Εισαγωγή

Σύμφωνα με την τεχνική σχεδίασης του τμηματικού προγραμματισμού για την επίλυση ενός προβλήματος απαραίτητη προϋπόθεση είναι η ανάλυσή του σε υποπροβλήματα που αποτελούν ανεξάρτητες ενότητες – modules.

Αντιμετωπίζοντας και επιλύοντας την κάθε ενότητα ξεχωριστά επιλύεται το αρχικό πρόβλημα.

Τμηματικός προγραμματισμός ονομάζεται η τεχνική σχεδίασης και ανάπτυξης προγραμμάτων ως ένα σύνολο από απλούστερα τμήματα προγραμμάτων.

Η σωστή διαίρεση είναι μια πολύπλοκη διαδικασία αλλά έχει πολλά πλεονεκτήματα.

Τα τρία βασικά τμήματα που συναντώνται σε κάθε σχεδόν πρόβλημα είναι (δείτε και το παράδειγμα που ακολουθεί):

- Εισαγωγή δεδομένων (καταχώρηση, έλεγχος)

- Επεξεργασία δεδομένων (πράξεις - υπολογισμοί).

- Έξοδος δεδομένων (εκτύπωση ή εμφάνιση).

Τα παραπάνω τμήματα μπορούν να αποτελούν τμήματα προγραμμάτων ως ξεχωριστές οντότητες, τα **υποπρογράμματα**. Τα υποπρογράμματα πρέπει να διακρίνονται από **3 βασικές αρχές :**

- Κάθε υποπρόγραμμα έχει μία είσοδο και μία έξοδο. Το υποπρόγραμμα δέχεται κάποιες τιμές - παραμέτρους, πραγματοποιεί κάποιες ενέργειες - υπολογισμούς και επιστρέφει μία ή περισσότερες τιμές κατά τον τερματισμό του.

- Κάθε υποπρόγραμμα πρέπει να είναι όσο το δυνατόν ανεξάρτητο από τα άλλα. Να είναι διαφανής ο τρόπος που λειτουργεί. Σε ιδεατό επίπεδο δέχεται κάποιες τιμές - παραμέτρους και επιστρέφει αποτέλεσμα – ενέργειες.

- Κάθε υποπρόγραμμα πρέπει να μην είναι πολύ μεγάλο. Αυτό είναι υποκειμενικό αλλά γενικά μπορούμε να πούμε πως κάθε πρόγραμμα πρέπει να επιτελεί συγκεκριμένη ενέργεια - υπολογισμό και να τερματίζεται (να επιστρέφει τον έλεγχο στο πρόγραμμα ή υποπρόγραμμα που το κάλεσε).

Πλεονεκτήματα από τη χρήση τμηματικού προγραμματισμού

- Διευκολύνει την ανάπτυξη του αλγορίθμου και του αντιστοίχου προγράμματος. Επιτρέπει την εξέταση και την επίλυση απλών προβλημάτων και όχι στην αντιμετώπιση του συνολικού προβλήματος. Με τη σταδιακή επίλυση των υποπροβλημάτων και τη δημιουργία των αντιστοίχων υποπρογραμμάτων τελικά επιλύεται το συνολικό πρόβλημα.

- Διευκολύνει την κατανόηση και διόρθωση του προγράμματος. Ο χωρισμός του προγράμματος σε μικρότερα αυτοτελή τμήματα επιτρέπει τη γρήγορη διόρθωση ενός συγκεκριμένου τμήματος του χωρίς οι αλλαγές αυτές να επηρεάσουν όλο το υπόλοιπο πρόγραμμα. Επίσης διευκολύνει οποιονδήποτε χρειαστεί να διαβάσει και να κατανοήσει τον τρόπο που λειτουργεί το πρόγραμμα.

- Απαιτεί λιγότερο χρόνο και προσπάθεια στη συγγραφή του προγράμματος. Πολύ συχνά χρειάζεται η ίδια λειτουργία σε διαφορετικά σημεία ενός προγράμματος. Από τη στιγμή που ένα υποπρόγραμμα έχει γραφεί, μπορεί το ίδιο να καλείται από πολλά σημεία του προγράμματος. Έτσι μειώνονται το μέγεθος του προγράμματος, ο χρόνος που απαιτείται για τη συγγραφή του και οι πιθανότητες λάθους, ενώ ταυτόχρονα το πρόγραμμα γίνεται πιο εύληπτο και κατανοητό.

- Επεκτείνει τις δυνατότητες των γλωσσών προγραμματισμού. Ένα υποπρόγραμμα που έχει γραφεί μπορεί να χρησιμοποιηθεί πολύ εύκολα και σε άλλα προγράμματα. Από τη στιγμή που έχει δημιουργηθεί, η χρήση του δεν διαφέρει από τη χρήση των ενσωματωμένων συναρτήσεων που παρέχει η γλώσσα προγραμματισμού, όπως για τον υπολογισμό του ημίτονου ή του συνημίτονου ή την εντολή με την οποία εκτελεί μία συγκεκριμένη διαδικασία, για παράδειγμα γράφει στην οθόνη (εντολή ΓΡΑΨΕ). Αν λοιπόν χρειάζεται συχνά κάποια λειτουργία που δεν υποστηρίζεται απευθείας από τη γλώσσα, όπως για παράδειγμα η εύρεση του μικρότερου δύο αριθμών, τότε μπορεί να γραφεί το αντίστοιχο υποπρόγραμμα. Η συγγραφή πολλών υποπρογραμμάτων και η δημιουργία βιβλιοθηκών με αυτά, ουσιαστικά επεκτείνουν την ίδια τη γλώσσα προγραμματισμού.

Παράμετροι

Σύμφωνα με όσα προαναφέρθηκαν στα πλαίσια του τμηματικού προγραμματισμού ευνοείται η ανάπτυξη υποπρογραμμάτων (ανεξαρτήτων και αυτόνομων τμημάτων προγραμμάτων) τα οποία ενεργοποιούνται από το κύριο πρόγραμμα ή από άλλα υποπρογράμματα. Η ενεργοποίηση αυτή πραγματοποιείται με την **κλήση τους**. Για την επικοινωνία των υποπρογραμμάτων χρησιμοποιούνται οι παράμετροι, δηλαδή μια ή περισσότερες μεταβλητές που επιτρέπουν το πέρασμα τιμών από ένα τμήμα προγράμματος σε άλλο. Οι μεταβλητές που είναι ορισμένες στο κύριο πρόγραμμα καλούνται και πραγματικές παράμετροι ενώ οι μεταβλητές ενός υποπρογράμματος τυπικές.

Είδη υποπρογραμμάτων

Υπάρχουν δυο είδη υποπρογραμμάτων, **οι διαδικασίες και οι συναρτήσεις**.

- Διαδικασία: τύπος υποπρογράμματος που μπορεί να εκτελέσει όλες τις λειτουργίες ενός προγράμματος δηλαδή να διαβάζει δεδομένα, να πραγματοποιεί υπολογισμούς, να τυπώνει μεταβλητές και μηνύματα κ.λ.π.

- Συνάρτηση: τύπος υποπρογράμματος δέχεται μια ομάδα τιμών, τις επεξεργάζεται, υπολογίζει και επιστρέφει μια μόνο τιμή.

Γενικοί κανόνες για την δημιουργία υποπρογραμμάτων

1. Διασπάμε το πρόβλημα σε υποπροβλήματα και αναπτύσσουμε το πρόγραμμα για κάθε ένα από αυτά. Τέλος, αναπτύσσουμε το πρόγραμμα – συνδετικό κρίκο. Στην παρουσίαση της τελικής λύσης, πρώτα παρατίθεται το κυρίως πρόγραμμα και στη συνέχεια τα υποπρογράμματα.

2. Μελετάμε τα υποπροβλήματα ώστε να καταλήξουμε στην επιλογή διαδικασίας ή συνάρτησης για την επίλυσή τους.

3. Κάθε υποπρόγραμμα έχει μία είσοδο και μία έξοδο.

4. Κάθε υποπρόγραμμα πρέπει να είναι όσο το δυνατόν ανεξάρτητο από τα άλλα.

5. Κάθε υποπρόγραμμα πρέπει να μην είναι πολύ μεγάλο.

6. Πρέπει να είναι σωστή η αντιστοίχιση των παραμέτρων σε μια διαδικασία, ως προς το πλήθος και τον τύπο.

7. Κατά την επιλογή συνάρτησης πρέπει να ορίζεται ο τύπος της, ο τύπος δηλαδή της τιμής που επιστρέφει η συνάρτηση.

Κλήσεις Υποπρογραμμάτων

Κατά την κλήση ενός υποπρογράμματος, διακόπτεται η εκτέλεση του κυρίως προγράμματος και περνάει ο έλεγχος στο υποπρόγραμμα. Τότε, από το κυρίως πρόγραμμα "περνούν" μέσω της λίστας παραμέτρων τιμές. Σε μια διαδικασία οι παράμετροι παίζουν επιπλέον το ρόλο επιστροφής τιμών στο κυρίως πρόγραμμα, κάτι που δεν συμβαίνει σε μια συνάρτηση. Βέβαια, κάθε υποπρόγραμμα μπορεί να καλείται από το κύριο πρόγραμμα αλλά και από άλλο υποπρόγραμμα. Στο επόμενο παράδειγμα αναλύονται οι λεπτομέρειες κλήσης υποπρογραμμάτων και το πέρασμα τιμών μεταξύ των τμημάτων προγραμμάτων.

Να σχηματίσετε τον πίνακα τιμών του παρακάτω αλγορίθμου. Τί θα εκτυπωθεί;

```
ΠΡΟΓΡΑΜΜΑ Κλήσεις_Υποπρογραμμάτων
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Α, Β
  ΠΡΑΓΜΑΤΙΚΕΣ : Γ, ΜΟ
ΑΡΧΗ
  Α <– 1
  Β <– 10
  Γ <– 31.6
  ΚΑΛΕΣΕ Πράξεις (Α, Γ)
  ΚΑΛΕΣΕ Πράξεις (Β, Γ)
  ΓΡΑΨΕ Α, Β, Γ
  Γ <– Γ ^ 2 + Α * Β
  Α <– Α + Τιμή (Γ)
  ΓΡΑΨΕ Α, Β, Γ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
! ========================================
ΔΙΑΔΙΚΑΣΙΑ Πράξεις (αριθμός1, αριθμός2)
ΣΤΑΘΕΡΕΣ
  Χ = 13.2
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: αριθμός1
  ΠΡΑΓΜΑΤΙΚΕΣ : αριθμός2
ΑΡΧΗ
  αριθμός1 <– Τιμή (αριθμός2)
  αριθμός2 <– αριθμός2 - Χ
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
! ========================================
ΣΥΝΑΡΤΗΣΗ Τιμή (Χ): ΑΚΕΡΑΙΗ
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ : Χ
  ΑΚΕΡΑΙΕΣ: εκτίμηση
ΑΡΧΗ
  εκτίμηση <- Α_Μ (Τ_Ρ (Χ)) + 1
  Τιμή <- εκτίμηση
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

	Κυρίως Πρόγραμμα			Διαδικασία		Συνάρτηση	
	A	B	Γ	αριθμός1	αριθμός2	εκτίμηση	X
Κυρίως πρόγραμμα :	1	10	31.6				
1η κλήση διαδικασίας :				1	31.6		
Κλήση συνάρτησης :							31.6
Εκτέλεση συνάρτησης :						6	
Επιστροφή στη διαδικασία :				6			
Εκτέλεση διαδικασίας :					18.4		
Επιστροφή στο κυρίως πρόγραμμα :	6		18.4				
2η κλήση διαδικασίας :				10	18.4		
Κλήση συνάρτησης :							18.4
Εκτέλεση συνάρτησης :						5	
Επιστροφή στη διαδικασία :				5			
Εκτέλεση διαδικασίας :					5.2		
Επιστροφή στο κυρίως πρόγραμμα :		5	5.2				
Εκτέλεση κύριου προγράμματος :			57.04				
Κλήση συνάρτησης :							57.04
Εκτέλεση συνάρτησης :						8	
Επιστροφή στο κυρίως πρόγραμμα και τερματισμός :	14						

Κατά την 1η κλήση της διαδικασίας πράξεις θα περάσουν: η τιμή της μεταβλητής Α (1) στην μεταβλητή αριθμός1 και η τιμή της μεταβλητής Γ (31.6) στην αριθμός2. Πρέπει να σημειωθεί οτι η σταθερά Χ που ορίστηκε στην διαδικασία δεν έχει σχέση με την ομώνυμη μεταβλητή που δηλώθηκε στη συνάρτηση Τιμή.

Στη συνέχεια περνάμε στην εκτέλεση της συνάρτησης Τιμή. Η τετραγωνική ρίζα του 31.6 είναι ο άρρητος αριθμός 5.621387 του οποίου η συνάρτηση A_M επιστρέφει την τιμή 5, άρα

η μεταβλητή εκτίμηση θα πάρει την τιμή 6, που είναι και η τιμή που θα επιστρέψει η συνάρτηση κατά την πρώτη κλήση της.

Κατά την 2η κλήση της διαδικασίας πράξεις θα περάσουν: η τιμή της μεταβλητής Β (10) που δεν έχει μεταβληθεί ακόμη στην μεταβλητή αριθμός1 και η τιμή της μεταβλητής Γ (18.4) στην αριθμός2.

Στη συνέχεια περνάμε στην εκτέλεση της συνάρτησης Τιμή. Η τετραγωνική ρίζα του 18.4 είναι ο άρρητος αριθμός 4,289522 του οποίου η συνάρτηση Α_Μ επιστρέφει την τιμή 4, άρα η μεταβλητή εκτίμηση θα πάρει την τιμή 5, που είναι και η τιμή που θα επιστρέψει η συνάρτηση κατά την δεύτερη κλήση της.

Στην πρώτη εντολή ΓΡΑΨΕ θα εκτυπωθούν οι τιμές: **6, 5, 5.2**

Στη συνέχεια εκτελείται η εντολή τροποποίησης του Γ.

Ακολουθεί η εκτέλεση της συνάρτησης Τιμή. Η τετραγωνική ρίζα του 35.2 είναι ο άρρητος αριθμός 7.552483 του οποίου η συνάρτηση Α_Μ επιστρέφει την τιμή 7, άρα η μεταβλητή εκτίμηση θα πάρει την τιμή 8, που είναι και η τιμή που θα επιστρέψει η συνάρτηση κατά την τρίτη κλήση της.

Στην δεύτερη εντολή ΓΡΑΨΕ θα εκτυπωθούν οι τμές: **14, 5, 57.04**

Παράδειγμα 1 (σελίδα 211 σχολικού βιβλίου)

Ας υποθέσουμε ότι πρέπει να γράψουμε πρόγραμμα που να διαβάζει την ακτίνα ενός κύκλου, να υπολογίζει και να εκτυπώνει το εμβαδόν του. Το πρόγραμμα παρουσιάζεται στη συνέχεια και λόγω της απλότητάς του δε χρήζει σχολιασμού. Οι εντολές του βασικού τμήματος του προγράμματος έχουν τοποθετηθεί σε πλαίσια ώστε να διευκολυνθεί η μετατροπή του προγράμματος σε υποπρογράμματα.

Σε όλα σχεδόν τα προγράμματα που έχουν μέχρι τώρα παρουσιαστεί η δομή τους είναι ίδια με αυτή του παραδείγματος αυτού. Οι πρώτες εντολές ζητούν από το χρήστη τα δεδομένα εισόδου τα οποία θα χρησιμοποιηθούν στους υπολογισμούς (τμήμα Α). Στη συνέχεια εκτελούνται οι υπολογισμοί που η εκφώνηση του προγράμματος απαιτεί (τμήμα Β). Τέλος, πραγματοποιούνται οι εκτυπώσεις των αποτελεσμάτων με τη διάταξη που η εκφώνηση της άσκησης ζητά (τμήμα Γ). Κάθε ένα από αυτά τα τμήματα του προγράμματος στην ουσία αποτελεί ένα υποπρόβλημα.

Σύμφωνα με τις αρχές του τμηματικού προγραμματισμού η διάσπαση ενός προβλήματος σε υποπροβλήματα και η μερική επίλυση των τελευταίων επιλύει το αρχικό πρόβλημα και εμπεριέχει και πολλά πλεονεκτήματα. Κάθε υποπρόβλημα στη ΓΛΩΣΣΑ μπορεί να επιλυθεί με ένα υποπρόγραμμα. Υποπρογράμματα αποτελούν οι διαδικασίες και οι συναρτήσεις. Το πρόγραμμα για τον υπολογισμό του εμβαδού του κύκλου είναι το εξής:

ΠΡΟΓΡΑΜΜΑ Υπολογισμός_Εμβαδού1 **ΣΤΑΘΕΡΕΣ** Π=3.14 **ΜΕΤΑΒΛΗΤΕΣ** **ΠΡΑΓΜΑΤΙΚΕΣ**: R, E **ΑΡΧΗ**	
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ **ΓΡΑΨΕ** Δώσε την ακτίνα **ΔΙΑΒΑΣΕ** R **ΜΕΧΡΙΣ_ΟΤΟΥ** R > 0	**Τμήμα Α**
E <- Π * R ^ 2	**Τμήμα Β**
ΓΡΑΨΕ 'Το εμβαδόν του κύκλου είναι :', E	**Τμήμα Γ**
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ	

Το τμήμα Α λοιπόν, του αρχικού προγράμματος *Υπολογισμός_Εμβαδού1* θα πρέπει να υλοποιηθεί σε ένα υποπρόγραμμα το οποίο θα ονομάζεται *Είσοδος_Δεδομένων*. Το υποπρόγραμμα αυτό αποτελεί μια διαδικασία και παρουσιάζεται στη συνέχεια. Η μεταβλητή *Αριθμός* αποτελεί παράμετρο. Παράμετρος ονομάζεται μια μεταβλητή που επιτρέπει το πέρασμα μιας τιμής από ένα υποπρόγραμμα σε ένα άλλο ή και στο κυρίως πρόγραμμα. Μια παράμετρος πρέπει να εμφανίζεται στο τμήμα δηλώσεων της διαδικασίας.

Επισημαίνεται ακόμη, ότι η μεταβλητή *Αριθμός* εκφράζει ουσιαστικά την ακτίνα του κύκλου (αυτή καλείται ο χρήστης να εισάγει) αλλά στα πλαίσια της διαδικασίας χρησιμοποιείται άλλο όνομα.

```
ΔΙΑΔΙΚΑΣΙΑ Είσοδος_δεδομένων (Αριθμός)
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ : Αριθμός
ΑΡΧΗ
  ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΡΑΨΕ 'Δώσε την ακτίνα'
    ΔΙΑΒΑΣΕ Αριθμός
  ΜΕΧΡΙΣ_ΟΤΟΥ Αριθμός > 0
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Το τμήμα Β του αρχικού προγράμματος εκτελεί τους υπολογισμούς. Στο συγκεκριμένο παράδειγμα οι υπολογισμοί αποτελούνται από μια και μόνο έκφραση αλλά γενικά μπορεί να αποτελούνται από δεκάδες γραμμές κώδικα. Δεδομένου ότι το υποπρόγραμμα αυτό πρέπει να επιστρέψει μία και μόνο τιμή (το εμβαδόν του κύκλου) θα επιλέξουμε την υλοποίηση με μια συνάρτηση, το όνομα της οποίας θα είναι *Εμβαδό_Κύκλου.*

Το όρισμα εισόδου της συνάρτησης είναι η ακτίνα του κύκλου, μια παράμετρος με το όνομα *ακτίνα*. Πρέπει να επισημανθεί ότι η τιμή που επιθυμούμε να επιστραφεί πρέπει υποχρεωτικά να εκχωρείται στο όνομα της συνάρτησης και αυτή η εντολή εκχώρησης είναι η τελευταία του τμήματος εντολών της συνάρτησης.

```
ΣΥΝΑΡΤΗΣΗ Εμβαδό_κύκλου (ακτίνα) : ΠΡΑΓΜΑΤΙΚΗ
ΣΤΑΘΕΡΕΣ
  Π=3.14
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ: ακτίνα
ΑΡΧΗ
  Εμβαδό_κύκλου <- Π * ακτίνα^2
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Το τμήμα Γ του αρχικού υποπρογράμματος πραγματοποιεί τις εκτυπώσεις των αποτελεσμάτων. Το υποπρόγραμμα θα υλοποιηθεί με μια διαδικασία. Η παράμετρος είναι ένας αριθμός ο οποίος εκτυπώνεται με το κατάλληλο μήνυμα στο βασικό τμήμα εντολών της διαδικασίας.

```
ΔΙΑΔΙΚΑΣΙΑ Εκτύπωση (Αποτέλεσμα)
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ : Αποτέλεσμα
ΑΡΧΗ
  ΓΡΑΨΕ 'Το εμβαδό του κύκλου είναι :', Αποτέλεσμα
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Το κυρίως πρόγραμμα πλέον έχει σκοπό την κλήση των υποπρογραμμάτων ώστε να εκτελεστεί ο κώδικας που αυτά περιέχουν. Οι διαδικασίες καλούνται με την εντολή ΚΑΛΕΣΕ της ΓΛΩΣΣΑΣ ενώ οι συναρτήσεις συμμετέχουν σε εκφράσεις εντολών εκχώρησης. Οι παράμετροι σε συναρτήσεις και διαδικασίες είναι μεταβλητές που έχουν δηλωθεί και χρησιμοποιούνται στο κυρίως πρόγραμμα.

Έτσι, κατά την κλήση της διαδικασίας *Είσοδος_Δεδομένων* η μεταβλητή R που έχει δηλωθεί στο κυρίως πρόγραμμα αντιστοιχίζεται με την μεταβλητή *Αριθμός* που έχει δηλωθεί στα πλαίσια της διαδικασίας που προαναφέρθηκε.

```
ΠΡΟΓΡΑΜΜΑ Υπολογισμός_Εμβαδού2
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ : R, E
ΑΡΧΗ
  ΚΑΛΕΣΕ Είσοδος_δεδομένων(R)
  E <- Εμβαδό_κύκλου(R)
  ΚΑΛΕΣΕ Εκτύπωση(E)
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Παρατηρήσεις

1. Το όνομα κάθε υποπρογράμματος ακολουθεί τους κανόνες ονοματολογίας των μεταβλητών (σελίδα 150 σχολικού βιβλίου).

2. Ο τύπος των δεδομένων που επιστρέφει μια συνάρτηση πρέπει να υποστηρίζονται από τη ΓΛΩΣΣΑ.

3. Η τελευταία εντολή σε μια συνάρτηση πρέπει να είναι εντολή εκχώρησης της τιμής που θα επιστραφεί στο όνομα της συνάρτησης.

4. Η κλήση της συνάρτησης πραγματοποιείται με το όνομά της με τη χρήση παραμέτρων εισόδου και εντολή εκχώρησης σε μεταβλητή του κυρίως προγράμματος ιδίου τύπου με την συνάρτηση.

5. Η κλήση μιας διαδικασίας πραγματοποιείται με την εντολή ΚΑΛΕΣΕ ακολουθούμενη από το όνομα της διαδικασίας και τις παραμέτρους εισόδου - εξόδου, Μια παράμετρος μπορεί να είναι εισόδου και εξόδου.

6. Η χρήση παραμέτρων δεν είναι απαραίτητη σε μια διαδικασία.

7. Ο μηχανισμός έχει ως εξής: Εκτελούνται οι εντολές του κυρίως προγράμματος μέχρι το σημείο κλήσης της συνάρτησης ή της διαδικασίας. Τότε περνούν οι τιμές των παραμέτρων από το κυρίως πρόγραμμα στο υποπρόγραμμα και εκτελούνται όλες οι εντολές του δευτέρου. Στη συνέχεια περνούν οι τιμές από το υποπρόγραμμα στο κυρίως πρόγραμμα και συνεχίζεται η ροή των εντολών του τελευταίου.

8. Κάθε μεταβλητή έχει νόημα και μπορεί να χρησιμοποιείται μόνο εντός του τμήματος προγράμματος που είναι δηλωμένες. Κατά συνέπεια υπάρχει η περίπτωση να εμφανίζονται μεταβλητές με το ίδιο όνομα σε διαφορετικά υποπρογράμματα (που φυσικά είναι έγκυρο). Αυτές οι μεταβλητές δε, δεν έχουν συσχέτιση μεταξύ τους.

9. Οι παράμετροι είναι μεταβλητές που χρησιμοποιούνται για το πέρασμα από το ένα τμήμα προγράμματος στο άλλο.

10. Κάθε παράμετρος καταγράφεται εντός των παρενθέσεων που ακολουθούν το όνομα της διαδικασίας ή της συνάρτησης.

π.χ. ΔΙΑΔΙΚΑΣΙΑ Είσοδος_Δεδομένων (R) ή ΣΥΝΑΡΤΗΣΗ Έλεγχος_Δεδομένων (Α, R, Χ) : ΛΟΓΙΚΗ

τα Α, R, Χ είναι παράμετροι.

11. Πρέπει να δηλώνουμε τις παραμέτρους αυτές, οι οποίες δεν πρέπει να ξεχνάμε ότι είναι μεταβλητές, εντός του σώματος του υποπρογράμματος με τον ίδιο τρόπο που δηλώνονται οι μεταβλητές σε ένα πρόγραμμα γραμμένο ΓΛΩΣΣΑ.

12. Οι πίνακες δηλώνονται μόνο με το όνομά τους πλησίον του ονόματος του υποπρογράμματος και ο τύπος αλλά και <u>οι διαστάσεις</u> τους δηλώνονται στη συνέχεια στο τμήμα δηλώσεων του υποπρογράμματος.

13. Σε ένα πρόγραμμα μπορούμε να ορίσουμε και σταθερές, οι οποίες όμως μπορούν να χρησιμοποιηθούν (εμβέλεια) μόνο στο υποπρόγραμμα που έχουν δηλωθεί.

14. Κατά την επιλογή υποπρογράμματος πρέπει να έχουμε υπόψη μας ότι μια συνάρτηση δέχεται μια σειρά από τιμές (με αντίστοιχες παραμέτρους) τις επεξεργάζεται και επιστρέφει ΜΙΑ τιμή. Σε οποιαδήποτε άλλη άσκηση πρέπει να χρησιμοποιηθεί διαδικασία. Δεν υπάρχει περίπτωση να δούμε τις εντολές ΓΡΑΨΕ και ΔΙΑΒΑΣΕ εντός μιας συνάρτησης.

Συμβουλές - Υποδείξεις (από τετράδιο μαθητή):

Πρέπει να μελετάμε τους κανόνες που υπάρχουν για την ανάπτυξη υποπρογραμμάτων στο προγραμματιστικό περιβάλλον που χρησιμοποιούμε.

1. Πρέπει να δημιουργούμε διάγραμμα ιεραρχίας για το κάθε υποπρόγραμμα, να αναπτύσσουμε τον αλγόριθμο επίλυσης κάθε υποπροβλήματος και στη συνέχεια να αναπτύσσουμε το υποπρόγραμμα.

2. Μελέτη υποπρογράμματος και επιλογή τύπου υποπρογράμματος που απαιτείται.

3. Κάποια υποπρογράμματα ίσως να έχουν αντιμετωπιστεί σε άλλη άσκηση, οπότε και είναι έτοιμα.

4. Ανεξαρτησία υποπρογραμμάτων.

5. Αν πρόκειται για συνάρτηση σωστός ορισμός του τύπου της και εκχώρησης τιμής σε αυτήν στο τέλος.

6. Προσοχή στο πλήθος και στον τύπο των παραμέτρων: <u>Δήλωση</u> και <u>κλήση</u> υποπρογραμμάτων.