DIDAKTIKH

00 ΑΕΠΠ - A BRIEF UNDERSTANDING

>_ Δρ. Στυλιανός Καραγιάννης, Ιόνιο Πανεπιστήμιο,
skaragiannis@ionio.gr

Ενότητα 1: Εισαγωγή στη ΓΛΩΣΣΑ

Η **ΓΛΩΣΣΑ** είναι μια τυποποιημένη ψευδογλώσσα που βοηθά τους μαθητές να εξοικειωθούν με τις βασικές αρχές του προγραμματισμού και της αλγοριθμικής. Είναι απλή στη σύνταξη, υποστηρίζοντας τους παρακάτω τύπους δεδομένων:

- **Ακέραιοι**
- **Πραγματικοί**
- **Χαρακτήρες**
- **Λογικοί**

Ενότητα 2: Σύνταξη και Δομή Προγράμματος

Ένα πρόγραμμα σε **ΓΛΩΣΣΑ** αποτελείται από τις παρακάτω ενότητες:

1.Δήλωση μεταβλητών

2.Κεντρικό πρόγραμμα που εκτελεί τις εντολές

```
ΠΡΟΓΡΑΜΜΑ Υπολογισμός_Μισθού
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: ώρες, μισθός_ανά_ώρα
  ΠΡΑΓΜΑΤΙΚΕΣ: μικτός_μισθός
ΑΡΧΗ
  ΓΡΑΨΕ "Δώσε ώρες εργασίας:"
  ΔΙΑΒΑΣΕ ώρες
  ΓΡΑΨΕ "Δώσε μισθό ανά ώρα:"
  ΔΙΑΒΑΣΕ μισθός_ανά_ώρα
  μικτός_μισθός <- ώρες * μισθός_ανά_ώρα
  ΓΡΑΨΕ "Ο μικτός μισθός είναι:", μικτός_μισθός
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Ενότητα 3: Μεταβλητές και Τύποι Δεδομένων

Οι **μεταβλητές** χρησιμοποιούνται για να αποθηκεύουν τιμές δεδομένων. Δηλώνονται στο τμήμα **ΜΕΤΑΒΛΗΤΕΣ** και μπορεί να είναι τύπου ακέραιος, πραγματικός, χαρακτήρας ή λογικός.

```
Πρόγραμμα ΜεταβλΠαράδειγμα
Μεταβλητές
  Ακέραιες: ηλικία, έτος
  Πραγματικές: βαθμός, ύψος
  Χαρακτήρες: αρχικό
  Λογικές: πέρασε
ΑΡΧΗ
  ηλικία <- 20
  έτος <- 2024
  βαθμός <- 8.5
  ύψος <- 1.75
  αρχικό <- 'Σ'
  πέρασε <- Αληθής
  Γράψε ηλικία
Τέλος_Προγράμματος
```

Ενότητα 4: Εντολές Εισόδου-Εξόδου

Η εντολή **ΔΙΑΒΑΣΕ** χρησιμοποιείται για την εισαγωγή δεδομένων από τον χρήστη, ενώ η εντολή **ΓΡΑΨΕ** εμφανίζει αποτελέσματα.

```
...
...
ΔΙΑΒΑΣΕ αριθμός
ΓΡΑΨΕ "Ο αριθμός που έδωσες είναι:", αριθμός
...
...
```

Ενότητα 5: Τελεστές και Εκφράσεις

Οι **αριθμητικοί τελεστές** είναι:

- Πρόσθεση (+)
- Αφαίρεση (-)
- Πολλαπλασιασμός (*)
- Διαίρεση (/)

```
άθροισμα <- 5 + 10
διαφορά <- 10 - 5
πολλαπλασιαμός <- 5 * 2
διαίρεση <- 10 / 2
```

Δομή Ακολουθίας

Η δομή ακολουθίας είναι η πιο απλή δομή προγραμματισμού και αποτελείται από μια σειρά εντολών που εκτελούνται η μία μετά την άλλη, με τη σειρά που έχουν γραφτεί. Στο παραπάνω παράδειγμα, οι εντολές εκτελούνται με τη σειρά: διαβάζεται η τιμή, υπολογίζεται η έκπτωση και στη συνέχεια υπολογίζεται η νέα τιμή με την έκπτωση.

```
ΠΡΟΓΡΑΜΜΑ Υπολογισμός_Έκπτωσης
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ: τιμή, έκπτωση, νέα_τιμή
ΑΡΧΗ
  ΓΡΑΨΕ "Δώσε την αρχική τιμή: "
  ΔΙΑΒΑΣΕ τιμή
  έκπτωση <- τιμή * 0.1
  νέα_τιμή <- τιμή - έκπτωση
  ΓΡΑΨΕ "Η τελική τιμή με έκπτωση είναι: ", νέα_τιμή
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Δομή Ακολουθίας

Σε αυτό το παράδειγμα, αν η τιμή της μεταβλητής ηλικία είναι 18 ή μεγαλύτερη, το πρόγραμμα εμφανίζει "Ενήλικας". Διαφορετικά, εμφανίζει "Ανήλικος".

```
ΠΡΟΓΡΑΜΜΑ Κατηγοριοποίηση_Ηλικίας
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: ηλικία
ΑΡΧΗ
  ΓΡΑΨΕ "Δώσε την ηλικία: "
  ΔΙΑΒΑΣΕ ηλικία
  ΑΝ ηλικία >= 18 ΤΟΤΕ
    ΓΡΑΨΕ "Ενήλικας"
  ΑΛΛΙΩΣ
    ΓΡΑΨΕ "Ανήλικος"
  ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Δομή Επανάληψης

Σε αυτό το παράδειγμα, το πρόγραμμα συνεχίζει να διαβάζει αριθμούς και να τους προσθέτει στο άθροισμα, μέχρι ο χρήστης να εισάγει τον αριθμό 0.

```
ΠΡΟΓΡΑΜΜΑ Άθροισμα_Αριθμών
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: αριθμός, άθροισμα
ΑΡΧΗ
  άθροισμα <- 0
  ΓΡΑΨΕ "Δώσε έναν αριθμό (0 για έξοδο): "
  ΔΙΑΒΑΣΕ αριθμός
  ΟΣΟ αριθμός ≠ 0 ΕΠΑΝΑΛΑΒΕ
    άθροισμα <- άθροισμα + αριθμός
    ΓΡΑΨΕ "Δώσε έναν αριθμό (0 για έξοδο): "
    ΔΙΑΒΑΣΕ αριθμός
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  ΓΡΑΨΕ "Το συνολικό άθροισμα είναι: ", άθροισμα
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Δομή Επανάληψης

Εδώ, η δομή ΓΙΑ επαναλαμβάνει τις εντολές 10 φορές, προσθέτοντας κάθε φορά την τιμή του i στο άθροισμα. Στο τέλος, το πρόγραμμα εμφανίζει το άθροισμα των αριθμών από 1 έως 10.

```
ΠΡΟΓΡΑΜΜΑ Άθροισμα_Πρώτων_Δέκα
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: άθροισμα, i
ΑΡΧΗ
  άθροισμα <- 0
  ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 10
    άθροισμα <- άθροισμα + i
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  ΓΡΑΨΕ "Το άθροισμα των πρώτων 10 αριθμών είναι: ",
άθροισμα
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Ενότητα 6: Δομές Ελέγχου - Η Εντολή ΑΝ

Η δομή **ΑΝ...ΤΟΤΕ...ΑΛΛΙΩΣ** επιτρέπει την εκτέλεση εντολών με βάση μια συνθήκη.

```
ΑΝ βαθμός >= 50 ΤΟΤΕ
  ΓΡΑΨΕ "Πέρασες το μάθημα!"
ΑΛΛΙΩΣ
  ΓΡΑΨΕ "Απέτυχες."
ΤΕΛΟΣ_ΑΝ
```

Ενότητα 7: Δομή Επανάληψης - Όσο...επανάλαβε

Η **εντολή ΟΣΟ...ΕΠΑΝΑΛΑΒΕ** επιτρέπει την εκτέλεση εντολών μέχρι να ικανοποιηθεί μια συνθήκη.

```
ΟΣΟ αριθμός < 10 ΕΠΑΝΑΛΑΒΕ
  ΓΡΑΨΕ "Ο αριθμός είναι μικρότερος του 10."
  αριθμός <- αριθμός + 1
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Ενότητα 8: Δομή Επανάληψης - Για...από...μέχρι

Η **εντολή ΓΙΑ...ΑΠΟ...ΜΕΧΡΙ** είναι μια άλλη δομή επανάληψης.

```
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 5
  ΓΡΑΨΕ "Η τιμή του i είναι:", i
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

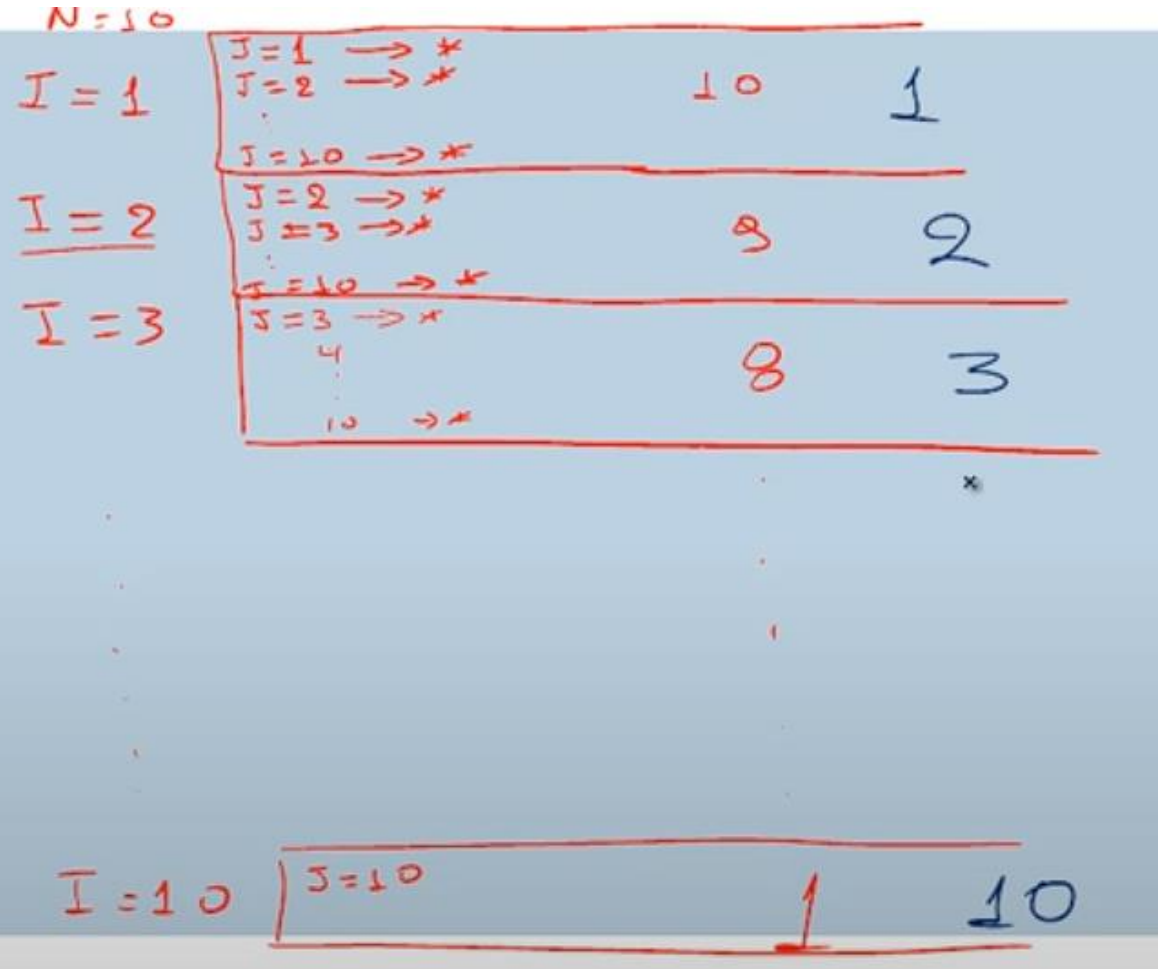
Σελίδα 9: Δομές Δεδομένων - Πίνακες

Οι **πίνακες** είναι δομές δεδομένων που επιτρέπουν την αποθήκευση πολλών τιμών.

```
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: πίνακας[5]
ΑΡΧΗ
  ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 5
    ΔΙΑΒΑΣΕ πίνακας[i]
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ
```

Ενότητα 8: Δομή Επανάληψης - Για...από...μέχρι

```
ΑΛΓΟΡΙΘΜΟΣ efarmogi6
ΔΕΔΟΜΕΝΑ
    I,J,N: INTEGER;
ΑΡΧΗ
    N:=10;
    ΓΙΑ I:=1 ΕΩΣ N ΕΠΑΝΑΛΑΒΕ
        ΓΙΑ J:=I ΕΩΣ N ΕΠΑΝΑΛΑΒΕ
            ΤΥΠΩΣΕ("*");
        ΓΙΑ-ΤΕΛΟΣ
        ΤΥΠΩΣΕ(EOLN);
    ΓΙΑ-ΤΕΛΟΣ
ΤΕΛΟΣ
```



Ενότητα 10: Εφαρμογή σε Πραγματικά Προβλήματα

```
ΠΡΟΓΡΑΜΜΑ Υπολογισμός_Μέσου
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: i, πλήθος
  ΠΡΑΓΜΑΤΙΚΕΣ: άθροισμα, μέσος_όρος,
αριθμός
ΑΡΧΗ
  άθροισμα <- 0
  ΓΡΑΨΕ "Δώσε το πλήθος των αριθμών:"
  ΔΙΑΒΑΣΕ πλήθος
  ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ πλήθος
    ΓΡΑΨΕ "Δώσε αριθμό:"
    ΔΙΑΒΑΣΕ αριθμός
    άθροισμα <- άθροισμα + αριθμός
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  μέσος_όρος <- άθροισμα / πλήθος
  ΓΡΑΨΕ "Ο μέσος όρος είναι:",
μέσος_όρος
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

```
# Initialize variables
sum_values = 0

# Get the number of values from the user
count = int(input("Enter the number of values:
"))

# Loop for the specified number of values
for i in range(1, count + 1):
    number = float(input("Enter a number: "))
    sum_values += number

# Calculate the average
average = sum_values / count

# Display the average
print("The average is:", average)
```

Ενότητα 11: Υποπρογράμματα – Διαδικασίες και Συναρτήσεις

Τα υποπρογράμματα είναι τμήματα κώδικα που μπορούν να κληθούν από το κύριο πρόγραμμα για την εκτέλεση μιας συγκεκριμένης λειτουργίας. Στη **ΓΛΩΣΣΑ**, τα υποπρογράμματα χωρίζονται σε:

- **Διαδικασίες**: Εκτελούν μια σειρά εντολών χωρίς να επιστρέφουν τιμή.
- **Συναρτήσεις**: Εκτελούν εντολές και επιστρέφουν μια τιμή.

```
ΔΙΑΔΙΚΑΣΙΑ Εμφάνισε_Μήνυμα
ΑΡΧΗ
  ΓΡΑΨΕ "Καλώς ήρθατε στο πρόγραμμα!"
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

```
ΣΥΝΑΡΤΗΣΗ Τετράγωνο(ακέραιος) : ΑΚΕΡΑΙΑ
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: αποτέλεσμα
ΑΡΧΗ
  αποτέλεσμα <- ακέραιος * ακέραιος
  ΤΕΤΡΑΓΩΝΟ <- αποτέλεσμα
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Ενότητα 12: Χρήση Συναρτήσεων και Διαδικασιών σε Πρόγραμμα

Στο κύριο πρόγραμμα, οι διαδικασίες και οι συναρτήσεις μπορούν να κληθούν με την κατάλληλη σύνταξη. Η χρήση τους βελτιώνει την αναγνωσιμότητα και την επαναχρησιμοποίηση του κώδικα.

```
ΠΡΟΓΡΑΜΜΑ Υπολογισμός_Κύβου
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: αριθμός, αποτέλεσμα
ΑΡΧΗ
  ΓΡΑΨΕ "Δώσε έναν αριθμό:"
  ΔΙΑΒΑΣΕ αριθμός
  αποτέλεσμα <- Τετράγωνο(αριθμός) * αριθμός
  ΓΡΑΨΕ "Ο κύβος του αριθμού είναι:", αποτέλεσμα
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Ενότητα 13: Αναδρομή

Η **αναδρομή** είναι μια τεχνική προγραμματισμού στην οποία ένα υποπρόγραμμα καλεί τον εαυτό του για να επιλύσει ένα πρόβλημα σε μικρότερα κομμάτια. Είναι χρήσιμη για την επίλυση προβλημάτων που μπορούν να αποδομηθούν επαναληπτικά.

Factorials

$n! = n(n-1)(n-2)\ldots 1$

$0! \equiv 1$ (by definition)

$1! = 1$

$2! = 2 \times 1 = 2$

$3! = 3 \times 2 \times 1 = 6$

```
ΣΥΝΑΡΤΗΣΗ Παραγοντικό(ακέραιος) : ΑΚΕΡΑΙΑ
ΑΡΧΗ
  ΑΝ ακέραιος = 0 ΤΟΤΕ
    Παραγοντικό <- 1
  ΑΛΛΙΩΣ
    Παραγοντικό <- ακέραιος * Παραγοντικό(ακέραιος - 1)
  ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Ενότητα 14: Δομές Δεδομένων - Πολυδιάστατοι Πίνακες

Οι **πολυδιάστατοι πίνακες** επιτρέπουν την αποθήκευση δεδομένων σε δύο ή περισσότερες διαστάσεις, κάτι ιδιαίτερα χρήσιμο για την επεξεργασία πινάκων ή πλεγμάτων.

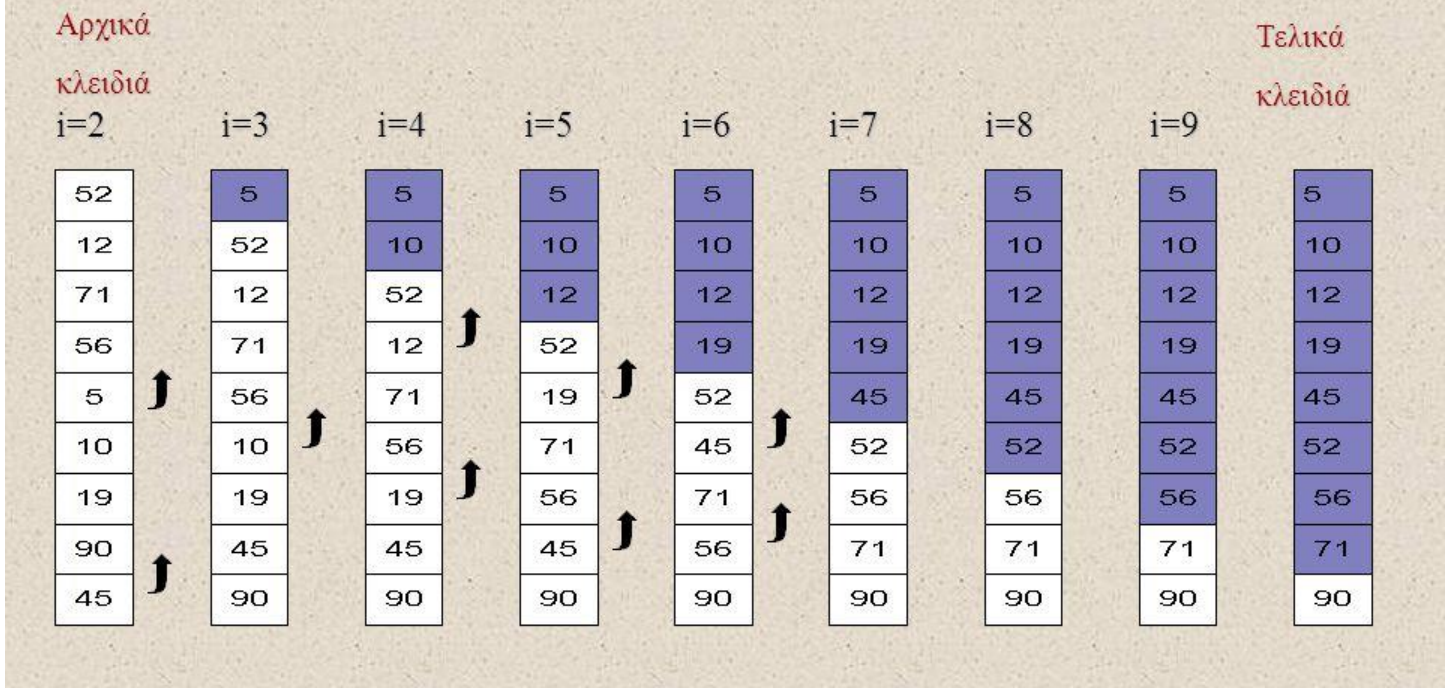
```
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: πίνακας[3, 3]
ΑΡΧΗ
  ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ 3
    ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ 3
      ΔΙΑΒΑΣΕ πίνακας[i, j]
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ
```

Ενότητα 15: Αλγόριθμοι Ταξινόμησης

Η ταξινόμηση είναι μια βασική λειτουργία στους αλγορίθμους, χρήσιμη για την οργάνωση δεδομένων. Η **φυσαλίδα** είναι μια απλή μέθοδος ταξινόμησης.

Παράδειγμα 1:Ταξινόμηση Φυσαλίδας

Έστω ότι ο αρχικός πίνακας αποτελείται από εννέα κλειδιά τα εξής:52,12,71,56,5,10,19,90 και 45. Η μέθοδος εφαρμοσμένη σε αυτά εννέα κλειδιά εξελίσσεται όπως φαίνεται στο επόμενο σχήμα.Κάθε φορά το ταξινομημένο τμήμα του πίνακα εμφανίζεται με χρώμα, ενώ τα στοιχεία που σαν φυσσαλίδες ανέρχονται μέσα στον πίνακα εντοπίζεται με το αντίστοιχο βέλος στα δεξιά τους.Κάθε φορά εμφανίζεται η τάξη της επανάληψης (i).



Ενότητα 15: Αλγόριθμοι Ταξινόμησης

Η ταξινόμηση είναι μια βασική λειτουργία στους αλγορίθμους, χρήσιμη για την οργάνωση δεδομένων. Η **φυσαλίδα** είναι μια απλή μέθοδος ταξινόμησης.

```
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ N - 1
  ΓΙΑ j ΑΠΟ 1 ΜΕΧΡΙ N - i
    ΑΝ πίνακας[j] > πίνακας[j + 1] ΤΟΤΕ
      temp <- πίνακας[j]
      πίνακας[j] <- πίνακας[j + 1]
      πίνακας[j + 1] <- temp
    ΤΕΛΟΣ_ΑΝ
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Ενότητα 16: Αναζήτηση σε Δεδομένα

Η αναζήτηση είναι άλλη μια βασική διαδικασία για την εύρεση στοιχείων μέσα σε μια συδιατρέχει κάθε στοιχείο του πίνακα έως ότου βρεθεί το ζητούμενο.λλογή δεδομένων. Ο **γραμμικός αλγόριθμος αναζήτησης**

```
βρέθηκε <- ΨΕΥΔΗΣ
ΓΙΑ i ΑΠΟ 1 ΜΕΧΡΙ N
  ΑΝ πίνακας[i] = ζητούμενο ΤΟΤΕ
    βρέθηκε <- ΑΛΗΘΗΣ
    ΕΞΟΔΟΣ_ΑΠΟ_ΕΠΑΝΑΛΗΨΗ
  ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΑΝ βρέθηκε ΤΟΤΕ
  ΓΡΑΨΕ "Το στοιχείο βρέθηκε."
ΑΛΛΙΩΣ
  ΓΡΑΨΕ "Το στοιχείο δεν βρέθηκε."
ΤΕΛΟΣ_ΑΝ
```

Παράδειγμα 1:Σειριακή αναζήτηση

1η Δοκιμή

key=63

i=1

Done=Ψευδής

→	24
	90
	38
	63
	16
	54
	71
	49

```
Όσο (done=ψευδής)
      και (i<=n) επανέλαβε
  Αν table[i]=key τότε
     done=αληθής
     position=i
Αλλιώς
     i=i+1
Τέλος_αν
```

key < > table[1]

63 < > 24

i=1+1=2

done= Ψευδής

Ενότητα 17: Δυαδική Αναζήτηση

Η **δυαδική αναζήτηση** είναι ένας αποδοτικός αλγόριθμος αναζήτησης που χρησιμοποιείται για να βρεθεί η θέση ενός στοιχείου σε έναν ταξινομημένο πίνακα. Η δυαδική αναζήτηση λειτουργεί μειώνοντας συνεχώς το εύρος αναζήτησης στο μισό, μέχρι να βρεθεί το στοιχείο ή να επιβεβαιωθεί ότι δεν υπάρχει στον πίνακα.

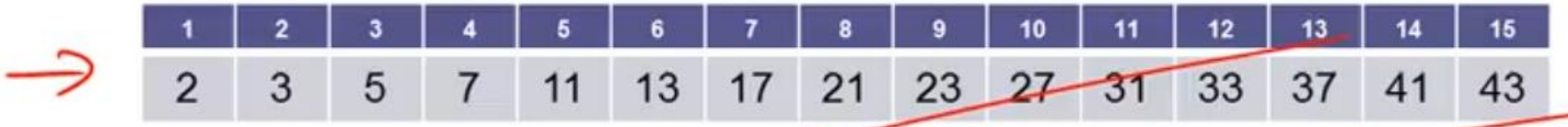
Η βασική ιδέα είναι να συγκρίνεται το στοιχείο που αναζητούμε με το μεσαίο στοιχείο του πίνακα:

1. Αν το μεσαίο στοιχείο είναι αυτό που ψάχνουμε, η αναζήτηση τελειώνει.
2. Αν το στοιχείο που ψάχνουμε είναι μικρότερο, η αναζήτηση περιορίζεται στο αριστερό μισό του πίνακα.
3. Αν είναι μεγαλύτερο, η αναζήτηση περιορίζεται στο δεξί μισό.

Αυτή η διαδικασία επαναλαμβάνεται, μειώνοντας το εύρος κατά το ήμισυ κάθε φορά, έως ότου βρεθεί το στοιχείο ή ο πίνακας δεν έχει άλλα στοιχεία να ελεγχθούν.

Ενότητα 17: Δυαδική Αναζήτηση

- Εκτελούμε τον αλγόριθμο ψάχνοντας το στοιχείο 11 στον πίνακα:



- Κλήση: BinarySearch(A,11,1,15): middle=(1+15) div 2=8. x<A[middle]

- Αναδρομική Κλήση: BinarySearch(A,11,1,7) : middle=(1+7) div 2=4 x>A[middle]

- Αναδρομική Κλήση: BinarySearch(A,11,5,7) : middle=(5+7) div 2=6 x<A[middle]

- Αναδρομική Κλήση: BinarySearch(A,11,5,5) : middle=(5+5) div 2=5 x=A[middle]

Ενότητα 17: Δυαδική Αναζήτηση

Η δυαδική αναζήτηση είναι πολύ πιο αποδοτική από την γραμμική αναζήτηση, με χρόνο εκτέλεσης O(logn)O(\log n)O(logn), όπου nnn είναι το πλήθος των στοιχείων του πίνακα. Αυτό συμβαίνει επειδή σε κάθε βήμα αποκλείεται το μισό τμήμα των δεδομένων, καθιστώντας την πολύ γρήγορη σε μεγάλους ταξινομημένους πίνακες.

```
ΣΥΝΑΡΤΗΣΗ Δυαδική_Αναζήτηση(πίνακας, αριστερό, δεξιό, στοιχείο) : ΑΚΕΡΑΙΑ
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: μέσο
ΑΡΧΗ
  ΟΣΟ αριστερό <= δεξιό ΕΠΑΝΑΛΑΒΕ
    μέσο <- (αριστερό + δεξιό) DIV 2
    ΑΝ πίνακας[μέσο] = στοιχείο ΤΟΤΕ
      Δυαδική_Αναζήτηση <- μέσο
      ΕΠΙΣΤΡΟΦΗ
    ΑΛΛΙΩΣ_ΑΝ πίνακας[μέσο] < στοιχείο ΤΟΤΕ
      αριστερό <- μέσο + 1
    ΑΛΛΙΩΣ
      δεξιό <- μέσο - 1
    ΤΕΛΟΣ_ΑΝ
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  Δυαδική_Αναζήτηση <- -1  ! Το στοιχείο δεν βρέθηκε
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Ενότητα 17: Δυαδική Αναζήτηση

```
ΠΡΟΓΡΑΜΜΑ Αναζήτηση_Στοιχείου
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: πίνακας[5], θέση, στοιχείο
ΑΡΧΗ
  πίνακας[1] <- 1
  πίνακας[2] <- 3
  πίνακας[3] <- 5
  πίνακας[4] <- 7
  πίνακας[5] <- 9

  ΓΡΑΨΕ "Δώσε το στοιχείο που ψάχνεις:"
  ΔΙΑΒΑΣΕ στοιχείο

  θέση <- Δυαδική_Αναζήτηση(πίνακας, 1, 5, στοιχείο)

  ΑΝ θέση ≠ -1 ΤΟΤΕ
    ΓΡΑΨΕ "Το στοιχείο βρέθηκε στη θέση ", θέση
  ΑΛΛΙΩΣ
    ΓΡΑΨΕ "Το στοιχείο δεν βρέθηκε."
  ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```