242 - Εισαγωγή στους Η/Υ

Τμήμα Μαθηματικών,
Πανεπιστήμιο Ιωαννίνων

Άρτια Α.Μ. (0-2-4-6-8)

Πίνακες στην ΕΑΓ

- Δήλωση:

δηλωση (<ονομα> [1 :Δ1, 1 : Δ2, ..., 1:Δν])
παραταξη **<τύπος μεταβλητής>** ;

- Π.χ.: **δηλωση** (A[1:20]) **παραταξη** **ακερ**;
δηλωση (B[1:2, 1:10]) **παραταξη** **ακερ**;

- Προσπέλαση: <ονομα> [δ1, δ2, ..., δν]
- Π.χ.: A[5] B[2, 10] B[I,J]
- Τι τύποι είναι πάντα οι δείκτες (δ1,δ2, ..., δν);

Πίνακες στη C++

- Δήλωση:

```
<τύπος μεταβλητής> <ονομα> [Δ1][Δ2]...[Δν];
```

- Π.χ.:
```
int A[10], C[6];
float B[5][5], D[2][10];
```
- **Προσοχή**: Η αρίθμηση ξεκινάει πάντα από το 0

  `int A[N]` → `A[0], A[1], ..., A[N-1]`

  `int A[4]` → `A[0], A[1], A[2], A[3]`

- Προσπέλαση:
```
<ονομα> [δ1] [δ2] ... [δν]
```
- Π.χ.: `A[0]` `B[0][10]` `B[I][J]`

Παράδειγμα διάβασμα-εκτύπωση Πίνακα

- Θέλουμε να καταχωρήσουμε 10 ακέραιες τιμές σε έναν πίνακα διαβάζοντας τις τιμές από το πληκτρολόγιο

και

- στη συνέχεια θέλουμε να εκτυπώσουμε τις τιμές του πίνακα

```
διαδικασια ΠΑΡΑΔΕΙΓΜΑ
δηλωση (Ι) ακερ;
δηλωση (ΠΙΝ (1:10) ) παραταξη ακερ;
αρχη
    για Ι ← 1 εως 10 επαναλαβε
    (
        διαβασε ΠΙΝ(Ι);
    )

    για Ι ← 1 εως 10 επαναλαβε
    (
        τυπωσε ΠΙΝ(Ι);
    )

τελος ΠΑΡΑΔΕΙΓΜΑ
```

```
int main()
{
   int I, A[10];

   for (I=0; I<10; I++)
   {
       cin >> A[I];
   }

   for (I=0; I<10; I++)
   {
        cout << A[I];
   }

}
```

```
#include <iostream.h>
main()
{
  int I, A[10];

  for (I=0; I<10; I++)
  {
      cin >> A[I];
  }

  for (I=0; I<10; I++)
  {
       cout << A[I];
  }

}
```

Διάταξη Ακολουθίας

- Θα ασχοληθούμε με αλγορίθμους που διατάσουν μια ακολουθία σε *φθίνουσα* ή *αύξουσα* τάξη (**αλγόριθμοι ταξινόμησης**)
- Θεωρούμε ότι

  X[100] ακέραια στοιχεία

- Θέλουμε να διατάξουμε τα στοιχεία του X έτσι ώστε (αύξουσα τάξη):

  $X[1] \leq X[2] \leq ... \leq X[99] \leq X[100]$

Αλγόριθμοι Ταξινόμησης

- Θα εξετάσουμε διάφορους αλγορίθμους που επιλύουν ίδια προβλήματα.
- Όταν σχεδιάζουμε έναν αλγόριθμο μας ενδιαφέρει η *αποτελεσματικότητά* του (ταχύτητα)
- Ταξινόμηση δεδομένων
  - Σκοπός: $X[1] \leq X[2] \leq ... \leq X[99] \leq X[100]$
  - Πολύ σπουδαία εφαρμογή
  - Σχεδόν κάθε οργανισμός πρέπει να ταξινομεί κάποια δεδομένα
    - Συνήθως οι επιχειρήσεις πρέπει να ταξινομούν ή να κρατάνε ταξινομημένα μεγάλο όγκο δεδομένων
    - Ταξινομούμε ένα πίνακα με n (=100) στοιχεία

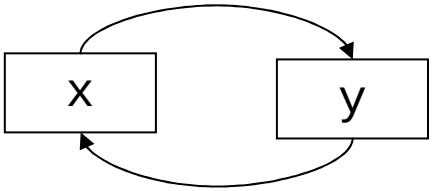
Αλγόριθμοι Ταξινόμησης

- Αλγόριθμος Επιλογής (Selection – Sort)
  - Μέθοδος ανταλλαγής των μεγίστων στοιχείων
  - Απλός και εύκολος στην υλοποίηση
  - Αργός
- Αλγόριθμος Φυσαλίδας (Bubble – Sort)
  - Μέθοδος προώθησης μεγάλων τιμών
  - Απλός και εύκολος στην υλοποίηση
  - Αργός
- Αλγόριθμος Συγχώνευσης (Merge – Sort)
  - Μας δίνονται δύο ήδη ταξινομημένοι πίνακες και ζητάμε την συγχώνευσή τους σε έναν ταξινομημένο πίνακα
  - Γρήγορος αλλά πιο πολύπλοκος στην υλοποίηση

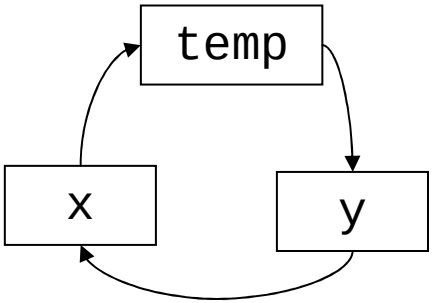
Ανταλλαγή Στοιχείων

- Ανταλλαγή (swap) των τιμών που αποθηκεύονται στις μεταβλητές x και y.
- Πρέπει να χρησιμοποιήσουμε μια νέα *βοηθητική* μεταβλητή.

```
int temp;
```




```
temp = x;
x = y;
y = temp;
```



Αλγόριθμος Επιλογής - Selection Sort (με ανταλλαγή μεγίστων)

- Μέθοδος ανταλλαγής των μεγίστων στοιχείων

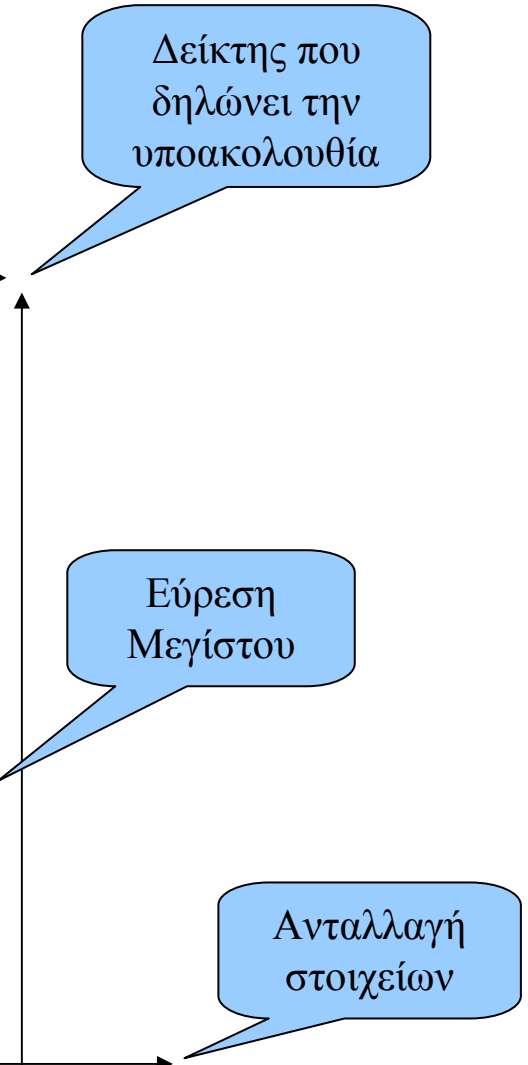
1. Προσδιορισμός στοιχείου που έχει την μέγιστη τιμή από τα N(=100) στοιχεία
2. Ανταλλαγή με το N-στό στοιχείο της (υπό)-ακολουθίας
3. Επανάληψη των βημάτων 1-2 με τα υπόλοιπα N-1 στοιχεία. Δηλαδή, για την
   - υποακολουθία των πρώτων N-1 στοιχείων
   - υποακολουθία των πρώτων N-2 στοιχείων
   - ....
   - υποακολουθία των πρώτων 2 στοιχείων

Αλγόριθμος Επιλογής

- Παράδειγμα

	– Αρχικός πίνακας:	**6**	**5**	**2**	**4**	**3**
Πέρασμα 1	– Εύρεση:	(**6**)	**5**	**2**	**4**	**3**
	– Ανταλλαγή:	**3**	**5**	**2**	**4**	**6**
Πέρασμα 2	– Εύρεση:	**3**	(**5**)	**2**	**4**	**6**
	– Ανταλλαγή:	**3**	**4**	**2**	**5**	**6**
Πέρασμα 3	– Εύρεση:	**3**	(**4**)	**2**	**5**	**6**
	– Ανταλλαγή:	**3**	**2**	**4**	**5**	**6**
Πέρασμα 4	– Εύρεση:	(**3**)	**2**	**4**	**5**	**6**
	– Ανταλλαγή:	**2**	**3**	**4**	**5**	**6**

```
διαδικασια ΤΑΞ_ΕΠΙΛΟΓΗ
δηλωση (Ι, Κ, Μ, Π, ΜΕΓ) ακερ;
δηλωση ( Χ [1:100] ) παραταξη ακερ;
αρχη
    Κ ← 100;
    εφοσον Κ > 1 επαναλαβε
    (
        ΜΕΓ ← Χ[1] ;
        Μ ← 1;
        για Ι ← 2 εως Κ επαναλαβε
        (   εαν Χ[Ι] > ΜΕΓ
                τοτε    (   ΜΕΓ ← Χ[Ι];
                            Μ ← Ι;  )
        )
        Π ←Χ[Μ]; Χ[Μ] ← Χ[Κ]; Χ[Κ]← Π;
        Κ ← Κ -1;
    )
τελος ΤΑΞ_ΕΠΙΛΟΓΗ
```



```
int I, X[100], K, MAX, PMAX, temp;

K = 99;

while( K > 0)
{
    MAX = X[0];
    PMAX = 0;
    for(I=1; I <= K; I++)
    {
        if( X[I] > MAX)
        {
            MAX = X[I];
            PMAX = I;
        }
    }
    temp = X[PMAX]; X[PMAX] = X[K]; X[K] = temp;
    K = K – 1;
}
```

Δείκτης που δηλώνει την υποακολουθία

(θέση του τελευταίου στοιχείου)

Εύρεση Μεγίστου

Ανταλλαγή στοιχείων

Αλγόριθμος Επιλογής - Selection Sort (με ανταλλαγή ελαχίστων)

- Μέθοδος ανταλλαγής των ελαχίστων στοιχείων

1. Προσδιορισμός στοιχείου που έχει την ελάχιστη τιμή από τα N(=100) στοιχεία
2. Ανταλλαγή με το 1-στό στοιχείο της (υπό)-ακολουθίας
3. Επανάληψη των βημάτων 1-2 με τα υπόλοιπα N-1 στοιχεία. Δηλαδή, για την

   - υποακολουθία των τελευταίων N-1 στοιχείων
   - υποακολουθία των τελευταίων N-2 στοιχείων
   - ....
   - υποακολουθία των τελευταίων 2 στοιχείων

Αλγόριθμος Φυσαλίδας (Bubble – Sort)

- Προωθεί τις μεγάλες τιμές προς τα δεξιά, συγκρίνοντας κάθε στοιχείο με το επόμενό του
- Δηλαδή ελέγχει το X[I] με το X[I+1]
- Εάν X[I] > X[I+1] τότε ανταλλάσουμε τα δύο στοιχεία.
- Εάν η διαδικασία επαναληφθεί για τα N-1 στοιχεία τότε το μέγιστο πηγαίνει στην θέση N.
- Συνεχίζοντας για τα υπόλοιπα N-1 στοιχεία (το μέγιστο έχει τοποθετηθεί σωστά) επιτυγχάνουμε την ταξινόμηση.

Αλγόριθμος Φυσαλίδας (Bubble – Sort)

- Παράδειγμα

Αρχικός πίνακας: **3 4 2 6 7**

Πέρασμα 1: **3 4 2 6 7**
Πέρασμα 1: **3 2 4 6 7**
Πέρασμα 1: **3 2 4 6 7**
Πέρασμα 1: **3 2 4 6 7**

Πέρασμα 2: **2 3 4 6 7**
Πέρασμα 2: **2 3 4 6 7**
Πέρασμα 2: **2 3 4 6 7**

Πέρασμα 3: **2 3 4 6 7**
Πέρασμα 3: **2 3 4 6 7**

```
διαδικασια ΤΑΞ_ΦΥΣΑΛ
δηλωση (Ι, Κ, Π, ΜΕΓ) ακερ;
δηλωση ( Χ [1:100] ) παραταξη ακερ;
αρχη
    ΔΙΑΒΑΣΜΑ
    Κ ← 100;
    εφοσον Κ > 1 επαναλαβε
    (
        για Ι ← 1 εως Κ επαναλαβε
        (   εαν Χ[Ι] > Χ[Ι+1]
                τοτε    (   Π ←Χ[Ι];
                            Χ[Ι] ← Χ[Ι+1];
                            Χ[Ι+1]← Π;    )
        )
        Κ ← Κ -1;
    )
    ΕΚΤΥΠΩΣΗ
τελος ΤΑΞ_ΦΥΣΑΛ
```

```
int I, X[100], K, temp;

K = 99;

while( K > 0 )
{
    for(I=0; I <= K-1; I++)
    {
        if( X[I] > X[I+1])
        {
          temp = X[I];
          X[I] = X[I+1];
          X[I+1] = temp;
        }
    }
    K = K – 1;
}
```

Αλγόριθμος Εισαγωγής – Insertion sort

- Μέθοδος εισαγωγής στοιχείων στην σωστή θέση

1. Αρχίζουμε από το στοιχείο στην δεύτερη θέση (΄το Χ[1] δηλαδή) και το συγκρίνουμε με τα προηγούμενα στοιχεία, ανταλλάσοντας αν χρειαστεί τα στοιχεία. Αυτό συνεχίζεται μέχρι το Χ[1] να είναι στην σωστή θέση δηλαδή το αμέσως προηγούμενο στον πίνακα είναι μικρότερο του.
2. Επαναλαμβάνουμε την διαδικασίας με τα υπόλοιπα στοιχεία Χ[2]…Χ[Ν-1]. Σε κάθε επανάληψη ένα στοιχείο μπαίνει στην σωστή θέση όσον αφορά την ταξινόμηση.

Αλγόριθμος Επιλογής

- Παράδειγμα

	– Αρχικός πίνακας:	**5**	**1**	**2**	**4**	**3**
Πέρασμα 1	– Στοιχείο:	**5**	**1**	**2**	**4**	**3**
	– Προώθηση:	**1**	**5**	**2**	**4**	**3**
Πέρασμα 2	– Στοιχείο:	**1**	**5**	**2**	**4**	**3**
	– Προώθηση:	**1**	**2**	**5**	**4**	**3**
Πέρασμα 3	– Στοιχείο:	**1**	**2**	**5**	**4**	**3**
	– Προώθηση:	**1**	**2**	**4**	**5**	**3**
Πέρασμα 4	– Στοιχείο:	**1**	**2**	**4**	**5**	**3**
	– Προώθηση :	**1**	**2**	**3**	**4**	**5**

```
int I, X[100], I, J, N, temp;

N = 99;

for( I=1; I<=N-1; I++)
{   J=I;
    while (J>0 && X[J]<X[J-1])
    {
      temp = X[J];
      X[J] = X[J-1];
      X[J-1] = temp;
      j--;
    }
}
```

Δείκτης που δηλώνει την θέση του τελευταίου στοιχείου

X[0]…X[99]

Προώθηση στοιχείων προς τα επάνω στην σωστή θέση

Ανταλλαγή στοιχείων

Αλγόριθμος Εισαγωγής – Insertion sort C++

```
/* insertion sort ascending order */
#include <iostream>
#include <cstdlib>
using namespace std;
int main(int argc, char *argv[])
{ int n, X[100], i, j, temp;
  cout<<"Dose plithos stoixeiwn: ";
  cin>>n;
  cout<<"Dose "<<n<<" stoixeia. \n";
  for (i = 0; i < n; i++) cin>>X[i];
```

Αλγόριθμος Εισαγωγής – Insertion sort C++

```
…… …  συνέχεια

for (i = 1 ; i <= n - 1; i++) {    //ταξινόμηση
    j = i;
    while ( j > 0 && X[j] < X[j-1]) {
      temp = X[j];
      X[j] = X[j-1];
      X[j-1] = temp;
      j--;
    }
  }
  cout<<“O taksinomhmenos pinakas: \n";
  for (i = 0; i <= n - 1; i++) cout<<X[i]<<" ";
  cout<<endl;
  system("Pause");
  return 0;
}
```

Αλγόριθμος Συγχώνευσης (Merge – Sort)

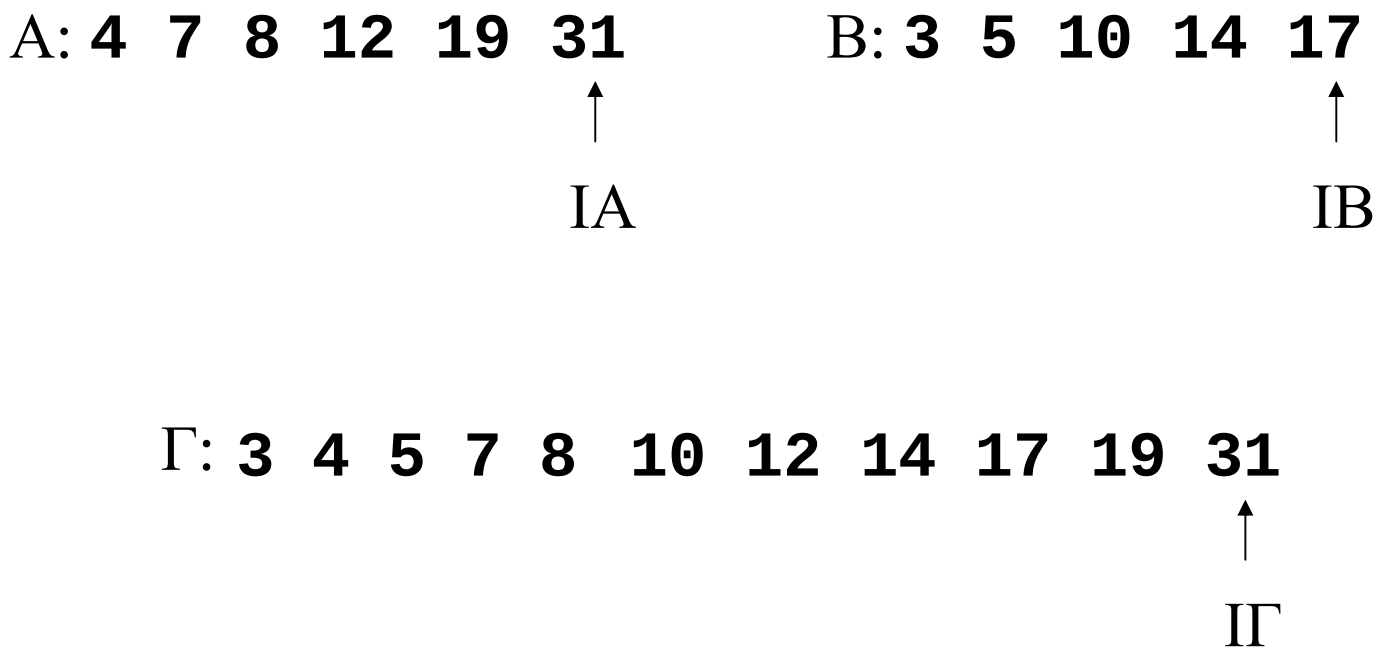
- Μας δίνονται δύο ταξινομημένοι πίνακες

  A[N] και B[M]

  και μας ζητάνε να συγχωνεύσουμε τους πίνακες A,B σε έναν ταξινομημένο πίνακα Γ[N+M].
- Ο πρώτος του Α συγκρίνεται με τον πρώτο του Β
  - Ο μικρότερος αποτελεί τον πρώτο του Γ.
  - Ο αριθμός που δεν καταχωρήθηκε συγκρίνεται με τον επόμενο που καταχωρήθηκε
  - Η διαδικασία συνεχίζεται μέχρι τα στοιχεία του Α ή Β να εξαντληθούν.
  - Τότε τα στοιχεία του πίνακα που δεν εξαντλήθηκε, προσκολλώνται στον πίνακα Γ.

Αλγόριθμος Συγχώνευσης (Merge – Sort)

- Παράδειγμα:



```
διαδικασια ΤΑΞ_ΣΥΓΧ
δηλωση (ΙΑ, ΙΒ, ΙΓ) ακερ;
δηλωση ( Α[1:50], Β[1:50], Γ[1:100] ) παραταξη ακερ;
αρχη
    ΔΙΑΒΑΣΜΑ ταξινομ. Α & Β
    ΝΑ ←50; ΝΒ ← 50; ΝΓ ←100;
    ΙΑ ←1;  ΙΒ ← 1; ΙΓ←1;
    εφοσον ΙΑ ≤ ΝΑ & ΙΒ ≤ ΝΒ επαναλαβε
    (       εαν Α[ΙΑ] < Β[ΙΒ]
                    τοτε    ( Γ[ΙΓ] ← Α[ΙΑ];
                              ΙΓ ← ΙΓ+1;
                              ΙΑ ← ΙΑ+1; )
                    αλλως   ( Γ[ΙΓ] ← Β[ΙΒ];
                              ΙΓ ← ΙΓ+1;
                              ΙΒ ← ΙΒ+1; )
    )
    ΕΠΙΚΟΛΗΣΗ ΣΤΟΙΧΕΙΩΝ ΤΟΥ ΠΙΝΑΚΑ
τελος ΤΑΞ_ΦΥΣΑΛ
```

```
εαν ΙΑ = ΝΑ
   τοτε (
     για Ι ← ΙΒ εως ΝΒ επαναλαβε
              (   Γ[ΙΓ] ← Β[Ι];
                  ΙΓ ← ΙΓ+1; ) )
   αλλως (
     για Ι ← ΙΑ εως ΝΑ επαναλαβε
              (   Γ[ΙΓ] ← Α[Ι];
                  ΙΓ ← ΙΓ+1; ) )
```

```
int A[50], B[50], C[100];
int IA, IB, IC, NA, NB, NC, I;
NA = 49; NB = 49; NC=99;
IA = 0; IB = 0; IC = 0;
while( (IA <= NA) && (IB <= NB) )
{
    if( A[IA] < B[IB] )
    {
          C[IC] = A[IA];
          IC = IC + 1;
          IA = IA + 1;
    }
    else
    {
          C[IC] = B[IB];
          IC = IC + 1;
          IB = IB + 1;
    }
}
```

```
// ΕΠΙΚΟΛΗΣΗ ΣΤΟΙΧΕΙΩΝ
if( IA == NA )
{
   for(I=IB; I<=NB; I++)
   {
        C[IC] = B[I];
        IC = IC + 1;
   }
else
{
   for(I=IA; I<=NA; I++)
   {
        C[IC] = A[I];
        IC = IC + 1;
   }
}
```

Διερεύνηση Πινάκων (Αναζήτηση στοιχείου)

- Αναζήτηση κάποιας τιμής (κλειδί) σε ένα μονοδιάστατο πίνακα.
- Υπάρχουν δύο βασικές μέθοδοι.
  - Διαδοχική Διερεύνηση (γραμμική αναζήτηση)
  - Δυαδική Αναζήτηση
- Θα πρέπει να εξετάζουμε και την περίπτωση που το κλειδί δεν υπάρχει στον πίνακα.
  - Δηλαδή να εκτυπώνεται ανάλογο μήνυμα.
  - Τέτοιου είδους τεχνικές χρησιμοποιούν μεταβλητές που λέγονται *σημαίες* ή *λογικές μεταβλητές*.

Διαδοχική Διερεύνηση

- Συγκρίνουμε κάθε στοιχείο με την τιμή κλειδί που αναζητάμε
- Απλή αλλά αργή μέθοδος (μπορεί να εξετάσουμε όλα τα στοιχεία του πίνακα).

	[1]	[2]	[3]	[4]	[5]	[6]
A:	**44**	**17**	**8**	**12**	**19**	**3**

X = **11**

```
διαδικασια ΓΡΑΜΜ_ΑΝΑΖΗΤ
δηλωση (Ι, Κ, Μ, ΚΛ) ακερ;
δηλωση ( Χ [1:100] ) παραταξη ακερ;
αρχη
    ΔΙΑΒΑΣΜΑ ΠΙΝΑΚΑ Χ & ΚΛΕΙΔΙ ΚΛ
    Κ ← 100;
    ΣΗΜ ← 1;
     εφοσον Ι ≤ Κ & ΣΗΜ ≠ 0 επαναλαβε
    (   εαν Χ[Ι] = ΚΛ
                    τοτε      (   ΣΗΜ ← 0;
                                  Μ ← Ι;   )
                    αλλως     (   Ι ← Ι +1; )
    )
    εαν  ΣΗΜ = 0
          τοτε      (    τυπωσε 'Βρέθηκε ', Μ ; )
          αλλως     (    τυπωσε 'Δεν βρέθηκε' ; )
τελος ΓΡΑΜΜ_ΑΝΑΖΗΤ
```

```
για Ι ← 1 εως 100 επαναλαβε
(    διαβασε Χ[Ι] ;  )
διαβασε ΚΛ;
```

```
int X[100], I, key, flag, N, POS;
N = 99;

flag = 0;
I = 0;
while( (I <= N) && (flag != 1) )
{
    if( X[I] == key )
    {
          flag = 1;
          POS = I;
    }
    else
          I = I + 1;
}
if(flag == 1)
    cout << “Vrethike sthn the thesi” << POS;
else
     cout << “Den vrethike”;
```

```
// διάβασμα πίνακα
// και στοιχείου προς
// αναζήτηση
cin >> key;
for(I=0; I<=N; I++)
  cin >> X[I];
```

Δυαδική Αναζήτηση

- Εάν ο πίνακας είναι ταξινομημένος τότε μπορούμε να αναζητήσουμε πιο γρήγορα το κλειδί.
  - Συγκρίνουμε το μεσαίο στοιχείο (middle) του πίνακα με το κλειδί (key) που αναζητάμε
    - Εάν Κλειδί = Μεσαίο, τότε βρέθηκε
    - Εάν Κλειδί < Μεσαίο, τότε η αναζήτηση γίνεται στο αριστερό ήμισυ του πίνακα
    - Εάν Κλειδί > Μεσαίο, τότε η αναζήτηση γίνεται στο δεξιό ήμισυ του πίνακα
    - Επαναλαμβάνουμε την δυαδική αναζήτηση στο ήμισυ που επιλέξαμε

Δυαδική Αναζήτηση

- Παράδειγμα
  - Όρια στον πίνακα: Κ: κάτω όριο, Π: πάνω όριο

Χ = **11**

Α:	**3**	**5**	**8**	**12**	**19**	**22**	**44**
		↑	↑	↑			
		Κ	Μεσο	Π			

```
διαδικασια ΔΥΑΔ_ΑΝΑΖΗΤ
δηλωση (Κ, Π, ΚΛ, ΜΕΣ) ακερ;
δηλωση ( Χ [1:100] ) παραταξη ακερ;
αρχη
    Κ ← 1; Π ←100;
    ΣΗΜ ← 1;
    εφοσον Κ < Π & ΣΗΜ ≠ 0 επαναλαβε
    (       ΜΕΣ ← (Κ+Π)/2;
            εαν ΚΛ = Χ[ΜΕΣ]
                τοτε    (   ΣΗΜ ← 0; )
                αλλως   (   εαν ΚΛ > Χ[ΜΕΣ]
                                τοτε   ( Κ ← ΜΕΣ; )
                                αλλως ( Π ← ΜΕΣ; )
                        )
    )
    ΕΚΤΥΠΩΣΗ ΜΕ ΒΑΣΗ ΤΗ ΣΗΜΑΙ
τελος ΔΥΑΔ_ΑΝΑΖΗΤ
```

Εαν τελειώσει η αναζήτηση στον πίνακα

Εαν βρεθεί το στοιχείο

```
εαν  ΣΗΜ = 0
    τοτε    ( τυπωσε 'Βρέθηκε ', ΜΕΣ;)
    αλλως ( τυπωσε 'Δεν βρέθηκε' ; )
```

```
int X[100], K, P, key, flag, MID;

flag = 0;
K = 0;
P = 99;
while( (K < P) && (flag != 1) )
{
    MID = ( K + P )/2;
    if( X[MID] == key )
       flag = 1;
    else
    {
       if( X[MID] < key )
          K = MID+1;
       else
          P = MID-1;
    }
}
```

```
if(flag == 1)
   cout << “Vrethike”;
else
    cout << “Den vrethike”;
```