242 - Εισαγωγή στους Η/Υ

Τμήμα Μαθηματικών,
Πανεπιστήμιο Ιωαννίνων

Ακαδημαϊκό Έτος 2015-2016

Άρτια Α.Μ. (0-2-4-6-8)

Πίνακες

Πίνακας-Παράταξη

Πίνακες

- Πίνακες
  - Δομή με πολλές θέσεις και με ίδιο τύπο δεδομένα σε όλες τις θέσεις
  - Στατικό μέγεθος κατά την διάρκεια του προγράμματος (για τώρα)
  - Υπάρχουν και τρόποι δήλωσης και χρήσης δυναμικών δομών (ίσως αργότερα)

Πίνακας-Παράταξη

- Η πιο κοινή δομή δεδομένων στον προγραμματισμό
- **Παράταξη**, Μήτρα (array, matrix)
- Οποιοδήποτε σύνολο στοιχείων μπορεί να αποθηκευθεί σε κάποιο πίνακα
- Ένας πίνακας θεωρείται ως ένα σύνολο θέσεων (κελιών), καθεμία από τις οποίες περιγράφεται με ένα σύνολο «συντεταγμένων».
- Αντί να συσχετίσουμε πολλά ονόματα με τις επιμέρους λέξεις μιας περιοχής μνήμης (μεταβλητές), συσχετίζουμε ένα όνομα με ολόκληρη την περιοχή και τις επιμέρους λέξεις τις διακρίνουμε με δείκτες.

Διαστάσεις Πινάκων

- Μονοδιάστατος Πίνακας 1xN:

1	2	3	...	N

- Δύο Διαστάσεων Πίνακας MxN:

1,1	1,2	1,3	…	1,n
2,1	2,2	2,3	…	2,n
…	…	…	…	…
m,1	m,2	m,3	…	m,n

M γραμμές

N στήλες

Χαρακτηριστικά Πινάκων

- Ποιες είναι οι διαστάσεις του πίνακα
- Πώς γίνεται αναφορά σε ένα στοιχείο του (είναι διαφορετικός ο τρόπος σε ΕΑΓ και C++)
- Ποιο είναι το μέγεθος του πίνακα, δηλαδή πόσα στοιχεία μπορεί να αποθηκεύσει.
- Ποιος είναι ο τύπος δεδομένων των στοιχείων του πίνακα (όλα τα στοιχεία ενός πίνακα είναι **υποχρεωτικά** του ίδιου τύπου δεδομένων)

Χαρακτηριστικά Πινάκων

- Ποιες είναι οι διαστάσεις του πίνακα
- Πώς γίνεται αναφορά σε ένα στοιχείο του (είναι διαφορετικός ο τρόπος σε ΕΑΓ και C++)
- Ποιο είναι το μέγεθος του πίνακα, δηλαδή πόσα στοιχεία μπορεί να αποθηκεύσει.
- Ποιος είναι ο τύπος δεδομένων των στοιχείων του πίνακα (όλα τα στοιχεία ενός πίνακα είναι **υποχρεωτικά** του ίδιου τύπου δεδομένων)

Τα χαρακτηριστικά προέρχονται από κάθε δήλωση:

ΕΑΓ: **δηλωση** (A[1:N, 1:M]) **παραταξη** **ακερ**;

C++: `int A[N][M];`

Πλεονεκτήματα - Μειονεκτήματα

- Η αρχικοποίηση και η προσπέλασή τους, τις περισσότερες φορές, απαιτεί τη χρήση δομών επανάληψης
- Όχι μόνο καταλαμβάνουν πολύ χώρο στη μνήμη, αλλά απαιτούν και περισσότερο χρόνο για την επεξεργασία τους.
- Έχουν το πλεονέκτημα ότι επιτρέπουν την τυχαία προσπέλαση οποιουδήποτε στοιχείου τους,
- αλλά και το μειονέκτημα ότι έχουν σταθερές διαστάσεις που *δεν αλλάζουν* κατά την εκτέλεση του προγράμματος.

Πίνακες στην ΕΑΓ

- Δήλωση:

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

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

- Προσπέλαση: <ονομα> [δ1, δ2, ..., δν]
- Π.χ.: Α[5] Β[2, 10] Β[Ι,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[N];`

  `float B[N][M], 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]`

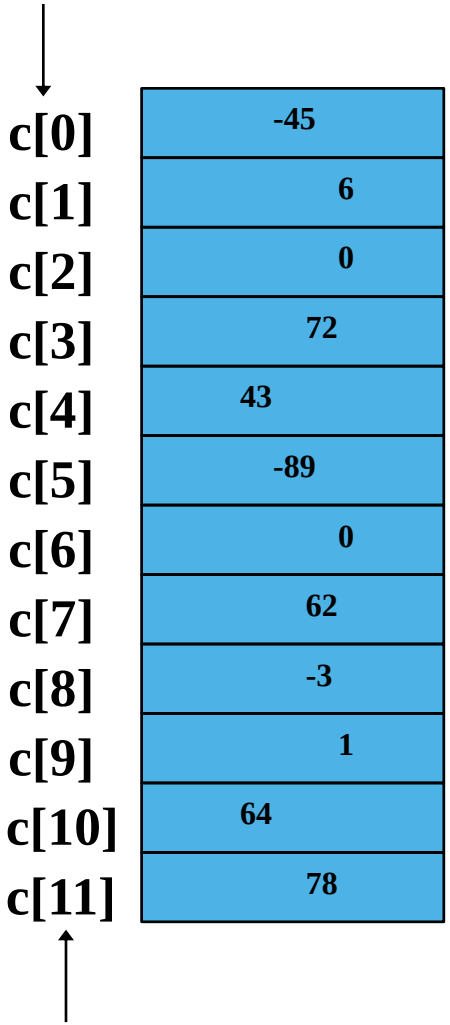
- Προσπέλαση:

  `<ονομα> [δ1] [δ2] ... [δν]`

- Π.χ.: `A[0]` `B[0][10]` `B[I][J]`

Πίνακες

Πίνακας με όνομα c και 12 στοιχεία

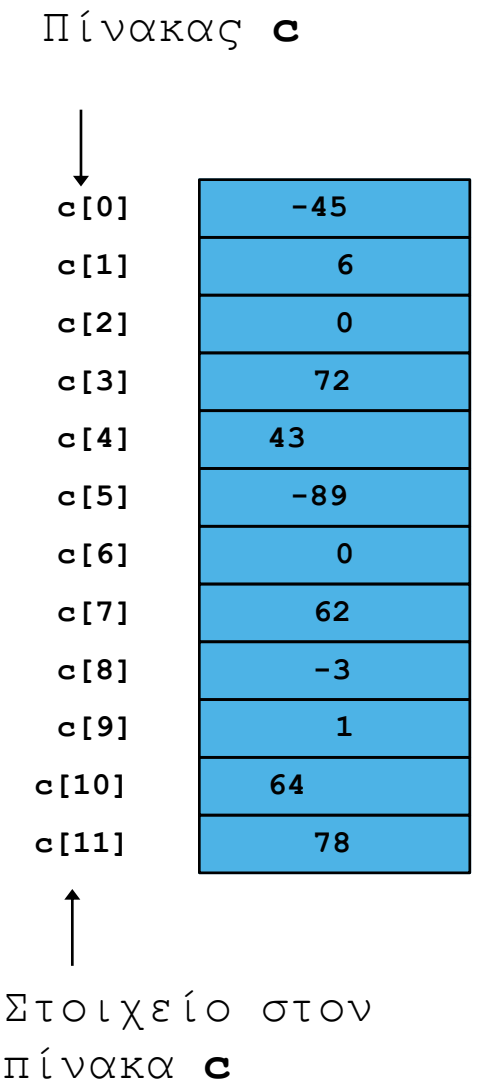


Δείκτης θέσης στοιχείων στον πίνακα c

Πίνακες

- Πίνακες
  - Συνεχόμενες θέσει στην μνήμη
  - Ίδιο όνομα και τύπος
- Αναφορά σε στοιχείο
  - Όνομα πίνακα
  - Θέσει στον πίνακα
- Συντακτικό

  όνομα **[** *δείκτης* **]**
  - Πρώτη θέση - **0**
  - **n** στοιχεία στον πίνακα **c**:
    - **c[0], c[1]...c[n – 1]**



Πίνακες

- Παράδειγμα – δεικτοφόρες μεταβλητές χρησιμοποιούνται σαν τις απλές μεταβλητές

```
c[ 0 ] =  3;
cout<< c[ 0 ];
```

  - Μπορούμε να κάνουμε πράξεις και με τους δείκτες
  - `x = 3; c[ 5 - 2 ] == c[ 3 ] == c[ x ]`

Δήλωση πίνακα

- Ένας πίνακας
  - Όνομα
  - Τύπος
  - Πλήθος στοιχείων

    ```
    arrayType arrayName[ size ];
    ```
  - `int c[ 10 ];`
  - `float myArray[ 3284 ];`
- Δήλωση πολλών πινάκων
  - `int b[ 100 ], x[ 27 ];`

Δήλωση πίνακα

- Αρχικοποίηση

```
int n[ 5 ] = { 1, 2, 3, 4, 5 };
```

  - Εάν δεν φτάνουν οι τιμές τότε τα δεξιότερα στοιχεία γίνονται **0**

```
int n[ 5 ] = { 0 } // όλα 0
```

  - Εάν δοθούν περισσότερες τιμες τότε υπάρχει συντακτικό λάθος
  - Προσοχή. Η C/C++ δεν κάνει έλεγχο ορίων στους πίνακες

- Αν το μέγεθος παραλειφθεί

```
int n[ ] = { 1, 2, 3, 4, 5 };
```

  - Ο πίνακας παίρνει το μέγεθος των διδομένων- εδώ πίνακας 5 στοιχείων

Είσοδος, εκτύπωση, άθροιση στοιχείων ενός πίνακα

```
int i, sum, A[5];
A[0] = 1;
A[1] = 1;
A[2] = 1;
A[3] = 1;
A[4] = 1;
for(i=0;i<5;i++)
{
    cout << "A[" << i << "] = " <<
                A[i] << endl;
    cout << "Nea timi: ";
    cin >> A[i];
}
sum = 0;
for(i=0; i<5; i++)
{
    sum = sum + A[i];
}
```

```
for(i=0; i<5; i++)
{
   cout << "A[" << i <<
       "] = " << A[i] <<
              endl;
}

cout<<"To athroisma tou
      pinaka einai "
     << sum <<endl;
```

Πρόγραμμα

```
#include <iostream>
#include <cstdlib>
using namespace std;
// δήλωση - αρχικοποίηση πινάκων
int main(int argc, char *argv[])
{ system("chcp 1253");
   int vec1[5]; // πίνακας με 5 θέσεις
   int vec2[]={10,20,30,40,50}; // πίνακας με 5 θέσεις
   int vec3[5]; // πίνακας με 5 θέσεις
   // αρχικοποίηση vec1
   for (int i=0; i<5; i++)
       vec1[i]=i*i; // τιμή i*i
   // εκτύπωση vec1
   cout <<"Στοιχεία του πίνακα vec1"<<endl;
   for (int j=0; j<5; j++)
       cout<<"vec1["<<j<<"]= "<<vec1[j]<<endl;
```

Πρόγραμμα

- …… συνέχεια

```
    // εκτύπωση vec2
    cout <<"Στοιχεία του πίνακα vec2"<<endl;
    for (int j=0; j<5; j++)
        cout<<"vec2["<<j<<"]= "<<vec2[j]<<endl;
    // τιμές για vec3 από χρήστη
    for (int i=0; i<5; i++) {
        cout<<"Τιμή για vec3["<<i<<"]: ";
        cin>>vec3[i]; // τιμή από χρήστη
        }
    // εκτύπωση vec3
    cout <<"Στοιχεία του πίνακα vec3"<<endl;
    for (int j=0; j<5; j++)
        cout<<"vec3["<<j<<"]= "<<vec3[j]<<endl;
      system("PAUSE");
      return 0;
}
```

Μέγιστο, Ελάχιστο, Μέσο όρο

- Έστω A[N]

- Μέγιστο:

```
max = -100000;
for (I=0; I < N; I++)
     if( max < A[I] )
          max = A[I];
```

- Μέσος όρος:

```
for (I=0; I < N; I++)
     sum = sum + A[I];
mesos = sum / N;
```

Παράδειγμα - Τυχαίοι αριθμοί

- Να δηλώσετε ένα ακέραιο μονοδιάστατο πίνακα Α με διάσταση 50 και να τον αρχικοποιήσετε με τυχαίους ακεραίους στο διάστημα [1,100]

- Να εκτυπώσετε τον πίνακα από Α[0] -> Α[49]

- Να εκτυπώσετε τον πίνακα από Α[49] -> Α[0]

- Να εκτυπώσετε τον πίνακα έτσι ώστε στην έξοδο να είναι 10 ακέραιοι ανά γραμμή

Παράδειγμα - Τυχαίοι αριθμοί

```
int i, N, A[50];
 N = 50;
 // Anathesi tyxaiwn timwn ston pinaka A:
 cout<<"h rand paragei tyxaious apo 0 evw "
<<RAND_MAX<<"emeis theloume 1-100"<<endl;
 for(i=0; i<N; i++)
 { A[i] = rand() % 100 + 1; }
 //Ektypwsi tou pinaka A:
 cout << "O pinakas A einai o exis: << endl;
 for(i=0; i<N; i++)
 { cout << A[i] << " - "; }
 cout << endl;
 cout << "O pinakas A antistrofa  einai o
          exis: << endl;
 for(i=0; i<N; i++)
 { cout << A[N-1-i] << " - "; }
 cout << endl;
```

Παράδειγμα - Τυχαίοι αριθμοί

```
/* random 1-dimensional array */
#include <iostream>
#include <cstdlib>
using namespace std;
int main(int argc, char *argv[])
{ int i, N, A[50];
  N = 50;
  // Anathesi tyxaiwn timwn ston pinaka A:
  cout<<"Enhmerwsh: h rand paragei tyxaious apo 0 ews "
<<RAND_MAX<<endl;
  cout<<"emeis theloume 1-100"<<endl;
  for(i=0; i<N; i++)
    { A[i] = rand() % 100 + 1; }
    //Ektypwsi tou pinaka A:
  cout << "O pinakas A einai o ekshs:"<< endl;
```

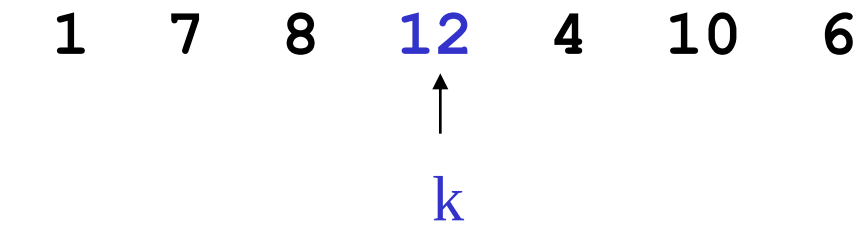
Παράδειγμα - Τυχαίοι αριθμοί

```
for(i=0; i<N; i++)
   { cout << A[i] << " "; }
cout << endl;
cout << "O pinakas A antistrofa  einai o ekshs:"<< endl;
for(i=0; i<N; i++)
  { cout << A[N-1-i] << " "; }
cout << endl;
cout << "O pinakas me 10 stoixeia ana grammh einai o ekshs:"<< endl;
for(i=0; i<N; i++)
  { cout << A[i] << " ";
    if ((i+1)%10==0) cout << endl; }
cout << endl;
system("Pause");
return 0;
}
```

Αναζήτηση 2ου μεγίστου

- Μας δίνεται ένας μονοδιάστατος πίνακας Α με 100 ακέραια στοιχεία και ζητάμε το 2ο μέγιστο του πίνακα Α.

1. Μπορούμε να ταξινομήσουμε τα στοιχεία του πίνακα Α κατά φθίνουσα τάξη και να τυπώσουμε το δεύτερο στοιχείο του πίνακα (Α[2]).
2. Άλλος (πιο γρήγορος) τρόπος;
   - Έστω ότι βρίσκουμε το μέγιστο του Α στην θέση k.
   - Μπορούμε να αναζητήσουμε το μέγιστο στα στοιχεία του πίνακα που δεν δεικτοδοτούνται από το k.



```
διαδικασια ΜΕΓ_ΔΕΥΤΕΡΟ
δηλωση (I, ΜΕΓ_I, ΜΕΓ_II) ακερ;
δηλωση ( A [1:100] ) παραταξη ακερ;
αρχη
    ΜΕΓ_I ← 1;
    για I ← 1 εως 100 επαναλαβε
    (   εαν A[I] > A[ΜΕΓ_I]
            τοτε      ( ΜΕΓ_I ←I;  )
    )

    ΜΕΓ_II ← 1;
    για I ← 1 εως 100 επαναλαβε
    (   εαν  I ≠ ΜΕΓ_I
            τοτε      ( εαν A[I] > A[ΜΕΓ_II]
                              τοτε   ( ΜΕΓ_II ←I;  )
                      )
    )
τελος ΜΕΓ_ΔΕΥΤΕΡΟ
```

Βρίσκουμε την *θέση* του μέγιστου στοιχείου

Ψάχνουμε πάλι το μέγιστο για όλες τις θέσεις του πίνακα εκτός από *την θέση του μεγίστου*

Στο τέλος τυπώνουμε το Α[ΜΕΓ_ΙΙ]

Παράδειγμα

- Διαβάζει έναν πίνακα 10 ακεραίων και
  - a) υπολογίζει τον ελάχιστο ακέραιο αριθμό (και την θέση) του πίνακα
  - b) υπολογίζει τον δεύτερο ελάχιστο ακέραιο αριθμό (και την θέση) του πίνακα.

```
int I, A[10], min1, min2, thesi1,thesi2;
for (I=0; I < 10; I++)
    cin >> A[I];
min1 = A[0];
thesi1 = 0;
for (I=0; I < 10; I++)
    if(min1 > A[I])
    {
          min1 = A[I];
          thesi1 = I;
    }
```

```
min2 = 9999999;
thesi2 = 0;
for (I=0; I < 10; I++)
  if(I != thesi1)
      if(min2 > A[I])
      {
            min2 = A[I];
            thesi2 = I;
      }
cout << ...;
```

Καταμέτρηση στοιχείων

- Μας δίνεται ένας μονοδιάστατος πίνακας Α με 100 ακέραια στοιχεία και ένας αριθμός Χ.
- Ζητάμε το πλήθος των στοιχείων του πίνακα Α τα οποία είναι μεγαλύτερα από τον αριθμό Χ.

	1	2	3	4	5	6	7
A:	**1**	**7**	**8**	**12**	**4**	**10**	**6**

X = 9 | 12>9,10>9 | Μετρητής = 2

```
διαδικασια ΚΑΤΑΜΕΤΡΗΣΗ
δηλωση (Ι, Χ, ΜΕΤΡ) ακερ;
δηλωση ( Α [1:100] ) παραταξη ακερ;
αρχη
    για Ι ← 1 εως 100 επαναλαβε
    (
        διαβασε  Α[Ι];
    )
     διαβασε  Χ;

    ΜΕΤΡ ← 0;
    για Ι ← 1 εως 100 επαναλαβε
    (   εαν Α[Ι] > Χ
        τοτε      (  ΜΕΤΡ ← ΜΕΤΡ + 1;   )
    )

    τυπωσε  ΜΕΤΡ;
τελος ΚΑΤΑΜΕΤΡΗΣΗ
```

Παράδειγμα, >X,<X,=X

- Διαβάζει ένα πίνακα Α, 10 ακεραίων, καθώς και έναν αριθμό Χ και υπολογίζει το πλήθος των στοιχείων

  α) μεγαλύτερα από το Χ β) ίσα με το Χ και γ) μικρότερα από το Χ.

```
int I, A[10], big, small, equal;

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

big = 0;
small = 0;
equal = 0;
```

```
for (I=0; I < 10; I++)
{
   if(A[I] > X)
      big = big + 1;
   if(A[I] < X)
      small = small + 1;
   if(A[I] == X)
      equal = equal + 1;
}
cout << big << small << equal;
```

Καταμέτρηση στοιχείων σε όρια

- Μας δίνεται ένας πίνακας Α με 100 ακέραια στοιχεία, και ζητάμε το πλήθος των στοιχείων του πίνακα που έχουν τιμή ανάμεσα από το πρώτο και το τελευταίο στοιχείο.

	1	2	3	4	5	6	7
A:	1	7	6	12	4	10	8

```
διαδικασια ΚΑΤΑΜΕΤΡΗΣΗ_ΟΡΙΑ
δηλωση (Ι, Χ, ΜΕΤΡ) ακερ;
δηλωση ( Α [1:100] ) παραταξη ακερ;
αρχη
    για Ι ← 1 εως 100 επαναλαβε
    (
        διαβασε  Α[Ι];
    )
    ΜΕΤΡ ← 0;
    για Ι ← 2 εως 99 επαναλαβε
    (
        εαν  Α[1] < Α[Ι]  &  Α[Ι] <Α[100]  τοτε
        (
            ΜΕΤΡ ← ΜΕΤΡ + 1;
        )
    )
    τυπωσε  ΜΕΤΡ;
τελος ΚΑΤΑΜΕΤΡΗΣΗ_ΟΡΙΑ
```

Καταμέτρηση στοιχείων σε όρια

```
// katametrhsh  A[0]<X<A[N-1]
#include <iostream>
#include <cstdlib>
using namespace std;
int main(int argc, char *argv[])
{ system("chcp 1253");
   int i,N=10, count;
   int A[N]; //data elements
   // eisodos A
   for (i=0; i<N; i++) {
      cout<<"DOSE TIMH GIA A["<<i<<"]: ";
      cin>>A[i]; // eisodos
      }
```

Καταμέτρηση στοιχείων σε όρια

- ……..συνέχεια
- // ektypwsh A
- cout <<"TA STOIXEI TOY PINAKA A POY EDOSE O XRHSTHS EINAI"<<endl;
- for (i=0; i<N; i++)
- cout<<"A["<<i<<"]= "<<A[i]<<endl;
- // metrhma
- count=0;
- for (i=0; i<N; i++)
- if (A[0]<A[i] && A[i]<A[N-1]) count++;
- cout<<"Plhthos A[0]<X<A[N-1] -> "<<count<<endl;
- system("PAUSE");
- return 0;
- }

Υπολογισμός σειράς

- Να γραφτεί πρόγραμμα σε που διαβάζει ένα n (ακέραιο) και υπολογίζει την τιμή του αθροίσματος:

$$\pi / 4 = \sum_{i=0}^{n} \frac{(-1)^i}{(2i+1)}$$

- Όταν το n->∞ το άθροισμα προσεγγίζει το π/4
- Σκεφτείτε πώς μπορούμε να υπολογίσουμε πρώτα κάθε τιμή του αθροίσματος...

```
S ← 0
για I ← 1 εως N επαναλαβε
(        S ← S+oros_athroismatos;     )
```

Υπολογισμός σειράς

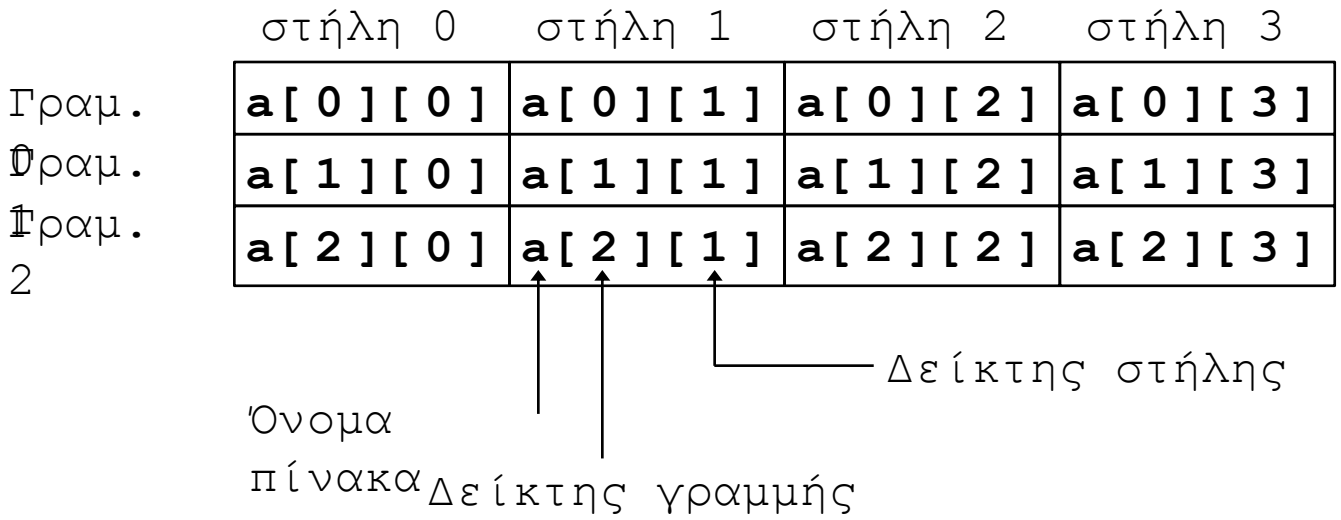
```
/* Pi/4 = seira  (-1)^i/(2i+1), i=0,... */
#include <iostream>
#include <cstdlib>
#include <cmath>
#define Max 200
using namespace std;
int main(int argc, char *argv[])
{ int i, N;
  long double X, ser, A[Max];
  cout<<"Dose plhthos orwn <"<<Max<<": ";
  cin>>N;
```

Υπολογισμός σειράς

```
ser=0;
for(i=0; i<N; i++)
  {ser=ser+pow(-1.0,i)/(2*i+1); A[i]=ser;}
cout<<"Athroisma seiras P="<<4*ser<<endl;
cout<<"Times ston pinaka gia Pi/4"<<endl;
cout<<"ektypwsh ana 10 times ston pinaka"<<endl;
for(i=0; i<N; i++)
  if ((i+1)%10==0) {cout<<A[i]<<" "; cout << endl;}
system("Pause");
return 0;
}
```

Πολυδιάστατοι Πίνακες

- Πίνακες 2 (η περισσοτέρων) διαστάσεων
  - Γραμμές και στήλες (nxm πίνακας)
  - 3x4 πίνακας



Δήλωση πίνακα

- Αρχικοπόιηση
  - `int b[2][2]={{1,2},{3,4}};`

1	2
3	4

  - Με χρήση {}
  - Εάν δεν δίνονται αρκετά στοιχεία τότε το υπόλοιπα αρχικοποιούνται στο 0
  - `int b[2][2]={{1},{3,4}};`

1	0
3	4

- Αναφορές στα στοιχεία
  - `cout<<b[0][1];`

Παράδειγμα

```
#include <iostream>
#include <cstdlib>
using namespace std;
// είσοδος-έξοδος-επεξεργασία δυσδιάστατου πίνακα
int main(int argc, char *argv[])
{ system("chcp 1253");
  int i,j,temp,sum;
  int n; /* διάσταση πίνακα */
  cout<<"Ο πίνακας είναι nxn - δώσε το n: ";
  cin>>n;
  int item[n][n];  /* n αριθμούς */
  cout<<"\nΔώσε τους αριθμούς\n";
  for (i=0; i<n; i++)
   for (j=0; j<n; j++) cin>>item[i][j];
```

Παράδειγμα …. συνέχεια

```
……. Συνέχεια

 // εκτύπωση πίνακα
  cout<<"Ο πίνακας είναι\n";
  for (i=0; i<n; i++) {
   for (j=0; j<n; j++) cout<<item[i][j]<<" ";
   cout<<"\n";
  }
  /* άθροισμα στοιχείων γραμμών */
  for (i=0; i<n; i++) {
    sum=0;
    for (j=0; j<n; j++) sum +=item[i][j];
    cout<<"άθροισμα "<<i<<" γραμμής "<<sum<<"\n";
   }
     system("PAUSE");
     return 0;
}
```

Παράδειγμα

- Θέλουμε να εισάγουμε ακέραια στοιχεία σε έναν πίνακα Α διάστασης 3x5 (κατά γραμμή) και να υπολογίσουμε τον ανάστροφο πίνακα του Α.
- Παράδειγμα:

A =

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

$B = A^T =$

1	6	11
2	7	12
3	8	13
4	9	14
5	10	15

```
διαδικασια ΑΝΑΣΤΡ
δηλωση (I, J) ακερ;
δηλωση (A(1:3, 1:5) ) παραταξη ακερ;
αρχη
    για I ← 1 εως 3 επαναλαβε
    (
          για J ← 1 εως 5 επαναλαβε
                (διαβασε A[ I, J ];  )
    )

    για J ← 1 εως 5 επαναλαβε
    (
          για I ← 1 εως 3 επαναλαβε
                (
                 B[ J ,  I ] ← A[ I , J ] ;
                )
    )
τελος ΑΝΑΣΤΡ
```

```
// όνομα αρχείου ANAS.cpp
#include <iostream>
main()
{
   int I, J, A[10][10];

   for (I=0; I < 3; I++)
       for (J=0; J < 5; J++)
             cin >> A[I][J];

   for (J=0; J < 5; J++)
       for (I=0; I < 3; I++)
           B[J][I] = A[I][J];
}
```

Μέγιστο, Ελάχιστο, Μέσο όρο

- Έστω B[N][M]
- Ελάχιστο:

```
min = 100000;
for (I=0; I < N; I++)
    for (J=0; J < M; J++)
        if( min > B[I][J] )
            min = B[I][J];
```

- Μέσος όρος:

```
for (I=0; I < N; I++)
    for (J=0; J < M; J++)
        sum = sum + B[I][J];
mesos = sum / (N*M);
```

Παράδειγμα k-στήλης

- Να γραφτεί ένα πρόγραμμα σε ΕΑΓ και σε C++ που θα διαβάζει έναν πίνακα 4x8, έναν αριθμό k (που θα αντιπροσωπεύει την k-στήλη) και θα επιστρέφει το άθροισμα των στοιχείων της k- στήλης.

5	6	8	8	8	4	4	6
3	5	7	8	4	3	5	5
9	8	7	6	5	5	4	6
4	5	1	1	9	9	9	5

k = 4 → 23

```
διαδικασια ΣΤΗΛΗ
δηλωση (I, J, K, ΑΘΡ) ακερ;
δηλωση (Α(1:4, 1:8) ) παραταξη ακερ;
αρχη
    για I ← 1 εως 4 επαναλαβε
    (
          για J ← 1 εως 8 επαναλαβε
                (διαβασε Α[ I, J ];  )
    )
    διαβασε K;

    ΑΘΡ ← 0;
    για I ← 1 εως 4 επαναλαβε
         ( ΑΘΡ ← ΑΘΡ + Α [I, K]; )
     τυπωσε ΑΘΡ;
τελος ΣΤΗΛΗ
```

```
// όνομα αρχείου COL.cpp
#include <iostream>
main()
{
   int I, J, K, SUM;
   int A[4][8];

   for (I=0; I < 4; I++)
       for (J=0; J < 8; J++)
             cin >> A[I][J];
   cin >> K;
   SUM = 0;
   for (I=0; I < 4; I++)
       SUM = SUM + A[I][K];
   cout << SUM;
}
```

Παράδειγμα: *πολλαπλασιαστικός πίνακας*

```
#include <iostream>
#include <iomanip>
#include <cstdlib>
#define max1 10
#define max2 10
using namespace std;
// δυσδιάστατοι πίνακες – πολλαπλασιαστικός πίνακας
int main(int argc, char *argv[])
{ system("chcp 1253");
  int twod[max1][max2]; //πίνακας 10χ10
  int i,j;
  // υπολογισμός
  for(i=0; i<max1; i++)
    for(j=0; j<max2; j++)
      twod[i][j] = (i+1)*(j+1);
```

Παράδειγμα …. συνέχεια

```
……. συνέχεια
  // εκτύπωση πολλαπλαστιαστικού πίνακα
  cout<<"Πολλαπλασιαστικός "<<max1<<" x "<<max2<<" πίνακας\n";
  for (i=0; i<max1; i++)  {
    for (j=0; j<max2; j++)
      cout<<setw(4)<<twod[i][j];
    cout<<"\n";
  }
     system("PAUSE");
     return 0;
}
```

Άθροισμα *περιφερειακών* στοιχείων

- Μας δίνεται ένας δισδιάστατος πίνακας Α ακεραίων στοιχείων με N γραμμές και M στήλες.
- Ζητάμε το άθροισμα των περιφερειακών (boundary) στοιχείων του πίνακα.

5	6	8	8	8	4	4	6
3	5	7	8	4	3	5	5
9	8	7	6	5	5	4	6
4	5	1	1	9	9	9	5

Πρέπει να υπολογίσουμε 4 μερικά αθροίσματα

```
διαδικασια ΠΕΡΙΦΕΡΕΙΑΚΑ
δηλωση (I, J, N, M, ΑΘΡ) ακερ;
δηλωση ( Α(1:100, 1:100) ) παραταξη ακερ;
αρχη
    διαβασε N, M;
    για I ← 1 εως N επαναλαβε
    (       για J ← 1 εως M επαναλαβε
            (  διαβασε A[ I, J ];  )
    )
    ΑΘΡ ← 0;
    για I ← 1 εως N επαναλαβε
            (    ΑΘΡ ← ΑΘΡ + A [I, 1];
                 ΑΘΡ ← ΑΘΡ + A [I, M];  )
    για J ← 1 εως M επαναλαβε
            (    ΑΘΡ ← ΑΘΡ + A [1, J];
                 ΑΘΡ ← ΑΘΡ + A [N, J];  )
     τυπωσε ΑΘΡ;
τελος ΠΕΡΙΦΕΡΕΙΑΚΑ
```

Άθροισμα **πρώτης στήλης**

Άθροισμα **τελευταίας στήλης**

Άθροισμα **πρώτης γραμμής**

Άθροισμα **τελευταίας γραμμής**

Άθροισμα *κυρίας διαγωνίου*

- Μας δίνεται ένας δισδιάστατος πίνακας Α ακεραίων στοιχείων με N γραμμές και N στήλες.
- Ζητάμε το άθροισμα των στοιχείων του πίνακα που βρίσκονται στην κύρια διαγώνιο.

5	6	8	8
3	5	7	8
9	8	7	6
4	5	1	1

```
A[1,1]
A[2,2]
A[3,3]
A[4,4]
```

```
διαδικασια ΑΘΡΚΥΡΔΙΑΓ
δηλωση (I, J, N, ΑΘΡ) ακερ;
δηλωση A[1:100, 1:100] παραταξη ακερ;
αρχη
     διαβασε N;
    για I ← 1 εως N επαναλαβε
    (
          για J ← 1 εως N επαναλαβε
           (    διαβασε A[ I, J ];  )
    )
    ΑΘΡ ← 0;
    για I ← 1 εως N επαναλαβε
    (
          ΑΘΡ ← ΑΘΡ + A[I, I];
    )
    τυπωσε ΑΘΡ;
τελος ΑΘΡΚΥΡΔΙΑΓ
```

Γινόμενο Πινάκων

- Μας δίνονται δύο δισδιάστατοι πίνακες A και B.
- A: N x N, B: N x N
- Ζητάμε να υπολογίσουμε το γινόμενο AB. Δηλαδή τον πίνακα C = AB διάστασης N x N.

$$C[i, j] = \sum_{k=1}^{n} A[i, k]B[k, j]$$

2	6	4
3	5	7
4	8	7

A

$\cdot$

2	6	8
2	5	1
3	8	2

B

$=$

28	62	30
37	..	..
..	..	..

C

```
διαδικασια ΓΙΝΟΜΕΝΟΜΗΤΡΩΝ
δηλωση (I, J, K, ΑΘΡ) ακερ;
δηλωση ( A(1:N, 1:N), B(1:N, 1:N), C[1:N, 1:N] ) παραταξη ακερ;
αρχη
    για I ← 1 εως N επαναλαβε
    (
        για J ← 1 εως N επαναλαβε
        (
            ΑΘΡ ← 0;
            για K ← 1 εως N επαναλαβε
            (
                ΑΘΡ ← ΑΘΡ + A[I, K] * B[K, J];
            )
            C[I, J] ← ΑΘΡ;
        )
    )
    ΕΚΤΥΠΩΣΗ
τελος ΓΙΝΟΜΕΝΟΜΗΤΡΩΝ
```

```
για I ← 1 εως N επαναλαβε
(       για J ← 1 εως N επαναλαβε
        (   τυπωσε C[I, J]; )
        τυπωσε "\n" ;
)
```

C=AxB

```
// Matrix multiplication C=AxB
#include <iostream>
#define MAX1 3
#define MAX2 3
#define MAX3 3
#define MAX4 3
using namespace std;
int main(int argc, char *argv[])
{  int A[MAX1][MAX2] = {{1,2,3}, {4,5,6}, {7,8,9}};
   int B[MAX3][MAX4] = {{1,0,0}, {0,1,0}, {0,0,1}};
   int i,j,k,C[MAX1][MAX4]={{0}};
   if (MAX2==MAX3) {
      cout<<"Matrix A\n";
      for (i = 0; i < MAX1; i++)     {
         for (j = 0; j < MAX2; j++)
            cout<<A[i][j];
         cout<<"\n";
      }
     cout<<"Matrix B\n";
     for (i = 0; i < MAX3; i++)     {
        for (j = 0; j < MAX4; j++)
           cout<<B[i][j];
        cout<<"\n";
     }
```

C=AxB

```
…. συνέχεια
/* Product */
   for (i = 0; i < MAX1; i++)
       for (j = 0; j < MAX4; j++) {
          C[i][j]=0;
          for (k = 0; k < MAX3; k++)
               C[i][j]=C[i][j]+A[i][k]*B[k][j];
        }

   cout<<"Matrix C=AxB\n";
   for (i = 0; i < MAX1; i++)     {
      for (j = 0; j < MAX4; j++)
          cout<<C[i][j];
       cout<<"\n";
     }
}
 else
       cout<<"Asymvates diastaseis pinakwn\n";
 system("PAUSE");
 return 0;
}
```

Ακολουθία Fibonacci

- Οι αριθμοί Fibonacci είναι μια ακολουθία ακεραίων που έχουν την ιδιότητα ότι κάθε όρος της ακολουθίας ισούται με το *άθροισμα* των δύο προηγούμενων:

  $a_i = a_{i-1} + a_{i-2}$ (νέος_όρος = τελευταίος + προτελευταίος)

- Οι πρώτοι δύο όροι της ακολουθίας είναι 0 και 1.

  $a_0 = 0$ και $a_1 = 1$ 0, 1, 1, 2, 3, 5, 8, 13, ...

- Ζητάμε τον μικρότερο αριθμό Fibonacci που θα υπερβαίνει την τιμή 5000
- Ζητάμε τον μεγαλύτερο αριθμό Fibonacci που δεν θα υπερβαίνει την τιμή 5000

```
διαδικασια ΦΙΜΠΟΝΑΤΣΙ;
δηλωση (ΠΡΟΤΕΛ, ΤΕΛ, ΑΘΡΟΙΣ) ακερ;
αρχη
    ΠΡΟΤΕΛ ← 0;
    ΤΕΛ ← 1;
    ΑΘΡΟΙΣ ← 1;

    εφοσον (ΑΘΡΟΙΣ ≤ 5000) επαναλαβε
        (
            ΠΡΟΤΕΛ ← ΤΕΛ ;
            ΤΕΛ ← ΑΘΡΟΙΣ;
            ΑΘΡΟΙΣ ← ΠΡΟΤΕΛ + ΤΕΛ;
        )

    τυπωσε  ΤΕΛ;
    τυπωσε  ΑΘΡΟΙΣ;
τελος ΦΙΜΠΟΝΑΤΣΙ
```

Εάν ζητάμε ΟΛΟΥΣ τους αριθμούς Fibonacci μέχρι το 5000;

Ακολουθία Fibonacci

```
/* Fibonacci       <5000<     */
#include <iostream>

using namespace std;
int main(int argc, char *argv[])
{ int n, i,
      protel=0, tel=1, epom; //f(i-1), f(i), f(i+1) fib arithmoi
  i=2; // deikths arithmou Fib sthn akolouthia
  while (epom < 5000)
   {
     protel = tel;
     tel = epom;
     epom = protel + tel;
     i++;
   }
  cout << "O megalyteros Fib mikroteros apo 5000 einai o " << tel << endl;
  cout << "O mikroteros Fib megalyteros apo 5000 einai o " << epom << endl;
  system("Pause");
  return 0;
}
```