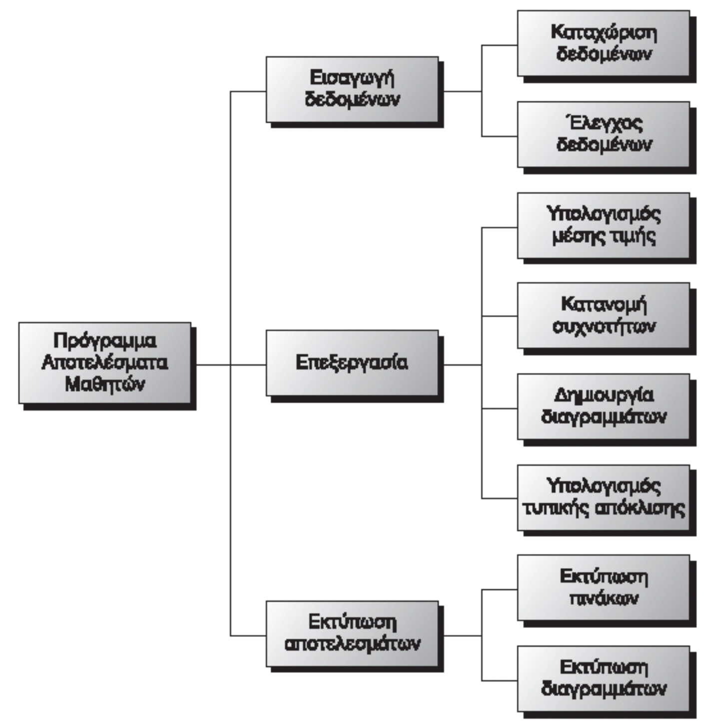
Κεφάλαιο 10 – Υποπρογράμματα

10.1 Τμηματικός προγραμματισμός

- Τμηματικός προγραμματισμός ονομάζεται η τεχνική σχεδίασης και ανάπτυξης των προγραμμάτων ως ένα σύνολο από απλούστερα τμήματα προγραμμάτων.
- Όταν ένα τμήμα προγράμματος επιτελεί ένα αυτόνομο έργο και έχει γραφεί χωριστά από το υπόλοιπο πρόγραμμα, τότε αναφερόμαστε σε **υποπρόγραμμα -** (subprogram).

10.2 Χαρακτηριστικά των υποπρογραμμάτων

- Κάθε υποπρόγραμμα έχει μόνο μία είσοδο (ενεργοποίησης) και μία μόνο έξοδο (απενεργοποίησης)
- Κάθε υποπρόγραμμα πρέπει να είναι ανεξάρτητο από τα άλλα (στο σχεδιασμό, την ανάπτυξη και τη συντήρηση).
- Κάθε υποπρόγραμμα πρέπει να μην είναι πολύ μεγάλο και να εκτελεί μία μόνο λειτουργία ➔ κατανοητό και ελεγχόμενο.



Σχ. 10.1 Διαγραμματική απεικόνιση της ανάλυσης προβλήματος

10.3 Πλεονεκτήματα του τμηματικού προγραμματισμού

- Διευκολύνει την ανάπτυξη του αλγορίθμου και του αντίστοιχου προγράμματος
- Διευκολύνει την κατανόηση και διόρθωση του προγράμματος
- Απαιτεί λιγότερο χρόνο και προσπάθεια στη συγγραφή του προγράμματος (κλήση υποπρογραμμάτων όπου χρειάζεται και όχι επανεγγραφή του κώδικα)
- Επεκτείνει τις δυνατότητες των γλωσσών προγραμματισμού (βιβλιοθήκες)

10.4 Παράμετροι

- **Παράμετρος** είναι μία μεταβλητή που επιτρέπει το πέρασμα της τιμής της από ένα τμήμα προγράμματος σε ένα άλλο
- Κάθε υποπρόγραμμα για να ενεργοποιηθεί καλείται, όπως λέγεται, από ένα άλλο υποπρόγραμμα ή το αρχικό πρόγραμμα, το οποίο ονομάζεται <u>κύριο πρόγραμμα</u>

10.5 Διαδικασίες και συναρτήσεις

- Η **συνάρτηση** είναι ένας τύπος υποπρογράμματος που υπολογίζει και επιστρέφει μόνο μία τιμή με το όνομά της (όπως οι μαθηματικές συναρτήσεις).
- Η **διαδικασία** είναι ένας τύπος υποπρογράμματος που μπορεί να εκτελεί όλες τις λειτουργίες ενός προγράμματος
- Οι <u>συναρτήσεις εκτελούνται</u> απλά με την εμφάνιση του <u>ονόματος</u> τους σε οποιαδήποτε <u>έκφραση</u>, ενώ για να εκτελεστούν οι <u>διαδικασίες</u> χρησιμοποιείται η ειδική εντολή <u>ΚΑΛΕΣΕ και το όνομα της διαδικασίας</u>

10.5.1 Ορισμός και κλήση συναρτήσεων

Κάθε συνάρτηση έχει την ακόλουθη δομή:

```
ΣΥΝΑΡΤΗΣΗ όνομα (λίστα παραμέτρων):τύπος συνάρτησης
Τμήμα δηλώσεων
ΑΡΧΗ
    ....
    όνομα <- έκφραση
    ...
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

10.5.2 Ορισμός και κλήση διαδικασιών

Κάθε διαδικασία έχει την ακόλουθη δομή:

```
ΔΙΑΔΙΚΑΣΙΑ Όνομα (λίστα παραμέτρων)
Τμήμα δηλώσεων
ΑΡΧΗ
    εντολές
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Δυνατά σημεία κλήσης μιας συνάρτησης (οπουδήποτε μπαίνει μία έκφραση):

- στο δεξί μέρος μιας εκχώρησης π.χ. x ← f(y)
- στην εντολή Γράψε π.χ. Γράψε f(x)
- σε μία λογική συνθήκη π.χ. Αν f(y) >= 0 τότε
- στη δομή επίλεξε π.χ. Επίλεξε f(y)
- στη λίστα παραμέτρων κλήσης συνάρτησης π.χ. T_P(f(y))
- σαν δείκτης αναφοράς σε στοιχείο πίνακα (για ακέραιες συναρτήσεις) π.χ. Γράψε A[f(y)]

- Η γενική μορφή της εντολής **ΚΑΛΕΣΕ** είναι:

  ΚΑΛΕΣΕ όνομα – διαδικασίας (λίστα – παραμέτρων)

- Λειτουργία: Η εκτέλεση του προγράμματος διακόπτεται και εκτελούνται οι εντολές της διαδικασίας που καλείται. Μετά το τέλος της διαδικασίας η εκτέλεση του προγράμματος συνεχίζεται από την εντολή που ακολουθεί . Η λίστα των παραμέτρων ορίζει τις τιμές που περνούν στη διαδικασία και τις τιμές που αυτή επιστρέφει. Η λίστα παραμέτρων δεν είναι υποχρεωτική.

Να γραφεί πρόγραμμα, το οποίο υπολογίζει το εμβαδό του κύκλου από την ακτίνα του.

```
ΠΡΟΓΡΑΜΜΑ Παράδειγμα_2
ΜΕΤΑΒΛΗΤΕΣ
    ΠΡΑΓΜΑΤΙΚΕΣ : R, E
ΑΡΧΗ
    ΚΑΛΕΣΕ Είσοδος_δεδομένων(R)
    E <- Εμβαδό_κύκλου(R)
    ΚΑΛΕΣΕ Εκτύπωση(E)
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

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

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

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

```
ΠΡΟΓΡΑΜΜΑ Ανταλλαγή
ΜΕΤΑΒΛΗΤΕΣ
 ΑΚΕΡΑΙΕΣ: α, β, ι, κ, temp
ΑΡΧΗ
 ΓΡΑΨΕ 'Δώσε τιμές για α και β'
 ΔΙΑΒΑΣΕ α, β
 temp <-- α
 α <-- β
 β <-- temp
 ΓΡΑΨΕ 'α=', α, 'β=', β
 ΓΡΑΨΕ 'Δώσε τιμές για ι και κ'
 ΔΙΑΒΑΣΕ  ι, κ
 temp <-- ι
 ι <-- κ
 κ <-- temp
 ΓΡΑΨΕ  'ι=', ι, 'κ=', κ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ Ανταλλαγή
```

Τμήμα προγράμματος που μπορεί να εκτελεστεί αυτοτελώς.

Μάλιστα επαναλαμβάνεται δύο φορές σε διαφορετικά σημεία στο πρόγραμμα.

ΠΡΟΓΡΑΜΜΑ Ανταλλαγή ΜΕΤΑΒΛΗΤΕΣ ΑΚΕΡΑΙΕΣ:α, β, ι, κ ΑΡΧΗ ΔΙΑΒΑΣΕ α, β ΚΑΛΕΣΕ Εναλλαγή_τιμών(α β) ΓΡΑΨΕ α, β ΔΙΑΒΑΣΕ ι, κ ΚΑΛΕΣΕ Εναλλαγή_τιμών(ι ,κ) ΓΡΑΨΕ ι, κ ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ Ανταλλαγή	Κύριο πρόγραμμα, που εκτελεί δύο φορές κατά σειρά τις εξής λειτουργίες: • *Διαβάζει σε δύο μεταβλητές, δύο ακέραιους αριθμούς.* • *Εναλλάσσει τις τιμές μεταξύ των μεταβλητών (διαδοχική κλήση διαδικασίας).* • *Εμφανίζει το περιεχόμενο των δύο μεταβλητών.*
ΔΙΑΔΙΚΑΣΙΑ Εναλλαγή_τιμών(κ, λ) ΜΕΤΑΒΛΗΤΕΣ ΑΚΕΡΑΙΕΣ:κ, λ,temp ΑΡΧΗ temp <- κ κ <- λ λ <- temp ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ Εναλλαγή_τιμών	*Διαδικασία που εναλλάσσει τις τιμές δύο μεταβλητών (κ,λ) μέσω της βοηθητικής μεταβλητής temp.*

10.5.3 Πραγματικές και τυπικές παράμετροι

Παράδειγμα:

Να γραφεί μια διαδικασία η οποία δέχεται στην είσοδο δύο τιμές και υπολογίζει και επιστρέφει το άθροισμα και τη διαφορά τους.

Οι μεταβλητές Α, Β, Διαφ1, Αθρ1, Α,Β, Διαφ2, Αθρ2 είναι μεταβλητές του προγράμματος Παράδειγμα_3 και αποτελούν τις **πραγματικές** παραμέτρους, ενώ οι μεταβλητές Χ,Υ, Διαφορά, Άθροισμα είναι μεταβλητές της διαδικασίας Πράξεις, και ονομάζονται **τυπικές** παράμετροι.

Όλες οι μεταβλητές ισχύουν **τοπικά** μόνο για το τμήμα του προγράμματος στο οποίο έχουν δηλωθεί!

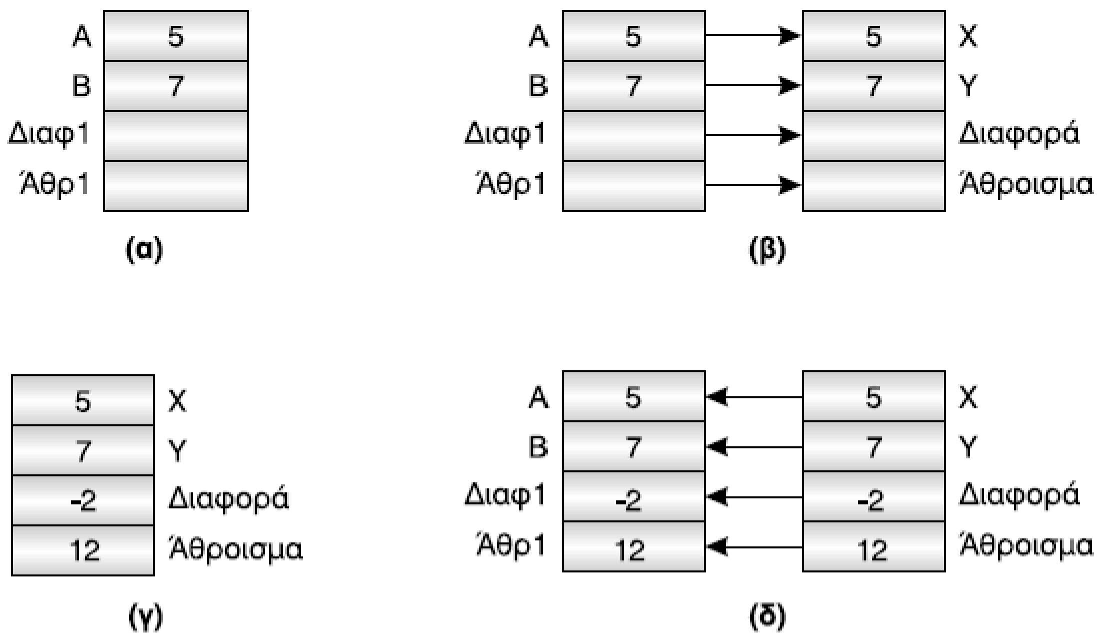
Μερικές γλώσσες προγραμματισμού ονομάζουν ορίσματα τις τυπικές παραμέτρους και απλά παραμέτρους τις πραγματικές

```
ΠΡΟΓΡΑΜΜΑ Παράδειγμα_3
...
Α<-5
Β<-7
ΚΑΛΕΣΕ Πράξεις(Α, Β, Διαφ1, Αθρ1)
...
Α<-9
Β<-6
ΚΑΛΕΣΕ Πράξεις(Α, Β, Διαφ2, Αθρ2)
...

          ΔΙΑΔΙΚΑΣΙΑ Πράξεις(Χ, Υ, Διαφορά, Άθροισμα)

          ΜΕΤΑΒΛΗΤΕΣ
             ΠΡΑΓΜΑΤΙΚΕΣ : Χ, Υ, Διαφορά, Άθροισμα

          ΑΡΧΗ
             Διαφορά  <-  Χ-Υ
             Άθροισμα <-  Χ+Υ

          ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Η λίστα των τυπικών παραμέτρων (formal parameter list) καθορίζει τις παραμέτρους στη δήλωση του υποπρογράμματος.

Η λίστα των πραγματικών παραμέτρων (actual parameter list) καθορίζει τις παραμέτρους στην κλήση του υποπρογράμματος.

10.5.3 Πραγματικές και τυπικές παράμετροι



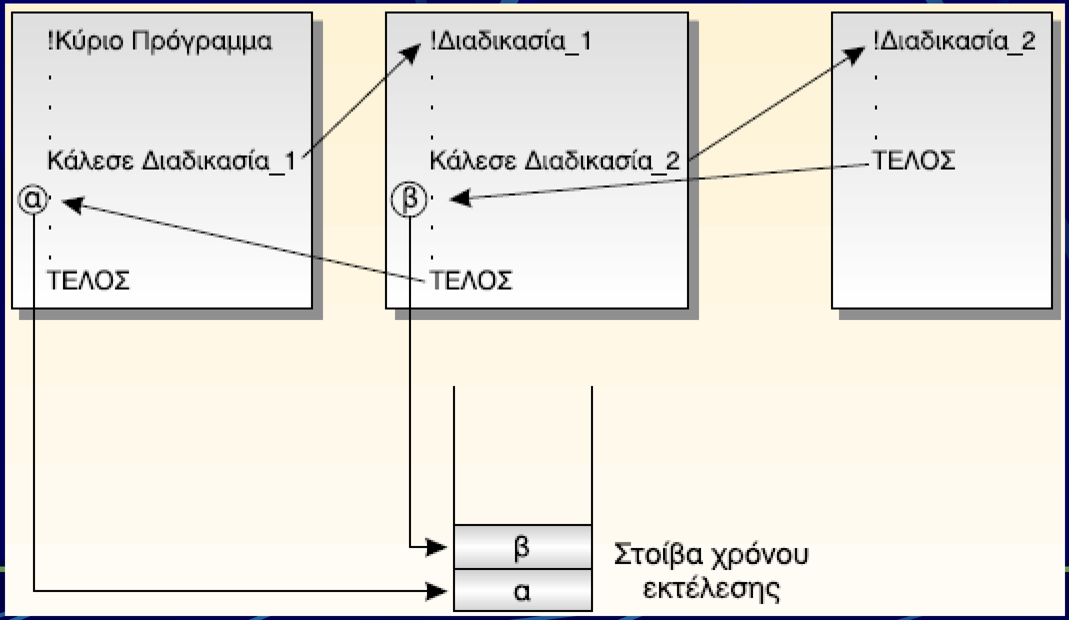
Σχ. 10.2. *Πέρασμα παραμέτρων κατά την κλήση διαδικασιών (α) Κατάσταση πριν την κλήση (β) Μεταβίβαση τιμών των μεταβλητών Α και Β στις Χ και Υ αντίστοιχα (γ) Στη διαδικασία εκχωρούνται τιμές στις μεταβλητές Διαφορά και 'Αθροισμα (δ) Οι τιμές των τελευταίων επιστρέφονται στις Διαφ1 και Αθρ1 μετά το τέλος της διαδικασίας.*

10.5.3 Πραγματικές και τυπικές παράμετροι

Η χρήση στοίβας στην κλήση υποπρογραμμάτων

Όταν μία διαδικασία ή συνάρτηση καλείται από το κύριο πρόγραμμα, τότε η αμέσως επόμενη διεύθυνση του κύριου προγράμματος, που ονομάζεται **διεύθυνση επιστροφής** (return address), αποθηκεύεται από το μεταφραστή σε μία στοίβα που ονομάζεται **στοίβα χρόνου εκτέλεσης** (execution time stack). Μετά την εκτέλεση της διαδικασίας ή της συνάρτησης η διεύθυνση επιστροφής απωθείται από τη στοίβα και έτσι ο έλεγχος του προγράμματος μεταφέρεται και πάλι στο κύριο πρόγραμμα. Η τεχνική αυτή εφαρμόζεται και γενικότερα, δηλαδή οποτεδήποτε μία διαδικασία ή συνάρτηση καλεί μία διαδικασία ή συνάρτηση. Για παράδειγμα, έστω ότι μία διαδικασία a καλεί τη διαδικασία b, που με τη σειρά της καλεί τη διαδικασία c κοκ. Στην περίπτωση αυτή οι διευθύνσεις επιστροφής εμφανίζονται στη στοίβα με σειρά c, b, a. Μετά την εκτέλεση κάθε διαδικασίας, η διεύθυνση επιστροφής απωθείται από τη στοίβα και ο έλεγχος μεταβιβάζεται στη διεύθυνση αυτή. Το παράδειγμα αυτό δείχνει μία από τις πολλές χρησιμότητες της LIFO ιδιότητας της στοίβας.

Η στοίβα χρόνου εκτέλεσης περιέχει τόσα στοιχεία όσα και τα υποπρογράμματα που έχουν κληθεί αλυσιδωτά



Οι λίστες των παραμέτρων πρέπει να ακολουθούν τους εξής **κανόνες**:

- Ο αριθμός των πραγματικών και των τυπικών παραμέτρων πρέπει να είναι ίδιος.
- Κάθε πραγματική παράμετρος αντιστοιχεί στην τυπική παράμετρο που βρίσκεται στην αντίστοιχη θέση. Για παράδειγμα η πρώτη της λίστας των τυπικών παραμέτρων στην πρώτη της λίστας των πραγματικών παραμέτρων κοκ.
- Η τυπική παράμετρος και η αντίστοιχη της πραγματική πρέπει να είναι του ιδίου τύπου.

Όταν ένα υποπρόγραμμα πρέπει να:

- επιστρέψει παραπάνω από μία τιμές
- αλλάξει τις τιμές των πραγματικών παραμέτρων
- εκτελέσει λειτουργίες εισαγωγής (Διάβασε) ή εξαγωγής (Γράψε)

τότε επιλέγουμε να το υλοποιήσουμε με **ΔΙΑΔΙΚΑΣΙΑ** μιας και μία **ΣΥΝΑΡΤΗΣΗ**: επιστρέφει μία τιμή, δεν αλλάζει τις τιμές των πραγματικών παραμέτρων, δεν χρησιμοποιείται για λειτουργίες εισαγωγής / εξαγωγής και δεν μπορεί να καλέσει διαδικασία.

Η πραγματική μεταβλητή μιας διαδικασίας **ΔΕΝ** μπορεί να είναι σταθερά ή έκφραση.

10.6 Εμβέλεια μεταβλητών-σταθερών

Κάθε κύριο πρόγραμμα όπως και κάθε υποπρόγραμμα περιλαμβάνει τις δικές του μεταβλητές και σταθερές. Οι μεταβλητές αυτές στη ΓΛΩΣΣΑ είναι γνωστές στο αντίστοιχο υποπρόγραμμα που δηλώνονται και μόνο σε αυτό. Είναι δηλ. τοπικές στο συγκεκριμένο τμήμα προγράμματος. Ο μόνος τρόπος για να περάσει μία τιμή από ένα υποπρόγραμμα σε ένα άλλο ή από το κυρίως πρόγραμμα είναι δια μέσου των παραμέτρων.

Αφού όλες οι μεταβλητές είναι τοπικές, το ίδιο όνομα μεταβλητής μπορεί να εμφανίζεται σε διαφορετικά τμήματα προγράμματος, χωρίς να αντιστοιχεί στην ίδια μεταβλητή.

Ότι ισχύει για την εμβέλεια των μεταβλητών ισχύει και για τις συμβολικές σταθερές.

Εμβέλεια (scope) των μεταβλητών: το τμήμα του προγράμματος που ισχύουν οι μεταβλητές.

10.6 Εμβέλεια μεταβλητών-σταθερών

ΕΙΔΗ ΕΜΒΕΛΕΙΑΣ

α) Απεριόριστη εμβέλεια
όλες οι μεταβλητές και όλες οι σταθερές είναι γνωστές σε οποιοδήποτε τμήμα του προγράμματος και λέγονται καθολικές (global)
Μειονεκτήματα:
1. καταστρατηγεί την αρχή της αυτονομίας των υποπρογραμμάτων
2. ο καθένας που γράφει κάποιο υποπρόγραμμα πρέπει να γνωρίζει όλες τις καθολικές μεταβλητές

β) Περιορισμένη εμβέλεια
Όλες οι μεταβλητές που χρησιμοποιούνται σε ένα τμήμα προγράμματος, πρέπει να δηλώνονται σε αυτό το τμήμα και λέγονται τοπικές (local), ισχύουν δηλαδή για το υποπρόγραμμα στο οποίο δηλώθηκαν. Στη **ΓΛΩΣΣΑ** έχουμε περιορισμένη εμβέλεια. Πλεονεκτήματα: η απόλυτη αυτονομία όλων των υποπρογραμμάτων και η δυνατότητα να χρησιμοποιείται οποιοδήποτε όνομα.

γ) Μερικώς περιορισμένη εμβέλεια.
Άλλες μεταβλητές είναι τοπικές και άλλες καθολικές. Προσφέρει μερικά πλεονεκτήματα στον πεπειραμένο προγραμματιστή, αλλά για τον αρχάριο περιπλέκει το πρόγραμμα.

Πίνακας παρακολούθησης τιμών μεταβλητών

```
ΠΡΟΓΡΑΜΜΑ ΠΤ1
  ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ: Α, Β, Γ
ΑΡΧΗ
    Α ← 3
    Β ← 13
    Γ ← 2
    ΓΡΑΨΕ Α, Β, Γ
    ΚΑΛΕΣΕ Διαδ (Β, Γ)
    ΓΡΑΨΕ Α, Β, Γ
    ΚΑΛΕΣΕ Διαδ (Γ, Α)
    ΓΡΑΨΕ Α, Β, Γ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
! ====================
ΔΙΑΔΙΚΑΣΙΑ Διαδ (α1, α2)
  ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: α1, α2
ΑΡΧΗ
          α1 ← α1 DIV 2
          α2 ← α2 ^ 3
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Α	Β	Γ	Οθόνη
3	13	2	3, 13, 2
3	6	8	3, 6, 8
27	6	4	27, 6, 4

(Β)	(Γ)
α1	α2
13	2
6	8

(Γ)	(Α)
α1	α2
8	3
4	27

Πίνακας παρακολούθησης τιμών μεταβλητών

```
ΠΡΟΓΡΑΜΜΑ ΠΤ3
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Α, Β, Γ
ΑΡΧΗ
  Α ← 27
  Β ← 2
  Γ ← 13
  ΚΑΛΕΣΕ Διαδ (Α, Β, Γ)
  ΓΡΑΨΕ Α, Β, Γ
  ΚΑΛΕΣΕ Διαδ (Γ, Α, Β)
  ΓΡΑΨΕ Α, Β, Γ
  ΚΑΛΕΣΕ Διαδ (Β, Γ, Α)
  ΓΡΑΨΕ Α, Β, Γ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
! ======================
ΔΙΑΔΙΚΑΣΙΑ Διαδ (α1, α2, α3)
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: α1, α2, α3
ΑΡΧΗ
  α3 ← α1 + α2 - α3
  α2 ← α2 - α1
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Α	Β	Γ	Οθόνη
27	2	13	
27	-25	16	27, -25, 16
11	68	16	11, 68, 16
73	68	-52	73, 68, -52

(Α)	(Β)	(Γ)
α1	α2	α3
27	2	13
27	-25	16

(Γ)	(Α)	(Β)
α1	α2	α3
16	27	-25
16	11	68

(Β)	(Γ)	(Α)
α1	α2	α3
68	16	11
68	-52	73

Πίνακας παρακολούθησης τιμών μεταβλητών

```
ΠΡΟΓΡΑΜΜΑ ΠΤ5
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: α, β, αποτ
ΑΡΧΗ
  α ← 2
  β ← 13
  αποτ ← Υπ(α, β)
  Γράψε αποτ
  α ← 5 * α + αποτ
  β ← Α_Τ(αποτ – 20)
  αποτ ← Υπ(β, α) – 3
  Γράψε α, β, αποτ
  αποτ ← Υπ(α, β)
  Γράψε αποτ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
! ====================
Συνάρτηση Υπ(x, y): Ακέραια
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: x, y, π
ΑΡΧΗ
  x ← x + 1
  y ← y - 1
  π ← (x + y) div x
  Υπ ← 2 – π
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

α	β	αποτ	Οθόνη
2	13	-	
2	13	-3	-3
7			
	23		
7	23	-2	7, 23, -2
		-1	-1

(α)	(β)		
x	y	π	Υπ
2	13	-	
3	12	5	-3

(β)	(α)		
x	y	π	Υπ
23	7	-	
24	6	1	1

(α)	(β)		
x	y	π	Υπ
7	23	-	
8	22	3	-1

Προγράμματα με υποπρογράμματα

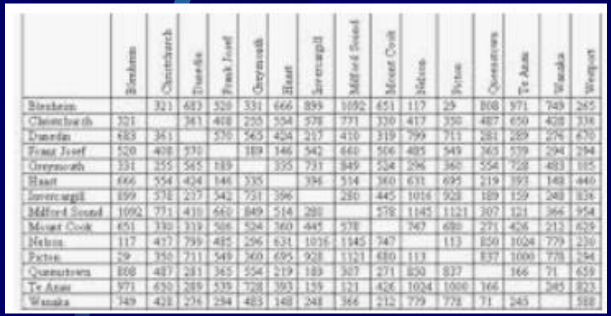
Σύμφωνα με το νέο κώδικα οδικής κυκλοφορίας τα πρόστιμα που δίνονται για 3 είδη παραβάσεων, είναι: «κινητό» ➔ 180 €, «κράνος» ➔ 250 €, «ζώνη» ➔ 300 €. Να γραφεί πρόγραμμα που δέχεται επαναληπτικά το είδος της παράβασης (‘ΚΙΝ’ / ‘ΚΡ’ / ‘Ζ’) και σταματάει μόλις συμπληρωθούν 100 παραβάσεις ή συνολικό ποσό προστίμων που είναι τουλάχιστον 20000 €. Σε κάθε επανάληψη, να υπολογίζει με υποπρόγραμμα (το οποίο και να γράψετε) το ποσό του προστίμου και να το εμφανίζει. Να εμφανίζει το συνολικό αριθμό των παραβάσεων και το συνολικό ποσό των προστίμων. Να υλοποιήσετε υποπρόγραμμα που δέχεται το είδος της παράβασης και επιστρέφει το αντίστοιχο πρόστιμο.

```
Πρόγραμμα ΚΟΚ
Μεταβλητές
  Ακέραιες: ΑΠ, ΣΠ, πρ
  Χαρακτήρες: είδος
Αρχή
  ΑΠ ← 0
  ΣΠ ← 0
  ΑρχήΕπανάληψης
    Διάβασε είδος
    πρ ← Πρόστιμο(είδος)
    Γράψε πρ
    ΑΠ ← ΑΠ + 1
    ΣΠ ← ΣΠ + πρ
  ΜέχριςΌτου ΑΠ = 100 Η ΣΠ >= 20000
  Γράψε ΑΠ, ΣΠ
Τέλος_Προγράμματος
```

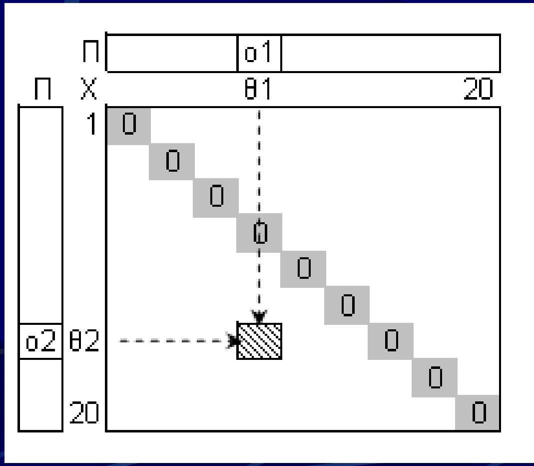
```
Συνάρτηση Πρόστιμο(είδος): Ακέραια
Μεταβλητές
  Ακέραιες: π
  Χαρακτήρες: είδος
Αρχή
  Αν είδος = ‘ΚΙΝ’ τότε
    π ← 180
  Αλλιώς Αν είδος = ‘ΚΡ’ τότε
    π ← 250
  Αλλιώς
    π ← 300
  ΤέλοςΑν
  Πρόστιμο ← π

ΤέλοςΣυνάρτησης
```

Προγράμματα με υποπρογράμματα

Να γραφεί η Διαδικασία Βρες(Π, θ , ον) η οποία διαβάζει μία τιμή τύπου χαρακτήρα με έλεγχο εγκυρότητας ώστε να ανήκει στον Π[20] χαρακτήρων. Να επιστρέφει την τιμή (ον) και τη θέση θ(1-20) που εντοπίστηκε. Να γραφεί κύριο πρόγραμμα που διαβάζει τους Π[20] και Χ[20, 20] με τα ονόματα 20 πόλεων και τις χιλιομετρικές τους αποστάσεις και με τη βοήθεια της παραπάνω διαδικασίας διαβάζει δύο έγκυρα ονόματα πόλεων. Να εμφανίζει τη μεταξύ τους απόσταση.



	Blenheim	Christchurch	Dunedin	Franz Josef	Greymouth	Haast	Invercargill	Milford Sound	Mount Cook	Nelson	Picton	Queenstown	Te Anau	Wanaka	Westport
Blenheim		321	683	520	331	666	899	1092	651	117	29	808	971	749	265
Christchurch	321		361	408	255	554	578	771	330	417	350	487	650	428	338
Dunedin	683	361		570	565	424	217	410	319	799	711	281	289	276	670
Franz Josef	520	408	570		189	146	542	660	506	485	549	365	539	294	294
Greymouth	331	255	565	189		335	731	849	524	296	360	554	728	483	105
Haast	666	554	424	146	335		396	514	360	631	695	219	393	148	440
Invercargill	899	578	217	542	731	396		280	445	1016	928	189	159	248	836
Milford Sound	1092	771	410	660	849	514	280		578	1145	1121	307	121	366	954
Mount Cook	651	330	319	506	524	360	445	578		747	680	271	426	212	629
Nelson	117	417	799	485	296	631	1016	1145	747		113	850	1024	779	230
Picton	29	350	711	549	360	695	928	1121	680	113		837	1000	778	294
Queenstown	808	487	281	365	554	219	189	307	271	850	837		166	71	659
Te Anau	971	650	289	539	728	393	159	121	426	1024	1000	166		285	823
Wanaka	749	428	276	294	483	148	248	366	212	779	778	71	245		588




```
Πρόγραμμα Απόσταση
...
Κάλεσε Βρες(Π, θ1, ον1)
Κάλεσε Βρες(Π, θ2, ον2)
Γράψε 'Η απόσταση των ', ον1, ' και ', ον2, ' είναι: ', Χ[θ1, θ2]
```

```
Διαδικασία Βρες(Π, θ, ον)
Μεταβλητές
  Χαρακτήρες: Π[20], ον
  Ακέραιες: θ, i
  Λογική: βρ
Αρχή
  ΑρχήΕπανάληψης
    Διάβασε ον
    i ← 1
    βρ ← Ψευδής
    Όσο i <= 20 ΚΑΙ βρ = Ψευδής επανάλαβε
      Αν Π[i] = ον) τότε
        βρ ← Αληθής
        θ ← i
      Αλλιώς
        i ← i + 1
      ΤέλοςΑν
    ΤέλοςΕπανάληψης
  ΜέχριςΌτου βρ = Αληθής
ΤέλοςΔιαδικασίας
```

Μετατροπή: Συνάρτηση – Διαδικασία – Χωρίς υποπρόγραμμα

Γενικά:

Πρόγραμμα Π1 … α ← Σ(x_1, x_2, …, x_n) … ΤέλοςΠρογράμματος	Πρόγραμμα Π2 … Κάλεσε Δ(x_1, x_2, …, x_n, y) α ← y … ΤέλοςΠρογράμματος
Συνάρτηση Σ(x_1, x_2, …, x_n): Τύπος … Σ ← αποτέλεσμα ΤέλοςΣυνάρτησης	Διαδικασία Δ(x_1, x_2, …, x_n, x_{n+1}) … x_{n+1} ← αποτέλεσμα ΤέλοςΔιαδικασίας

π.χ.

Πρόγραμμα Με_Συνάρτηση
…
Διάβασε Α, Β
Γ ← 0
Όσο (Β <> 0) επανάλαβε
    Γ ← Γ + Συν(Α, Β)
    Α ← 2 * Α
    Β ← Β div 2
ΤέλοςΕπανάληψης
Γράψε Γ
	Πρόγραμμα Με_Διαδικασία
	…
	Διάβασε Α, Β
	Γ ← 0
	Όσο (Β <> 0) επανάλαβε
	    Κάλεσε Διαδ(Α, Β, Ω)
	    Γ ← Γ + Ω
	    Α ← 2 * Α
	    Β ← Β div 2
	ΤέλοςΕπανάληψης
	Γράψε Γ
		Πρόγραμμα Χωρίς_Υποπρόγραμμα
		…
		Διάβασε Α, Β
		Γ ← 0
		Όσο (Β <> 0) επανάλαβε
		    Αν (Β mod 2 <> 0) τότε
		        ζ ← Α
		    Αλλιώς
		        ζ ← 0
		    ΤέλοςΑν
		    Ω ← ζ
		    Γ ← Γ + Ω
		    Α ← 2 * Α
		    Β ← Β div 2
		ΤέλοςΕπανάληψης
		Γράψε Γ

Συνάρτηση Συν(χ, ψ): Ακέραια
...
Αν (ψ mod 2 <> 0) τότε
    ζ ← χ
Αλλιώς
    ζ ← 0
ΤέλοςΑν
Συν ← ζ
	Διαδικασία Διαδ(χ, ψ, ω)
	...
	Αν (ψ mod 2 <> 0) τότε
	    ζ ← χ
	Αλλιώς
	    ζ ← 0
	ΤέλοςΑν
	ω ← ζ	

Μετατροπή: Συνάρτηση – Διαδικασία – Χωρίς υποπρόγραμμα

π.χ. περίπτωση συνάρτησης που αλλάζει τις τιμές των τυπικών παραμέτρων ➔ αντίγραφα τυπικών παραμέτρων

```
ΣΥΝΑΡΤΗΣΗ Τιμή (A, B): ΛΟΓΙΚΗ
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: A, B
ΑΡΧΗ
  A←A+8
  B←B ^3
  ΑΝ (A + B) mod 2 = 0 ΤΟΤΕ
    Τιμή ←ΑΛΗΘΗΣ
  ΑΛΛΙΩΣ
    Τιμή ←ΨΕΥΔΗΣ
  ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

➔

```
ΔΙΑΔΙΚΑΣΙΑ Τιμή (Α, Β,Γ)
 ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Α, Β, Α2, Β2
  ΛΟΓΙΚΕΣ: Γ
ΑΡΧΗ
 Α2 ← Α
 Β2 ← Β
 Α2 ← Α2 + 8
 Β2 ← Β2 ^ 3
 ΑΝ (Α2 + Β2) mod 2 = 0 ΤΟΤΕ
  Γ ← ΑΛΗΘΗΣ
 ΑΛΛΙΩΣ
  Γ ← ΨΕΥΔΗΣ
 ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

```
a ← f1(x) + f2(y)
⇔ με διαδικασίες:
ΚΑΛΕΣΕ D1(x, a1)
ΚΑΛΕΣΕ D2(y, a2)
a ← a1 + a2
```

Διάφορα

1. Αν η ακέραια συνάρτηση f δέχεται σαν παράμετρο μία ακέραια τιμή τότε η εντολή: Γράψε f(f(2)) είναι συντακτικά σωστή (Σωστό/Λάθος;). Ισχύει το ίδιο για λογική συνάρτηση f;
2. Έστω η συνάρτηση Sum η οποία δέχεται 2 πραγματικούς και επιστρέφει το άθροισμά τους. Συμπληρώστε την παρακάτω εντολή (χωρίς χρήση των τελεστών +, -) ώστε να εκχωρείται στη μεταβλητή x το άθροισμα των πραγματικών μεταβλητών a,b,c: x ← ………………
3. Έστω η συνάρτηση max(α, β):Πραγματική η οποία επιστρέφει τη μεγαλύτερη τιμή των πραγματικών α, β. Συμπληρώστε τις παρακάτω εντολές ώστε να εκχωρείται στη μεταβλητή min η μικρότερη τιμή των α, β.

min ← ... - max(α, β) min ← -max(..., ...)

Συμπληρώστε το κενό ώστε να εκχωρείται στη μεταβλητή max4 η μεγαλύτερη τιμή των α, β, γ, δ: max4 ← ……

Μετατροπή Διαδικασίας σε Συνάρτηση/Συναρτήσεις

π.χ. Να μετατραπούν τα παρακάτω με Συναρτήσεις:

```
Πρόγραμμα Άσκηση
  Μεταβλητές
    Ακέραιες: A[100], i, min, max
Αρχή
  για i από 1 μέχρι 100
    Διάβασε A[i]
  ΤέλοςΕπανάληψης
  Κάλεσε MinMax(A, min, max)
  Γράψε max-min
ΤέλοςΠρογράμματος

Διαδικασία MinMax(A, min, max)
  Μεταβλητές
    Ακέραιες: A[100], i, min, max
Αρχή
  min <-- A[1]
  για i από 2 μέχρι 100
    Αν A[i] < min τότε
      min  <-- A[i]
    ΤέλοςΑν
  ΤέλοςΕπανάληψης
  max <-- A[1]
  για i από 2 μέχρι 100
    Αν A[i] > max τότε
      max  <-- A[i]
    ΤέλοςΑν
  ΤέλοςΕπανάληψης
ΤέλοςΔιαδικασίας
```

➔

```
Πρόγραμμα Άσκηση
  Μεταβλητές
    Ακέραιες: A[100], i, min, max
Αρχή
  για i από 1 μέχρι 100
    Διάβασε A[i]
  ΤέλοςΕπανάληψης
  min <-- Minimum(A)
  max <-- Maximum(A)
  Γράψε max-min
ΤέλοςΠρογράμματος

Συνάρτηση Minimum(A):Ακέραια
  Μεταβλητές
    Ακέραιες: A[100], i, min
Αρχή
  min <-- A[1]
  για i από 2 μέχρι 100
    Αν A[i] < min τότε
      min  <-- A[i]
    ΤέλοςΑν
  ΤέλοςΕπανάληψης
  Minimum <-- min
ΤέλοςΣυνάρτησης

Συνάρτηση Maximum(A):Ακέραια
  Μεταβλητές
    Ακέραιες: A[100], i, min
Αρχή
  max <-- A[1]
  για i από 2 μέχρι 100
    Αν A[i] > max τότε
      max  <-- A[i]
    ΤέλοςΑν
  ΤέλοςΕπανάληψης
  Maximum <-- max
ΤέλοςΣυνάρτησης
```

```
ΠΡΟΓΡΑΜΜΑ  kef10_Alysidwth_klhsh
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: x, y, z, A[3], i
ΑΡΧΗ
  x <-- 1
  y <-- 2
  z <-- 3
  ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 3
    A[i] <-- i+3
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  ΓΡΑΨΕ x, y, z, A[1], A[2], A[3]
  ΚΑΛΕΣΕ D1(y, z, x)
  ΓΡΑΨΕ x, y, z
  ΚΑΛΕΣΕ D1(A[2], A[1], A[3])
  ΓΡΑΨΕ A[1], A[2], A[3]
  ΚΑΛΕΣΕ D2(A[2], z, A[1])
  ΓΡΑΨΕ x, y, z, A[1], A[2], A[3]
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
! --------------------------
ΔΙΑΔΙΚΑΣΙΑ  D1(p, q, r)
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: p, q, r
ΑΡΧΗ
  p <-- p + 2
  q <-- q - 2
  r <-- r * 7
  ΚΑΛΕΣΕ D2(r, p, q)
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
! --------------------------
ΔΙΑΔΙΚΑΣΙΑ D2(a, b, c)
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: a, b, c
ΑΡΧΗ
  a <-- a div 3
  b <-- b mod 9
  c <-- c^2
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Πίνακας παρακολούθησης τιμών μεταβλητών

x	y	z	A[1]	A[2]	A[3]	Οθόνη
1	2	3	4	5	6	1,2,3,4,5,6
2	4	1				2,4,1
			4	7	14	4,7,14
		1	16	2		2,4,1,16,2,14

D1

y	z	x
p	q	r
2	3	1
4	1	7
4	1	2

D2

r	p	q
a	b	c
7	4	1
2	4	1

D1

A[2]	A[1]	A[3]
p	q	r
5	4	6
7	2	42
7	4	14

D2

r	p	q
a	b	c
42	7	2
14	7	4

D2

A[2]	z	A[1]
a	b	c
7	1	4
2	1	16

```
1  2  3  4  5  6
2  4  1
4  7  14
2  4  1  16  2  14
```

```
ΠΡΟΓΡΑΜΜΑ  kef10_pinakas_parametros
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Α[10], i
ΑΡΧΗ
  ΚΑΛΕΣΕ ReadArray(A)
  ΚΑΛΕΣΕ PrintArrayInverse(A)
  ΓΡΑΨΕ  'Άθροισμα = ', SumArray(A)
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
 ! ------------------------------
ΔΙΑΔΙΚΑΣΙΑ  ReadArray(A)
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Α[10], i
ΑΡΧΗ
  ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 10
    ΔΙΑΒΑΣΕ Α[i]
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
 ! ------------------------------
ΔΙΑΔΙΚΑΣΙΑ  PrintArrayInverse(A)
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Α[10], i
ΑΡΧΗ
  ΓΙΑ i ΑΠΟ 10 ΜΕΧΡΙ 1  ΜΕ ΒΗΜΑ  -1
    ΓΡΑΨΕ Α[i]
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
 ! ------------------------------
ΣΥΝΑΡΤΗΣΗ  SumArray(A): ΑΚΕΡΑΙΑ
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: Α[10], i, s
ΑΡΧΗ
  s <-- 0
  ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 10
    s <-- s + A[i]
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  SumArray <-- s
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Πίνακες ως παράμετροι

Πρόγραμμα που με κατάλληλα υποπρογράμματα: α) διαβάζει ακέραιο Α[10] β) τον εμφανίζει ανάποδα γ) υπολογίζει το άθροισμα των στοιχείων του

Συνάρτηση που παίρνει ως παραμέτρους 2 αλφαριθμητικά και τα συγκρίνει.

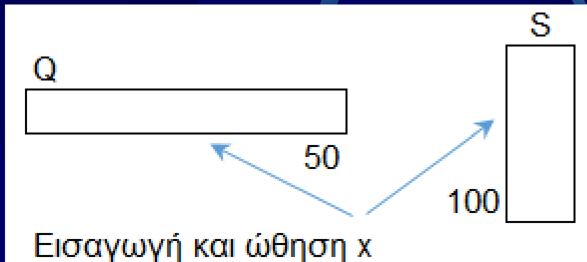
```
Συνάρτηση Compare(x, y): ...
Μεταβλητές
 Χαρακτήρες: x, y
Αρχή
 Αν x < y τότε
   ... ← ...
 ΑλλιώςΑν x > y τότε
   ... ← ...
 Αλλιώς
   ... ← ...
 ΤέλοςΑν
ΤέλοςΣυνάρτησης
```

Να γραφεί υποπρόγραμμα το οποίο...

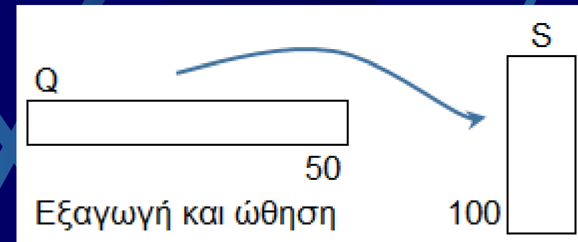
- "διαβάζει/δέχεται σαν είδοδο" (Διάβασε: διαδικασία)
- "γράφει/εμφανίζει" (Γράψε: διαδικασία)
- "δέχεται" (παράμετρος: διαδικασία ή συνάρτηση)
- "επιστρέφει" (παράμετρος: διαδικασία ή συνάρτηση για μία τιμή)

Στοίβα και Ουρά - συνδυασμοί

Έστω ακέραια στοίβα S[100] με δείκτη top και ακέραια ουρά Q[50] με δείκτες f και r. Να γίνουν διαδικασίες που επιτελούν τα παρακάτω ζευγάρια πράξεων (εφόσον γίνονται και οι δύο). Να επιστρέφουν μέσω λογικής μεταβλητής Αληθής/Ψευδής αναλόγως εάν έγιναν και οι δύο πράξεις ή όχι.



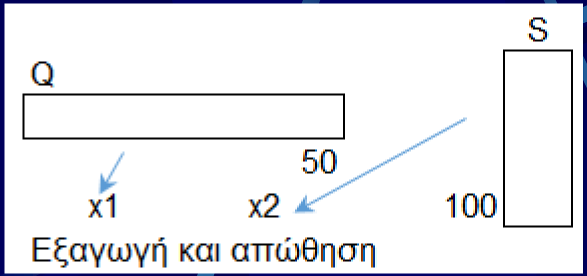

```
Διαδικασία ΕισαγωγήΏθηση(Q, f, r, S, top, x, done)
  Μεταβλητές
    Ακέραιες: Q[50], f, r, S[100], top, x
    Λογικές: done
Αρχή
  Αν r=50 Η top=100 τότε
    done <-- Ψευδής
  Αλλιώς
    Αν f=0 ΚΑΙ r=0 τότε
      f <-- 1
      r <-- 1
    Αλλιώς
      r <-- r + 1
    ΤέλοςΑν
    Q[r] <-- x
    top <-- top + 1
    S[top] <-- x
    done <-- Αληθής
  ΤέλοςΑν
ΤέλοςΔιαδικασίας
```



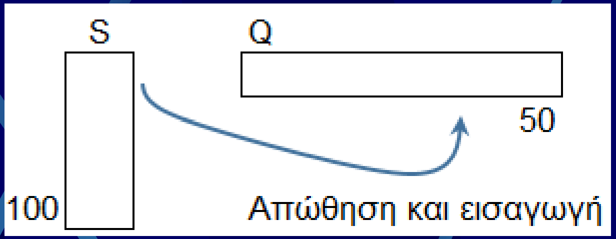

```
Διαδικασία ΕξαγωγήΏθηση(Q, f, r, S, top, done)
  Μεταβλητές
    Ακέραιες: Q[50], f, r, S[100], top
    Λογικές: done
Αρχή
  Αν f=0 ΚΑΙ r=0 Η top=100 τότε
    done <-- Ψευδής
  Αλλιώς
    x <-- Q[f]
    Αν f=r τότε
      f <-- 0
      r <-- 0
    Αλλιώς
      f <-- f + 1
    ΤέλοςΑν
    top <-- top + 1
    S[top] <-- x
    done <-- Αληθής
  ΤέλοςΑν
ΤέλοςΔιαδικασίας
```

Στοίβα και Ουρά - συνδυασμοί

Έστω ακέραια στοίβα S[100] με δείκτη top και ακέραια ουρά Q[50] με δείκτες f και r. Να γίνουν διαδικασίες που επιτελούν τα παρακάτω ζευγάρια πράξεων (εφόσον γίνονται και οι δύο). Να επιστρέφουν μέσω λογικής μεταβλητής Αληθής/Ψευδής αναλόγως εάν έγιναν και οι δύο πράξεις ή όχι.



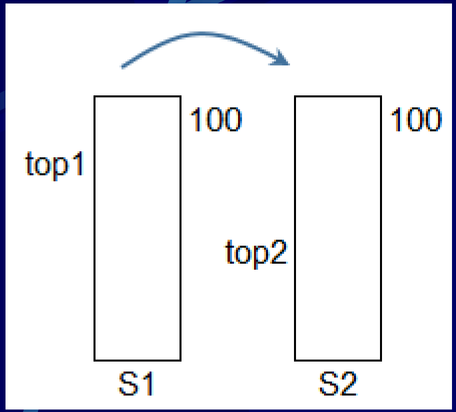

```
Διαδικασία ΕξαγωγήΑπώθηση(Q, f, r, x1, S, top, x2, done)
 Μεταβλητές
  Ακέραιες: Q[50], f, r, S[100], top, x1, x2
  Λογικές: done
Αρχή
 Αν f=0 ΚΑΙ r=0 Η top=0 τότε
   done <-- Ψευδής
 Αλλιώς
   x1 <-- Q[f]
   Αν f=r τότε
     f <-- 0
     r <-- 0
   Αλλιώς
     f <-- f + 1
   ΤέλοςΑν
   x2 <-- S[top]
   top <-- top - 1
   done <-- Αληθής
 ΤέλοςΑν
ΤέλοςΔιαδικασίας
```




```
Διαδικασία ΑπώθησηΕισαγωγή(Q,f,r,S,top,done)
 Μεταβλητές
  Ακέραιες: Q[50], f, r, S[100], top, x
  Λογικές: done
Αρχή
 Αν top=0 Η r=50 τότε
   done <-- Ψευδής
 Αλλιώς
   x <-- S[top]
   top <-- top-1
   Αν f=0 ΚΑΙ r=0 τότε
     f <-- 1
     r <-- 1
   Αλλιώς
     r <-- r + 1
   ΤέλοςΑν
   Q[r] <-- x
   done <-- Αληθής
 ΤέλοςΑν
ΤέλοςΔιαδικασίας
```

Στοίβα και Ουρά - συνδυασμοί

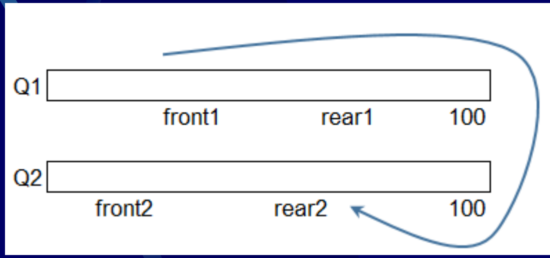
Έστω δύο ακέραιες στοίβες S1[100] και S2[100] με δείκτες top1 και top2 αντίστοιχα. Να γραφεί διαδικασία απωθεί ένα στοιχείο από την S1 και την ωθεί στην S2 (εφόσον γίνονται και τα δύο). Να επιστρέφει μέσω λογικής μεταβλητής Αληθής/Ψευδής αναλόγως εάν έγιναν και οι δύο πράξεις ή όχι.




```
Διαδικασία ΑπώθησηΏθηση(S1,top1,S2,top2,done)
  Μεταβλητές
    Ακέραιες: S1[100], top1, S2[100], top2
    Λογικές: done
Αρχή
  Αν top1=0 Η top2=100 τότε
    done <-- Ψευδής
  Αλλιώς
    done <-- Αληθής
    top2 <-- top2 + 1
    S2[top2] <-- S1[top1]
    top1 <-- top1 - 1
  ΤέλοςΑν
ΤέλοςΔιαδικασίας
```

Έστω δύο ακέραιες ουρές Q1[100] και Q2[100] με δείκτες front1, rear1 και front2, rear2 αντίστοιχα. Να γραφεί διαδικασία εξάγει ένα στοιχείο από την Q1 και την εισάγει στην Q2 (εφόσον γίνονται και τα δύο). Να επιστρέφει μέσω λογικής μεταβλητής Αληθής/Ψευδής αναλόγως εάν έγιναν και οι δύο πράξεις ή όχι.

```
Διαδικασία ΕξαγωγήΕισαγωγή (Q1,front1,rear1,Q2,front2,rear2,done)
  Μεταβλητές
    Ακέραιες: Q1[100],front1,rear1,Q2[100],front2,rear2,x
    Λογικές: done
Αρχή
  Αν (front1=0 ΚΑΙ rear1=0) Η rear2=100 τότε
    done <-- Ψευδής
  Αλλιώς
    done <-- Αληθής
    x <-- Q1[front1]
    Αν front1=rear1 τότε
      front1 = 0
      rear1 = 0
    Αλλιώς
      front1 = front1 + 1
    ΤέλοςΑν
    Αν front2=0 ΚΑΙ rear2=0 τότε
      front2 <-- 1
      rear2 <-- 1
    Αλλιώς
      rear2 <-- rear2 + 1
    ΤέλοςΑν
    Q2[rear2] <-- x
  ΤέλοςΑν
ΤέλοςΔιαδικασίας
```



Προγράμματα με υποπρογράμματα

Ένα παιχνίδι τένις, χωρίζεται σε πόντους (points), games και sets. Κερδίζει αυτός που θα πάρει τρία σετ. Ένα set αποτελείται από games (τουλάχιστον έξι), τα οποία με τη σειρά τους αποτελούνται από πόντους. Λέμε ότι ένας τενίστας πήρε το σετ, όταν έχει κερδίσει έξι ή επτά games, με διαφορά δύο games από τον αντίπαλό του (πχ 6-3, 6-4 ή 7-5). Αν το σετ είναι ισόπαλο 6-6 games, οι δύο μονομάχοι λύνουν τις διαφορές τους στο λεγόμενο tie break που αποτελείται από πόντους και για να το κερδίσει κάποιος θα πρέπει να φτάσει πρώτος στους 7 τουλάχιστον πόντους, με διαφορά δύο πόντων πχ 7-5. Αν στο tie break υπάρχει ισοπαλία 6-6, τότε αυτό κρίνεται στους οχτώ πόντους και πάει λέγοντας. Ένα game αποτελείται από σειρά πόντων. Για να κερδίσει ο παίκτης ένα game, πρέπει να πάρει τουλάχιστον 4 πόντους με ένα περιθώριο δύο ή περισσότερων πόντων απ'τον αντίπαλό του. Αν το game πάει στο 4-4 θα κερδίσει αυτός που θα προηγηθεί με δύο πόντους διαφορά. α) να γραφεί η διαδικασία Game η οποία προσομοιώνει ένα game διαβάζοντας επαναληπτικά τους πόντους των δύο παικτών και επιστρέφοντας τον νικητή 1/2 β) να γραφεί η διαδικασία Set η οποία προσομοιώνει ένα set καλώντας επαναληπτικά τη διαδικασία Game και αν χρειασθεί τους πόντους στο tie break. Να επιστρέφει τον νικητή 1/2 γ) κύριο πρόγραμμα το οποίο διαβάζει τα ονόματα των δύο παικτών και προσομοιώνει ένα παιχνίδι τένις καλώντας επαναληπτικά τη διαδικασία Set. Να εμφανίζει το όνομα του νικητή.

```
Πρόγραμμα Tennis
Μεταβλητές
  Χαρακτήρες: ον1,ον2
  Ακέραιες: s1,s2, winner
Αρχή
  Διάβασε ον1,ον2
  s1 <-- 0 s2 <-- 0
  ΑρχήΕπανάληψης
    Κάλεσε Set(winner)
    Αν winner=1 τότε
      s1 <-- s1 + 1
    Αλλιώς
      s2 <-- s2 + 1
    ΤέλοςΑν
  ΜέχριςΌτου s1=3 Η s2=3
  Αν s1=3 τότε
    Γράψε ον1
  Αλλιώς
    Γράψε ον2
  ΤέλοςΑν
ΤέλοςΠρογράμματος
```

```
Διαδικασία Set(winner)
Μεταβλητές
  Ακέραιες:
winner,w,g1,g2,min,max,p1,p2,p
Αρχή
  g1 <-- 0 g2 <-- 0
  ΑρχήΕπανάληψης
    Κάλεσε Game(w)
    Αν w=1 τότε
      g1 <-- g1 + 1
    Αλλιώς
      g2 <-- g2 + 1
    ΤέλοςΑν
    max <-- g1
    min <-- g2
    Αν g2 > g1 τότε
      max <-- g2
      min <-- g1
    ΤέλοςΑν
  ΜέχριςΌτου max>=6 ΚΑΙ max-
min>=2 Η min=6 ΚΑΙ max=6
  Αν g1 > g2 τότε
    winner <-- 1
  ΑλλιώςΑν g2 > g1 τότε
    winner <-- 2
```

```
  Αλλιώς ! tie break
    p1 <-- 0 p2 <-- 0
    ΑρχήΕπανάληψης
      Διάβασε p
      Αν p=1 τότε
        p1 <-- p1 + 1
      Αλλιώς
        p2 <-- p2 + 1
      ΤέλοςΑν
      max <-- p1
      min <-- p2
      Αν p2 > p1 τότε
        max <-- p2
        min <-- p1
      ΤέλοςΑν
    ΜέχριςΌτου max>=7 ΚΑΙ max-min>=2
    Αν p1 > p2 τότε
      winner <-- 1
    Αλλιώς
      winner <-- 2
    ΤέλοςΑν
  ΤέλοςΑν
ΤέλοςΔιαδικασίας
```

```
Διαδικασία Game(winner)
Μεταβλητές
 Ακέραιες:
winner,p1,p2,p,min,max
Αρχή
 p1 <-- 0 p2 <-- 0
 ΑρχήΕπανάληψης
  Διάβασε p
  Αν p=1 τότε
   p1 <-- p1 + 1
  Αλλιώς
   p2 <-- p2 + 1
  ΤέλοςΑν
  max <-- p1
  min <-- p2
  Αν p2 > p1 τότε
   max <-- p2
   min <-- p1
  ΤέλοςΑν
 ΜέχριςΌτου max>=4 ΚΑΙ
max-min>=2
 Αν p1 > p2 τότε
  winner <-- 1
 Αλλιώς
  winner <-- 2
 ΤέλοςΑν
ΤέλοςΔιαδικασίας
```

Να γραφεί Διαδικασία ισοδύναμη με την παρακάτω Συνάρτηση:

```
Συνάρτηση ΜΟ(Α):Πραγματική
Μεταβλητές
 Ακέραιες: A[100], i, j, tmp, s
Αρχή
 για i από 2 μέχρι 100
  για j από 100 μέχρι i μεβήμα -1
   Αν A[j-1] > A[j] τότε
    tmp <-- A[j-1]
    A[j-1] <-- A[j]
    A[j] <-- tmp
   ΤέλοςΑν
  ΤέλοςΕπανάληψης
 ΤέλοςΕπανάληψης
 s <-- 0
 για i από 91 μέχρι 100
  s <-- s + A[i]
 ΤέλοςΕπανάληψης
 ΜΟ <-- s/10
ΤέλοςΣυνάρτησης
```

```
Διαδικασία Δ(Α, ΜΟ)
Μεταβλητές
 Ακέραιες: A[100], A2[100] i, j, tmp, s
 Πραγματικές: ΜΟ
Αρχή
 για i από 1 μέχρι 100
  A2[i] <-- A[i]
 ΤέλοςΕπανάληψης
 για i από 2 μέχρι 100
  για j από 100 μέχρι i μεβήμα -1
   Αν A2[j-1] > A2[j] τότε
    tmp <-- A2[j-1]
    A2[j-1] <-- A2[j]
    A2[j] <-- tmp
   ΤέλοςΑν
  ΤέλοςΕπανάληψης
 ΤέλοςΕπανάληψης
 s <-- 0
 για i από 91 μέχρι 100
  s <-- s + A2[i]
 ΤέλοςΕπανάληψης
 ΜΟ <-- s/10
ΤέλοςΣυνάρτησης
```

Να γράψετε Συνάρτηση ισοδύναμη με την παρακάτω Διαδικασία:

```
Διαδικασία Δ(A, max)
  Μεταβλητές
    Ακέραιες: A[100], max, i
Αρχή
  max <-- A[1]
  για i από 2 μέχρι 100
    Αν A[i] > max τότε
      max <--A[i]
    ΤέλοςΑν
  ΤέλοςΕπανάληψης
ΤέλοςΔιαδικασίας
```

```
Συνάρτηση Σ(A): Ακέραια
  Μεταβλητές
    Ακέραιες: A[100], max, i
Αρχή
  max <-- A[1]
  για i από 2 μέχρι 100
    Αν A[i] > max τότε
      max <--A[i]
    ΤέλοςΑν
  ΤέλοςΕπανάληψης
  Σ <-- max
ΤέλοςΣυνάρτησης
```

Παραδείγματα υποπρογραμμάτων

Σε πραγματικό πίνακα Α[100] εύρεση min(x=0)/max(x=1)

```
Συνάρτηση MinMax(A, x): Πραγματική
...
  m <-- A[1]
  για i από 2 μέχρι 100
    Αν (x=0 ΚΑΙ A[i]<m) Η (x=1 ΚΑΙ A[i]>m) τότε
      m <-- A[i]
    ΤέλοςΑν
  ΤέλοςΕπανάληψης
  MinMax <-- m
ΤέλοςΣυνάρτησης
```

Σε πραγματικό πίνακα Α[100] εύρεση αθροίσματος άρτιων(x=0)/περιττών(x=1)

```
Συνάρτηση SumArtiwnPerittwn(A, x): Πραγματική
...
  sum <-- 0
  για i από 1 μέχρι 100
    Αν (x=0 ΚΑΙ A[i] mod 2 = 0) Η (x=1 ΚΑΙ A[i] mod 2 <> 0) τότε
      sum <-- sum + A[i]
    ΤέλοςΑν
  ΤέλοςΕπανάληψης
  SumArtiwnPerittwn <-- sum
ΤέλοςΣυνάρτησης
```

Έλεγχος εγκυρότητας για εισαγωγή πραγματικού x σε διάστημα [a, b]

```
Διαδικασία ΕισαγωγήΜεΕγκυρότητα(x, a, b)
...
  ΑρχήΕπανάληψης
    Διάβασε x
  ΜέχριςΌτου x >= a ΚΑΙ x <= b
ΤέλοςΔιαδικασίας
```

Σε πραγματικό πίνακα Α[100] σειριακή αναζήτηση του x ξεκινώντας από την αρχή(s=0)/τέλος(s=1)

```
Συνάρτηση Αναζήτηση(A, x, s): Ακέραια
...
  Αν s = 0 τότε
    i <-- 1
  Αλλιώς
    i <-- 100
  ΤέλοςΑν
  βρ <-- Ψευδής
  θέση <-- -1
  Όσο (s=0 ΚΑΙ i<=100 Η s=1 ΚΑΙ i>=1) ΚΑΙ βρ = Ψευδής επανάλαβε
    Αν A[i] = x τότε
      βρ <-- Αληθής
      θέση <-- i
    Αλλιώς
      Αν s = 0 τότε
        i <-- i + 1
      Αλλιώς
        i <-- i - 1
      ΤέλοςΑν
    ΤέλοςΑν
  ΤέλοςΕπανάληψης
  Αναζήτηση <-- θέση
ΤέλοςΣυνάρτησης
```

Παραδείγματα υποπρογραμμάτων

Σε πραγματικό πίνακα A[100] ταξινόμηση κατά αύξουσα(x=0)/φθίνουσα(x=1) σειρά

```
Διαδιακασία Ταξινόμηση(A, x)
...
  για i από 2 μέχρι 100
    για j από 100 μέχρι i μεβήμα -1
      Αν (x=0 ΚΑΙ A[j-i]>A[j]) Η (x=1 ΚΑΙ A[j-i]<A[j]) τότε
        tmp <-- A[j-1]
        A[j-1] <-- A[j]
        A[j] <-- tmp
      ΤέλοςΑν
    ΤέλοςΕπανάληψης
  ΤέλοςΕπανάληψης
ΤέλοςΔιαδικασίας
```

Σε πραγματικό πίνακα A[50, 100] επιστροφή μέσου όρου γραμμής k(x=0)/στήλης k(x=1)

```
Συνάρτηση ΜΟ(A, k, x): Πραγματική
...
  s <-- 0
  Αν x = 0 τότε
    για j από 1 μέχρι 100
      s <-- s + A[k, j]
    ΤέλοςΕπανάληψης
    ΜΟ <-- s / 100
  Αλλιώς
    για i από 1 μέχρι 50
      s <-- s + A[i, k]
    ΤέλοςΕπανάληψης
    ΜΟ <-- s / 50
  ΤέλοςΑν
ΤέλοςΣυνάρτησης
```

Σε πραγματικό πίνακα A[50, 100] εμφάνιση κατά γραμμές(x=0)/στήλες(x=1)

```
Διαδιακασία Εμφάνιση(A, x)
...
  Αν x = 0 τότε
    για i από 1 μέχρι 50
      για j από 1 μέχρι 100
        Γράψε A[i, j]
      ΤέλοςΕπανάληψης
    ΤέλοςΕπανάληψης
  Αλλιώς
    για j από 1 μέχρι 100
      για i από 1 μέχρι 50
        Γράψε A[i, j]
      ΤέλοςΕπανάληψης
    ΤέλοςΕπανάληψης
  ΤέλοςΑν
ΤέλοςΔιαδικασίας
```

Σε πραγματικό πίνακα A[50, 100] επιστροφή μέσου όρου της υποπεριοχής γραμμών γ1-γ2 και στηλών σ1-σ2

```
Συνάρτηση ΜΟ(A, γ1, γ2, σ1, σ2): Πραγματική
...
  s <-- 0
  για i από γ1 μέχρι γ2
    για j από σ1 μέχρι σ2
      s <-- s + A[i, j]
    ΤέλοςΕπανάληψης
  ΤέλοςΕπανάληψης
  ΜΟ <-- s / ((γ2-γ1+1)*(σ2-σ1+1))
ΤέλοςΣυνάρτησης
```

Παραδείγματα υποπρογραμμάτων

Σε πραγματικό πίνακα Α[100, 100] επιστροφή του αθροίσματος των στοιχείων άνω(x=1)/κάτω(x=2) της κύριας διαγωνίου

```
Συνάρτηση SumD(A, x): Πραγματική
...
  s <-- 0
  για i από 1 μέχρι 100
    για j από 1 μέχρι 100
      Αν (x=1 ΚΑΙ i<j) Η (x=2 ΚΑΙ i>j) τότε
        s <-- s + A[i, j]
      ΤέλοςΑν
    ΤέλοςΕπανάληψης
  ΤέλοςΕπανάληψης
  SumD <-- s
ΤέλοςΣυνάρτησης
```

Σε πραγματικό πίνακα Α[100, 100] επιστροφή του αθροίσματος της 1ης(d=1)/2ης(d=2) κύριας διαγωνίου

```
Συνάρτηση SumD(A, d): Πραγματική
...
  s <-- 0
  για i από 1 μέχρι 100
    Αν d=1 τότε
      s <-- s + A[i, i]
    Αλλιώς
      s <-- s + A[i, 101-i]
    ΤέλοςΑν
  ΤέλοςΕπανάληψης
  SumD <-- s
ΤέλοςΣυνάρτησης
```

ή

```
Συνάρτηση SumD(A, d): Πραγματική
...
  Αν d=1 τότε
    p <-- 1
    y <-- 0
  Αλλιώς
    p <-- -1
    y <-- 101
  ΤέλοςΑν
  s <-- 0
  για i από 1 μέχρι 100
    s <-- s + A[i, y + p*i]
  ΤέλοςΕπανάληψης
  SumD <-- s
ΤέλοςΣυνάρτησης
```