

Δείκτες και Πίνακες

Δείκτες και Πίνακες

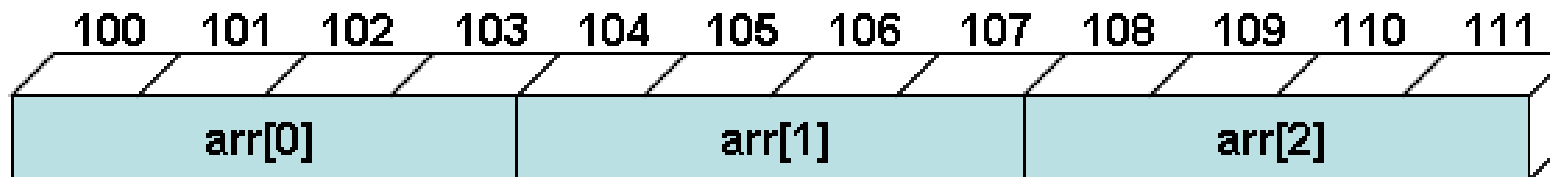
- Τα στοιχεία ενός πίνακα αποθηκεύονται σε διαδοχικές θέσεις μνήμης, με το πρώτο στοιχείο στη χαμηλότερη διεύθυνση
- Τα επόμενα στοιχεία του πίνακα αποθηκεύονται στις υψηλότερες διευθύνσεις
- Το πόσο υψηλότερα, εξαρτάται από τον τύπο δεδομένων του πίνακα (`char`, `int`, `float`, ..)
- Π.χ. σε έναν πίνακα χαρακτήρων (`char`), κάθε στοιχείο του πίνακα βρίσκεται 1 byte μετά από το προηγούμενο στοιχείο και η διεύθυνση κάθε στοιχείου είναι 1 θέση υψηλότερα από τη διεύθυνση του προηγούμενου στοιχείου
- Παρομοίως, σε έναν πίνακα ακεραίων (`int`), κάθε στοιχείο του πίνακα βρίσκεται 4 bytes μετά από το προηγούμενο στοιχείο και η διεύθυνση κάθε στοιχείου είναι 4 θέσεις υψηλότερα από τη διεύθυνση του προηγούμενου στοιχείου

Παράδειγμα

- Έστω η δήλωση του πίνακα:

```
int arr[3];
```

- Αν θεωρήσουμε ότι η διεύθυνση του πρώτου στοιχείου είναι η θέση 100 στη μνήμη, τότε η διεύθυνση του δεύτερου στοιχείου είναι η 104 και του τρίτου η 108
- Αντίστοιχα, η τιμή του πρώτου στοιχείου του πίνακα (του `arr[0]`) αποθηκεύεται στις θέσεις 100 έως και 103, η τιμή του δεύτερου στοιχείου (του `arr[1]`) στις θέσεις 104 έως και 107 και η τιμή του τρίτου στοιχείου (του `arr[2]`) στις θέσεις 108 έως και 111

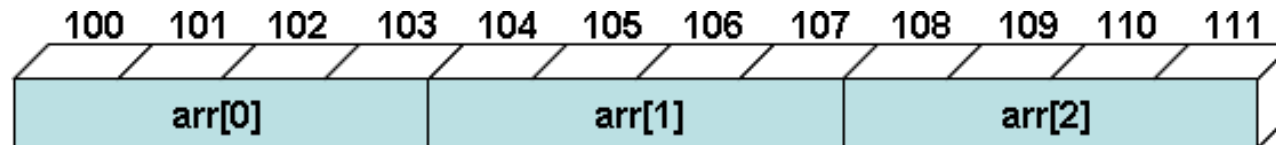


Δείκτες και Πίνακες (I)

- Ένα όνομα πίνακα χωρίς αγκύλες είναι ένας δείκτης στο πρώτο στοιχείο του πίνακα
- Με άλλα λόγια, η τιμή του ονόματος του πίνακα (χωρίς αγκύλες) ισούται με τη διεύθυνση του πρώτου στοιχείου του πίνακα
- Π.χ. αν έχει δηλωθεί ο πίνακας

```
int arr[50];
```

τότε η τιμή του `arr` είναι ίση με τη διεύθυνση του πρώτου στοιχείου του πίνακα (δηλ. ίση με `&arr[0]`) και αν η μνήμη του υπολογιστή ήταν όπως αυτή του παρακάτω σχήματος, η τιμή τους θα ήταν ίση με 100



- Συμπερασματικά, οι εκφράσεις `arr` και `&arr[0]` είναι ισοδύναμες

Δείκτες και Πίνακες (II)

- Υπενθυμίζεται από την αριθμητική δεικτών, ότι, όταν προστίθεται ένας ακέραιος αριθμός n σε έναν δείκτη, τότε ο δείκτης δείχνει σε μία νέα διεύθυνση που απέχει (σε bytes):

$n * \text{μέγεθος του τύπου}$ (στον οποίο δείχνει)

- Βάσει της λογικής αυτής, η έκφραση `arr+1` είναι ένας δείκτης που δείχνει στο δεύτερο στοιχείο του πίνακα, άρα οι εκφράσεις `arr+1` και `&arr[1]` είναι ισοδύναμες, αφού και οι δύο είναι ίσες με τη διεύθυνση του δεύτερου στοιχείου του πίνακα, κ.ο.κ.
- Δηλαδή, γενικά ισχύει ότι:

```
arr == &arr[0]
arr + 1 == &arr[1]
arr + 2 == &arr[2]
...
arr + n == &arr[n]
```

Δείκτες και Πίνακες (III)

- Αφού το όνομα ενός πίνακα είναι δείκτης στη διεύθυνση του πρώτου στοιχείου του, τότε το περιεχόμενό του θα είναι ίσο με την τιμή του πρώτου στοιχείου του
- Δηλαδή, ισχύει ότι το `*arr` είναι ίσο με `arr[0]`
- Αντίστοιχα, αφού το `arr+1` είναι δείκτης στο δεύτερο στοιχείο του πίνακα, τότε ισχύει ότι `*(arr+1)` είναι ίσο με `arr[1]`, κ.ο.κ.
- Δηλαδή, γενικά ισχύει ότι (προσοχή στις παρενθέσεις):

```
*arr == arr[0]
*(arr + 1) == arr[1]
*(arr + 2) == arr[2]
...
*(arr + n) == arr[n]
```

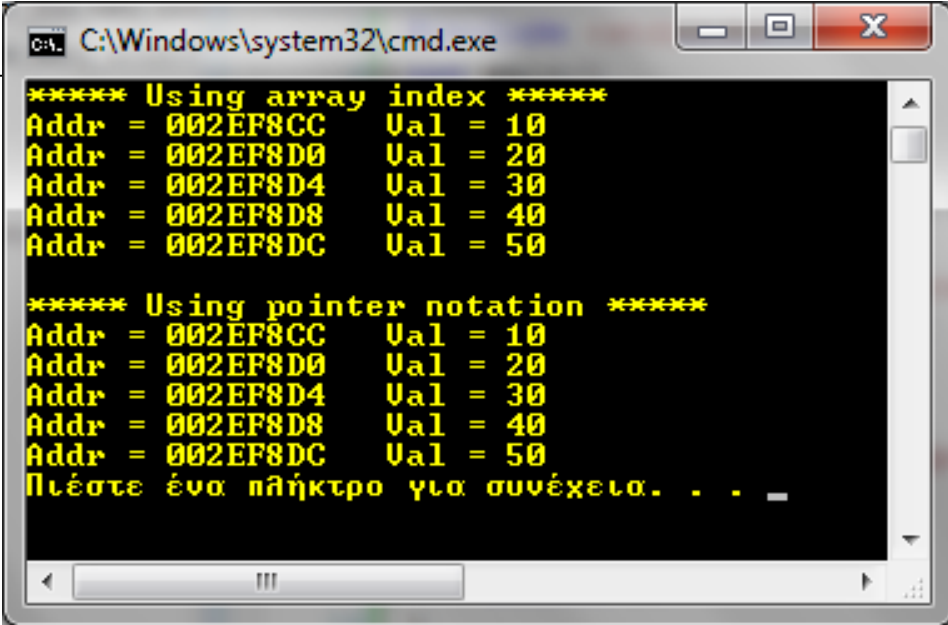
Παράδειγμα (I)

```
#include <stdio.h>
int main()
{
    int i;
    int arr[5] = {10,20,30,40,50};

    printf("***** Using array index *****\n");
    for(i = 0; i < 5; i++)
        printf("Addr = %p    Val = %d\n",&arr[i],arr[i]);

    printf("\n***** Using pointer notation *****\n");
    for(i = 0; i < 5; i++)
        printf("Addr = %p    Val = %d\n",arr+i,*(arr+i));
    return 0;
}
```

Πιθανή
Έξοδος:



```
C:\Windows\system32\cmd.exe

***** Using array index *****
Addr = 002EF8CC    Val = 10
Addr = 002EF8D0    Val = 20
Addr = 002EF8D4    Val = 30
Addr = 002EF8D8    Val = 40
Addr = 002EF8DC    Val = 50

***** Using pointer notation *****
Addr = 002EF8CC    Val = 10
Addr = 002EF8D0    Val = 20
Addr = 002EF8D4    Val = 30
Addr = 002EF8D8    Val = 40
Addr = 002EF8DC    Val = 50
Πιέστε ένα πλήκτρο για συνέχεια. . .
```

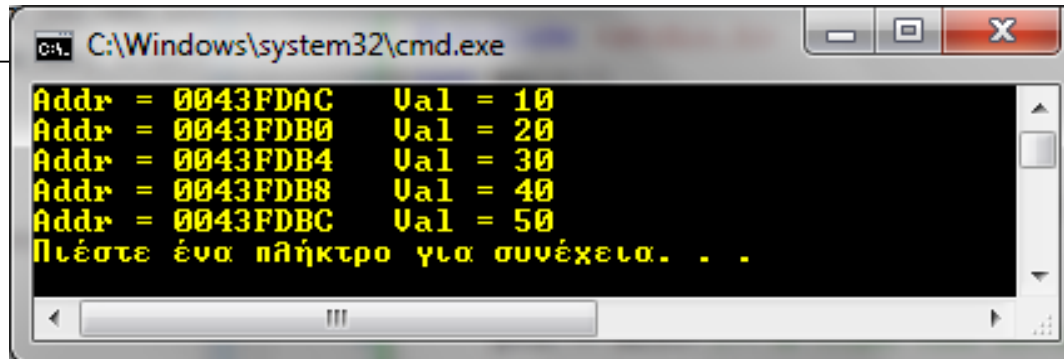
Παράδειγμα (II)

```
#include <stdio.h>
int main()
{
    int i;
    int arr[5] = {10,20,30,40,50};
    int *ptr;

    ptr = arr; /* Η τιμή του δείκτη ptr γίνεται ίση με τη
διεύθυνση του πρώτου στοιχείου του πίνακα. */

    for(i = 0; i < 5; i++)
    {
        printf("Addr = %p    Val = %d\n",ptr,*ptr);
        ptr++; /* Η τιμή του δείκτη ptr γίνεται ίση με τη
διεύθυνση του επόμενου στοιχείου του πίνακα. Ισοδύναμα, μπορούμε
να γράψουμε ptr = &arr[i]; */
    }
    return 0;
}
```

Πιθανή
Έξοδος:



```
C:\Windows\system32\cmd.exe

Addr = 0043FDAC    Val = 10
Addr = 0043FDB0    Val = 20
Addr = 0043FDB4    Val = 30
Addr = 0043FDB8    Val = 40
Addr = 0043FDBC    Val = 50
Πιέστε ένα πλήκτρο για συνέχεια. . .
```


Παρατηρήσεις (I)

- SOS!!!!!! Ποιες είναι οι κύριες διαφορές μεταξύ των δηλώσεων:

α) `int ptr[100];` και β) `int *ptr;`

1) Δεσμευμένη μνήμη?

α) Πίνακας 100 ακεραίων, άρα:

$100 * \text{sizeof}(\text{int}) = 100 * 4 = 400 \text{ bytes}$

β) Ένας “απλός” δείκτης, άρα:

$\text{sizeof}(\text{ptr}) = 4 \text{ bytes}$

2) Πού δείχνουν στη μνήμη?

α) Το όνομα του πίνακα χωρίς τις αγκύλες, δηλαδή το `ptr`, δείχνει στην αρχή αυτής της δεσμευμένης μνήμης

Η τιμή του δεν μπορεί να αλλάξει, δηλαδή **ΔΕΝ επιτρέπεται να δείξει σε κάποια άλλη διεύθυνση μνήμης**

β) Ο δείκτης `ptr` **δεν δείχνει** σε κάποια διεύθυνση μνήμης

Επομένως, πριν χρησιμοποιηθεί, πρέπει να του εκχωρηθεί μία υπαρκτή διεύθυνση, δηλαδή να δείχνει στη διεύθυνση κάποιας μεταβλητής

Μετά την αρχική εκχώρηση, η τιμή του μπορεί να αλλάξει, δηλαδή **επιτρέπεται να δείξει σε κάποια άλλη διεύθυνση μνήμης**

Παρατηρήσεις (II)

- **SOS!!!!!!**

Ποιον από τους δύο προηγούμενους τρόπους να χρησιμοποιήσετε?

α) `int arr[100];` ή β) `int *arr;`

- Για τον χειρισμό των στοιχείων ενός πίνακα προτείνεται το πρώτο από τα παραπάνω είδη σύνταξης, στο οποίο αποτυπώνεται **εμφανώς η θέση του στοιχείου** στον πίνακα και **όχι η σύνταξη με τη σημειογραφία δείκτη**, γιατί ο κώδικας του προγράμματος είναι περισσότερο ευανάγνωστος και λιγότερο επιρρεπής σε λάθη
- Δηλαδή, το `arr[i]` είναι πιο κατανοητό και πιο ασφαλές από το `*(arr+i)`
Π.χ. αν ξεχάσουμε π.χ. τις παρενθέσεις, τότε εισάγεται ένα σφάλμα στον κώδικα (bug) που είναι δύσκολο να εντοπιστεί

Παρατηρήσεις (III)

- Αν ένας δείκτης δείχνει σε κάποιο στοιχείο ενός πίνακα, μπορούμε να χρησιμοποιήσουμε τον δείκτη σαν πίνακα
- Ωστόσο, να θυμάστε ότι, παρά τη στενή σχέση δεικτών και πινάκων, ένας δείκτης δεν είναι πίνακας

Για παράδειγμα, το επόμενο πρόγραμμα εμφανίζει τα στοιχεία ενός πίνακα χρησιμοποιώντας τον δείκτη `ptr` σαν να ήταν πίνακας

```
#include <stdio.h>
int main()
{
    int* ptr;
    int i, arr[5] = {10, 20, 30, 40, 50};

    ptr = arr; /* Η τιμή του δείκτη ptr γίνεται ίση με τη
διεύθυνση του 1ου στοιχείου του πίνακα. */

    for(i = 0; i < 5; i++)
        printf("%d\n", ptr[i]); /* Χρήση του δείκτη ptr με τη
μορφή πίνακα. */
    return 0;
}
```

Παραδείγματα (I)

- Ποια είναι η έξοδος του παρακάτω προγράμματος ???

```
#include <stdio.h>
int main()
{
    int* ptr;
    int arr[] = {10,20,30,40,50};

    ptr = arr;
    printf("Val1 = %d Val2 = %d\n", *ptr+2, *(ptr+2));
    return 0;
}
```

Έξοδος: Val1 = 12 Val2 = 30

Παραδείγματα (II)

- Ποια είναι η έξοδος του παρακάτω προγράμματος ???

```
#include <stdio.h>
int main()
{
    int* ptr;
    int i,j,arr[] = {10,20,30,40,50};

    ptr = arr;
    *ptr = 3;

    ptr += 2;
    *ptr = 5;

    printf("Val = %d\n",arr[0] + arr[2] + arr[4]);
    return 0;
}
```

Έξοδος: Val = 58

Παραδείγματα (III)

- Ποια είναι η έξοδος του παρακάτω προγράμματος ???

```
#include <stdio.h>
int main()
{
    int* ptr1, *ptr2;
    int arr[] = {10,20,30,40,50};

    ptr1 = &arr[0];
    ptr2 = &arr[3];

    printf("%d\n",ptr2-ptr1);
    return 0;
}
```

Έξοδος: 3

Παραδείγματα (IV)

- Ποια είναι η έξοδος του παρακάτω προγράμματος ???

```
#include <stdio.h>
int main()
{
    int* ptr;
    int i,arr[5] = {10,20,30,40,50};

    ptr = arr+2;
    for(i = 0; i < 5; i++)
        printf("%d ",ptr[i]);
    return 0;
}
```

Έξοδος: 30 40 50 (και δύο τυχαίες τιμές)

Παραδείγματα (V)

- Υπάρχει κάποιο bug στο παρακάτω πρόγραμμα ???

```
#include <stdio.h>
int main()
{
    int i,arr[5] = {10,20,30,40,50};

    printf("%d\n",0[arr]);
    printf("%d\n",2[arr]);
    printf("%d\n",4[arr]);
    return 0;
}
```

Απάντηση: Όχι...

«σκονάκι No1»: θυμηθείτε ότι $*(arr+i) = arr[i]$

Κι άλλη υπόδειξη???

«σκονάκι No2»: $*(arr+i) = *(i+arr)$

Κι άλλο???

«σκονάκι No3»: $arr[i] = *(arr+i) = *(i+arr) = i[arr]$

Πίνακας Δεικτών

- Ένας πίνακας δεικτών είναι ένας πίνακας, όπου κάθε στοιχείο του είναι ένας δείκτης σε έναν συγκεκριμένο τύπο δεδομένων
- Για να δηλώσουμε έναν πίνακα δεικτών χρησιμοποιούμε τον τελεστή * πριν από το όνομα του πίνακα

Π.χ.

```
int *array[10];
```

Δήλωση ενός πίνακα δεικτών με όνομα `array`, ο οποίος περιέχει 10 στοιχεία και το καθένα από αυτά είναι ένας δείκτης σε μία ακέραια μεταβλητή (`int`)

Π.χ.

```
char *arr[5];
```

Δήλωση ενός πίνακα δεικτών με όνομα `arr`, ο οποίος περιέχει 5 στοιχεία και το καθένα από αυτά είναι ένας δείκτης σε έναν χαρακτήρα (`char`)

Παρατηρήσεις

- Όταν δηλώνεται ένας πίνακας δεικτών, το όνομα του πίνακα δεν πρέπει να περικλείεται σε παρενθέσεις
- Π.χ. με τη δήλωση:

```
int (*arr) [3];
```

η μεταβλητή `arr` δηλώνεται ως δείκτης προς έναν πίνακα τριών ακεραίων και όχι σαν πίνακας τριών δεικτών

Παράδειγμα

- Ποια είναι η έξοδος του παρακάτω προγράμματος ???

```
#include <stdio.h>
int main()
{
    int i, pin[3] = {10, 20, 30};
    int* arr[3];

    for(i = 0; i < 3; i++)
    {
        arr[i] = &pin[i];
        printf("%d ", *arr[i]);
    }
    return 0;
}
```

Έξοδος: 10 20 30

Δείκτης σε Δείκτη

- Όταν δηλώνεται ένας δείκτης, ο μεταγλωττιστής, όπως κάνει για οποιαδήποτε μεταβλητή, δεσμεύει τις απαραίτητες θέσεις μνήμης για να αποθηκεύσει την τιμή του
- Επομένως, αφού έχει δεσμευτεί μία διεύθυνση μνήμης για έναν δείκτη μπορούμε να δηλώσουμε έναν άλλον δείκτη που να δείχνει σε αυτή τη διεύθυνση
- Για να δηλώσουμε έναν δείκτη σε κάποιον άλλον δείκτη χρησιμοποιούμε δύο φορές τον τελεστή *
- Παραδείγματα Δηλώσεων «Δείκτη σε Δείκτη»

```
int** ptr; /* Η μεταβλητή ptr δηλώνεται σαν δείκτης προς  
κάποιον άλλον δείκτη, ο οποίος με τη σειρά του δείχνει στη  
διεύθυνση μίας ακέραιας μεταβλητής. */
```

```
char** ptr; /* Η μεταβλητή ptr δηλώνεται σαν δείκτης προς  
κάποιον άλλον δείκτη, ο οποίος με τη σειρά του δείχνει στη  
διεύθυνση μίας μεταβλητής χαρακτήρα. */
```

Χρήση «Δείκτη σε Δείκτη»

- Αν έχουμε δηλώσει έναν δείκτη σε έναν δεύτερο δείκτη, τότε με τον τελεστή * έχουμε πρόσβαση στη διεύθυνση του δεύτερου δείκτη και με τον διπλό τελεστή ** έχουμε πρόσβαση στη μεταβλητή που δείχνει ο δεύτερος δείκτης

```
#include <stdio.h>
int main()
{
    int i = 20;
    int* ptr1;
    int** ptr;
    ptr1 = &i;
    ptr = &ptr1;

    printf("Value = %d\n", **ptr);
    return 0;
}
```

Δήλωση Μεταβλητής (**int**)

Δήλωση Δείκτη (**ptr1**) σε **int**

Δήλωση Δείκτη (**ptr**) σε Δείκτη
(που δείχνει σε **int**)

Ο δείκτης **ptr1** δείχνει στον **i**

Ο δεύτερος δείκτης **ptr** δείχνει
στον πρώτο δείκτη **ptr1**

****ptr = * (ptr1) = i**

Παράδειγμα

- Ποια είναι η έξοδος του παρακάτω προγράμματος ???

```
#include <stdio.h>
int main()
{
    int i = 20;
    int* ptr1;
    int** ptr;

    ptr1 = &i;
    ptr = &ptr1;

    **ptr += 100;
    printf("%d %d %d\n", **ptr, *ptr1, i);
    return 0;
}
```

Έξοδος: 120 120 120

Δείκτες και Διδιάστατοι Πίνακες (I)

- Όπως και στους μονοδιάστατους πίνακες, έτσι και στους πολυδιάστατους, η κάθε διάσταση δηλώνεται μέσα σε αγκύλες []
- Π.χ. με την εντολή: `int arr[2][3];`
δηλώνεται ένας διδιάστατος πίνακας, ο οποίος αποτελείται από 2 γραμμές και 3 στήλες (δηλ. συνολικά περιέχει 6 ακέραιες μεταβλητές) και σχηματικά απεικονίζεται όπως παρακάτω

<code>arr[0][0]</code>	<code>arr[0][1]</code>	<code>arr[0][2]</code>
<code>arr[1][0]</code>	<code>arr[1][1]</code>	<code>arr[1][2]</code>

- Τα στοιχεία του πίνακα αποθηκεύονται σε διαδοχικές θέσεις στη μνήμη ξεκινώντας από τα στοιχεία της 1η γραμμής, συνεχίζοντας με τα στοιχεία της 2ης γραμμής, κ.ο.κ.
- Άρα, η σειρά αποθήκευσης των στοιχείων του παραπάνω πίνακα `arr` στη μνήμη είναι: `arr[0][0]`, `arr[0][1]`, `arr[0][2]`, `arr[1][0]`, `arr[1][1]` και `arr[1][2]`

Δείκτες και Διδιάστατοι Πίνακες (II)

- Για να χειριστούμε έναν διδιάστατο πίνακα με χρήση δεικτών, έστω `arr[N][M]`, μπορούμε να θεωρήσουμε ότι ο πίνακας `arr` αποτελείται από έναν πίνακα δεικτών `N` στοιχείων, `arr[0]`, `arr[1]`, ..., `arr[N-1]`, όπου καθένα από αυτά είναι δείκτης σε έναν πίνακα `M` στοιχείων

- Π.χ. με την εντολή:

```
int arr[2][3];
```

το `arr[0]` μπορεί να χρησιμοποιηθεί σαν δείκτης προς έναν πίνακα 3 ακεραίων που περιέχει τα στοιχεία της πρώτης γραμμής, δηλαδή τα `arr[0][0]`, `arr[0][1]` και `arr[0][2]`

- Συγκεκριμένα, το `arr[0]` είναι δείκτης στο πρώτο στοιχείο του πίνακα, δηλαδή στο `arr[0][0]`
- Άρα, η τιμή του `*arr[0]` είναι ίση με το `arr[0][0]`

Δείκτες και Διδιάστατοι Πίνακες (III)

- Επίσης, σύμφωνα με την αριθμητική δεικτών:
 - το `arr[0]+1` είναι δείκτης στο δεύτερο στοιχείο του πίνακα, δηλαδή στο `arr[0][1]`
 - το `arr[0]+2` είναι δείκτης στο τρίτο στοιχείο του πίνακα, δηλαδή στο `arr[0][2]`, κ.ο.κ.
 - ...
 - συνεπώς, στη γενική περίπτωση ισχύει ότι το `arr[0]+κ` είναι δείκτης στο στοιχείο `arr[0][κ]` της πρώτης γραμμής του διδιάστατου πίνακα
- Δηλαδή, ισχύει ότι:
 - το `arr[0]+κ` είναι ισοδύναμο με `&arr[0][κ]`
 - η τιμή του `* (arr[0]+κ)` είναι ίση με `arr[0][κ]`

Δείκτες και Διδιάστατοι Πίνακες (IV)

- Αντίστοιχα, το `arr[1]` μπορεί να χρησιμοποιηθεί σαν δείκτης προς έναν πίνακα 3 ακεραίων που περιέχει τα στοιχεία της δεύτερης γραμμής, δηλαδή τα `arr[1][0]`, `arr[1][1]` και `arr[1][2]`
- Συγκεκριμένα:
 - το `arr[1]` είναι δείκτης στο πρώτο στοιχείο του πίνακα, δηλαδή στο `arr[1][0]`
 - Άρα, η τιμή του `*arr[1]` είναι ίση με το `arr[1][0]`
- Παρομοίως με πριν, ισχύει ότι το `arr[1]+κ` είναι δείκτης στο στοιχείο `arr[1][κ]` της δεύτερης γραμμής του διδιάστατου πίνακα
- Δηλαδή, ισχύει ότι:
 - το `arr[1]+κ` είναι ισοδύναμο με `&arr[1][κ]`
 - η τιμή του `*(arr[1]+κ)` είναι ίση με `arr[1][κ]`

Δείκτες και Διδιάστατοι Πίνακες (V)

- Γενικά, θεωρούμε ότι τα στοιχεία ενός πίνακα $\text{arr}[N][M]$, είναι τα $\text{arr}[0], \text{arr}[1], \dots, \text{arr}[N-1]$ τα οποία είναι **δείκτες σε πίνακες** που περιέχουν M στοιχεία της αντίστοιχης γραμμής

- Δηλαδή,
 - το **πρώτο** στοιχείο του πίνακα $\text{arr}[N][M]$ είναι το $\text{arr}[0]$, το οποίο είναι δείκτης σε έναν πίνακα που περιέχει τα M στοιχεία της **πρώτης** γραμμής
 - το **δεύτερο** στοιχείο του πίνακα $\text{arr}[N][M]$ είναι το $\text{arr}[1]$, το οποίο είναι δείκτης σε έναν πίνακα που περιέχει τα M στοιχεία της **δεύτερης** γραμμής
 - ...
 - ενώ το **τελευταίο** στοιχείο είναι το $\text{arr}[N-1]$, το οποίο είναι δείκτης σε έναν πίνακα που περιέχει τα M στοιχεία της **τελευταίας** (της N -οστής) γραμμής

Παράδειγμα

- Τι κάνει το παρακάτω πρόγραμμα ???

```
#include <stdio.h>
int main()
{
    int i,k,arr[2][3] = {10,20,30,40,50,60};

    for(i = 0; i < 2; i++)
        for(k = 0; k < 3; k++)
            printf("Value of [%d][%d] element is: %d\n",
i,k,*(arr[i] + k));
    return 0;
}
```

Εμφανίζει τις τιμές όλων των στοιχείων του πίνακα με
χρήση δείκτη !!!

Χειρισμός Διδιάστατου Πίνακα με «δείκτη σε δείκτη» (I)

- Ένας εναλλακτικός τρόπος για να διαχειριστούμε έναν διδιάστατο πίνακα με χρήση δείκτη, είναι χρησιμοποιώντας **το όνομα του πίνακα**
- Θυμηθείτε ότι **το όνομα** ενός πίνακα χωρίς τις αγκύλες είναι ισοδύναμο με **τη διεύθυνση του πρώτου στοιχείου** του πίνακα
- Π.χ. αν θεωρήσουμε την παρακάτω δήλωση:

`int arr[2][3];`

το όνομα του πίνακα `arr` είναι δείκτης στο πρώτο στοιχείο του πίνακα, δηλ. στο `arr[0]`
- Όμως, όπως είδαμε προηγουμένως, το πρώτο στοιχείο του πίνακα (το `arr[0]`) είναι με τη σειρά του δείκτης σε έναν πίνακα που περιέχει τα 3 στοιχεία της **πρώτης** γραμμής
- Συγκεκριμένα, το `arr[0]` είναι δείκτης στο πρώτο στοιχείο του πίνακα, δηλαδή στο `arr[0][0]`

Χειρισμός Διδιάστατου Πίνακα με «δείκτη σε δείκτη» (II)

- Άρα, ισχύει ότι το `arr` είναι δείκτης στο `arr[0]` και το `arr[0]` είναι δείκτης στο `arr[0][0]`
- Επομένως, πώς μπορούμε να χειριστούμε το όνομα του πίνακα `arr` ???
- Η απάντηση είναι ότι μπορούμε να το χειριστούμε σαν **δείκτη προς δείκτη**
- Π.χ. αφού το `arr` είναι δείκτης σε έναν δείκτη που δείχνει στη διεύθυνση του στοιχείου `arr[0][0]`, τότε το `**arr` είναι ίσο με την τιμή `arr[0][0]`
- Παρομοίως, το `arr+1` είναι δείκτης στο `arr[1]` και το `arr[1]` είναι δείκτης που δείχνει στη διεύθυνση του στοιχείου `arr[1][0]`
- Άρα, το `** (arr+1)` είναι ίσο με την τιμή `arr[1][0]`
- Στη γενική περίπτωση, ισχύει ότι:
 - το `arr+k` είναι ισοδύναμο με `&arr[k]`
 - το `* (arr+k)` είναι ισοδύναμο με `arr[k]`, δηλαδή με `&arr[k][0]`
 - το `** (arr+k)` είναι ισοδύναμο με `arr[k][0]`

Παράδειγμα

- Τι κάνει το παρακάτω πρόγραμμα ???

```
#include <stdio.h>
int main()
{
    int i,k,arr[2][3] = {10,20,30,40,50,60};

    for(i = 0; i < 2; i++)
        for(k = 0; k < 3; k++)
            printf("Value of [%d][%d] element is: %d\n",
i,k,*(*(arr + i) + k));
    return 0;
}
```

Εμφανίζει τις τιμές όλων των στοιχείων του πίνακα με χρήση του ονόματος του πίνακα ως «δείκτη σε δείκτη» !!!

Παρατηρήσεις

- Προφανώς, η διαχείριση των στοιχείων ενός διδιάστατου πίνακα με χρήση δεικτών (είτε απλού δείκτη είτε «δείκτη σε δείκτη») οδηγεί σε δυσνόητο και μη ευανάγνωστο κώδικα
- Για τον λόγο αυτό, προτείνουμε η διαχείριση των στοιχείων να γίνεται με τη χρήση των αγκυλών [] [] και των αντιστοίχων θέσεων στον πίνακα