

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 $1 \times N$:

1	2	3	...	N
---	---	---	-----	---

- Δύο Διαστάσεων Πίνακας $M \times N$:

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:Δn])
παραταξη **<τύπος μεταβλητής>** ;

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

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

Πίνακες στη C++

- Δήλωση:

<τύπος μεταβλητής> <ονομα> [$\Delta 1$] [$\Delta 2$] ... [Δn];

- 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> [$\delta 1$] [$\delta 2$] ... [δn]

- Π.χ.: `A[0] B[0][10] B[I][J]`

Πίνακες

Πίνακας με όνομα *c* και 12 στοιχεία

c[0]	-45
c[1]	6
c[2]	0
c[3]	72
c[4]	43
c[5]	-89
c[6]	0
c[7]	62
c[8]	-3
c[9]	1
c[10]	64
c[11]	78

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 :
 - $c[0], c[1] \dots c[n - 1]$

Πίνακας c

↓	
$c[0]$	-45
$c[1]$	6
$c[2]$	0
$c[3]$	72
$c[4]$	43
$c[5]$	-89
$c[6]$	0
$c[7]$	62
$c[8]$	-3
$c[9]$	1
$c[10]$	64
$c[11]$	78

↑
Στοιχείο στον
πίνακα c

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;`
- `}`

Παραδείγματα

- `#include <iostream>`
- `#include <cstdlib>`
- **using namespace** std;
- **int** main(**int** argc, **char** *argv[])
- { system("chcp 1253");
- */* δίνουμε στον m αρχικές τιμές εδώ */*
- **int** mon[12], m[]={31,28,31,30,31,30,31,31,30,31,30,31};
- **int** k;
- */* Αρχικοποίηση του πίνακα mon */*
- **for** (k = 0; k < 12; k++) mon[k] = 0;
- */* Ανάθεση τιμών στον πίνακα */*
- mon[0] = 31; mon[1] = 28; mon[2] = 31; mon[3] = 30;
- mon[4] = 31; mon[5] = 30; mon[6] = 31; mon[7] = 31;
- mon[8] = 30; mon[9] = 31; mon[10] = 30; mon[11] = 31;

Παραδείγματα

- συνέχεια
- */* Ανάκληση τιμών. Διευθύνσεις στην μνήμη. */*
- `cout<<"Στοιχ.\tΤιμή\tΜνήμη\tΣτοιχ.\tΤιμή\tΜνήμη\n";`
- `for (k = 0; k < 12; k++)`
- `cout<<"mon["<<k<<"]\t"<<mon[k]<<"\t"<<&mon[k]<<"\tm["<<k<<"]\t"<<m[k]<<"\t"<<&m[k]<<endl;`
- `system("PAUSE");`
- `return 0;`
- `}`

Παράδειγμα Προπαίδειας

- Θέλουμε να καταχωρήσουμε το αποτέλεσμα του πολλαπλασιασμού ενός ακέραιου αριθμού Z με όλους τους αριθμούς από 1 έως 10.
- Το αποτέλεσμα θα αποθηκευτεί σε έναν πίνακα ΠΡΟΠ (PROP) που θα έχει 10 θέσεις.
- Στο τέλος θέλουμε να εκτυπώσουμε τα στοιχεία του πίνακα ΠΡΟΠ (PROP).

διαδικασία ΠΡΟΠΑΙΔΕΙΑ1

δηλώση (Z, I) ακερ;

δηλώση (ΠΡΟΠ (1:10)) παραταξη ακερ;

αρχη

διαβασε Z;

για I ← 1 εως 10 επαναλαβε

(ΠΡΟΠ[I] ← Z*I;)

για I ← 1 εως 10 επαναλαβε

(τυπωσε ΠΡΟΠ[I];)

τελος ΠΡΟΠΑΙΔΕΙΑ1

```
// όνομα αρχείου PRO1.cpp
#include <iostream>
main()
{
    int Z, I, PROP[10];

    cin >> Z;

    for (I=0; I<10; I++)
        PROP[I] = Z*I ;

    for (I=0; I<10; I++)
        cout << PROP[I];

}
```

Πολυδιάστατοι Πίνακες

- Πίνακες 2 (η περισσότερων) διαστάσεων
 - Γραμμές και στήλες (n x m πίνακας)
 - 3x4 πίνακας

	στήλη 0	στήλη 1	στήλη 2	στήλη 3
Γραμ. 0	<code>a[0][0]</code>	<code>a[0][1]</code>	<code>a[0][2]</code>	<code>a[0][3]</code>
Γραμ. 1	<code>a[1][0]</code>	<code>a[1][1]</code>	<code>a[1][2]</code>	<code>a[1][3]</code>
Γραμ. 2	<code>a[2][0]</code>	<code>a[2][1]</code>	<code>a[2][2]</code>	<code>a[2][3]</code>

Όνομα πίνακα

Δείκτης γραμμής

Δείκτης στήλης

Δήλωση πίνακα

- Αρχικοποίηση

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

1	2
3	4

- Με χρήση {}

- Εάν δεν δίνονται αρκετά στοιχεία τότε το υπόλοιπο αρχικοποιούνται στο 0

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

1	0
3	4

- Αναφορές στα στοιχεία

- `cout<<b[0][1];`

Παράδειγμα Προπαίδειας 10x10

- Θέλουμε να τυπώσουμε την προπαίδεια των αριθμών 1 έως 10.
- Δηλαδή να βρούμε όλους του δυνατούς συνδυασμούς των αριθμών 1, 2, ..., 10 με τους αριθμούς 1, 2, ..., 10. (το γινόμενό τους)
- Θα τους καταχωρήσουμε σε έναν πίνακα A .
- Τι διάσταση πρέπει να έχει ο A ;
- Πόσα θα είναι τα στοιχεία του;
- Τι τύπος θα είναι ο A ;

Παράδειγμα Προπαίδειας 10x10

- Είσοδος;
- Έξοδος: Έστω ότι υπολογίζουμε τον πίνακα
 $A[10][10]$ (ή $A[1:10, 1:10]$).

Πώς εκτυπώνουμε τα στοιχεία του
(γραμμή-γραμμή);

```
for (I=0; I<10; I++)
  for (J=0; J<10; J++)
    cout << A[I][J];
```

```
1 2 3 4 5 6 7 8 9 10 2 4 6 8 10 12 ...
```

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

```
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 ...
...

```

διαδικασία ΠΡΟΠΑΙΔΕΙΑ2δηλώση (I, J) ακερ;δηλώση (A(1:10, 1:10)) παραταξη ακερ;αρχηγια I ← 1 εως 10 επαναλαβε

(

για J ← 1 εως 10 επαναλαβε
(A[I, J] ← I*J;)

)

για I ← 1 εως 10 επαναλαβε

(

για J ← 1 εως 10 επαναλαβε
(τυπωσε A[I, J];)τυπωσε “\n”;

)

τελος ΠΡΟΠΑΙΔΕΙΑ2

```
// όνομα αρχείου PRO2.cpp
#include <iostream>
main()
{
    int I, J, A[10][10];

    for (I=0; I<10; I++)
        for (J=0; J<10; J++)
            A[I][J] = I*J;

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

Παράδειγμα

- Θέλουμε να εισάγουμε ακέραια στοιχεία σε έναν πίνακα A διάστασης 3×5 (κατά γραμμή) και να υπολογίσουμε τον ανάστροφο πίνακα του A .
- 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

διαδικασία ΑΝΑΣΤΡδηλώση (I, J) ακερ;δηλώση (A(1:3, 1:5)) παραταξη ακερ;αρχηγια 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 $A[N]$

- 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;
```

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

$$k = 4 \rightarrow 23$$

διαδικασία ΣΤΗΛΗ

δηλώση (I, J, K, ΑΘΡ) ακερ;

δηλώση (A(1:4, 1:8)) παραταξη ακερ;

αρχη

για I ← 1 εως 4 επαναλαβε

(

για J ← 1 εως 8 επαναλαβε
(διαβασε A[I, J];)

)

διαβασε K;

ΑΘΡ ← 0;

για I ← 1 εως 4 επαναλαβε

(ΑΘΡ ← ΑΘΡ + A [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 – πλησιέστερου μέσου όρου

- Να γράψετε ένα πρόγραμμα που θα διαβάζει έναν μονοδιάστατο πίνακα πραγματικών 6 τιμών και θα επιστρέφει την τιμή του στοιχείου που είναι το πλησιέστερο στον μέσο όρο όλων των στοιχείων.

4,3	-5,2	6,4	7,1	7,7	-2,3
-----	------	-----	-----	-----	------

$$MO = 18 / 6 = 3$$

- Θα εκτελέσουμε την ίδια διαδικασία που βρίσκουμε το ελάχιστο για να βρούμε το πλησιέστερο.
- Διότι πλησιέστερο σημαίνει το στοιχείο εκείνο που η διαφορά του από τον ΜΟ είναι η μικρότερη δυνατή.
- Προσοχή: πάντα θα κρατάμε την θετική διαφορά!

διαδικασία ΠΛΗΣΙΕΣ

δηλώση (I) ακερ;

δηλώση (ΜΟ, ΑΘΡ, ΠΛ) πραγματικ;

δηλώση (Α(1:6)) παραταξη πραγματικ;

αρχη

για I ← 1 εως 6 επαναλαβε

(διαβασε A[I];)

ΑΘΡ ← 0;

για I ← 1 εως 6 επαναλαβε

(ΑΘΡ ← ΑΘΡ + Α [I];)

ΜΟ ← ΑΘΡ / 6 ;

ΠΛ ← Α[1];

για I ← 1 εως 6 επαναλαβε

(εαν |ΠΛ-ΜΟ| > |Α [I]-ΜΟ|

τοτε ΠΛ ← Α[I];)

τυπωσε ΠΛ;

τελος ΠΛΗΣΙΕΣ

```
// όνομα αρχείου SHORT.cpp
```

```
#include <iostream>
```

```
#include <cmath>
```

```
main()
```

```
{
```

```
    int I;
```

```
    float MO, SUM, PL, A[6];
```

```
    for (I=0; I < 6; I++)
```

```
        cin >> A[I];
```

```
    SUM = 0;
```

```
    for (I=0; I < 6; I++)
```

```
        SUM = SUM + A[I];
```

```
    MO = SUM / 6 ;
```

```
    PL = A[0];
```

```
    for (I=0; I < 6; I++)
```

```
        if (ABS (PL-MO) > ABS (A [I]-MO) )
```

```
            PL = A [I];
```

```
    cout << PL;
```

```
}
```

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: 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 10x10*
- **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 πολλαπλασιαστικού 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;`
- `}`

Πίνακες χαρακτήρων

- Πίνακες χαρακτήρων
 - Η αλυσίδα “Hello” είναι ένας πίνακας χαρακτήρων
 - Πίνακες χαρακτήρων μπορούν να αρχικοποιηθούν όπως όλοι οι πίνακες
 - `char string1[] = "first";`
 - `'\0'` για τέλος αλυσίδας
 - `string1` έχει 6 στοιχεία
 - Είναι το ίδιο με
 - `char string1[] = { 'f', 'i', 'r', 's', 't', '\0' };`
 - Για συγκεκριμένες θέσεις
 - `string1[3]` is character `'s'`
 - Μπορούμε να εισάγουμε πίνακες χαρακτήρων αλλά...
 - `cin >> string2; //απλή cin`
 - Διαβάζει αλυσίδες χωρίς κενά (μέχρι το πρώτο κενό)
 - Για αλυσίδες χωρίς κενά χρειάζεται η `getline` της `c++` οι άλλες συναρτήσεις ειδικά για πίνακες χαρακτήρων όπως `gets`, `puts`, ...
 - Τον τύπο `string` θα εξετάσουμε αργότερα

Παράδειγμα

- `#include <iostream>`
- `#include <cstdlib>`
- `using namespace std;`
- `// πίνακες χαρακτήρων`
- `int main(int argc, char *argv[])`
- `{ system("chcp 1253");`
- `char s1[] = { 'H','e','l','l','o','\0' };`
- `char s2[6] = { 'H','e','l','l','o','\0' };`
- `char s3[6] = { "Hello" };`
- `char s4[] = { "Hello there" };`
- `char s5[] = "Hello again";`
- `cout<<s1<<"\t"<<s2<<"\t"<<s3<<"\t"<<s4<<"\t"<<s5<<endl;`
- `cout<<"Seventh char of s5 is-> "<<s5[6]<<endl;`
- `system("PAUSE");`
- `return 0;`
- `}`