ΠΛΗ10

ΕΝΟΤΗΤΑ 2: Αλγόριθμοι και Ψευδογλώσσα

Μάθημα 2.6:
Συναρτήσεις και Διαδικασίες

Δημήτρης Ψούνης

www.psounis.gr

Περιεχόμενα Μαθήματος

Α. Συναρτήσεις

1. Πότε Γράφουμε Συναρτήσεις

- Μία συνάρτηση της Ψευδογλώσσας είναι το αντίστοιχο της μαθηματικής συνάρτησης.
  - Θεωρήστε για παράδειγμα την μαθηματική συνάρτηση $f(x) = 5x + 1$
    - Το f είναι το **όνομα** της συνάρτησης
    - Το x ειναι το **όρισμα** της συνάρτησης
    - Το $5x + 1$ είναι το **σώμα** της συνάρτησης
  - Τώρα πως χρησιμοποιούμε μια συνάρτηση.
    - Π.χ. με όρισμα το 2, δηλαδή $f(2)$ (Στην Ψευδογλώσσα θα λέμε «**καλώντας την f με όρισμα 2**»)
      - Η συνάρτηση υπολογίζεται: $5 \cdot 2 + 1$
      - $5 \cdot 2 + 1 = 11$ (Στην Ψευδογλώσσα θα λέμε «**επιστρέφει 11**»)
    - Π.χ. με όρισμα το 15, δηλαδή το $f(15)$ (Στην Ψευδογλώσσα λέμε «καλώντας την f με όρισμα 15»)
      - Η συνάρτηση υπολογίζεται στο $5 \cdot 15 + 1$
      - $5 \cdot 15 + 1 = 76$ (Στην Ψευδογλώσσα θα λέμε «επιστρέφει 76»)

Α. Συναρτήσεις

1. Πότε Γράφουμε Συναρτήσεις

- Οι γενικοί κανόνες που μας καθοδηγούν στο να δημιουργήσουμε μια συνάρτηση στο πρόγραμμά μας είναι:
  - **Γράφουμε συναρτήσεις όταν πολλές φορές στο πρόγραμμα μας κάνουμε τις ίδιες ενέργειες με τον ίδιο κώδικα.**
    - Π.χ. Αν το πρόγραμμα μας κάνει μία αναζήτηση σε πίνακα πολλές φορές, τότε θα ορίσουμε μια συνάρτηση με όνομα π.χ. `search()` και καλούμε την συνάρτηση αυτή κάθε φορά που θέλουμε να κάνουμε την αναζήτηση στον πίνακα.
  - Και όταν θέλουμε να απλοποιήσουμε την μορφή του προγράμματος μας. **Είναι κακό να έχουμε έναν κώδικα-«μακαρόνι»**, δηλαδή μια τεράστια main που να κάνει πάρα πολλά πράγματα! Προτιμούμε να διασπάμε τον κώδικα σε μέρη και να καλούμε τις αντίστοιχες συναρτήσεις που θα υλοποιούν κάθε αυτόνομη ενέργεια.

Α. Συναρτήσεις
2. Πως Γράφουμε Συναρτήσεις
1. Γενικό Σχήμα

- Το γενικό σχήμα για την σύνταξη μιας συνάρτησης στην ψευδογλώσσα είναι το ακόλουθο:

```
ΑΛΓΟΡΙΘΜΟΣ orismos_sinartisis
ΔΕΔΟΜΕΝΑ
   a,b: INTEGER;

ΣΥΝΑΡΤΗΣΗ cube(x): INTEGER          <- Εδώ ξεκινά η δήλωση της συνάρτησης
ΔΙΕΠΑΦΗ                             <-| Στη διεπαφή ορίζουμε τον τύπο δεδομένων
   ΕΙΣΟΔΟΣ                            |     των ορισμάτων της συνάρτησης
      x: INTEGER;                     |
   ΕΞΟΔΟΣ                             |     και της επιστρεφόμενης τιμής
      cube: INTEGER;
ΑΡΧΗ
   cube:=x*x*x;                      <- Εδώ είναι το σώμα της συνάρτησης
ΤΕΛΟΣ-ΣΥΝΑΡΤΗΣΗΣ

ΑΡΧΗ
   ΔΙΑΒΑΣΕ(a);

   b:=cube(a);                       <- Εδώ είναι η κλήση της συνάρτησης
   ΤΥΠΩΣΕ(EOLN,b);
ΤΕΛΟΣ
```


Α. Συναρτήσεις
2. Πως Γράφουμε Συναρτήσεις
2. Δήλωση Συνάρτησης

- ΠΑΝΤΑ πριν από την ΑΡΧΗ του προγράμματος γράφουμε την δήλωση της συνάρτησης (ή των συναρτήσεων αν θέλουμε να χρησιμοποιήσουμε περισσότερες). Ο τρόπος σύνταξης της δήλωσης της συνάρτησης είναι:

```
ΣΥΝΑΡΤΗΣΗ ονομα_συνάρτησης(ορίσματα): τύπος_επιστρεφόμενης_τιμής
```

- Όπως στην συνάρτηση μάς:

```
ΣΥΝΑΡΤΗΣΗ      cube(x)              : INTEGER
```

  - όπου περιγράφουμε ότι πρόκειται να ορίσουμε μια συνάρτηση με όνομα cube: που παίρνει μία μεταβλητή x σαν όρισμα και επιστρέφει μια ακέραια μεταβλητή.
    - Το όνομα συνάρτησης το επιλέγουμε εμείς ώστε να αντικατοπτρίζει τον υπολογισμό που γίνεται.
    - Αν θέλουμε περισσότερα ορίσματα, τότε τα χωρίζουμε με κόμματα.
    - Στα ορίσματα βάζουμε το όνομα των μεταβλητών, ενώ στην επιστρεφόμενη τιμή βάζουμε τον τύπο δεδομένων της.

Α. Συναρτήσεις
2. Πως Γράφουμε Συναρτήσεις
3. Η Διεπαφή της Συνάρτησης

- Στην Διεπαφή ορίζουμε πιο αναλυτικά τον τύπο δεδομένων της εισόδου και της εξόδου χρησιμοποιώντας τις δεσμευμένες λέξεις:
  - ΕΙΣΟΔΟΣ: όπου περιγράφουμε τα ορίσματα και τον τύπο δεδομένων τους
  - ΕΞΟΔΟΣ: όπου περιγράφουμε την επιστρεφόμενη τιμή (υποχρεωτικά με το όνομα της συνάρτησης) και τον τύπο δεδομένων

```
ΔΙΕΠΑΦΗ
   ΕΙΣΟΔΟΣ
      ονομα_ορίσματος: τύπος_δεδομένων;
   ΕΞΟΔΟΣ
      ονομα_ορίσματος: τύπος_δεδομένων;
```

- Όπως το πραγματοποιήσαμε στην συνάρτηση μας:

```
ΔΙΕΠΑΦΗ
   ΕΙΣΟΔΟΣ
      x: INTEGER;
   ΕΞΟΔΟΣ
      cube: INTEGER;
```

- Παρατήρηση 1: Αν δεν έχουμε είσοδο, τότε το τμήμα ΕΙΣΟΔΟΣ μπορεί να παραληφθεί.
- Παρατήρηση 2: Το τμήμα ΕΞΟΔΟΣ μπορεί να παραληφθεί.

Α. Συναρτήσεις
2. Πως Γράφουμε Συναρτήσεις
4. Το Σώμα της Συνάρτησης

- Το σώμα της συνάρτησης αποτελεί την περιγραφή των εντολών που εκτελεί η συνάρτηση. Πάντα θα είναι ΜΕΤΑ την Διεπαφή και οι εντολές της θα βρίσκονται ανάμεσα στις λέξεις ΑΡΧΗ και ΤΕΛΟΣ-ΣΥΝΑΡΤΗΣΗΣ

```
ΔΕΔΟΜΕΝΑ
   /* Εδώ δηλώνουμε τοπικές μεταβλητές της συνάρτησης */
ΑΡΧΗ
   /* Εδώ βρίσκονται οι εντολές της συνάρτησης */
ΤΕΛΟΣ-ΣΥΝΑΡΤΗΣΗΣ
```

- Παρατηρήστε ότι υπάρχει ξεχωριστό τμήμα δεδομένων (μπορούμε να ορίσουμε τοπικές μεταβλητές αν τις χρειαζόμαστε)
- Οι εντολές θα τρέξουν σειριακά (όπως στον αλγόριθμο).
- Η επιστρεφόμενη τιμή της συνάρτησης αποθηκεύεται στην μεταβλητή που έχει υποχρεωτικά ίδιο όνομα με την συνάρτηση.

```
ΑΡΧΗ
   cube:=x*x*x;
ΤΕΛΟΣ-ΣΥΝΑΡΤΗΣΗΣ
```

A. Συναρτήσεις
2. Πως Γράφουμε Συναρτήσεις
5. Κλήση Συνάρτησης

- Αφού γράψουμε την συνάρτησή μας, έχουμε δικαίωμα να την καλέσουμε οπουδήποτε μέσα στο πρόγραμμα μας. Για να την καλέσουμε:
  - Γράφουμε το όνομα της και διοχετεύουμε κατάλληλα ορίσματα που θα είναι:
    - Είτε απευθείας συγκεκριμένες αριθμητικές τιμές.
    - Είτε ονόματα μεταβλητών που χρησιμοποιούμε ήδη στο πρόγραμμα μας. Προσοχή! Απλά γράφουμε τα ονόματα των μεταβλητών ως ορίσματα και όχι τον τύπο δεδομένων
    - Είτε γενικότερα υπολογιζόμενες παραστάσεις (όπως τις ορίσαμε στο μάθημα 2 που μελετήσαμε τον τελεστή εκχώρησης)
  - Η επιστρεφόμενη τιμή θα αντικαταστήσει το όνομα της συνάρτησης
    - Έτσι κρατάμε το αποτέλεσμα αποθηκεύοντας το σε μία μεταβλητή.

```
b:=cube(a);                <- Εδώ είναι η κλήση της συνάρτησης
```


A. Συναρτήσεις
2. Πως Γράφουμε Συναρτήσεις
6. Άσκηση με ορισμό συναρτήσεων

- **Άσκηση:** Τροποποιήστε το πρόγραμμα ορίζοντας δύο ακόμη συναρτήσεις:
  - Την συνάρτηση square που δέχεται σαν όρισμα έναν ακέραιο και επιστρέφει το τετράγωνό του.
  - Την συνάρτηση f που δέχεται σαν όρισμα έναν ακέραιο x και επιστρέφει το αποτέλεσμα: 2x+1
- Έπειτα το κυρίως πρόγραμμα:
  - Να διαβάζει από την είσοδο το a και να υπολογίζει το τετράγωνό της.
  - Να διαβάζει από την είσοδο το b και να υπολογίζει τον κύβο της
  - Να διαβάζει από την είσοδο το c και να υπολογίζει το f(c).
  - Τελικά να τυπώνει το άθροισμα των παραπάνω αποτελεσμάτων.

A. Συναρτήσεις
2. Πως Γράφουμε Συναρτήσεις
7. Τοπικές και Καθολικές Μεταβλητές

- **Τοπικές Μεταβλητές:** Είναι μεταβλητές που δηλώνονται στην αρχή μιας συνάρτησης και τις οποίες τις «βλέπει» (έχει πρόσβαση) η συνάρτηση και ΜΟΝΟΝ αυτή (όχι δηλαδή οι άλλες συναρτήσεις ή ο αλγόριθμος
  - Προσοχή! Κάθε συνάρτηση έχει τις δικές της μεταβλητές, έτσι π.χ. μπορούν δύο συναρτήσεις να έχουν μεταβλητές με το ίδιο όνομα. Κάθε συνάρτηση θα «βλέπει» μόνο τις δικές της μεταβλητές.
- **Καθολικές Μεταβλητές:** Είναι μεταβλητές που δηλώνονται στην αρχή του αλγόριθμου και τις βλέπουν όλες οι συναρτήσεις και ο αλγόριθμος.

- Μεταγλωττίστε και εκτελέστε το πρόγραμμα της επόμενης διαφάνειας που συνοψίζει τις παρατηρήσεις για τις καθολικές και τις τοπικές μεταβλητές..

Συμβουλή:
- Θεωρείται **κακή προγραμματιστική τακτική να χρησιμοποιούμε καθολικές μεταβλητές**. Θα πρέπει να γνωρίζουμε πως δουλεύουν, αλλά να μην τις χρησιμοποιούμε στα προγράμματά μας!
- Στην ψευδογλώσσα εξ' ορισμού οι μεταβλητές του αλγορίθμου είναι προσβάσιμες και στις συναρτήσεις. Αυτό δεν συνηθίζεται στις συμβατικές γλώσσες προγραμματισμού. Για τον λόγο αυτό θα πρέπει να σκεφτόμαστε ότι τις μεταβλητές του αλγορίθμου ΔΕΝ τις βλέπουν οι συναρτήσεις.

A. Συναρτήσεις
2. Πως Γράφουμε Συναρτήσεις
7. Τοπικές και Καθολικές Μεταβλητές

```
ΑΛΓΟΡΙΘΜΟΣ local_global
ΔΕΔΟΜΕΝΑ
   x,ret:INTEGER;  /* Katholiki metavliti:
                      Tin vlepoun oloi */

ΣΥΝΑΡΤΗΣΗ f1(): INTEGER
ΔΙΕΠΑΦΗ
   ΕΙΣΟΔΟΣ
   ΕΞΟΔΟΣ
      f1: INTEGER;
ΔΕΔΟΜΕΝΑ
   a,x:INTEGER; /* Topikes metavlites tis
                   f1*/
ΑΡΧΗ
   a:=2;
   x:=0;
   /* Exoyme diplo onoma stin x.
      Epikratei to topiko onoma */
   ΤΥΠΩΣΕ(EOLN,"f1: a=",a,",x=",x);
   f1:=0;
ΤΕΛΟΣ-ΣΥΝΑΡΤΗΣΗΣ
```

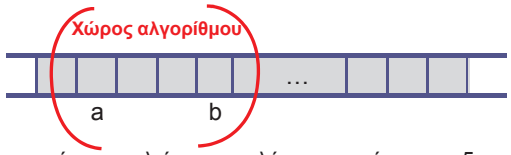
```
ΣΥΝΑΡΤΗΣΗ f2(): INTEGER
ΔΙΕΠΑΦΗ
   ΕΙΣΟΔΟΣ
   ΕΞΟΔΟΣ
      f2: INTEGER;
ΔΕΔΟΜΕΝΑ
   a:INTEGER; /* Topikes metavlites
                 tis f2*/
ΑΡΧΗ
   a:=8;
   x:=7; /*Anaferetai sti katholiki x */
   ΤΥΠΩΣΕ(EOLN,"f2: a=",a,",x=",x);
   f2:=0;
ΤΕΛΟΣ-ΣΥΝΑΡΤΗΣΗΣ

ΑΡΧΗ
   x:=5;
   ΤΥΠΩΣΕ(EOLN,"ΑΛΓ: x=",x);
   ret:=f1();
   ΤΥΠΩΣΕ(EOLN,"ΑΛΓ: x=",x);
   ret:=f2();
   ΤΥΠΩΣΕ(EOLN,"ΑΛΓ: x=",x);
ΤΕΛΟΣ
```

Α. Συναρτήσεις

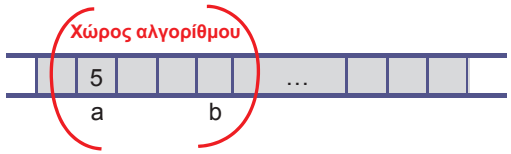
3. Πως Λειτουργούν οι Συναρτήσεις

1. Συναρτήσεις και Χώρος στη Μνήμη

- Είναι σημαντικό να καταλάβουμε ότι κάθε συνάρτηση έχει το δικό της «χώρο» στη μνήμη, στον οποίο αποθηκεύει τις μεταβλητές της.
- Για παράδειγμα έστω το τμήμα κώδικα που φαίνεται στα δεξιά
- Όταν ξεκινάει να εκτελείται ο κώδικας υπάρχει ο χώρος αποθήκευσης μόνο για τον αλγόριθμο!



- Έπειτα όταν εκτελείται η εντολή αρχικοποίησης a:=5, η κατάσταση της μνήμης είναι:



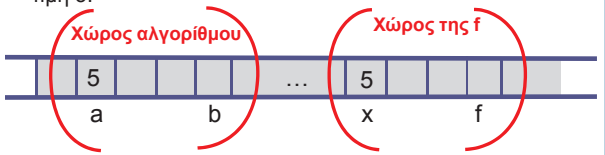

```
ΑΛΓΟΡΙΘΜΟΣ functions
ΔΕΔΟΜΕΝΑ
  a,b: INTEGER;

ΣΥΝΑΡΤΗΣΗ f(x):INTEGER
ΔΙΕΠΑΦΗ
  ΕΙΣΟΔΟΣ
    x:INTEGER
  ΕΞΟΔΟΣ
    f:INTEGER
ΑΡΧΗ
   f:=x*x;
ΤΕΛΟΣ-ΣΥΝΑΡΤΗΣΗΣ

ΑΡΧΗ
  a:=5;
  b:=f(a);
ΤΕΛΟΣ
```


Α. Συναρτήσεις

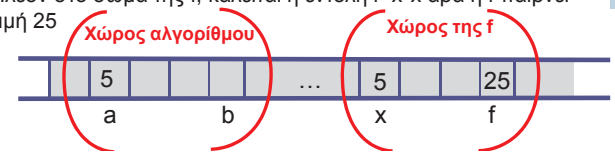
3. Πως Λειτουργούν οι Συναρτήσεις

1. Συναρτήσεις και Χώρος στη Μνήμη

- Έπειτα καλείται η f με όρισμα a.
  - Προσοχή! Αυτό σημαίνει, ότι δημιουργείται χώρος αποθήκευσης για την f.
  - Και στον χώρο αποθήκευσης της f, η μεταβλητή x θα πάρει την τιμή του ορίσματος που βάλαμε, άρα η x θα πάρει την τιμή 5.



  - Είναι σημαντικό να καταλάβουμε ότι από εδώ και πέρα η x δεν έχει καμία σχέση με την a.
- Πλέον στο σώμα της f, καλείται η εντολή f=x*x άρα η f παίρνει τιμή 25



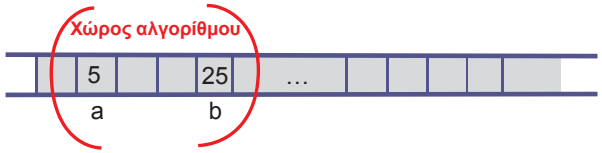

```
ΑΛΓΟΡΙΘΜΟΣ functions
ΔΕΔΟΜΕΝΑ
  a,b: INTEGER;

ΣΥΝΑΡΤΗΣΗ f(x):INTEGER
ΔΙΕΠΑΦΗ
  ΕΙΣΟΔΟΣ
    x:INTEGER
  ΕΞΟΔΟΣ
    f:INTEGER
ΑΡΧΗ
   f:=x*x;
ΤΕΛΟΣ-ΣΥΝΑΡΤΗΣΗΣ

ΑΡΧΗ
  a:=5;
  b:=f(a);
ΤΕΛΟΣ
```


Α. Συναρτήσεις

3. Πως Λειτουργούν οι Συναρτήσεις

1. Συναρτήσεις και Χώρος στη Μνήμη

- Η συνάρτηση ολοκληρώνει την λειτουργία της, έχοντας αποθηκεύσει στην f την τιμή 25 που είναι και η επιστρεφόμενη τιμή. Αυτό σημαίνει ότι επιστρέφουμε στον αλγόριθμο!
  - Η επιστρεφόμενη τιμή (25) αποθηκεύεται στην μεταβλητή b.
  - Είναι σημαντικό ότι μετά την επιστροφή τιμής ο χώρος της f, απελευθερώνεται για να μπορεί να χρησιμοποιηθεί από άλλες συναρτήσεις:




```
ΑΛΓΟΡΙΘΜΟΣ functions
ΔΕΔΟΜΕΝΑ
  a,b: INTEGER;

ΣΥΝΑΡΤΗΣΗ
f(x):INTEGER
ΔΙΕΠΑΦΗ
  ΕΙΣΟΔΟΣ
    x:INTEGER
  ΕΞΟΔΟΣ
    f:INTEGER
ΑΡΧΗ
   f:=x*x;
ΤΕΛΟΣ-ΣΥΝΑΡΤΗΣΗΣ

ΑΡΧΗ
  a:=5;
  b:=f(a);
ΤΕΛΟΣ
```

- Το παράδειγμα αυτό αναδεικνύει δύο σημαντικά θέματα!
  - Κάθε κλήση συνάρτησης δημιουργεί τον δικό της χώρο στην μνήμη!
  - Παρόλο που όλες οι συναρτήσεις έχουν πρόσβαση στις καθολικές μεταβλητές του αλγορίθμου, είναι καλό να θεωρήσουμε (γιατί έτσι είναι στις γλώσσες προγραμματισμού):
    - Ότι ο μόνος δίαυλος επικοινωνίας με την καλούσα συνάρτηση είναι τα ορίσματα (στην αρχή) και η επιστρεφόμενη τιμή (στο τέλος)

Β. Διαδικασίες

1. Πότε Γράφουμε Διαδικασίες

- Ορίζουμε διαδικασίες σε δύο (+μισή) περιπτώσεις:
  - Επιθυμούμε να γράψουμε μια «συνάρτηση» που δεν επιστρέφει τιμή, αλλά:
    - Π.χ: Κάνει μια συγκεκριμένη δουλειά που δεν έχει έξοδο, όπως π.χ. να τυπώνει τα περιεχόμενα ενός πίνακα στην οθόνη.
  - Επιθυμούμε η «συνάρτηση» να έχει περισσότερες από μία εξόδους
    - Π.χ: Θα θέλαμε να υλοποιήσουμε μία εκδοχή της δυαδικής αναζήτησης που να επιστρέφει όχι μόνο αν βρέθηκε το στοιχείο που αναζητούμε, αλλά και την θέση στο οποίο το εντοπίσαμε.
  - Επιθυμούμε η «συνάρτηση» να επεμβαίνει στα δεδομένα που δέχεται ως ορίσματα και αυτή η αλλαγή να είναι ορατή και εκτός της συνάρτησης.
    - Π.χ. επιθυμούμε να γράψουμε μια συνάρτηση που να παίρνει σαν όρισμα έναν πίνακα και να τον ταξινομεί. Αυτή η αλλαγή στον πίνακα να είναι ορατή και στο πέρας της διαδικασίας.

Β. Διαδικασίες
2. Πως Γράφουμε Διαδικασίες

- Το συντακτικό των διαδικασιών είναι ίδιο με αυτό των συναρτήσεων με μία βασική διαφορά: Τα ορίσματα πλέον μπορούν να είναι και εγγραφής, εκτός από ανάγνωσης-μόνο ώστε να επιτύχουν τους στόχους που θέσαμε στην προηγούμενη διαφάνεια:
  - Ένα όρισμα που είναι **ανάγνωσης μόνο** συντάσσεται ακριβώς όπως στις διαδικασίες.
    - Ένα τέτοιο όρισμα θα μπορούμε να το χρησιμοποιήσουμε στην διαδικασία μας, αλλά τυχόν αλλαγές στην τιμή δεν είναι εφικτές.
    - Η κλήση που γίνεται ονομάζεται και **κλήση-με-τιμή (call-by-value)**.
  - Ένα όρισμα που είναι **ανάγνωσης και εγγραφής** συντάσσεται με τον ίδιο τρόπο, αλλά στην δήλωση της διαδικασίας, βάζουμε μπροστά το % από το όρισμα.
    - Ένα τέτοιο όρισμα θα μπορούμε να το χρησιμοποιήσουμε στην διαδικασία μας, και τυχόν αλλαγές στην τιμή θα είναι ορατές έξω από την διαδικασία.
    - Η κλήση που γίνεται ονομάζεται και **κλήση-με-αναφορά (call-by-reference)**.
    - Στην κλήση που γίνεται βάζουμε το % μπροστά από το όρισμα.
- Η κλήση των διαδικασιών γίνεται με την λέξη «ΥΠΟΛΟΓΙΣΕ» μπροστά από την κλήση.

Β. Διαδικασίες
2. Πως Γράφουμε Διαδικασίες
1. Κλήση με Τιμή

```
ΑΛΓΟΡΙΘΜΟΣ call_by_value
ΔΕΔΟΜΕΝΑ
   x:INTEGER;

ΔΙΑΔΙΚΑΣΙΑ f(a)
ΔΙΕΠΑΦΗ
   ΕΙΣΟΔΟΣ
      a: INTEGER;                  /* Οι μεταβλητές εισόδου ορίζονται στην είσοδο */
ΕΞΟΔΟΣ
                                   /* Δεν υπάρχει έξοδος */
ΔΕΔΟΜΕΝΑ
                                   /* Κενό Τμήμα Δεδομένων */
ΑΡΧΗ
   a:=2;                            /* ΔΕΝ ΔΟΥΛΕΥΕΙ αλλαγή τιμής σε call by value */
   ΤΥΠΩΣΕ(EOLN,"f: a=",a);
ΤΕΛΟΣ-ΔΙΑΔΙΚΑΣΙΑΣ

ΑΡΧΗ
   x:=0;
   ΤΥΠΩΣΕ(EOLN,"ΑΛΓ: x=",x);       /* Τυπώνει 0 */
   ΥΠΟΛΟΓΙΣΕ f(x);                 /* Κλήση της διαδικασίας με την λέξη ΥΠΟΛΟΓΙΣΕ */
   ΤΥΠΩΣΕ(EOLN,"ΑΛΓ: x=",x);      /* Τυπώνει 0.Η αλλαγή της τιμής δεν διατηρήθηκε */
ΤΕΛΟΣ
```


Β. Διαδικασίες
2. Πως Γράφουμε Διαδικασίες
2. Κλήση με Αναφορά

```
ΑΛΓΟΡΙΘΜΟΣ call_by_reference
ΔΕΔΟΜΕΝΑ
   x:INTEGER;

ΔΙΑΔΙΚΑΣΙΑ f(%a)
ΔΙΕΠΑΦΗ
   ΕΙΣΟΔΟΣ
      a: INTEGER;                /* Το a είναι ανάγνωσης. Το γράφουμε στην είσοδο */
ΕΞΟΔΟΣ
      a: INTEGER;                /* Αλλά και εγγραφής. Το γράφουμε και στην έξοδο */
ΔΕΔΟΜΕΝΑ
                                 /* Κενό Τμήμα Δεδομένων */
ΑΡΧΗ
   a:=2;
   ΤΥΠΩΣΕ(EOLN,"f: a=",a);       /* Τυπώνει 2 */
ΤΕΛΟΣ-ΔΙΑΔΙΚΑΣΙΑΣ

ΑΡΧΗ
   x:=0;
   ΤΥΠΩΣΕ(EOLN,"ΑΛΓ: x=",x);     /* Τυπώνει 0 */
   ΥΠΟΛΟΓΙΣΕ f(%x);               /* Κλήση της διαδικασίας με την λέξη ΥΠΟΛΟΓΙΣΕ */
   ΤΥΠΩΣΕ(EOLN,"ΑΛΓ: x=",x);     /* Τυπώνει 2.Η αλλαγή της τιμής διατηρήθηκε */
ΤΕΛΟΣ
```


…και ένα σχόλιο

- Το συντακτικό των συναρτήσεων και των διαδικασιών που μελετήσαμε ήταν σχετικά απλό.
- Το θέμα όμως δεν είναι το συντακτικό, αλλά η αξιοποίηση των συναρτήσεων ως προγραμματιστικά εργαλεία. Αυτό μπορεί να γίνει κατανοητό μόνο με την επίλυση ασκήσεων.
- Ως εκ τούτου όλες οι εφαρμογές που ακολουθούν απαιτείται να μελετηθούν σε βάθος, διότι αναδεικνύουν σημαντικούς προγραμματιστικούς συλλογισμούς.
- Επιπλέον σταδιακά είμαστε σε θέση να χτίσουμε και τα πρώτα μας αξιόλογα προγράμματα, όπως και θα το κάνουμε στις επόμενες εφαρμογές.

Γ. Ασκήσεις

Εφαρμογή 1: Συναρτήσεις Ελέγχου Εισόδου

- Ορίστε την συνάρτηση:
  - `get_integer(start, finish)`: Θα λαμβάνει ως είσοδο ένα εύρος τιμών ακεραίων `[start…finish]` και θα διαβάζει έναν ακέραιο σε αυτό το εύρος. Θα επιστρέφει τον αριθμό που διαβάστηκε.
- Ορίστε τον αλγόριθμο να διαβάζει δύο ακέραιους `a,b` στο διάστημα 1..10 και έναν ακέραιο `n` στο διάστημα 2..5 και θα υπολογίζει την ποσότητα n*(a-b) και θα χρησιμοποιεί τη συνάρτηση που ορίσατε.

Γ. Ασκήσεις

Εφαρμογή 2: Μια Βιβλιοθήκη Μελέτης Αριθμών

- Ορίζουμε τις συναρτήσεις:
  - `is_even(n)`: Θα επιστρέφει TRUE ή FALSE ανάλογα με το αν ο αριθμός n είναι άρτιος
  - `is_odd(n)`: Θα επιστρέφει TRUE ή FALSE ανάλογα με το αν ο αριθμός n είναι περιττός
  - `is_square(n)`: Θα επιστρέφει TRUE ή FALSE ανάλογα με το αν ο αριθμός n είναι τετράγωνο ενός φυσικού
  - `is_cube(n)`: Θα επιστρέφει TRUE ή FALSE ανάλογα με το αν ο αριθμός n είναι κύβος ενός φυσικού
- Ορίζουμε την main που θα ζητάει από το χρήστη είτε να εισάγει έναν αριθμό και θα εξετάζει αν ο αριθμός έχει κάποιες από αυτές τις ιδιότητες.
- Παράδειγμα Εκτέλεσης:

```
Εισάγετε τον αριθμό: 8
Είναι Άρτιος
Είναι Κύβος Αριθμού
```

```
Εισάγετε τον αριθμό: 9
Είναι Περιττός
Είναι Τετράγωνο Αριθμού
```


Γ. Ασκήσεις

Εφαρμογή 3: Πρώτοι Αριθμοί

- Ένας φυσικός αριθμός λέμε ότι είναι πρώτος αν διαιρείται (ακριβώς) μόνο με τον εαυτό του και τη μονάδα. Το 1 θεωρείται ότι δεν είναι πρώτος.
- Κατασκευάστε ένα πρόγραμμα το οποίο:
  - Θα ορίζει μία συνάρτηση με όνομα is_prime(n) η οποία θα δέχεται ως όρισμα έναν ακέραιο αριθμό n, θα εξετάζει αν είναι πρώτος και θα επιστρέφει TRUE αν είναι πρώτος και FALSE αν δεν είναι.
  - Ο αλγόριθμος θα διαβάζει δύο φυσικούς (ελέγχοντας στην είσοδο να είναι >0) που θα ορίζουν την αρχή και το τέλος ενός κλειστού διαστήματος (π.χ. a=5, b=8) και θα τυπώνει τους φυσικούς σε αυτό το διάστημα που είναι πρώτοι.
- Παράδειγμα εκτέλεσης του ζητούμενου προγράμματος:

```
Εισάγετε την αρχή του διαστήματος: 5
Εισάγετε το πέρας του διαστήματος: 15

Το 5 είναι πρώτος
Το 7 είναι πρώτος
Το 11 είναι πρώτος
Το 13 είναι πρώτος
```


Γ. Ασκήσεις

Εφαρμογή 4: Ανταλλαγή τιμών μεταβλητών (swap)

- Ορίστε μια διαδικασία με όνομα swap που να παίρνει δύο ορίσματα ανάγνωσης εγγραφής και να ανταλλάσει τις τιμές τους
- Επειτα κατασκευάστε ένα πρόγραμμα που να κανει την ακόλουθη εκτύπωση και να χρησιμοποιεί την swap:

```
Δώσε το a: 5
Δώσε το b: 8

Τιμές: a=8, b=5
Γίνεται ανταλλαγή των τιμών
Νέες Τιμές: a=5, b=8
```

Γ. Ασκήσεις

Εφαρμογή 5.1: Διαδικασία: Ανάγνωση Πίνακα

Πολλές φορές στο προηγούμενο μάθημα γράψαμε κώδικα που διαβάζει τα περιεχόμενα ενός πίνακα.

- Μετατρέψτε σε διαδικασία τον κώδικα αυτό, με όνομα read_array. Θα παίρνει δύο ορίσματα τον πίνακα και το μέγεθος του και θα διαβάζει τα στοιχεία του πίνακα.

Γ. Ασκήσεις

Εφαρμογή 5.2: Διαδικασία: Εκτύπωση Πίνακα

Πολλές φορές στο προηγούμενο μάθημα γράψαμε κώδικα που εκτυπώνει τα περιεχόμενα ενός πίνακα στην οθόνη.

- Μετατρέψτε σε διαδικασία τον κώδικα αυτό, με όνομα print_array. Θα παίρνει δύο ορίσματα τον πίνακα και το μέγεθος του και θα τυπώνει στην οθόνη τα περιεχόμενα του πίνακα.

Γ. Ασκήσεις

Εφαρμογή 5.3: Συνάρτηση: Ελάχιστος Πίνακα

Στο προηγούμενο μάθημα κατασκευάσαμε έναν αλγόριθμο που υπολογίζει τον ελάχιστο ενός πίνακα.

- Μετατρέψτε σε συνάρτηση τον αλγόριθμο εύρεσης ελαχίστου, με όνομα min_array. Θα παίρνει δύο ορίσματα τον πίνακα και το μέγεθος του και θα επιστρέφει τον ελάχιστο αριθμό του πίνακα.

Γ. Ασκήσεις

Εφαρμογή 5.4: Συνάρτηση: Μέγιστος Πίνακα

Στο προηγούμενο μάθημα κατασκευάσαμε έναν αλγόριθμο που υπολογίζει τον μέγιστο ενός πίνακα.

- Μετατρέψτε σε συνάρτηση τον αλγόριθμο εύρεσης μεγίστου, με όνομα max_array. Θα παίρνει δύο ορίσματα τον πίνακα και το μέγεθος του και θα επιστρέφει τον ελάχιστο αριθμό του πίνακα.

Γ. Ασκήσεις

Εφαρμογή 5.5: Συνάρτηση: Άθροισμα στοιχείων Πίνακα

Στο προηγούμενο μάθημα κατασκευάσαμε έναν αλγόριθμο που υπολογίζει το άθροισμα των στοιχείων ενός πίνακα

- Μετατρέψτε σε συνάρτηση τον αλγόριθμο εύρεσης μέσου όρου, με όνομα mo_array. Θα παίρνει δύο ορίσματα τον πίνακα και το μέγεθος του και θα επιστρέφει τον ελάχιστο αριθμό του πίνακα.

Γ. Ασκήσεις

Εφαρμογή 5.6: Συνάρτηση: Γινόμενο στοιχείων Πίνακα

Στο προηγούμενο μάθημα κατασκευάσαμε έναν αλγόριθμο που υπολογίζει το άθροισμα των στοιχείων ενός πίνακα

- Μετατρέψτε σε συνάρτηση τον αλγόριθμο εύρεσης μέσου όρου, με όνομα mo_array. Θα παίρνει δύο ορίσματα τον πίνακα και το μέγεθος του και θα επιστρέφει τον ελάχιστο αριθμό του πίνακα.

Γ. Ασκήσεις

Εφαρμογή 5.7: Συνάρτηση: Μέσος Όρος Στοιχείων Πίνακα

Στο προηγούμενο μάθημα κατασκευάσαμε έναν αλγόριθμο που υπολογίζει τον μέσο όρο των στοιχείων ενός πίνακα.

- Μετατρέψτε σε συνάρτηση τον αλγόριθμο εύρεσης μέσου όρου, με όνομα mo_array. Θα παίρνει δύο ορίσματα τον πίνακα και το μέγεθος του και θα επιστρέφει τον ελάχιστο αριθμό του πίνακα.

Γ. Ασκήσεις

Εφαρμογή 5.8: Πρόγραμμα: Μελέτη Πινάκων

Κάνοντας χρήση των προηγούμενων συναρτήσεων:

- Κατασκευάστε έναν αλγόριθμο που θα διαβάζει έναν πίνακα Ν θέσεων (5…10) με αμυντικό προγραμματισμό
- Θα εμφανίζει τον πίνακα που διαβάστηκε στην οθόνη.
- Θα υπολογίζει και θα εκτυπώνει διαδοχικά τον ελάχιστο, τον μέγιστο, το άθροισμα, το γινόμενο και τον μέσο όρο των στοιχείων του πίνακα.

Γ. Ασκήσεις

Εφαρμογή 6.1: Συνάρτηση: Σειριακή Αναζήτηση

Στο προηγούμενο μάθημα κατασκευάσαμε τον αλγόριθμο Σειριακής Αναζήτησης που αναζητά αν ένα στοιχείο υπάρχει σε έναν πίνακα.

- Μετατρέψτε σε συνάρτηση τον αλγόριθμο, με όνομα linear_search. Θα παίρνει τρια ορίσματα τον πίνακα, το μέγεθος του και το προς αναζήτηση στοιχείο και θα επιστρέφει TRUE αν το στοιχείο υπάρχει στον πίνακα και FALSE αν το στοιχείο δεν υπάρχει στον πίνακα.

Γ. Ασκήσεις

Εφαρμογή 6.2: Συνάρτηση: Δυαδική Αναζήτηση

Στο προηγούμενο μάθημα κατασκευάσαμε τον αλγόριθμο Δυαδικής Αναζήτησης που αναζητά αν ένα στοιχείο υπάρχει σε έναν πίνακα.

- Μετατρέψτε σε συνάρτηση τον αλγόριθμο, με όνομα binary_search. Θα παίρνει τρια ορίσματα τον πίνακα, το μέγεθος του και το προς αναζήτηση στοιχείο και θα επιστρέφει TRUE αν το στοιχείο υπάρχει στον πίνακα και FALSE αν το στοιχείο δεν υπάρχει στον πίνακα.

Γ. Ασκήσεις

Εφαρμογή 6.3: Πρόγραμμα: Αναζήτηση

Κάνοντας χρήση των προηγούμενων συναρτήσεων:

- Κατασκευάστε έναν αλγόριθμο που θα διαβάζει έναν πίνακα Ν θέσεων (5…10) με αμυντικό προγραμματισμό (χρησιμοποιήστε την συνάρτηση get_integer και κατασκευάστε μια συνάρτηση για το διάβασμα των στοιχείων του πίνακα).
- Θα ρωτά το χρήστη ποιον αριθμό επιθυμεί να αναζητήσει στον πίνακα και να επιλέξει τον αλγόριθμο που θα εκτελέσει (σειριακή ή δυαδική αναζήτηση).
- Τελικά θα εμφανίζει αν το στοιχείο υπάρχει ή δεν υπάρχει στον πίνακα.

Γ. Ασκήσεις

Εφαρμογή 7.1: Διαδικασία: Ταξινόμηση με Επιλογή

Στο προηγούμενο μάθημα κατασκευάσαμε τον αλγόριθμο Selection Sort που ταξινομεί έναν πίνακα ακεραίων.

- Μετατρέψτε σε διαδικασία τον αλγόριθμο, με όνομα selection_sort. Θα παίρνει δύο ορίσματα τον πίνακα και το μέγεθος του.

Γ. Ασκήσεις

Εφαρμογή 7.2: Διαδικασία: Ταξινόμηση με Εισαγωγή

Στο προηγούμενο μάθημα κατασκευάσαμε τον αλγόριθμο Insertion Sort που ταξινομεί έναν πίνακα ακεραίων.

- Μετατρέψτε σε διαδικασία τον αλγόριθμο, με όνομα insertion_sort. Θα παίρνει δύο ορίσματα τον πίνακα και το μέγεθος του.

Γ. Ασκήσεις

Εφαρμογή 7.3: Διαδικασία: Ταξινόμηση με Εισαγωγή

Στο προηγούμενο μάθημα κατασκευάσαμε τον αλγόριθμο Bubble Sort που ταξινομεί έναν πίνακα ακεραίων.

- Μετατρέψτε σε διαδικασία τον αλγόριθμο, με όνομα bubble_sort. Θα παίρνει δύο ορίσματα τον πίνακα και το μέγεθος του.

Γ. Ασκήσεις

Εφαρμογή 7.4: Πρόγραμμα: Ταξινόμηση Πίνακα

Κάνοντας χρήση των προηγούμενων συναρτήσεων:

- Κατασκευάστε έναν αλγόριθμο που θα διαβάζει έναν πίνακα Ν θέσεων (5…10) με αμυντικό προγραμματισμό (χρησιμοποιήστε την συνάρτηση get_integer και κατασκευάστε μια συνάρτηση για το διάβασμα των στοιχείων του πίνακα).
- Θα ρωτά το χρήστη ποιον αλγόριθμο ταξινόμησης επιθυμεί να εκτελέσει.
- Τελικά θα εμφανίζει τον ταξινομημένο πίνακα.