Ανάπτυξη εφαρμογών σε προγραμματιστικό περιβάλλον

Γ’ Λυκείου

ΚΕΦΑΛΑΙΟ 10

Υποπρογράμματα

Χρήστος Μουρατίδης - Έκδοση 2021
mouratx@yahoo.com
http://users.sch.gr/mouratx

Περιεχόμενα

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ 3

ΠΑΡΑΔΕΙΓΜΑ ΤΜΗΜΑΤΟΠΟΙΗΣΗΣ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ 3

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΩΝ ΥΠΟΠΡΟΓΡΑΜΜΑΤΩΝ 5

ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΤΟΥ ΤΜΗΜΑΤΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ 6

ΠΑΡΑΜΕΤΡΟΙ 7

ΕΙΔΗ ΥΠΟΠΡΟΓΡΑΜΜΑΤΩΝ 8

ΔΙΑΔΙΚΑΣΙΕΣ 9

Ορισμος διαδικασιας 9
Παραδειγματα διαδικασιων 10
Κληση διαδικασιας 13
Ολοκληρωμενα παραδειγματα δημιουργιας και κλησης διαδικασιων 13

ΣΥΝΑΡΤΗΣΕΙΣ 23

Ορισμος συναρτησης 23
Παραδειγματα συναρτησεων 24
Κληση συναρτησης 25
Ολοκληρωμενα παραδειγματα δημιουργιας και κλησης συναρτησεων 27

ΠΡΑΓΜΑΤΙΚΕΣ ΚΑΙ ΤΥΠΙΚΕΣ ΠΑΡΑΜΕΤΡΟΙ 36

ΕΜΒΕΛΕΙΑ ΜΕΤΑΒΛΗΤΩΝ-ΣΤΑΘΕΡΩΝ 40

ΕΡΩΤΗΣΕΙΣ ΚΑΤΑΝΟΗΣΗΣ 41

ΠΑΡΑΡΤΗΜΑ 45

Αναδρομη 45

Εισαγωγή στον τμηματικό προγραμματισμό

Τα περισσότερα προβλήματα είναι σύνθετα και μερικά εξ' αυτών μεγάλης κλίμακας (π.χ. η δημιουργία ενός πληροφοριακού συστήματος όπως το taxisnet ή το myschool), κάτι που απαιτεί μία συγκεκριμένη μεθοδολογία για την επίλυσή τους.

Στο Κεφάλαιο 6 εξετάσαμε τεχνικές σχεδίασης προγραμμάτων, όπως η **ιεραρχική σχεδίαση** με την οποία διασπάμε το αρχικό (σύνθετο) πρόβλημα σε επιμέρους υπο-προβλήματα, καθένα εκ των οποίων είναι ευκολότερο να επιλυθεί ή/και να αναλυθεί σε άλλα επιμέρους υπο-προβλήματα κ.ο.κ. Οι τελικοί αλγόριθμοι που προκύπτουν για καθένα υπο-πρόβλημα υλοποιούνται ως μία **αυτόνομη ενότητα προγράμματος**, που ονομάζεται **υποπρόγραμμα**. Έτσι, **το τελικό (μεγάλο) πρόγραμμα συντίθεται από τα διάφορα υποπρογράμματα που συνενώνονται μεταξύ τους**.

Ορισμός (*από το σχολικό βιβλίο*)

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

Παράδειγμα τμηματοποίησης του προβλήματος

Μία μετεωρολογική υπηρεσία επιθυμεί να συγκεντρώσει και καταχωρήσει τις ημερήσιες θερμοκρασίες του μηνός Μαρτίου και να πληροφορηθεί :

- Ποιά ήταν η μέση θερμοκρασία του μήνα.
- Ποιά ήταν η μέγιστη και ελάχιστη θερμοκρασία και σε ποιές ημέρες παρουσιάστηκαν.
- Δημιουργία ενός ιστογράμματος που να παρουσιάζει, με γραφικό τρόπο, την θερμοκρασιακή εξέλιξη του μήνα.

Πρόκειται περί ενός σύνθετου προβλήματος, το οποίο μπορεί να διασπαστεί σε επιμέρους υπο-προβλήματα. Μία **ανάλυση** θα μπορούσε να καταδείξει τα εξής **τμήματα**:

1. **Εισαγωγή των θερμοκρασιών**
2. **Επεξεργασία των θερμοκρασιών**
3. **Εμφάνιση των αποτελεσμάτων**

Προφανώς, **το κάθε τμήμα πρέπει να αναλυθεί ακόμα περισσότερο**:

1. **Εισαγωγή των θερμοκρασιών**
   1.1 Καταχώρηση (διάβασμα) των θερμοκρασιών
   1.2 Έλεγχος εγκυρότητας

2. **Επεξεργασία των θερμοκρασιών**
   2.1 Υπολογισμός της μέσης θερμοκρασίας
   2.2 Υπολογισμός της μέγιστης θερμοκρασίας και της ημέρας που πραγματοποιήθηκε
   2.3 Υπολογισμός της ελάχιστης θερμοκρασίας και της ημέρας που πραγματοποιήθηκε
   2.4 Δημιουργία της γραφικής παράστασης (ιστογράμματος)

3. **Εμφάνιση των αποτελεσμάτων**
   3.1 Εκτύπωση της μέσης/μέγιστης/ελάχιστης θερμοκρασίας
   3.2 Εκτύπωση της γραφικής παράστασης

Η **διαγραμματική αναπαράσταση της *τμηματικής ανάλυσης* του προβλήματος** είναι η εξής:

Για κάθε υποπρόβλημα μπορεί να γραφεί και το κατάλληλο τμήμα (ή ενότητα) προγράμματος. **Η τελική ανάπτυξη του προγράμματος πραγματοποιείται από την ανάπτυξη των επιμέρους τμημάτων (υποπρογραμμάτων) και τη σύνδεση μεταξύ τους**.

Τις περισσότερες φορές, τα υποπρογράμματα είναι ήδη έτοιμα, καθώς έχουν ήδη δημιουργηθεί για την επίλυση ανάλογων προβλημάτων. Για παράδειγμα, για τον υπολογισμό του μέσου όρου μίας ομάδας τιμών, είναι πολύ πιθανό να υπάρχει έτοιμο υποπρόγραμμα και αυτό που μένει είναι να συνδεθεί κατάλληλα με το κυρίως πρόγραμμα[1].

Χαρακτηριστικά των υποπρογραμμάτων

Τα υποπρογράμματα πρέπει να έχουν τις εξής **τρεις ιδιότητες**:

1. **Κάθε υποπρόγραμμα έχει μόνο μία είσοδο και μία έξοδο**. Αυτό σημαίνει ότι:
   - Ενεργοποιείται με την είσοδο σε αυτό, που γίνεται πάντα από την αρχή του (επικεφαλίδα)
   - Εκτελεί τις εντολές που περιέχει στο σώμα του και
   - Απενεργοποιείται με την έξοδο απ' αυτό, που γίνεται πάντα από το τέλος του.

2. **Κάθε υποπρόγραμμα είναι ανεξάρτητο από τα άλλα.** Αυτό σημαίνει ότι κάθε υποπρόγραμμα μπορεί να σχεδιαστεί, να αναπτυχθεί και να συντηρηθεί αυτόνομα (μερικές φορές και από διαφορετικές ομάδες προγραμματιστών) χωρίς να επηρεάζει τα άλλα υποπρογράμματα.[2]

3. **Κάθε υποπρόγραμμα δεν πρέπει να είναι πολύ μεγάλο.** Η ιδέα είναι ότι κάθε υποπρόγραμμα είναι ένα σχετικά μικρό τμήμα προγράμματος που εκτελεί μία συγκεκριμένη λειτουργία (π.χ. υπολογισμός μέσου όρου). Αν κάνει και άλλες λειτουργίες (π.χ. εύρεση μεγίστου/ελαχίστου) τότε το μέγεθός του μεγαλώνει αλλά προκαλεί και σύγχυση για το τί ακριβώς κάνει. Αυτό έχει επίπτωση στην περαιτέρω βελτίωση και συντήρησή του. Στην περίπτωση αυτή, το τμήμα αυτό πρέπει να διασπαστεί σε άλλα μικρότερα, ώστε οι επιπλέον λειτουργίες να απομονωθούν και να υλοποιηθούν σε ξεχωριστά υποπρογράμματα.

[1] Ένα σύνολο έτοιμων υποπρογραμμάτων αποτελεί μία *βιβλιοθήκη*.
[2] Η *απόλυτη ανεξαρτησία των υποπρογραμμάτων* είναι, βέβαια, ένας ιδεατός επιθυμητός στόχος. Στην πράξη, όμως, είναι δύσκολο να επιτευχθεί.

Πλεονεκτήματα του τμηματικού προγραμματισμού

- **Διευκολύνει την ανάπτυξη του αλγορίθμου και του αντίστοιχου προγράμματος.**
  Επικεντρωνόμαστε στην επίλυση μικρών προβλημάτων αντί στη συνολική αντιμετώπιση του σύνθετου προβλήματος. Κατασκευάζοντας μικρά τμήματα προγραμμάτων και σταδιακά συνδέοντάς τα στο κυρίως πρόγραμμα επιτυγχάνουμε την επίλυση του σύνθετου προβλήματος.

- **Διευκολύνει την κατανόηση και διόρθωση του προγράμματος.**
  Κάθε υποπρόγραμμα αποτελεί αυτοτελές τμήμα προγράμματος που μπορεί να συντηρηθεί ξεχωριστά από τα υπόλοιπα. Αν χρειαστούν διορθώσεις τότε επικεντρωνόμαστε σε αυτό μόνο. Επιπλέον, ένας τρίτος προγραμματιστής μπορεί να το διαβάσει και να κατανοήσει ευκολότερα τη λειτουργία του.

- **Γρηγορότερη ανάπτυξη του προγράμματος**.
  Πολλά υποπρογράμματα είναι έτοιμα και *επαναχρησιμοποιήσιμα*, δηλαδή με καθόλου ή ελάχιστες μετατροπές μπορούν να συνδεθούν στο κυρίως πρόγραμμα. Με αυτόν τον τρόπο, η ανάπτυξη του προγράμματος γίνεται γρήγορα.

- **Επεκτείνει τις δυνατότητες των γλωσσών προγραμματισμού**.
  Πολλές εντολές σε μία γλώσσα προγραμματισμού (όπως π.χ. η εντολή **ΓΡΑΨΕ** της ΓΛΩΣΣΑΣ) στην πραγματικότητα είναι υποπρογράμματα που εκτελούν μία συγκεκριμένη λειτουργία (π.χ. η εντολή **ΓΡΑΨΕ** εμφανίζει ένα μήνυμα στην οθόνη). Γράφοντας δικά μας υποπρογράμματα ή συνδέοντας έτοιμα, ουσιαστικά είναι σαν να προσθέτουμε νέες εντολές στη γλώσσα προγραμματισμού επεκτείνοντας το λεξιλόγιό της. Η ομαδοποίηση των υποπρογραμμάτων σε ένα ενιαίο σύνολο συνιστά μία **βιβλιοθήκη**, η οποία επεκτείνει τη γλώσσα προγραμματισμού.
  Για παράδειγμα, αν αναπτύξουμε υποπρογράμματα σχετικά με μαθηματικές επεξεργασίες (μέσος όρος, ελάχιστο/μέγιστο, τυπική απόκλιση, διακύμανση κ.λπ.) τότε όλα αυτά μπορούν να ενσωματωθούν σε μία βιβλιοθήκη υποπρογραμμάτων σχετική με μαθηματικές λειτουργίες. Αργότερα, η βιβλιοθήκη αυτή μπορεί να συνδεθεί σε οποιοδήποτε πρόγραμμα χρειάζεται μαθηματικές επεξεργασίες.

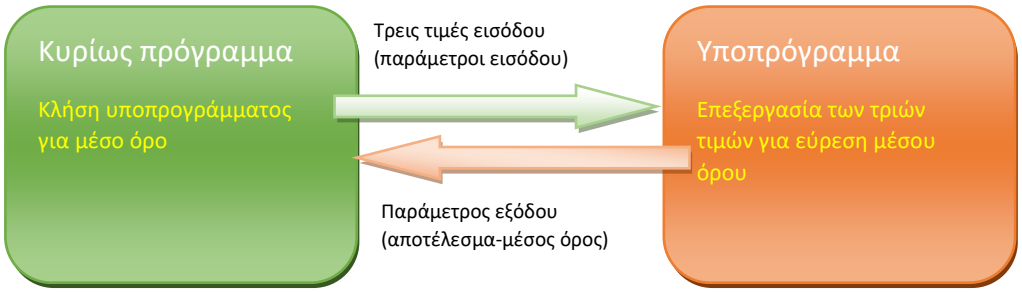
Παράμετροι

Οι παράμετροι είναι τιμές που περνάνε από το κυρίως πρόγραμμα ή υποπρόγραμμα σε ένα άλλο υποπρόγραμμα. Με αυτόν τον τρόπο διευκολύνεται η επικοινωνία μεταξύ τους.

Για παράδειγμα, το κυρίως πρόγραμμα χρειάζεται σε ένα σημείο του υπολογισμό του μέσου όρου τριών αριθμών. Υπάρχει ήδη ένα υποπρόγραμμα για το σκοπό αυτό. Τότε :

- Το κυρίως πρόγραμμα ενεργοποιεί/καλεί το υποπρόγραμμα περνώντας του, ως παραμέτρους τους τρεις αριθμούς.
- Το υποπρόγραμμα, αφού λάβει τους τρεις αριθμούς, κάνει τον υπολογισμό του μέσου όρου.
- Το αποτέλεσμα του υποπρογράμματος επιστρέφει στο κυρίως πρόγραμμα (στο αντίστοιχο σημείο στο οποίο έγινε η κλήση/ενεργοποίηση.
- Το υποπρόγραμμα απενεργοποιείται.

Στο παρακάτω αφαιρετικό σχήμα, φαίνεται γενικά, το πέρασμα των τιμών των παραμέτρων από το κυρίως πρόγραμμα στο υποπρόγραμμα και η επιστροφή αποτελέσματος.



Όταν το κυρίως πρόγραμμα καλεί/ενεργοποιεί[3] το υποπρόγραμμα τότε η ροή εκτέλεσης μεταφέρεται στο υποπρόγραμμα. Μόλις η λειτουργία του υποπρογράμματος ολοκληρωθεί, η ροή εκτέλεσης επιστρέφει στο σημείο όπου έγινε η κλήση ή στην επόμενη εντολή που ακολουθεί μετά την κλήση του υποπρογράμματος.

Το πού θα επιστρέψει η ροή εκτέλεσης εξαρτάται από τον τύπο του υποπρογράμματος, όπως θα δούμε στη συνέχεια.

[3] Θα προτιμήσουμε τον όρο *καλεί (call)*.

Ορισμός (*από το σχολικό βιβλίο*)

Παράμετρος είναι μία μεταβλητή που επιτρέπει το πέρασμα της τιμής της από ένα τμήμα προγράμματος σε ένα άλλο.

Παρακάτω, θα δούμε πιο συγκεκριμένα τα είδη και τον μηχανισμό λειτουργίας των υποπρογραμμάτων και των παραμέτρων.

Είδη υποπρογραμμάτων

Παρακάτω, βλέπουμε τα είδη των υποπρογραμμάτων και τη χρησιμότητα του καθενός:

Διαδικασία

Είναι υποπρόγραμμα που μπορεί να εκτελέσει οποιαδήποτε λειτουργία ενός προγράμματος.

Για παράδειγμα, το διάβασμα (εισαγωγή) των δεδομένων, η εκτύπωση (έξοδος) των αποτελεσμάτων ακόμα και η επεξεργασία δεδομένων και επιστροφή *κανενός*, *ενός* ή *περισσοτέρων* αποτελεσμάτων μέσω των παραμέτρων της.

Συνάρτηση

Είναι υποπρόγραμμα που εκτελεί μία συγκεκριμένη επεξεργασία δεδομένων.

Για παράδειγμα, η επεξεργασία ενός συνόλου αριθμών και ο υπολογισμός του μέσου όρου τους. Το αποτέλεσμα επιστρέφει μέσω του ονόματος της συνάρτησης.

Παρατηρήσεις:

- Η **συνάρτηση** σε σχέση με την διαδικασία έχει περιορισμένη λειτουργικότητα που εστιάζεται στην επεξεργασία κάποιων δεδομένων και **υπολογισμό ενός μόνο αποτελέσματος**.

- Μία διαδικασία, ως ένα μεγαλύτερης ευρύτητας είδος υποπρογράμματος, μπορεί να αντικαταστήσει μία συνάρτηση. Με άλλα λόγια, αντί να χρησιμοποιήσουμε συνάρτηση για κάποια λειτουργικότητα, κάλλιστα μπορούμε να χρησιμοποιήσουμε, αντί αυτής, μία διαδικασία.

- Μία συνάρτηση μοιάζει με μία συνάρτηση των μαθηματικών καθώς και με τις ενσωματωμένες συναρτήσεις της `ΓΛΩΣΣΑΣ` (Π.χ. `ΗΜ(x)`, `Τ_Ρ(x)` κ.α.).

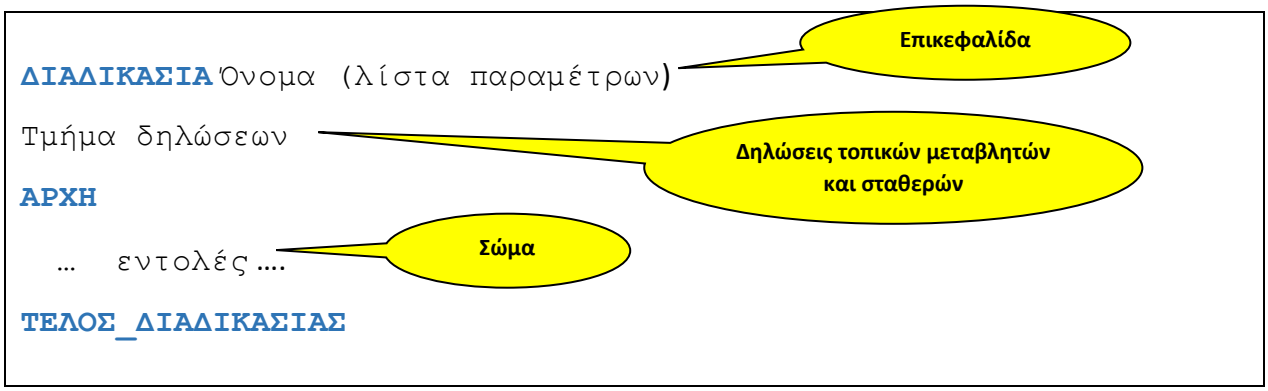
- Τα υποπρογράμματα γράφονται στο τέλος, μετά το κυρίως πρόγραμμα.

Παρακάτω, θα εξετάσουμε πιο συγκεκριμένα τον μηχανισμό κλήσης και λειτουργίας των διαδικασιών και συναρτήσεων.

Διαδικασίες

Ορισμός διαδικασίας

Μία διαδικασία έχει την εξής **συντακτική δομή**:



- Η **επικεφαλίδα** περιέχει **το όνομα της διαδικασίας** και ***προαιρετικά*** μία **λίστα παραμέτρων**, ανάλογα με το αν θα μεταβιβαστούν τιμές προς την διαδικασία (παράμετροι εισόδου) ή θα επιστραφούν τιμές αποτελεσμάτων προς το κυρίως πρόγραμμα (παράμετροι εξόδου).

- Το **τμήμα δηλώσεων** περιλαμβάνει τις **μεταβλητές και σταθερές** που θα χρησιμοποιηθούν εντός της διαδικασίας. Λέγονται **τοπικές μεταβλητές και σταθερές** διότι η εμβέλειά τους (scope) είναι μόνο εντός της διαδικασίας.

- Το **σώμα** της διαδικασίας περιέχει εντολές ή ακόμα και κλήσεις προς άλλα υποπρογράμματα.

Στο τμήμα δηλώσεων ενός υποπρογράμματος περιλαμβάνονται:

- Οι μεταβλητές που αφορούν τις παραμέτρους.
- Οι μεταβλητές που θα χρησιμοποιηθούν εσωτερικά αποκλειστικά για τη λειτουργία του υποπρογράμματος.

Παραδείγματα διαδικασιών

Παράδειγμα 1: Γράψτε μία διαδικασία που τυπώνει το μήνυμα «Καλωσήρθατε στον προγραμματισμό»

```
ΔΙΑΔΙΚΑΣΙΑ Εκτύπωσε_μήνυμα
ΑΡΧΗ

    ΓΡΑΨΕ 'Καλωσήρθατε στον προγραμματισμό'

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Πρόκειται για μία απλούστατη διαδικασία χωρίς λίστα παραμέτρων και τμήμα δηλώσεων μεταβλητών/σταθερών. Το σώμα περιέχει μία μόνο εντολή εξόδου.

Παράδειγμα 2: Γράψτε μία διαδικασία που δέχεται ένα μήνυμα και το τυπώνει το στην οθόνη.

```
ΔΙΑΔΙΚΑΣΙΑ Εκτύπωσε_μήνυμα(μήνυμα)
ΜΕΤΑΒΛΗΤΕΣ
     ΧΑΡΑΚΤΗΡΕΣ: μήνυμα
ΑΡΧΗ

   ΓΡΑΨΕ μήνυμα

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Πρόκειται για μία απλή διαδικασία με μία παράμετρο εισόδου, τύπου χαρακτήρα. Λαμβάνει λοιπόν, την τιμή του μηνύματος και το τυπώνει στην οθόνη.

Παράδειγμα 3: Γράψτε μία διαδικασία που λαμβάνει 2 ακέραιες τιμές και υπολογίζει κι επιστρέφει τον μέσο όρο τους.

```
ΔΙΑΔΙΚΑΣΙΑ Υπολόγισε_μέσο_όρο(x, y, ΜΟ)
ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: x, y
     ΠΡΑΓΜΑΤΙΚΕΣ: ΜΟ
ΑΡΧΗ

   ΜΟ ← (x + y) / 2

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Ένα υποπρόγραμμα που επιστρέφει μόνο μία τιμή μπορεί να υλοποιηθεί είτε ως διαδικασία είτε ως συνάρτηση.

- Στην περίπτωση της διαδικασίας, η τιμή θα επιστραφεί μέσω της αντίστοιχης παραμέτρου.
- Στην περίπτωση της συνάρτησης, η τιμή θα επιστραφεί μέσω του ονόματος της συνάρτησης, όπως θα δούμε παρακάτω.

Παράδειγμα 4: Γράψτε μία διαδικασία που λαμβάνει 2 ακέραιες τιμές και υπολογίζει κι επιστρέφει το άθροισμα και τον μέσο όρο τους.

```
ΔΙΑΔΙΚΑΣΙΑ Υπολόγισε_πράξεις(x, y, Sum, MO)
ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: x, y, Sum
     ΠΡΑΓΜΑΤΙΚΕΣ: MO
ΑΡΧΗ

   Sum ← x + y
   MO ← (x + y) / 2

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Παράδειγμα 5: Γράψτε μία διαδικασία που λαμβάνει έναν πίνακα ακεραίων Π[20] κι επιστρέφει την μέγιστη κι ελάχιστη τιμή.

Ο αλγόριθμοι εύρεσης μεγίστου/ελαχίστου σε μονοδιάστατο πίνακα είναι γνωστοί και απλά υλοποιούνται εντός μίας διαδικασίας. Φυσικά, η έξοδος των αποτελεσμάτων (δηλαδή, οι τιμές των μεγίστου και ελαχίστου) επιστρέφονται από τη διαδικασία μέσω των αντίστοιχων παραμέτρων της.

```
ΔΙΑΔΙΚΑΣΙΑ Εύρεση_ελάχιστου_μέγιστου(Π, MIN, MAX)
ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: Π[20], MIN, MAX, i
ΑΡΧΗ

   !Αρχικές τιμές.
   MIN ← Π[1]
   MAX ← Π[1]

   !Σάρωση του πίνακα για την εύρεση του ελαχίστου/μεγίστου.
   ΓΙΑ i ΑΠΟ 2 ΜΕΧΡΙ 20

      ΑΝ Π[i] < MIN ΤΟΤΕ
          MIN ← Π[i]
      ΤΕΛΟΣ_ΑΝ

      ΑΝ Π[i] > MAX ΤΟΤΕ
          MAX ← Π[i]
      ΤΕΛΟΣ_ΑΝ

   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Κλήση διαδικασίας

Αφού είδαμε πώς δημιουργούμε μία διαδικασία, με ή χωρίς παραμέτρους, ήρθε η ώρα να δούμε πώς ενεργοποιείται, δηλαδή **πώς καλείται από το κυρίως πρόγραμμα (ή ένα άλλο υποπρόγραμμα).**

Η **κλήση της διαδικασίας** στο κυρίως πρόγραμμα ή υποπρόγραμμα γίνεται με την **εντολή** **ΚΑΛΕΣΕ** που έχει την εξής **γενική σύνταξη**:

```
ΚΑΛΕΣΕ Όνομα-διαδικασίας (λίστα παραμέτρων)
```

Στα προηγούμενα παραδείγματα **θα καλούσαμε τις διαδικασίες** ως εξής:

- **ΚΑΛΕΣΕ** `Εκτύπωσε_μήνυμα`
- **ΚΑΛΕΣΕ** `Εκτύπωσε_μήνυμα('Καλωσήρθατε στον προγραμματισμό')`
- **ΚΑΛΕΣΕ** `Υπολόγισε_μέσο_όρο(5,6, Μέσος)` , όπου οι τιμή `5` θα μεταβιβαστεί στην παράμετρο `x` και η τιμή 6 θα μεταβιβαστεί στην παράμετρο y. `Μέσος` είναι μία μεταβλητή του κύριου προγράμματος στην οποία θα επιστρέψει το αποτέλεσμα μέσω της αντίστοιχης παραμέτρου `ΜΟ` της διαδικασίας.
- **ΚΑΛΕΣΕ** `Υπολόγισε_μέσο_όρο(a, b, Μέσος)` , όπου οι τιμή της μεταβλητής `a` του κύριου προγράμματος θα μεταβιβαστεί στην παράμετρο `x` και η τιμή της μεταβλητής `b` του κύριου προγράμματος θα μεταβιβαστεί στην παράμετρο y. `Μέσος` είναι μία μεταβλητή του κυρίως προγράμματος στην οποία θα επιστρέψει το αποτέλεσμα μέσω της αντίστοιχης παραμέτρου `ΜΟ` της διαδικασίας.

Ολοκληρωμένα παραδείγματα δημιουργίας και κλήσης διαδικασιών

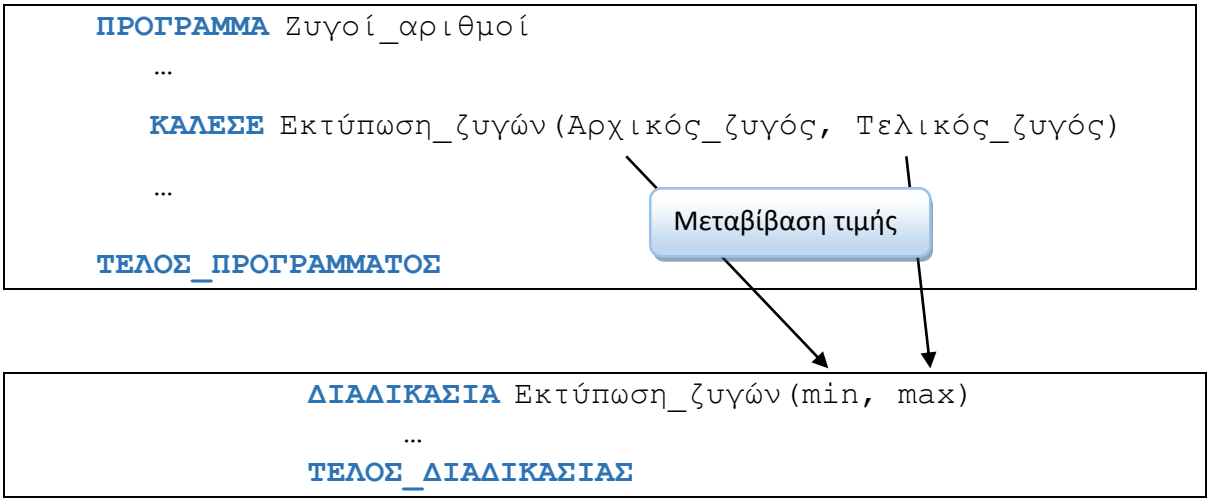
Παράδειγμα 1: Να γραφτεί πρόγραμμα που εκτυπώνει όλους τους ζυγούς αριθμούς από τον ελάχιστο μέχρι τον μέγιστο αριθμό που θα δώσει ο χρήστης. Με άλλα λόγια, την αρχική και τελική τιμή του ζυγού θα τη διαβάσει το πρόγραμμα από τον χρήστη. Θεωρούμε ότι ο χρήστης θα δώσει έγκυρες αρχικές και τελικές τιμές ζυγών.

Δειγματικά, αν η αρχική τιμή ζυγού είναι 2 και η τελική τιμή ζυγού είναι 100 τότε το πρόγραμμα θα τυπώσει όλους τους ζυγούς από το 2 μέχρι το 100.

```
ΠΡΟΓΡΑΜΜΑ Ζυγοί_αριθμοί
ΜΕΤΑΒΛΗΤΕΣ
   ΑΚΕΡΑΙΕΣ: Αρχικός_ζυγός, Τελικός_ζυγός
ΑΡΧΗ
   !Είσοδος δεδομένων.
   ΓΡΑΨΕ 'Παρακαλώ, δώστε την αρχική τιμή του ζυγού'
   ΔΙΑΒΑΣΕ Αρχικός_ζυγός

   ΓΡΑΨΕ 'Παρακαλώ, δώστε την τελική τιμή του ζυγού'
   ΔΙΑΒΑΣΕ Τελικός_ζυγός

  !Κλήση διαδικασίας εκτύπωσης. Οι τιμές των μεταβλητών του
  !κυρίως προγράμματος μεταβιβάζονται στις αντίστοιχες
  !παραμέτρους της διαδικασίας.
  ΚΑΛΕΣΕ Εκτύπωση_ζυγών(Αρχικός_ζυγός, Τελικός_ζυγός)

  !Ενημερωτικό μήνυμα ότι η εκτύπωση ολοκληρώθηκε.
  ΓΡΑΨΕ 'Η εκτύπωση των ζυγών ολοκληρώθηκε'

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

!=============================================================
!Ορισμός της διαδικασίας.
!=============================================================

ΔΙΑΔΙΚΑΣΙΑ Εκτύπωση_ζυγών(min, max)
ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: min, max, i
ΑΡΧΗ
   !Εκτύπωση ζυγών ξεκινώντας από τον min και φτάνοντας
   !στον max.
   ΓΙΑ i ΑΠΟ min ΜΕΧΡΙ max ΜΕ_ΒΗΜΑ 2

      ΓΡΑΨΕ i

   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

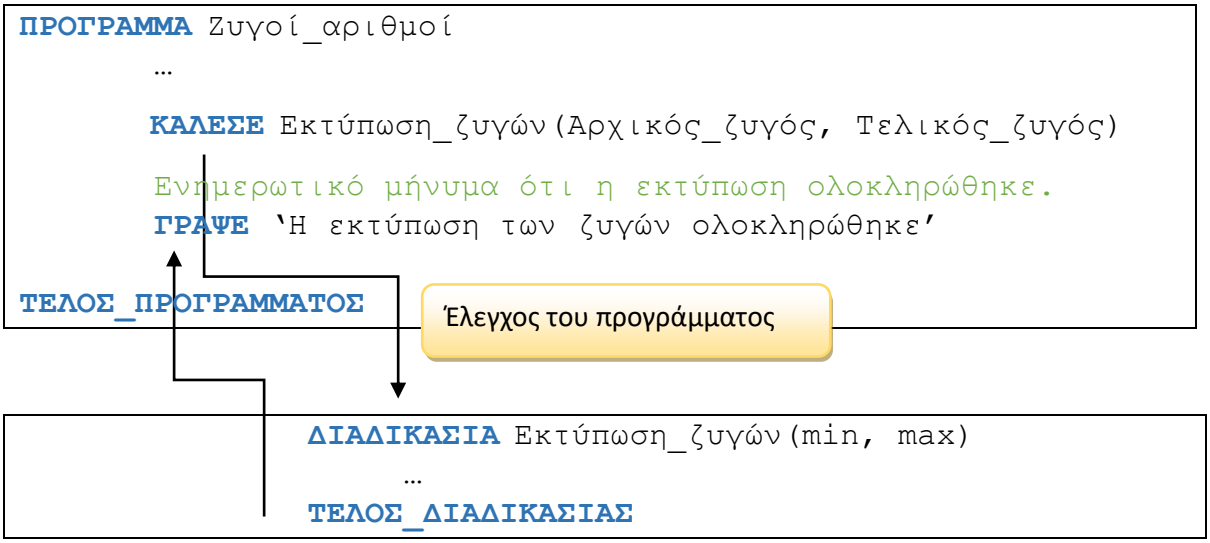
Παρατηρήσεις:

- Στο κυρίως πρόγραμμα, η διαδικασία καλείται με την εντολή **ΚΑΛΕΣΕ** περνώντας τις τιμές των μεταβλητών `Αρχικός_ζυγός` και `Τελικός_ζυγός` στις αντίστοιχες παραμέτρους `min` και `max` της διαδικασίας.



- **Δεν είναι απαραίτητο τα ονόματα των μεταβλητών του κύριου προγράμματος που μεταβιβάζουν τις τιμές τους να είναι τα ίδια με τα ονόματα των παραμέτρων της διαδικασίας που δέχονται αυτές τις τιμές**. Φυσικά, δεν απαγορεύεται να είναι τα ίδια.

- **Με την εκτέλεση της εντολής ΚΑΛΕΣΕ, ο έλεγχος του προγράμματος μεταφέρεται στην διαδικασία**. Μόλις ολοκληρωθεί η εκτέλεση των εντολών της διαδικασίας, ο έλεγχος του προγράμματος μεταφέρεται στην επόμενη εντολή μετά την **ΚΑΛΕΣΕ**. Στο παράδειγμα παραπάνω, ο έλεγχος μεταφέρεται στην τελική εντολή **ΓΡΑΨΕ** του κύριου προγράμματος.



Παράδειγμα 2: Στο προηγούμενο πρόγραμμα με την εκτύπωση των ζυγών αριθμών, υπάρχει ένα τμήμα που διαβάζει ποιός είναι ο αρχικός ζυγός και ποιός ο τελικός ζυγός (είσοδος δεδομένων). Υλοποιήστε το τμήμα αυτό με διαδικασία και διαμορφώστε το κυρίως πρόγραμμα ανάλογα.

```
ΠΡΟΓΡΑΜΜΑ Ζυγοί_αριθμοί_έκδοση_2
ΜΕΤΑΒΛΗΤΕΣ
   ΑΚΕΡΑΙΕΣ: Αρχικός_ζυγός, Τελικός_ζυγός
ΑΡΧΗ
  !Κλήση της διαδικασίας που διαβάζει τα δεδομένα της αρχικής
  !και τελικής τιμής.
  ΚΑΛΕΣΕ Διάβασμα_δεδομένων(Αρχικός_ζυγός, Τελικός_ζυγός)

  !Κλήση διαδικασίας εκτύπωσης. Οι τιμές των μεταβλητών του
  !κυρίως προγράμματος μεταβιβάζονται στις αντίστοιχες
  !παραμέτρους της διαδικασίας.
  ΚΑΛΕΣΕ Εκτύπωση_ζυγών(Αρχικός_ζυγός, Τελικός_ζυγός)

  !Ενημερωτικό μήνυμα ότι η εκτύπωση ολοκληρώθηκε.
  ΓΡΑΨΕ 'Η εκτύπωση των ζυγών ολοκληρώθηκε'

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ


!=============================================================
!Ορισμοί διαδικασιών.
!=============================================================

ΔΙΑΔΙΚΑΣΙΑ Διάβασμα_δεδομένων(min, max)
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ: min, max
```

```
ΑΡΧΗ
    !Είσοδος δεδομένων.
   ΓΡΑΨΕ 'Παρακαλώ, δώστε την αρχική τιμή του ζυγού'
   ΔΙΑΒΑΣΕ min

   ΓΡΑΨΕ 'Παρακαλώ, δώστε την τελική τιμή του ζυγού'
   ΔΙΑΒΑΣΕ max

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ


ΔΙΑΔΙΚΑΣΙΑ Εκτύπωση_ζυγών(min, max)
ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: min, max, i
ΑΡΧΗ
   !Εκτύπωση ζυγών ξεκινώντας από τον min και φτάνοντας
   !στον max.
   ΓΙΑ i ΑΠΟ min ΜΕΧΡΙ max ΜΕ_ΒΗΜΑ 2

      ΓΡΑΨΕ i

   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Παράδειγμα 3: Στο προηγούμενο πρόγραμμα, η διαδικασία Διάβασμα_δεδομένων διαβάζει τις αρχικές και τελικές τιμές των ζυγών χωρίς να πραγματοποιεί **έλεγχο εγκυρότητας, δηλαδή αν αυτές οι τιμές που διαβάστηκαν είναι όντως ζυγοί αριθμοί.** Τροποποιήστε τη διαδικασία ώστε να κάνει έλεγχο εγκυρότητας και για τα δύο δεδομένα.

Η διαδικασία παραπάνω τροποποιείται ως εξής:

1η υλοποίηση

```
ΔΙΑΔΙΚΑΣΙΑ Διάβασμα_δεδομένων(min, max)
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ: min, max
ΑΡΧΗ
    !Είσοδος δεδομένων.
   ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ

      ΓΡΑΨΕ 'Παρακαλώ, δώστε την αρχική τιμή του ζυγού'
      ΔΙΑΒΑΣΕ min
```

```
      ΑΝ (min MOD 2) <> 0 ΤΟΤΕ
         ΓΡΑΨΕ 'Η τιμή που δώσατε δεν είναι ζυγός αριθμός'
      ΤΕΛΟΣ_ΑΝ

   ΜΕΧΡΙΣ_ΟΤΟΥ (min MOD 2) = 0

   ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ

      ΓΡΑΨΕ 'Παρακαλώ, δώστε την τελική τιμή του ζυγού'
      ΔΙΑΒΑΣΕ max
      ΑΝ (max MOD 2) <> 0 ΤΟΤΕ
         ΓΡΑΨΕ 'Η τιμή που δώσατε δεν είναι ζυγός αριθμός'
      ΤΕΛΟΣ_ΑΝ

   ΜΕΧΡΙΣ_ΟΤΟΥ (max MOD 2) = 0

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Παρατηρούμε ότι **ο ίδιος κώδικας για τον έλεγχο εγκυρότητας πραγματοποιείται 2 φορές με τη διαφορά ότι ο πρώτος αφορά την αρχική τιμή και ο δεύτερος την τελική τιμή**. Συνεπώς, αυτό το **τμήμα του ελέγχου** μπορεί να υλοποιηθεί σε **νέα ξεχωριστή διαδικασία** και να κληθεί κατάλληλα από την παρούσα διαδικασία (*περίπτωση κλήσης υποπρογράμματος από άλλο υποπρόγραμμα)*.

2η υλοποίηση (καλύτερη)

```
ΔΙΑΔΙΚΑΣΙΑ Διάβασμα_δεδομένων(min, max)
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ: min, max
   ΛΟΓΙΚΕΣ: Έγκυρος_ζυγός
ΑΡΧΗ
    !Είσοδος δεδομένων.
    !Αρχική τιμή της λογικής μεταβλητής, για τον πρώτο ζυγό.
    Έγκυρος_ζυγός ← ΨΕΥΔΗΣ

    ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
       ΓΡΑΨΕ 'Παρακαλώ, δώστε την αρχική τιμή του ζυγού'
       ΔΙΑΒΑΣΕ min
       ΚΑΛΕΣΕ Έλεγχος_δεδομένου(min, Έγκυρος_ζυγός)
    ΜΕΧΡΙΣ_ΟΤΟΥ Έγκυρος_ζυγός

    !Αρχική τιμή της λογικής μεταβλητής, για τον τελευταίο
    !ζυγό.
    Έγκυρος_ζυγός ← ΨΕΥΔΗΣ
```

```
    ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
        ΓΡΑΨΕ 'Παρακαλώ, δώστε την τελική τιμή του ζυγού'
        ΔΙΑΒΑΣΕ max
        ΚΑΛΕΣΕ Έλεγχος_δεδομένου(max, Έγκυρος_ζυγός)
    ΜΕΧΡΙΣ_ΟΤΟΥ Έγκυρος_ζυγός

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ


ΔΙΑΔΙΚΑΣΙΑ Έλεγχος_δεδομένου(x, Έγκυρος)
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ: x
    ΛΟΓΙΚΕΣ: Έγκυρος
ΑΡΧΗ
    !Έλεγχος δεδομένου.
   ΑΝ (x MOD 2) = 0 ΤΟΤΕ
        Έγκυρος ← ΑΛΗΘΗΣ
   ΑΛΛΙΩΣ
        ΓΡΑΨΕ 'Η τιμή που δώσατε δεν είναι ζυγός αριθμός'
        Έγκυρος ← ΨΕΥΔΗΣ
   ΤΕΛΟΣ_ΑΝ

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Γενικά, οι έλεγχοι εγκυρότητας των εισαγόμενων τιμών επειδή επιστρέφουν μία λογική τιμή (ΑΛΗΘΗΣ/ΨΕΥΔΗΣ) είναι προτιμότερο να υλοποιούνται με συναρτήσεις.

Παράδειγμα 4: Μία εταιρεία έχει συγκεντρώσει την αξία των πωλήσεων που πραγματοποίησαν οι 15 πωλητές της την χρονιά που μας πέρασε. Επιθυμεί να δώσει ένα bonus στον καλύτερο πωλητή της, ανάλογα με την αξία των πωλήσεων που έκανε. Το bonus αυτό προκύπτει από τον εξής πίνακα:

Αξία πωλήσεων (€)	Bonus (€)
< 5000	200
>=5000 και <10000	300
>=10000	500

Γράψτε ένα πρόγραμμα που διαβάζει τα ονοματεπώνυμα των πωλητών και την αξία πωλήσεων που πραγματοποίησαν, βρίσκει τον «καλύτερο» πωλητή, και υπολογίζει το bonus του. Το πρόγραμμα θα εμφανίζει στην οθόνη το ονοματεπώνυμο του «καλύτερου» πωλητή, την αξία των πωλήσεων και το bonus του.

Αναλύοντας το πρόβλημα, παρατηρούμε ότι χωρίζεται σε 4 υπο-προβλήματα:

1. Διάβασμα των δεδομένων.
2. Υπολογισμός του «καλύτερου» πωλητή.
3. Υπολογισμός του bonus του και
4. Εμφάνιση των αποτελεσμάτων στην οθόνη.

Συνεπώς, θα **τμηματοποιήσουμε το πρόγραμμά μας** ανάλογα, φτιάχνοντας αντίστοιχες **διαδικασίες**.

```
ΠΡΟΓΡΑΜΜΑ Καλύτερος_πωλητής
ΜΕΤΑΒΛΗΤΕΣ
   ΧΑΡΑΚΤΗΡΕΣ: Πωλητής[15], MAX_Πωλητής
   ΠΡΑΓΜΑΤΙΚΕΣ: Πώληση[15], MAX_Πώληση, Bonus
   ΑΚΕΡΑΙΕΣ: i
ΑΡΧΗ
   !Είσοδος δεδομένων.
  !Οι δύο πίνακες είναι παράμετροι εξόδου, δηλαδή θα
  !επιστραφούν στο κυρίως πρόγραμμα γεμάτοι με δεδομένα.
   ΚΑΛΕΣΕ Διάβασμα_δεδομένων(Πωλητής, Πώληση)

   !Εύρεση του «καλύτερου» πωλητή.
   !Οι δύο πίνακες είναι παράμετροι εισόδου, δηλαδή δίνουν
   !δεδομένα στη διαδικασία.
   !Οι δύο MAX μεταβλητές είναι παράμετροι εξόδου, δηλαδή
   !επιστρέφουν αποτέλεσμα στο κυρίως πρόγραμμα.
   ΚΑΛΕΣΕ Εύρεση_του_καλύτερου_πωλητή(Πωλητής, Πώληση,
                              &  MAX_Πωλητής, MAX_Πώληση)


   !Εύρεση του bonus για τον «καλύτερο» πωλητή.
   !Το μόνο που χρειάζεται ως είσοδος είναι η μέγιστη πώληση.
   !Επιστρεφόμενο αποτέλεσμα είναι το bonus.
   ΚΑΛΕΣΕ Εύρεση_του_bonus(MAX_Πώληση, Bonus)

   !Εμφάνιση αποτελεσμάτων.
   !Όλες είναι παράμετροι εισόδου.
   !Το αποτέλεσμα είναι μηνύματα στην οθόνη.
   ΚΑΛΕΣΕ Εμφάνιση_αποτελεσμάτων(MAX_Πωλητής, MAX_Πώληση,
                                                & Bonus)

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

```
!============================================================
!Ορισμοί των διαδικασιών. Στις παραμέτρους θα χρησιμοποιήσουμε
τα ίδια ονόματα όπως και στις κλήσεις, αν και όπως γνωρίζουμε,
δεν είναι απαραίτητο αλλά διευκολύνει την ανάγνωση.
!============================================================

!Διάβασμα των δεδομένων στους μονοδιάστατους πίνακες.
ΔΙΑΔΙΚΑΣΙΑ Διάβασμα_δεδομένων(Πωλητής, Πώληση)
ΜΕΤΑΒΛΗΤΕΣ
     ΧΑΡΑΚΤΗΡΕΣ: Πωλητής[15]
     ΠΡΑΓΜΑΤΙΚΕΣ: Πώληση[15]
     ΑΚΕΡΑΙΕΣ: i
ΑΡΧΗ
   ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 15

      ΓΡΑΨΕ 'Παρακαλώ, δώστε το ονοματεπώνυμο του ',i, 'ου
                                                    πωλητή'
      ΔΙΑΒΑΣΕ Πωλητής[i]

      ΓΡΑΨΕ 'Παρακαλώ, δώστε τις πωλήσεις του ',i, 'ου
                                                    πωλητή'
      ΔΙΑΒΑΣΕ Πώληση[i]

   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ


!Εύρεση του «καλύτερου» πωλητή.
ΔΙΑΔΙΚΑΣΙΑ Εύρεση_του_καλύτερου_πωλητή(Πωλητής, Πώληση,
                              &  MAX_Πωλητής, MAX_Πώληση)
ΜΕΤΑΒΛΗΤΕΣ
     ΧΑΡΑΚΤΗΡΕΣ: Πωλητής[15], MAX_Πωλητής
     ΠΡΑΓΜΑΤΙΚΕΣ: Πώληση[15], MAX_Πώληση
     ΑΚΕΡΑΙΕΣ: i
ΑΡΧΗ
   !Υλοποιούμε τον τυπικό αλγόριθμο εύρεσης μεγίστου σε
   !μονοδιάστατο πίνακα.
   MAX_Πωλητής ← Πωλητής[1]
   MAX_Πώληση ← Πώληση[1]

   ΓΙΑ i ΑΠΟ 2 ΜΕΧΡΙ 15

      ΑΝ Πώληση[i] > MAX_Πώληση ΤΟΤΕ

         MAX_Πώληση ← Πώληση[i]
```

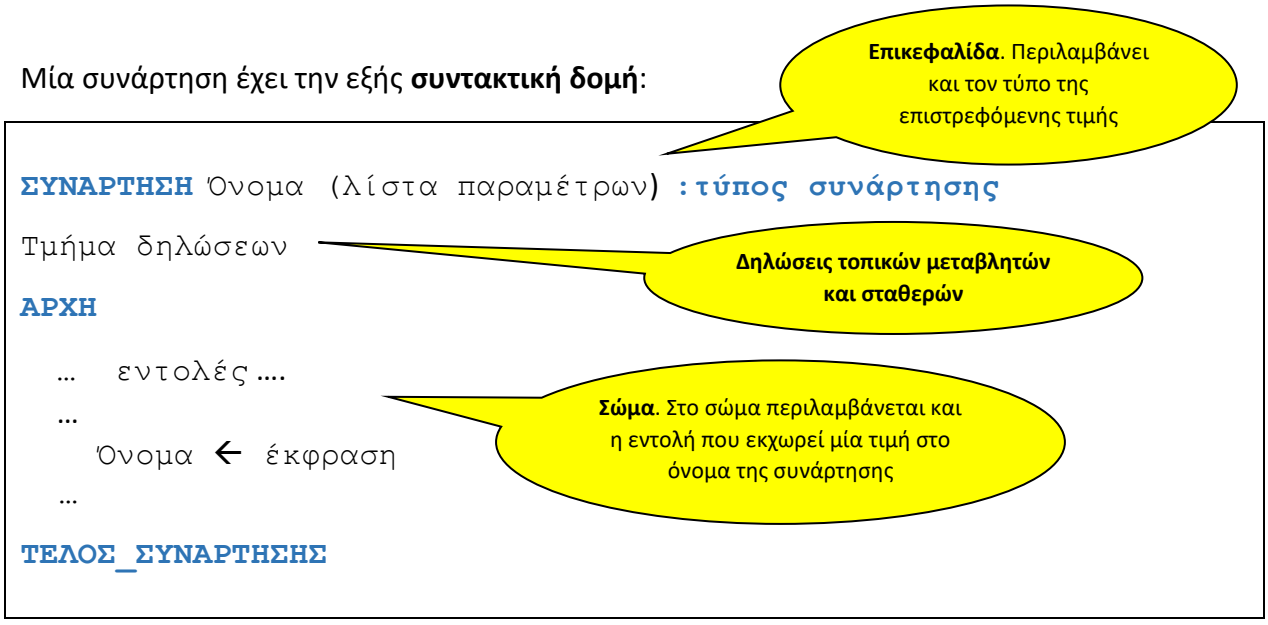
```
            ΜΑΧ_Πωλητής ← Πωλητής[i]

        ΤΕΛΟΣ_ΑΝ

    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ


!Εύρεση του bonus για τον «καλύτερο» πωλητή.
!Έχουμε υπόψη το πινακάκι της εκφώνησης.
ΔΙΑΔΙΚΑΣΙΑ Εύρεση_του_bonus(ΜΑΧ_Πώληση, Bonus)
ΜΕΤΑΒΛΗΤΕΣ
     ΠΡΑΓΜΑΤΙΚΕΣ: ΜΑΧ_Πώληση, Bonus
     ΑΚΕΡΑΙΕΣ: i
ΑΡΧΗ

    ΑΝ ΜΑΧ_Πώληση < 5000 ΤΟΤΕ

        Bonus ← 200

   ΑΛΛΙΩΣ_ΑΝ ΜΑΧ_Πώληση < 10000 ΤΟΤΕ

        Bonus ← 300

   ΑΛΛΙΩΣ

        Bonus ← 500

   ΤΕΛΟΣ_ΑΝ

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ


!Εμφάνιση αποτελεσμάτων.
ΔΙΑΔΙΚΑΣΙΑ Εμφάνιση_αποτελεσμάτων(ΜΑΧ_Πωλητής, ΜΑΧ_Πώληση,
                                                    & Bonus)
ΜΕΤΑΒΛΗΤΕΣ
     ΧΑΡΑΚΤΗΡΕΣ: ΜΑΧ_Πωλητής
     ΠΡΑΓΜΑΤΙΚΕΣ: ΜΑΧ_Πώληση, Bonus
ΑΡΧΗ

     ΓΡΑΨΕ 'Ο καλύτερος πωλητής είναι ο/η:', ΜΑΧ_Πωλητής
     ΓΡΑΨΕ 'με αξία πωλήσεων: ', ΜΑΧ_Πώληση
     ΓΡΑΨΕ 'Το bonus που δικαιούται είναι: ', Bonus
```

```
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Συναρτήσεις

Ορισμός συνάρτησης

Μία συνάρτηση έχει την εξής **συντακτική δομή**:



- Η **επικεφαλίδα** περιέχει **το όνομα της συνάρτησης** και ***προαιρετικά*** μία **λίστα παραμέτρων**, ανάλογα με το αν θα μεταβιβαστούν τιμές προς την συνάρτηση (παράμετροι εισόδου). **Η έξοδος (επιστροφή) ενός μόνο αποτελέσματος γίνεται μέσω του ονόματός της, οπότε εδώ δηλώνουμε τον τύπο της τιμής που επιστρέφει (*τύπος συνάρτησης*).**

- **Ο τύπος της συνάρτησης υποδηλώνει τον τύπο της τιμής που επιστρέφει μέσω του ονόματός της** και μπορεί να είναι οποιοσδήποτε τύπος που υποστηρίζει η ΓΛΩΣΣΑ (Ακέραιες, Πραγματικές, Λογικές, Χαρακτήρες) οπότε αυτόματα η συνάρτηση χαρακτηρίζεται ως **ΑΚΕΡΑΙΗ, ΠΡΑΓΜΑΤΙΚΗ, ΛΟΓΙΚΗ, ΧΑΡΑΚΤΗΡΑ.**

- Το **τμήμα δηλώσεων** περιλαμβάνει τις **μεταβλητές και σταθερές** που θα χρησιμοποιηθούν εντός της συνάρτησης. Λέγονται **τοπικές μεταβλητές και σταθερές** διότι η εμβέλειά τους (scope) είναι μόνο εντός της συνάρτησης.

- Το **σώμα** της συνάρτησης περιέχει εντολές ή ακόμα και κλήσεις προς άλλα υποπρογράμματα. **Πρέπει, όμως, υποχρεωτικά, να περιέχει μία εντολή όπου εκχωρείται τιμή στο όνομά της**.

Παραδείγματα συναρτήσεων

Παράδειγμα 1: Γράψτε μία συνάρτηση που υπολογίζει το εμβαδόν ενός τριγώνου έχοντας ως δεδομένα τη βάση και το ύψος.

```
ΣΥΝΑΡΤΗΣΗ Εμβαδόν_τριγώνου(βάση, ύψος) :ΠΡΑΓΜΑΤΙΚΗ
ΜΕΤΑΒΛΗΤΕΣ
   ΠΡΑΓΜΑΤΙΚΕΣ: βάση, ύψος
ΑΡΧΗ

   Εμβαδόν_τριγώνου ← (βάση * ύψος ) / 2

ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Η συνάρτηση έχει δύο παραμέτρους για είσοδο δεδομένων (`βάση` και `ύψος`) και ο τύπος της είναι `ΠΡΑΓΜΑΤΙΚΗ`, δηλαδή θα επιστρέψει μία πραγματική τιμή ως αποτέλεσμα. Το σώμα περιέχει μόνο μία εντολή εκχώρησης: Γίνεται η αριθμητική πράξη και το αποτέλεσμα εκχωρείται στο όνομα της συνάρτησης.

Παράδειγμα 2: Γράψτε μία συνάρτηση που λαμβάνει 2 ακέραιες τιμές και υπολογίζει κι επιστρέφει τον μέσο όρο τους.

```
ΣΥΝΑΡΤΗΣΗ ΜΟ(x, y) :ΠΡΑΓΜΑΤΙΚΗ
ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: x, y
ΑΡΧΗ
```

```
    ΜΟ ← (x + y) / 2

ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Παράδειγμα 3: Γράψτε μία συνάρτηση που λαμβάνει έναν πίνακα ακεραίων Π[20] κι επιστρέφει τον μέγιστο αυτών.

Ο αλγόριθμος εύρεσης μεγίστου σε μονοδιάστατο πίνακα είναι γνωστός και απλά υλοποιείται εντός μίας συνάρτησης. Φυσικά, η έξοδος του αποτελέσματος (δηλαδή, η μέγιστη τιμή) επιστρέφεται από τη συνάρτηση μέσω του ονόματός της.

```
ΣΥΝΑΡΤΗΣΗ Μέγιστη_τιμή(Π) :ΑΚΕΡΑΙΗ
ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: Π[20], MAX, i
ΑΡΧΗ

   !Αρχική τιμή.
   MAX ← Π[1]

   !Σάρωση του πίνακα για την εύρεση του μεγίστου.
   ΓΙΑ i ΑΠΟ 2 ΜΕΧΡΙ 20

      ΑΝ Π[i] > MAX ΤΟΤΕ
          MAX ← Π[i]
      ΤΕΛΟΣ_ΑΝ

   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

   !Εκχώρηση της τιμής της MAX στο όνομα της συνάρτησης.
   Μέγιστη_τιμή ← MAX

ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Κλήση συνάρτησης

Αφού είδαμε πώς δημιουργούμε μία συνάρτηση, ήρθε η ώρα να δούμε πώς ενεργοποιείται, δηλαδή **πώς καλείται από το κυρίως πρόγραμμα (ή ένα άλλο υποπρόγραμμα).**

Η **κλήση της συνάρτησης** στο κυρίως πρόγραμμα ή υποπρόγραμμα γίνεται στο σημείο εκείνο όπου χρειάζεται μία τιμή:

- Σε μία εντολή **ΓΡΑΨΕ**
- Σε μία εντολή εκχώρησης
- Γενικά, όπου υπάρχει μία έκφραση

Στα προηγούμενα παραδείγματα **θα καλούσαμε τις συναρτήσεις** ως εξής:

```
ΓΡΑΨΕ 'Το εμβαδόν τριγώνου είναι ' , Εμβαδόν_τριγώνου(10,2)
```

Κλήση της συνάρτησης όπου μεταβιβάζονται οι τιμές 10 για τη βάση και 2 για το ύψος.
Η τιμή που επιστρέφεται μέσω του ονόματός της θα τυπωθεί στο μήνυμα.

```
ΓΡΑΨΕ 'Το εμβαδόν τριγώνου είναι ' , Εμβαδόν_τριγώνου(22,4)
ΓΡΑΨΕ 'Το εμβαδόν τριγώνου είναι ' , Εμβαδόν_τριγώνου(β,υ)
```

Κλήση της συνάρτησης όπου μεταβιβάζονται οι τιμές των μεταβλητών β και υ του κυρίως προγράμματος στις αντίστοιχες παραμέτρους βάση και ύψος της συνάρτησης.
Η τιμή που επιστρέφεται μέσω του ονόματός της θα τυπωθεί στο μήνυμα.

```
Κ ← 5 * Εμβαδόν_τριγώνου(14, 3)
```

Κλήση της συνάρτησης όπου μεταβιβάζονται οι τιμές 14 για τη βάση και 3 για το ύψος.
Η τιμή που επιστρέφεται μέσω του ονόματός της θα συμμετάσχει στην αριθμητική πράξη.

Τί τιμή θα έχει η μεταβλητή Κ; Πρώτα, θα κληθεί η συνάρτηση Εμβαδόν_τριγώνου με τις τιμές 14 και 3 για βάση και ύψος αντίστοιχα. Θα επιστρέψει 26. Οπότε η έκφραση γίνεται 5 * 26. Η τιμή 130 εκχωρείται στην μεταβλητή Κ.

- Στην παρακάτω εντολή:

```
Κ ← 12 * ΜΟ(10,7) / 2
```

Τί τιμή θα έχει η μεταβλητή `Κ`;

- Για το παρακάτω τμήμα κώδικα που βρίσκεται στο κυρίως πρόγραμμα:

```
Χ ← 5
Υ ← 8
Κ ← 12 * ΜΟ(Χ,Υ) / 2
```

Πάλι, τί τιμή θα έχει η μεταβλητή `Κ`;

- Έστω ότι ο πίνακας `Π` έχει τις ακέραιες τιμές `4, -2, 13, 8, -1, 7`. Τί τιμή θα εκχωρηθεί στην μεταβλητή `Κ` στην παρακάτω εντολή;

```
Κ ← 12 * Μέγιστη_τιμή(Π) / 2
```

Απάντηση: Η κλήση της συνάρτησης `Μέγιστη_τιμή` με παράμετρο τον πίνακα `Π` θα επιστρέψει την τιμή `13` μέσω του ονόματός της. Συνεπώς, η έκφραση γίνεται `12 * 13 / 2` και θα αποτιμηθεί σε `78`.

- Επίσης, **μία συνάρτηση μπορεί να χρησιμοποιηθεί σε μία λογική έκφραση** μίας εντολής **ΑΝ** ή μίας **ΟΣΟ..ΕΠΑΝΑΛΑΒΕ**:

```
ΑΝ Μέγιστη_τιμή(Π) > 10 ΤΟΤΕ
    …
ΤΕΛΟΣ_ΑΝ

ΟΣΟ ΜΟ(Χ,Υ) < 10 ΕΠΑΝΑΛΑΒΕ
   …
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Ολοκληρωμένα παραδείγματα δημιουργίας και κλήσης συναρτήσεων

Παράδειγμα 1: Να γραφτεί πρόγραμμα που διαβάζει έναν ακέραιο αριθμό και ελέγχει αν αυτός βρίσκεται στο διάστημα [0-100]. Ο έλεγχος να πραγματοποιηθεί με μία συνάρτηση.

```
ΠΡΟΓΡΑΜΜΑ Διάβασμα_και_έλεγχος_αριθμού
ΜΕΤΑΒΛΗΤΕΣ
   ΑΚΕΡΑΙΕΣ: x
ΑΡΧΗ

   ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ

      ΓΡΑΨΕ 'Παρακαλώ, δώστε τον αριθμό'
      ΔΙΑΒΑΣΕ x

   ΜΕΧΡΙΣ_ΟΤΟΥ Έγκυρος_αριθμός(x)

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

!============================================================
!Ορισμός της συνάρτησης.
!============================================================

ΣΥΝΑΡΤΗΣΗ Έγκυρος_αριθμός(x) :ΛΟΓΙΚΗ
ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: x
ΑΡΧΗ

   ΑΝ x >= 0 ΚΑΙ x <= 100 ΤΟΤΕ
      Έγκυρος_αριθμός ← ΑΛΗΘΗΣ

   ΑΛΛΙΩΣ
      Έγκυρος_αριθμός ← ΨΕΥΔΗΣ

   ΤΕΛΟΣ_ΑΝ

ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Παράδειγμα 2: Ένας επιχειρηματίας επιθυμεί να προσφέρει έκπτωση στους πελάτες του όταν αγοράζουν προϊόν κάποιας συγκεκριμένης αξίας σύμφωνα με τον παρακάτω πίνακα:

Αξία προϊόντος (€)	Έκπτωση (%)
>500	2
>1000	5
>5000	12

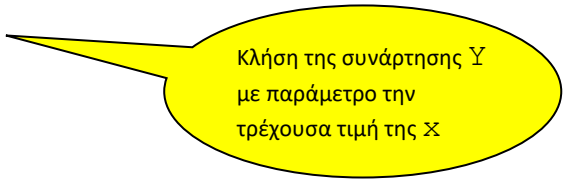
Γράψτε ένα πρόγραμμα που διαβάζει την αξία του προϊόντος και υπολογίζει την έκπτωση που δικαιούται ο πελάτης. Ο υπολογισμός της έκπτωσης να γίνει με συνάρτηση.

```
ΠΡΟΓΡΑΜΜΑ Έκπτωση_προϊόντος
ΜΕΤΑΒΛΗΤΕΣ
   ΠΡΑΓΜΑΤΙΚΕΣ: Αξία
ΑΡΧΗ
   ΓΡΑΨΕ 'Παρακαλώ, δώστε την αξία του προϊόντος'
   ΔΙΑΒΑΣΕ Αξία

   !Κλήση της συνάρτησης Έκπτωση, με παράμετρο την τιμή της
   !μεταβλητής Αξία, μέσα στην εντολή ΓΡΑΨΕ.
   ΓΡΑΨΕ 'Ο πελάτης δικαιούται ', Έκπτωση(Αξία), '€ έκπτωση'

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

!============================================================
!Ορισμός της συνάρτησης.
!============================================================

ΣΥΝΑΡΤΗΣΗ Έκπτωση(x) :ΠΡΑΓΜΑΤΙΚΗ
ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: x
ΑΡΧΗ

   ΑΝ x > 5000 ΤΟΤΕ
      Έκπτωση ← x * 0.12

   ΑΛΛΙΩΣ_ΑΝ x > 1000
      Έκπτωση ← x * 0.05

   ΑΛΛΙΩΣ_ΑΝ x > 500
      Έκπτωση ← x * 0.02

   ΑΛΛΙΩΣ
      Έκπτωση ← 0

   ΤΕΛΟΣ_ΑΝ

ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Παράδειγμα 3: Ένας μαθηματικός επιθυμεί να γραφτεί ένα πρόγραμμα που θα τυπώνει τις τιμές της συνάρτησης $y=x^2-5x+1$ για όλες τις τιμές του x από 1 μέχρι 3 σε βήματα του 0.1.

```
ΠΡΟΓΡΑΜΜΑ Τιμές_συνάρτησης
ΜΕΤΑΒΛΗΤΕΣ
   ΠΡΑΓΜΑΤΙΚΕΣ: x
ΑΡΧΗ
```

```
   ΓΙΑ x ΑΠΟ 1 ΜΕΧΡΙ 3 ΜΕ_ΒΗΜΑ 0.1

      ΓΡΑΨΕ 'x=',x, '   y=', Y(x)

   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```




```
!=============================================================
!Ορισμός της συνάρτησης.
!=============================================================

ΣΥΝΑΡΤΗΣΗ Y(x) :ΠΡΑΓΜΑΤΙΚΗ
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ: x
ΑΡΧΗ

   Y ← x * x - 5 * x + 1

ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Παράδειγμα 4: Ένα ασφαλιστικό ταμείο κρατάει σε έναν πίνακα τα έτη γέννησης των 5000 ασφαλισμένων του. Ένας εργαζόμενος μόλις συμπληρώσει το 67ο έτος της ηλικίας του πρέπει να αποχωρήσει από την ενεργό δράση και να συνταξιοδοτηθεί υποχρεωτικά.
Γράψτε ένα πρόγραμμα που διαβάζει σε έναν πίνακα τα έτη γέννησης των 5000 ασφαλισμένων του. Στη συνέχεια, να διαβάζει το τρέχον έτος και να υπολογίζει πόσοι από αυτούς πρέπει να συνταξιοδοτηθούν. Ο υπολογισμός του πλήθους των ασφαλισμένων που πρέπει να συνταξιοδοτηθούν να γίνει με συνάρτηση.

Το πρόβλημα μπορεί να χωριστεί σε 3 υπο-προβλήματα:

1. Στο διάβασμα των ετών γέννησης των 5000 ασφαλισμένων σε έναν πίνακα
2. Στο διάβασμα του τρέχοντος έτους
3. Στον υπολογισμό του πλήθους των ασφαλισμένων που πρέπει να συνταξιοδοτηθούν, εφόσον έχουν ξεπεράσει το 67ο έτος της ηλικίας τους.

Για τα (1) και (3) θα δημιουργήσουμε σχετικά υποπρογράμματα. Για το (1) μία διαδικασία και για το (3) μία συνάρτηση (*επιστρέφει ένα αποτέλεσμα: το πλήθος των ασφαλισμένων προς συνταξιοδότηση*).

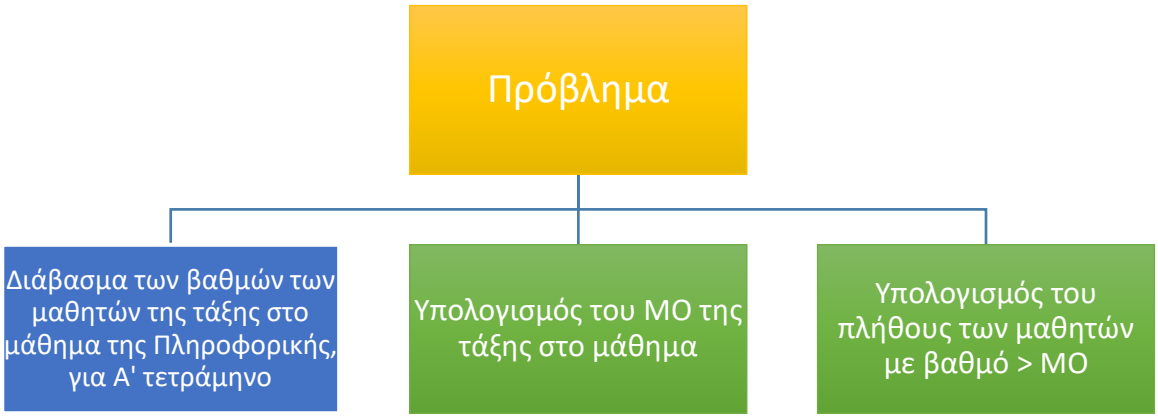
```
ΠΡΟΓΡΑΜΜΑ Συνταξιοδότηση_ασφαλισμένων
ΜΕΤΑΒΛΗΤΕΣ
   ΑΚΕΡΑΙΕΣ: Ασφαλισμένος[5000], Τρέχον_έτος
ΑΡΧΗ
```

```
   !Κλήση της διαδικασίας για διάβασμα των δεδομένων, εδώ των
   !ετών ηλικίας των ασφαλισμένων. Παράμετρος είναι ο πίνακας
   !που θα γεμίσει με δεδομένα.
   Διάβασμα_ετών_ηλικίας_ασφαλισμένων(Ασφαλισμένος)

   !Διάβασμα του τρέχοντος έτους. Θα μπορούσαμε να κάνουμε
   !έλεγχο εγκυρότητας.
   ΓΡΑΨΕ 'Παρακαλώ, δώστε το τρέχον έτος'
   ΔΙΑΒΑΣΕ Τρέχον_έτος

   !Κλήση της συνάρτησης υπολογισμού των προς συνταξιοδότηση
   !ασφαλισμένων, με 2 παραμέτρους:
   !α) Τον πίνακα των ασφαλισμένων
   !β) Το τρέχον έτος
   ΓΡΑΨΕ 'Το πλήθος των ασφαλισμένων προς συνταξιοδότηση
                                                & είναι:'
   ΓΡΑΨΕ Πλήθος_προς_συνταξιοδότηση(Ασφαλισμένος,Τρέχον_έτος)

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

!=============================================================
!Ορισμός της διαδικασίας που διαβάζει τα δεδομένα στον πίνακα.
!=============================================================

ΔΙΑΔΙΚΑΣΙΑ Διάβασμα_ετών_ηλικίας_ασφαλισμένων(Π)
ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: Π[5000], i
ΑΡΧΗ
   !Θα μπορούσαμε να κάνουμε έλεγχο εγκυρότητας.
   ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 5000

       ΓΡΑΨΕ 'Παρακαλώ, δώστε το έτος ηλικίας του ',i,
                                        & 'ου ασφαλισμένου
       ΔΙΑΒΑΣΕ Π[i]

   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

!=============================================================
!Ορισμός της συνάρτησης.  Λαμβάνει ως δεδομένα τον πίνακα των
!ασφαλισμένων και το τρέχον έτος. Επιστρέφει το πλήθος αυτών
!που υπερβαίνουν το 67ο έτος της ηλικίας τους.
!=============================================================

ΣΥΝΑΡΤΗΣΗ Πλήθος_προς_συνταξιοδότηση(Π, Τρέχον_έτος) :ΑΚΕΡΑΙΗ
ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: Π[5000], Τρέχον_έτος, i, Μετρητής
ΑΡΧΗ
```

```
    !Αρχική τιμή του μετρητή, ο οποίος κρατάει πόσοι είναι πάνω
    !από 67.
    Μετρητής ← 0

    !Σαρώνουμε τον πίνακα των ασφαλισμένων και όποιος έχει έτος
    !ηλικίας > 67 τότε αυξάνουμε τον μετρητή κατά 1.
    ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 5000

        ΑΝ Π[i] > 67 ΤΟΤΕ
           Μετρητής ← Μετρητής + 1
        ΤΕΛΟΣ_ΑΝ

        !Εκχώρηση της τελικής τιμής του μετρητή στο όνομα της
        !συνάρτησης.
        Πλήθος_προς_συνταξιοδότηση ← Μετρητής


    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Παράδειγμα 5: Σε ένα Λύκειο, θέλουμε να πληροφορηθούμε πόσοι μαθητές της Γ΄ τάξης έχουν βαθμό επίδοσης πάνω από τον μέσο όρο στο μάθημα της Πληροφορικής, για το Α' τετράμηνο. Γράψτε ένα πρόγραμμα όπου:
α) Θα διαβάζει τους βαθμούς των 65 μαθητών, στο μάθημα της Πληροφορικής στο Α΄ τετράμηνο. Κατά το διάβασμα να ελέγχεται αν ο βαθμός είναι έγκυρος (μεταξύ 1-20)
β) Θα υπολογίζει ποιός είναι ο μέσος όρος της Γ' τάξης στο μάθημα αυτό.
γ) Θα υπολογίζει πόσοι μαθητές κατάφεραν βαθμολογία πάνω από αυτόν τον μέσο όρο.

Το πρόβλημα μπορούμε να το σχηματοποιήσουμε σε υπο-προβλήματα ως εξής:



Καθένα υπο-πρόβλημα θα υλοποιηθεί με αντίστοιχο υποπρόγραμμα. Συγκεκριμένα, το **πρώτο με διαδικασία** και **τα άλλα δύο με συναρτήσεις**. Επιπλέον, θα δημιουργήσουμε μία λογική συνάρτηση που θα λαμβάνει έναν βαθμό και θα επιστρέφει ΑΛΗΘΗΣ (αν είναι έγκυρος) ή ΨΕΥΔΗΣ (δεν είναι έγκυρος).

```
ΠΡΟΓΡΑΜΜΑ Μαθητές_πάνω_από_τον_μέσο_όρο
ΜΕΤΑΒΛΗΤΕΣ
   ΠΡΑΓΜΑΤΙΚΕΣ: Βαθμός[65], ΜΟ
   ΑΚΕΡΑΙΕΣ: Πλήθος_άνω_ΜΟ
ΑΡΧΗ
   !Κλήση της διαδικασίας για διάβασμα των δεδομένων, εδώ των
   !βαθμών των μαθητών. Παράμετρος είναι ο πίνακας
   !που θα γεμίσει με δεδομένα.
   Διάβασμα_βαθμών(Βαθμός)

   !Κλήση της συνάρτησης για υπολογισμό του μέσου όρου (ΜΟ) των
   !βαθμών. Παράμετρος εισόδου είναι ο πίνακας των βαθμών.
   ΜΟ ← ΜΟ_βαθμών(Βαθμός)

   !Κλήση της συνάρτησης για υπολογισμό του πλήθους των μαθητών
   !με βαθμό άνω του μέσου όρου. Παράμετροι εισόδου είναι δύο:
   !α) Ο πίνακας των βαθμών
   !β) Ο ΜΟ
   Πλήθος_άνω_ΜΟ ← Πλήθος_μαθητών_με_βαθμό_άνω_ΜΟ(Βαθμός, ΜΟ)

   !Πληροφόρηση επί της οθόνης με κατάλληλα μηνύματα.
   ΓΡΑΨΕ 'Ο μέσος όρος της βαθμολογίας είναι ', ΜΟ
```

```
   ΓΡΑΨΕ 'Πλήθος μαθητών άνω του μέσου όρου ', Πλήθος_άνω_ΜΟ

ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

!=============================================================
!Ορισμός της διαδικασίας που διαβάζει τα δεδομένα στον πίνακα.
!=============================================================

ΔΙΑΔΙΚΑΣΙΑ Διάβασμα_βαθμών(Π)
ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: Π[65], i
ΑΡΧΗ
  ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 65

     !Με έλεγχο εγκυρότητας.
     ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
       ΓΡΑΨΕ 'Παρακαλώ, δώστε τον βαθμό του ',i,
                                          & 'ου μαθητή
       ΔΙΑΒΑΣΕ Π[i]
     ΜΕΧΡΙΣ_ΟΤΟΥ Έγκυρος(Π[i])

   ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ

!=============================================================
!Ορισμός της συνάρτησης που ελέγχει την εγκυρότητα του
!εισαγόμενου βαθμού.
!=============================================================

ΣΥΝΑΡΤΗΣΗ Έγκυρος(x) :ΛΟΓΙΚΗ
ΜΕΤΑΒΛΗΤΕΣ
     ΠΡΑΓΜΑΤΙΚΕΣ: x
ΑΡΧΗ
     Έγκυρος ← x >= 1 ΚΑΙ x <= 20

ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ

!=============================================================
!Ορισμός της συνάρτησης που υπολογίζει τον μέσο όρο των βαθμών.
!Τα δεδομένα που μεταβιβάζονται στη συνάρτηση (παράμετρος
!εισόδου) είναι ο πίνακας με τους βαθμούς.
!=============================================================

ΣΥΝΑΡΤΗΣΗ ΜΟ_βαθμών(Π) :ΠΡΑΓΜΑΤΙΚΗ
ΜΕΤΑΒΛΗΤΕΣ
     ΠΡΑΓΜΑΤΙΚΕΣ: Π[65], Sum
     ΑΚΕΡΑΙΕΣ: i
ΑΡΧΗ
    !Αρχική τιμή στον αθροιστή βαθμών.
```

```
    Sum ← 0

    !Σάρωση του πίνακα και κάθε βαθμός προστίθεται στον
    !αθροιστή βαθμών.
    ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 65

       Sum ← Sum + Π[i]

    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

    !Υπολόγισε τον μέσο όρο και εκχώρησέ τον στο όνομα της
    !συνάρτησης.
    ΜΟ_βαθμών ← Sum / 65

ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ


!=============================================================
!Ορισμός της συνάρτησης που υπολογίζει το πλήθος των βαθμών που
!είναι άνω του μέσου όρου. Τα δεδομένα που μεταβιβάζονται στη
!συνάρτηση είναι ο πίνακας με τους βαθμούς και ο μέσος όρος.
!=============================================================

ΣΥΝΑΡΤΗΣΗ Πλήθος_μαθητών_με_βαθμό_άνω_ΜΟ(Π, ΜΟ) :ΑΚΕΡΑΙΗ
ΜΕΤΑΒΛΗΤΕΣ
     ΠΡΑΓΜΑΤΙΚΕΣ: Π[65], ΜΟ
     ΑΚΕΡΑΙΕΣ: i, μετρητής
ΑΡΧΗ
    !Αρχική τιμή στον μετρητή βαθμών.
    μετρητής ← 0

    !Σάρωση του πίνακα και κάθε βαθμός που είναι άνω του μέσου
    !όρου προσθέτει 1 στον μετρητή.
    ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 65

       ΑΝ Π[i] > ΜΟ ΤΟΤΕ
           μετρητής ← μετρητής + 1
       ΤΕΛΟΣ_ΑΝ

    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

    !Εκχώρησε την τιμή του μετρητή στο όνομα της συνάρτησης.
    Πλήθος_μαθητών_με_βαθμό_άνω_ΜΟ ← μετρητής

ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Πραγματικές και τυπικές παράμετροι

Μεταξύ του κυρίως προγράμματος και των υποπρογραμμάτων γίνεται ανταλλαγή τιμών μέσω των παραμέτρων.

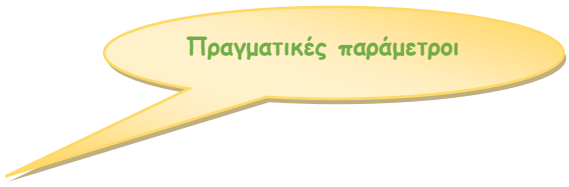
Πραγματικές παράμετροι: Είναι οι τιμές που μεταβιβάζονται *κατά την κλήση του υποπρογράμματος*.

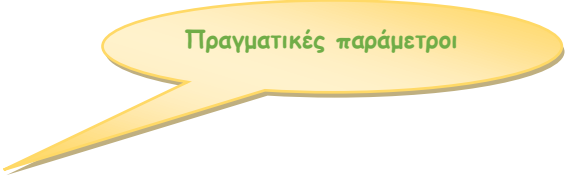
Τυπικές παράμετροι: Είναι οι μεταβλητές που *ορίζονται στην επικεφαλίδα του υποπρογράμματος*, εντός των παρενθέσεων.[4]

Ας δούμε το επόμενο παράδειγμα, όπου ορίζουμε τη διαδικασία `Υπολογισμοί` να δέχεται δύο πραγματικούς αριθμούς κι επιστρέφει το άθροισμα και τον μέσο όρο τους. Στο κυρίως πρόγραμμα καλούμε τη διαδικασία `Υπολογισμοί` 2 φορές:

- Την 1η φορά με τιμές `5` και `3` (και μηδενικές τιμές στο άθροισμα και μέσο όρο)
- Την 2η φορά με τιμές `8` και `12` (και μηδενικές τιμές στο άθροισμα και μέσο όρο)

```
ΠΡΟΓΡΑΜΜΑ Αθροισμα_και_μέσος_όρος
ΜΕΤΑΒΛΗΤΕΣ
   ΠΡΑΓΜΑΤΙΚΕΣ: x, y, Sum, MO
ΑΡΧΗ

   !Για την 1η κλήση.
   x ← 5
   y ← 3
   Sum ← 0
   MO ← 0
   ΚΑΛΕΣΕ Υπολογισμοί(x, y, Sum, MO)
   ΓΡΑΨΕ Sum, MO

   !Για την 2η κλήση.
   x ← 8
   y ← 12
   Sum ← 0
   MO ← 0
   ΚΑΛΕΣΕ Υπολογισμοί(x, y, Sum, MO)
   ΓΡΑΨΕ Sum, MO
```



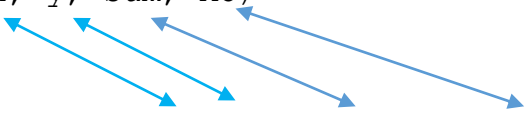


[4] Λέγονται, επίσης, και ορίσματα (arguments).

```
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

!=============================================================
!Ορισμός της διαδικασίας. Η επικεφαλίδα περιλαμβάνει τις
!ΤΥΠΙΚΕΣ ΠΑΡΑΜΕΤΡΟΥΣ, οι οποίες είναι μεταβλητές που ορίζονται
!τοπικά εντός της διαδικασίας.
!=============================================================

ΔΙΑΔΙΚΑΣΙΑ Υπολογισμοί(α, β, Άθροισμα, Μέσος)
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ: α, β, Άθροισμα, Μέσος
ΑΡΧΗ
    Άθροισμα ← α + β
    Μέσος ← (α + β ) / 2

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Στην **1η κλήση** οι τιμές των x, y, Sum και MO είναι οι πραγματικές παράμετροι. Οι τιμές τους είναι 5, 3, 0 και 0 αντίστοιχα. Αυτές οι τιμές μεταβιβάζονται στις αντίστοιχες τυπικές παραμέτρους α, β, Άθροισμα και Μέσος της διαδικασίας.

```
x ← 5
y ← 3
Sum ← 0
MO ← 0
ΚΑΛΕΣΕ Υπολογισμοί(x, y, Sum, MO)
….

        ΔΙΑΔΙΚΑΣΙΑ Υπολογισμοί(α, β, Άθροισμα, Μέσος)
```



Στο κυρίως πρόγραμμα, κατά την κλήση της διαδικασίας:

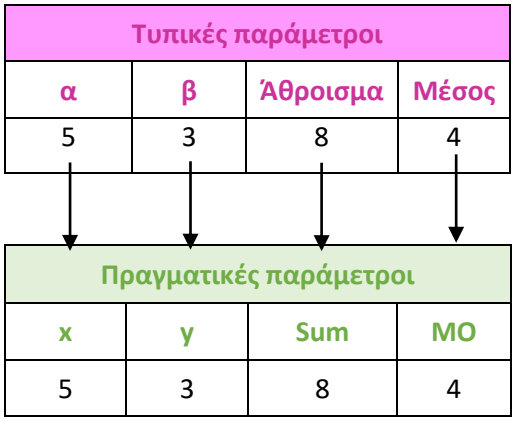
Πραγματικές παράμετροι			
x	y	Sum	MO
5	3	0	0

Στην διαδικασία, λαμβάνονται οι τιμές:

Τυπικές παράμετροι			
α	β	Άθροισμα	Μέσος
5	3	0	0

Όταν η ροή εκτέλεσης επιστρέφει στο κυρίως πρόγραμμα και η διαδικασία επιστρέφει αποτελέσματα μέσω των παραμέτρων της:

Στην περίπτωση των `x, y` δεν αλλάζουν οι τιμές, ενώ οι τιμές των `Sum` και `MO` έχουν αλλάξει, πράγμα που σημαίνει ότι μέσω αυτών των παραμέτρων επιστρέφονται αποτελέσματα.



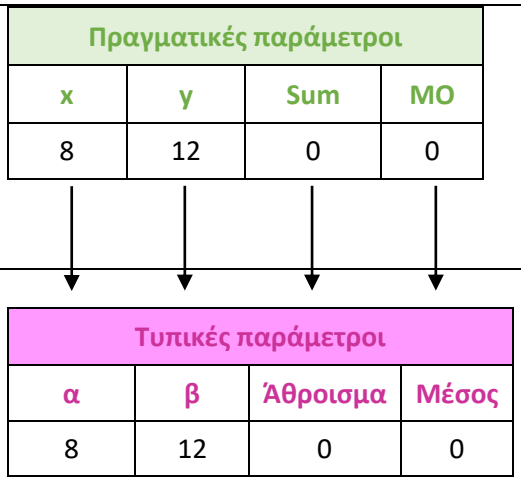
Στην **2η κλήση** οι τιμές των `x, y, Sum` και `MO` είναι `8, 12, 0` και `0` αντίστοιχα. Αυτές οι τιμές μεταβιβάζονται στις αντίστοιχες τυπικές παραμέτρους `α, β, Άθροισμα` και `Μέσος` της διαδικασίας.

```
x ← 8
y ← 12
Sum ← 0
MO ← 0
ΚΑΛΕΣΕ Υπολογισμοί(x, y, Sum, MO)
….
```

```
ΔΙΑΔΙΚΑΣΙΑ Υπολογισμοί(α, β, Άθροισμα, Μέσος)
```

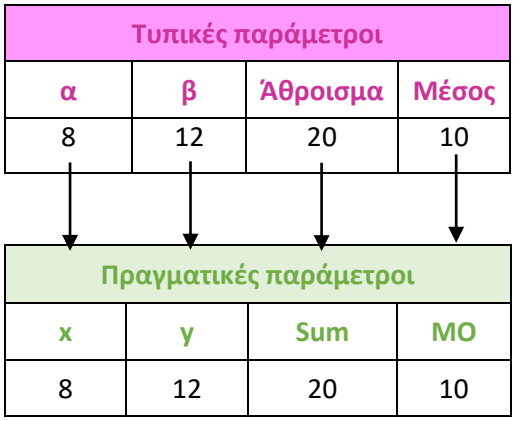
Στο κυρίως πρόγραμμα, κατά την κλήση της διαδικασίας:

Στην διαδικασία, λαμβάνονται οι τιμές:



Όταν η ροή εκτέλεσης επιστρέφει στο κυρίως πρόγραμμα και η διαδικασία επιστρέφει αποτελέσματα μέσω των παραμέτρων της:

Στην περίπτωση των `x, y` δεν αλλάζουν οι τιμές, ενώ οι τιμές των `Sum` και `ΜΟ` έχουν αλλάξει, πράγμα που σημαίνει ότι μέσω αυτών των παραμέτρων επιστρέφονται αποτελέσματα.



Παρατηρήσεις:

- **Ο αριθμός των πραγματικών και τυπικών παραμέτρων πρέπει να είναι ο ίδιος.**

- **Κάθε πραγματική παράμετρος αντιστοιχεί σε μία τυπική παράμετρο που βρίσκεται στην αντίστοιχη θέση.** Για παράδειγμα, η πρώτη πραγματική παράμετρος (*στην κλήση του υποπρογράμματος*) στην πρώτη τυπική παράμετρο (*στην επικεφαλίδα του υποπρογράμματος*) κ.ο.κ.

- **Τα ονόματα των πραγματικών και τυπικών παραμέτρων δεν είναι υποχρεωτικό να είναι ίδια**. Για παράδειγμα, μία τυπική παράμετρος μπορεί να έχει όνομα `α` και η αντίστοιχη πραγματική παράμετρος το όνομα `x`.

- **Προσοχή χρειάζεται στον τύπο των παραμέτρων**. Αν, για παράδειγμα, μία τυπική παράμετρος είναι τύπου `ΑΚΕΡΑΙΟΣ` τότε και η αντίστοιχη πραγματική παράμετρος πρέπει να είναι τύπου `ΑΚΕΡΑΙΟΣ`.

- **Οι τυπικές παράμετροι είναι τοπικές μεταβλητές, δηλαδή δηλώνονται και *ισχύουν*[5] μόνο στο υποπρόγραμμα. Αντίστοιχα, οι πραγματικές παράμετροι είναι μεταβλητές που δηλώνονται και *ισχύουν* μόνο στο κυρίως πρόγραμμα.**
  Με άλλα λόγια, αν σε ένα υποπρόγραμμα έχει δηλωθεί μία μεταβλητή με όνομα `α` τότε αυτή μπορεί να χρησιμοποιηθεί μόνο εντός του υποπρογράμματος. Το κυρίως πρόγραμμα ή άλλα υποπρογράμματα δεν γνωρίζουν την ύπαρξή της. Έτσι, διατηρείται η ανεξαρτησία των διαφόρων τμημάτων του προγράμματος και η επικοινωνία τους γίνεται μέσω των παραμέτρων.

- **Συμβατικά, μπορούμε να ξεχωρίσουμε τις παραμέτρους σε εισόδου και εξόδου.** Ως εισόδου θεωρούμε αυτές που μεταβιβάζουν στο υποπρόγραμμα τα απαραίτητα για

[5] Το χαρακτηριστικό αυτό ονομάζεται ***εμβέλεια (scope)*** και καθορίζει σε ποιά τμήματα μία μεταβλητή είναι ορατή, δηλαδή μπορεί να χρησιμοποιηθεί. Στη `ΓΛΩΣΣΑ`, μία μεταβλητή έχει εμβέλεια μόνο στο τμήμα που δηλώνεται (κυρίως πρόγραμμα ή υποπρόγραμμα).

την επεξεργασία δεδομένα. Ως εξόδου θεωρούμε αυτές που μεταβιβάζουν τιμές πίσω στο κυρίως πρόγραμμα, δηλαδή επιστρέφουν αποτελέσματα.

Εμβέλεια μεταβλητών-σταθερών

Κάθε τμήμα προγράμματος (κυρίως πρόγραμμα ή υποπρόγραμμα) χρησιμοποιεί τις δικές του μεταβλητές και σταθερές. Στη ΓΛΩΣΣΑ, αυτές **δηλώνονται και ισχύουν τοπικά**, δηλαδή μόνο στο συγκεκριμένο τμήμα.

Το πέρασμα τιμών από το ένα τμήμα στο άλλο γίνεται μέσω των παραμέτρων, κατά την κλήση του υποπρογράμματος και κατά το τέλος της εκτέλεσης του υποπρογράμματος.

Εμβέλεια μεταβλητής και σταθεράς = **Το τμήμα ή τα τμήματα προγράμματος στα οποία είναι ορατή, δηλαδή μπορεί να χρησιμοποιηθεί**.

Σε άλλες γλώσσες προγραμματισμού, η εμβέλεια των μεταβλητών/σταθερών είναι διαφορετική.

Έτσι, έχουμε συνολικά τις εξής περιπτώσεις:

Απεριόριστη (καθολική) εμβέλεια

Μία μεταβλητή/σταθερά μπορεί να χρησιμοποιηθεί σε όλα τα τμήματα ανεξαρτήτως που έχει δηλωθεί.
Αυτό έχει το μειονέκτημα ότι η ανάπτυξη πολλών υποπρογραμμάτων καθίσταται δύσκολη, αφού ο προγραμματιστής πρέπει να γνωρίζει όλα τα ονόματα των μεταβλητών που έχουν δηλωθεί σε άλλα υποπρογράμματα. Έτσι, μία αλλαγή στο όνομα μίας μεταβλητής επιφέρει την αναγκαστική αλλαγή του ονόματος σε όλα τα υποπρογράμματα που χρησιμοποιείται. Η αρχή της ανεξαρτησίας των διαφόρων τμημάτων ενός προγράμματος καταστρατηγείται.

Περιορισμένη εμβέλεια

Μία μεταβλητή/σταθερά μπορεί να χρησιμοποιηθεί μόνο τοπικά, δηλαδή στο τμήμα του προγράμματος που δηλώνεται.

Το πλεονέκτημα είναι ότι κάθε τμήμα είναι αυτόνομο και δεν επηρεάζεται η ονομασία των μεταβλητών/σταθερών από το γεγονός ότι το ίδιο όνομα μπορεί να έχει δοθεί σε άλλο τμήμα.
Η ΓΛΩΣΣΑ ακολουθεί τη στρατηγική της περιορισμένης εμβέλειας.

Μερικώς περιορισμένη εμβέλεια

Αποτελεί συνδυασμό των δύο προηγούμενων. Δηλαδή, άλλες μεταβλητές/σταθερές έχουν καθολική εμβέλεια και άλλες τοπική. Σε αυτήν την περίπτωση, χρειάζεται μεγάλη προσοχή στον χειρισμό των μεταβλητών/σταθερών κάτι που μπορεί να δυσκολέψει τον αρχάριο προγραμματιστή.

Ερωτήσεις κατανόησης

1. Τί είναι ο τμηματικός προγραμματισμός;
2. Ποιά είναι τα χαρακτηριστικά των υποπρογραμμάτων;
3. Περιγράψτε τα πλεονεκτήματα του τμηματικού προγραμματισμού.
4. Τί είναι η παράμετρος και ποιά η χρησιμότητά της;
5. Ένα υποπρόγραμμα ενδέχεται να έχει παραμέτρους και απλές μεταβλητές. Ποιά η διαφορά τους;
6. Στο παρακάτω υποπρόγραμμα, ποιές είναι οι παράμετροι και ποιές οι απλές μεταβλητές;

```
ΔΙΑΔΙΚΑΣΙΑ Μέσος_όρος(Π, ΜΟ)
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ: Π[20], Sum, ΜΟ
    ΑΚΕΡΑΙΕΣ: i
ΑΡΧΗ
    ….
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

7. Ο φίλος σας ισχυρίζεται ότι ένα υποπρόγραμμα πρέπει να έχει τουλάχιστον μία παράμετρο. Έχει δίκιο;

8. Πάλι ο φίλος σας ισχυρίζεται επίμονα ότι μία διαδικασία δεν είναι υποχρεωτικό να επιστρέψει κάποιο αποτέλεσμα όπως μία συνάρτηση. Έχει δίκιο;

9. Δίνεται η παρακάτω διαδικασία, η οποία δέχεται 3 αριθμούς κι επιστρέφει τον μέσο όρο τους:

```
ΔΙΑΔΙΚΑΣΙΑ Μέσος_όρος(x, y, z, ΜΟ)
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ: x,y,z,ΜΟ
ΑΡΧΗ
    ΜΟ ← (x + y + z) / 3
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Μετατρέψτε τη σε συνάρτηση.

10. Πώς καλείται μία διαδικασία και πώς μία συνάρτηση;

11. Δίνεται το παρακάτω τμήμα προγράμματος:

```
ΠΡΟΓΡΑΜΜΑ Αποτελέσματα
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ: x,y,z,Sum,ΜΟ
ΑΡΧΗ
    ΔΙΑΒΑΣΕ x, y, z
    ΚΑΛΕΣΕ Υπολογισμοί(x,y,z,Sum,ΜΟ)
    ΓΡΑΨΕ Sum, ΜΟ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΔΙΑΔΙΚΑΣΙΑ Υπολογισμοί(a,b,c,Sum,Average)
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ: a,b,c,Sum,Average
ΑΡΧΗ
    Sum ← a + b + c
    Average ← (a + b + c) / 3
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Υποδείξτε ποιές είναι οι πραγματικές και ποιές οι τυπικές παράμετροι.

12. Τί είναι η εμβέλεια (scope) μία μεταβλητής;

13. **α)** Πώς χαρακτηρίζεται μία μεταβλητή που έχει εμβέλεια μόνο στο τμήμα του προγράμματος στο οποίο δηλώνεται;

β) Πώς χαρακτηρίζεται μία μεταβλητή που έχει εμβέλεια σε όλα τα τμήματα του προγράμματος, ανεξάρτητα πού δηλώνεται;

14. Στο παρακάτω πρόγραμμα, εντοπίστε τα συντακτικά λάθη:

```
ΠΡΟΓΡΑΜΜΑ Υπολογισμός_εμβαδού_τριγώνου
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ: βάση, ύψος, Ε
ΑΡΧΗ
    ΔΙΑΒΑΣΕ βάση
    ΔΙΑΒΑΣΕ ύψος
    Ε ← Εμβαδόν_τριγώνου(βάση, ύψος)
    ΓΡΑΨΕ 'Το εμβαδόν τριγώνου είναι', Ε
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΣΥΝΑΡΤΗΣΗ Εμβαδόν_τριγώνου(β, υ, Ε)
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ: β, υ
    ΠΡΑΓΜΑΤΙΚΕΣ: Ε
ΑΡΧΗ
    Ε ← (β * υ) / 2
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

15. Στα μαθηματικά, ένας **ταυτοτικός πίνακας NxN** είναι εκείνος που όλα τα **στοιχεία της κυρίας διαγωνίου είναι ίσα με 1** και **όλα τα υπόλοιπα είναι ίσα με 0**, όπως φαίνεται στο παρακάτω σχήμα:

Π_{5x5} =

1	0	0	0	0
0	**1**	0	0	0
0	0	**1**	0	0
0	0	0	**1**	0
0	0	0	0	**1**

Ο πίνακας έχει δηλωθεί στο κυρίως πρόγραμμα ως:

```
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Π[5,5]
```

και υποθέτουμε ότι έχουμε διαβάσει όλα τα στοιχεία του.

Γράψτε μία **λογική συνάρτηση** με όνομα `Είναι_ταυτοτικός` που λαμβάνει ως είσοδο τον παραπάνω πίνακα `Π` κι επιστρέφει `ΑΛΗΘΗΣ` ή `ΨΕΥΔΗΣ` ανάλογα αν ο πίνακας αυτός είναι ταυτοτικός ή όχι.

Παράρτημα

Αναδρομή

Ως αναδρομή ορίζεται η δυνατότητα ενός υποπρογράμματος να καλεί τον εαυτό του.

Κάθε αναδρομικό υποπρόγραμμα περιλαμβάνει **δύο τμήματα**:

1. Την **αναδρομική σχέση**, η οποία περιλαμβάνει την συνεχή κλήση προς τον εαυτό του.
2. Την **συνθήκη τερματισμού** (ή τιμή βάσης), η οποία σταματάει την συνεχή κλήση προς τον εαυτό του.

Γενικά, κάθε επεξεργασία που περιέχει μία επαναληπτικότητα διατυπώνεται με τις συνήθεις εντολές επανάληψης αλλά μπορεί να διατυπωθεί, επίσης, με ένα αναδρομικό υποπρόγραμμα.

Παράδειγμα 1: Να γραφτεί συνάρτηση που υπολογίζει το άθροισμα των ζυγών αριθμών από μία αρχική μέχρι μία τελική τιμή. Η αρχική και τελική τιμή ζυγού θα δίνονται ως παράμετροι στη συνάρτηση.

1ο τρόπος: Με τυπική συνάρτηση που περιέχει εντολή επανάληψης.

Η επίλυση του συγκεκριμένου προβλήματος με την γνωστή εντολή επανάληψης **ΓΙΑ…ΑΠΟ…ΜΕΧΡΙ** με μία «κανονική» συνάρτηση είναι η εξής:

```
ΣΥΝΑΡΤΗΣΗ Sum(α, β):ΑΚΕΡΑΙΗ
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ: α, β, i, S
ΑΡΧΗ
    'Αρχική τιμή στον αθροιστή.
    S ← 0

    'Σε κάθε επανάληψη, η τιμή της i προστίθεται στον αθροιστή
    'Sum.
    ΓΙΑ i ΑΠΟ α ΜΕΧΡΙ β ΜΕ_ΒΗΜΑ 2

        S ← S + i
```

```
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

    'Εκχώρηση της τιμή του αθροιστή στο όνομα της συνάρτησης.
    Sum ← S

ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

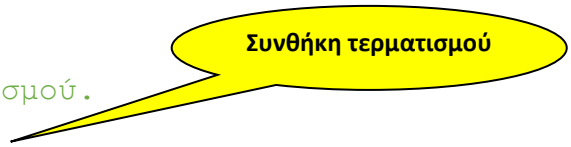
Μία κλήση της συνάρτησης μπορεί να είναι σε μία έκφραση π.χ.
`K ← 8 * Sum(2,10)` ή

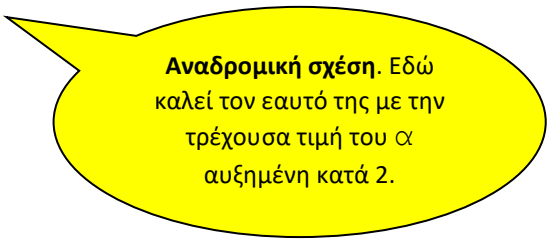
```
ΓΡΑΨΕ 'Το άθροισμα των ζυγών από το 2 μέχρι το 10 είναι',
                                        & Sum(2,10)
```

Στην παραπάνω κλήση `Sum(2,10)` η συνάρτηση επιστρέφει την τιμή `30`. Στην κλήση `Sum(6,20)` επιστρέφει `104`.

2ο τρόπος: Με αναδρομική συνάρτηση

```
ΣΥΝΑΡΤΗΣΗ Sum(α, β):ΑΚΕΡΑΙΗ
ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ: α, β
ΑΡΧΗ
    'Συνθήκη τερματισμού.
    ΑΝ α > β ΤΟΤΕ

        Sum ← 0

    ΑΛΛΙΩΣ
        'Αναδρομική σχέση. Εδώ καλεί τον εαυτό της περνώντας ως
        'παράμετρο την τρέχουσα τιμή του α συν 2 καθώς και την
        'τιμή β.
        Sum ← α + Sum(α + 2, β)

    ΤΕΛΟΣ_ΑΝ

ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```





Η αναδρομική συνάρτηση παρότι είναι πιο σύντομη είναι δυσκολότερη στην κατανόηση του τρόπου που λειτουργεί. Παρακάτω, θα εξηγήσουμε τον μηχανισμό λειτουργίας που περιλαμβάνει μία *στοίβα κλήσεων,* αφού η αναδρομική συνάρτηση καλεί συνεχώς τον εαυτό της μέχρι να ικανοποιηθεί η συνθήκη τερματισμού.

Έστω, ότι καλείται (1η κλήση) σε μία έκφραση ως εξής `Sum(2,10)`. Ο παρακάτω πίνακας εξηγεί τον μηχανισμό λειτουργίας των διαδοχικών κλήσεων, ξεκινώντας από τη βάση του

πίνακα:

Αριθμ.	Στοίβα κλήσεων	Επιστροφή τιμής από κλήση
	Τερματισμός	0
6η	`Sum ← 10 + Sum(12,10)`	`Sum ← 10 + 0`
5η	`Sum ← 8 + Sum(10,10)`	`Sum ← 8 + 10`
4η	`Sum ← 6 + Sum(8,10)`	`Sum ← 6 + 18`
3η	`Sum ← 4 + Sum(6,10)`	`Sum ← 4 + 24`
2η	`Sum ← 2 + Sum(4,10)`	`Sum ← 2 + 28`
1η	`Sum(2,10)`	`30`

Παρατηρούμε ότι η `Sum` θα καλέσει τον εαυτό της 5 φορές με διαφορετική τιμή του `α`. **Κάθε κλήση καταχωρείται σε μία στοίβα** (*με σειρά από κάτω προς τα πάνω στον πίνακα*). Όταν ικανοποιηθεί η συνθήκη τερματισμού, η κάθε κλήση επιστρέφει μία τιμή με αντίθετη σειρά (*από πάνω προς τα κάτω στον πίνακα. Άλλωστε η απώθηση από τη στοίβα γίνεται από την κορυφή, λειτουργία LIFO*). Έτσι, η τελευταία κλήση επιστρέφει `0`, η προτελευταία `10`, η επόμενη `18` κ.ο.κ

Άσκηση: Ο αναγνώστης μπορεί να κάνει τον πίνακα κλήσεων για τη συνάρτηση `Sum(6,20)`. Η επιστρεφόμενη τιμή πρέπει να είναι `104`.

Παράδειγμα 2: Να γραφτεί αναδρομική διαδικασία που κάνει δυαδική αναζήτηση ενός επωνύμου σε έναν μονοδιάστατο πίνακα που περιέχει σε αλφαβητική διάταξη τα επώνυμα των 7 μαθητών ενός μικρού group. Η διαδικασία θα δέχεται ως δεδομένα τον πίνακα με τα επώνυμα των μαθητών, το προς εύρεση επώνυμο κι επιστρέφει αν βρέθηκε ή όχι.

Για την δυαδική αναζήτηση που χρησιμοποιεί την τεχνική «*διαίρει και βασίλευε*» έχουμε αναφερθεί στο Κεφάλαιο 4. Επίσης, στο Κεφάλαιο 9 και στην ενότητα 'Αναζήτηση στοιχείου' είδαμε και την σχετική υλοποίηση στη `ΓΛΩΣΣΑ`. Εδώ, θα κατασκευάσουμε μία αντίστοιχη αναδρομική διαδικασία.

Θα χρειαστούμε τις εξής παραμέτρους:

- Τον πίνακα με τα επώνυμα των μαθητών. Θα την ονομάσουμε `Π`.
- Το προς εύρεση επώνυμο. Θα την ονομάσουμε `key`.
- Την επιστρεφόμενη λογική τιμή που προσδιορίζει αν βρέθηκε ή όχι. Θα την ονομάσουμε `βρέθηκε`.

- Δύο ακόμα παραμέτρους που προσδιορίζουν τα όρια εύρεσης (π.χ. πάνω μισό, κάτω μισό του πίνακα). Θα τις ονομάσουμε `i` (κάτω όριο) και `j` (πάνω όριο).

```
ΔΙΑΔΙΚΑΣΙΑ Δυαδική_αναζήτηση(Π, key, βρέθηκε, i, j)
ΜΕΤΑΒΛΗΤΕΣ
    ΧΑΡΑΚΤΗΡΕΣ: Π[7], key
    ΛΟΓΙΚΗ: βρέθηκε
    ΑΚΕΡΑΙΕΣ: i, j, m
ΑΡΧΗ
    'Εύρεση του μεσαίου στοιχείου.
    m ← (i + j) DIV 2

    'Συνθήκη τερματισμού.
    ΑΝ i > j ΤΟΤΕ
        βρέθηκε ← ΨΕΥΔΗΣ

    ΑΛΛΙΩΣ
        ΑΝ Π[m] = key ΤΟΤΕ
            βρέθηκε ← ΨΕΥΔΗΣ

        ΑΛΛΙΩΣ_ΑΝ Π[m] < key ΤΟΤΕ
            'Ψάξε στο δεύτερο μισό του πίνακα. Αναδρομική σχέση.
            ΚΑΛΕΣΕ Δυαδική_αναζήτηση(Π, key, βρέθηκε, m+1, j)

        ΑΛΛΙΩΣ
            'Ψάξε στο πρώτο μισό του πίνακα. Αναδρομική σχέση.
            ΚΑΛΕΣΕ Δυαδική_αναζήτηση(Π, key, βρέθηκε, i, m-1)

        ΤΕΛΟΣ_ΑΝ

    ΤΕΛΟΣ_ΑΝ

ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Συνθήκη τερματισμού

Αναδρομική σχέση

Αναδρομική σχέση

Έστω ότι στο κυρίως πρόγραμμα έχουμε ορίσει έναν πίνακα `ΜΑΘΗΤΗΣ[7]` που περιέχει τις εξής τιμές:

'ΑΛΕΞΙΟΥ'	'ΔΙΑΜΑΝΤΑΚΟΣ'	'ΕΛΕΥΘΕΡΙΟΥ'	'ΖΑΧΑΡΑΚΗΣ'	'ΙΩΑΝΝΟΥ'	'ΤΡΑΝΟΣ'	'ΥΦΑΝΤΙΔΟΥ'

και αναζητούμε το επώνυμο 'ΙΩΑΝΝΟΥ', που αποτελεί το `key`.

Επίσης, στο κυρίως πρόγραμμα έχουμε δηλώσει μία λογική μεταβλητή με όνομα `found`. Η 1η κλήση της διαδικασίας, από το κυρίως πρόγραμμα, θα ήταν:

```
ΚΑΛΕΣΕ Δυαδική_Αναζήτηση(ΜΑΘΗΤΗΣ, 'ΙΩΑΝΝΟΥ', found, 1, 7)
```

Ο παρακάτω πίνακας εξηγεί τον μηχανισμό λειτουργίας των διαδοχικών κλήσεων, ξεκινώντας από τη βάση του πίνακα:

Αριθμ.		Στοίβα κλήσεων		Επιστροφή τιμής από κλήση στην παράμετρο βρέθηκε	
4η	↑	m ← (5+5) DIV 2 , Π[5] περιέχει 'ΙΩΑΝΝΟΥ' Π[5] = 'ΙΩΑΝΝΟΥ' συνεπώς, βρέθηκε ← ΑΛΗΘΗΣ Τερματισμός		βρέθηκε ← ΑΛΗΘΗΣ	↓
3η		m ← (5+7) DIV 2 , Π[6] περιέχει 'ΤΡΑΝΟΣ' Π[6] > 'ΙΩΑΝΝΟΥ' συνεπώς, j ← m-1 **ΚΑΛΕΣΕ** Δυαδική_Αναζήτηση(ΜΑΘΗΤΗΣ, "ΙΩΑΝΝΟΥ", βρέθηκε, 5, 5)		ΑΛΗΘΗΣ	
2η		m ← (1+7) DIV 2 , Π[4] περιέχει 'ΖΑΧΑΡΑΚΗΣ' Π[4] < 'ΙΩΑΝΝΟΥ' συνεπώς, i ← m+1 **ΚΑΛΕΣΕ** Δυαδική_Αναζήτηση(ΜΑΘΗΤΗΣ, 'ΙΩΑΝΝΟΥ', βρέθηκε, 5, 7)		ΑΛΗΘΗΣ	
1η		**ΚΑΛΕΣΕ** Δυαδική_Αναζήτηση(ΜΑΘΗΤΗΣ, 'ΙΩΑΝΝΟΥ', βρέθηκε, 1, 7)		ΑΛΗΘΗΣ	

Έτσι, η τυπική παράμετρος βρέθηκε της διαδικασίας θα επιστρέψει την τιμή ΑΛΗΘΗΣ στην πραγματική παράμετρο found στο κυρίως πρόγραμμα.

Άσκηση: Ο αναγνώστης μπορεί να κάνει τον πίνακα κλήσεων για την αναζήτηση του μαθητή 'ΕΛΕΥΘΕΡΙΟΥ' και της μαθήτριας 'ΑΛΕΞΙΑΔΗ' (*Η επιστρεφόμενη τιμή της βρέθηκε στην πρώτη περίπτωση είναι ΑΛΗΘΗΣ και στην δεύτερη περίπτωση ΨΕΥΔΗΣ*).

Γενικά, **τα αναδρομικά υποπρογράμματα είναι πιο αργά στην εκτέλεση** σε σχέση με τις «κανονικές» επαναληπτικές διαδικασίες. **Προσθέτουν επιπλέον φόρτο στο σύστημα** λόγω του ότι ο μηχανισμός κλήσεων χρησιμοποιεί τη δομή της στοίβας για την αποθήκευση όλων των δεδομένων του υποπρογράμματος (τιμές των μεταβλητών) σε κάθε κλήση.

ΤΕΛΟΣ ΣΗΜΕΙΩΣΕΩΝ ΚΕΦΑΛΑΙΟΥ 10