Δ.Π.Θ Δομημένος Προγραμματισμός

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ & ΔΙΟΙΚΗΣΗΣ

Πολυδιάστατοι Πίνακες

Dr. Αθανάσιος Μπαλαφούτης
Εργαστηριακό Διδακτικό Προσωπικό
Τομέας Συστημάτων Παραγωγής
Εργαστήριο Ρομποτικής και Αυτοματισμών
abalafou@pme.duth.gr
Γραφείο 304, τηλ.: 25410 – 79892

Δήλωση πίνακα 1D

Υπενθύμιση: Δήλωση μονοδιάστατου πίνακα

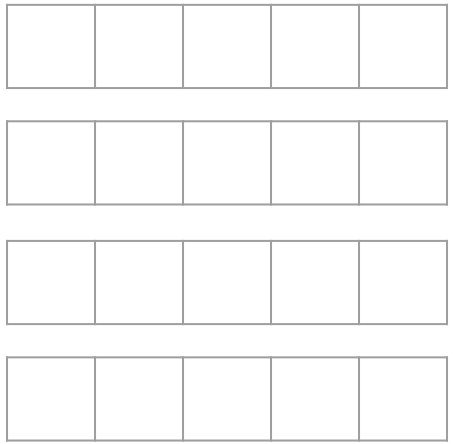
```
int arr[5];
```

Δήλωση πίνακα 2D

Ένας πίνακας 2 διαστάσεων (2D), ουσιαστικά είναι ένας πίνακας από 1D πίνακες.

```
int arr[4][5];
```

arr[4][5]



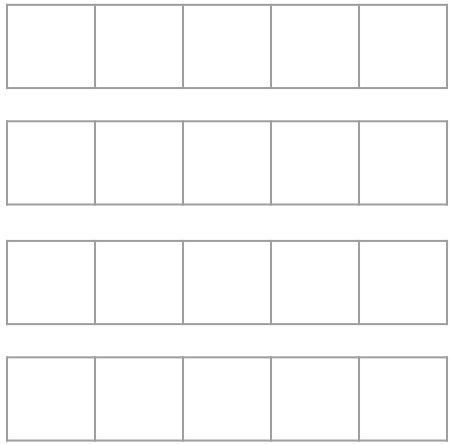
Δήλωση πίνακα 2D

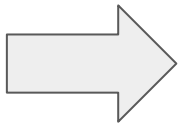
Ένας πίνακας 2 διαστάσεων (2D), ουσιαστικά είναι ένας πίνακας από 1D πίνακες.

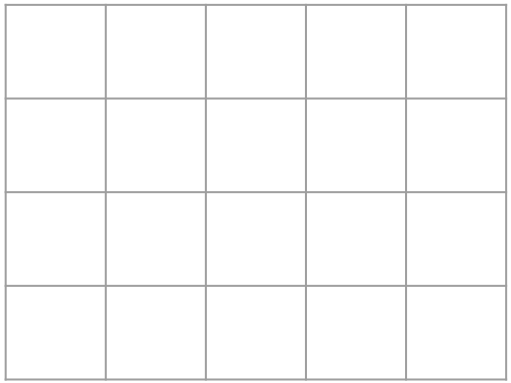
```
int arr[4][5];
```

arr[4][5] (4 γραμμές - 5 στήλες)

arr[4][5]







Αρχικοποίηση πίνακα 2D

1η Μέθοδος:

```
int arr[2][3] = {1, 2, 3, 4, 5, 6};
```

1	2	3
4	5	6

Αρχικοποίηση πίνακα 2D

2η Μέθοδος:

```
int arr[2][3] = {{1, 2, 3}, {4, 5, 6}};
```

1	2	3
4	5	6

Πρόσβαση σε στοιχείο του πίνακα 2D


```
arr[0][1] = 7;
```

1	7	3
4	5	6

Εκτύπωση πίνακα 2D


```c
#include <stdio.h>

int main(){
    int arr[2][3] = {{1, 2, 3}, {4, 5, 6}};

    for(int i = 0; i < 2; i++){
        for(int j = 0; j < 3; j++){
            printf("%d ", arr[i][j]);
        }
    }
    return 0;
}
```

Έξοδος:

```
1 2 3 4 5 6
```

Εκτύπωση πίνακα 2D


```c
#include <stdio.h>

int main(){
    int arr[2][3] = {{1, 2, 3}, {4, 5, 6}};

    for(int i = 0; i < 2; i++){
        for(int j = 0; j < 3; j++){
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Έξοδος:

```
1 2 3
4 5 6
```

Εκτύπωση πίνακα 2D


```c
#include <stdio.h>

int main(){
    int arr[2][3] = {{1, 2, 3}, {4, 5, 6}};

    for(int i = 0; i < 2; i++){
        for(int j = 0; j < 3; j++){
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Έξοδος:

```
1 2 3
4 5 6
```

Αν ήθελα να εκτυπώσω τον πίνακα ανά στήλες;

Έξοδος:

```
1 4
2 5
3 6
```

Εκτύπωση πίνακα 2D


```c
#include <stdio.h>

int main(){
    int arr[2][3] = {{1, 2, 3}, {4, 5, 6}};

    for(int j = 0; j < 3; j++){
        for(int i = 0; i < 2; i++){
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

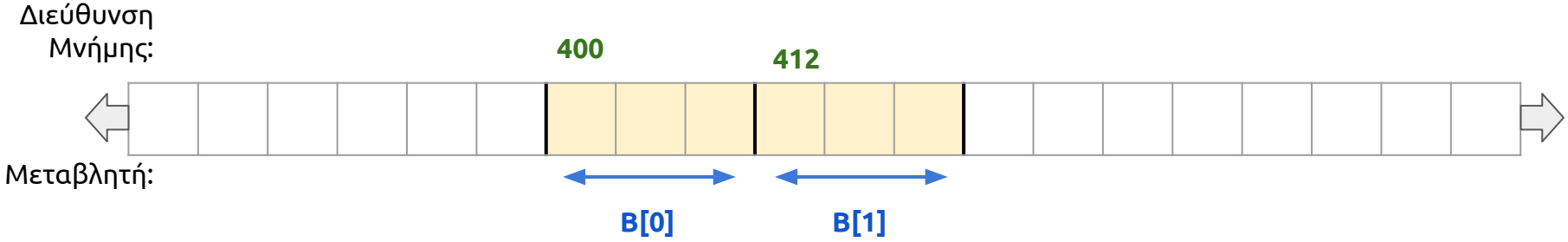
Αν ήθελα να εκτυπώσω τον πίνακα ανά στήλες;

Έξοδος:

```
1 4
2 5
3 6
```

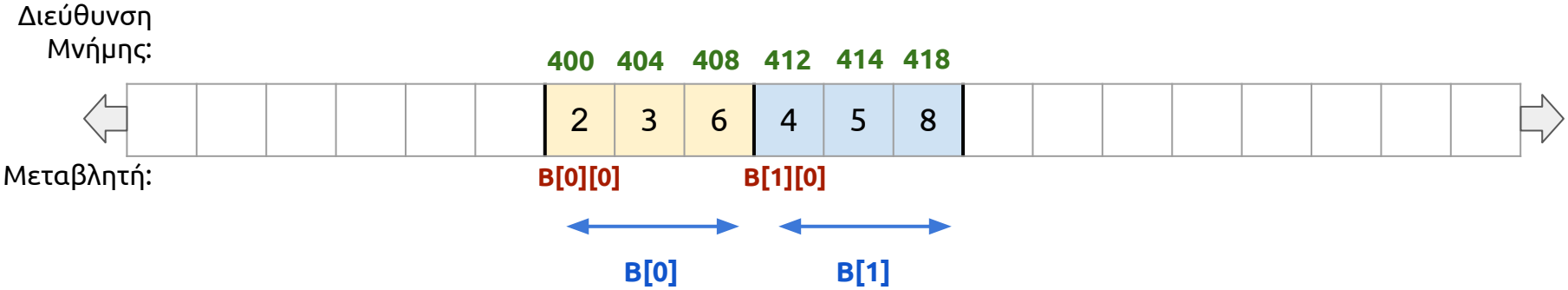
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>

int main(){
    int B[2][3];

}
```



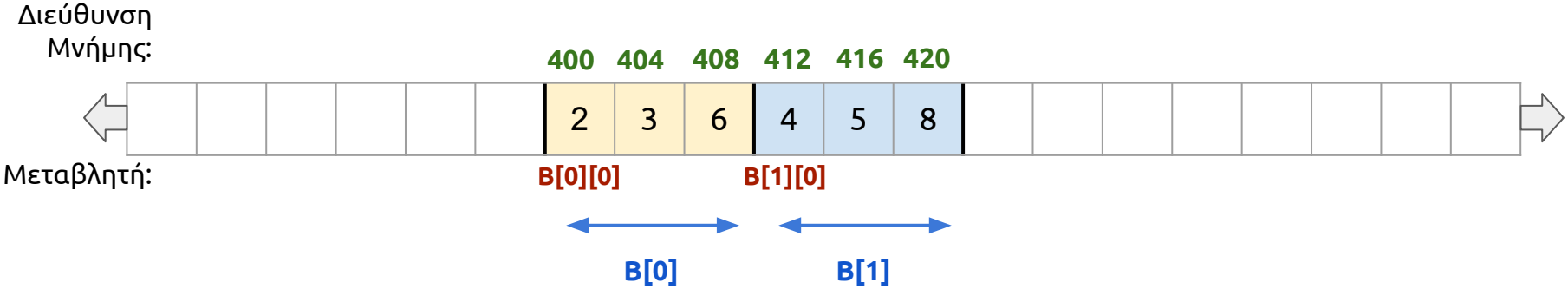
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>

int main(){
    int B[2][3] = { {2,3,6}, {4,5,8} };

}
```



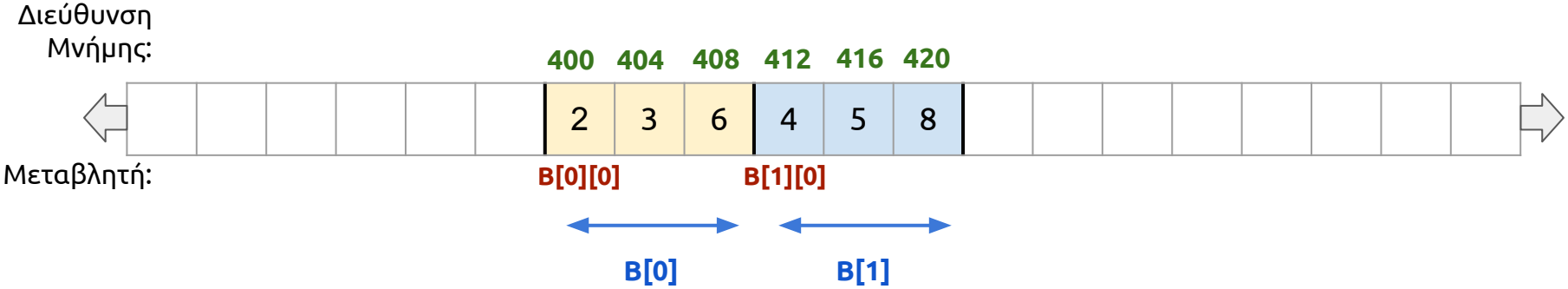
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {2,3,6}, {4,5,8} };
    printf("%d \n",B);       // ???



}
```



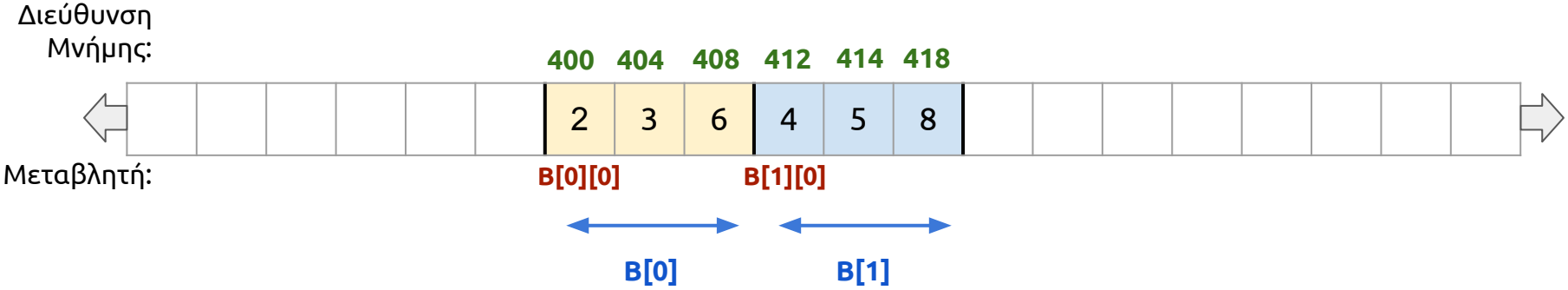
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {2,3,6}, {4,5,8} };
    printf("%d \n",B);       // 400
    printf("%d \n", &B[0]); // ???



}
```



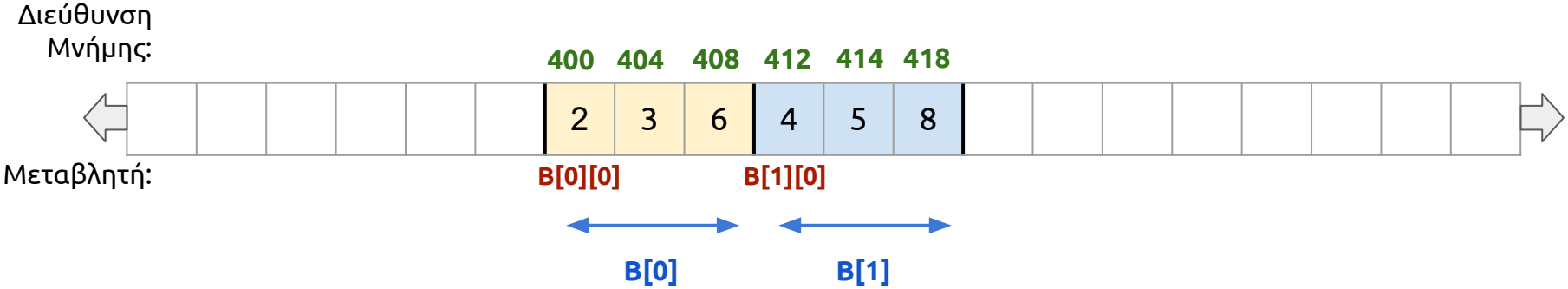
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {2,3,6}, {4,5,8} };
    printf("%d \n",B);       // 400
    printf("%d \n", &B[0]); // 400
    printf("%d \n", *B);     // ???


}
```



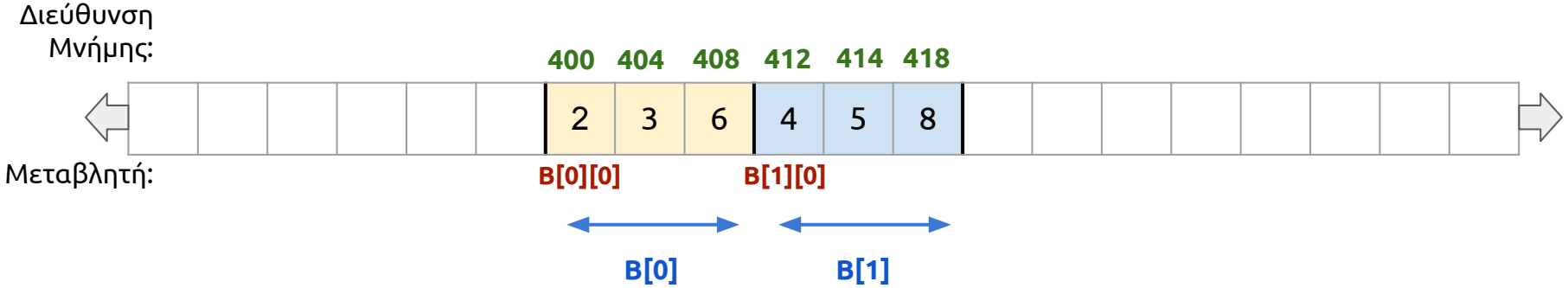
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {2,3,6}, {4,5,8} };
    printf("%d \n",B);       // 400
    printf("%d \n", &B[0]); // 400
    printf("%d \n", *B);     // 400
    printf("%d \n", B[0]);   // ???

}
```



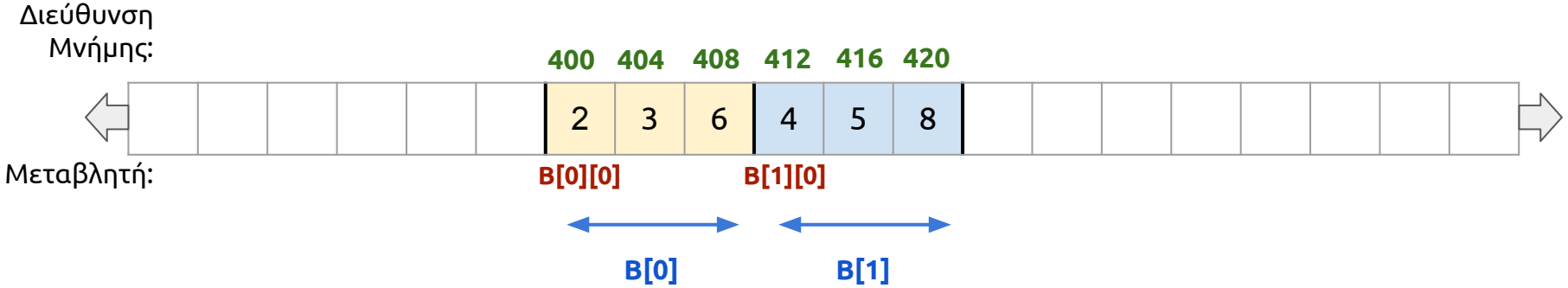
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {2,3,6}, {4,5,8} };
    printf("%d \n",B);       // 400
    printf("%d \n", &B[0]); // 400
    printf("%d \n", *B);     // 400
    printf("%d \n", B[0]);  // 400
    printf("%d \n", &B[0][0]);   // ???
}
```



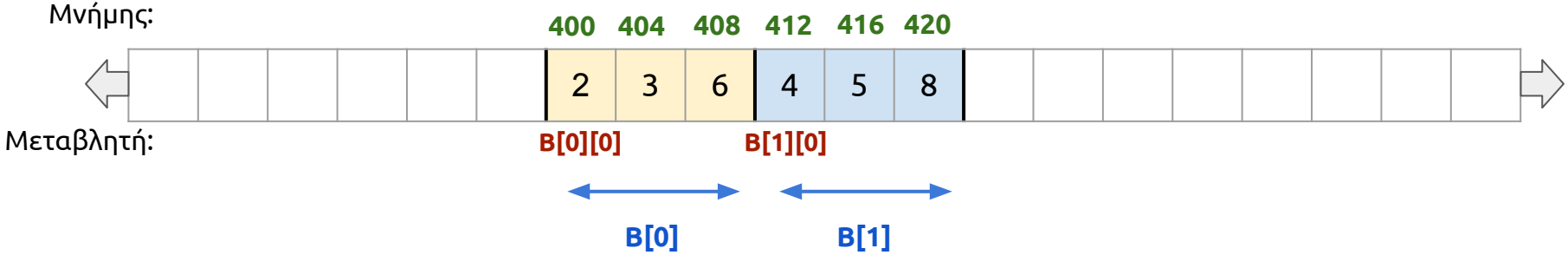
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {2,3,6}, {4,5,8} };
    printf("%d \n",B);       // 400
    printf("%d \n", &B[0]); // 400
    printf("%d \n", *B);     // 400
    printf("%d \n", B[0]);  // 400
    printf("%d \n", &B[0][0]);   // 400
}
```



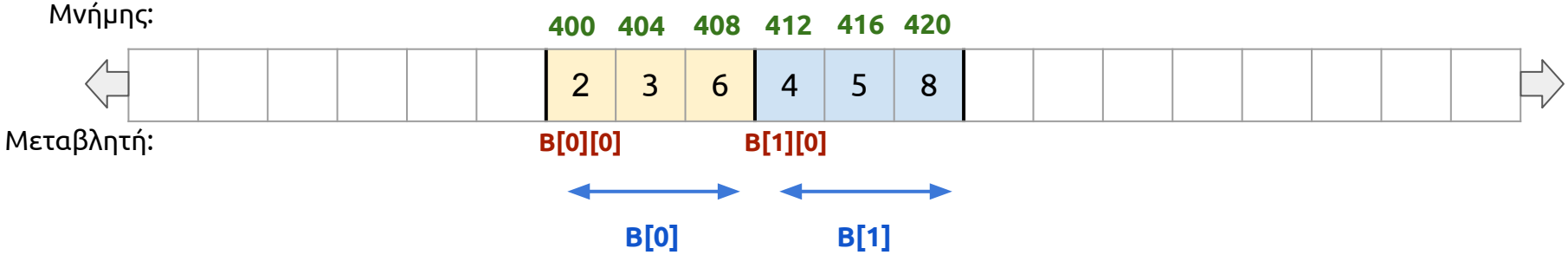
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {2,3,6}, {4,5,8} };
    printf("%d \n",B);       // 400
    printf("%d \n", B + 1); // ???



}
```



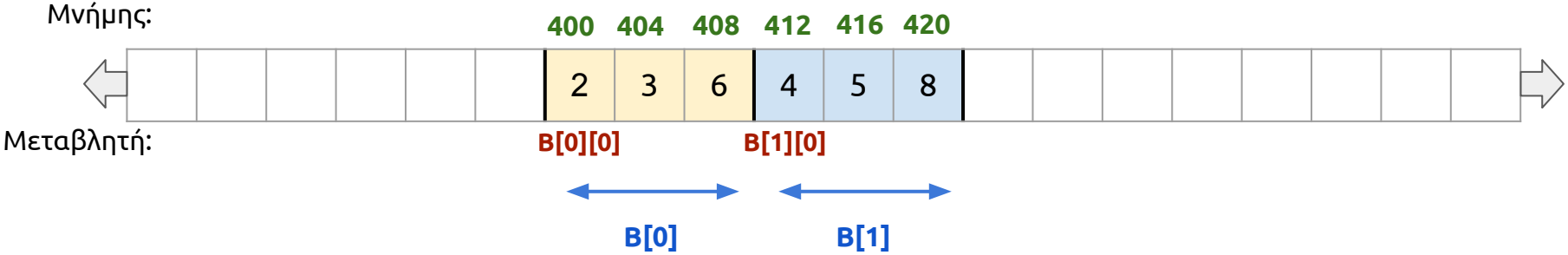
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {2,3,6}, {4,5,8} };
    printf("%d \n",B);       // 400
    printf("%d \n", B + 1); // 400 + 12 = 412
    printf("%d \n", &B[1]); // ???



}
```



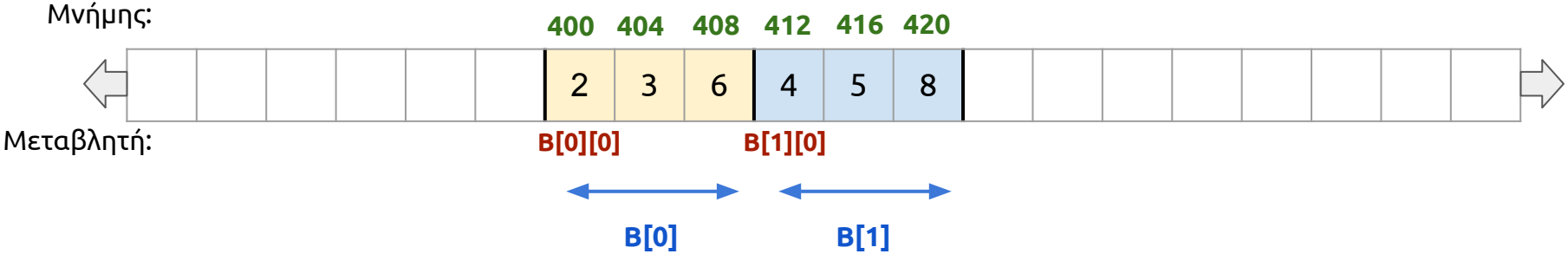
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {2,3,6}, {4,5,8} };
    printf("%d \n",B);       // 400
    printf("%d \n", B + 1); // 400 + 12 = 412
    printf("%d \n", &B[1]); // 400 + 12 = 412
    printf("%d \n", *(B + 1));   // ???


}
```



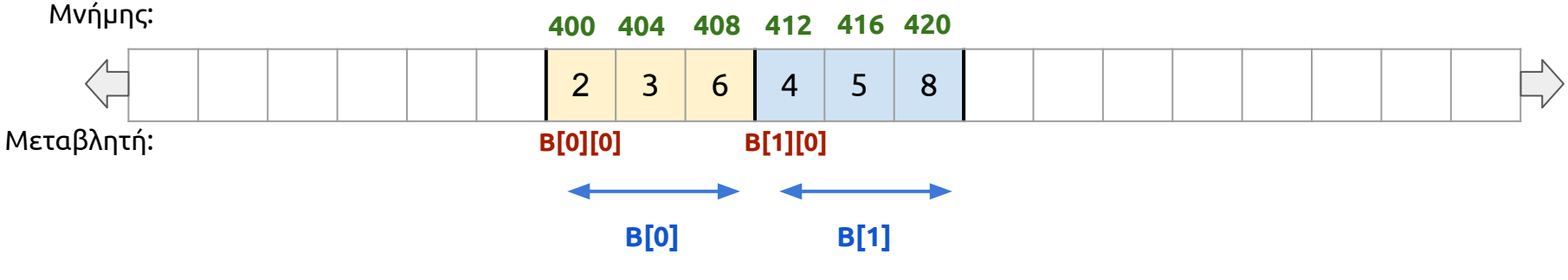
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {2,3,6}, {4,5,8} };
    printf("%d \n",B);       // 400
    printf("%d \n", B + 1); // 400 + 12 = 412
    printf("%d \n", &B[1]); // 400 + 12 = 412
    printf("%d \n", *(B + 1));   // 400 + 12 = 412
    printf("%d \n", B[1]);       // ???

}
```



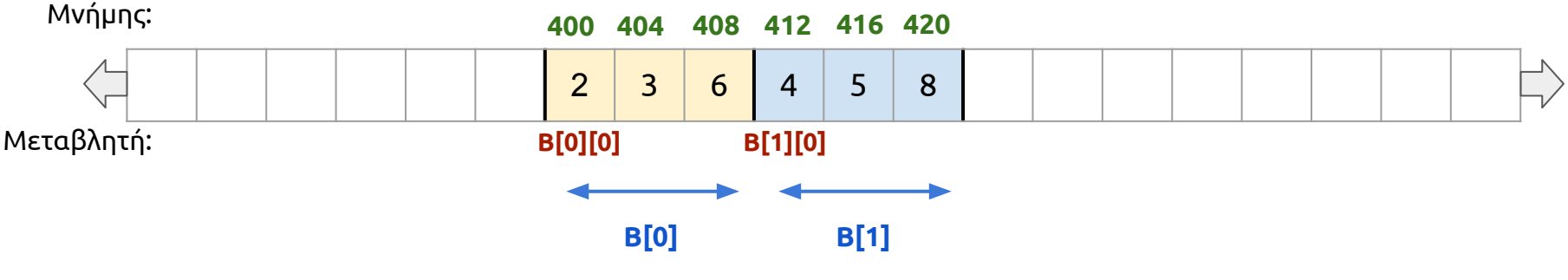
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {2,3,6}, {4,5,8} };
    printf("%d \n",B);       // 400
    printf("%d \n", B + 1); // 400 + 12 = 412
    printf("%d \n", &B[1]); // 400 + 12 = 412
    printf("%d \n", *(B + 1));   // 400 + 12 = 412
    printf("%d \n", B[1]);       // 412
    printf("%d \n", &B[1][0]);   // ???
}
```



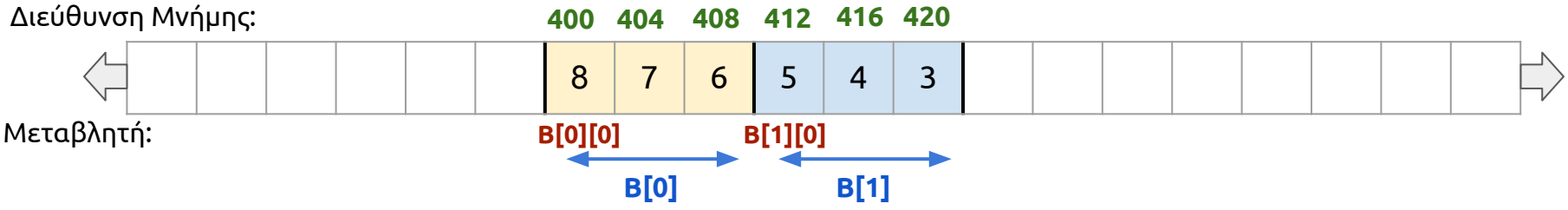
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {2,3,6}, {4,5,8} };
    printf("%d \n",B);       // 400
    printf("%d \n", B + 1); // 400 + 12 = 412
    printf("%d \n", &B[1]); // 400 + 12 = 412
    printf("%d \n", *(B + 1));   // 400 + 12 = 412
    printf("%d \n", B[1]);       // 412
    printf("%d \n", &B[1][0]);   // 412
}
```



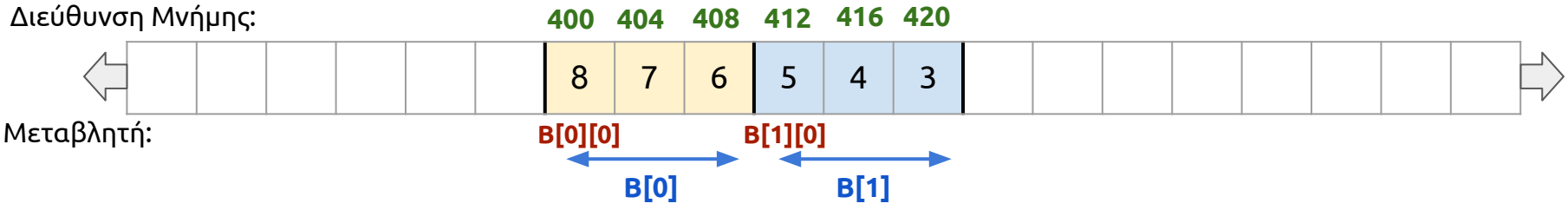
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {8,7,6}, {5,4,3} };
    printf("%d \n",*(B + 1) + 2);     // ???



}
```



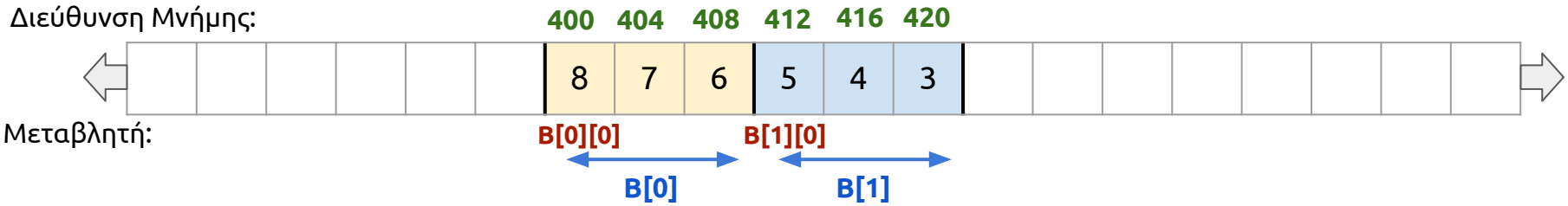
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {8,7,6}, {5,4,3} };
    printf("%d \n",*(B + 1) + 2);     // 412 + 8 = 420
    printf("%d \n",**B);     // ???



}
```



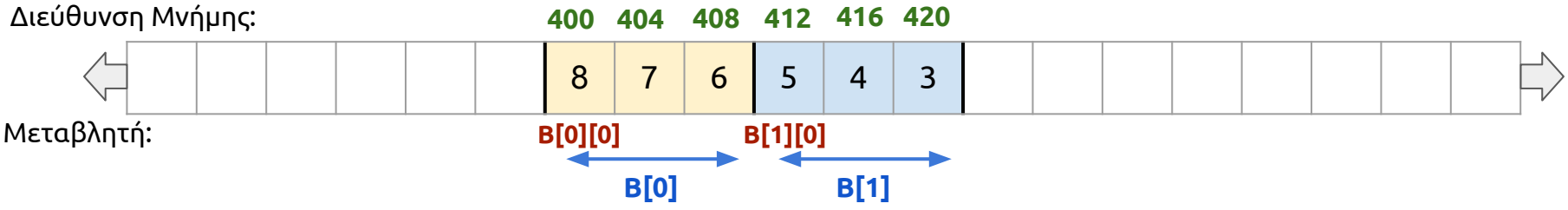
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {8,7,6}, {5,4,3} };
    printf("%d \n",*(B + 1) + 2);     // 412 + 8 = 420
    printf("%d \n",**B);     // *400 =  8
    printf("%d \n",**B+1);   // ???



}
```



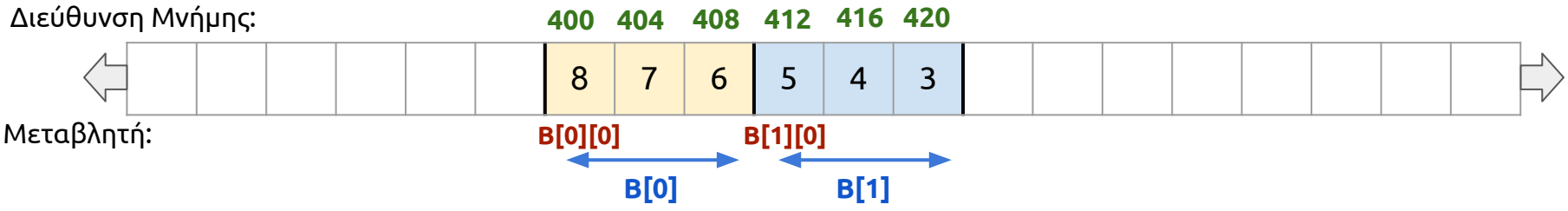
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {8,7,6}, {5,4,3} };
    printf("%d \n",*(B + 1) + 2);     // 412 + 8 = 420
    printf("%d \n",**B);     // *400 =  8
    printf("%d \n",**B+1);   // *400 + 1 =  9
    printf("%d \n",*(*B + 1));      // ???



}
```



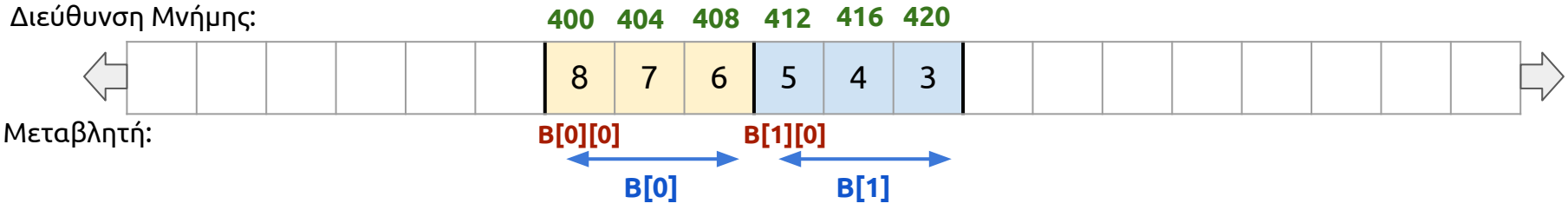
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {8,7,6}, {5,4,3} };
    printf("%d \n",*(B + 1) + 2);     // 412 + 8 = 420
    printf("%d \n",**B);     // *400 =  8
    printf("%d \n",**B+1);   // *400 + 1 =  9
    printf("%d \n",*(*B + 1));      // *(400 + 4) = *404 = 7
    printf("%d \n",*(*B + 2));      // ???


}
```



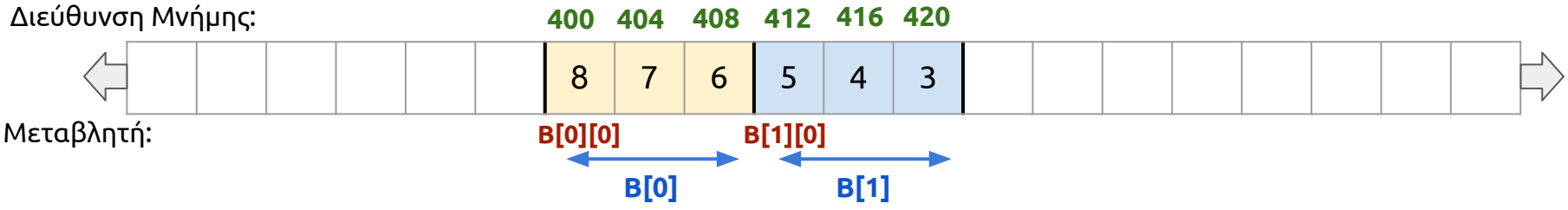
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {8,7,6}, {5,4,3} };
    printf("%d \n",*(B + 1) + 2);     // 412 + 8 = 420
    printf("%d \n",**B);     // *400 =  8
    printf("%d \n",**B+1);   // *400 + 1 =  9
    printf("%d \n",*(*B + 1));       // *(400 + 4) = *404 = 7
    printf("%d \n",*(*B + 2));       // *(400 + 8) = *408 = 6
    printf("%d \n",*(*B + 3));       // ???


}
```



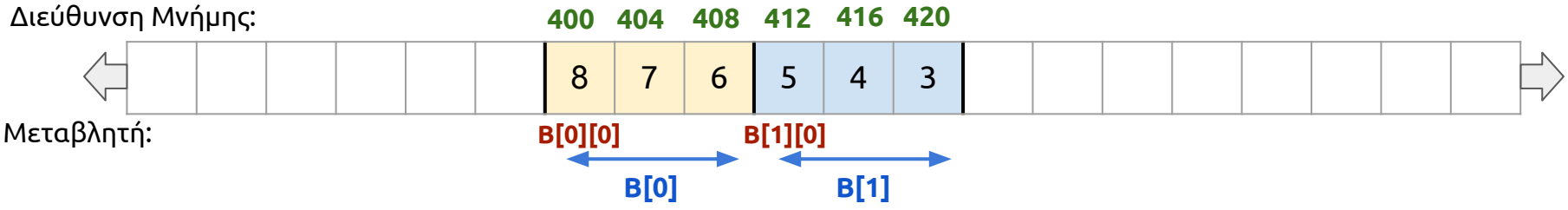
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {8,7,6}, {5,4,3} };
    printf("%d \n",*(B + 1) + 2);     // 412 + 8 = 420
    printf("%d \n",**B);     // *400 =  8
    printf("%d \n",**B+1);   // *400 + 1 =  9
    printf("%d \n",*(*B + 1));       // *(400 + 4) = *404 = 7
    printf("%d \n",*(*B + 2));       // *(400 + 8) = *408 = 6
    printf("%d \n",*(*B + 3));       // *(400 + 12) = *412 = 5
    printf("%d \n",*(*(B + 1)));     // ???

}
```



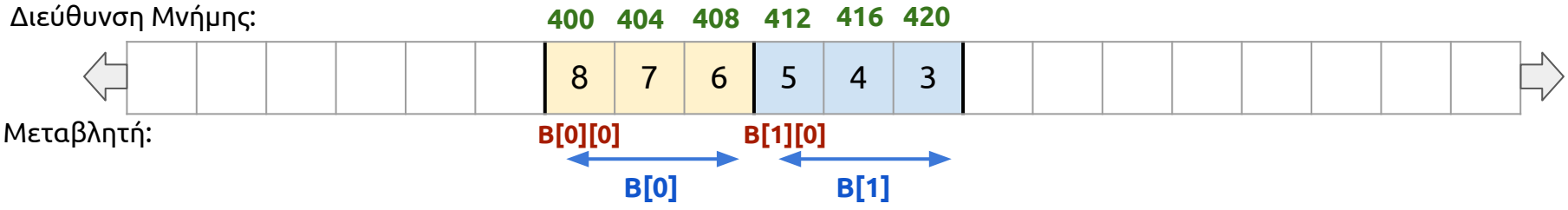
Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {8,7,6}, {5,4,3} };
    printf("%d \n",*(B + 1) + 2);     // 412 + 8 = 420
    printf("%d \n",**B);     // *400 =  8
    printf("%d \n",**B+1);   // *400 + 1 =  9
    printf("%d \n",*(*B + 1));       // *(400 + 4) = *404 = 7
    printf("%d \n",*(*B + 2));       // *(400 + 8) = *408 = 6
    printf("%d \n",*(*B + 3));       // *(400 + 12) = *412 = 5
    printf("%d \n",*(*(B + 1)));     // *412 = 5
    printf("%d \n",*(*(B + 1)+1));   // ???
}
```



Δείκτες σε πίνακες 2 διαστάσεων


```c
#include <stdio.h>
int main(){
    int B[2][3] = { {8,7,6}, {5,4,3} };
    printf("%d \n",*(B + 1) + 2);     // 412 + 8 = 420
    printf("%d \n",**B);     // *400 =  8
    printf("%d \n",**B+1);   // *400 + 1 =  9
    printf("%d \n",*(*B + 1));       // *(400 + 4) = *404 = 7
    printf("%d \n",*(*B + 2));       // *(400 + 8) = *408 = 6
    printf("%d \n",*(*B + 3));       // *(400 + 12) = *412 = 5
    printf("%d \n",*(*(B + 1)));     // *412 = 5
    printf("%d \n",*(*(B + 1)+1));   // *(412 + 4) = 4
}
```



Άσκηση 5.1

Να γράψετε κατάλληλες εντολές σε γλώσσα C, για να δείξετε ότι για τον πίνακα `int a[3][5]`,
οι ακόλουθες 4 εκφράσεις, είναι ισοδύναμες με το `a[i][j]`:

1. `*(a[i] + j)`
2. `(*(a + i))[j]`
3. `*((*(a + i)) + j)`
4. `*(&a[0][0] + 5*i + j)`

Άσκηση 5.1 - Λύση


```c
#include <stdio.h>

int main(){
    int a[3][5] = { {1,2,3,4,5}, {6,7,8,9,10}, {11,12,13,14,15} };
    int i = 1;
    int j = 4;
    printf("a[i][j] = %d \n",a[i][j]);  // ???




}
```

Άσκηση 5.1 - Λύση


```c
#include <stdio.h>

int main(){
    int a[3][5] = { {1,2,3,4,5}, {6,7,8,9,10}, {11,12,13,14,15} };
    int i = 1;
    int j = 4;
    printf("a[i][j] = %d \n",a[i][j]);  // 10
    printf("Ερώτημα 1: *(a[i] + j)  = %d \n",*(a[i] + j)); // ???



}
```

Άσκηση 5.1 - Λύση


```c
#include <stdio.h>

int main(){
    int a[3][5] = { {1,2,3,4,5}, {6,7,8,9,10}, {11,12,13,14,15} };
    int i = 1;
    int j = 4;
    printf("a[i][j] = %d \n",a[i][j]);  // 10
    printf("Ερώτημα 1: *(a[i] + j)  = %d \n",*(a[i] + j)); // 10
    printf("Ερώτημα 2: (*(a + i))[j] = %d \n",(*(a + i))[j]);  // ???


}
```

Άσκηση 5.1 - Λύση


```c
#include <stdio.h>

int main(){
    int a[3][5] = { {1,2,3,4,5}, {6,7,8,9,10}, {11,12,13,14,15} };
    int i = 1;
    int j = 4;
    printf("a[i][j] = %d \n",a[i][j]);  // 10
    printf("Ερώτημα 1: *(a[i] + j)  = %d \n",*(a[i] + j)); // 10
    printf("Ερώτημα 2: (*(a + i))[j] = %d \n",(*(a + i))[j]);  // 10
    printf("Ερώτημα 3: *((*(a + i)) + j) = %d \n",*((*(a + i)) + j)); // ???

}
```

Άσκηση 5.1 - Λύση


```c
#include <stdio.h>

int main(){
    int a[3][5] = { {1,2,3,4,5}, {6,7,8,9,10}, {11,12,13,14,15} };
    int i = 1;
    int j = 4;
    printf("a[i][j] = %d \n",a[i][j]);  // 10
    printf("Ερώτημα 1: *(a[i] + j)  = %d \n",*(a[i] + j)); // 10
    printf("Ερώτημα 2: (*(a + i))[j] = %d \n",(*(a + i))[j]);  // 10
    printf("Ερώτημα 3: *((*(a + i)) + j) = %d \n",*((*(a + i)) + j)); // 10
    printf("Ερώτημα 4: *(&a[0][0] + 5*i +j) = %d \n",*(&a[0][0] + 5*i + j)); // ???

}
```

Άσκηση 5.1 - Λύση


```c
#include <stdio.h>

int main(){
    int a[3][5] = { {1,2,3,4,5}, {6,7,8,9,10}, {11,12,13,14,15} };
    int i = 1;
    int j = 4;
    printf("a[i][j] = %d \n",a[i][j]);  // 10
    printf("Ερώτημα 1: *(a[i] + j)  = %d \n",*(a[i] + j)); // 10
    printf("Ερώτημα 2: (*(a + i))[j] = %d \n",(*(a + i))[j]);  // 10
    printf("Ερώτημα 3: *((*(a + i)) + j) = %d \n",*((*(a + i)) + j)); // 10
    printf("Ερώτημα 4: *(&a[0][0] + 5*i +j) = %d \n",*(&a[0][0] + 5*i + j)); // 10

}
```

Στατική δέσμευση μνήμης στο stack

Δήλωση Μονοδιάστατου Πίνακα (1D Array):

```
int arr[5] = {10, 20, 30, 40, 50};
```

Δήλωση Δισδιάστατου Πίνακα (2D Array):

```
int matrix[3][3] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
```

Δυναμική δέσμευση μνήμης στο heap

Δήλωση Μονοδιάστατου Πίνακα (1D Array):

```
int *arr;
int size = 5;
arr = (int *)malloc(size * sizeof(int));

if (arr == NULL) {
  printf("Αποτυχία δέσμευσης μνήμης\n");
  return 1;
}

...

free(arr); // Αποδέσμευση μνήμης
```

Δυναμική δέσμευση μνήμης στο heap

Δήλωση Δισδιάστατου Πίνακα (2D Array):

```c
int rows = 3, cols = 3;
int **matrix;
// Δέσμευση μνήμης για δείκτες γραμμών
matrix = (int **)malloc(rows * sizeof(int *));
if (matrix == NULL) {
    printf("Αποτυχία δέσμευσης μνήμης!\n");
    return 1;
}

// ... συνεχίζεται ...
```

Δυναμική δέσμευση μνήμης στο heap

Δήλωση Δισδιάστατου Πίνακα (2D Array):

```
for (int i = 0; i < rows; i++){
   matrix[i] = (int *)malloc(cols * sizeof(int)); // Δέσμευση για κάθε γραμμή
   if (matrix[i] == NULL) {
        printf("Αποτυχία δέσμευσης μνήμης στη γραμμή %d\n", i);
        // Αν αποτύχει η δέσμευση, αποδεσμεύουμε ό,τι δεσμεύτηκε μέχρι τώρα
        for (int j = 0; j < i; j++)
            free(matrix[j]);
        free(matrix);
        return 1;
    }
}
// Αποδέσμευση μνήμης
for (int i = 0; i < rows; i++)
    free(matrix[i]);
free(matrix);
```

Ισοδύναμες εκφράσεις


```
int **A = malloc(row * sizeof(int *));
```

ισοδύναμα

```
int **A;
A = (int **)malloc(row * sizeof(int *));
```

Άσκηση 5.2

Να γραφούν δύο συναρτήσεις σε γλώσσα C που να επιστρέφουν:

- Τον όγκο ενός κυλίνδρου με ακτίνα R και ύψος H ($V = \pi R^2 H$)
- Τον όγκο ενός ορθογωνίου ύψους H με βάση τετράγωνο πλευράς A ($V = A^2 H$)

Η συνάρτηση `main()` θέλουμε να :

1. γεμίζει ένα πίνακα `X[N][2]` με τυχαίους αριθμούς τύπου `double` με εύρος τιμών `[1.0-999.0]`. (Η τιμή του `N` ορίζεται ως σταθερά του προγράμματος).

2. καλεί για κάθε σειρά του πίνακα και τις δύο συναρτήσεις και θα εμφανίζει τις αντίστοιχες τιμές των συναρτήσεων (Η πρώτη στήλη του πίνακα αντιστοιχεί στα R ή A και η δεύτερη στήλη στο ύψος H).

Άσκηση 5.2 - Λύση


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#define N 10

double random_double(double min, double max) {
  return min + ((double)rand() / RAND_MAX) * (max - min);
}

double cylinder(double r, double h){
   double vol;
   vol= M_PI * (r * r) * h;
   return vol;
}
```

Άσκηση 5.2 - Λύση


```
double rectangle(double a, double h){
    double vol;
    vol=(a * a) * h;
    return vol;
}

int main(){
   double x[N][2];
   srand(time(0));
   for (int i = 0; i < N; i++){
       for (int j = 0; j < 2; j++)
           x[i][j] = random_double(1.0, 999.0);
   }
   for (int i = 0; i < N; i++)
       printf("%.2lf %.2lf \n", cylinder(x[i][0], x[i][1]), rectangle(x[i][0], x[i][1]));
}
```

Άσκηση 5.3

Να δημιουργήσετε μία συνάρτηση `displayMatrix` για την εμφάνιση ενός δισδιάστατου πίνακα ακεραίων αριθμών.

Η συνάρτηση `main()` θέλουμε να :

1. γεμίζει ένα πίνακα `A[20][2]` με τυχαίους αριθμούς τύπου `int` με εύρος τιμών `[0-99]`.

2. δημιουργεί και να εμφανίζει ένας νέος πίνακας `B` που να περιέχει εκείνες τις γραμμές του πίνακα `A`, στις οποίες η διαφορά των τιμών στις 2 στήλες της κάθε γραμμής, είναι κατά μέγιστο 10.

Άσκηση 5.3 - displayMatrix


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void displayMatrix(int **array, int rows, int columns){

     for(int i = 0; i < rows; i++){
          for(int j = 0; j < columns; j++){
             printf("%4d", array[i][j]);
          }
           printf("\n");
     }
}
```

Άσκηση 5.3 - main


```c
int main(){
    int **A = malloc(20 * sizeof(int *));
    for (int i = 0; i < 20; i++)
        A[i] = malloc(2 * sizeof(int));

    srand(time(0));
    int count = 0;

    for (int i = 0; i < 20; i++){
        for (int j = 0; j < 2; j++)
            A[i][j] = rand() % 100;

        if(abs(A[i][0] - A[i][1]) <= 10)
           count++;
     }

    displayMatrix(A, 20, 2);
```

```c
    int **B = malloc(count * sizeof(int *));
    for (int i = 0; i < count; i++)
        B[i] = malloc(2 * sizeof(int));

    count = 0;
    for (int i = 0; i < 20; i++){
        if (abs(A[i][0] - A[i][1]) <= 10){
            B[count][0] = A[i][0];
            B[count][1] = A[i][1];
            count++;
        }
    }
    printf("Εκτύπωση πίνακα B %d \n", count);
    displayMatrix(B, count, 2);
}
```

Άσκηση 5.4

Να γραφεί μια συνάρτηση `find_maxmin` που θα βρίσκει το μέγιστο και το ελάχιστο στοιχείο κάθε γραμμής ενός δισδιάστατου πίνακα.

Η συνάρτηση `main()` θέλουμε να :

1. γεμίζει ένα πίνακα `A[10][5]` με τυχαίους αριθμούς τύπου `int` με εύρος τιμών `[10-99]`.

2. καλεί την `find_maxmin` και να εμφανίζει τα μέγιστα και ελάχιστα κάθε γραμμής μαζί με τον πίνακα σε κατάλληλη θέση, όπως φαίνεται στο παρακάτω παράδειγμα:

19	45	77	43	34	**77**	**19**
55	79	65	27	45	**79**	**27**
27	13	16	37	88	**88**	**13**

Άσκηση 5.4 - find_maxmin


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int **find_maxmin(int **A, int rows, int cols){
    int max, min;

    int **B = malloc(10 * sizeof(int *));
    for (int i = 0; i < 10; i++) {
        B[i] = malloc(7 * sizeof(int));
    }
```

```c
    for (int i = 0; i < rows; i++){
        max = A[i][0];
        min = A[i][0];

        for (int j = 1; j < cols; j++){
            if (max < A[i][j])
                max = A[i][j];
            if (min > A[i][j])
                min = A[i][j];
            B[i][j] = A[i][j];
        }
        B[i][cols] = max;
        B[i][cols+1] = min;
    }
    return B;
}
```

Άσκηση 5.4 - main


```
int main(){

    int **A = malloc(10 * sizeof(int *));
    for (int i = 0; i < 10; i++)
        A[i] = malloc(5 * sizeof(int));

    srand(time(0));

    for (int i = 0; i < 10; i++){
        for (int j = 0; j < 5; j++){
            A[i][j] =  rand() % (99 - 10 + 1) + 10;
        }
     }
     int **B = find_maxmin(A, 10, 5);
     displayMatrix(B, 10, 7);
}
```

Συνάρτηση από την Άσκηση 5.3

Άσκηση 5.5

Να γραφεί μια συνάρτηση με όνομα `addMatrix` που να προσθέτει όλα τα στοιχεία ενός πίνακα ακεραίων αριθμών, 2 διαστάσεων (γραμμές και στήλες).

Επίσης, να γραφεί μια συνάρτηση με όνομα `displayMatrix` που να εμφανίζει ένα πίνακα ακεραίων αριθμών, 2 διαστάσεων (γραμμές και στήλες).

Η συνάρτηση `main()` θέλουμε να :

1. γεμίζει έναν δισδιάστατο πίνακα ακεραίων με τυχαίους θετικούς αριθμούς τύπου `int` στο διάστημα`[0, 99]`.
2. καλεί τη συνάρτηση `displayMatrix`,
3. καλεί τη συνάρτηση `addMatrix` και να εμφανίζει το αποτέλεσμα που επιστρέφεται.

(ΥΠΟΔΕΙΞΗ: Οι διαστάσεις του πίνακα να ορίζονται κατά την εκτέλεση του προγράμματος. Ο πίνακας δεδομένων να δημιουργείται με χρήση της `malloc()`).

Άσκηση 5.5 - addMatrix


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int addMatrix(int **matrix, int rows, int cols) {
    int sum = 0;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            sum += matrix[i][j];
        }
    }
    return sum;
}
```

Άσκηση 5.5 - main (1/2)


```c
int main() {
    int rows, cols;
    printf("Δώσε αριθμό για γραμμες και στήλες: ");
    scanf("%d %d", &rows, &cols);
    int **matrix = (int **)malloc(rows * sizeof(int *));
    if (matrix == NULL) {
        printf("Πρόβλημα παραχώρησης μνήμης!\n");
        return 1;
    }
    for (int i = 0; i < rows; i++) {
        matrix[i] = (int *)malloc(cols * sizeof(int));
        if (matrix[i] == NULL) {
            printf("Πρόβλημα παραχώρησης μνήμης!\n");
            return 1;
        }
    }
```

Άσκηση 5.5 - main (2/2)


```
    srand(time(0));

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++)
            matrix[i][j] = rand() % 100;
    }
    displayMatrix(matrix, rows, cols);

    int sum = addMatrix(matrix, rows, cols);
    printf("Άθροισμα στοιχείων πίνακα: %d\n", sum);

    for (int i = 0; i < rows; i++)
        free(matrix[i]);
    free(matrix);
    return 0;
}
```

Άσκηση 5.6

Να γραφεί μια συνάρτηση με όνομα `convert_to_1d` η οποία θα μετασχηματίζει ένα δισδιάστατο πίνακα ακεραίων θετικών αριθμών σε έναν μονοδιάστατο πίνακα. Η συνάρτηση θα επιστρέφει:

- έναν μονοδιάστατο πίνακα που θα περιλαμβάνει τα στοιχεία του αρχικού πίνακα δύο διαστάσεων, γραμμή προς γραμμή.
- έναν μονοδιάστατο πίνακα που θα περιλαμβάνει τα στοιχεία του αρχικού πίνακα δύο διαστάσεων, στήλη προς στήλη.

Στη συνάρτηση `main()`:

1. Να δημιουργηθεί ο δισδιάστατος πίνακας μέσω δυναμικής παραχώρησης μνήμης και μέσω προσδιορισμού των διαστάσεών του κατά τη διάρκεια εκτέλεσης του προγράμματος. Οι τιμές του πίνακα είναι τυχαίες ακέραιες στην περιοχή [`10, 99`].
2. Να κληθεί η συνάρτηση `convert_to_1d`.
3. Να χρησιμοποιήσετε τις κατάλληλες συναρτήσεις `displayMatrix` και `display_array` για την εμφάνιση δεδομένων και αποτελεσμάτων.

Άσκηση 5.6 - display_array


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void displayMatrix(int **array, int rows, int columns){
...
}

void display_array(int *array, int size) {
   for (int i = 0; i < size; i++) {
       printf("%2d ", array[i]);
   }
   printf("\n");
}
```

Άσκηση 5.6 - display_array


```
void convert_to_1d(int **matrix, int rows, int cols, int *row_major, int *col_major) {
    int index = 0;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            row_major[index] = matrix[i][j];
            index++;
        }
    }
    index = 0;
    for (int j = 0; j < cols; j++) {
        for (int i = 0; i < rows; i++) {
            col_major[index] = matrix[i][j];
            index++;
        }
    }
}
```

Άσκηση 5.6 - main (1/3)


```c
int main() {
    srand(time(NULL));
    int rows, cols;

    printf("Δώσε αριθμό γραμμών και στηλών: ");
    scanf("%d %d", &rows, &cols);
    int **matrix = malloc(rows * sizeof(int *));
    if (matrix == NULL) {
        printf("Πρόβλημα παραχώρησης μνήμης!\n");
        return 1;
    }
    for (int i = 0; i < rows; i++) {
        matrix[i] = malloc(cols * sizeof(int));
        if (matrix[i] == NULL) {
            printf("Πρόβλημα παραχώρησης μνήμης!\n");
            return 1;
        }
    }
```

Άσκηση 5.6 - main (2/3)


```
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            matrix[i][j] = rand() % 90 + 10;
        }
    }

    printf("Αρχικός Πίνακας:\n");
    displayMatrix(matrix, rows, cols);

    int *row_major = (int *)malloc(rows * cols * sizeof(int));
    int *col_major = (int *)malloc(rows * cols * sizeof(int));

    if (row_major == NULL || col_major == NULL) {
        printf("Πρόβλημα παραχώρησης μνήμης!\n");
        exit(1);
    }
```

Άσκηση 5.6 - main (3/3)


```
    convert_to_1d(matrix, rows, cols, row_major, col_major);

    printf("\nΚατά γραμμές:\n");
    display_array(row_major, rows * cols);

    printf("\nΚατά στήλες:\n");
    display_array(col_major, rows * cols);

    for (int i = 0; i < rows; i++) {
        free(matrix[i]);
    }
    free(matrix);
    free(row_major);
    free(col_major);
    return 0;
}
```