

1. ΒΑΣΙΚΑ ΣΤΟΙΧΕΙΑ

1.1. ΣΥΝΟΛΟ ΧΑΡΑΚΤΗΡΩΝ

Το σύνολο των χαρακτήρων της Turbo Pascal απαρτίζεται από τους αριθμητικούς χαρακτήρες (0-9), τους αλφαριθμητικούς κεφαλαία (A-Z) και πεζά (a-z), καθώς και τους περισσότερους ειδικούς χαρακτήρες.

1.2. ΣΤΑΘΕΡΕΣ

1.2.1. Αριθμητικές σταθερές

Υπάρχουν οι εξής τύποι:

- ⇒ “μικροί” ακέραιοι (short integers) σε 1 byte με τιμές από -128 μέχρι 127
- ⇒ ακέραιοι σε 2 bytes με τιμές από -32768 μέχρι 32767
- ⇒ “μακροί” ακέραιοι (long integers) σε 4 bytes από -2147483648 μέχρι 2147483647
- ⇒ byte σε 1 byte με τιμές από 0 μέχρι 255
- ⇒ word σε 2 bytes με τιμές από 0 μέχρι 65535
- ⇒ πραγματικούς σε 6 bytes με τιμές από $2.9e^{-39}$ μέχρι $1.7e^{38}$ και ακρίβεια 11-12 δεκαδικών ψηφίων
- ⇒ πραγματικούς απλής ακρίβειας (real single precision) σε 4 bytes με τιμές από $1.5e^{-45}$ μέχρι $3.4e^{38}$ και ακρίβεια 7-8 δεκαδικών ψηφίων
- ⇒ πραγματικούς διπλής ακρίβειας (real double precision) σε 8 bytes με τιμές από $5.0e^{-324}$ μέχρι $1.7e^{308}$ και ακρίβεια 15-16 δεκαδικών ψηφίων
- ⇒ εκτεταμένοι πραγματικοί αριθμοί (extended) σε 10 bytes με τιμές από $3.4e^{-4932}$ μέχρι $1.1e^{4932}$ και ακρίβεια 19-20 δεκαδικών ψηφίων
- ⇒ πραγματικοί αριθμοί τύπου comp σε 8 bytes με τιμές από $-9.2e^{18}$ μέχρι $9.2e^{18}$ και ακρίβεια 19-20 δεκαδικών ψηφίων

1.2.2. Αλφαριθμητικές σταθερές

Μια αλφαριθμητική σταθερά είναι μια συμβολοσειρά (string) με μήκος μέχρι 32767 χαρακτήρες περικλειόμενη σε αγκύλες.

1.2.3. Συμβολικές σταθερές

Οι συμβολικές σταθερές ορίζονται με την εντολή CONST και μπορούν να χρησιμοποιηθούν αντί των αριθμητικών ή αλφαριθμητικών σταθερών, π.χ.

```
CONST max_Item = 255;
Matr = array[1..max_Item] of integer;
```

Ορίζεται η συμβολική σταθερά max_Item ίση με 255, η οποία στη συνέχεια χρησιμοποιείται για να ορίσει τη διάσταση ενός πίνακα. Οι συμβολικές σταθερές

μπορεί να είναι οποιουδήποτε τύπου και το όνομά τους σχηματίζεται με βάση τους κανόνες δημιουργίας μεταβλητών της Turbo Pascal.

1.2.4 Λογικές σταθερές.

Οι λογικές σταθερές (Boolean) μπορούν να πάρουν μόνο δύο τιμές TRUE ή FALSE.

1.3. ΜΕΤΑΒΛΗΤΕΣ

Μια μεταβλητή είναι ένα όνομα το οποίο αναφέρεται σε ένα αντικείμενο (έναν αριθμό, μια συμβολοσειρά ή μια εγγραφή).

Το όνομα μιας μεταβλητής σχηματίζεται από αριθμητικούς και αλφαριθμητικούς χαρακτήρες με τον πρώτο υποχρεωτικά αλφαριθμητικό.

Στους αλφαριθμητικούς χαρακτήρες των ονομάτων μεταβλητών δεν έχει σημασία αν είναι κεφαλαία ή πεζά γράμματα. Η Turbo Pascal δεν ξεχωρίζει τα κεφαλαία από τα πεζά γράμματα. Έτσι τα ονόματα LETTER, Letter ή LeTter αναφέρονται στην ίδια θέση μνήμης, πρόκειται δηλαδή για τις ίδιες μεταβλητές. Ο χρήστης μπορεί να χρησιμοποιήσει κεφαλαία ή πεζά γράμματα κατά την κρίση του. Η Turbo Pascal δεν μετατρέπει τα πεζά γράμματα των μεταβλητών σε κεφαλαία και αντίστροφα. Στη συνέχεια, για λόγους καλύτερης αναγνωσιμότητας όλες οι μεταβλητές γράφονται με πεζά. Σε πολλές περιπτώσεις όμως όταν οι μεταβλητές ονοματίζονται, έτσι ώστε το όνομα να προσδιορίζει και το περιεχόμενο, τότε χρησιμοποιούνται και κεφαλαία γράμματα για έμφαση π.χ. MatrixIndex, ArrayPointer, κλπ.

Μια μεταβλητή μπορεί να είναι αριθμητική, αλφαριθμητική ή εγγραφή. Ο προσδιορισμός τους γίνεται δηλώνοντας τον τύπο της μεταβλητής σε δηλωτικές εντολές του τύπου:

Όνομα-μεταβλητής : τύπος

όπου τύπος μπορεί να είναι ένας από τους INTEGER, LONGINT, SINGLE, DOUBLE, REAL, STRING, CHAR, ARRAY, POINTER, RECORD, BOOLEAN ή τύπος δεδομένων του χρήστη.

π.χ. με την εντολή:

x : real;

η μεταβλητή x ορίζεται πραγματική.

Στον ορισμό αλφαριθμητικών μεταβλητών υπάρχουν δύο δυνατότητες, μεταβλητές σταθερού ή μεταβλητού μήκους. Π.χ. με την εντολή:

Mat1 : string;

Ορίζεται μια μεταβλητή με μήκος που μπορεί να εκτείνεται από 0 μέχρι 32767 χαρακτήρες.

Με την εντολή:

Mat1 : string[20];

Ορίζεται μια μεταβλητή με μήκος ακριβώς 20 χαρακτήρες.

Επίσης η Pascal επιτρέπει τη δημιουργία νέων τύπων δεδομένων από το χρήστη. Οι τύποι αυτοί ορίζονται με απλή απαρίθμηση των στοιχείων τους στο τμήμα δήλωσης ορισμού τύπων που τοποθετείται πριν από τη δήλωση των μεταβλητών και ορίζεται από τη δεσμευμένη λέξη TYPE. Στο τμήμα δήλωσης των μεταβλητών **μπορούν να οριστούν** μεταβλητές αυτού του τύπου.

Παράδειγμα.

```
TYPE
    Month={Jan, Feb, Mar, Apr, Mai, Jun,
          Jul, Aug, Sep, Oct, Nov, Dec}
VAR
    A, B:Month
```

Με τον ίδιο τρόπο μπορεί να δηλωθεί μια μεταβλητή εγγραφής, αρκεί προηγούμενα να έχει ορισθεί η εγγραφή με την εντολή TYPE. Π.χ.

```
TYPE
    StockItem = Record
        Code      : String[4];
        Description: String[20];
        UnitPrice  : Single;
        Quantity   : Long;
    End;
VAR
    CurrentItem : StockItem;
```

Μετά τον ορισμό μιας εγγραφής, η αναφορά σε ένα πεδίο της μπορεί να γίνει χρησιμοποιώντας τον τύπο όνομα-εγγραφής.όνομα-πεδίου, π.χ. η μεταβλητή CurrentItem.Code αναφέρεται στο πεδίο κωδικός της εγγραφής με το όνομα CurrentItem.

1.4. ΠΙΝΑΚΕΣ

Ένας πίνακας είναι ένα σύνολο αντικειμένων επί των οποίων η αναφορά γίνεται με το ίδιο όνομα μεταβλητής. Κάθε ένα από τα αντικείμενα που απαρτίζουν τον πίνακα λέγεται στοιχείο του πίνακα. Η αναφορά σε ατομικά στοιχεία γίνεται με το όνομα του πίνακα ακολουθούμενο από έναν ή περισσότερους δείκτες μέσα σε τετραγωνικές αγκύλες. Οι δείκτες προσδιορίζουν την τάξη του στοιχείου μέσα στον πίνακα.

Π.χ. Mat[2,4], Table[j-1]

Σε έναν πίνακα, ο αριθμός των δεικτών προσδιορίζει το πλήθος των διαστάσεων, ενώ η μέγιστη δυνατή τιμή κάθε δείκτη προσδιορίζει το πλήθος των στοιχείων του πίνακα ανά διάσταση. Πριν χρησιμοποιηθεί ένας πίνακας, πρέπει να οριστούν οι διαστάσεις του. Με τον όρο αυτό αναφερόμαστε τόσο στον ορισμό του πλήθους των διαστάσεων, όσο και των μέγιστων τιμών κάθε μιας. Ο ορισμός των διαστάσεων ενός πίνακα επιτυγχάνεται αν δηλώσουμε το όνομα του πίνακα

τύπου ARRAY. Οι διαστάσεις του πίνακα διαχωρίζονται μεταξύ τους με κόμμα (,).

Π.χ. Mat1 : array[1..50] of integer;

Με αυτή την εντολή ορίζουμε έναν πίνακα με όνομα Mat1, που θα καταλαμβάνει 50 θέσεις μνήμης, και τα στοιχεία που θα αποθηκεύονται θα είναι τύπου ακέραιοι (integer).

Να σημειωθεί εδώ η δυνατότητα να ορίζονται με τον τύπο αυτό και αρνητικές τιμές δεικτών

π.χ. array[-5..5] of integer; Έτσι, μπορεί να ορισθεί οποιαδήποτε περιοχή τιμών των δεικτών από -32768 έως 32767.

Ένας πίνακας που μόλις έχει οριστεί, πρέπει στο κυρίως πρόγραμμα να αρχικοποιηθεί (initiallze). Για ονόματα πινάκων μπορούν να χρησιμοποιηθούν οποιοσδήποτε μεταβλητές οποιοσδήποτε τύπου. Ως δείκτες χρησιμοποιούνται ακέραιοι ή ακέραιες εκφράσεις (μπορούν να χρησιμοποιηθούν και μη ακέραιοι αλλά στρογγυλοποιούνται πριν από τη χρήση).

Για τον ορισμό ενός πίνακα με στοιχεία εγγραφής αρχείου πρέπει πρώτα να δηλωθεί ο τύπος της εγγραφής και μετά οι διαστάσεις του πίνακα.

```
TYPE
    QueueNode = Record
        Data : String[20];
        QPtr : Integer;
    End;
VAR
    Qnode : array[1..100] of QueueNode;
```

Κάθε στοιχείο του πίνακα Qnode είναι μια εγγραφή τύπου QueueNode. Η αναφορά σε ένα πεδίο της εγγραφής σαν στοιχείο πίνακα γίνεται με τη χρήση του τύπου όνομα-πίνακα.πεδίο-εγγραφής,

π.χ. Writeln(Qnode.Data);

1.5. ΕΚΦΡΑΣΕΙΣ

Σε ένα πρόγραμμα Pascal μπορεί να συνυπάρχουν σταθερές, λογικοί (boolean) ή αριθμητικοί τελεστές καθώς και συναρτήσεις. Υπάρχουν από τη Γλώσσα οι σημαντικότεροι λογικοί τελεστές που είναι οι AND, OR, NOT και XOR. ενώ οι αριθμητικοί τελεστές είναι + (πρόσθεση), - (αφαίρεση), / (πολλαπλασιασμός), * (διαίρεση), div που είναι η ακέραια διαίρεση και το mod που εξάγει το υπόλοιπο της διαίρεσης. Στη συνέχεια παρατίθεται σε πίνακα η προτεραιότητα τελεστών:

Προτεραιότητα

- 1 (υψηλή)
- 2
- 3
- 4 (χαμηλή)

Τελεστές

- NOT, μοναδιαίοι +, -
 *, /, DIV, MOD, AND
 δυαδικά +, -, OR, XOR
 =, <, >, <=, >=

Οι λογικοί τελεστές εκτελούν λογικές πράξεις στους τελεστέους καθένας από τους οποίους θωρεί-

ται λογικά “αληθής” ή “ψευδής”. Η Turbo Pascal αναγνωρίζει τη διαφορά μεταξύ των δύο παραπάνω κατηγοριών από τα συμφραζόμενα της έκφρασης. Οι λογικοί τελεστές φαίνονται στον παρακάτω πίνακα:

Τελεστής	Σύνταξη	Έννοια
NOT	NOT <έκφραση>	Αρνηση
AND	<έκφραση1> AND <έκφραση2>	Ένωση
OR	<έκφραση1> OR <έκφραση2>	Διάζευξη
XOR	<έκφραση1> XOR <έκφραση2>	Αποκλειστική Διάζευξη

Τελεστές Σύγκρισης

Οι τελεστές σύγκρισης συγκρίνουν δύο εκφράσεις και επιστρέφουν “αληθές” εάν η συνθήκη που ορίζεται από τον τελεστή ικανοποιείται ή “ψευδές” εάν όχι. Οι τελεστές σύγκρισης φαίνονται στον κατωτέρω πίνακα.

Τελεστής	Σύνταξη	Έννοια
=	<έκφραση1> = <έκφραση2>	Οι εκφράσεις είναι ίσες
<>	<έκφραση1> <> <έκφραση2>	Οι εκφράσεις δεν είναι ίσες
<	<έκφραση1> < <έκφραση2>	Η <έκφραση1> είναι μικρότερη από την <έκφραση2>
<=	<έκφραση1> <= <έκφραση2>	Η <έκφραση1> είναι μικρότερη ή ίση από την <έκφραση2>
>	<έκφραση1> > <έκφραση2>	Η <έκφραση1> είναι μεγαλύτερη από την <έκφραση2>
>=	<έκφραση1> >= <έκφραση2>	Η <έκφραση1> είναι μεγαλύτερη ή ίση από την <έκφραση2>

Η Pascal διαθέτει επίσης ένα πλούσιο ρεπερτόριο ενσωματωμένων μαθηματικών συναρτήσεων οι σπουδαιότερες των οποίων είναι:

Συνάρτηση	Λειτουργία
ABS	Εξάγει το απόλυτο μέρος ενός αριθμού
INT	Εξάγει το ακέραιο μέρος ενός δεκαδικού αριθμού
SQR	Υψώνει έναν αριθμό στο τετράγωνο
SQRT	Βρίσκει την τετραγωνική ρίζα
SIN	Βρίσκει το ημίτονο μιας γωνίας
COS	Βρίσκει το συνημίτονο μιας γωνίας
ARC	Βρίσκει το τόξο μιας γωνίας
ARCTAN	Βρίσκει το τόξο εφαπτομένης μιας γωνίας
LN	Βρίσκει τον λογάριθμο ενός αριθμού

1.6. ΔΟΜΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

Η γενική μορφή ενός προγράμματος Pascal είναι η εξής:

```
Program Ìíííá_ðñíãñÛííáôíð;
(Ãçëððáëð ððáëãñí)
(Ãçëððáëð ðçðíð ìáðáãëçðñí)
(Ãçëððáëð ìáðáãëçðñí)

BEGIN
(Ãíðíëÿð)

END.
```

Το όνομα του προγράμματος μπορεί να είναι ένα οποιοδήποτε αλφαριθμητικό όνομα. Μετά το όνομα του προγράμματος πρέπει να προστίθεται πάντα ένα ελληνικό ερωτηματικό. Το ελληνικό ερωτηματικό (;) στην Pascal γενικότερα παίζει μεγάλο ρόλο και η ύπαρξή του είναι καθοριστική. Υποδηλώνει το τέλος μιας εντολής ή δήλωσης και την αρχή μιας νέας. Η τελευταία εντολή του προγράμματος πρέπει να είναι η END και ακολουθείται υποχρεωτικά από την τελεία ‘.’.

Στην περιοχή δηλώσεων δεδομένων, καταγράφονται όλες οι σταθερές, οι τύποι δεδομένων και οι μεταβλητές που θα χρησιμοποιηθούν από το πρόγραμμα. Φυσικά δεν είναι υποχρεωτικό να υπάρχουν όλα αυτά τα τμήματα, για παράδειγμα αν το πρόγραμμα δεν χρησιμοποιεί σταθερές το αντίστοιχο τμήμα θα απουσιάζει.

Οι σταθερές δηλώνονται κάτω από την ετικέτα CONST με το όνομά τους και την τιμή που έχουν. π.χ. CONST Max = 100;

Στο παράδειγμα αυτό προσδιορίζεται μια σταθερά με όνομα Max που έχει την τιμή 100. Την τιμή αυτή θα την έχει έως το πέρας του προγράμματος. Το περιεχόμενο των σταθερών δεν επιτρέπεται αλλαχθεί μέσα στο πρόγραμμα.

Οι τύποι που θα χρησιμοποιούνται μέσα σε ένα πρόγραμμα ορίζονται στην περιοχή TYPE όνομα_τύπου = Τύπος_Δεδομένου;

π.χ. TYPE MyArray = array[1..10] of integer;

Στο παράδειγμα αυτό ορίζεται ένας τύπος πίνακα MyArray που θα καταλαμβάνει 10 θέσεις στη μνήμη του υπολογιστή και η κάθε θέση θα έχει έναν ακέραιο αριθμό. (Βλέπε πίνακες). Οι μεταβλητές του προγράμματος ορίζονται στην περιοχή

```
VAR ùíííá_ìáðáãëçðñð : ðçðíð_ãããíÿííð;
π.χ. VAR number : integer;
```

Στο παράδειγμα αυτό ορίστηκε η μεταβλητή number που είναι τύπου ακεραίου.

```
VAR Arr1 : MyArray;
```

Στο παράδειγμα αυτό ορίστηκε η μεταβλητή πίνακα Arr1, σαν τύπου MyArray. Ο τύπος αυτός δεν είναι τύπος της Turbo Pascal αλλά έχει ορισθεί στο προηγούμενο παράδειγμα που αφορούσε την περιοχή δηλώσεων TYPE. Έτσι, δίνεται η δυνατότητα ορισμού τύπων δεδομένων του χρήστη και μέσω της περιοχής δηλώσεων VAR ορίζεται μια μεταβλητή του αυτού τύπου και η οποία θα χρησιμοποιηθεί τελικά μέσα στο πρόγραμμα. Έτσι, το παράδειγμα συνολικά θα δείχνει ως εξής:

TYPE

MyArray = array[1..10] of integer;

VAR

Arr1 : MyArray;

Μετά τις δηλώσεις δεδομένων ακολουθεί ο ορισμός τυχόν διαδικασιών ή συναρτήσεων που υπάρχουν στο πρόγραμμα. Αναφορά στις διαδικασίες και τις συναρτήσεις θα γίνει στη συνέχεια.

Ένα πρόγραμμα αποτελείται από γραμμές προγράμματος. Κάθε γραμμή προγράμματος περιέχει μια ή περισσότερες εντολές προγράμματος. Αν υπάρχουν περισσότερες από μια εντολές σε μία γραμμή, τότε πρέπει να χωρίζονται με το χαρακτήρα ελληνικό ερωτηματικό (;). Μια γραμμή προγράμματος μπορεί να εκτείνεται σε περισσότερες από μία φυσικές γραμμές.

2. ΕΚΧΩΡΗΣΗ ΤΙΜΩΝ, ΕΙΣΟΔΟΣ-ΕΞΟΔΟΣ

Εκχώρηση τιμών

Η πιο θεμελιώδης δήλωση στην Pascal είναι η δήλωση εκχώρησης τιμής η οποία εκχωρεί μια τιμή σε μια μεταβλητή. Η ταυτότητα (όνομα) της μεταβλητής που πρόκειται να αλλάξει δηλώνεται ακολουθούμενη από τον τελεστή εκχώρησης (:=) και την νέα τιμή.

π.χ. `quantity := 40;`
`message := 'File not found';`
`answer := Y;`

Στην πρώτη γραμμή εκχωρείται μια ακέραια τιμή στην μεταβλητή `quantity`, στην δεύτερη γραμμή εκχωρείται ένα αλφαριθμητικό στην μεταβλητή `message` ενώ στην Τρίτη μεταβλητή εκχωρείται το περιεχόμενο της μεταβλητής `Y` στην μεταβλητή `answer`.

Επίσης αντί για συγκεκριμένες τιμές, σε μια μεταβλητή μπορεί να εκχωρείται το αποτέλεσμα μιας έκφρασης ή συνάρτησης.

π.χ. `x := (3 * Y) / 6;`
`z := Sqrt (Sqr (x) + Sqr (x));`

Είσοδος

Η είσοδος τιμών ή δεδομένων γίνεται με τις εντολές `readln` και `read`. Η σύνταξή τους είναι: `readln(όνομα_μεταβλητής);`

Εφόσον δεν ορίζεται συσκευή εισόδου θεωρείται ότι η είσοδος γίνεται από το πληκτρολόγιο.

Η διαφορά της `readln` από την απλή `read` είναι ότι στην πρώτη περίπτωση η είσοδος γίνεται στην αρχή μιας νέας σειράς σε αντίθεση με την `read`, στην οποία η είσοδος γίνεται στο σημείο που βρίσκεται την τρέχουσα στιγμή ο δρομέας. π.χ. `readln(var1);`

Έξοδος

Αντίστοιχα, η έξοδος δεδομένων γίνεται με τις εντολές `writeln` και `write`. Η γενική της σύνταξη είναι:

`writeln(όνομα_μεταβλητής);`

Η έξοδος αν δεν οριστεί κάποια άλλη συσκευή εξόδου γίνεται στην οθόνη.

Και σε αυτή την περίπτωση, η διαφορά μεταξύ των δύο εντολών είναι ότι στη `writeln`, η έξοδος των δεδομένων πραγματοποιείται σε μια νέα γραμμή ενώ στην `write`, η έξοδος γίνεται στην τρέχουσα θέση του δρομέα.

π.χ. `writeln(var1);` εμφάνιση του περιεχομένου της μεταβλητής `var1` στην οθόνη
`write('Turbo Pascal');` εμφάνιση του λεκτικού "Turbo Pascal" στην οθόνη.

Για να καθοριστεί ο ακριβής τρόπος με τον οποίο θα εμφανίζονται τα δεδομένα χρησιμοποιείται η παράμετρος πλάτους πεδίου. Η τιμή του πλάτους πεδίου ορίζεται μετά τον χαρακτήρα άνω και κάτω τελεία ':'. Η τιμή κάθε παράστασης τυπώνεται σε τόσες θέσεις όσες είναι η τιμή του πλάτους του πεδίου. Για παράδειγμα η εντολή `write A:5`, θα τυπώσει το περιεχόμενο της μεταβλητής `A` σε 5 θέσεις.

Αν η μεταβλητή `A` περιέχει τον αριθμό 10 τότε το αποτέλεσμα της εντολής `write A:5` είναι να αφήσει τρία κενά και μετά να εκτυπώσει τον αριθμό 10.

Σε περίπτωση πραγματικών αριθμών χρησιμοποιούνται δύο τιμές, η πρώτη καθορίζει το συνολικό πλάτος του αριθμού ενώ ο δεύτερος τα δεκαδικά στοιχεία που θα εκτυπωθούν.

Αν το `A` έχει την τιμή 3.14156 τότε η εντολή `write A:4:2` θα εκτυπώσει 3.14

3. ΔΟΜΕΣ ΕΛΕΓΧΟΥ

Οι δομές ελέγχου είναι ένα εργαλείο που προσφέρουν οι διαδικαστικές γλώσσες προγραμματισμού για τον έλεγχο της σειράς εκτέλεσης των εντολών. Οι δομές ελέγχου είναι βασικά τρεις: η ακολουθία (sequence), η επιλογή (selection) και η ανακύκλωση ή βρόχος (loop). Οι δύο τελευταίες έχουν διάφορες παραλλαγές.

3.1. ΑΚΟΛΟΥΘΙΑ

Πρόκειται για την απλούστερη δομή κατά την οποία οι εντολές του προγράμματος εκτελούνται η μία μετά την άλλη, με τη σειρά που γράφονται στο πρόγραμμα, δηλ. Από την αρχή του κειμένου προς το τέλος και στην ίδια γραμμή από αριστερά προς τα δεξιά. Μεταξύ δύο συνεχόμενων εντολών που βρίσκονται στην ίδια γραμμή απαιτείται η παρουσία ενός διαχωριστή εντολών, που εδώ είναι ο χαρακτήρας ελληνικό ερωτηματικό ';':

3.2. ΕΠΙΛΟΓΗ

3.2.1. Επιλογή

Κατά την εκτέλεση ενός προγράμματος είναι δυνατό να επιλεγεί η εκτέλεση ορισμένων εντολών και να αποφευχθεί η εκτέλεση άλλων. Η πιο απλή δομή για κάτι τέτοιο είναι η IF..THEN..ELSE. Η σύνταξη της εντολής είναι:

```
IF B THEN E1 ELSE E2
```

Η B είναι μια λογική παράσταση (μια συνθήκη) και η οποία έχει τιμές: αληθής (true) αν η συνθήκη ισχύει και ψευδής (false) αν η συνθήκη δεν ισχύει. Οι E1 και E2 είναι εντολές ή ομάδες εντολών. Η λειτουργία της εντολής είναι η εξής: υπολογίζεται πρώτα η λογική τιμή της παράστασης B. Αν η τιμή της B είναι αληθής τότε εκτελείται η εντολή E1, αλλιώς εκτελείται η E2.

Οι εντολές E1 και E2 μπορούν να είναι ατομικές εντολές ή σύνολο εντολών χωρισμένες με ";". Όλες σχεδόν οι εντολές της γλώσσας μπορούν να υπάρχουν μέσα στις E1 και E2, ακόμη και εντολές IF..THEN..ELSE, οπότε δημιουργούνται τα λεγόμενα εμφωλευμένα (nested) IF. Επίσης είναι δυνατόν το τμήμα ELSE να απουσιάζει, οπότε αν η συνθήκη δεν ισχύει, η εκτέλεση του προγράμματος συνεχίζεται με την εντολή που ακολουθεί την IF..THEN.

Παράδειγμα. Ένας άνδρας χαρακτηρίζεται ελαφρύς (βαρύς) αν είναι κάτω (πάνω) από 80 εκ. Επίσης χαρακτηρίζεται κοντός (ψηλός) αν είναι κάτω (πάνω) από 1.75 εκ. Στο επόμενο τμήμα προγράμματος εισάγονται το βάρος και το ύψος ενός ατόμου και εξάγεται ο χαρακτηρισμός του με σύνθετο IF.

```
Write('Άεόϑάδâ ôï âϑñïð : '); readln(b);
Write('Άεόϑάδâ ôï ýðïð : '); readln(y);
If (b<80) then
  if (y<1.75) then
    writeln('Êïíðüð êáé âéáðñýð')
  Else
    writeln('øçëüð êáé âéáðñýð')
Else
  if (y<75) then
    writeln('êïíðüð êáé âáñýð')
  Else
    writeln('øçëüð êáé âáñýð');
```

Ένας άλλος τύπος εντολής IF είναι η εντολή IF.. ELSE IF..ELSE, με την οποία παρέχεται μεγαλύτερη ευελιξία στη σύνταξη πολύπλοκων δομών.

```
Writeln('Άρôâ Ýíáí ðáñáêðñâ : ');
Readln(a);
If (a>'0') and (a<='9') then
  writeln('øçðñï')
Else if (a>='A') and (a<='z') then
  writeln('Ëáðéíéëü ãññïíá')
Else if (a>='Á') and (a<='ù') then
  writeln('Ëðéçíéëü ãññïíá')
Else
  writeln('ìç áéðáñéëìçðéëüð ðáñáêðñâð');
```

Η Turbo Pascal εξετάζει κάθε μια από τις συνθήκες της IF και των ELSE IF δηλώσεων από πάνω προς τα κάτω υπερπηδώντας τις ομάδες των εντολών που ακολουθούν μέχρι να βρεθεί η πρώτη αληθής συνθήκη. Τότε εκτελούνται οι εντολές που αντιστοιχούν και μετά γίνεται διακλάδωση στην εντολή που ακολουθεί το μετά το τέλος της IF.

3.2.2. Πολλαπλή επιλογή

Με τον όρο αυτό αναφερόμαστε στη δυνατότητα μιας δομής να επιτυγχάνει την εκτέλεση μιας ομάδας εντολών ανάμεσα σε πολλές ή τη διακλάδωση του προγράμματος σε περισσότερα από ένα σημεία του προγράμματος ανάλογα με την τιμή της εξεταζόμενης συνθήκης, η σχετική εντολή είναι η CASE..END. Πρόκειται για εντολή πολλαπλών επιλογών απόφασης ανάλογη, από πλευράς τελικού αποτελέσματος, με το ομαδοποιημένο IF..THEN..ELSE. Η βασική διαφορά των δύο εντολών είναι ότι η πρώτη εξετάζει μια απλή έκφραση και ανάλογα με το αποτέλεσμα εκτελεί διαφορετικές εντολές ή προκαλεί διακλάδωση σε διαφορετικά σημεία, ενώ η δεύτερη εξετάζει απλές ή σύνθετες λογικές εκφράσεις. Αν και η λειτουργία της CASE..END μπορεί να πραγματοποιηθεί και με την IF..THEN..ELSE, λόγω της συμπαγούς δομής της η χρήση της προσφέρει σημαντικά πλεονεκτήματα στον προγραμματιστή.

Η γενική μορφή της εντολής είναι:

```
CASE üíííâ_íàðáâçðð OF
  Êðððâ_ðéìðí_1 : áíðíëýð1;
  Êðððâ_ðéìðí_2 : áíðíëýð2;
  .....
ELSE
  Áíðíëýð_í;
END;
```

Το όνομα μεταβλητής μπορεί να είναι αριθμητική ή αλφαριθμητική. Οι λίστες τιμών μπορούν να περιλαμβάνουν μία ή περισσότερες τιμές, περιοχή τιμών από-έως, ή τιμές που υπακούουν σε μια λογική συνθήκη. Η εντολή εξετάζει την έκφραση και ανάλογα με την τιμή της εκτελεί τις εντολές που βρίσκονται μετά το διαχωριστικό ":". Αν η τιμή της μεταβλητής δεν αντιστοιχεί σε καμία από τις λίστες τιμών, τότε εκτελούνται οι εντολές που ακολουθούν την ELSE. Μετά την εκτέλεση των εντολών κάποιος CASE, η ροή του προγράμματος συνεχίζεται μετά το END της εντολής.

Η λειτουργία της εντολής καθώς και οι δυνατότητές της φαίνονται στα επόμενα παραδείγματα:

Παράδειγμα 1.

```
Writeln('Άρôâ Ýíáí áéýñáéí áðü 1 üð 3 : ');
Readln(number);
CASE number OF
  1 : writeln('Ï áñéëìüð áßíáé 1');
  2 : writeln('Ï áñéëìüð áßíáé 2');
  3 : writeln('Ï áñéëìüð áßíáé 3');
ELSE
  Writeln('ËÜëìð áéðáâüð');
END;
```

Παράδειγμα 2.

```

Writeln('Άρσά Υίαί άέΰñάέί άόü 1 üð 20: ');
Readln(number);
CASE number OF
  1..10 : writeln('1ç ääëÛää');
  11..20 : writeln('2ç ääëÛää');
ELSE
  Writeln('ËÛëið áέόääüäÞ');
END;

```

Υπάρχει επιπλέον η δυνατότητα ορισμού περιοχής τιμών από-έως για τις λίστες τιμών. Η περιοχή τιμών δίνεται με το συνδυαστικό δύο τελειών “..”, ανάμεσα από την αρχική και την τελική τιμή.

3.3. ΑΝΑΚΥΚΛΩΣΗ

Η δομή της ανακύκλωσης ή του βρόχου αναφέρεται στην επαναληπτική εκτέλεση μιας ομάδας εντολών προγράμματος κατά ένα προκαθορισμένο αριθμό φορών ή μέχρι να ισχύσει κάποια συνθήκη. Η δομή της ανακύκλωσης υλοποιείται με εντολές FOR..DO, WHILE..DO, REPEAT..UNTIL.

3.3.1. FOR..DO

Η εντολή αυτή εκτελεί τις εντολές προγράμματος που βρίσκονται μετά το FOR και ανάμεσα στο BEGIN..END που ακολουθεί και εκτελείται για όλες τις τιμές της μεταβλητής ελέγχου. Η μεταβλητή ελέγχου που αναφέρεται στη FOR, συνοδεύεται από την αρχική και τελική τιμή και σε κάθε επανάληψη αυξάνεται κατά μία μονάδα. Στο επόμενο τμήμα προγράμματος χρησιμοποιείται η εντολή FOR..DO για να διατρέξει διαδοχικά όλα τα στοιχεία του πίνακα K, και αν κάποιο στοιχείο είναι ίσο με τον αριθμό τότε τυπώνεται η θέση του στοιχείου.

```

Write('ΆέόÛääðä ðíí êüääéü :'); readln(le);
Index:=0; ex:=0;
For i:=1 to 50 do
  If cle=k[i] then
    Writeln (I);

```

Υπάρχει και μια άλλη έκδοση αυτής της εντολής η FOR..DOWNT0..DO. Εδώ η μεταβλητή ελέγχου που συνοδεύει το βρόχο περιέχει αρχική τιμή μεγαλύτερη από την τελική τιμή του βρόχου. Έτσι, η επανάληψη γίνεται από την μεγαλύτερη στην μικρότερη τιμή. Σε κάθε επανάληψη η μεταβλητή μειώνεται κατά μία μονάδα.

3.3.2. WHILE..DO

Με την εντολή αυτή εκτελούνται όλες οι εντολές προγράμματος που υπάρχουν κάτω από την WHILE και ανάμεσα στο BEGIN..END, όσο ισχύει η συνθήκη που συνοδεύει το WHILE. Το επόμενο τμήμα προγράμ-

ματος κάνει ότι και το προηγούμενο αλλά πλέον η επαναληπτική εντολή είναι η WHILE..DO. Αν κατά την εξέταση όλων των στοιχείων ενός πίνακα k βρεθεί κάποιο στοιχείο που να ισούται με τη τιμή της μεταβλητής cle που έδωσε ο χρήστης από το πληκτρολόγιο, τυπώνεται η θέση του πίνακα.

```

Write('Άρσά ðíí êüääéü :'); readln(cle);
i:=1;
While i<=50 do
Begin
  If cle=k[i] then
    Writeln (I);
    i:=i+1;
End;

```

3.3.3. REPEAT..UNTIL

Παρόμοια στη λογική με τη WHILE..DO είναι και η REPEAT..UNTIL. Με την εντολή αυτή θα εκτελεστούν σίγουρα μία φορά όλες οι εντολές που περιλαμβάνονται μέσα στο REPEAT..UNTIL και αυτό γιατί η συνθήκη ελέγχου βρίσκεται στο τέλος του βρόχου και όχι στην αρχή (περίπτωση WHILE..DO). Έως ότου η συνθήκη αυτή γίνει ψευδής, η εκτέλεση των εντολών θα επαναλαμβάνεται συνεχώς..

Σε περίπτωση που δεν υπάρχει καθόλου συνθήκη ελέγχου, ο βρόχος θα εκτελείται συνεχώς.

Το επόμενο τμήμα προγράμματος κάνει ότι και το προηγούμενο αλλά πλέον η επαναληπτική εντολή είναι η REPEAT.. UNTIL. Αν κατά την εξέταση όλων των στοιχείων ενός πίνακα k βρεθεί κάποιο στοιχείο που να ισούται με τη τιμή της μεταβλητής cle που έδωσε ο χρήστης από το πληκτρολόγιο, τυπώνεται η θέση του πίνακα.

```

Write('Άρσά ðíí êüääéü :'); readln(cle);
i:=1;
repeat
  If cle=k[i] then
    Writeln (I);
    i:=i+1;
until I>50

```

Μία συχνή χρήση της εντολής REPEAT...UNTIL παρουσιάζεται στο επόμενο τμήμα προγράμματος

```

Write('ΆίðÛîää ; (Í/Ī) : ');
REPEAT
  Readln(a);
UNTIL (UPCASE(a) = 'Í')OR (UPCASE(a)='Ó');

```

Σε αυτό το παράδειγμα ζητείται η εισαγωγή των χαρακτήρων N για Ναι και O για Όχι. Εστω κι αν δοθεί αντίστοιχος πεζός χαρακτήρας η συνάρτηση UPCASE αναλαμβάνει να τον μετατρέψει σε κεφαλαίο. Μόνο όταν δοθεί N (Ναι) ή O (Όχι) η ροή του προγράμματος θα συνεχίσει κανονικά. Σε αντίθετη περίπτωση ο βρόχος επαναλαμβάνεται.

Το όνομα της διαδικασίας πρέπει να είναι μοναδικό σε όλο το πρόγραμμα και δεν πρέπει να εμφανίζεται σε άλλο σημείο του προγράμματος.

Η Λίστα_Τυπικών_Παραμέτρων περιέχει σταθερές, μεταβλητές και πίνακες ή γενικότερα τύπους δεδομένων που υποστηρίζονται από την Turbo Pascal. Οι παράμετροι χωρίζονται με κόμμα. Επίσης, η κάθε μεταβλητή που ορίζεται για πρώτη φορά στην Λίστα_Τυπικών_Παραμέτρων πρέπει να δηλώνεται και ο τύπος της.

Π.χ. PROCEDURE Display_Name(Title : String, TLng : Integer);

Στην περιοχή CONST, ορίζονται οι τυχόν σταθερές που μπορεί να χρησιμοποιούνται μέσα στη διαδικασία. Στην περιοχή VAR, ορίζονται οι τοπικές μεταβλητές της διαδικασίας και οι οποίες είναι ορατές μόνο μέσα στη διαδικασία αυτή. Ανάμεσα στο BEGIN και το END, μπορεί να υπάρχει οποιοσδήποτε αριθμός εντολών. Μέσα σε μια διαδικασία δεν μπορεί να έχει οριστεί μια άλλη διαδικασία. Η κάθε διαδικασία είναι αυτόνομη και διατηρεί την ακεραιότητά της.

Παράδειγμα. Η επόμενη διαδικασία πραγματοποιεί τη λύση των συντελεστών a και b και επιστρέφει ως αποτέλεσμα την τιμή του x εφόσον η εξίσωση έχει λύση, καθώς και την τιμή μιας μεταβλητής flag, η οποία είναι 1 αν υπάρχει λύση, διαφορετικά είναι 0. Η λειτουργία του Var πριν από τις παραμέτρους θα εξηγηθεί στη συνέχεια.

```
PROCEDURE EqSolve1(VAR a,b,x,flag :
Integer);
BEGIN
    flag:=1;x:=0;
    IF a=0 THEN flag=0
    ELSE x:=-b/a;
END;
```

Μια διαδικασία μπορεί να κληθεί από το κυρίως πρόγραμμα ή από άλλες διαδικασίες με το όνομά της.

Γενική κλήση μιας διαδικασίας είναι:

Όνομα_Διαδικασίας (Λίστα_Πραγματικών_Παραμέτρων);

π.χ. EqSolve1(c, d, z, myflag);

Όταν καλείται μια διαδικασία, οι πραγματικές παράμετροι που στέλνονται σε αυτήν τοποθετούνται σε σειρά σε μια ειδική περιοχή στην μνήμη που καλείται σωρός (stack). Η διαδικασία και οι συναρτήσεις διαβάζουν τα δεδομένα αυτά με την σειρά που εξάγονται από τον σωρό.

Ο αριθμός και ο τύπος των ορισμάτων κατά την κλήση μιας διαδικασίας πρέπει να είναι ίδιος με τον αριθμό και τον τύπο της λίστας παραμέτρων της διαδικασίας. Οι παράμετροι και τα ορίσματα πρέπει να δίνονται με την ίδια σειρά και χωρίζονται με κόμμα.

Παράδειγμα.

```
TYPE
    ArrType : array[1..10] of Integer;

PROCEDURE VectorSum( Var ArrName : ArrType;
                    n,s : Integer );

BEGIN
    s:=0;
    FOR i:=1 to n DO
        s := s + ArrName[i];
    END;
```

Σε μια διαδικασία ο ορισμός ενός πίνακα στην λίστα τυπικών παραμέτρων γίνεται με τον ακόλουθο τρόπο:

PROCEDURE όνομα_Διαδικασίας(Όνομα_Πίνακα : Τύπος_Πίνακα);

π.χ. PROCEDURE VectorSum(VAR ArrName : ArrType);

Αν σε μια διαδικασία δεν απαιτείται να περαστεί ως παράμετρος ολόκληρος ο πίνακας, μπορεί να μεταβιβάστούν μόνο τα στοιχεία του πίνακα που χρειάζονται. Για να μεταβιβάστεί ένα μόνο στοιχείο του πίνακα αρκεί να προστεθούν και οι τιμές των δεικτών του στοιχείου μέσα στις πραγματικές παραμέτρους κλήσης της διαδικασίας.

π.χ. Vector(ArrName[3]);

Εγγραφές και πεδία. Σε μια διαδικασία μπορούν να μεταβιβάστούν και ολόκληρες εγγραφές ή ατομικά πεδία εγγραφής. Στο επόμενο παράδειγμα αφού οριστεί ο τύπος της εγγραφής και δηλωθεί μια μεταβλητή εγγραφής μεταβιβάζεται ολόκληρο το περιεχόμενό της στη διαδικασία PrintRec.

```
TYPE StockElem = RECORD
    Code : String[4];
    Description : String[20];
    UnitPrice : Single;
    Quantity : Integer;
END;

VAR
    StockRec : StockElem;

PROCEDURE PrintRec(VAR RecVar : StockElem);
BEGIN
    .....
END;

BEGIN
    .....
    PrintRec(StockRec);
    .....
END.
```

Ατομικά πεδία μπορούν να μεταβιβάστούν αν στη λίστα πραγματικών παραμέτρων αναφερθούν τα ονόματά τους. Στο ίδιο παράδειγμα μεταβιβάζεται στη διαδικασία SearchKey μόνο το πεδίο Code.

```
PROCEDURE SearchKey(VAR CodVar:String[4]);
```

```

BEGIN
    .....
END;

BEGIN
    .....
    SearchKey(StockElem.Code);
    .....
END.

```

Το StockElem είναι η μεταβλητή εγγραφής που χρησιμοποιείται για την διαχείριση των εγγραφών. Κάθε φορά που γίνεται μια αναφορά στα πεδία της εγγραφής StockElem, πρέπει να γίνεται ως εξής:

StockElem.Code, για τον κωδικό,

StockElem.Description για την περιγραφή, κ.ο.κ.

Η Turbo Pascal περιλαμβάνει μια εντολή με την οποία ο χρήστης διευκολύνεται αρκετά κάθε φορά που απαιτείται αναφορά σε εγγραφές. Η εντολή αυτή είναι WITH Μεταβλητή_Εγγραφής DO. Οι εντολές περιλαμβάνονται σε BEGIN και END και ολοκληρώνεται με το ελληνικό ερωτηματικό π.χ.

```

WITH StockElem DO
BEGIN
    Readln(Code);
    Readln(Description);
    Readln(UnitPrice);
    Readln(Quantity);
END;

```

Οι παράμετροι στη Pascal μπορούν να περάσουν με δύο τρόπους με τιμή ή με αναφορά.

Πέρασμα παραμέτρων με αναφορά

Οι παράμετροι που περνούν με αναφορά έχουν υποχρεωτικά πριν από τη δήλωσή τους στη λίστα των τυπικών παραμέτρων τη λέξη VAR. Μία παράμετρος η οποία περνάει με αναφορά σημαίνει απλά ότι θα επιστρέψει τη νέα της τιμή μετά το τέλος της διαδικασίας.

Οι πραγματικές παράμετροι στον ορισμό μιας διαδικασίας ή συνάρτησης περνούν με αναφορά όταν πρέπει τα δεδομένα να προστελαστούν απευθείας. Αυτό σημαίνει ότι αντί να χρησιμοποιείται η στοιβή για την αποθήκευση των πραγματικών παραμέτρων, η Turbo Pascal αποθηκεύει έναν δείκτη που δείχνει στη διεύθυνση μνήμης όπου βρίσκεται η πραγματική μεταβλητή. Καθώς η διαδικασία εκτελείται, κάθε αναφορά στις τυπικές παραμέτρους, περνιούνται μέσω του δείκτη στις πραγματικές.

Έτσι, όταν αλλάζει το περιεχόμενο μιας τυπικής παραμέτρου στην ουσία αλλάζει και της πραγματικής.

Πέρασμα παραμέτρων με τιμή

Σε αυτή την περίπτωση, οι τιμές των πραγματικών παραμέτρων μιας διαδικασίας ή συνάρτησης, αντιγράφονται στη στοιβή. Οι τυπικές παράμετροι χρησιμοποιούν απλά ένα αντίγραφο της τιμής που βρίσκεται

στη στοιβή. Οι αλλαγές στις τυπικές παραμέτρους δεν αλλάζουν την τιμή των πραγματικών παραμέτρων.

Τα δύο είδη περάσματος των παραμέτρων φαίνονται στο παρακάτω τμήμα προγράμματος όπου η μεταβλητή A περνάει με αναφορά ενώ η B με τιμή. Έτσι η μεταβλητή D όταν μεταβάλλεται από τη διαδικασία test αλλάζει και την τιμή της μεταβλητής B ενώ η μεταβλητή A μένει αναλλοίωτη.

```

.....
Procedure test(C:integer, Var D:Integer);
Begin
    C:=C*C;
    D:=D*D;
End;

.....
A:=10;
B:=20;
test (A,B)
Write (A,B)
....

```

Το πρόγραμμα θα τυπώσει τους αριθμούς: 10 400

4.2. ΣΥΝΑΡΤΗΣΕΙΣ

Μια συνάρτηση είναι μια διαδικασία που επιστρέφει μια τιμή με το όνομά της. Η συνάρτηση διαφέρει από τη διαδικασία γιατί ενώ μπορεί να δεχθεί πολλές παραμέτρους, επιστρέφει πάντα μια μόνο τιμή. Οι συναρτήσεις μπορούν να εκτελούν οποιαδήποτε λειτουργία επιθυμεί ο χρήστης γι' αυτό και καλούνται συναρτήσεις ορισμένες από τον χρήστη (user-defined functions) σε αντιδιαστολή με τις ενσωματωμένες συναρτήσεις της γλώσσας. Για να ορισθεί μια συνάρτηση χρησιμοποιείται η εντολή FUNCTION.

Μία συνάρτηση όπως έχει ήδη αναφερθεί, μπορεί να κληθεί μέσα από το κυρίως πρόγραμμα, άλλες διαδικασίες ή συναρτήσεις του προγράμματος. Οι συναρτήσεις όπως και οι διαδικασίες στην Turbo Pascal έχουν την δυνατότητα της αναδρομικής κλήσης τους (recursive call) - η ιδιότητα της διαδικασίας ή της συνάρτησης να καλεί τον εαυτό της μέσα από την ίδια.

Για το όνομα μιας συνάρτησης ισχύουν ότι και για τις διαδικασίες. Στη συνάρτηση υπάρχει η δυνατότητα όπως περιγράφεται παρακάτω, χρησιμοποίησης τοπικών αλλά και καθολικών μεταβλητών.

4.2.2. Ορισμός Συναρτήσεων

Μια συνάρτηση ορίζεται με την εντολή FUNCTION. Η σύνταξη της εντολής είναι:

FUNCTION όνομα_συνάρτησης (Λίστα_Τυπικών_Παραμέτρων) : Τύπος_Δεδομένου

Το όνομα_συνάρτησης δεν μπορεί να αποδοθεί σε άλλη διαδικασία ή συνάρτηση. Για τη λίστα τυπικών παραμέτρων ισχύουν όσα αναφέρθηκαν για τις διαδικασίες. Μετά τη λίστα τυπικών παραμέτρων ακολουθεί η άνω και κάτω τελεία και συνεχίζεται ο προσδιορι-

σμός του τύπου δεδομένου που θα επιστρέψει η συνάρτηση. Άρα, η κάθε συνάρτηση επιστρέφει πάντοτε μια τιμή. Έτσι, η κλήση της συνάρτησης δεν γίνεται μόνο καλώντας το όνομά της, αλλά εκχωρώντας την σε μια μεταβλητή ίδιου τύπου με αυτόν που θα εξάγει η συνάρτηση.

π.χ. αν υφίσταται μια συνάρτηση `FUNCTION Func1(a : integer) : Longint;`

η κλήση της θα γίνει ως εξής:

`a := Func1(variable1);`

όπου η μεταβλητή `a` πρέπει να είναι τύπου `Longint` σύμφωνα με τον τύπο δεδομένου που επιστρέφει η συνάρτηση.

Οι εντολές μιας συνάρτησης περικλείονται σε `BEGIN` και `END`. Μεταξύ των εντολών αυτών μπορούν να υπάρχουν οποιεσδήποτε εντολές εκτός από αυτές που ορίζουν συναρτήσεις ή άλλες διαδικασίες.

Παράδειγμα. Μέγιστος Κοινός Διαιρέτης

```
FUNCTION MKD(a,b : longint) : longint;
BEGIN
  writeln(a:10, b:10);
  while b<>0 do
    begin
      c := a mod b;
      a := b;
      b := c;
      writeln(a:10, b:10);
    end;
  MKD := a;
END;
```

Η συνάρτηση του παραδείγματος, βρίσκει το μέγιστο κοινό διαιρέτη. Η διαφορά των συναρτήσεων με τις διαδικασίες είναι ότι οι συναρτήσεις επιστρέφουν πάντα μια τιμή, που εκχωρείται συνήθως σε μια μεταβλητή. Στις εντολές της συνάρτησης του παραδείγμα-

τος υπάρχει η εντολή `MKD:=a`. Με τον τρόπο αυτό, επιτυγχάνεται επιστροφή της τιμής στο κυρίως πρόγραμμα. Αν η εντολή αυτή δεν υπήρχε, η συνάρτηση δεν θα είχε επιστρέψει κανένα δεδομένο στο κυρίως πρόγραμμα.

4.3. ΑΝΑΔΡΟΜΗ

Ο όρος αναδρομή αναφέρεται στη δυνατότητα μιας διαδικασίας να χρησιμοποιεί την ίδια την διαδικασία δηλαδή να μπορεί να καλεί τον εαυτό της. Οι διαδικασίες και οι συναρτήσεις μπορεί να είναι αναδρομικές (recursive). Κλασικό παράδειγμα αναδρομής είναι ο υπολογισμός του $n!$ (παραγοντικού). Το παραγοντικό ενός μη αρνητικού αριθμού n ορίζεται ως εξής:

$n! = n*(n-1)!$ και

$0! = 1$

π.χ. για $n=5 \Rightarrow 5! = 5*4*3*2*1 = 120$

Η αναδρομική συνάρτηση `Fact` υπολογισμού του $n!$ υλοποιείται στο επόμενο πρόγραμμα:

```
PROGRAM Fact1;
VAR
  number : longint;
FUNCTION Fact(n : longint) : longint;
BEGIN
  IF n=1 THEN
    Fact := 1
  ELSE
    Fact := n * Fact(n-1);
  END;
BEGIN
  write('ΆεόŸääôâ ýíáí áñéèìü áðü ðï 1
      ýùð ðï 20 : ');
  readln(number);
  writeln('ðï ðáñääííðéëü ðïð áñéèìý
    ',number:2,' âßíáé ',Fact(number));
END.
```