Βασικές Αρχές Προγραμματισμού

Κεφάλαιο 14

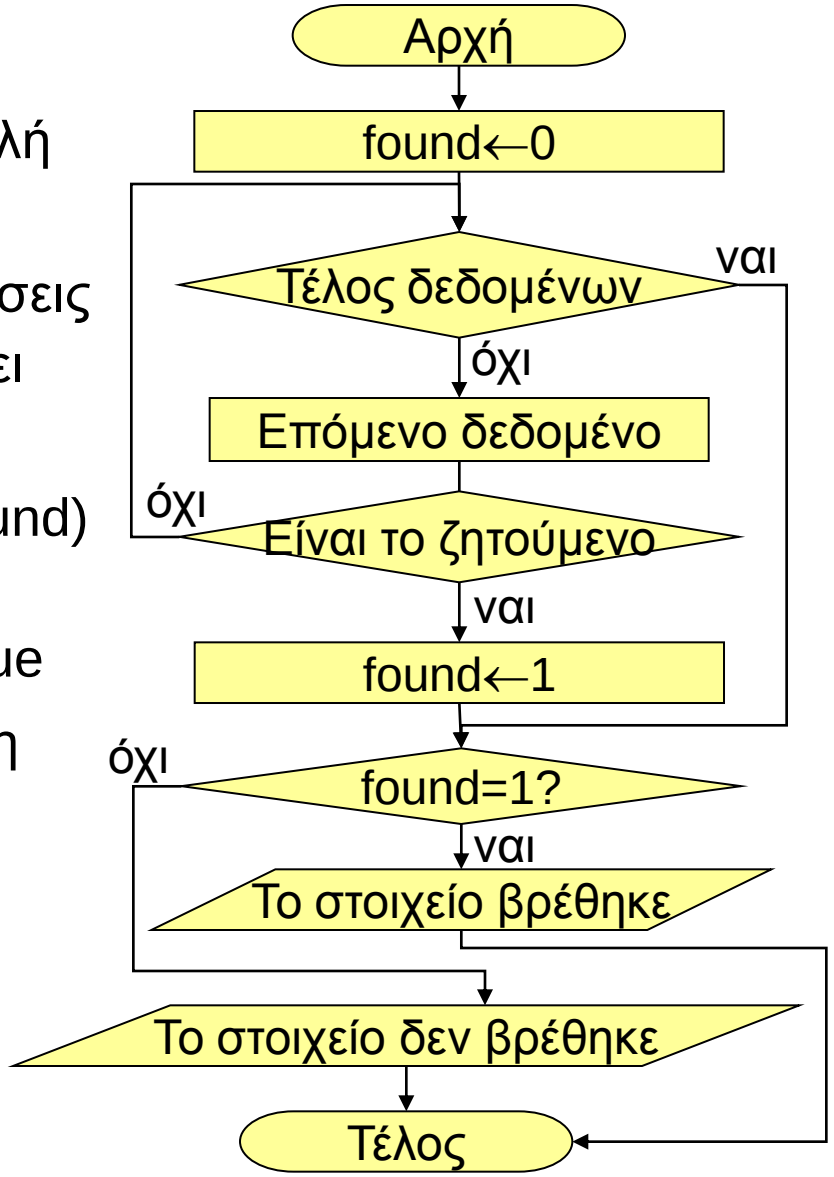
Αναζήτηση & Ταξινόμηση

Αναζήτηση & Ταξινόμηση

- Δύο από τις σημαντικότερες λειτουργίες στα στοιχεία ενός πίνακα είναι η αναζήτηση και η ταξινόμηση
- Αναζήτηση είναι ο έλεγχος για το αν υπάρχει μια συγκεκριμένη τιμή μέσα στον πίνακα
- Ταξινόμηση είναι η διάταξη των στοιχείων του πίνακα κατά αύξουσα ή φθίνουσα σειρά
- Οι αλγόριθμοι αναζήτησης και ταξινόμησης είναι γενικοί και εφαρμόζονται πέρα από τους απλούς πίνακες σε άλλες δομές δεδομένων αλλά και σε δεδομένα που βρίσκονται σε δευτερεύοντα μέσα αποθήκευσης (αρχεία)
- Ο βασικός στόχος κάθε αλγόριθμου είναι η μείωση της πολυπλοκότητας του, δηλαδή η αύξηση της ταχύτητας εκτέλεσης του

Σειριακή Αναζήτηση

- Η σειριακή αναζήτηση είναι η πιο απλή μέθοδος αναζήτησης τιμών
- Ο αλγόριθμος σαρώνει μία-μία τις θέσεις μέχρι να εντοπιστεί η τιμή ή να φτάσει στο τέλος της δομής δεδομένων
- Χρησιμοποιεί μία μεταβλητή (π.χ. found) με αρχική τιμή false και αν βρεθεί το στοιχείο τότε αλλάζει η τιμή της σε true
- Στο τέλος του αλγορίθμου ελέγχεται η τιμή της μεταβλητής
  - Αν true τότε το στοιχείο υπάρχει
  - Αν false το στοιχείο δεν υπάρχει



Κώδικας Σειριακής Αναζήτησης

- Το επόμενο τμήμα κώδικα ζητά έναν αριθμό από το πληκτρολόγιο και τον αναζητά σε έναν πίνακα 100 ακεραίων

```
main()
{
   int i,k,A[100],found=0;
   for(i=0; i<100; i++)
        A[i]=rand();
   printf("Δώσε έναν ακέραιο: ");
   scanf("%d",&k);
   for(i=0;i<100;i++)
   {
        if (A[i]==k)
        {
            found=1;
        }
   }
   if (found==1)
        printf("Ο αριθμός υπάρχει στον πίνακα\n");
   else
        printf("Ο αριθμός δεν υπάρχει στον πίνακα\n");
   system("pause");
}
```

Κώδικας Σειριακής Αναζήτησης 2

```
main()
{
    int i,k,A[100];
    for(i=0; i<100; i++)
        A[i]=rand();
    printf("Δώσε έναν ακέραιο: ");
    scanf("%d",&k);
    i=0;
    while (A[i]!=k && i<100)
    {
        i++;
    }
    if (i<100)
        printf("Ο αριθμός υπάρχει στον πίνακα\n");
    else
        printf("Ο αριθμός δεν υπάρχει στον πίνακα\n");
    system("pause");
}
```

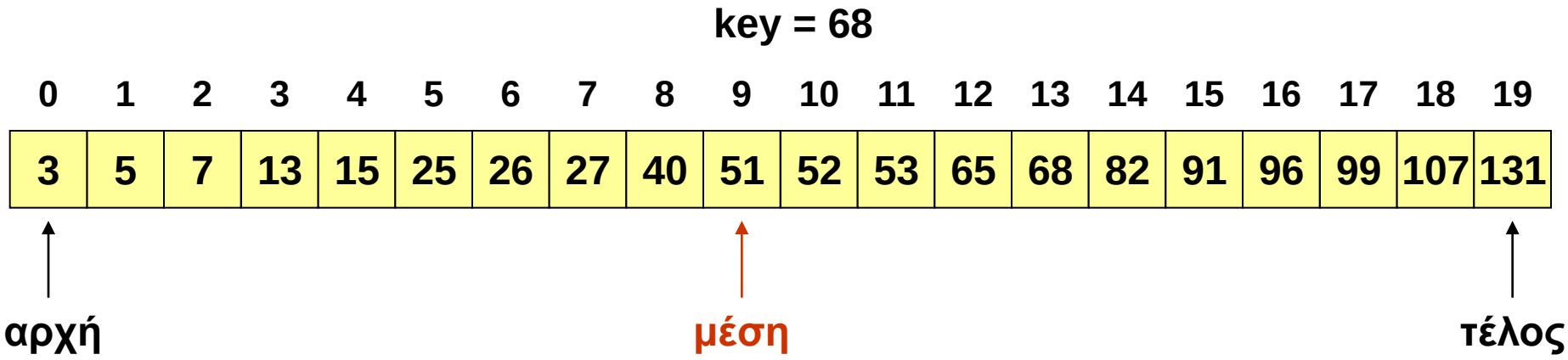
Συνάρτηση Σειριακής Αναζήτησης

```
int seqSearch(int arr[], int n, int key)
{
   int i;
   for(i=0;i<n;i++)
   {
        if (arr[i]==key)
            return 1;
   }
   return 0;
}
main()
{
   int i,k,A[100],found;
   for(i=0; i<100; i++)
        A[i]=rand();
   printf("Δώσε έναν ακέραιο: ");
   scanf("%d",&k);
   found = seqSearch(A,100,k);
   if (found==1)
        printf("Ο αριθμός υπάρχει στον πίνακα\n");
   else
        printf("Ο αριθμός δεν υπάρχει στον πίνακα\n");
   system("pause");
}
```

Δυαδική Αναζήτηση

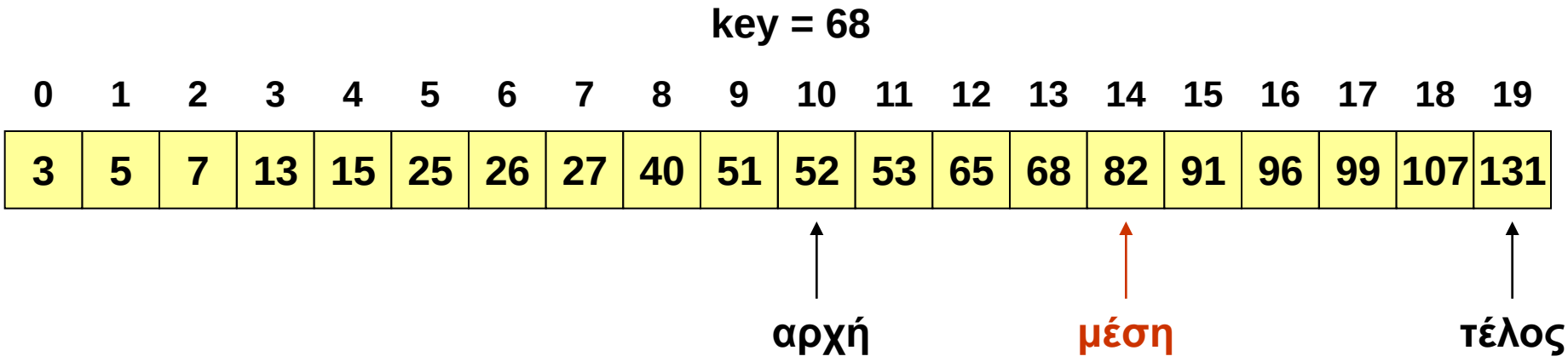
- Η δυαδική αναζήτηση (binary search) είναι ο πιο αποδοτικός τρόπος αναζήτησης σε δομές δεδομένων
  - Η πολυπλοκότητα του αλγορίθμου είναι $O(\log_2 N)$ σε αντίθεση με το $O(N)$ της σειριακής αναζήτησης
  - Το μειονέκτημα του αλγορίθμου είναι ότι εφαρμόζεται μόνο σε ταξινομημένα δεδομένα (κατά αύξουσα ή φθίνουσα σειρά)
- Η λογική του αλγορίθμου είναι ότι ελέγχουμε το μεσαίο στοιχείο των δεδομένων
  - Αν ταυτίζεται με το αναζητούμενο δεδομένο τότε ο αλγόριθμος τερματίζει
  - Αν είναι μεγαλύτερο τότε εφαρμόζουμε τον αλγόριθμο στο πρώτο μισό τμήμα των δεδομένων
  - Αν είναι μικρότερο τότε εφαρμόζουμε τον αλγόριθμο στο δεύτερο μισό τμήμα των δεδομένων

Παράδειγμα 1



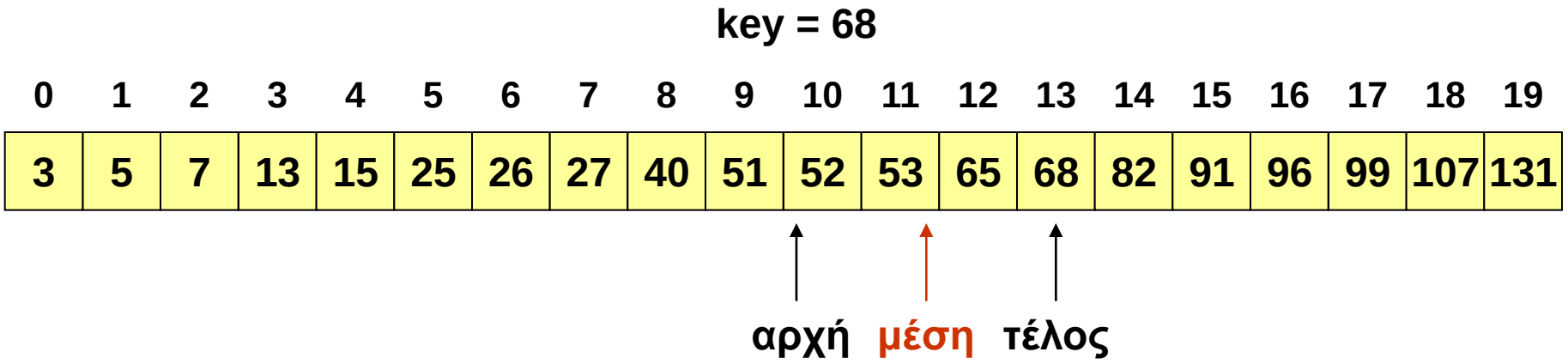
51<68 Άρα ψάχνουμε στο 2ο μισό

Παράδειγμα 1



82>68 Άρα ψάχνουμε στο 1ο μισό

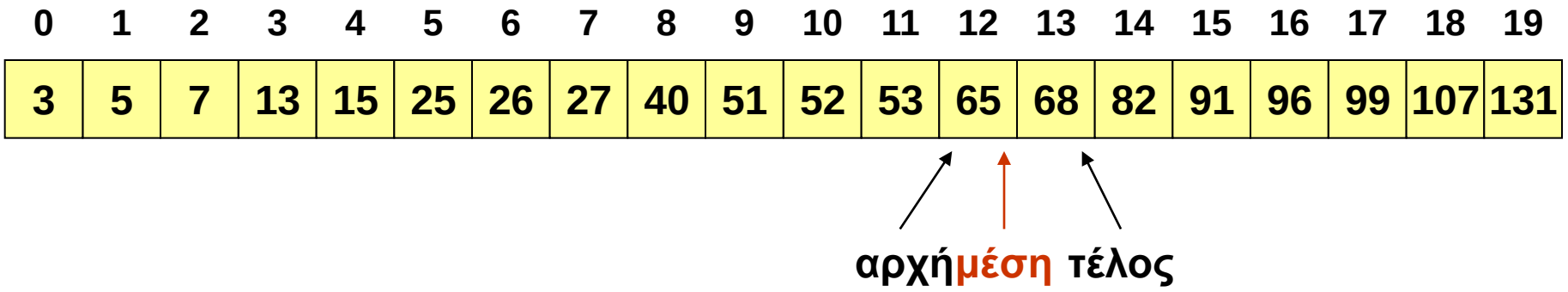
Παράδειγμα 1



53<68 Άρα ψάχνουμε στο 2ο μισό

Παράδειγμα 1

key = 68

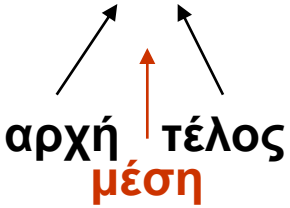


65<68 Άρα ψάχνουμε στο 2ο μισό

Παράδειγμα 1

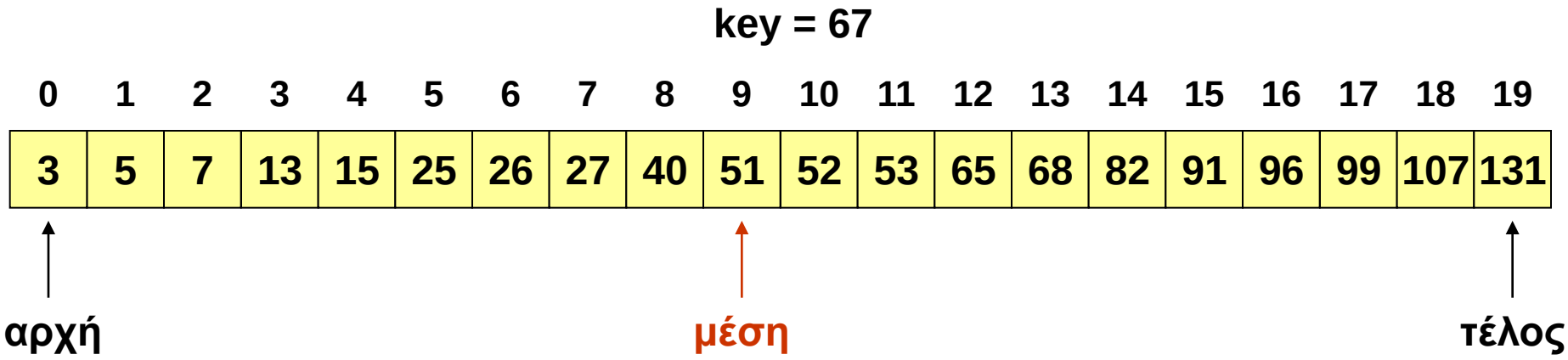
key = 68

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
3	5	7	13	15	25	26	27	40	51	52	53	65	68	82	91	96	99	107	131



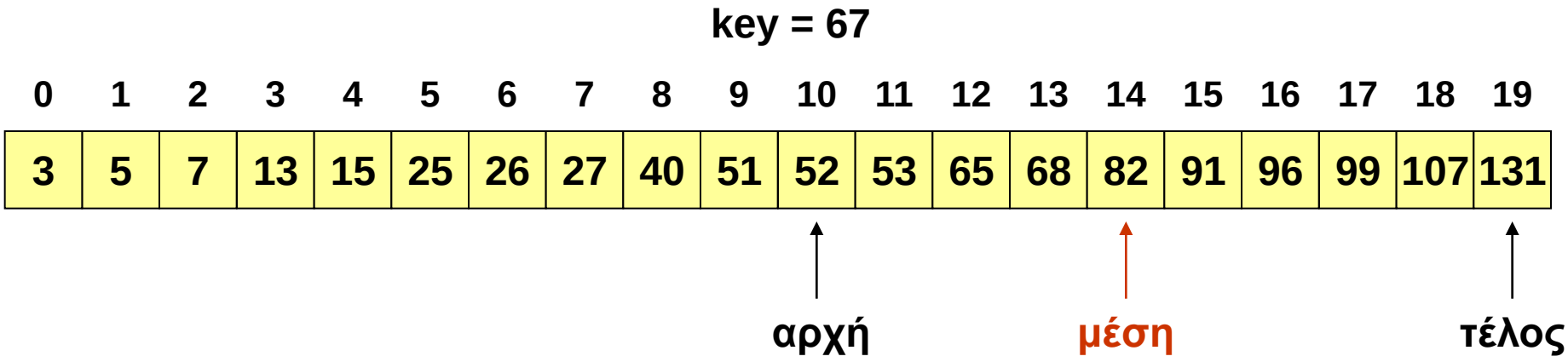
68=68 Άρα ο αλγόριθμος τερματίζει μετά από 5 συγκρίσεις

Παράδειγμα 2



51<67 Άρα ψάχνουμε στο 2ο μισό

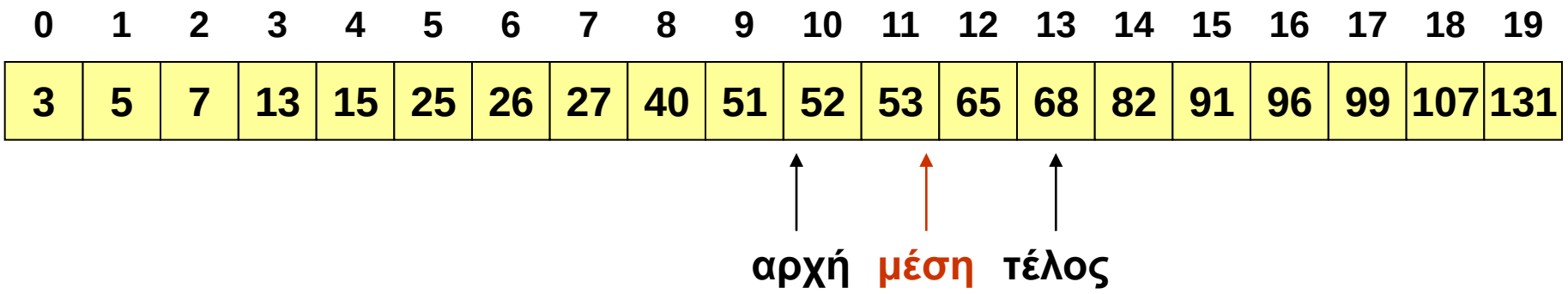
Παράδειγμα 2



82>67 Άρα ψάχνουμε στο 1ο μισό

Παράδειγμα 2

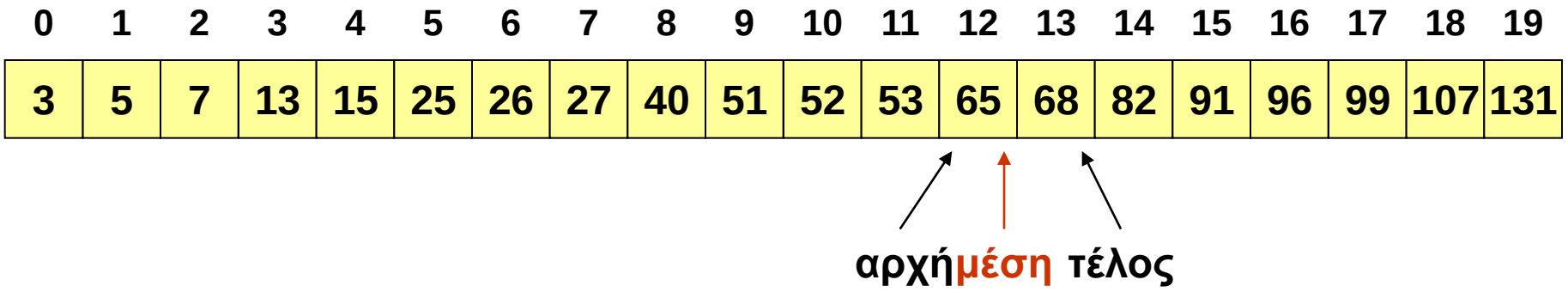
key = 67



53<67 Άρα ψάχνουμε στο 2ο μισό

Παράδειγμα Δυαδικής Αναζήτησης

key = 67

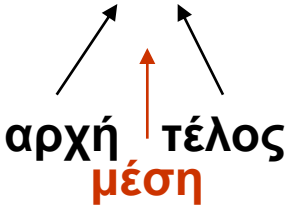


65<67 Άρα ψάχνουμε στο 2ο μισό

Παράδειγμα 2

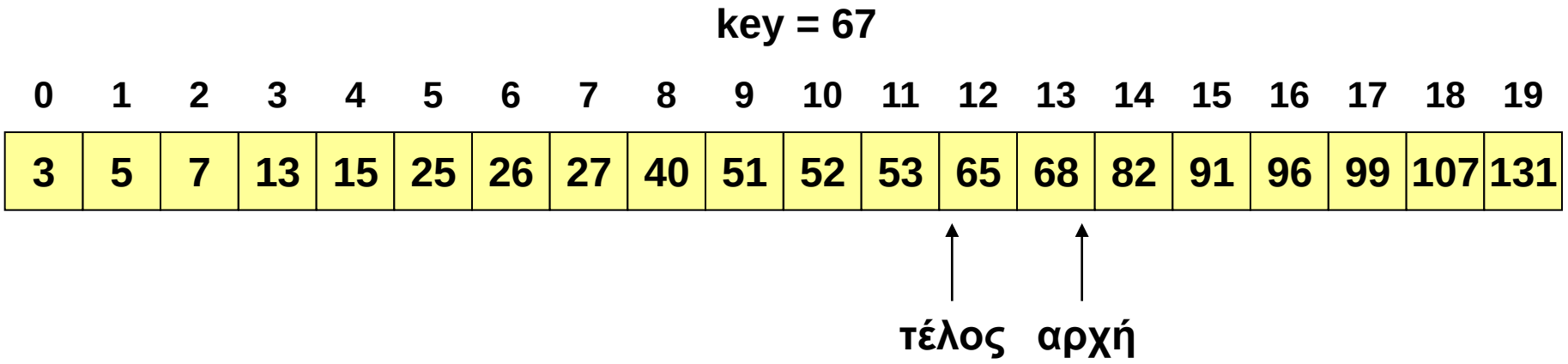
key = 67

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
3	5	7	13	15	25	26	27	40	51	52	53	65	68	82	91	96	99	107	131



68>67 Άρα ψάχνουμε στο 1ο μισό

Παράδειγμα 2



αρχή>τέλος Άρα ο αλγόριθμος τερματίζει αρνητικά

Κώδικας Δυαδικής Αναζήτησης

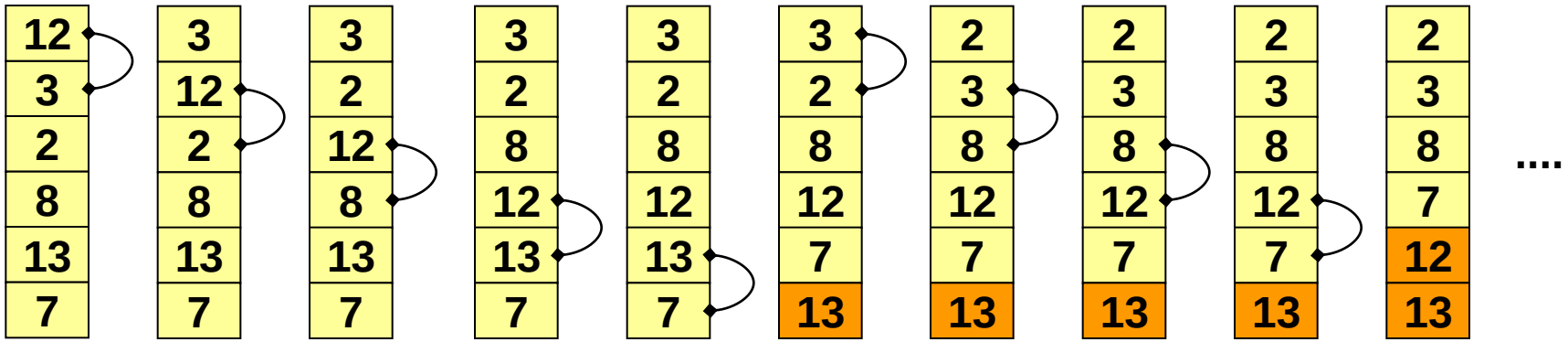
```
int binSearch(int arr[], int n, int key)
{
   int arxi,telos,meso;
   arxi=0;
   telos=n-1;
   while (arxi<=telos)
   {
        meso = (arxi+telos)/2;
        if (key<arr[meso])
            telos=meso-1;
        else
            if (key>arr[meso])
                arxi = meso+1;
            else
                return meso;
   }
   return -1;
}
main()
{
   int i,k,A[100];
   …
   if (binSearch(A,100,k)>-1)
        printf("Ο αριθμός υπάρχει στον πίνακα\n");
   …
}
```

Αναδρομικός Κώδικας Δυαδικής Αναζήτησης

```
int binSearch(int arr[], int arxi, int telos, int key)
{
   int meso;
   if (arxi<=telos)
   {
        meso = (arxi+telos)/2;
        if (key<arr[meso])
            return binSearch(arr,arxi,meso-1,key);
        else
            if (key>arr[meso])
                return binSearch(arr,meso+1,telos,key);
            else
                return meso;
   }
   else
        return -1;
}
main()
{
   int i,k,A[100];
   …
   if (binSearch(A,0,99,k)>-1)
        printf("Ο αριθμός υπάρχει στον πίνακα\n");
   …
}
```

Ταξινόμηση Φυσαλίδας (bubble sort)

- Είναι ένας απλός αλλά όχι πολύ αποδοτικός αλγόριθμος ταξινόμησης
- Η λογική του αλγορίθμου είναι η εξής
  - Κάθε στοιχείο του πίνακα συγκρίνεται με το επόμενο του και αν είναι σε λάθος σειρά αντιμετατίθενται
  - Μόλις ο αλγόριθμος ελέγξει και το τελευταίο ζευγάρι, τότε σίγουρα στο τέλος θα έχει τοποθετηθεί ο μεγαλύτερος από όλους τους αριθμούς
  - Στη συνέχεια ο αλγόριθμος αγνοεί το τελευταίο στοιχείο και επαναλαμβάνεται για τα υπόλοιπα (Το δεύτερο μεγαλύτερο στοιχείο θα τοποθετηθεί στην προτελευταία θέση)
  - Μετά από N-1 βήματα τα στοιχεία θα είναι ταξινομημένα

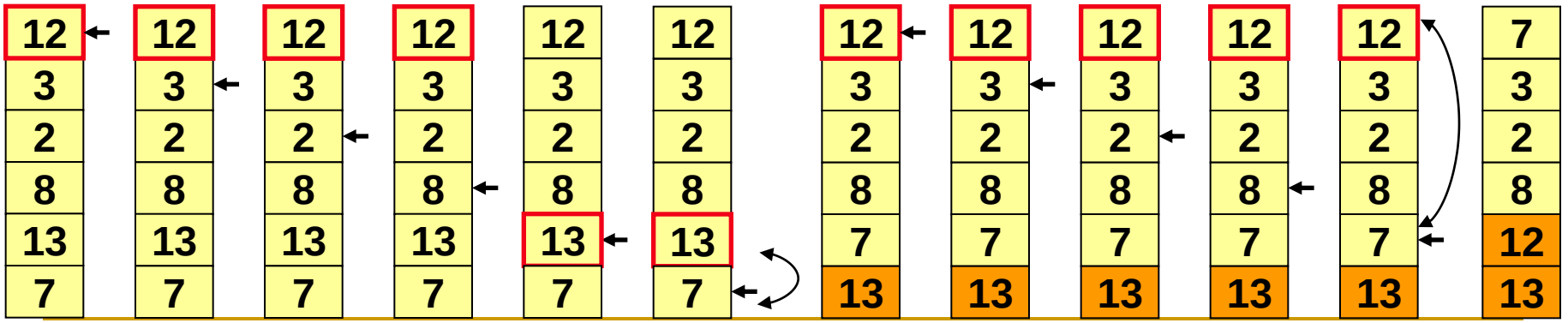


Κώδικας Ταξινόμησης Φυσαλίδας

```
void bubbleSort(int arr[], int n)
{
   int i,j,temp;
   for(i=0;i<n-1;i++)
   {
        for (j=0;j<n-i-1;j++)
        {
             if (arr[j]>arr[j+1])
             {
                  temp=arr[j];
                  arr[j]=arr[j+1];
                  arr[j+1]=temp;
             }
        }
   }
}
main()
{
   int i,k,A[100],found;
   for(i=0; i<100; i++)
        A[i]=rand();
   bubbleSort(A,100);
   …
   system("pause");
}
```

Ταξινόμηση επιλογής (selection sort)

- Ο bubble sort σε κάθε βήμα προσπαθεί να μεταφέρει το τοπικό μέγιστο στην τελευταία θέση αντιμεταθέτοντας συνεχώς στοιχεία
- Ο αλγόριθμος επιλογής λειτουργεί με παρόμοιο τρόπο προσπαθώντας να ελαχιστοποιήσει τις ανταλλαγές στοιχείων
  - Στο πρώτο πέρασμα ο αλγόριθμος εντοπίζει το μέγιστο όλων των στοιχείων και το αντιμεταθέτει με το τελευταίο στοιχείο
  - Στη συνέχεια αγνοεί το τελευταίο στοιχείο και επαναλαμβάνεται για τα υπόλοιπα
  - Μετά από N-1 βήματα τα στοιχεία θα είναι ταξινομημένα



Κώδικας Ταξινόμησης Επιλογής

```
void selectionSort(int arr[], int n)
{
   int i,j,temp,maxpos;
   for(i=n-1;i>0;i--)
   {
        maxpos=0;
        for (j=1;j<=i;j++)
        {
             if (arr[j]>arr[maxpos])
             {
                  maxpos=j;
             }
        }
        temp=arr[i];
        arr[i]=arr[maxpos];
        arr[maxpos]=temp;
   }
}
main()
{
   int i,k,A[100],found;
   …
   selectionSort(A,100);
   …
   system("pause");
}
```

Παράδειγμα: Ταξινόμηση πίνακα συμβολοσειρών

- Έστω ένας πίνακας N συμβολοσειρών (40 χαρακτήρων), δηλαδή ένας δισδιάστατος πίνακας [N][40] χαρακτήρων, ο οποίος πρέπει να ταξινομηθεί λεξικογραφικά

```
void stringBubbleSort(char arr[][40], int n)
{
   int i,j;
   char temp[40];
   for(i=0;i<n-1;i++)
   {
     for (j=0;j<n-i-1;j++)
     {
          if (strcmp(arr[j],arr[j+1])==1)
          {
              strcpy(temp,arr[j]);
              strcpy(arr[j],arr[j+1]);
              strcpy(arr[j+1],temp);
          }
     }
   }
}
```