```
#include <iostream>
int count(int x[], int my_n, int n);
int main()
{
   const int SIZE = 100;
   int x[SIZE], new_n = 0, next;

   cin >> next;
   while( next != -1 || new_n < 100) {
        x[new_n] = next;
        new_n++;
        cin >> next;
   }
   cout << count(x, new_n, SIZE);
   return 0;
}
int count(int x[], int my_n, int n)
{
  int p;
  p = 0;
  for(int i = 0; i < my_n; i++)
      if((x[0]<x[i] && x[i]<x[my_n-1])||(x[0]>x[i] && x[i]>x[my_n-1]))
       p++;
  return p;
}
```

2ος τρόπος:

- Διαβάζει το επόμενο στοιχείο μέχρι το "-1"
- παρακολουθεί το πλήθος των στοιχείων

Πλήθος συγκεκριμένων στοιχείων ενός πίνακα

- Να γραφεί μια συνάρτηση που θα δέχεται έναν **μερικώς συμπληρωμένο** μονοδιάστατο πίνακα τύπου int και θα επιστρέφει το πλήθος των στοιχείων που έχουν τιμή μεταξύ της τιμής του πρώτου στοιχείου και του τελευταίου.
- Ενσωματώστε την συνάρτηση σε ένα κατάλληλο πρόγραμμα:
  - Θα ορίσετε και θα χρησιμοποιείτε μια επιπλέον συνάρτηση για το διάβασμα των στοιχείων του πίνακα
  - Θα εκτυπώνετε το αποτέλεσμα της συνάρτησης

```
#include <iostream>
void read(int x[], int& my_n, int n);
int count(int x[], int my_n, int n);
int main()
{
   const int SIZE = 100;
   int x[SIZE], new_n = 0, next;

   read(x, new_n, SIZE);


   cout << count(x, new_n, SIZE);
   return 0;
}
int count(int x[], int my_n, int n)
{
  int p;
  p = 0;
  for(int i = 0; i < my_n; i++)
      if((x[0]<x[i] && x[i]<x[my_n-1])
       ||(x[0]>x[i] && x[i]>x[my_n-1]))
        p++;
  return p;
}
```

```
void read(int x[], int& my_n, int n)
{
   cin >> next;
   while( next != -1 || new_n < 100)
   {
        x[new_n] = next;
        new_n++;
        cin >> next;
   }
}
```

Πλησιέστερο στο μέσο όρο

- Να γραφεί ένα πρόγραμμα που θα διαβάζει έναν μερικώς συμπληρωμένο πίνακα τύπου double και θα επιστρέφει την τιμή του στοιχείου που είναι το *πλησιέστερο* στον μέσο όρο όλων των στοιχείων.
- Θα πρέπει να χρησιμοποιήσετε τουλάχιστον τρείς συν/σεις:
  - μια για διάβασμα του πίνακα μέχρι 100 στοιχείων
  - δύο για υπολογισμό

```
double avg(double a[], int my_n, int n)
{
  double sum;
  sum = 0.0;
  for(int i = 0; i < my_n; i++)
     sum = sum + a[i];
  if(my_n <= 0)
  {
     cout << "No avg!";
     return -1.0;
  }
  return (sum / my_n);
}
```

```
double closer(double a[], int my_n, int n)
{
  double p, mo;
  mo = avg(a,my_n,n);
  p = a[0];
  for(int i = 0; i < my_n; i++)
        if( fabs(a[i]-mo) < fabs(p-mo) )
              p = a[i];
  return p;
}
```

```
#include <cmath>
```

```
void read(double a[], int& my_n, int n)
{
   cin >> next;
   while( next != -1 || new_n < 100)
   {
        a[new_n] = next;
        new_n++;
        cin >> next;
   }
}
```

```
#include <iostream>
#include <cmath>

void read(double a[], int& my_n, int n);
double avg(double a[], int my_n, int n);
double closer(double a[], int my_n, int n);

int main()
{
   const int SIZE = 100;
   double a[SIZE],
   int new_n = 0;

   read(a, new_n, SIZE);

   cout << closer(a, new_n, SIZE);
   return 0;
}

void read(double a[], int& my_n, int n)
...
double closer(double a[], int my_n, int n)
...
double avg(double a[], int my_n, int n)
...
```

Ελάχιστη απόλυτη τιμή

- Να γραφεί μια συνάρτηση η οποία θα δέχεται έναν μονοδιάστατο πίνακα τύπου double και θα επιστρέφει την ελάχιστη από τις απόλυτες τιμές των στοιχείων του

Ελάχιστη απόλυτη τιμή

- Να γραφεί μια συνάρτηση η οποία θα δέχεται έναν μονοδιάστατο πίνακα τύπου double και θα επιστρέφει την ελάχιστη από τις απόλυτες τιμές των στοιχείων του

```
double closerabs(double a[], int n)
{
  double u;

  u = fabs(a[0]);
  for(int i = 0; i < n; i++)
  {
    if( fabs(a[i]) < u )
        u = fabs(a[i]);
  }

  return u;
}
```

Υπολογισμός πολυωνύμου

- Για να υπολογίσουμε την τιμή ενός πολυωνύμου σ' ένα σημείο x, πρέπει να υπολογίσουμε το άθροισμα:

$$v_0 + v_1x + v_2x^2 + v_3x^3 + ...$$

- Να γραφεί μια συνάρτηση η οποία

α) Θα δέχεται έναν μονοδιάστατο πίνακα v τύπου double και μια τιμή x, επίσης double.

β) Θα υπολογίζει και θα επιστρέφει την τιμή του πολυωνύμου.

Υπολογισμός πολυωνύμου

- Για να υπολογίσουμε την τιμή ενός πολυωνύμου σ' ένα σημείο x, πρέπει να υπολογίσουμε το άθροισμα:

$$v_0 + v_1x + v_2x^2 + v_3x^3 + \ldots$$

- Να γραφεί μια συνάρτηση η οποία

α) Θα δέχεται έναν μονοδιάστατο πίνακα v τύπου double και μια τιμή x, επίσης double.

β) Θα υπολογίζει και θα επιστρέφει την τιμή του πολυωνύμου.

```
double compute(double v[], int n, double x)
{
  double s;
  s = 0;
  for(int i = 0; i < n; i++)
        s = s + v[i]*pow(x,i);
  return s;
}
```

Εύρεση max από 2 πίνακες

- Να γραφεί συνάρτηση που δέχεται δύο πίνακες a,b και επιστρέφει ποιος πίνακας από τους δύο έχει το μεγαλύτερο άθροισμα.
- Θεωρούμε ότι αν η συνάρτηση επιστρέφει
  - 1 τότε αναφερόμαστε στον πίνακα a (sum(a) > sum(b)),
  - 2 τότε αναφερόμαστε στον πίνακα b (sum(a) < sum(b)), και
  - 3 τότε αναφερόμαστε και στους 2 πίνακες (sum(a)=sum(b))

```
int maxAB(int a[], int na, int b[], int nb)
{
    int i, suma=0, sumb=0;

    for(i = 0; i < na; i++)
        suma = suma + a[i];

    for(i = 0; i < nb; i++)
        sumb = sumb + b[i];

    if(suma > sumb)
        return 1;
    if(suma < sumb)
        return 2;
    if(suma = sumb)
        return 3;
}
```

Το τρίγωνο του Pascal

- Να γραφεί μια συνάρτηση που θα δέχεται έναν δισδιάστατο ακέραιο πίνακα με το πολύ 100 στήλες και θα τον γεμίζει με τις τιμές των στοιχείων του τριγώνου του Pascal.
- Το τρίγωνο του Pascal περιέχει σε κάθε γραμμή τους συντελεστές της ανάπτυξης του $(A+B)^K$, όπου K είναι ο αριθμός της γραμμής:

K = 1	1	1				
K = 2	1	2	1			
K = 3	1	3	3	1		
K = 4	1	4	6	4	1	
K = 5	1	5	10	10	5	1

- Η γραμμή K του τριγώνου έχει K+1 στοιχεία.
  - Το πρώτο και το τελευταίο είναι μονάδα.
  - Κάθε ενδιάμεσο στοιχείο σχηματίζεται σαν άθροισμα των δυο στοιχείων της προηγούμενης γραμμής που βρίσκονται ακριβώς πάνω από αυτό και στην αμέσως προς αριστερά θέση.
- Φυσικά, αυτός ο κανόνας δεν ισχύει για την πρώτη γραμμή που δεν έχει ενδιάμεσα στοιχεία, ούτε υπάρχει προηγούμενη γραμμή.

- Τα υπόλοιπα στοιχεία του πίνακα έχουν τιμή 0.

Το τρίγωνο του Pascal

```
void pascal(int p[][100], int n1)
{
  int i,j;

  for(i = 0; i < n1; i++)
        for(j = 0; j < 100; j++)
                p[i][j]=0;

  p[1][0]=1;
  p[1][1]=1;

  for(i=2; i<n1; i++)
  {
        p[i][0]=1;
        for(j = 1; j < 100; j++)
                p[i][j] = p[i-1][j-1] + p[i-1][j];
  }
}
```

Ανάστροφος

- Να γραφεί μια μέθοδος η οποία θα δέχεται έναν μερικώς συμπληρωμένο δισδιάστατο τετράγωνο πίνακα τύπου double με το πολύ 100 στήλες και θα αναστρέφει τον πίνακα εσωτερικά, μέσα στον ίδιο πίνακα.

Ανάστροφος

- Να γραφεί μια μέθοδος η οποία θα δέχεται έναν μερικώς συμπληρωμένο δισδιάστατο τετράγωνο πίνακα τύπου double με το πολύ 100 στήλες και θα αναστρέφει τον πίνακα εσωτερικά, μέσα στον ίδιο πίνακα.

```
void reversed(int a[][100], int n1)
{

  for(int i = 0; i < n1; i++)
  {
        for(int j = 0; j < i; j++)
        {
            double z=a[i][j];
            a[i][j]=a[j][i];
            a[j][i]=z;
        }
  }

}
```

Ανάστροφος

- Να γραφεί μια μέθοδος η οποία θα δέχεται έναν μερικώς συμπληρωμένο δισδιάστατο τετράγωνο πίνακα τύπου double με το πολύ 100 στήλες και θα αναστρέφει τον πίνακα εσωτερικά, μέσα στον ίδιο πίνακα.

```
void reversed(int a[][100], int n1)
{

  for(int i = 0; i < n1; i++)
  {
        for(int j = 0; j < i; j++)
        {
            double z=a[i][j];
            a[i][j]=a[j][i];
            a[j][i]=z;
        }
  }

}
```

ΠΡΟΣΟΧΗ: Αν είχαμε a[i][j]=a[j][i];
τότε θα καταστρέφαμε τον πίνακα a.

Ανάστροφος

- Να γραφεί μια μέθοδος η οποία θα δέχεται έναν μερικώς συμπληρωμένο δισδιάστατο τετράγωνο πίνακα τύπου double με το πολύ 100 στήλες και θα αναστρέφει τον πίνακα εσωτερικά, μέσα στον ίδιο πίνακα.

```
void reversed(int a[][100], int n1)
{

  for(int i = 0; i < n1; i++)
  {
        for(int j = 0; j < i; j++)
        {
            double z=a[i][j];
            a[i][j]=a[j][i];
            a[j][i]=z;
        }
  }

}
```

ΠΡΟΣΟΧΗ: Αν αντί για j < i γράψουμε j<100, τότε η συν/ση δεν θα δουλέψει. Προσπαθήστε να εντοπίσετε την αιτία της αποτυχίας.

ΠΡΟΣΟΧΗ: Αν είχαμε a[i][j]=a[j][i]; τότε θα καταστρέφαμε τον πίνακα a.

Εύρεση γραμμής μεγίστου στοιχείου

- Να γραφεί μια συνάρτηση η οποία θα δέχεται έναν πίνακα τύπου double με 50 στήλες και θα επιστρέφει τον αριθμό της γραμμής στην οποία ανήκει το μεγαλύτερο στοιχείο.

Εύρεση γραμμής μεγίστου στοιχείου

- Να γραφεί μια συνάρτηση η οποία θα δέχεται έναν πίνακα τύπου double με 50 στήλες και θα επιστρέφει τον αριθμό της γραμμής στην οποία ανήκει το μεγαλύτερο στοιχείο.

```
int maxline(int a[][50], int n1)
{
  int i,j,im,jm;
  im = 0; jm = 0;
  for(i = 0; i < n1; i++)
     for(j = 0; j < 50; j++)
     {
        if( b[i][j] > b[im][jm] )
        {
           im=i;
           jm=j;
        }
     }
  return im;
}
```

Καλή Μελέτη

- **Βιβλιογραφία**

[1] W. Savitch, Πλήρης C++, Εκδόσεις Τζιόλα, 2011

[2] H. Deitel and P. Deitel, C++ Προγραμματισμός 6η Εκδοση, Εκδόσεις Μ. Γκιούρδας, 2013

Ύλη βιβλιογραφίας	
[1]:	Κεφάλαια: **1**, **2**, **3**, **4**, **5**, 9, 13 Ενότητες: 6.1 Παραρτήματα: 1, 2, 3
[2]:	Κεφάλαια: **1**, **2**, **4**, **5**, **6**, **7**, 18, 19, 21 Ενότητες: 8.6, 17.1-17.10 Παραρτήματα: Α, Β, Γ, Δ, ΣΤ

Ανοικτά Ακαδημαϊκά Μαθήματα

Πανεπιστήμιο Ιωαννίνων

Τέλος Ενότητας

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Ιωαννίνων**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημειώματα

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης