδα.

Κεφάλαιο 3.9 – Διαίρεση Κλασμάτων

Υπενθυμίζουμε ότι η διαίρεση 2 κλασμάτων ισοδυναμεί με πολλαπλασιασμό του διαιρετέου με τον αντίστροφο του διαιρέτη. Δηλαδή $\boxed{\frac{\alpha}{\beta}:\frac{\gamma}{\delta}=\frac{\alpha}{\beta}\cdot\frac{\delta}{\gamma}}$

3.9.1 Στο παρακάτω πρόγραμμα πραγματοποιείται διαίρεση δυο κλασμάτων. Καθώς εισάγει ο χρήστης καθένα αριθμητή και παρονομαστή του κάθε κλάσματος, γίνεται έλεγχος αν οι παρονομαστές είναι μη μηδενικοί. Τα ονόματα των μεταβλητών είναι num1 για τον αριθμητή και den1 για τον παρονομαστή του πρώτου κλάσματος, num2 για τον αριθμητή και den2 για τον παρονομαστή του δεύτερου κλάσματος και τέλος num3 για τον αριθμητή και den3 για τον παρονομαστή του αποτελέσματος.

```
program Division;
begin
        loop begin
                TwoFractionsEntry;   AskUserWToContinue;
                FractionsDivision;   FractionsDivOutput
        end
end;
```

```
procedure FractionsDivision;
        begin   num3:=num1 * den2;   den3:= den1 * num2   end;
```

```
procedure FractionsDivOutput;
begin
        write(num1,"/ ",den," / ",num2,"/ ",den," = ",num3,"/ ",den3);   writeln
end.
```

Εμφάνιση στην οθόνη
Δώστε τον αριθμητή του πρώτου κλάσματος:1
Δώστε τον παρονομαστή του πρώτου κλάσματος:0
Δώστε τον παρονομαστή του πρώτου κλάσματος:2
Δώστε τον αριθμητή του δεύτερου κλάσματος:3
Δώστε τον παρονομαστή του δεύτερου κλάσματος:4
1/2 / 3/4 = 4/6

3.9.2 Μπορούμε να συνενώσουμε τα προγράμματα πρόσθεσης, αφαίρεσης πολλαπλασιασμού και διαίρεσης κλασμάτων σε ένα που να διαβάζει τα κλάσματα και το σύμβολο της πράξης.
Στο παρακάτω πρόγραμμα ο χρήστης εισάγει τους αριθμητές και τους παρονομαστές των δύο κλασμάτων. Ο υπολογιστής τους διαβάζει και ελέγχει αν οι παρονομαστές είναι μη μηδενικοί. Στη συνέχεια ο χρήστης εισάγει το σύμβολο της πράξης, ο υπολογιστής το ελέγχει και υπολογίζει το αποτέλεσμα. Στην περίπτωση που έχουμε δώσει το δεύτερο κλάσμα μεγαλύτερο από το πρώτο και έχουμε αφαίρεση η πράξη δε μπορεί να γίνει οπότε εμφανίζεται κατάλληλο μήνυμα στην οθόνη. Τα ονόματα των μεταβλητών είναι ίδια με τα προηγούμενα προγράμματα. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program Prakseis;
begin
    loop begin
        TwoFractionsEntry;  AskUserWToContinue;  SymbolEntry;
        if symbol = 1 then
        begin
                if den1<>den2 then
                        begin  FindLcm;  FractionsToEquivalentComputation end;
                EquivalentFractionsSum;  FractionsSumOutput
        end;
        if symbol = 2 then
                begin
                        if den1<>den2 then
                                begin  FindLcm;  FractionsToEquivalentComputation  end;
                        if num1>num2 then
                                begin  EquivalentFractionsSub;  FractionsSubOutput  end
                                        else SubErrorOutput
                end;
        if symbol = 3 then begin  FractionsMultiplication;  FractionsMultOutput  end;
        if symbol = 4 then begin  FractionsDivision;  FractionsDivOutput  end
    end
end;
```

```
procedure SymbolEntry;
begin
        loop begin
            write("Δώστε το σύμβολο της πράξης, 1 για πρόσθεση, 2 για");  writeln;
            write("αφαίρεση, 3 για πολλαπλασιασμό, 4 για διαίρεση : ");  readln(symbol);
            if symbol < 1 then exit;  if symbol > 4 then exit
        end
end.
```

Εμφάνιση στην οθόνη
Δώστε τον αριθμητή του πρώτου κλάσματος : 2
Δώστε τον παρονομαστή του πρώτου κλάσματος : 0
Δώστε τον παρονομαστή του πρώτου κλάσματος : 3
Δώστε τον αριθμητή του δεύτερου κλάσματος : 3
Δώστε τον παρονομαστή του δεύτερου κλάσματος : 4
Δώστε το σύμβολο της πράξης, 1 για πρόσθεση, 2 για αφαίρεση, 3 για πολλαπλασιασμό, 4 για διαίρεση : 2
Δεν είναι δυνατή η αφαίρεση.

Άσκηση

1. Να τροποποιηθεί το πρόγραμμα 3.9.2 ώστε να δέχεται σύνθετα κλάσματα τα οποία πρώτα να μετατρέπει σε απλά και στη συνέχεια να εκτελεί τις πράξεις.

Κεφάλαιο 3.10 – Δεκαδικά Κλάσματα

3.10.1 Το παρακάτω πρόγραμμα μετατρέπει δεκαδικά κλάσματα σε δεκαδικούς αριθμούς με τόσα δεκαδικά ψηφία όσα μηδενικά έχει ο παρονομαστής τους. Για καθένα μηδενικό του παρονομαστή μετακινεί την υποδιαστολή του αριθμητή κατά μια θέση προς τα αριστερά. Στην αρχή βέβαια γίνεται έλεγχος αν το κλάσμα που δόθηκε είναι όντως δεκαδικό. Το πρόγραμμα σταματάει όταν δοθεί αριθμητής κλάσματος ίσος με το μηδέν.

```
program Decimal1;
begin
        loop begin
                DecimalFractionEntry;   AskUserWToContinue;
                DecimalFractionTransform;   DecimalFractionOutput;
        end
end;
```

```
procedure DecimalFractionEntry;
begin
        write (“Δώστε τον αριθμητή του κλάσματος : “);   readln(a);
        loop begin
                write(“Δώστε τον παρονομαστή του κλάσματος,”);   writeln;
                write(“(ακέραιο πολλαπλάσιο του 10 αλλά όχι μηδέν : “);   readln (b);
                if b<>0 then if  b mod 10 = 0  then exit
        end
end;
```

```
procedure AskUserWToContinue;
        begin   if  a = 0  then exit   end;
```

```
procedure DecimalFractionTransform;
begin
        loop begin
                b := b mod 10;   a := a / 10;   if b=0 then exit
        end
end;
```

```
procedure DecimalFractionOutput;
        begin   write("Ο δεκαδικός αριθμός είναι : ", a);   writeln   end.
```

Κεφάλαιο 3.11 – Τροπή Κλάσματος σε Δεκαδικό

3.11.1 Το παρακάτω πρόγραμμα μετατρέπει κλάσματα σε δεκαδικούς και μικτούς αριθμούς. Το πρόγραμμα σταματάει όταν δοθεί αριθμητής κλάσματος ίσος με το μηδέν και παρονομαστής ίσος με τη μονάδα.

```
program Decimal2;
begin
        loop begin
                FractionEntry;   AskUserWToContinue;
                DecimalFractionComputation;   DecimalFractionOutput;
                if a>=b then begin   MixedComputation;  MixedOutput   end
        end
end;
```

```
procedure AskUserWToContinue;
        begin   if  a = 0  then if  b = 1  then exit   end;
```

```
procedure DecimalFractionComputation;
        begin   c:=a/b   end;
```

```
procedure MixedComputation;
        begin   d:=a div b;   e:=a mod b   end;
```

```
procedure MixedOutput;
        begin   write("Ο μικτός αριθμός είναι : ",d,"  ",e,"/",b);   writeln   end.
```

Άσκηση

1. Να βελτιωθεί το πρόγραμμα που πραγματοποιεί τις τέσσερις πράξεις των κλασμάτων έτσι ώστε να γράφει το αποτέλεσμα σε μορφή μικτού και δεκαδικού αριθμού.

Κεφάλαιο 3.12 – Η Έννοια του Ποσοστού

3.12.1 Ένα κατάστημα την περίοδο των εκπτώσεων κάνει στα παντελόνια έκπτωση 15% και στις μπλούζες έκπτωση 20%. Το παρακάτω πρόγραμμα δέχεται την αρχική τιμή του κάθε ρούχου και υπολογίζει το ποσό της έκπτωσης και το συνολικό ποσό που πρέπει να πληρώσει ο πελάτης.

```
program Discount;
begin
        loop begin
                TrousersEntry;   TShirtsEntry;   DiscountComputations;
                DiscountOutput;   AskUserWToContinue;
        end
end;
```

```
procedure TrousersEntry;
begin
        write (“Δώστε τον αριθμό των παντελονιών : “);   readln (trousers);
        sumtrousers:=0;
        for i:=1 to trousers do
        begin
                write (“Δώστε την αρχική τιμή του παντελονιού:”);   readln (pricetrousers);
                sumtrousers:= sumtrousers + pricetrousers
        end
end;
```

```
procedure TShirtsEntry;
begin
        write (“Δώστε τον αριθμό των T-shirt : “);   readln (tshirts);
        sumtshirts:=0;
        for i:=1 to tshirts do
        begin
                write(“Δώστε την αρχική τιμή του T-shirt : “);   readln(pricetshirts);
                sumtshirts := sumtshirts + pricetshirts
        end
end;
```

```
procedure DiscountComputations;
begin
        discounttrousers := sumtrousers * 0,15;
        sumtrousers := sumtrousers - discounttrousers;
        discounttshirts := sumtshirts * 0,2;
        sumtshirts := sumtshirts - discounttshirts;
        sum := sumtrousers+sumtshirts;
        discount := discounttrousers + discounttshirts
end;
```

```
procedure DiscountOutput;
begin
        write (“Συνολικό ποσό : “,sum,”ευρώ”);   writeln;
        write (“Συνολική έκπτωση : “,discount,”ευρώ”);   writeln
end;
```

```
procedure AskUserWToContinue;
begin
        loop begin
                write (“Δώστε 1 για να επαναλάβετε, 0 για τέλος : “);   readln(flag);
                if  flag = 0  then exit;   if  flag = 1  then exit;
        end;
        if  flag = 0  then exit
end.
```

Εμφάνιση στην οθόνη

Δώστε τον αριθμό των παντελονιών : 2
Δώστε την αρχική τιμή του παντελονιού : 70
Δώστε την αρχική τιμή του παντελονιού : 65
Δώστε τον αριθμό των T-shirt : 3
Δώστε την αρχική τιμή του T-shirt : 28
Δώστε την αρχική τιμή του T-shirt : 35
Δώστε την αρχική τιμή του T-shirt : 40
Συνολικό ποσό : 197,15 ευρώ
Συνολική έκπτωση : 40,85 ευρώ

3.12.2 Το παρακάτω πρόγραμμα δέχεται την αρχική τιμή και την τελική τιμή ενός προϊόντος και υπολογίζει την έκπτωση και το ποσοστό της έκπτωσης. Αν η τελική τιμή είναι υψηλότερη της αρχικής, τότε το πρόγραμμα τυπώνει κατάλληλο μήνυμα στην οθόνη.

```
program Percentage;
begin
        write (“Δώστε την αρχική τιμή του προϊόντος : “);   readln(initial);
        write (“Δώστε την τελική τιμή μετά την έκπτωση : “);   readln(final);
        discount := initial - final;
        if  discount > 0  then begin
                discountpercentage := discount / initial;
                discountpercentage := discountpercentage * 100;
                write (“Έκπτωση : “,discount,”%”);   writeln;
                write (“Ποσοστό έκπτωσης : “,discountpercentage,”%”)
        end
                        else write (“Δεν έχει γίνει έκπτωση στην αρχική τιμή.”);
        writeln
end.
```

Άσκηση

1. Να γραφεί το πρόγραμμα 3.12.2 με χρήση επανάληψης και διαδικασιών.

Κεφάλαιο 4

Ανάλογα Ποσά

Κεφάλαιο 4.1 – Η Έννοια των Ανάλογων Ποσών

4.1.1 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται ζεύγη αντίστοιχων τιμών δύο ποσών. Για κάθε ζεύγος που εισάγεται, το πρόγραμμα πρέπει να εξετάζει αν τα ποσά προκύπτουν ανάλογα για όλα τα ζεύγη που έχουν ήδη εισαχθεί. Αν τα ποσά πράγματι είναι ανάλογα, τότε το πρόγραμμα ρωτά τον χρήστη αν επιθυμεί να εισάγει νέο ζεύγος, ή όχι, οπότε εμφανίζει κατάλληλο μήνυμα στην οθόνη. Αν τα ποσά δεν είναι ανάλογα, τότε το πρόγραμμα σταματά και εμφανίζει στην οθόνη κατάλληλο μήνυμα.

```
program ProportionalSums;
begin
        IntroductionProportionalSums;  GetValue1;  GetValue2;  ComputeRatio1;
        loop begin
                GetValue1;  GetValue2;  ComputeRatio2;
                if  ratio1 <> ratio2  then Negative else Positive;
                AskUserWToContinue;
        end
end;
```

```
procedure IntroductionProportionalSums;
begin
        write("Αυτό το πρόγραμμα εξετάζει τα ζεύγη αντίστοιχων"); writeln;
        write("τιμών δύο ποσών και εξετάζει αν τα δύο ποσά"); writeln;
        write("είναι ανάλογα."); writeln
end;
```

```
procedure GetValue1;
        begin  write("Δώστε την πρώτη τιμή του ζεύγους : ");  readln(value1)  end;
```

```
procedure GetValue2;
begin
    loop begin
        write("Δώστε τη δεύτερη τιμή του ζεύγους (όχι όμως μηδέν) : ");  readln(value2);
        if  value2 <> 0  then exit
    end
end;
```

```
procedure ComputeRatio1;
        begin   ratio1 := value2 / value1   end;
```

```
procedure ComputeRatio2;
        begin   ratio2 := value2 / value1   end;
```

```
procedure Negative;
        begin   write("Τα ποσά δεν είναι ανάλογα!");   writeln   end;
```

```
procedure Positive;
        begin   write("Τα ποσά είναι ανάλογα!");   writeln   end.
```

Κεφάλαιο 4.2 – Εφαρμογές των Ανάλογων Ποσών

4.2.1 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται ένα ζεύγος τιμών δύο ανάλογων ποσών και μία τιμή ενός δεύτερου ζεύγους. Το πρόγραμμα υπολογίζει την τιμή που αντιστοιχεί στην τελευταία είσοδο του χρήστη. Σχηματικά, το πρόγραμμα υπολογίζει τον άγνωστο Χ του παρακάτω πίνακα, με την προϋπόθεση ότι τα δύο ποσά είναι ανάλογα.

Ποσό Α	a1 = 2	a2 = 3
Ποσό Β	b1 = 400	b2 = X

```
program SolveProportionalSums;
begin
        IntroductionSolveProportionalSums;
        loop begin
                GetA1;  GetB1;  GetA2;  ComputeOutputProportionalSums;
                AskUserWToContinue;
        end
end;
```

```
procedure IntroductionSolveProportionalSums;
begin
        write(“Αυτό το πρόγραμμα δέχεται ένα ζεύγος τιμών “); writeln;
        write(“ανάλογων ποσών και μια τιμή από ένα δεύτερο ζεύγος “); writeln;
        write(“και υπολογίζει τη δεύτερη τιμή του ζεύγους. “); writeln
end;
```

```
procedure GetA1;
begin
        loop begin
                write(“Δώστε την πρώτη τιμή του πρώτου ζεύγους (όχι μηδέν) : “);
                readln(a1);   if  a1 <> 0  then exit
        end
end;
```

```
procedure GetB1;
begin
        loop begin
                write(“Δώστε τη δεύτερη τιμή του πρώτου ζεύγους (όχι μηδέν) : “);
                readln(b1);   if  b1 <> 0  then exit
        end
end;
```

```
procedure GetA2;
begin
        loop begin
                write(“Δώστε την πρώτη τιμή του δεύτερου ζεύγους (όχι μηδέν) : “);
                readln(a2);   if  a2 <> 0  then exit
        end
end;
```

```
procedure ComputeOutputProportionalSums;
begin
        b2 := b1 * a2 / a1;
        write(“Δεύτερη τιμή, δεύτερου ζεύγους : “,b2);   writeln
end.
```

4.2.2 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται την αρχική τιμή ενός προϊόντος, το ποσοστό της έκπτωσης ή της αύξησης της τιμής και να υπολογίζει την τελική τιμή του.

```
program ComputePrices;
begin
        IntroductionComputePrices;
        loop begin
                GetFirstPrice;   GetPercentage;   ComputeOutputComputePrices;
                AskUserWToContinue;
        end
end;
```

```
procedure IntroductionComputePrices;
begin
        write("Αυτό το πρόγραμμα δέχεται την αρχική τιμή ενός"); writeln;
        write("προϊόντος, το ποσοστό της έκπτωσης ή της αύξησης"); writeln;
        write("της τιμής και υπολογίζει την τελική τιμή του."); writeln
end;
```

```
procedure GetFirstPrice;
begin
        loop begin
                write("Δώστε την αρχική τιμή του προϊόντος : ");   readln(firstprice);
                if  firstprice <> 0  then exit
        end
end;
```

```
procedure GetPercentage;
begin
        loop begin
                write("Αύξηση ή μείωση τιμής. Δώστε 1 για αύξηση 0 για μείωση : ");
                readln(plusminus);
                if plusminus=0 then exit;   if plusminus=1 then exit
        end;
        write("Δώστε ποσοστό επί τοις εκατό : ");   readln(percentage)
end;
```

```
procedure ComputeOutputComputePrices;
begin
        change := firstprice * percentage / 100;
        if plusminus=1 then  finalprice := firstprice + change
                       else  finalprice := firstprice – change;
        write(“Τελική τιμή : “,finalprice);   writeln
end.
```

4.2.3 Να γραφτεί, με χρήση διαδικασιών, πρόγραμμα το οποίο να δέχεται το κεφάλαιο που κατατίθεται στην τράπεζα, το επιτόκιο και να υπολογίζει τα χρήματα που θα εισπραχθούν μετά από ένα έτος.

```
program Bank;
begin
        IntroductionBank;   GetCapital;   GetInterest;   ComputeShowNewCapital;
end;
```

```
procedure IntroductionBank;
begin
        write(“Το πρόγραμμα δέχεται το κεφάλαιο που κατατίθεται”); writeln;
        write(“στην τράπεζα, το επιτόκιο και να υπολογίζει τα”); writeln;
        write(“ χρήματα που θα εισπραχθούν μετά από ένα έτος.”); writeln
end;
```

```
procedure GetCapital;
begin
        loop begin
                write(“Δώστε το αρχικό κεφάλαιο : “);   readln(capital);
                if  capital > 0  then exit
        end
end;
```

```
procedure GetInterest;
begin
        loop begin
                write(“Δώστε το επιτόκιο : “);   readln(interest);
                if  interest > 0  then exit
        end
end;
```

```
procedure ComputeShowNewCapital;
begin
        newcapital := capital + (capital * interest);
        write(“Κεφάλαιο το επόμενο έτος : “,newcapital);   writeln
end.
```

4.2.4 Ένα σιδερένιο σώμα έχει τη μορφή ορθογωνίου παραλληλεπιπέδου με μήκος length1, πλάτος width1, ύψος height1 και βάρος weight1. Ένα δεύτερο ορθογώνιο παραλληλεπίπεδο από το ίδιο υλικό έχει μήκος length2, πλάτος width2, ύψος height2 και βάρος weight2. Να γραφτεί, με χρήση διαδικασιών, πρόγραμμα το οποίο να δέχεται τις διαστάσεις των δύο ορθογωνίων παραλληλεπιπέδων και το βάρος του πρώτου και να υπολογίζει το βάρος του δεύτερου. Το πρόγραμμα σταματάει όταν δοθούν: length1 = 0, width1 = 0 και height1 = 0.

Τα βάρη των ορθογωνίων παραλληλεπιπέδων είναι ανάλογα του όγκου τους. Ο όγκος του ορθογωνίου παραλληλογράμμου υπολογίζεται από το γινόμενο του μήκους, του πλάτους και του ύψους του.

```
program Rectangle;
begin
    loop begin
        GetDimensions1;   AskUserWToContinue;
        GetWeight1;   GetDimensions2;   ComputeShowWeight2
    end
end;
```

```
procedure GetDimensions1;
begin
        write(“Δώστε το μήκος του 1ου ορθογωνίου : “);   readln(length1);
        write(“Δώστε το πλάτος του 1ου ορθογωνίου : “);   readln(width1);
        write(“Δώστε το ύψος του 1ου ορθογωνίου : “);   readln(height1)
end;
```

```
procedure AskUserWToContinue;
        begin   if length1=0 then if width1=0 then if height1=0 then exit   end;
```

```
procedure GetWeight1;
begin
    loop begin  write(“Δώστε το βάρος του 1ου ορθογωνίου : “);   readln(weight1)   end
end;
```

```
procedure GetDimensions2;
begin
        write(“Δώστε το μήκος του 2ου ορθογωνίου : “);   readln(length2);
        write(“Δώστε το πλάτος του 2ου ορθογωνίου : “);   readln(width2);
        write(“Δώστε το ύψος του 2ου ορθογωνίου : “);   readln(height2)
end;
```

```
procedure ComputeShowWeight2;
begin
        volume1 := length1 * width1 * height1;
        volume2 := length2 * width2 * height2;
        weight2 := volume2 * weight1 / volume1;
        write(“Δώστε το βάρος του 2ου ορθογωνίου : “,weight2);   writeln
end.
```

Ασκήσεις

1. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται την αρχική και την τελική τιμή ενός προϊόντος και να υπολογίζει το ποσοστό έκπτωσης ή αύξησης της τιμής. Το πρόγραμμα σταματάει όταν δοθεί μηδενική αρχική και τελική τιμή.

2. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται την τελική τιμή και το ποσοστό έκπτωσης ή αύξησης της τιμής ενός προϊόντος και να υπολογίζει την αρχική του τιμή. Το πρόγραμμα σταματάει όταν δοθεί μηδενική τελική τιμή.

3. Να γραφτεί, με βάση το πρόγραμμα 4.2.3, ένα πρόγραμμα το οποίο να δέχεται το κεφάλαιο που κατατίθεται στην τράπεζα, το επιτόκιο, ο αριθμός των ετών και να υπολογίζει τα χρήματα που θα εισπραχθούν μετά το πέρας των ετών που εισάγει ο χρήστης. Προσέξτε ότι ο τόκος κάθε έτους προστίθεται στο κεφάλαιο και το άθροισμα αποτελεί το κεφάλαιο του επόμενου έτους.

Κεφάλαιο 4.3 – Κλίμακες

4.3.1 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται την απόσταση δύο σημείων στο χάρτη και την κλίμακα του χάρτη και να υπολογίζει την πραγματική απόσταση. Το πρόγραμμα σταματάει όταν δοθεί μηδενική απόσταση στο χάρτη.

```
program Climax;
begin
        loop begin
                GetMapDistance;   AskUserWToContinue;
                GetMapClimax;   ComputeShowRealDistance
        end
end;
```

```
procedure GetMapDistance;
        begin  write(“Δώστε την απόσταση στο χάρτη : “);  readln(mapdistance)   end;
```

```
procedure AskUserWToContinue;
        begin   if mapdistance=0 then exit   end;
```

```
procedure GetMapClimax;
begin
        write(“Η κλίμακα του χάρτη δίνεται στη μορφή 1/α”);   writeln;
        write(“Δώστε το α : “);   readln(mapclimax)
end;
```

```
procedure ComputeShowRealDistance;
begin
        realdistance := mapdistance * mapclimax;
        write(“Η πραγματική απόσταση είναι : “, realdistance);   writeln
end.
```

Ασκήσεις

1. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται την πραγματική απόσταση δύο σημείων και την κλίμακα του χάρτη και να υπολογίζει την απόσταση στο χάρτη. Το πρόγραμμα σταματάει όταν δοθεί μηδενική πραγματική απόσταση.

2. Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να δέχεται την απόσταση δύο σημείων στο χάρτη και την πραγματική απόσταση και να υπολογίζει την κλίμακα του χάρτη. Το πρόγραμμα σταματάει όταν δοθεί μηδενική απόσταση στο χάρτη και μηδενική πραγματική απόσταση.

Κεφάλαιο 4.4 – Μερισμός σε Μέρη Ανάλογα

4.4.1 Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο να λύνει προβλήματα μερισμού σε μέρη ανάλογα, βάσει της σχέσης:

$$\frac{x}{\alpha}=\frac{y}{\beta}=\frac{z}{\gamma}=\frac{x+y+z}{\alpha+\beta+\gamma}$$

Το πρόγραμμα δέχεται τις τιμές x, y, z και το άθροισμα α+β+γ και υπολογίζει τις τιμές α, β και γ. Το πρόγραμμα σταματάει όταν δοθεί μηδενικό άθροισμα α + β + γ ή όταν δοθούν συγχρόνως οι τιμές x = 0, y = 0 και z = 0.

```
program ProportionalSeparation;
begin
    IntroductionProportionalSeparation;
    loop begin
        GetX;  GetY;  GetZ;  AskUserWToContinue;
        GetABC;  if abc=0 then exit;
        ProportionalSeparationComputeResults;  ProportionalSeparationShowResults
    end
end;
```

```
procedure IntroductionProportionalSeparation;
begin
        write("Το πρόγραμμα λύνει προβλήματα μερισμού σε μέρη ανάλογα");  writeln
end;
```

```
procedure GetX;
        begin  write("Δώστε την τιμή του x : ");  readln(x)  end;
```

```
procedure GetY;
        begin  write("Δώστε την τιμή του y : ");  readln(y)  end;
```

```
procedure GetZ;
        begin  write("Δώστε την τιμή του z : ");  readln(z)  end;
```

```
procedure AskUserWToContinue;
        begin   if x=0 then if y=0 then if z=0 then exit   end;
```

```
procedure GetABC;
        begin   write(“Δώστε την τιμή του αθροίσματος a+b+c : “);   readln(abc)   end;
```

```
procedure ProportionalSeparationComputeResults;
        begin   xyz := x+y+z;   l := xyz/abc;   a := x/l;   b := y/l;   c := z/l  end;
```

```
procedure ProportionalSeparationShowResults;
begin
        write(“Η τιμή του α είναι : “, a);   writeln;
        write(“Η τιμή του β είναι : “, b);   writeln;
        write(“Η τιμή του γ είναι : “, c);   writeln
end.
```

Κεφάλαιο 5
Συναρτήσεις και Αριθμοί

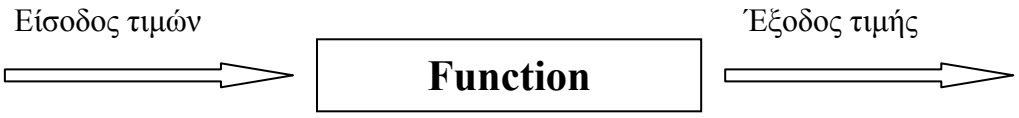
ΣΥΝΑΡΤΗΣΕΙΣ

Στο κεφάλαιο 2 έγινε εισαγωγή στις διαδικασίες. Υπενθυμίζεται ότι οι **διαδικασίες (procedures)** είναι υποπρογράμματα. Χρησιμοποιούνται ως δομικά στοιχεία για την κατασκευή του κυρίως προγράμματος. Στο παρόν κεφάλαιο θα γίνει εισαγωγή στις συναρτήσεις.

Οι **συναρτήσεις** είναι και αυτές υποπρογράμματα και χρησιμοποιούνται ως δομικά στοιχεία για την κατασκευή προγραμμάτων. Παρακάτω παρουσιάζονται οι διαφορές διαδικασιών και συναρτήσεων.

Η βασικότερη διαφορά είναι ότι οι συναρτήσεις, κατά την κλήση τους από το κυρίως πρόγραμμα, δέχονται από αυτό μία ή περισσότερες τιμές, που ονομάζονται **παράμετροι**. Η ιδιότητα αυτή δίνει τη δυνατότητα στον προγραμματιστή να καλεί στο κυρίως πρόγραμμα την ίδια συνάρτηση με διαφορετικές παραμέτρους.

Εκτός αυτού, κάθε συνάρτηση μετά την εκτέλεσή της, επιστρέφει στο κυρίως πρόγραμμα ένα αποτέλεσμα, το οποίο μπορεί το κυρίως πρόγραμμα να αποθηκεύει σε κάποια μεταβλητή.



Όπως και στις διαδικασίες, έτσι και στις συναρτήσεις θα παρουσιαστούν ξεχωριστά η **κλήση** και η **περιγραφή συνάρτησης**.

Η εμφάνιση μιας συνάρτησης στο κυρίως πρόγραμμα ονομάζεται **κλήση ή αποτίμηση συνάρτησης**, επειδή με αυτό τον τρόπο το κυρίως πρόγραμμα καλεί τη συνάρτηση.

Η κλήση συνάρτησης έχει την εξής γενική μορφή:

ΌνομαΜεταβλητής := *ΌνομαΣυνάρτησης*(*Όρισμα1*,*Όρισμα2*, . . . , *ΌρισμαΝ*);

Η κλήση συνάρτησης διαφέρει από την περιγραφή της.

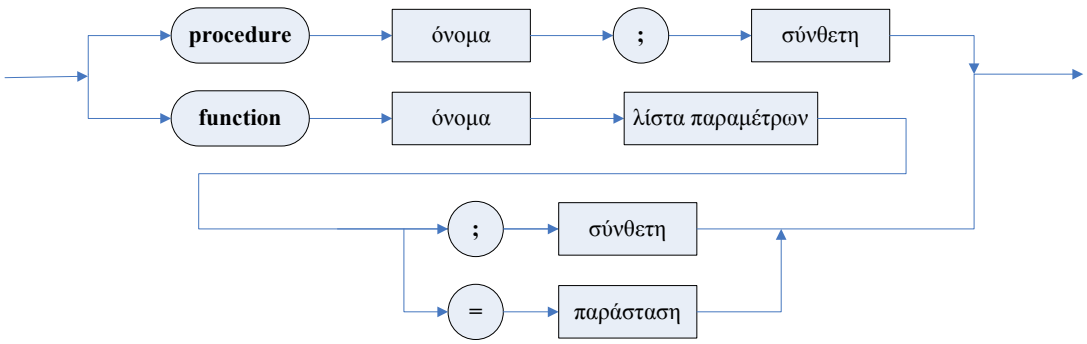
Η **περιγραφή** μιας **συνάρτησης** έχει τις εξής δύο γενικές μορφές:

- **function** *ΟνομαΣυνάρτησης*(*Όρισμα1*,*Όρισμα2*, . . . , *ΌρισμαΝ*)*;*
  Σύνθετη Εντολή;
- **function** *ΟνομαΣυνάρτησης*(*Όρισμα1*,*Όρισμα2*, . . . , *ΌρισμαΝ*) =
  ΜαθηματικήΠαράσταση;

  Όπου *ΜαθηματικήΠαράσταση* είναι μια παράσταση μου περιλαμβάνει αριθμούς και μεταβλητές.

Υπενθυμίζεται ότι **σύνθετη εντολή** ονομάζεται μια ακολουθία εντολών που περικλείεται από τις δεσμευμένες λέξεις **begin** και **end**.

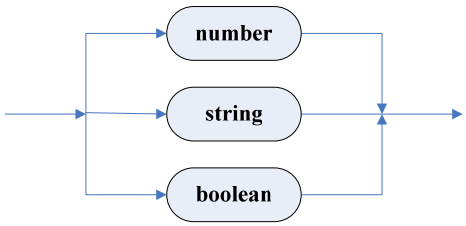
Τα συντακτικά διαγράμματα της γλώσσας αναδιαμορφώνονται έτσι ώστε να περιλαμβάνουν και τις συναρτήσεις:



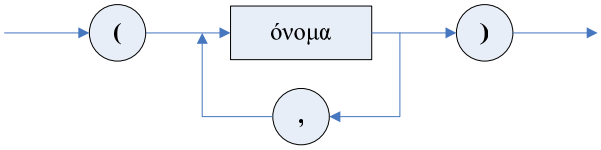
Συντακτικό διάγραμμα: ***υποπρόγραμμα***

ΠΡΟΣΟΧΗ!!

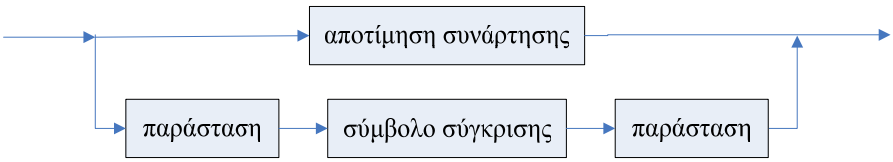
Τονίζεται ότι οι μεταβλητές που χρησιμοποιούνται μέσα στην περιγραφή των συναρτήσεων είναι νέες και διαφορετικές από τις μεταβλητές του κυρίως προγράμματος. Αυτό συμβαίνει γιατί οι συναρτήσεις δεν μπορούν να αλλάξουν, κατά την εκτέλεσή τους, τιμές μεταβλητών του κυρίως προγράμματος.



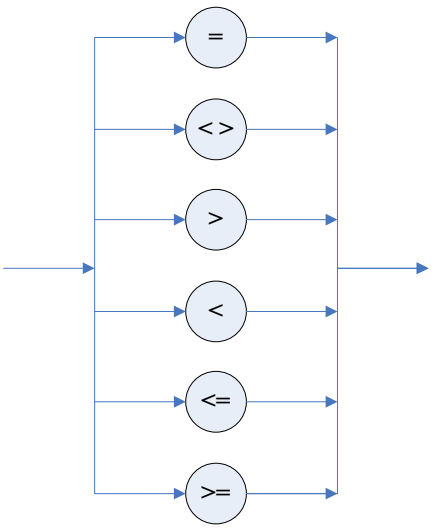
Συντακτικό διάγραμμα: ***τύπος τιμής***



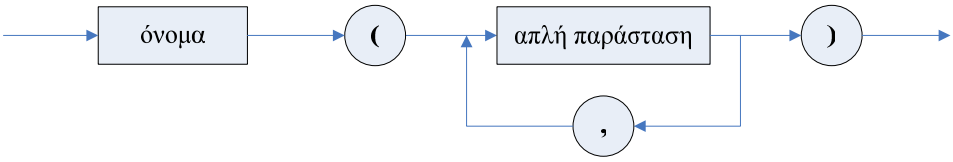
Συντακτικό διάγραμμα: ***λίστα παραμέτρων***



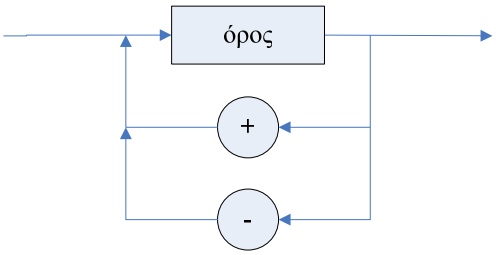
Συντακτικό διάγραμμα: ***συνθήκη***



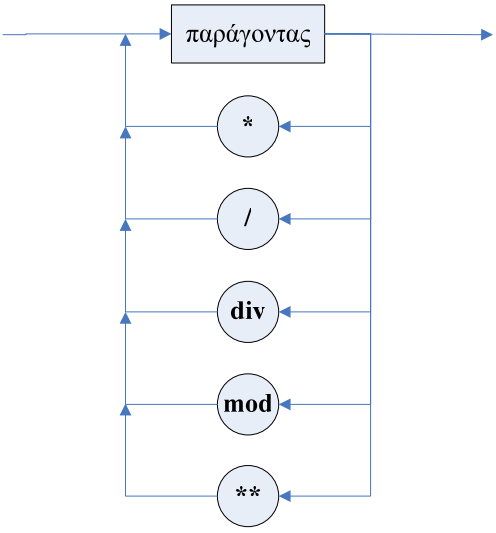
Συντακτικό διάγραμμα: ***σύμβολο σύγκρισης***



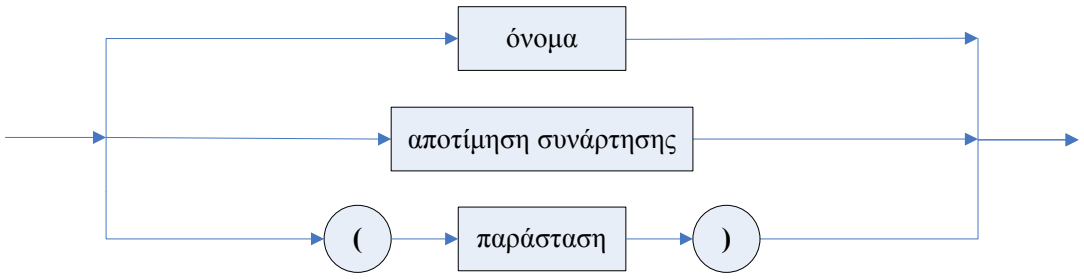
Συντακτικό διάγραμμα: ***αποτίμηση συνάρτησης***



Συντακτικό διάγραμμα: ***παράσταση***



Συντακτικό διάγραμμα: ***παράσταση***



Συντακτικό διάγραμμα: ***παράγοντας***

Ο προγραμματισμός με συναρτήσεις επιτρέπει στον προγραμματιστή να δημιουργεί **μία συνάρτηση για κάθε ανεξάρτητη λογική ενότητα** του προγράμματος. Επιπλέον, αν υπάρχουν επαναλαμβανόμενες λογικές ενότητες με διαφορετικές μεταβλητές, τότε αρκεί η δημιουργία μιας μόνο συνάρτησης με ορίσματα τις μεταβλητές αυτές.

Ακολουθούν παραδείγματα προγραμματισμού με functions.

Παράδειγμα 1:

```
program Hellofunc;
begin
        i:=1;
        for number(i) times do write(“Hello World !!!”);   writeln
end;
```

```
function number(i) = i * 2 + 5.
```

Παράδειγμα 2:

```
program Variables;
        begin   res:= result(2,4);   write(res);   writeln   end.
```

```
function result(big, small) = big + small + 10.
```

Παράδειγμα 3:

Συνάρτηση που επιστρέφει το ελάχιστο δύο αριθμών:

```
function min(x, y);
      begin   if x<y then return x else return y   end.
```

Συνάρτηση που επιστρέφει το μέγιστο δύο αριθμών:

```
function max(x, y);
      begin   if x>y then return x else return y   end.
```

Παράδειγμα 4:

```
program MathExpression;
begin
        for i:=1 to 4 do
        begin
                j:= i+1;   write("x = ",i,"   y = ",j);   writeln;
                res := Expression(i, j);   write("Αποτέλεσμα : ",res);   writeln
        end
end;
```

```
function Expression(x, y);
begin
        temp1 := x * 8 + y * 5;   temp2 := y / 10;
        result := temp1 * temp2;   return result
end.
```

output
x =1 y =2
αποτέλεσμα : 3,6
x =2 y =3
αποτέλεσμα : 9,3
x =3 y =4
αποτέλεσμα : 17,6
x =4 y =5
αποτέλεσμα : 28,5
x =5 y =6
αποτέλεσμα : 42

Παράδειγμα 5:

Function για πρόσθεση, αφαίρεση, πολλαπλασιασμό και διαίρεση:

```
function Add(a, b) = a + b

function Subtract(a, b) = a – b

function Multiply(a, b) = a * b

function Divide(a, b) = a / b
```

Στα κεφάλαια 5, 6 και 7 θα παρουσιαστούν προγράμματα που υλοποιήθηκαν στα κεφάλαια 1, 2, 3 και 4, υλοποιημένα με τη χρήση συναρτήσεων.

Κεφάλαιο 5.1 – Φυσικοί Αριθμοί με Συναρτήσεις

5.1.1 (Άσκηση εξοικείωσης 3, κεφάλαιο 1) Να γραφεί, με χρήση συναρτήσεων, πρόγραμμα το οποίο θα υπολογίζει το άθροισμα των αριθμών μέχρι το 50, δηλαδή:

$$sum = 1 + 2 + 3 + 4 + ... + 47 + 48 + 49 + 50$$

```
program NaturalSum;
        begin   result := Sum( );   write(result);   writeln   end;
```

```
function Sum( );
begin
        sum := 0;   for i:=1 to 50 do   sum := sum + i;
        return sum
end.
```

Ασκήσεις

1. Να γραφτεί με χρήση συναρτήσεων το πρόγραμμα της άσκησης εξοικείωσης 4 του κεφαλαίου 1.
2. Να γραφτεί με χρήση συναρτήσεων το πρόγραμμα της άσκησης εξοικείωσης 5 του κεφαλαίου 1.
3. Να γραφτεί με χρήση συναρτήσεων το πρόγραμμα της άσκησης εξοικείωσης 6 του κεφαλαίου 1.
4. Να γραφτεί με χρήση συναρτήσεων το πρόγραμμα της άσκησης εξοικείωσης 7 του κεφαλαίου 1.
5. Να γραφτεί με χρήση συναρτήσεων το πρόγραμμα της άσκησης εξοικείωσης 8 του κεφαλαίου 1.

Κεφάλαιο 5.2 – Σύγκριση Φυσικών Αριθμών με Συναρτήσεις

5.2.1 (Πρόγραμμα 1.3.1) Να γραφτεί, με χρήση συναρτήσεων, πρόγραμμα το οποίο να δέχεται δύο αριθμούς και να τους εμφανίζει στην οθόνη με το κατάλληλο σύμβολο ανισότητας ανάμεσά τους.

```
program CompareProgram;
begin
        write(“Δώστε τον πρώτο αριθμό : “); readln(num1);
        write(“ Δώστε το δεύτερο αριθμό : “); readln(num2);
        result:=compare(num1, num2);
        if result=1 then write(num1,” > “,num2)
                    else if result=2 then write(num1,” < “,num2)
                                     else if result:=0 then write(num1,” = “,num2);
        writeln
end;
```

```
function Compare(num1, num2);
begin
        if num1>num2 then compare:=1
                     else if num1<num2 then compare:=2
                                       else compare:=0;
        return compare
end.
```

5.2.2 (Πρόγραμμα 1.4.1) Να γραφτεί, με χρήση συναρτήσεων, πρόγραμμα το οποίο να δέχεται έναν αριθμό και να τον εμφανίζει στην οθόνη στρογγυλοποιημένο στη δεκάδα.

```
program Round10Program;
begin
        write(“Δώστε έναν αριθμό : “);   readln(num);
        result := Round10(num);
        write(“Αποτέλεσμα : “,result);   writeln
end;
```

```
function Round10(num);
begin
        lastdigit := num mod 10;   nwoutld := num – lastdigit;
        if lastdigit>4 then nwoutld := nwoutld + 10;
        return nwoutld
end.
```

Άσκηση

1. Να γραφτεί το πρόγραμμα 1.4.2. με χρήση συναρτήσεων.

Κεφάλαιο 5.3 – Μεταβλητές με Συναρτήσεις

5.3.1 (Πρόγραμμα 1.5.1) Το παρακάτω πρόγραμμα δέχεται ως είσοδο την τιμή που κοστίζει μια τηλεφωνική μονάδα. Έπειτα, δέχεται το πλήθος των μονάδων μιας τηλεφωνικής συνδιάλεξης και υπολογίζει το κόστος της. Κατόπιν, επιστρέφει στην κατάσταση αναμονής του πλήθους των μονάδων τηλεφωνικής συνδιάλεξης και η διαδικασία συνεχίζεται έτσι, μέχρι ο χρήστης να εισάγει τον αριθμό 0.

```
program Telephone;
begin
        write("Τιμή μονάδας (σε λεπτά) : ");   readln(unitcost);
        loop begin
                writeln;
                write("Πλήθος μονάδων : ");   readln(units);
                if units=0 then exit;
                result := Multiply(unitcost, units);
                write("Κόστος συνδιάλεξης : ",result," λεπτά");   writeln
        end
end.
```

5.3.2 (Πρόγραμμα 1.5.2) Να γραφτεί, με χρήση συναρτήσεων, πρόγραμμα το οποίο να υπολογίζει την αξία διαδρομών με ταξί, αν υποθέσουμε ότι η αξία της σημαίας είναι 90 λεπτά και πληρώνουμε 20 λεπτά για κάθε χιλιόμετρο απλής ταρίφας και 36 λεπτά για κάθε χιλιόμετρο διπλής ταρίφας. Το πρόγραμμα να σταματά όταν δίνονται ως είσοδος 0 χιλιόμετρα απλής ταρίφας και 0 χιλιόμετρα διπλής ταρίφας.

```
program Taxi;
begin
        loop begin
                write("Τιμή απλής ταρίφας :");   readln(simpletarifprice);
                write("Χιλιόμετρα απλής ταρίφας : ");   readln(simpletariffkm);
                write("Τιμή διπλής ταρίφας :");   readln(doubletarifprice);
                write("Χιλιόμετρα διπλής ταρίφας : ");   readln(doubletariffkm);
                if  simpletariffkm = 0  then if  doubletariffkm = 0  then exit;
                price1 := Multiply(simpletarifprice, simpletarifkm);
                price2 := Multiply(doubletarifprice, doubletarifkm);
                c:= Cost(price1, price2);
                write("Κόστος διαδρομής : ",c ," λεπτά");   writeln
        end
end;
```

```
function Cost(s, d) = 90 + s + d.
```

Κεφάλαιο 5.4 – Εξισώσεις με Συναρτήσεις

5.4.1 (Πρόγραμμα 1.6.2) Το παρακάτω πρόγραμμα λύνει την εξίσωση x + a = b. Δέχεται ως είσοδο την τιμή το a και του b και υπολογίζει την τιμή του x.

```
program Equation1;
begin
        write("Αυτό το πρόγραμμα λύνει την εξίσωση  x + a = b."); writeln;
        write("Δώστε την τιμή του a :"); readln(a);
        write("Δώστε την τιμή του b :"); readln(b);
        result := Subtract(b, a);
        write("Αποτέλεσμα :  x = ",result); writeln
end;
```

5.4.2 (Πρόγραμμα 1.7.2) Το παρακάτω πρόγραμμα δέχεται ως είσοδο το μήκος της πλευράς ενός τετραγώνου και υπολογίζει την περίμετρό του.

```
program SquarePerimeter;
begin
        write("Μήκος πλευράς τετραγώνου : "); readln(a);
        result := PerimeterCompute(a);
        write("Περίμετρος τετραγώνου : ",result); writeln
end;
```

```
function PerimeterCompute(a) = 4 * a.
```

Άσκηση

1. Να γραφτεί πρόγραμμα, με χρήση συναρτήσεων, για τον υπολογισμό της περιμέτρου ορθογωνίου παραλληλογράμμου. Να δέχεται ως είσοδο το μήκος και το πλάτος του.

Κεφάλαιο 5.5 – Πρόσθεση και Αφαίρεση με Συναρτήσεις

5.5.1 (Ιστορικό Σημείωμα «Οι Τρίγωνοι Αριθμοί») Το παρακάτω πρόγραμμα δέχεται ως είσοδο έναν αριθμό και υπολογίζει τον αντίστοιχο τρίγωνο αριθμό. Για παράδειγμα, με είσοδο 3 υπολογίζει τον τρίτο τρίγωνο αριθμό δηλαδή έχει έξοδο 6.

```
program TriangleNumbers;
begin
        write("Δώστε έναν αριθμό : ",a);   readln(a);
        result := Triangle(a);
        write(a,"ος τρίγωνος αριθμός : ",result);   writeln
end;
```

```
function Triangle(b);
        begin   sum := 0;   for i:=1 to b do sum := sum + i;   return sum   end.
```

5.5.2 (Πρόγραμμα 1.8.1) Το παρακάτω πρόγραμμα εκτελεί αφαίρεση. Δέχεται ως είσοδο τον μειωτέο και τον αφαιρετέο και τυπώνει στην οθόνη τη διαφορά. Στην περίπτωση που ο αφαιρετέος είναι μεγαλύτερος από το μειωτέο, το πρόγραμμα μπαίνει σε **loop** και ζητά νέο αφαιρετέο, μέχρι να δοθεί αφαιρετέος μικρότερος ή ίσος του μειωτέου.

```
program Subtraction;
begin
        num1 := GetNum();   num2 := GetDivider(num1);
        result := Subtract(num1, num2);   DisplayResults
end;
```

```
function GetNum( );
        begin   write("Δώστε το μειωτέο : ");   readln(a);   return a   end;
```

```
function GetDivider(a);
begin
      loop begin
        write("Αφαιρετέος όχι όμως μεγαλύτερος από το μειωτέο : ");   readln(b);
        if  b <= a  then exit
      end;
      return b
end;
```

```
procedure DisplayResults;
        begin  write(“Διαφορά : “,result);  writeln  end.
```

5.5.3 (Πρόγραμμα 1.8.2) Το παρακάτω πρόγραμμα δέχεται ως είσοδο ένα σύμβολο (+ ή –) και δύο αριθμούς και εκτελεί πρόσθεση ή αφαίρεση μεταξύ των αριθμών, ανάλογα με το σύμβολο. Στην περίπτωση της αφαίρεσης, γίνεται έλεγχος, αν ο μειωτέος είναι πράγματι μεγαλύτερος ή ίσος με του αφαιρετέου. Ουσιαστικά, το πρόγραμμα αυτό είναι συνδυασμός των προγραμμάτων 1.8.1 και 1.7.1.

```
program AdditionSubtraction;
begin
        write(“Αυτό το πρόγραμμα εκτελεί πρόσθεση ή αφαίρεση.”);  writeln;
        symbol := InputSymbol( );  num1 := GetNum( );
        if symbol=2 then  num2 := GetDivider(num1)  else  num2 := GetNum( );
        if symbol=1 then  result := Add(num1, num2)
                    else result := Subtract(num1, num2);
        DisplayResults
end.
```

```
function InputSymbol( );
begin
    loop begin
            write(“Δώστε 1 για + ή 2 για – : “);  readln(symbol);
            if  symbol = 1  then exit;  if  symbol = 2  then exit
    end;
    return symbol
end;
```

Κεφάλαιο 5.6 – Δυνάμεις και Επιμεριστική Ιδιότητα με Συναρτήσεις

5.6.1 (Ιστορικό Σημείωμα - «Μια Μικρή Ιδιοφυΐα - Gauss») Το παρακάτω πρόγραμμα υπολογίζει το άθροισμα διαδοχικών ακέραιων αριθμών με τη μέθοδο του Gauss. Δέχεται ως είσοδο δύο αριθμούς, το μεγαλύτερο και το μικρότερο αριθμό που συμμετέχουν στο άθροισμα και εξασφαλίζει ότι ο πρώτος από αυτούς είναι μικρότερος ή ίσος με το δεύτερο.

```
program Gauss;
begin
        write("Άθροισμα διαδοχικών ακεραίων."); writeln;
        num1 := GetNum1( );   num2 := GetNum2(num1);
        result := Sum(num1, num2);   DisplayResults
end;
```

```
function GetNum1( );
        begin   write("Δώστε τον πρώτο αριθμό : ");   readln(a);   return a   end;
```

```
function GetNum2(a);
begin
    loop begin
        write("Δώστε τον τελευταίο (και μεγαλύτερο) αριθμό : ");    readln(b);
        if  b >= a  then exit
    end;
    return b
end;
```

```
function Sum(a, b) = (b + a) * (b – a + 1) / 2;
```

```
procedure DisplayResults;
        begin   write ("Αποτέλεσμα :", result);   writeln   end.
```

5.6.2 (Πρόγραμμα 1.11.1) Το παρακάτω πρόγραμμα δέχεται έναν αριθμό και τυπώνει το τετράγωνό του.

```
program Square;
begin   num := InputNumber( );   result := SquareN(num);   DisplayResults   end;
```

```
function InputNumber( );
        begin   write("Δώστε αριθμό : ");   readln(a);   return a   end;
```

```
function SquareN(n) = n * n.
```

5.6.3 (Πρόγραμμα 1.11.2) Το πρόγραμμα που ακολουθεί δέχεται δύο αριθμούς. Ο πρώτος αποτελεί τη βάση και ο δεύτερος τον εκθέτη. Το πρόγραμμα υπολογίζει το αποτέλεσμα της ύψωσης της βάσης στον εκθέτη.

```
program Power;
begin
      base := InputBase( );   exponent := InputExponent( );
      result := ComputePower(base, exponent);
      write(base," ^ ",exponent," = ",result);   writeln
end;
```

```
function InputBase( );
        begin   write("Δώστε τη βάση : ");   readln(b);   return b   end;
```

```
function InputExponent( );
        begin   write("Δώστε τον εκθέτη : ");   readln(e);   return e  end;
```

```
function ComputePower(b, e);
begin
        res := 1;
        for i:=1 to e do  res := res * b;
        return res
end.
```

5.6.4 (Πρόγραμμα 1.12.1) Το παρακάτω πρόγραμμα δέχεται τρεις αριθμούς a, b και c, και επαληθεύει ότι πράγματι ισχύει η επιμεριστική ιδιότητα, δηλαδή: a (b + c) = ab + ac

```
program Distributive;
begin
        a := InputNumber( );   b := InputNumber( );   c := InputNumber( );
        res1 := ComputeResult1(a, b, c);   res2 := ComputeResult2(a, b, c);
        write(a,” (“,b,” + “,c,”) = “,res1);   writeln;
        write(a,” * “,b,” + “,a,” * “,c,” = “,res2);   writeln;
        if  res1 = res2  then write(“Ισχύει η επιμεριστική ιδιότητα.”)
                         else write(“Δεν ισχύει η επιμεριστική ιδιότητα.”);
        writeln
end;
```

```
function ComputeResult1(x, y, z) = x * ( y + z );
```

```
function ComputeResult2(x, y, z) = ( x * y ) + ( x * z ).
```

Κεφάλαιο 6

Συναρτήσεις και Μετρήσεις Μεγεθών

Κεφάλαιο 6.1 – Μήκη με Συναρτήσεις

6.1.1 (Πρόγραμμα 2.5.1) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται το μήκος ενός ευθύγραμμου τμήματος σε μέτρα και να το μετατρέπει σε εκατοστόμετρα. Το πρόγραμμα σταματάει όταν δοθεί μήκος τμήματος ίσο με το μηδέν.

```
program LengthUnit;
begin
        TitleMessage;
        loop begin
                length := InputLength( );   if  length = 0  then exit
                cmlength := ComputeCmLength(length);
                write(“Μήκος σε εκατοστόμετρα (cm) : “, cmlength);   writeln
        end
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα μετατρέπει μήκη”);   writeln;
        write(“μετρημένα σε μέτρα (m), σε μήκη”);   writeln;
        write(“μετρημένα σε εκατοστόμετρα (cm).”);   writeln
end;
```

```
function InputLength( );
        begin   write(“Δώστε μήκος σε μέτρα (m) : “);   readln(l);   return l   end;
```

```
function ComputeCmLength(l) = l * 100.
```

6.1.2 (Πρόγραμμα 2.5.2) Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο δέχεται το μήκος ενός ευθύγραμμου τμήματος, τη μονάδα μέτρησης του και τη μονάδα στην οποία ζητείται να γίνει η μετατροπή. Το πρόγραμμα κάνει τη μετατροπή και εμφανίζει το αποτέλεσμα στην οθόνη. Η επανάληψη τελειώνει και το πρόγραμμα σταματάει όταν δοθεί μήκος τμήματος ίσο με το μηδέν.

```
program LengthTransformer;
begin
        TitleMessage;
        loop begin
                length := GetLength( );   if  length = 0  then exit;
                unitA := GetUnit( );   unitB := GetUnit( );
                result := Transform(length, unitA, unitB, 10);
                DisplayResults
        end
end;
```

```
procedure TitleMessage;
begin
        write("Αυτό το πρόγραμμα κάνει μετατροπές μονάδων μήκους");   writeln
end;
```

```
function GetLength( );
        begin   write("Δώστε μήκος : ");   readln(l);   return l   end;
```

```
function GetUnit( );
begin
        write("7 για χιλιομετρα (km), 6 για εκατόμετρα (hm), ");   writeln;
        write("5 για δεκάμετρα (km), 4 για μέτρα (m), ");   writeln;
        write("3 για δεκατόμετρα (dm), 2 για εκατοστό-");   writeln;
        write("μετρα (cm), 1 για χιλιοστόμετρα (dm) : ");   writeln;
        loop begin
                write("Δώστε μονάδα μέτρησης (1 έως 7) : ");   readln(u);
                if u>0 then if u<8 then exit
        end;
        return u
end;
```

```
function Transform(l, uA, uB, c);
begin
        if  uA > uB  then begin                          (*Μετατροπή σε υποδιαίρεση*)
                steps := uA – uB;
                for steps times l := l * c
        end
        else begin                                       (*Μετατροπή σε πολλαπλάσιο*)
                steps := uB – uA;
                for steps times l := l / c
        end;
        return l
end;
```

```
procedure DisplayResults;
        begin   write(“Αποτέλεσμα : “,result);   writeln   end.
```

6.1.3 (Πρόγραμμα 2.7.1) Να γραφτεί, με χρήση επανάληψης και διαδικασιών, πρόγραμμα το οποίο δέχεται το μήκος ενός ευθύγραμμου τμήματος σε μορφή συμμιγούς αριθμού και το μετατρέπει σε δεκαδική παράσταση. Το πρόγραμμα σταματάει όταν δοθεί μήκος τμήματος ίσο με το μηδέν.

```
program LengthDec;
begin
        TitleMessage;
        loop begin
                InsertData;
                result := ComputeResult(km, hm, dam, m, dm, cm, mm);
                if  result = 0  then exit;   DisplayResult
        end
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα μετατρέπει μήκη σε μορφή”);   writeln;
        write(“συμμιγούς αριθμού, σε μήκη δεκαδικής παράστασης”); writeln
end;
```

```
procedure InsertData;
begin
        write(“Δώστε πλήθος χιλιόμετρων (km) : “);   readln(km);
        write(“Δώστε πλήθος εκατόμετρων (hm) : “);   readln(hm);
        write(“Δώστε πλήθος δεκάμετρων (dam) : “);   readln(dam);
        write(“Δώστε πλήθος μέτρων (m) : “);   readln(m);
        write(“Δώστε πλήθος δεκατόμετρων (dm) : “);   readln(dm);
        write(“Δώστε πλήθος εκατοστόμετρων (cm) : “);   readln(cm);
        write(“Δώστε πλήθος χιλιοστόμετρων (mm) : “);   readln(mm)
end;
```

```
function ComputeResult(km, hm, dam, m, dm, cm, mm) =
                        1000*km+100*hm+10*dam+m+0.1*dm+0.01*cm+0.001*mm;
```

```
procedure DisplayResult;
        begin   write(“Συνολικό μήκος : “,result);   writeln   end.
```

Κεφάλαιο 6.2 – Εμβαδά με Συναρτήσεις

6.2.1 (Πρόγραμμα 2.10.2) Να γραφτεί με χρήση επανάληψης και συναρτήσεων ένα πρόγραμμα το οποίο να δέχεται το εμβαδό μιας επιφάνειας, τη μονάδα μέτρησης του και τη μονάδα στην οποία ζητείται να γίνει η μετατροπή. Το πρόγραμμα κάνει τη μετατροπή και εμφανίζει το αποτέλεσμα στην οθόνη. Η επανάληψη τελειώνει και το πρόγραμμα σταματάει όταν δοθεί εμβαδό επιφάνειας ίσο με το μηδέν.

```
program AreaTransformer;
begin
        TitleMessage;
        loop begin
                area := GetArea( );   if  area = 0  then exit;
                unitA := GetUnit( );   unitB := GetUnit( );
                result := Transform(area, unitA, unitB, 100);
                DisplayResults
        end
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα κάνει μετατροπές μονάδων εμβαδού”); writeln
end;
```

```
function GetArea( );
        begin   write(“Δώστε εμβαδό : “);   readln(a);   return a   end;
```

```
function GetUnit( );
begin
        write(“7 για τ. χιλιομετρα (km^2), 6 για τ. εκατόμετρα (hm^2), “);
        writeln;
        write(“5 για τ. δεκάμετρα (km^2), 4 για τ. μέτρα (m^2), “);   writeln;
        write(“3 για τ. δεκατόμετρα (dm^2), 2 για τ. εκατοστό-”);   writeln;
        write(“μετρα (cm^2), 1 για τ. χιλιοστόμετρα (dm^2) : “);   writeln;
        loop begin
                write(“Δώστε μονάδα μέτρησης (1 έως 7) : “);   readln(u);
                if u>0 then if u<8 then exit
        end;
        return u
end.
```

6.2.2 (Πρόγραμμα 2.11.1) Να γραφτεί με χρήση επανάληψης, διαδικασιών και συναρτήσεων ένα πρόγραμμα το οποίο να δέχεται τα μήκη των πλευρών ορθογώνιου παραλληλόγραμμου και να υπολογίζει την περίμετρο και το εμβαδό του. Το πρόγραμμα σταματάει όταν δοθούν μηδενικές και πλευρές ορθογώνιου παραλληλόγραμμου.

```
program AreaAB;
begin
        TitleMessage;
        loop begin
                side1 := GetSide(1);   side2 := GetSide(2);
                if  side1 = 0  then if  side2 = 0  then exit;
                per := ComputePerimeter(side1, side2);
                area := ComputeArea(side1, side2);
                DisplayResults;
        end
end;
```

```
procedure TitleMessage;
begin
        write("Αυτό το πρόγραμμα υπολογίζει το εμβαδό");   writeln;
        write("ορθογωνίου παραλληλογράμμου");   writeln
end;
```

```
function GetSide(num);
        begin   write("Δώστε μήκος πλευράς ",num," : ");   readln(s)   return s   end;
```

```
function ComputePerimeter(a, b) = 2 * a + 2 * b;
```

```
function ComputeArea(a, b) = a * b;
```

```
procedure DisplayResults;
begin
        write("Περίμετρος ορθογώνιου παραλληλόγραμμου : ", per); writeln;
        write("Εμβαδό ορθογώνιου παραλληλόγραμμου : ", area); writeln
end.
```

6.2.3 (Πρόγραμμα 2.11.2) Να γραφτεί με χρήση επανάληψης, διαδικασιών και συναρτήσεων ένα πρόγραμμα το οποίο να δέχεται το μήκος πλευράς τετραγώνου και να υπολογίζει την περίμετρο και το εμβαδό του. Το πρόγραμμα σταματάει όταν δοθεί μηδενικό μήκος πλευράς τετραγώνου.

```
program AreaAA;
begin
        TitleMessage;
        loop begin
                side1 := GetSide(1);   if  side1 = 0  then exit;
                per := ComputePerimeter(side1, side1);
                area := ComputeArea(side1, side1);   DisplayResults;
        end
end.
```

Κεφάλαιο 6.3 – Όγκοι με Συναρτήσεις

6.3.1 (Πρόγραμμα 2.13.1) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, ένα πρόγραμμα το οποίο να δέχεται τον όγκο ενός στερεού σε κυβικά μέτρα και να τον μετατρέπουν σε κυβικά εκατοστόμετρα. Το πρόγραμμα σταματάει όταν δοθεί όγκος στερεού ίσος με το μηδέν.

```
program VolumeUnit;
begin
        TitleMessage;
        loop begin
                mvolume := VolumeInput( );
                if  mvolume = 0  then exit;
                cmvolume := VolumeTransform(mvolume);
                DisplayResult
        end
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα μετατρέπει όγκους”);   writeln;
        write(“μετρημένους σε κυβικά μέτρα (m^3), σε όγκους”);   writeln;
        write(“μετρημένους σε κυβικά εκατοστόμετρα (cm^3).”);   writeln
end;
```

```
function VolumeInput( );
begin
        write(“Δώστε όγκο σε κυβικά μέτρα (m^3) : “);   readln(vol);   return vol
end;
```

```
function VolumeTransform(vol) = 1000000 * vol;
```

```
procedure DisplayResult;
begin
        write(“Όγκος σε κυβικά εκατοστόμετρα (cm^3): “,cmvolume);   writeln
end.
```

6.3.2 (Πρόγραμμα 2.13.2) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, ένα πρόγραμμα το οποίο να δέχεται τον όγκο ενός στερεού, τη μονάδα μέτρησης του και τη μονάδα στην οποία ζητείται να γίνει η μετατροπή. Το πρόγραμμα κάνει τη μετατροπή και εμφανίζει το αποτέλεσμα στην οθόνη. Η επανάληψη τελειώνει και το πρόγραμμα σταματάει όταν δοθεί όγκος στερεού ίσος με το μηδέν.

```
program VolumeTransformer;
begin
        TitleMessage;
        loop begin
                volume := GetVolume( );   if  volume = 0  then exit;
                unitA := GetUnit( );   unitB := GetUnit( );
                result := Transform(volume, unitA, unitB, 1000);
                DisplayResults;
        end
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα κάνει μετατροπές μονάδων όγκου”); writeln
end;
```

```
function GetVolume( );
        begin   write("Δώστε όγκο : ");   readln(vol);   return vol   end
```

```
function GetUnit( );
begin
        write("7 για κ. χιλιομετρα (km^3), 6 για κ. εκατόμετρα (hm^3), ");
        writeln;
        write("5 για κ. δεκάμετρα (km^3), 4 για κ. μέτρα (m^3), ");   writeln;
        write("3 για κ. δεκατόμετρα (dm^3), 2 για κ. εκατοστό-");   writeln;
        write("μετρα (cm^3), 1 για κ. χιλιοστόμετρα (dm^3) : ");   writeln;
        loop begin
                write("Δώστε μονάδα μέτρησης (1 έως 7) : ");   readln(u);
                if u>0 then if u<8 then exit
        end
end.
```

6.3.3 (Πρόγραμμα 2.14.1) Να γραφτεί με χρήση με χρήση επανάληψης, διαδικασιών και συναρτήσεων ένα πρόγραμμα, το οποίο να δέχεται τα μήκη των πλευρών ορθογώνιου παραλληλεπιπέδου και να υπολογίζει τα εξής:

- τον όγκο του
- το εμβαδό των βάσεων
- το εμβαδό της παράπλευρης επιφάνειας
- το συνολικό μήκος των ακμών του.

Το πρόγραμμα σταματάει όταν δοθεί μηδενικές και οι τρεις διαστάσεις του ορθογώνιου παραλληλεπιπέδου.

```
program VolumeABC;
begin
        TitleMessage;
        loop begin
                sideA := GetSide(1);   sideB := GetSide(2);   sideC := GetSide(3);
                if  sideA = 0  then if  sideB = 0  then if  sideC = 0  then exit;
                volume:= ComputeVolume(sideA, sideB, sideC);
                basesarea := ComputeBasesArea(sideA, sideB, sideC);
                sidearea := ComputeSideArea(sideA, sideB, sideC);
                edgessum := ComputeEdgesSum(sideA, sideB, sideC);
                DisplayResults
        end
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα υπολογίζει τον όγκο, το”);   writeln;
        write(“εμβαδό των βάσεων και της παράπλευρης”);   writeln;
        write(“επιφάνειας και το συνολικό μήκος των ακμών”);   writeln;
        write(“ ορθογώνιου παραλληλεπιπέδου.”);   writeln
end;
```

```
function GetSide(n);
begin   write(“Δώστε διάσταση “,n,”ορθογωνίου : “);   readln(a)   return a   end;
```

```
function ComputeVolume(a, b, c) = a * b * c;
```

```
function ComputeBasesArea(a, b, c) = 2 * a * b;
```

```
function ComputeSideArea(a, b, c) = 2 * c * (a + b);
```

```
function ComputeEdgesSum(a, b, c) = 4 (a + b + c);
```

```
procedure DisplayResults;
begin
        write(“Όγκος ορθογώνιου : “,volume); writeln;
        write(“Εμβαδό βάσεων : “,basearea); writeln;
        write(“Εμβαδό παράπλευρης επιφάνειας : “,sidearea); writeln;
        write(“Συνολικό μήκος ακμών : “,edgesum); writeln
end.
```

6.3.4 (Πρόγραμμα 2.14.2) Να γραφτεί με χρήση επανάληψης, διαδικασιών και συναρτήσεων ένα πρόγραμμα το οποίο να δέχεται τα μήκη των πλευρών ορθογώνιου παραλληλεπιπέδου και να υπολογίζει τα εξής:

- τον όγκο του
- το εμβαδό των βάσεων
- το εμβαδό της παράπλευρης επιφάνειας
- το συνολικό μήκος των ακμών του.

Το πρόγραμμα σταματάει όταν δοθεί μηδενικές και οι τρεις διαστάσεις του κύβου.

```
program VolumeAAA;
begin
        TitleMessage;
        loop begin
                sideA := GetSide(1);
                if  sideA = 0  then exit;
                volume:= ComputeVolume(sideA, sideA, sideA);
                basesarea := ComputeBasesArea(sideA, sideA, sideA);
                sidearea := ComputeSideArea(sideA, sideA, sideA);
                edgessum := ComputeEdgesSum(sideA, sideA, sideA);
                DisplayResults
        end
end.
```

6.3.5 (Πρόγραμμα 2.15.1) Να γραφτεί, με χρήση επανάληψης και διαδικασιών, ένα πρόγραμμα το οποίο να δέχεται τη διάρκεια ενός χρονικού διαστήματος σε δευτερόλεπτα και να τη μετατρέπει σε συμμιγή αριθμό, δηλαδή υπολογίζει πόσες ώρες, πόσα λεπτά και πόσα δευτερόλεπτα περιέχει η διάρκεια αυτή. Το πρόγραμμα σταματάει όταν δοθεί χρονικό διάστημα ίσο με το μηδέν.

```
program VolumeUnit;
begin
        TitleMessage;
        loop begin
                seconds := TimeInput( );   if  seconds = 0  then exit;
                hours := ComputeDiv(seconds, 3600);
                seconds:= ComputeMod(seconds, 3600);
                minutes:= ComputeDiv(seconds, 60);
                seconds:= ComputeMod(seconds, 60);
                DisplayResults
        end
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα μετατρέπει δευτερόλεπτα”);   writeln;
        write(“σε ώρες, λεπτά και δευτερόλεπτα.”);   writeln
end;
```

```
function TimeInput( );
begin
        write(“Δώστε χρονική διάρκεια σε δευτερόλεπτα : “);
        readln(sec);   return sec
end;
```

```
function ComputeDiv(sec, c) = sec div c;
```

```
function ComputeMod(sec, c) = sec mod c;
```

```
procedure DisplayResults;
begin
        write(hours,” ώρες, “,minutes,” λεπτά, “,seconds,” δευτερόλεπτα);   writeln
end.
```

Ασκήσεις.

1. Να γραφτεί το πρόγραμμα με χρήση επανάληψης, διαδικασιών και συναρτήσεων το οποίο δέχεται το μήκος, το πλάτος και το ύψος ενός ορθογώνιου παραλληλόγραμμου μετρημένα σε διάφορες μονάδες μέτρησης (μέτρα, δεκατόμετρα, εκατοστόμετρα ή χιλιοστόμετρα). Μετατρέπει τις μονάδες μέτρησης σε μετρά και υπολογίζει τον όγκο του ορθογώνιου παραλληλόγραμμου. Το πρόγραμμα σταματάει όταν δοθεί μηδενικό μήκος, πλάτος και ύψος.

2. Να γραφτεί πρόγραμμα, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, το οποίο δέχεται έναν όγκο μετρημένο σε λίτρα (l), κυβικά μέτρα (m^3) ή κυβικά πόδια (ft^3) και εκτελεί όλες τις δυνατές μετατροπές μεταξύ των παραπάνω μονάδων.

3. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, ένα πρόγραμμα το οποίο να δέχεται τη διάρκεια ενός χρονικού διαστήματος σε δευτερόλεπτα, λεπτά ή ώρες και κάνει μετατροπή σε οποιαδήποτε από τις παραπάνω χρονικές μονάδες μέτρησης. Το πρόγραμμα σταματάει όταν δοθεί χρονικό διάστημα ίσο με το μηδέν.

Κεφάλαιο 6.4 – Μονάδες Μάζας και νομισματικές Μονάδες με Συναρτήσεις

6.4.1 (Πρόγραμμα 2.16.1) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, ένα πρόγραμμα το οποίο να δέχεται τη μάζα ενός σώματος σε τόνους, κιλά ή γραμμάρια και κάνει μετατροπή σε οποιαδήποτε από τις παραπάνω μονάδες μέτρησης μάζας. Το πρόγραμμα σταματάει όταν δοθεί χρονικό διάστημα ίσο με το μηδέν.

```
program MassTransformer;
begin
        TitleMessage;
        loop begin
                mass := GetMass( );   if  mass = 0  then exit;
                unitA := GetUnit( );   unitB := GetUnit( );
                result := Transform(mass, unitA, unitB, 1000);
                DisplayResult
        end
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα κάνει μετατροπές μονάδων μάζας.”);   writeln
end;
```

```
function GetMass( );
        begin   write(“Δώστε μάζα : “);   readln(m);   return m   end;
```

```
function GetUnit( );
begin
        write(“3 για τόνους, 2 για κιλά, 1 για γραμμάρια.”);   writeln;
        loop begin
                write(“Δώστε μονάδα μέτρησης (1 έως 3) : “);   readln(u);
                if u>0 then if u<4 then exit
        end;
        return u
end.
```

6.4.2 (Πρόγραμμα 2.17.1) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, ένα πρόγραμμα το οποίο να κάνει όλες τις δυνατές μετατροπές μεταξύ των παρακάτω νομισματικών μονάδων: Ευρώ, Δολάριο Η.Π.Α., Λίρα Αγγλίας και Γιέν Ιάπωνιας. Το πρόγραμμα σταματάει όταν δοθεί ποσό ίσο με το μηδέν. Το πρόγραμμα χρησιμοποιεί τον πίνακα ισοτιμιών και την τεχνική μετατροπής όλων των νομισμάτων σε Ευρώ.

```
program CurrencyTransformer;
begin
        TitleMessage;
        loop begin
                amount := GetAmount( );   if  amount = 0  then exit;
                unitA := GetUnit( );   unitB := GetUnit( );
                result := Transform(amount, unitA, unitB);   DisplayResult
        end
end;
```

```
procedure TitleMessage;
begin
        write(“Αυτό το πρόγραμμα κάνει όλες τις δυνατές μετατροπές”);  writeln;
        write(“μεταξύ των παρακάτω νομισματικών μονάδων: Ευρώ,”);  writeln;
        write(“Δολάριο Η.Π.Α., Λίρα Αγγλίας και Γιέν Ιάπωνιας.”); writeln
end;
```

```
function GetAmount( );
        begin   write(“Δώστε χρηματικό ποσό : “);   readln(a);   return a   end;
```

```
function GetUnit( );
begin
        write(“1 για Ευρώ, 2 για Δολάρια Η.Π.Α.,”);   writeln;
        write(“3 για Λίρα Αγγλίας, 4 για Γιέν Ιάπωνας (σε εκατοντάδες).”);   writeln;
        loop begin
                write(“Δώστε νόμισμα (1 έως 4) : “);   readln(u);
                if u>0 then if u<5 then exit
        end;
        return u
end;
```

```
function Transform(amount, uA, uB);
begin
        if  uA=2 then begin amount := amount / 0.9053;
        if  uA=3 then begin amount := amount / 0.6272;
        if  uA=4 then begin amount := amount / 119,587;
        (*Σε αυτό το σημείο, οποιοδήποτε ποσό έχει μετατραπεί σε Ευρώ*)
        if  uB=2 then begin amount := amount * 0.8812;
        if  uB=3 then begin amount := amount * 0.6105;
        if  uB=4 then begin amount := amount * 116.406;
        return amount
end.
```

Ασκήσεις.

1. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, ένα πρόγραμμα το οποίο να δέχεται τη μάζα ενός σώματος σε γραμμάρια και να τη μετατρέπει σε συμμιγή αριθμό, δηλαδή υπολογίζει πόσους τόνους, πόσα κιλά και πόσα γραμμάρια περιέχει η μάζα αυτή. Το πρόγραμμα σταματάει όταν δοθεί μάζα ίση με το μηδέν.

2. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, ένα πρόγραμμα το οποίο να δέχεται ένα χρηματικό ποσό σε δραχμές και να το μετατρέπει σε ευρώ. Το πρόγραμμα σταματάει όταν δοθεί ποσό ίσο με το μηδέν.

3. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, ένα πρόγραμμα το οποίο να δέχεται ένα χρηματικό ποσό σε ευρώ και να το μετατρέπει σε δραχμές. Το πρόγραμμα σταματάει όταν δοθεί ποσό ίσο με το μηδέν.

4. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, ένα πρόγραμμα το οποίο να δέχεται ένα ποσό σε ευρώ και να υπολογίζει τα χαρτονομίσματα και τα κέρματα που το αποτελούν. Το πρόγραμμα σταματάει όταν δοθεί ποσό ίσο με το μηδέν.

Κεφάλαιο 7

Συναρτήσεις, Κλάσματα και Ανάλογα Ποσά

Κεφάλαιο 7.1 – Ομώνυμα και Ετερόνομα Κλάσματα με Συναρτήσεις

7.1.1 (Πρόγραμμα 3.1.1) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται έναν ακέραιο αριθμό a και ένα κλάσμα $\frac{b}{c}$ και να υπολογίζει έναν ακέραιο αριθμό d, τέτοιο ώστε να είναι ίσος με το μέρος $\frac{b}{c}$ του αριθμού a. Για παράδειγμα, αν a=150 και $\frac{b}{c}=\frac{1}{3}$ τότε d=50, διότι το $\frac{1}{3}$ του αριθμού 150 ισούται με 50. Αν το αποτέλεσμα δεν είναι ακέραιος αριθμός τότε το πρόγραμμα τον εμφανίζει στην οθόνη στρογυλοποιημένο στη μονάδα. Το πρόγραμμα σταματάει όταν δοθούν a=0, b=0 και c=1.

```
program QuantityAndRatio;
begin
        loop begin
                quantity := QuantityEntry( );
                num := NumeratorEntry( );   den := DenominatorEntry( );
                if  quantity = 0  then if  num = 0  then if  den = 1  exit;
                result := QuantityAndRatioComputations(quantity, num, den);
                QuantityAndRatioOutput
        end
end;
```

```
function QuantityEntry( );
begin
        write("Δώστε την ποσότητα");   readln(a);   return a
end;
```

```
function NumeratorEntry( );
begin
        write("Δώσε τον αριθμητή του κλάσματος");   readln(b);   return b
end;
```

```
function DenominatorEntry( );
begin
        loop begin
                write("Δώσε τον παρονομαστή του κλάσματος"); readln(c);
                if  c<>0  then exit;
        end;
        return c
end;
```

```
function QuantityAndRatioComputations(a, b, c) = a * (b / c);
```

```
procedure QuantityAndRatioOutput;
begin
        write(num,"/",den," του ",quantity," είναι ",result);   writeln
end.
```

7.1.2 (Πρόγραμμα 3.1.2) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται δύο κλάσματα, να εξετάζει αν είναι ομώνυμα ή ετερώνυμα και να εμφανίζει το αντίστοιχο μήνυμα στην οθόνη. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program SameDenominator;
begin
        loop begin
                num1 := NumeratorEntry( );   den1 := DenominatorEntry( );
                num2 := NumeratorEntry( );   den2 := DenominatorEntry( );
                if num1=0 then if num2=0 then if den1=1 then if den2=1 exit;
                SameDenominatorOutput;
        end
end;
```

```
procedure SameDenominatorOutput;
begin
        if  den1 <> den2  then write("Είναι ετερώνυμα")
        else write("Είναι ομώνυμα")
end.
```

7.1.3 (Πρόγραμμα 3.1.4) Να γραφτεί, με χρήση συναρτήσεων, πρόγραμμα το οποίο επιλύει προβλήματα με τη μέθοδο της **αναγωγής στη μονάδα**. Δέχεται ως είσοδο ένα κλάσμα $\frac{a}{b}$ και έναν αριθμό c. Οι δύο αυτές είσοδοι αποτελούν τα δεδομένα της μεθόδου, δηλαδή ισχύει ότι το μέρος $\frac{a}{b}$ μιας ποσότητας αντιστοιχεί στον αριθμό c. Για παράδειγμα τα $\frac{3}{5}$ ενός τραπεζικού λογαριασμού είναι 450 Ευρώ. Επίσης, το πρόγραμμα δέχεται ένα ακόμη κλάσμα $\frac{d}{b}$, ομώνυμο με το κλάσμα $\frac{a}{b}$, το οποίο αντιπροσωπεύει το μέρος της ποσότητας που ζητείται να υπολογιστεί. Για παράδειγμα, ζητούνται να υπολογιστούν τα $\frac{4}{5}$ του τραπεζικού λογαριασμού. Το πρόγραμμα, αν δοθούν οι είσοδοι του παραπάνω παραδείγματος, εμφανίζει στην οθόνη:

3/5 αντιστοιχούν σε 150
1/5 αντιστοιχεί σε 50
4/5 αντιστοιχούν σε 200

```
program UnitReduction;
begin
        num1 := NumeratorEntry( );   den1 := DenominatorEntry( );
        quantity := QuantityEntry( );   num2 := NumeratorEntry( );
        result := ReductionComputationsOutput(num1, den1, quantity, num2)
end;
```

```
function ReductionComputationsOutput(a, b, c, d) ;
begin
        writeln;   write(a,"/",b," αντιστοιχούν σε ",c);   writeln;
        unit := c/a;
        write(1,"/",b," αντιστοιχούν σε ",unit);   writeln;
        res := d * unit;
        write(d,"/",b," αντιστοιχούν σε ",res);   writeln;
        return res
end.
```

7.1.4 (Πρόγραμμα 3.1.5) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται δύο ποσότητες, quantity1 και quantity2, και να υπολογίζει τι μέρος της πρώτης είναι η δεύτερη ποσότητα. Παράδειγμα:

quantity1=20, quantity 2=5, result=0.25

Το πρόγραμμα σταματάει όταν η πρώτη ποσότητα δοθεί ίση με το μηδέν.

```
program Quantities;
begin
        loop begin
                quantity1 := QuantityEntry( );
                if  quantity1=0  then exit;
                quantity2 := QuantityEntry( );
                result := QuantityComputationsOutput(quantity1, quantity2)
        end
end;
```

```
function QuantityComputationsOutput(q1, q2) ;
begin
        res := q2 / q1;
        write("το ", q2," είναι το μέρος ",res," του ",q1);   writeln;
        return res
end.
```

Ασκήσεις

1. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα που να δέχεται ένα μήκος Α σε cm και ένα μήκος Β σε mm και να υπολογίζει τι μέρος του Α είναι το Β.

2. Να γραφτεί το ίδιο πρόγραμμα για:
   - m και cm
   - kg και γραμμάρια
   - cm και dm

3. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα που να δέχεται σαν είσοδο το πλήθος των τμημάτων στα οποία χωρίζεται η συνολική ποσότητα και την αξία κάθε τμήματος και να υπολογίζει τη συνολική ποσότητα.

4. Να γραφτεί το πρόγραμμα 3.1.3 με χρήση συναρτήσεων.

5. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα που να δέχεται την αρχική τιμή ενός προϊόντος και το ποσό της έκπτωσης και να εμφανίζει τι μέρος της αρχικής τιμής είναι η έκπτωση και τι μέρος της αρχικής τιμής πληρώσαμε τελικά.

6. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα που να δέχεται την αρχική τιμή ενός προϊόντος και το μέρος της έκπτωσης και να υπολογίζει την τελική τιμή που θα πληρώσουμε.

7. Να γραφεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα που να δέχεται την τιμή των ¾ του κιλού και να εμφανίζει την τιμή των ⅝ του κιλού.

Κεφάλαιο 7.2 – Κλάσματα και Ευκλείδεια Διαίρεση με Συναρτήσεις

7.2.1 (Πρόγραμμα 3.2.1) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται δύο ακέραιους αριθμούς, διαιρετέο και διαιρέτη, και να εμφανίζει στην οθόνη το κλάσμα με το οποίο ισοδυναμεί η διαίρεση αυτή (σε μορφή $\frac{a}{b}$). Επίσης, να υπολογίζει το πηλίκο της διαίρεσης που ορίζεται από το κλάσμα. Το πρόγραμμα σταματάει όταν δοθεί αριθμητής ίσος με το μηδέν και παρονομαστής ίσος με τη μονάδα.

```
program IntegerDivision;
begin
        loop begin
                num := NumEntry( );   divider := DividerEntry( );
                if num=0 then if divider=1 then exit;
                quotient := DivisionComputationsOutput(num, divider)
        end
end;
```

```
function NumEntry( );
begin
        write("Δώσε τον διαιρετέο");   readln(num);   return num
end;
```

```
function DividerEntry( );
begin
        loop begin
                write("Δώσε τον διαιρέτη");   readln(div);
                if div<>0 then exit
        end;
        return div
end;
```

```
function DivisionComputationsOutput(n, div);
begin
        write(n);   writeln;
        write("-----");   writeln;
        write(div);   writeln;   writeln;
        qent := n / div;
        write("Πηλίκο διαίρεσης : ", qent);   writeln;
        return qent
end.
```

7.2.2 (Πρόγραμμα 3.2.2) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται τον αριθμητή και τον παρονομαστή ενός κλάσματος και να ελέγχει αν το κλάσμα είναι ίσο με 0 ή 1.

```
program CheckZeroOne;
begin
        loop begin
                num := NumeratorEntry( );   den := DenominatorEntry( );
                ZeroOneOutput;
                if  AskUserToContinue( )  then exit
        end
end;
```

```
procedure ZeroOneOutput;
begin
        if  num = 0  then write ("Το κλάσμα ισούται με 0.")
                     else if  num = den  then write("Το κλάσμα ισούται με 1.")
                                         else write("Το κλάσμα δεν ισούται με 0 ή 1.");
        writeln
end;
```

```
function AskUserToContinue( );
begin
        loop begin
                write ("Δώστε 0 για να επαναλάβετε, 1 για τέλος : ");   readln(f);
                if  f = 0  then exit;   if  f = 1  then exit;
        end;
        return f
end.
```

7.2.3 (Πρόγραμμα 3.2.3) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται τον αριθμητή και τον παρονομαστή ενός κλάσματος και να ελέγχει αν το πηλίκο της διαίρεσης που αυτό ορίζει είναι ακέραιος αριθμός.

```
program IntegerCheck;
begin
        loop begin
                num := NumeratorEntry( );   den := DenominatorEntry( );
                remainder := IntegerCheckComputation(num, den);
                quotient := IntegerCheckOutput(num, den, remainder);
                if  AskUserToContinue( )  then exit
        end
end;
```

```
function IntegerCheckComputation(n, d)  = n mod d;
```

```
function IntegerCheckOutput(n, d, rem) ;
begin
        qent := n div d;
        if  rem = 0  then   qent write("Το κλάσμα ισούται με ", qent)
                        else write("Το κλάσμα δεν ισούται με ακέραιο αριθμό.");
        writeln;
        return qent
end.
```

7.2.4 (Πρόγραμμα 3.2.4) Να γραφτεί πρόγραμμα, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, το οποίο να επιλύει την εξίσωση $\frac{x-a}{b} = 0$, όπου τα a, b δίνονται από το χρήστη.

```
program SolveEq;
begin
        loop begin
                a := AParameterEntry( );   b := BParameterEntry( );
                SolveEqOutput;
                if  AskUserToContinue( )  then exit
        end
end;
```

```
function AParametersEntry( );
begin
        write("Δώστε την παράμετρο a : "); readln(a); return a
end;
```

```
function BParametersEntry( );
begin
        loop begin
                write("Δώστε την παράμετρο b : "); readln(b);
                if b<>0 then exit
        end;
        return b
end;
```

```
procedure SolveEqOutput;
        begin write("x = ",a); writeln end.
```

Ασκήσεις

1. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα που να υπολογίζει σε κιλά το κλάσμα μιας αρχικής ποσότητας (σε κιλά) που θα δίνεται από τον χρήστη.

2. Να λυθούν εξισώσεις της μορφής: $\frac{x+a}{b}=0$,$\frac{x+a}{b}=1$,$\frac{a-x}{b}=1$,$\frac{x+a}{b}=-1$

Κεφάλαιο 7.3 – Ισοδύναμα και Ανάγωγα Κλάσματα με Συναρτήσεις

7.3.1 (Πρόγραμμα 3.3.1) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται ένα κλάσμα $\frac{a}{b}$ και ένα ακέραιο αριθμό n, να κατασκευάζει και να εμφανίζει στην οθόνη n ισοδύναμα κλάσματα με αυτό. Το πρόγραμμα σταματάει όταν δοθούν a=0 και b=1.

```
program EquivalentFractions;
begin
        loop begin
                num := NumeratorEntry( );   den := DenominatorEntry( );
                if  num = 0  then if  den = 1  then exit;
                counter := EquivalentFractionsEntry( );
                i := 2;
                for counter times do
                begin
                        newnum := Multiply(num, i);
                        newden := Multiply(den, i);
                        EquivalentFractionsOutput;   i:=i+1
                end
        end
end;
```

```
function EquivalentFractionsEntry( );
begin
        loop begin
                write("Δώσε αριθμό ισοδύναμων κλασμάτων");   readln(n);
                if n>0 then exit;
        end;
        return n
end;
```

```
procedure EquivalentFractionsOutput;
        begin  write(num," / ",den," = ",newnum," / ",newden);  writeln   end.
```

7.3.2 (Πρόγραμμα 3.3.2) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται δύο κλάσματα, να εξετάζει αν είναι ισοδύναμα και να εμφανίζει το αντίστοιχο μήνυμα στην οθόνη. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program CheckEquivalent;
begin
        loop begin
                num1 := NumeratorEntry( );   den1 := DenominatorEntry( );
                num2 := NumeratorEntry( );   den2 := DenominatorEntry( );
                if num1=0 then if den1=1 then if num2=0 then if den2=1 then exit;
                flag:=CheckEquivalentComputationOutput(num1, num2, den1, den2);
        end
end;
```

```
function CheckEquivalentComputationOutput(n1, d1, n2, d2) ;
begin
        flag:= num1 * den2 – num2 * den1;
        if flag = 0 then   write("Τα κλάσματα είναι ισοδύναμα.")
                        else  write("Τα κλάσματα δεν είναι ισοδύναμα.");
        writeln;
        return flag
end.
```

7.3.3 (Πρόγραμμα 3.3.3) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται ένα κλάσμα $\frac{a}{b}$, εξετάζει αν το κλάσμα είναι ανάγωγο και εμφανίζει στην οθόνη κατάλληλο μήνυμα. Αν όχι, υπολογίζει το ισοδύναμο ανάγωγο κλάσμα. Το πρόγραμμα σταματάει όταν δοθούν a=0 και b=1.

```
program FractionSimplification;
begin
        num := NumeratorEntry( );   den := DenominatorEntry( );
        if  num = 0  then if  den = 1  then exit;
        mkd := FindMkd(num, den);
        if mkd=1 then SimplificationOutput1
        else begin
                newnum := SimplificationComputation(num, mkd);
                newden := SimplificationComputation(den, mkd);
                SimplificationOutput2
        end
end;
```

```
function FindMkd(x, y) ;
begin
        a := x;   b := y;
        loop   begin
                if a=0 then exit;   if b=0 then exit;
                if  a > b  then  a := a mod b  else  b := b mod a
        end;
        mkd := a + b;
        return mkd
end;
```

```
procedure SimplificationOutput1;
        begin   write("Το κλάσμα είναι ανάγωγο.");   writeln   end;
```

```
function SimplificationComputation(x, y)  = x div y;
```

```
procedure SimplificationOutput2;
begin
        write("Το κλάσμα δεν είναι ανάγωγο.");   writeln;
        write(num," / ",den," = ",newnum," / ",newden);   writeln
end.
```

7.3.4 (Πρόγραμμα 3.3.4) Έστω δύο κλάσματα $\frac{x}{a}$ και $\frac{b}{c}$. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται τους αριθμούς a, b και c και υπολογίζει τον αριθμό x, ώστε τα δύο κλάσματα να είναι ισοδύναμα. Το πρόγραμμα σταματάει όταν δοθούν a=0, b=1 και c=0.

```
program SolveEq1;
begin
        loop begin
                den1 := DenominatorEntry( );
                num2 := NumeratorEntry( );   den2 := DenominatorEntry( );
                if  den1 = 1  then if  den2 = 1  then if  num2 = 0  then exit;
                x := SolveEqComputation(den1, num1, den2);   SolveEqOutput
        end
end;
```

```
function SolveEqComputation(den1, num1, den2) ;
begin
        if  den1 = den2  then   result := num2
                         else   result := num1 * num2 / den2;
        return result
end;
```

```
procedure SolveEqOutput;
        begin   write("Η λύση είναι x = ",x);   writeln   end.
```

Ασκήσεις

1. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται δύο ακέραιους αριθμούς a και b, και εμφανίζει στην οθόνη τον αριθμό a γραμμένο υπό μορφή κλάσματος με παρονομαστή ίσο με b. Για παράδειγμα αν a=4 και b=7, τότε το πρόγραμμα εμφανίζει:
   4 = 28 / 7
   Το πρόγραμμα σταματάει όταν δοθούν a=0 και b=1.

2. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται ένα κλάσμα $\frac{a}{b}$ και έναν ακέραιο αριθμό c, και να εμφανίζει στην οθόνη, αν είναι δυνατό, ένα κλάσμα ισοδύναμο με το $\frac{a}{b}$ με παρονομαστή ίσο με c. Αν δεν είναι δυνατό, τότε το πρόγραμμα εμφανίζει στην οθόνη κατάλληλο μήνυμα. Το πρόγραμμα σταματάει όταν δοθούν a=0, b=1 και c=0.

Κεφάλαιο 7.4 – Σύγκριση Ομώνυμων και Ετερώνυμων Κλασμάτων με Συναρτήσεις

7.4.1 (Πρόγραμμα 3.4.1) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται δύο ομώνυμα κλάσματα, να τα συγκρίνει και τα εμφανίζει στην οθόνη με το κατάλληλο σύμβολο ανάμεσά τους (<, > ή =). Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program EquivalentFractionsComparison;
begin
        loop begin
                num1 := NumeratorEntry( );   den := DenominatorEntry( );
                num2 := NumeratorEntry( );
                if num1= 0 then if num2=0 then if den=1 then exit;
                EquivalentComparisonOutput
        end
end;
```

```
procedure EquivalentComparisonOutput;
begin
        if num1= num2 then write(num1," / ",den," = ",num2," / ", den)
                else if num1>num2 then write(ar1," / ",den," > ", ar2," / ",den)
                        else write(num1," / ",den," < ", num2," / ",den);
        writeln
end.
```

7.4.2 (Πρόγραμμα 3.4.2) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται δύο κλάσματα με τον ίδιο αριθμητή, να τα συγκρίνει και τα εμφανίζει στην οθόνη με το κατάλληλο σύμβολο ανάμεσά τους (<, > ή =). Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program SameNumeratorFractionsComparison;
begin
        loop begin
                num := NumeratorEntry( );   den1 := DenominatorEntry( );
                den2 := DenominatorEntry( );
                if  num = 0 then if  den1 = 1  then if  den2 = 1  then exit;
                SameNumeratorComparisonOutput
        end
end;
```

```
procedure SameNumeratorComparisonOutput;
begin
        if den1 = den2 then write(num," / ",den1," = ",num," / ", den2)
                else if  den1 > den2  then write(num," / ",den1," < ",num," / ",den2)
                        else write(num," / ",den1," > ",num," / ",den2);
        writeln
end.
```

7.4.3 (Πρόγραμμα 3.4.3) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται δύο κλάσματα, και να τα τρέπει σε ομώνυμα. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές ίσοι με τη μονάδα.

```
program FractionsToEquivalent;
begin
        loop begin
                num1 := NumeratorEntry( );   den1 := DenominatorEntry( );
                num2 := NumeratorEntry( );   den2 := DenominatorEntry( );
                if num1=0 then if num2=0 then if den1=1 then if den2=1 then exit;
                if den1 <> den2 then
                begin                                        (*μετατροπή σε ομώνυμα*)
                        lcm := FindLcm(den1, den2);
                        FractionsToEquivalentComputation
                end;
                FractionsToEquivalentOutput
        end
end;
```

```
function FindLcm(x, y) ;
begin
        a:=x;   b:=y;
        loop begin
                if  a = 0 then exit;   if  b = 0 then exit;
                (*Αντικατάσταση του μεγαλύτερου με το πηλίκο της διαίρεσης*)
                if  a > b  then  a := a mod b  else  b := b mod a
        end;
        mkd := a + b;
        lcm := a* b div mkd;
        return lcm
end;
```

```
procedure FractionsToEquivalentComputation;
begin
        temp:=lcm div den1;   num1:=num1 * temp;   den1:= lcm;
        temp:=lcm div den2;   num2:=num2 * temp;   den2:= lcm
end;
```

```
procedure FractionsToEquivalentOutput;
begin
        write("Πρώτο κλάσμα : ",num1," / ," den1);   writeln;
        write("Δεύτερο κλάσμα : ",num2," / ," den2);   writeln
end.
```

7.4.4 (Πρόγραμμα 3.4.4) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται δύο κλάσματα, να τα συγκρίνει και τα εμφανίζει στην οθόνη με το κατάλληλο σύμβολο ανάμεσά τους (<, > ή =). Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program FractionsComparison;
begin
        loop begin
                num1 := NumeratorEntry( );   den1 := DenominatorEntry( );
                num2 := NumeratorEntry( );   den2 := DenominatorEntry( );
                if num1=0 then if num2=0 then if den1=1 then if den2=1 then exit;
                if den1<>den2 then begin
                        lcm := FindLcm(den1, den2);
                        FractionsToEquivalentComputation
                end;
                EquivalentComparisonOutput
        end
end.
```

7.4.5 (Πρόγραμμα 3.4.5) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται δύο ακέραιους αριθμούς a και b και να υπολογίζει ένα κλάσμα μεγαλύτερο από το $\frac{a}{b}$ και μικρότερο από το $\frac{a+1}{b}$. Το πρόγραμμα σταματάει όταν δοθούν α=0 και β=1.

```
program IntermediaryFraction;
begin
        loop begin
                num := NumeratorEntry( );   den := DenominatorEntry( );
                if  num = 0  then if  den = 1  then  exit;
                IntermediaryFractionComputationOutput;
        end
end;
```

```
procedure IntermediaryFractionComputationOutput;
begin
        newden := den * 2;   newnum := num * 2 + 1;   num2 := a+1;
        write(num," / ",den," < ",newnum," / ",newden," < ",num2," / ",den);
        writeln
end.
```

Ασκήσεις

1. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται ένα κλάσμα, να το συγκρίνει μα τη μονάδα και να εμφανίζει στην οθόνη κατάλληλο μήνυμα. Το πρόγραμμα σταματάει όταν δοθεί αριθμητής ίσος με το μηδέν και παρονομαστής ίσος με τη μονάδα.

2. Να γραφεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα που να δέχεται ένα κλάσμα και να υπολογίζει:
   a. ένα μεγαλύτερο κλάσμα με
      i. διαφορετικό παρονομαστή
      ii. διαφορετικό αριθμητή
   b. ένα μικρότερο κλάσμα με
      i. διαφορετικό παρονομαστή
      ii. διαφορετικό αριθμητή

3. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα που να προσθέτει μία σταθερά στον αριθμητή και τον παρονομαστή ενός κλάσματος και να υπολογίζει αν το νέο κλάσμα είναι μικρότερο από το αρχικό.

Κεφάλαιο 7.5 – Πρόσθεση Κλασμάτων με Συναρτήσεις

7.5.1 (Πρόγραμμα 3.5.1) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται δύο ομώνυμα κλάσματα, τα προσθέτει και εμφανίζει το άθροισμά τους στη οθόνη. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program EquivalentSum;
begin
        loop begin
                num1 := NumeratorEntry( );   den := DenominatorEntry( );
                num2 := NumeratorEntry( );
                if num1= 0 then if num2=0 then if den=1 then exit;
                num3 := Add(num1, num2);
                FractionsSumOutput
        end
end;
```

```
procedure FractionsSumOutput;
begin
        write(num1,"/ ",den," + ",num2,"/ ",den," = ",num3,"/ ",den);  writeln
end.
```

7.5.2 (Πρόγραμμα 3.5.2) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται δύο όχι απαραίτητα ομώνυμα κλάσματα, τα προσθέτει και εμφανίζει το άθροισμά τους στη οθόνη. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program FractionsSum;
begin
        loop begin
                num1 := NumeratorEntry( );   den1 := DenominatorEntry( );
                num2 := NumeratorEntry( );   den2 := DenominatorEntry( );
                if num1=0 if num2=0 then if den1=1 then if den2=1 then exit;
                if den1<>den2 then begin
                        den := FindLcm(den1, den2);
                        FractionsToEquivalentComputation
                end;
                num3 := Add(num1, num2);
                FractionsSumOutput
        end
end.
```

Ασκήσεις

1. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται ένα κλάσμα και έναν ακέραιο αριθμό, τα προσθέτει και εμφανίζει το άθροισμά τους στη οθόνη. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

2. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται ένα κλάσμα, να το μετατρέπει σε μεικτό αριθμό και να το εμφανίζει στη οθόνη. Το πρόγραμμα σταματάει όταν δοθεί αριθμητής ίσος με το μηδέν και παρονομαστής ίσος με τη μονάδα.

3. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται έναν μεικτό αριθμό , να τον μετατρέπει σε κλάσμα και να το εμφανίζει στη οθόνη. Το πρόγραμμα σταματάει όταν δοθεί ο μεικτός αριθμός $0\frac{0}{1}$.

4. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται δύο μεικτούς αριθμούς, να τους προσθέτει και να εμφανίζει το άθροισμα στην οθόνη σε μορφή μικτού αριθμού. Το πρόγραμμα σταματάει όταν δοθεί ο μεικτοί αριθμοί ίσοι με $0\frac{0}{1}$.

Κεφάλαιο 7.6 – Αφαίρεση Κλασμάτων με Συναρτήσεις

7.6.1 (Πρόγραμμα 3.6.1) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται δύο ομώνυμα κλάσματα, τα αφαιρεί και εμφανίζει τη διαφορά τους στη οθόνη. Σημειώστε ότι το πρόγραμμα πρέπει να ελέγχει να είναι δυνατό να γίνει η αφαίρεση. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program EquivalentSub;
begin
        loop begin
                num1 := NumeratorEntry( );   den := DenominatorEntry( );
                num2 := NumeratorEntry( );
                if num1=0 then if num2=0 then if den=1 then  exit;
                if num1>num2 then begin
                        num3:=Subtract(num1, num2);
                        FractionsSubOutput
                end
                else SubErrorOutput
        end
end;
```

```
procedure FractionsSubOutput;
begin
        write(num1,"/ ",den," - ",num2,"/ ",den," = ",num3,"/ ",den);  writeln
end;
```

```
procedure SubErrorOutput;
        begin  write("Δεν είναι δυνατή η αφαίρεση."); writeln  end.
```

7.6.2 (Πρόγραμμα 3.6.2) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται δύο κλάσματα, τα αφαιρεί και εμφανίζει τη διαφορά τους στη οθόνη.

```
program FractionsSub;
begin
        loop begin
                num1 := NumeratorEntry( );   den1 := DenominatorEntry( );
                num2 := NumeratorEntry( );   den2 := DenominatorEntry( );
                if num1=0 if num2=0 then if den1=1 then if den2=1 then exit;
                if den1<>den2 then begin
                        den := FindLcm(den1, den2);
                        FractionsToEquivalentComputation
                end;
                if num1>num2 then begin
                        num3:=Subtract(num1, num2);
                        FractionsSubOutput
                end
                else SubErrorOutput
        end
end.
```

Κεφάλαιο 7.7 – Πολλαπλασιασμός Κλασμάτων με Συναρτήσεις

7.7.1 (Πρόγραμμα 3.7.1) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται δύο κλάσματα, τα πολλαπλασιάζει και εμφανίζει τη διαφορά τους στη οθόνη. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program Multiplication;
begin
        loop begin
                num1 := NumeratorEntry( );   den1 := DenominatorEntry( );
                num2 := NumeratorEntry( );   den2 := DenominatorEntry( );
                if num1=0 if num2=0 then if den1=1 then if den2=1 then exit;
                num3:= Multiply(num1, num2);
                den3 := Multiply(den1, den2);
                FractionsMultOutput
        end
end.
```

```
procedure FractionsMultOutput;
begin
        write(num1,"/ ",den," * ",num2,"/ ",den," = ",num3,"/ ",den3);   writeln;
end.
```

Ασκήσεις

1. Ένα μαγαζί κάνει έκπτωση σε όλα του τα είδη ίση με $\frac{10}{115}$ της αξίας κάθε προϊόντος. Να γραφεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται τιμές προϊόντων (μία κάθε φορά), να υπολογίζει και να εμφανίζει την τιμή του προϊόντος μετά την έκπτωση. Το πρόγραμμα να σταματάει όταν δοθεί τιμή προϊόντος ίση με το μηδέν.

2. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται ένα κλάσμα και να υπολογίζει το αντίστροφό του. Το πρόγραμμα σταματάει όταν δοθεί αριθμητής ίσος με το μηδέν και παρονομαστής ίσος με τη μονάδα.

3. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται δύο κλάσματα και να εξετάζει αν το πρώτο είναι αντίστροφο του δεύτερου. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές ίσοι με το μηδέν και παρονομαστές ίσοι με τη μονάδα.

Κεφάλαιο 7.8 – Διαίρεση Κλασμάτων με Συναρτήσεις

7.8.1 (Πρόγραμμα 3.9.1) Στο παρακάτω πρόγραμμα πραγματοποιείται διαίρεση δυο κλασμάτων. Καθώς εισάγει ο χρήστης καθένα αριθμητή και παρονομαστή του κάθε κλάσματος, γίνεται έλεγχος αν οι παρονομαστές είναι μη μηδενικοί.

```
program Division;
begin
        loop begin
                num1 := NumeratorEntry( );   den1 := DenominatorEntry( );
                num2 := NumeratorEntry( );   den2 := DenominatorEntry( );
                if num1=0 if num2=0 then if den1=1 then if den2=1 then exit;
                num3:= Multiply(num1, den2);
                den3 := Multiply(den1, num2);
                FractionsDivOutput
        end
end;
```

```
procedure FractionsDivOutput;
begin
        write(num1,"/ ",den," / ",num2,"/ ",den," = ",num3,"/ ",den3);  writeln;
end.
```

Εμφάνιση στην οθόνη
Δώστε τον αριθμητή του κλάσματος : 1
Δώστε τον παρονομαστή του κλάσματος : 0
Δώστε τον παρονομαστή του κλάσματος : 2
Δώστε τον αριθμητή του κλάσματος : 3
Δώστε τον παρονομαστή του κλάσματος : 4
1/2 / 3/4 = 4/6

7.8.2 (Πρόγραμμα 3.9.2) Στο παρακάτω πρόγραμμα ο χρήστης εισάγει τους αριθμητές και τους παρονομαστές των δύο κλασμάτων. Ο υπολογιστής τους διαβάζει και ελέγχει αν οι παρονομαστές είναι μη μηδενικοί. Στη συνέχεια ο χρήστης εισάγει το σύμβολο της πράξης, ο υπολογιστής το ελέγχει και υπολογίζει το αποτέλεσμα. Στην περίπτωση που έχουμε δώσει το δεύτερο κλάσμα μεγαλύτερο από το πρώτο και έχουμε αφαίρεση η πράξη δε μπορεί να γίνει οπότε εμφανίζεται κατάλληλο μήνυμα στην οθόνη. Τα ονόματα των μεταβλητών είναι ίδια με τα προηγούμενα προγράμματα. Το πρόγραμμα σταματάει όταν δοθούν αριθμητές κλασμάτων ίσοι με το μηδέν και παρονομαστές κλασμάτων ίσοι με τη μονάδα.

```
program Prakseis;
begin
    loop begin
        num1 := NumeratorEntry( );   den1 := DenominatorEntry( );
        num2 := NumeratorEntry( );   den2 := DenominatorEntry( );
        if num1=0 if num2=0 then if den1=1 then if den2=1 then exit;
        symbol := SymbolEntry( );
        if  symbol = 1  then begin
                if den1<>den2 then begin
                        den := FindLcm(den1, den2);
                        FractionsToEquivalentComputation
                end;
                num3 := Add(num1, num2);
                FractionsSumOutput;
        end;
        if  symbol = 2  then begin
                if den1<>den2 then begin
                        den := FindLcm(den1, den2);
                        FractionsToEquivalentComputation
                end;
                if num1>num2 then begin
                        num3:=Subtract(num1, num2);
                        FractionsSubOutput
                end
                else SubErrorOutput
        end;
        if  symbol = 3  then begin
                num3:= Multiply(num1, num2);
                den3 := Multiply(den1, den2);
                FractionsMultOutput
        end;
        if  symbol = 4  then begin
                num3:= Multiply(num1, den2);
                den3 := Multiply(den1, num2);
                FractionsDivOutput
        end
    end
end;
```

```
function SymbolEntry( );
begin
    loop begin
        write("Δώστε το σύμβολο της πράξης, 1 για πρόσθεση, 2 για");   writeln;
        write("αφαίρεση, 3 για πολλαπλασιασμό, 4 για διαίρεση : ");   readln(s);
        if  s < 1  then exit;   if  s > 4  then exit
    end;
    return s
end.
```

Εμφάνιση στην οθόνη
Δώστε τον αριθμητή του πρώτου κλάσματος:2
Δώστε τον παρονομαστή του πρώτου κλάσματος:0
Δώστε τον παρονομαστή του πρώτου κλάσματος:3
Δώστε τον αριθμητή του δεύτερου κλάσματος:3
Δώστε τον παρονομαστή του δεύτερου κλάσματος:4
Δώστε το σύμβολο της πράξης, 1 για πρόσθεση, 2 για
αφαίρεση, 3 για πολλαπλασιασμό, 4 για διαίρεση : 2
Δεν είναι δυνατή η αφαίρεση.

Κεφάλαιο 7.9 – Δεκαδικά Κλάσματα με Συναρτήσεις

7.9.1 (Πρόγραμμα 3.10.1) Το παρακάτω πρόγραμμα μετατρέπει δεκαδικά κλάσματα σε δεκαδικούς αριθμούς με τόσα δεκαδικά ψηφία όσα μηδενικά έχει ο παρονομαστής τους. Για καθένα μηδενικό του παρονομαστή μετακινεί την υποδιαστολή του αριθμητή κατά μια θέση προς τα αριστερά. Στην αρχή βέβαια γίνεται έλεγχος αν το κλάσμα που δόθηκε είναι όντως δεκαδικό. Το πρόγραμμα σταματάει όταν δοθεί αριθμητής κλάσματος ίσος με το μηδέν.

```
program Decimal1;
begin
        loop begin
                num := NumeratorEntry( );
                den := DecimalDenominatorEntry( );
                if  num = 0  then exit;
                result := DecimalFractionTransform(num, den);
                DecimalFractionOutput
        end
end;
```

```
function DecimalDenominatorEntry( );
begin
        loop begin
                write("Δώστε τον παρονομαστή του κλάσματος,");   writeln;
                write("(ακέραιο πολλαπλάσιο του 10 αλλά όχι μηδέν : ");
                readln(c);   if  c<>0  then if c mod 10 = 0  then exit
        end;
        return c
end;
```

```
function DecimalFractionTransform(a, b);
begin
        loop begin
                b := b mod 10;   a := a / 10;
                if b=0 then exit
        end;
        return a
end;
```

```
procedure DecimalFractionOutput;
        begin   write("Ο δεκαδικός αριθμός είναι : ", result);   writeln   end.
```

Κεφάλαιο 7.10 – Δεκαδικοί και Μικτοί Αριθμοί με Συναρτήσεις

7.10.1 (Πρόγραμμα 3.11.1) Το παρακάτω πρόγραμμα μετατρέπει κλάσματα σε δεκαδικούς και μικτούς αριθμούς. Το πρόγραμμα σταματάει όταν δοθεί αριθμητής κλάσματος ίσος με το μηδέν και παρονομαστής ίσος με τη μονάδα.

```
program Decimal2;
begin
        loop begin
                num := NumeratorEntry( );   den := DenominatorEntry( );
                if  num = 0  then if  den = 1  then exit;
                c := Divide(a, b);
                DecimalFractionOutput;
                if a>=b then begin
                        d := MixedComputation(a, b);   e := MixedOutput(a, b)
                end
        end
end;
```

```
function MixedNumberComputation(x, y)  = x div y;
```

```
function MixedNumeratorComputation(x, y)  = x mod y;
```

```
function MixedOutput;
        begin   write("Ο μικτός αριθμός είναι : ",d,"  ","e,"/",b);   writeln   end.
```

Άσκηση.

1. Να βελτιωθεί το πρόγραμμα που πραγματοποιεί τις τέσσερις πράξεις των κλασμάτων έτσι ώστε να γράφει το αποτέλεσμα σε μορφή μικτού και δεκαδικού αριθμού.

Κεφάλαιο 7.11 – Εφαρμογές Ποσοστών με Συναρτήσεις

7.11.1 (Πρόγραμμα 3.12.1) Ένα κατάστημα την περίοδο των εκπτώσεων κάνει στα παντελόνια έκπτωση 15% και στις μπλούζες έκπτωση 20%. Το παρακάτω πρόγραμμα δέχεται την αρχική τιμή του κάθε ρούχου και υπολογίζει το ποσό της έκπτωσης και το συνολικό ποσό που πρέπει να πληρώσει ο πελάτης.

```
program Discount;
begin
        loop begin
                trousers := TrousersNumEntry( );
                sumtrousers := PricesEntry(trousers);
                tshirts := TShirtsEntry( );   sumtshirts := PricesEntry(tshirts);
                DiscountComputations;   DiscountOutput;
                if  AskUserToContinue( )  then exit
        end
end;
```

```
function TrousersNumEntry( );
begin
        write ("Δώστε τον αριθμό των παντελονιών : ");   readln (n);   return n
end;
```

```
function TShirtsEntry( );
begin
        write ("Δώστε τον αριθμό των tshirts : ");   readln (n);   return n
end;
```

```
function PricesEntry(n) ;
begin
        sum:=0;
        for i:=1 to n do
        begin
                write ("Προϊόν ",i,".  Αρχική τιμή : ");   readln (price);
                sum:= sum + price
        end;
        return sum
end;
```

```
procedure DiscountComputations;
begin
        discounttrousers := sumtrousers * 0,15;
        sumtrousers := sumtrousers - discounttrousers;
        discounttshirts := sumtshirts * 0,2;
        sumtshirts := sumtshirts - discounttshirts;
        sum := sumtrousers+sumtshirts;
        discount := discounttrousers + discounttshirts
end;
```

```
procedure DiscountOutput;
begin
        write (“Συνολικό ποσό : “,sum,”ευρώ”);   writeln;
        write (“Συνολική έκπτωση : “,discount,”ευρώ”);   writeln
end.
```

Εμφάνιση στην οθόνη

Δώστε τον αριθμό των παντελονιών:2
Προϊόν 1. Αρχική τιμή : 70
Προϊόν 2. Αρχική τιμή : 65
Δώστε τον αριθμό των T-shirt:3
Προϊόν 1. Αρχική τιμή : 28
Προϊόν 2. Αρχική τιμή : 35
Προϊόν 3. Αρχική τιμή : 40
Συνολικό ποσό: 197,15 ευρώ
Συνολική έκπτωση: 40,85 ευρώ

Κεφάλαιο 7.12 – Η Έννοια των Ανάλογων Ποσών με Συναρτήσεις

7.12.1 (Πρόγραμμα 4.1.1) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται ζεύγη αντίστοιχων τιμών δύο ποσών. Για κάθε ζεύγος που εισάγεται, το πρόγραμμα πρέπει να εξετάζει αν τα ποσά προκύπτουν ανάλογα για όλα τα ζεύγη που έχουν εισαχθεί μέχρι τότε. Αν τα ποσά πράγματι είναι ανάλογα, τότε το πρόγραμμα ρωτά τον χρήστη αν επιθυμεί να εισάγει νέο ζεύγος, ή όχι, οπότε εμφανίζει κατάλληλο μήνυμα στην οθόνη. Αν τα ποσά δεν είναι ανάλογα, τότε το πρόγραμμα σταματά και εμφανίζει στην οθόνη κατάλληλο μήνυμα.

```
program ProportionalSums;
begin
        IntroductionProportionalSums;
        value1 := GetValue1( );   value2 := GetValue2( );
        ratio1 := ComputeRatio(value1, value2);
        loop begin
                value1 := GetValue1( );   value2 := GetValue2( );
                ratio2 := Divide(value1, value2);
                if  ratio1 <> ratio2  then Negative else Positive;
                if  AskUserToContinue( )  then exit
        end
end;
```

```
procedure IntroductionProportionalSums;
begin
        write("Αυτό το πρόγραμμα εξετάζει τα ζεύγη αντίστοιχων");   writeln;
        write("τιμών δύο ποσών και εξετάζει αν τα δύο ποσά");   writeln;
        write("είναι ανάλογα.");   writeln
end;
```

```
function GetValue1( );
begin
        write("Δώστε την πρώτη τιμή του ζεύγους : ");   readln(v1);   return v1;
end;
```

```
function GetValue2( );
begin
        loop begin
                write("Δώστε τη δεύτερη τιμή του ζεύγους (όχι όμως μηδέν) : ");
                readln(v2);   if  v2 <> 0  then exit
        end;
        return v2
end;
```

```
procedure Negative;
        begin   write("Τα ποσά δεν είναι ανάλογα!");   writeln   end;
```

```
procedure Positive;
        begin   write("Τα ποσά είναι ανάλογα!");   writeln   end.
```

Κεφάλαιο 7.13 – Εφαρμογές Ανάλογων Ποσών με Συναρτήσεις

7.13.1 (Πρόγραμμα 4.2.1) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται ένα ζεύγος τιμών δύο ανάλογων ποσών και μία τιμή ενός δεύτερου ζεύγους. Το πρόγραμμα υπολογίζει την τιμή που αντιστοιχεί στην τελευταία είσοδο του χρήστη. Σχηματικά, το πρόγραμμα υπολογίζει τον άγνωστο Χ του παρακάτω πίνακα, με την προϋπόθεση ότι τα δύο ποσά είναι ανάλογα.

Ποσό Α	a1 = 2	a2 = 3
Ποσό Β	b1 = 400	b2 = X

```
program SolveProportionalSums;
begin
        IntroductionSolveProportionalSums;
        loop begin
                a1 := GetA( );   b1 := GetB( );   a2 := GetA( );
                b2 := ComputeProportionalSums(a1, a2, b1);
                OutputProportionalSums;
                if  AskUserToContinue( )  then exit
        end
end;
```

```
procedure IntroductionSolveProportionalSums;
begin
        write("Αυτό το πρόγραμμα δέχεται ένα ζεύγος τιμών "); writeln;
        write("ανάλογων ποσών και μια τιμή από ένα δεύτερο ζεύγος "); writeln;
        write("και υπολογίζει τη δεύτερη τιμή του ζεύγους. "); writeln
end;
```

```
function GetA( );
begin
        loop begin
                write("Δώστε την πρώτη τιμή ζεύγους (όχι μηδέν) : ");
                readln(a);   if  a <> 0  then exit
        end;
        return a
end;
```

```
function GetB( );
begin
        loop begin
                write("Δώστε τη δεύτερη τιμή του ζεύγους (όχι μηδέν) : ");
                readln(b);   if  b <> 0  then exit
        end;
        return b
end;
```

```
function ComputeProportionalSums(a1, a2, b1)  = b1 * a2 / a1;
```

```
procedure OutputProportionalSums;
        begin   write("Δεύτερη τιμή, δεύτερου ζεύγους : ",b2);   writeln   end.
```

7.13.2 (Πρόγραμμα 4.2.2) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται την αρχική τιμή ενός προϊόντος, το ποσοστό της έκπτωσης ή της αύξησης της τιμής και να υπολογίζει την τελική τιμή του. Το πρόγραμμα σταματάει όταν δοθεί μηδενική αρχική τιμή.

```
program ComputePrices;
begin
        IntroductionComputePrices;
        loop begin
                firstprice := GetFirstPrice( );   plusminus := GetPlusMinus( );
                percentage := GetPercentage( );
                ComputeOutputComputePrices;
                if  AskUserToContinue( )  then exit
        end
end;
```

```
procedure IntroductionComputePrices;
begin
        write("Αυτό το πρόγραμμα δέχεται την αρχική τιμή ενός"); writeln;
        write("προϊόντος, το ποσοστό της έκπτωσης ή της αύξησης"); writeln;
        write("της τιμής και υπολογίζει την τελική τιμή του."); writeln
end;
```

```
function GetFirstPrice( );
begin
        loop begin
                write("Δώστε την αρχική τιμή του προϊόντος : ");
                readln(fp);   if  fp <> 0  then exit
        end;
        return fp
end;
```

```
function GetPlusMinus( );
begin
        loop begin
                write("Αύξηση ή μείωση τιμής. Δώστε 1 για αύξηση 0 για μείωση : ");
                readln(pm);   if pm=0 then exit;   if pm=1 then exit
        end;
        return pm
end;
```

```
function GetPercentage( );
begin
        write(“Δώστε ποσοστό επί τοις εκατό : “);   readln(p);   return p
end;
```

```
procedure ComputeOutputComputePrices;
begin
        change := firstprice * percentage / 100;
        if plusminus=1 then  finalprice := firstprice + change
                       else  finalprice := firstprice – change;
        write(“Τελική τιμή : “,finalprice);   writeln
end.
```

7.13.3 (Πρόγραμμα 4.2.3) Να γραφτεί, με χρήση συναρτήσεων, πρόγραμμα το οποίο να δέχεται το κεφάλαιο που κατατίθεται στην τράπεζα, το επιτόκιο και να υπολογίζει τα χρήματα που θα εισπραχθούν μετά από ένα έτος.

```
program Bank;
begin
        IntroductionBank;   capital := GetCapital( );
        interest := GetInterest( );
        newcapital := ComputeNewCapital(capital, interest);
        ShowNewCapital
end;
```

```
procedure IntroductionBank;
begin
        write(“Το πρόγραμμα δέχεται το κεφάλαιο που κατατίθεται”); writeln;
        write(“στην τράπεζα, το επιτόκιο και να υπολογίζει τα”); writeln;
        write(“ χρήματα που θα εισπραχθούν μετά από ένα έτος.”); writeln;
end;
```

```
function GetCapital( );
begin
        loop begin
                write("Δώστε το αρχικό κεφάλαιο : "); readln(c);
                if  c > 0  then exit
        end
end;
```

```
function GetInterest( );
begin
        loop begin
                write("Δώστε το επιτόκιο : "); readln(i);
                if  i > 0  then exit
        end
end;
```

```
function ComputeNewCapital(c, i)  = c + (c * i);
```

```
procedure ShowNewCapital;
        begin   write("Κεφάλαιο το επόμενο έτος : ",newcapital);   writeln   end.
```

7.13.4 (Πρόγραμμα 4.2.4) Ένα σιδερένιο σώμα έχει τη μορφή ορθογωνίου παραλληλεπιπέδου με μήκος length1, πλάτος width1, ύψος height1 και βάρος weight1. Ένα δεύτερο ορθογώνιο παραλληλεπίπεδο από το ίδιο υλικό έχει μήκος length2, πλάτος width2, ύψος height2 και βάρος weight2. Να γραφτεί, με χρήση συναρτήσεων, πρόγραμμα το οποίο να δέχεται τις διαστάσεις των δύο ορθογωνίων παραλληλεπιπέδων και το βάρος του πρώτου και να υπολογίζει το βάρος του δεύτερου. Το πρόγραμμα σταματάει όταν δοθούν: length1 = 0, width1 = 0 και height1 = 0.

Τα βάρη των ορθογωνίων παραλληλεπιπέδων είναι ανάλογα του όγκου τους. Ο όγκος του ορθογωνίου παραλληλογράμμου υπολογίζεται από το γινόμενο του μήκους, του πλάτους και του ύψους του.

```
program Rectangle;
begin
        loop begin
                l1 := GetLength( );   w1 := GetWidth( );   h1 := GetHeight( );
                if l1=0 then if w1=0 then if h1=0 then exit;
                weight1 := GetWeight( );
                l2 := GetLength( );   w2 := GetWidth( );   h2 := GetHeight( );
                v1 := ComputeVolume(l1, w1, h1);
                v2 := ComputeVolume(l2, w2, h2);
                ShowWeight2
        end
end;
```

```
function GetLength( );
begin
        write("Δώστε το μήκος του ορθογωνίου : ");   readln(l);   return l
end;
```

```
function GetWidth( );
begin
        write("Δώστε το πλάτος του ορθογωνίου : ");   readln(w);   return w
end;
```

```
function GetHeight( );
begin
        write("Δώστε το ύψος του ορθογωνίου : ");   readln(h);   return h
end;
```

```
function GetWeight( );
begin
        write("Δώστε το βάρος του ορθογωνίου : ");   readln(w);   return w
end;
```

```
function ComputeVolume(x, y, z)  = x * y * z;
```

```
procedure ShowWeight2;
begin
        w2 := v2 * w1 / v1;
        write("Βάρος του 2ου ορθογωνίου : ",w2);   writeln
end.
```

Ασκήσεις

1. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται την αρχική και την τελική τιμή ενός προϊόντος και να υπολογίζει το ποσοστό έκπτωσης ή αύξησης της τιμής. Το πρόγραμμα σταματάει όταν δοθεί μηδενική αρχική και τελική τιμή.

2. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται την τελική τιμή και το ποσοστό έκπτωσης ή αύξησης της τιμής ενός προϊόντος και να υπολογίζει την αρχική του τιμή. Το πρόγραμμα τερματίζει όταν δοθεί μηδενική τελική τιμή.

3. Να γραφτεί, με βάση το πρόγραμμα 4.2.3, ένα πρόγραμμα το οποίο να δέχεται το κεφάλαιο που κατατίθεται στην τράπεζα, το επιτόκιο, ο αριθμός των ετών και να υπολογίζει τα χρήματα που θα εισπραχθούν μετά το πέρας των ετών που εισάγει ο χρήστης. Προσέξτε ότι ο τόκος κάθε έτους προστίθεται στο κεφάλαιο και το άθροισμα αποτελεί το κεφάλαιο του επόμενου έτους.

Κεφάλαιο 7.14 – Κλίμακες με Συναρτήσεις

7.14.1 (Πρόγραμμα 4.3.1) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται την απόσταση δύο σημείων στο χάρτη και την κλίμακα του χάρτη και να υπολογίζει την πραγματική απόσταση. Το πρόγραμμα σταματάει όταν δοθεί μηδενική απόσταση στο χάρτη.

```
program Climax;
begin
        loop begin
                mapdistance := GetMapDistance( );
                if mapdistance=0 then exit;
                mapclimax := GetMapClimax( );
                ComputeShowRealDistance
        end
end;
```

```
function GetMapDistance( );
begin
        write("Δώστε την απόσταση στο χάρτη : "); readln(md); return md
end;
```

```
function GetMapClimax( );
begin
        write("Η κλίμακα του χάρτη δίνεται στη μορφή 1/α"); writeln;
        write("Δώστε το α : "); readln(mc); return mc
end;
```

```
procedure ComputeShowRealDistance;
begin
        realdistance := mapdistance * mapclimax;
        write("Η πραγματική απόσταση είναι : ", realdistance); writeln
end.
```

Ασκήσεις

1. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται την πραγματική απόσταση δύο σημείων και την κλίμακα του χάρτη και να υπολογίζει την απόσταση στο χάρτη. Το πρόγραμμα τερματίζει όταν δοθεί μηδενική πραγματική απόσταση.

2. Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να δέχεται την απόσταση δύο σημείων στο χάρτη και την πραγματική απόσταση και να υπολογίζει την κλίμακα του χάρτη. Το πρόγραμμα τερματίζει όταν δοθεί μηδενική απόσταση στο χάρτη και μηδενική πραγματική απόσταση.

Κεφάλαιο 7.15 – Μερισμός σε Μέρη Ανάλογα με Συναρτήσεις

7.15.1 (Πρόγραμμα 4.4.1) Να γραφτεί, με χρήση επανάληψης, διαδικασιών και συναρτήσεων, πρόγραμμα το οποίο να λύνει προβλήματα μερισμού σε μέρη ανάλογα, βάσει της σχέσης:

$$\frac{x}{\alpha}=\frac{y}{\beta}=\frac{z}{\gamma}=\frac{x+y+z}{\alpha+\beta+\gamma}$$

Το πρόγραμμα δέχεται τις τιμές x, y, z και το άθροισμα α+β+γ και υπολογίζει τις τιμές α, β και γ. Το πρόγραμμα σταματάει όταν δοθεί μηδενικό άθροισμα α + β + γ ή όταν δοθούν συγχρόνως οι τιμές x = 0, y = 0 και z = 0.

```
program ProportionalSeparation;
begin
        IntroductionProportionalSeparation;
        loop begin
                x := GetX( );   y := GetY( );   z := GetZ( );
                if x=0 then if y=0 then if z=0 then exit;
                abc := GetABC( );   if abc=0 then exit;
                ComputeResults;  ShowResults
        end
end;
```

```
procedure IntroductionProportionalSeparation;
begin
        write("Το πρόγραμμα λύνει προβλήματα μερισμού σε μέρη ανάλογα");
        writeln
end;
```

```
function GetX( );
        begin  write(“Δώστε την τιμή του x : “); readln(x);  return x   end;
```

```
function GetY( );
        begin  write(“Δώστε την τιμή του y : “); readln(y);  return y   end;
```

```
function GetZ( );
        begin  write(“Δώστε την τιμή του z : “); readln(z);   return z   end;
```

```
function GetABC( );
begin
        write(“Δώστε την τιμή του αθροίσματος a+b+c : “);
        readln(abc);   return abc
end;
```

```
procedure ComputeResults;
begin
        xyz := x + y + z;   l := xyz / abc;
        a := x / l;   b := y / l;   c := z / l
end;
```

```
procedure ShowResults;
begin
        write(“Η τιμή του α είναι : “, a);   writeln;
        write(“Η τιμή του β είναι : “, b);   writeln;
        write(“Η τιμή του γ είναι : “, c);   writeln
end.
```

Κεφάλαιο 8

Οι Ρητοί Αριθμοί

ΚΕΦΑΛΑΙΟ 8.1 – Οι θετικοί και οι αρνητικοί αριθμοί.

Ως τώρα, οι αριθμοί που γνωρίζουμε είναι μεγαλύτεροι ή ίσοι του μηδενός. Οι αριθμοί αυτοί αποτελούν το σύνολο των **φυσικών αριθμών (N)**.

Χαρακτηριστικά των φυσικών αριθμών είναι:

- όλοι οι φυσικοί αριθμοί είναι μεγαλύτεροι ή ίσοι του μηδενός.
- αριστερά από τα ψηφία κάθε φυσικού αριθμού υπάρχει το σύμβολο «+». Τις περισσότερες φορές, όμως, το σύμβολο αυτό παραλείπεται.

Για παράδειγμα, μερικοί φυσικοί αριθμοί είναι: +7, +23, +56, +102, +102365 οι οποίοι έχουν ακριβώς την ίδια σημασία με τους αριθμούς: 7, 23, 56, 102, 102365 αντίστοιχα.

Το σύνολο των φυσικών αριθμών συμβολίζεται με N, δηλαδή:

$$N=\{0, 1, 2, 3, 4, 5\ldots\}$$

Όμως, πολλές φορές στην καθημερινή μας ζωή χρειάζεται να εκφράσουμε κάποια ποσότητα η οποία είναι μικρότερη από το μηδέν. Για παράδειγμα, πολλές φορές η θερμοκρασία το χειμώνα είναι $3^{\circ}C$ βαθμούς κάτω από το μηδέν.

Αν χρησιμοποιήσουμε τους φυσικούς αριθμούς είναι αδύνατο να συμβολίσουμε την έννοια ενός αριθμού μικρότερου από το μηδέν. Αυτό συμβαίνει επειδή το σύνολο των φυσικών αριθμών περιέχει αριθμούς μεγαλύτερους ή ίσους με το μηδέν, άρα εξ'ορισμού δεν μπορούν να εκφράσουν το ζητούμενο.

Για αυτό το λόγο, χρησιμοποιούμε τους **αρνητικούς αριθμούς**.

Χαρακτηριστικά των αρνητικών αριθμών είναι ότι:

- όλοι οι αρνητικοί αριθμοί είναι μικρότεροι από το μηδέν
- αριστερά από τα ψηφία του αριθμού υπάρχει το σύμβολο «–».

ΠΡΟΣΟΧΗ!!!

Το σύμβολο «–» είναι απαραίτητο όταν αναφερόμαστε σε αρνητικούς αριθμούς, για να αποφεύγεται η σύγχυση με τους φυσικούς.

Για παράδειγμα: $+2=2$
$-2\neq 2$

Τα σύμβολα «+» και «–» καλούνται **πρόσημα**.

Επομένως, χρησιμοποιώντας τους αρνητικούς αριθμούς μπορούμε να εκφράσουμε τιμές μεγεθών μικρότερες από το μηδέν.

Για παράδειγμα:

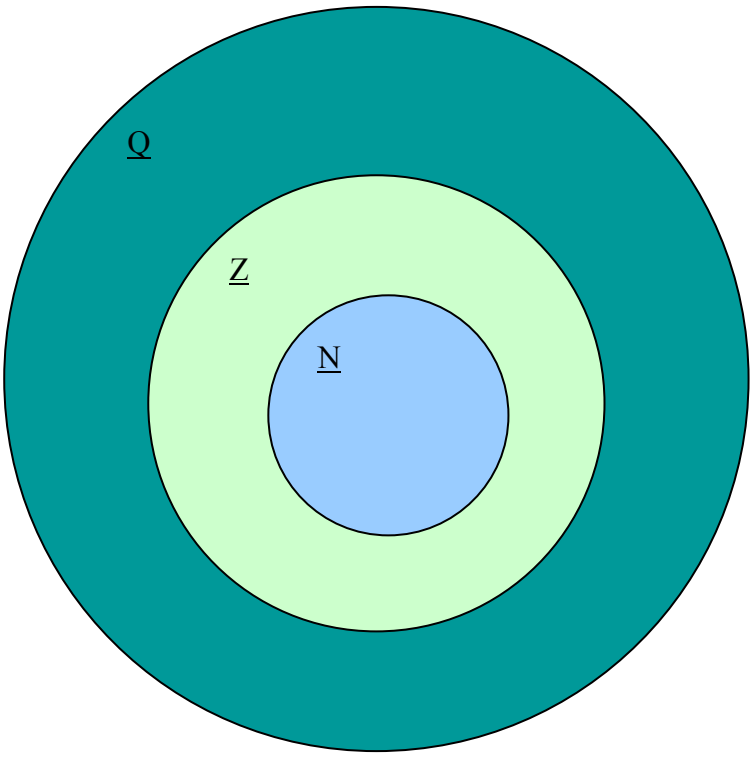
- $7^\circ C$ κάτω από το μηδέν = $-7^\circ C$
- 23 μέτρα κάτω από την επιφάνεια της θάλασσας = -23 μέτρα

Το σύνολο που περιλαμβάνει τους φυσικούς και τους αρνητικούς αριθμούς, λέγεται **σύνολο των ακεραίων αριθμών (Z)**, δηλαδή:

$$Z=\{\ldots -4, -3, -2, -1, 0, 1, 2, 3, 4, \ldots\}$$

Όλοι οι γνωστοί μέχρι τώρα αριθμοί, δηλαδή οι φυσικοί αριθμοί (N), οι αρνητικοί αριθμοί, τα κλάσματα και οι δεκαδικοί αριθμοί συναποτελούν το **σύνολο των ρητών αριθμών** (Q).

Είναι φανερό ότι μεταξύ Q, N, Z ισχύει:



- Δύο μη μηδενικοί ρητοί αριθμοί λέγονται **ομόσημοι** αν έχουν το ίδιο πρόσημο.
- Δύο μη μηδενικοί ρητοί αριθμοί λέγονται **ετερόσημοι** αν έχουν διαφορετικό πρόσημο.

Το ερώτημα που τίθεται είναι αν ο ηλεκτρονικός υπολογιστής υποστηρίζει τους αρνητικούς αριθμούς.

Ο ηλεκτρονικός υπολογιστής δέχεται οποιονδήποτε ρητό αριθμό. Για τους αρνητικούς αρκεί να προσθέσουμε το πρόσημο «–». Οι ιδιότητες των αριθμών παραμένουν αναλλοίωτες.

8.1.1 Θα κατασκευάσουμε ένα πρόγραμμα το οποίο δέχεται έναν αριθμό, ελέγχει αν είναι θετικός ή αρνητικός και εμφανίζει αντίστοιχο μήνυμα στην οθόνη.

```
program PositiveNegativeCheck;
begin
        write(“Δώστε αριθμό : “);   readln(num);
        if num<0 then write(“Ο αριθμός είναι αρνητικός.”)
                 else write(“Ο αριθμός είναι θετικός.”);
        writeln
end.
```

ΚΕΦΑΛΑΙΟ 8.4 – Απόλυτη τιμή ρητού αριθμού. Αντίθετοι αριθμοί.

Ως απόλυτη τιμή, ορίζεται η απόσταση ενός αριθμού από το μηδέν. Η απόλυτη τιμή ενός αριθμού a συμβολίζεται ως $|a|$ και είναι πάντα θετικός αριθμός.

Απόλυτη τιμή ορίζεται για όλους τους αριθμούς.

Ας δούμε μερικά παραδείγματα:

$$|+5| = 5\,,\quad |63| = 63\,,\quad \left|\frac{3}{4}\right| = \frac{3}{4}\,,\quad |-54| = 54\,,\quad \left|-\frac{9}{4}\right| = \frac{9}{4}\,,\quad |0{,}325| = 0{,}325$$

Από τα παραδείγματα φαίνεται ότι μπορεί να διατυπωθεί ένας κανόνας για την εύρεση της απόλυτης τιμής ενός αριθμού:

- ✓ Η απόλυτη τιμή ενός **θετικού** αριθμού είναι ο ίδιος ο αριθμός
- ✓ Η απόλυτη τιμή ενός **αρνητικού** αριθμού είναι ο αριθμός με αντίθετο πρόσημο.

Για παράδειγμα:

$$|-5| = 5$$
$$|+5| = 5$$

Αυτοί οι αριθμοί λέγονται αντίθετοι. Δηλαδή:

Δύο ετερόσημοι αριθμοί λέγονται **αντίθετοι** αν έχουν την ίδια απόλυτη τιμή.

Σύμφωνα με τον παραπάνω ορισμό, **ο αντίθετος του αριθμού** a **είναι ο** $-a$**.**

8.4.1 Το παρακάτω πρόγραμμα υπολογίζει την απόλυτη τιμή του αριθμού που δέχεται ως είσοδο.

```
program Abs1;
begin
        write(“Δώστε αριθμό : “);   readln(num);
        if  num < 0  then    absnum := – num   else   absnum := num;
        write(“Η απόλυτη τιμή του ”, num,”είναι: “,absnum);   writeln
end.
```

Υπάρχει η έτοιμη συνάρτηση abs, η οποία επιστρέφει την απόλυτη τιμή του αριθμού που δίνεται σε αυτήν ως όρισμα. Οπότε, προκύπτει το ισοδύναμο πρόγραμμα:

```
program Abs2;
begin
        write(“Δώστε αριθμό : “);   readln(num);
        absnum := abs(num);
        write(“Η απόλυτη τιμή του ”, num,”είναι: “,absnum);   writeln
end.
```

ΚΕΦΑΛΑΙΟ 8.6 – Πρόσθεση ρητών αριθμών.

- Για να προσθέσουμε δύο ομόσημους αριθμούς, προσθέτουμε τις απόλυτες τιμές τους και στο άθροισμα τους βάζουμε το κοινό τους πρόσημο.
- Για προσθέσουμε δύο ετερόσημους αριθμούς, αφαιρούμε τη μικρότερη απόλυτη τιμή από τη μεγαλύτερη και στη διαφορά αυτή βάζουμε το πρόσημο του αριθμού που έχει τη μεγαλύτερη απόλυτη τιμή.

Ο υπολογιστής εφαρμόζει τους παραπάνω κανόνες και υπολογίζει κατευθείαν το αποτέλεσμα της πρόσθεσης.

Για παράδειγμα, το πρόγραμμα *Variables1,* που έχει δοθεί στην ανάλυση της εντολής readln, ισχύει για όλους τους αριθμούς, θετικούς και αρνητικούς.

Άσκηση.

1. Εκτελέστε τα προγράμματα πρόσθεσης (1.7.1) και σύγκρισης αριθμών (5.2.1), εισάγοντας θετικούς και αρνητικούς αριθμούς. Κρίνετε τη συμπεριφορά τους ικανοποιητική;

ΚΕΦΑΛΑΙΟ 8.7 – Αφαίρεση ρητών αριθμών.

Πριν γνωρίσουμε τους αρνητικούς αριθμούς, η πράξη $7-9$ ήταν αδύνατη. Όμως, με χρήση των αρνητικών αριθμών, μια τέτοια αφαίρεση είναι δυνατή. Οι κανόνες που ισχύουν για την αφαίρεση δύο αριθμών είναι οι εξής:

Για να βρούμε τη διαφορά δύο αριθμών, προσθέτουμε στον μειωτέο τον αντίθετο του αφαιρετέου.

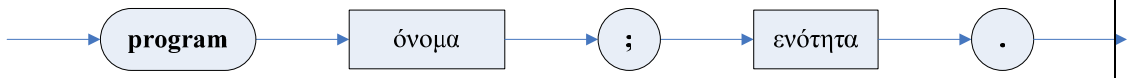
Δηλαδή προσδιορίζουμε τον αντίθετο του αφαιρετέου και μετατρέπουμε την αφαίρεση σε πρόσθεση. Όπως στην πρόσθεση έτσι και στη αφαίρεση, ο ηλεκτρονικός υπολογιστής εφαρμόζει τον παραπάνω κανόνα.

Άσκηση.

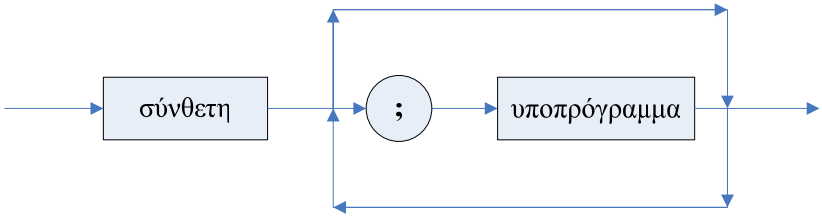
1. Εκτελέστε τα προγράμματα αφαίρεσης 1.8.1, 5.5.2 και 5.5.3, εισάγοντας θετικούς και αρνητικούς αριθμούς. Τα αποτελέσματα είναι τα αναμενόμενα;

ΠΑΡΑΡΤΗΜΑ Α΄

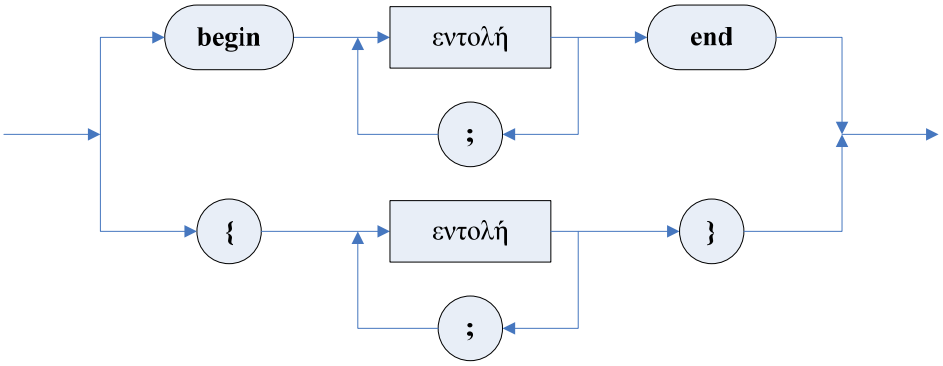
ΣΥΝΤΑΚΤΙΚΑ ΔΙΑΓΡΑΜΜΑΤΑ PASCHOOL I



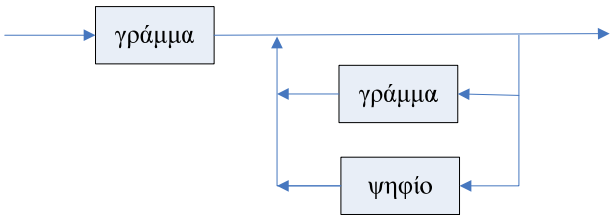
Συντακτικό διάγραμμα 1: ***πρόγραμμα***



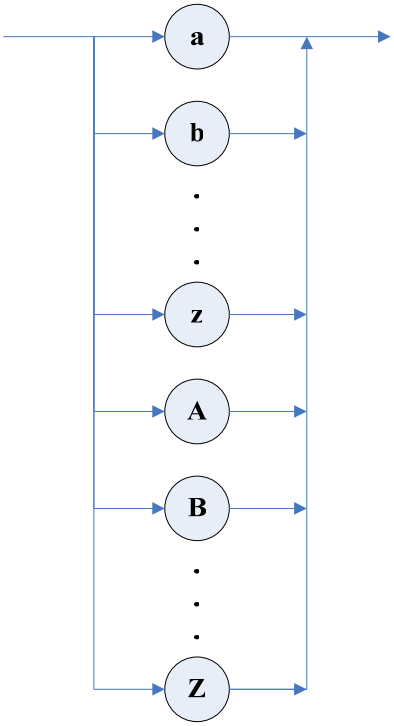
Συντακτικό διάγραμμα 2: ***ενότητα***



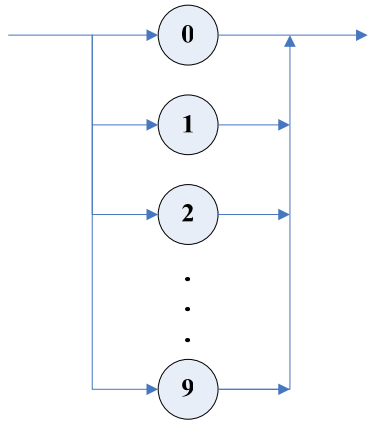
Συντακτικό διάγραμμα 3: ***σύνθετη***



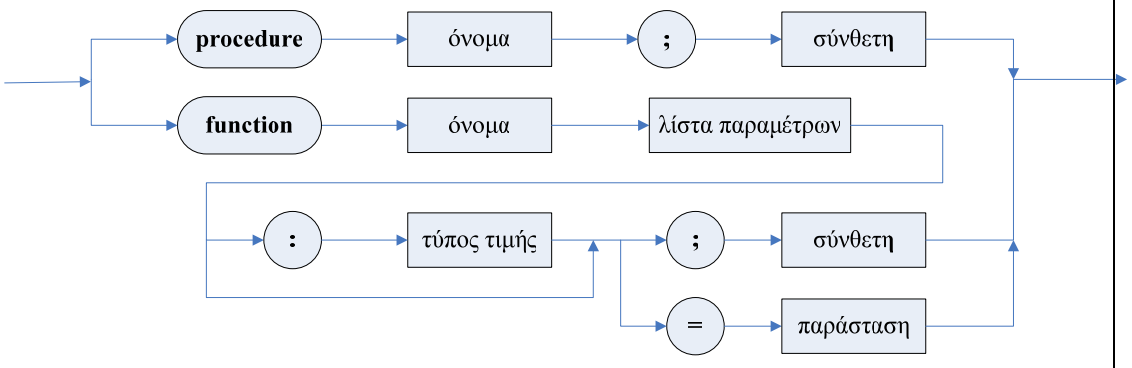
Συντακτικό διάγραμμα 4: ***όνομα***



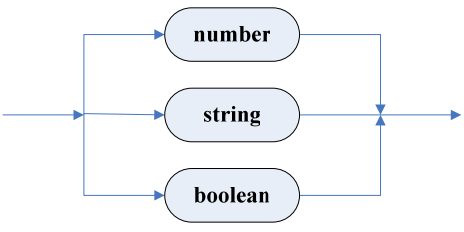
Συντακτικό διάγραμμα 5: ***γράμμα***



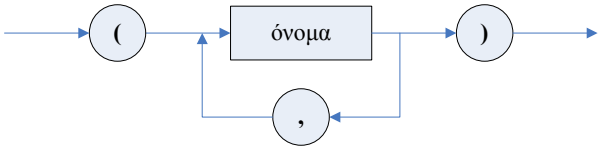
Συντακτικό διάγραμμα 6: ***ψηφίο***



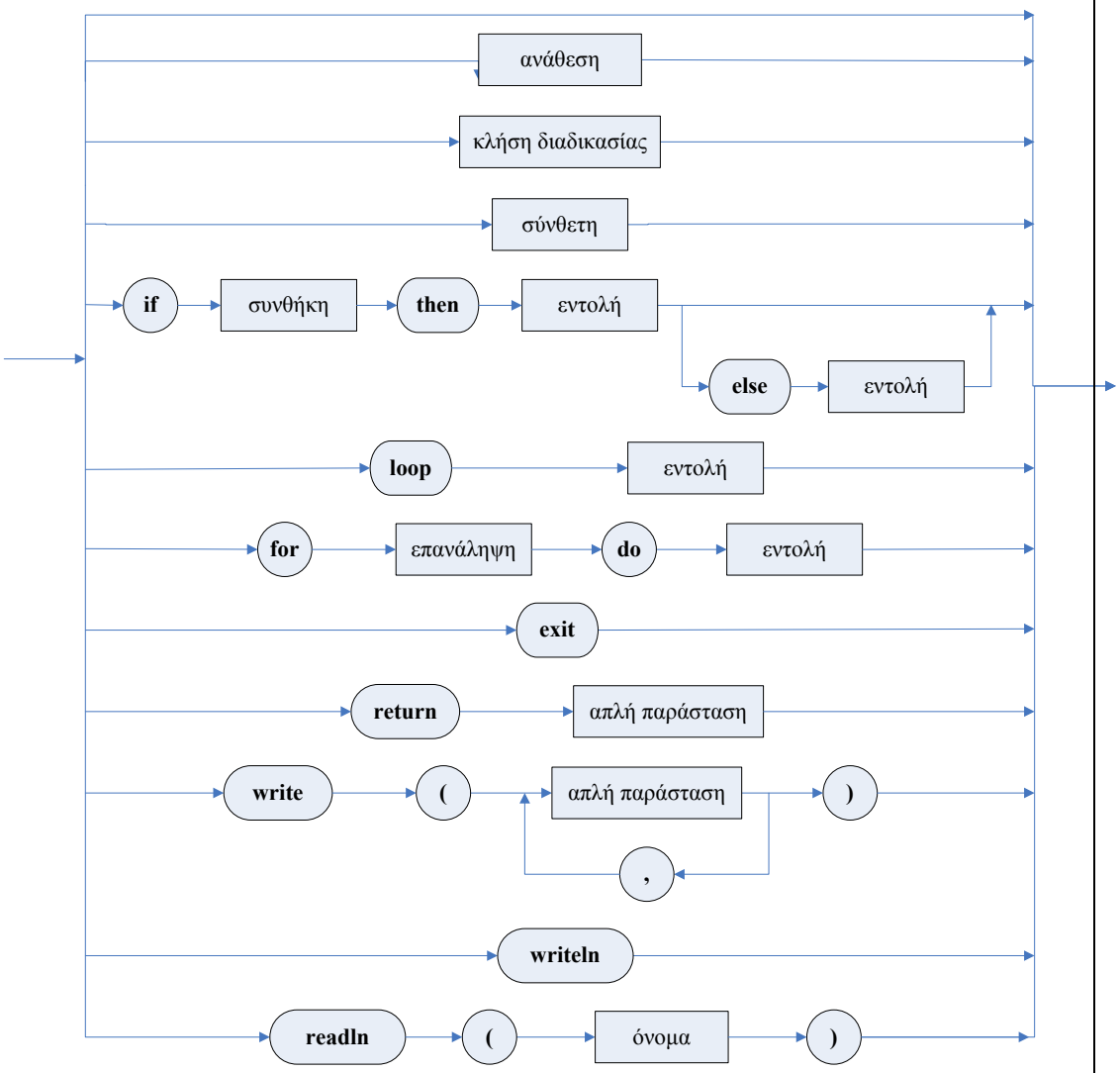
Συντακτικό διάγραμμα 7: ***υποπρόγραμμα***



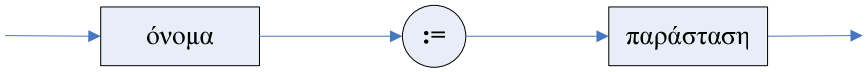
Συντακτικό διάγραμμα 8: ***τύπος τιμής***



Συντακτικό διάγραμμα 9: ***λίστα παραμέτρων***

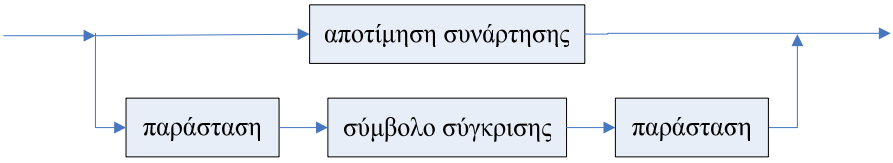


Συντακτικό διάγραμμα 10: ***εντολή***

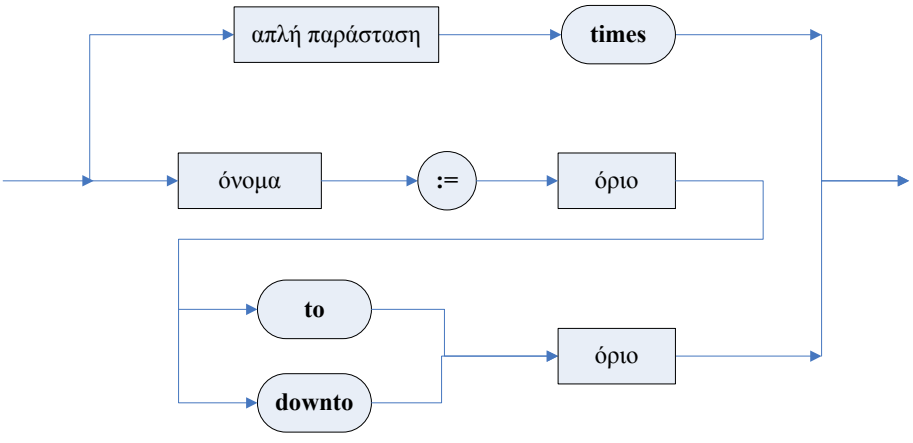


Συντακτικό διάγραμμα 11: ***ανάθεση***

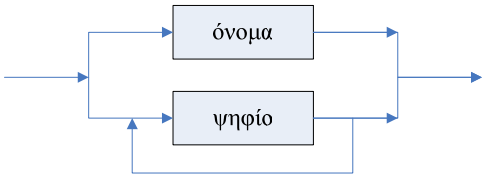
Συντακτικό διάγραμμα 12: ***κλήση διαδικασίας***



Συντακτικό διάγραμμα 13: ***συνθήκη***

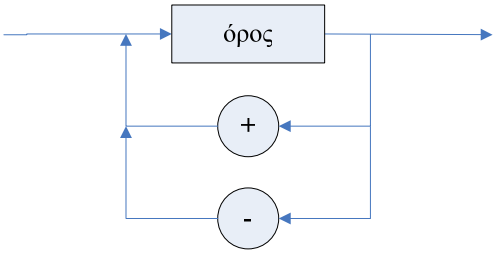


Συντακτικό διάγραμμα 14: ***επανάληψη***

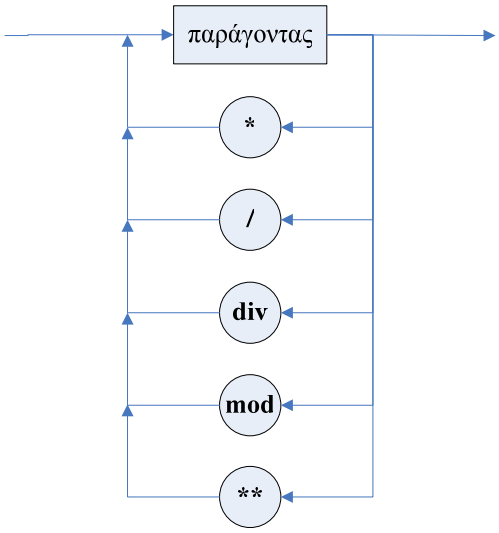


Συντακτικό διάγραμμα 15: ***όριο***

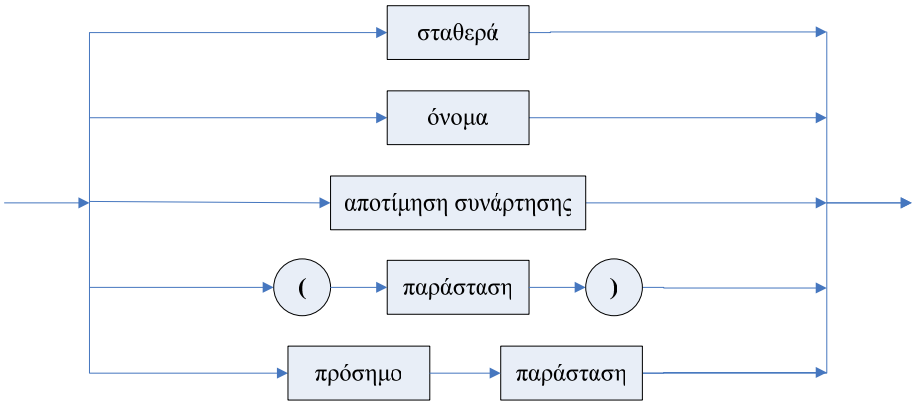
Συντακτικό διάγραμμα 16: ***απλή παράσταση***



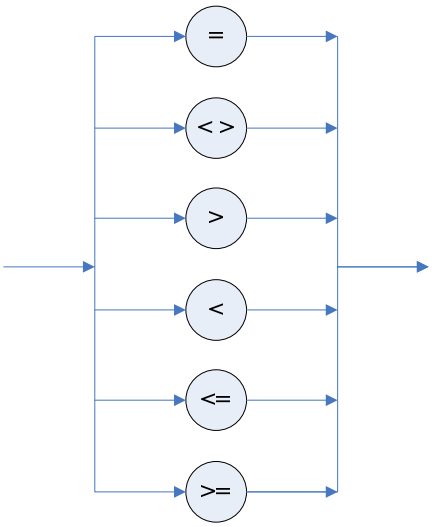
Συντακτικό διάγραμμα 17: ***παράσταση***



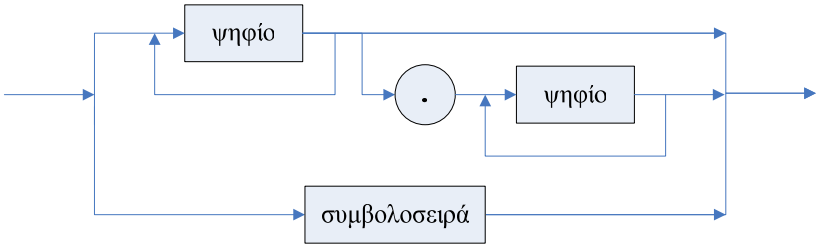
Συντακτικό διάγραμμα 18: ***παράσταση***



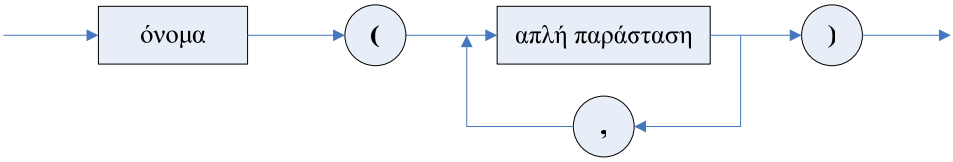
Συντακτικό διάγραμμα 19: ***παράγοντας***



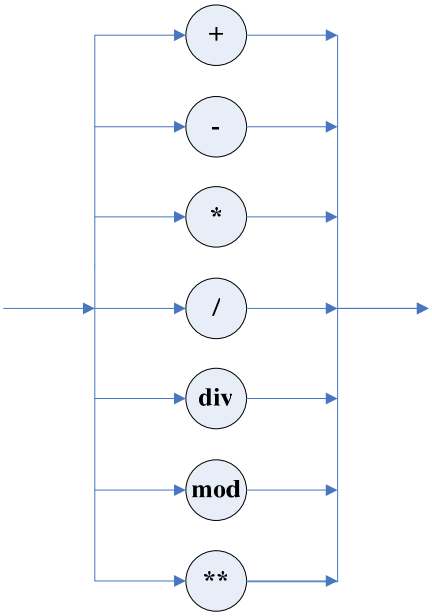
Συντακτικό διάγραμμα 20: ***σύμβολο σύγκρισης***



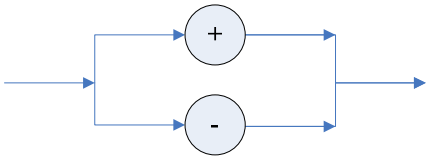
Συντακτικό διάγραμμα 21: ***σταθερά***



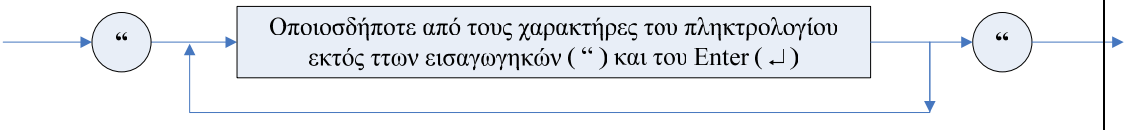
Συντακτικό διάγραμμα 22: ***αποτίμηση συνάρτησης***



Συντακτικό διάγραμμα 23: ***σύμβολο πράξης***



Συντακτικό διάγραμμα 24: ***πρόσημο***



Συντακτικό διάγραμμα 25: ***συμβολοσειρά***