Εύρεση μεγίστου (ελαχίστου) στοιχείου μονοδιάστατου πίνακα

Παράδειγμα 7.1

Να γράψετε πρόγραμμα που θα διαβάζει 300 πραγματικούς αριθμούς και θα τους τοποθετεί σε ένα μονοδιάστατο πίνακα 300 θέσεων. Στη συνέχεια θα τυπώνει το μέγιστο στοιχείο του πίνακα με ένα σχετικό μήνυμα.

```
#include<iostream>
using namespace std;
int main()
{
    float b[300], maxb;
    int i;
    cout<<"\nAssign values to the array b: ";
    for (i=0; i<300; i++)
    {
        cout<<"\nAssign value to the  "<<i+1
                <<" element of the array ";
        cin>>b[i];
    }
```

```
    maxb=b[0];
    for (i=1; i<300; i++)
    {
        if (maxb<b[i])
            maxb=b[i];
    }
    cout<<"\nThe maximum element of the array is: "
                                <<maxb;
    return 0;
}
```

Παράδειγμα 7.2

Να γράψετε πρόγραμμα που θα διαβάζει 300 πραγματικούς αριθμούς και θα τους τοποθετεί σε ένα μονοδιάστατο πίνακα 300 θέσεων. Στη συνέχεια θα τυπώνει το μέγιστο και ελάχιστο στοιχείο του πίνακα με ένα σχετικό μήνυμα.

```
#include<iostream>
using namespace std;
int main()
{
    float b[300], maxb, minb;
    int i;
    cout<<"\nAssign values to the array b: ";
    for (i=0; i<300; i++)
    {
        cout<<"\nAssign value to the "<<i+1
                                <<" element of b: ";
        cin>>b[i];
}
```

```
    maxb=b[0];
    minb=b[0];
    for (i=1; i<300; i++)
    {
        if (maxb<b[i])
            maxb=b[i];
        if (minb>b[i])
            minb=b[i];
    }
    cout<<"\nThe maximum and the minimum elements
                of  the array are respectively,"<<'\n'
                <<maxb<<'\n'<<" and "<<'\n'<<minb;
    return 0;
}
```

Παράδειγμα 7.3

Να γράψετε πρόγραμμα που θα διαβάζει 300 θετικούς ακέραιους αριθμούς (απαιτείται έλεγχος ορθότητας δεδομένων) θα τους τοποθετεί σε ένα μονοδιάστατο πίνακα 300 θέσεων και θα βρίσκει το μέσο όρο τους. Στη συνέχεια θα τυπώνει τη μικρότερη και τη μεγαλύτερη, κατ' απόλυτη τιμή, διαφορά των στοιχείων του πίνακα από το μέσο όρο αντίστοιχα.

```
#include<iostream>
#include<math.h>
using namespace std;

#define ARRAYSIZE 300

int main()
{
    int a[ARRAYSIZE],i;
    float m, m1, m2;
```

```
for (i=0; i<ARRAYSIZE; i++)
{
      do
      {
            cout<<"\Assign a positive value to the "
                  <<i+1<<" element of the array: ";
            cin>>a[i];
       }
       while (a[i]<=0);
}
m=a[0];
for (i=1; i<ARRAYSIZE; i++)
      m=m+a[i];
m=m/ARRAYSIZE;
```

```
    m1=fabs(a[0]-m);   //m1 για τη μέγιστη τιμή
    m2=fabs(a[0]-m);   //m2 για τη ελάχιστη τιμή
    for (i=1; i<ARRAYSIZE; i++)
    {
        if (m1<fabs(a[i]-m))
            m1=fabs(a[i]-m);
        if (m2>fabs(a[i]-m))
            m2=fabs(a[i]-m);
    }
    cout<<"\nThe required values are: "<<m1
                    <<" and "<<m2;
    return 0;
}
```

Αλγόριθμοι ταξινόμησης

- https://en.wikipedia.org/wiki/Sorting_algorithm
- https://lamfo-unb.github.io/2019/04/21/Sorting-algorithms/

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

Ταξινόμηση μονοδιάστατου πίνακα με τον αλγόριθμο *Bubble Sort*

```
#include<iostream>
using namespace std;
int main()
{
   float x[100],y;
   int i, j;
   cout<<“\nAssign values to the array x: ”;
```

```
for (i=0; i<100; i++)
{
    cout<<"\nAssign value to the "<<i+1
            <<" element of the array: ";
    cin>>x[i];
}
for (i=0; i<100; i++)
    cout<<x[i]<<"   ";
cout<<endl;
//Εκτύπωση του πίνακα πριν τη ταξινόμηση
```

```
for (i=0; i<99; i++)    // ταξινόμηση
{
    for (j=i+1; j<100; j++)
    {
        if (x[i]>x[j])
        {
            y=x[i];
            x[i]=x[j];
            x[j]=y;
        }
    }
}
```

```
    for (i=0; i<100; i++)
        cout<<x[i]<<“   ”;
    //Εκτύπωση του πίνακα μετά τη ταξινόμηση
    return 0;
}
```

Άσκηση 7.1

Να γράψετε πρόγραμμα που θα διαβάζει ένα μονοδιάστατο πίνακα Α 10 θέσεων και θα δημιουργεί τον πίνακα Β του οποίου τα στοιχεία δίνονται από τον τύπο

B[i]=A[11-i] για κάθε i=1,…,10.

Το πρόγραμμα θα εκτυπώνει και τους δύο πίνακες.

Άσκηση 7.2

Να γράψετε πρόγραμμα που θα διαβάζει δύο μονοδιάστατους πίνακες Α και Β 10 θέσεων και θα δημιουργεί το μονοδιάστατο πίνακα C 20 θέσεων για τον οποίο ισχύει

C[i]=A[i] αν i=1,…,10, και

C[i]=B[i-10] αν i=11,…,20.

Το πρόγραμμα θα τυπώνει και τους τρεις πίνακες.

Άσκηση 7.3

Να γράψετε πρόγραμμα που θα διαβάζει δύο μονοδιάστατους πίνακες 10 θέσεων και θα βρίσκει το άθροισμά τους. Στη συνέχεια θα τυπώνει τους δυο πίνακες καθώς και τον πίνακα που προκύπτει από το άθροισμά τους, αφού τους ταξινομήσει σε φθίνουσα σειρά.

Άσκηση 7.4

Να γράψετε πρόγραμμα που θα διαβάζει δύο μονοδιάστατους πίνακες 10 θέσεων και θα βρίσκει το Hadamard γινόμενό τους. Στη συνέχεια θα τυπώνει τους δυο πίνακες καθώς και τον πίνακα που προκύπτει από το Hadamard γινόμενό τους, αφού τους ταξινομήσει σε φθίνουσα σειρά.