Δ.Π.Θ Δομημένος Προγραμματισμός

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ & ΔΙΟΙΚΗΣΗΣ

Δείκτες και Πίνακες

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

Κλήση Συνάρτησης με Τιμή

Όταν δημιουργούμε έναν πίνακα ακεραίων με 5 στοιχεία, π.χ:

int A[5]

Ουσιαστικά δημιουργούμε 5 νέες μεταβλητές:

A[0]

A[1]

A[2]

A[3]

A[4]

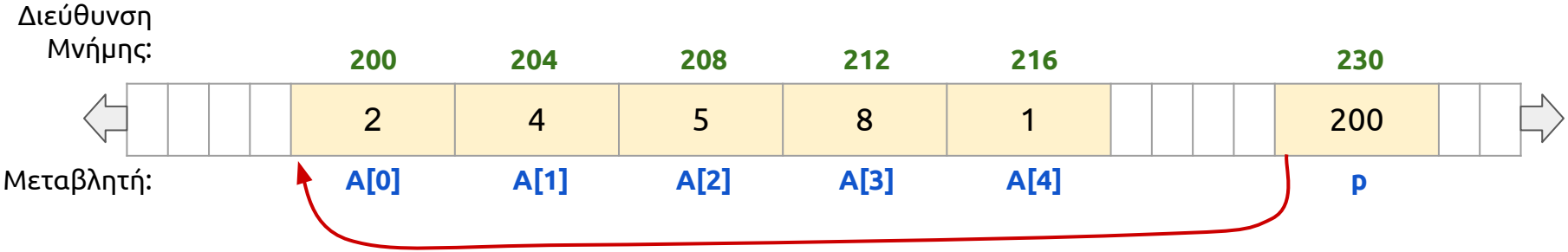
Οι μεταβλητές τοποθετούνται στην μνήμη, σε συνεχόμενες θέσεις

Διεύθυνση	Μεταβλητή
…	
216…219	A[4]
212…215	A[3]
208…211	A[2]
204…207	A[1]
200…203	A[0]
…	

Δείκτες και Πίνακες


```c
#include <stdio.h>

int main(){
    int A[] = {2, 4, 5, 8, 1};

    int *p;
    p = &A[0];
```

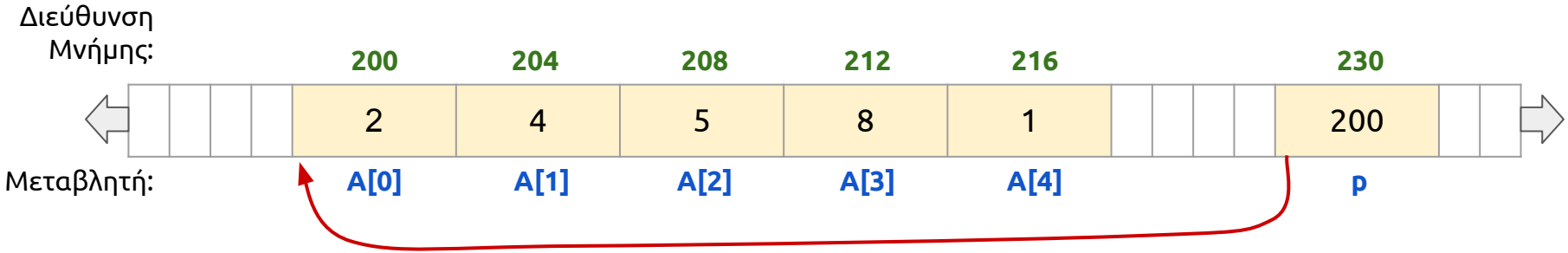
```c
    printf("%d\n", p);   // ???


}
```



Δείκτες και Πίνακες


```c
#include <stdio.h>

int main(){
    int A[] = {2, 4, 5, 8, 1};

    int *p;
    p = &A[0];
```

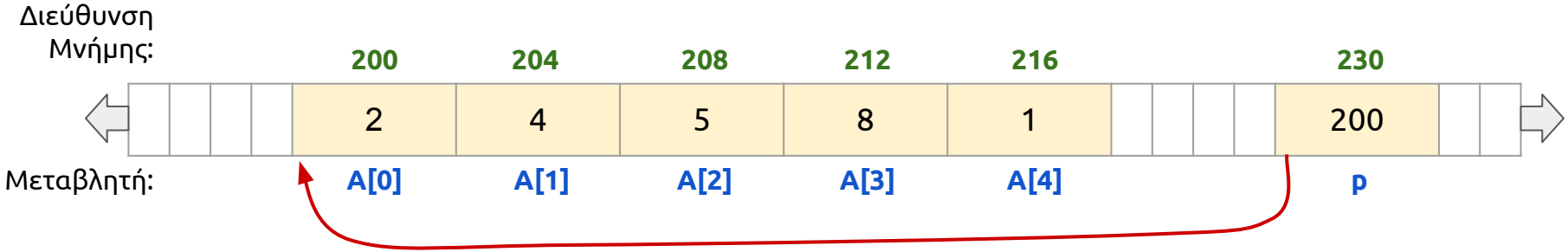
```c
    printf("%d\n", p);   // 200
    printf("%d\n", *p);  // ???


}
```



Δείκτες και Πίνακες


```c
#include <stdio.h>

int main(){
    int A[] = {2, 4, 5, 8, 1};

    int *p;
    p = &A[0];
```

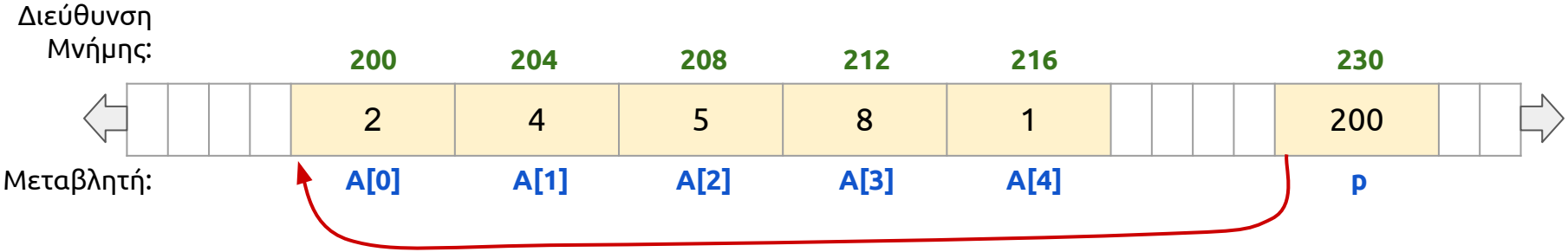
```c
    printf("%d\n", p);      // 200
    printf("%d\n", *p);     // 2
    printf("%d\n", p+1);    // ???

}
```



Δείκτες και Πίνακες


```c
#include <stdio.h>

int main(){
    int A[] = {2, 4, 5, 8, 1};

    int *p;
    p = &A[0];
```

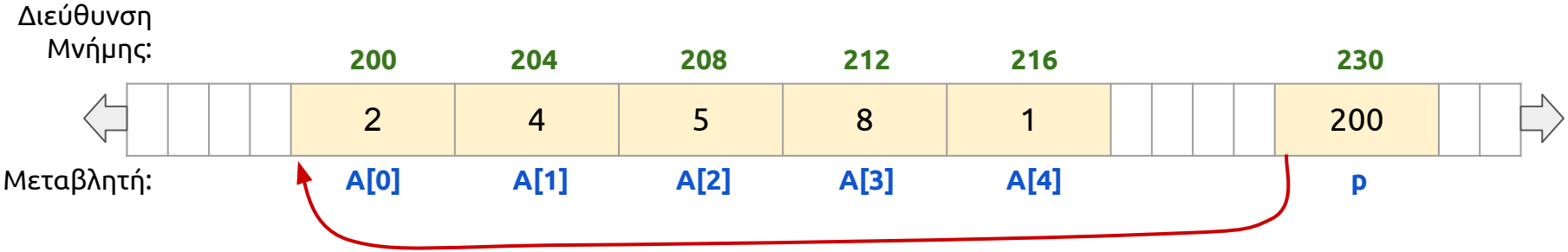
```c
    printf("%d\n", p);      // 200
    printf("%d\n", *p);     // 2
    printf("%d\n", p+1);    // 204
    printf("%d\n", *(p+1)); // ???

}
```



Δείκτες και Πίνακες


```c
#include <stdio.h>

int main(){
    int A[] = {2, 4, 5, 8, 1};

    int *p;
    p = &A[0];
```

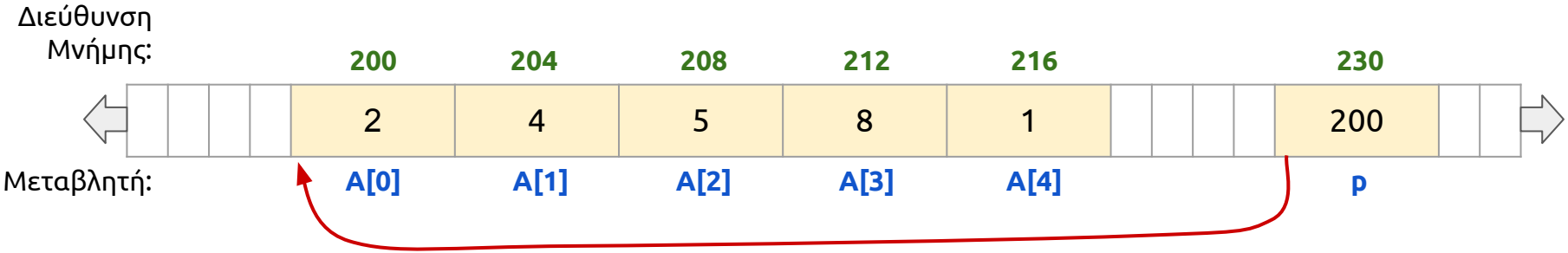
```c
    printf("%d\n", p);      // 200
    printf("%d\n", *p);     // 2
    printf("%d\n", p+1);    // 204
    printf("%d\n", *(p+1)); // 4
    printf("%d\n", *(p+4)); // ???
}
```



Δείκτες και Πίνακες

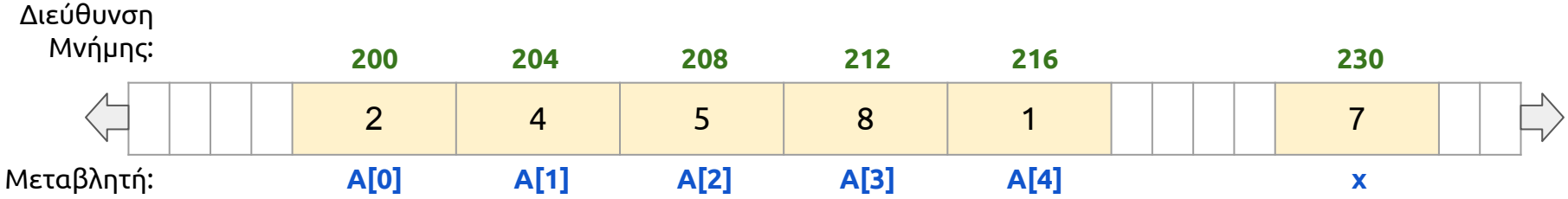

```c
#include <stdio.h>

int main(){
    int A[] = {2, 4, 5, 8, 1};

    int *p;
    p = &A[0];
```

```c
    printf("%d\n", p);      // 200
    printf("%d\n", *p);     // 2
    printf("%d\n", p+1);    // 204
    printf("%d\n", *(p+1)); // 4
    printf("%d\n", *(p+4)); // 1
}
```



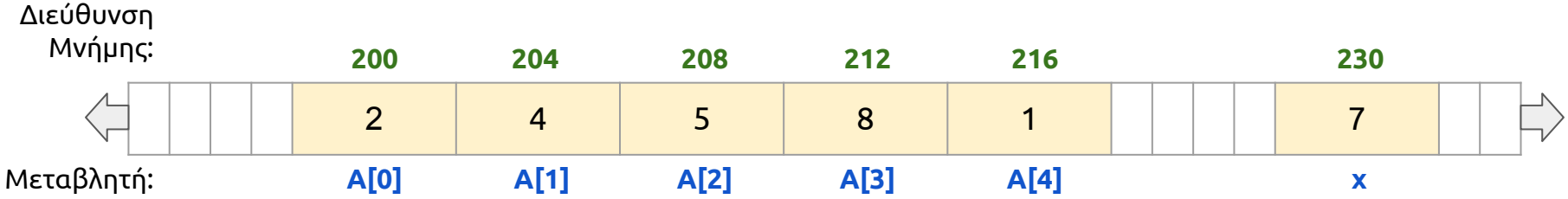
Δείκτες και Πίνακες


```c
#include <stdio.h>

int main(){
   int A[] = {2, 4, 5, 8, 1};
   int x = 7;

   printf("%d\n", &x);  // ???
}
```



Δείκτες και Πίνακες

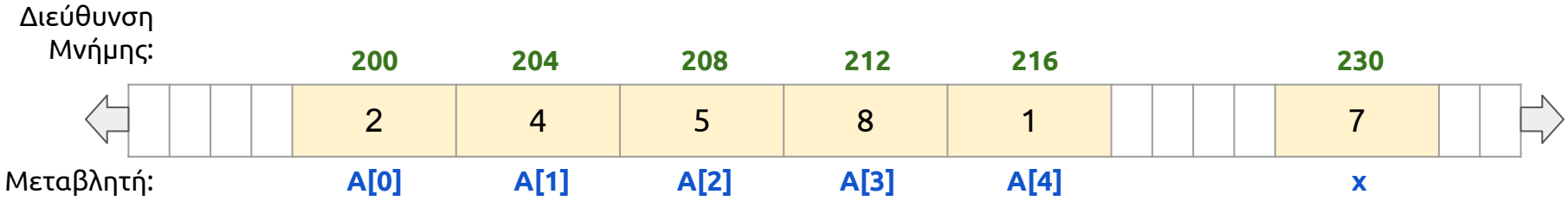

```c
#include <stdio.h>

int main(){
   int A[] = {2, 4, 5, 8, 1};
   int x = 7;
   printf("%d\n", &x);  // 230
   printf("%d\n", A);   // ???
}
```



Δείκτες και Πίνακες


```c
#include <stdio.h>

int main(){
   int A[] = {2, 4, 5, 8, 1};
   int x = 7;
   printf("%d\n", &x);  // 230
   printf("%d\n", A);   // 200
}
```

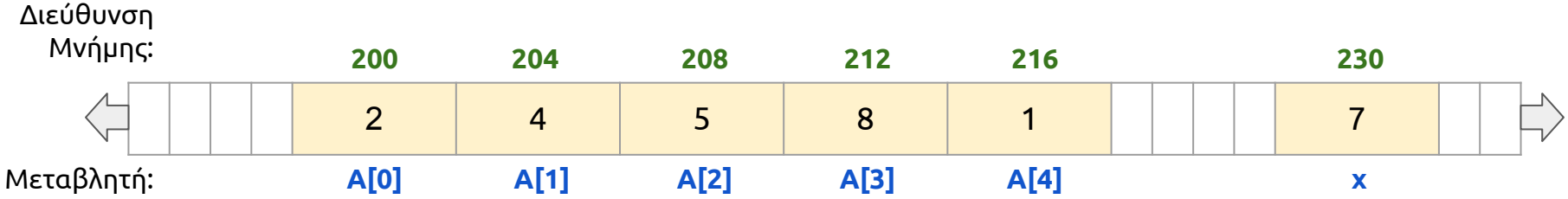
Τυπώνει τη Διεύθυνση μνήμης του 1ου στοιχείου του πίνακα A



Δείκτες και Πίνακες


```c
#include <stdio.h>

int main(){
   int A[] = {2, 4, 5, 8, 1};
   int x = 7;
   printf("%d\n", &x);  // 230
   printf("%d\n", A);   // 200
```

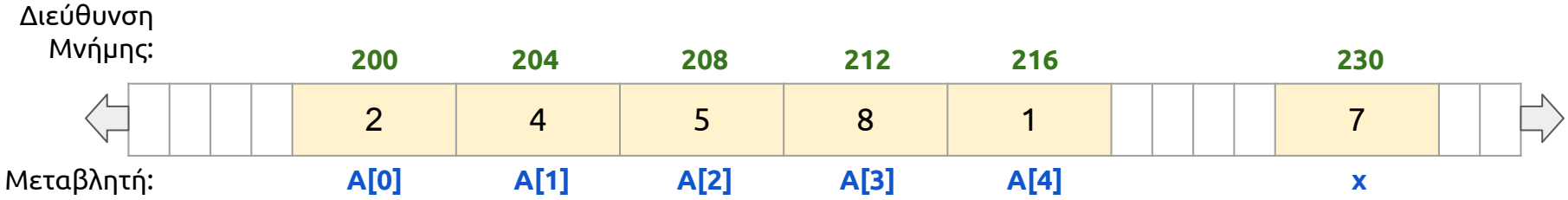
```c
   printf("%d\n", *A);    // ???



}
```



Δείκτες και Πίνακες


```c
#include <stdio.h>

int main(){
   int A[] = {2, 4, 5, 8, 1};
   int x = 7;
   printf("%d\n", &x);  // 230
   printf("%d\n", A);   // 200
```

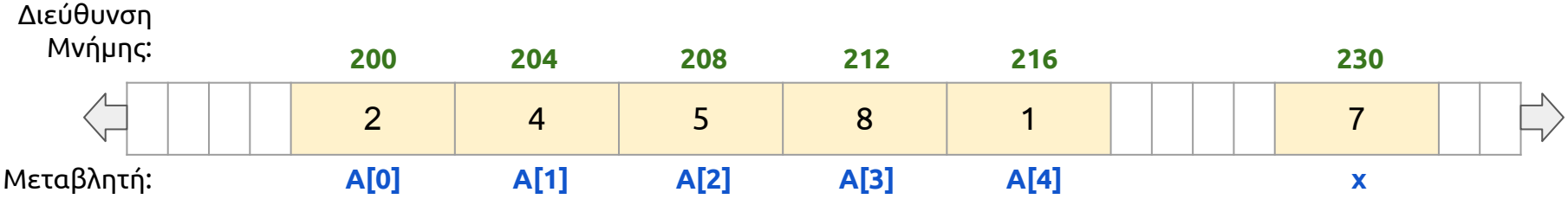
```c
   printf("%d\n", *A);    // 2
   printf("%d\n", A+1);   // ???



}
```



Δείκτες και Πίνακες


```c
#include <stdio.h>

int main(){
   int A[] = {2, 4, 5, 8, 1};
   int x = 7;
   printf("%d\n", &x);  // 230
   printf("%d\n", A);   // 200
```

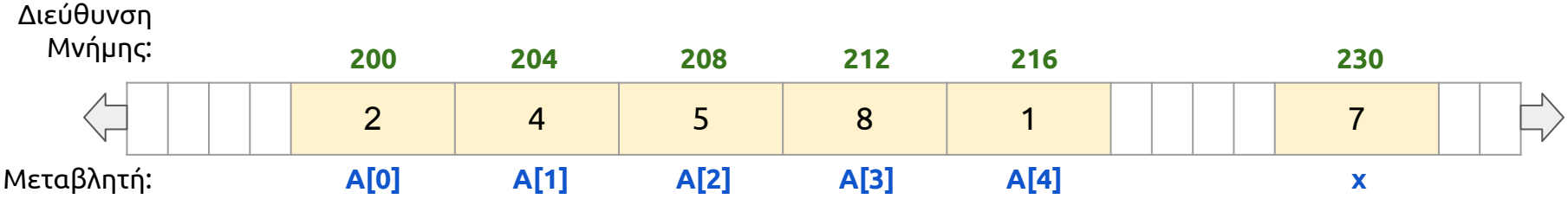
```c
   printf("%d\n", *A);    // 2
   printf("%d\n", A+1);   // 204
   printf("%d\n", *(A+1));  // ???


}
```



Δείκτες και Πίνακες


```c
#include <stdio.h>

int main(){
   int A[] = {2, 4, 5, 8, 1};
   int x = 7;
   printf("%d\n", &x);  // 230
   printf("%d\n", A);   // 200
```

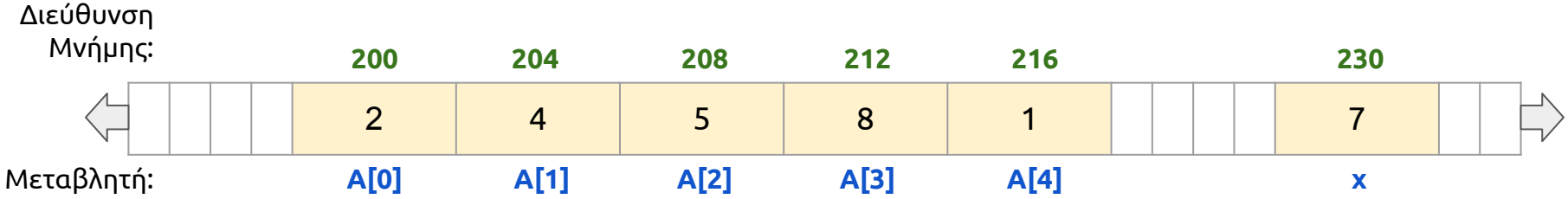
```c
   printf("%d\n", *A);    // 2
   printf("%d\n", A+1);   // 204
   printf("%d\n", *(A+1));  // 4
   printf("%d\n", *(A+4));  // ???

}
```



Δείκτες και Πίνακες


```c
#include <stdio.h>

int main(){
   int A[] = {2, 4, 5, 8, 1};
   int x = 7;
   printf("%d\n", &x);  // 230
   printf("%d\n", A);   // 200
```

```c
   printf("%d\n", *A);    // 2
   printf("%d\n", A+1);   // 204
   printf("%d\n", *(A+1));  // 4
   printf("%d\n", *(A+4));  // 1

}
```



Δείκτες και Πίνακες

Ισοδύναμες Εκφράσεις

```c
#include <stdio.h>

int main(){
   int A[] = {2, 4, 5, 8, 1};

   printf("%d\n", A);
   printf("%d\n", A+1);
   printf("%d\n", A+2);
   printf("%d\n", A+3);
   printf("%d\n", A+4);
```

```c
   printf("%d\n", &A[0]);
   printf("%d\n", &A[1]);
   printf("%d\n", &A[2]);
   printf("%d\n", &A[3]);
   printf("%d\n", &A[4]);

}
```

Δείκτες και Πίνακες

Ισοδύναμες Εκφράσεις

```c
#include <stdio.h>

int main(){
   int A[] = {2, 4, 5, 8, 1};

   printf("%d\n", *A);
   printf("%d\n", *(A+1));
   printf("%d\n", *(A+2));
   printf("%d\n", *(A+3));
   printf("%d\n", *(A+4));
```

```c
   printf("%d\n", A[0]);
   printf("%d\n", A[1]);
   printf("%d\n", A[2]);
   printf("%d\n", A[3]);
   printf("%d\n", A[4]);

}
```

Παράδειγμα

Διερευνήστε την ορθότητα των παρακάτω εκφράσεων:

```
a[i] == (*(a + i))   // ???
```

Παράδειγμα

Διερευνήστε την ορθότητα των παρακάτω εκφράσεων:

```
a[i] == (*(a + i))   // Σωστό
a[0] == (*(a + 0))   // ???
```

Παράδειγμα

Διερευνήστε την ορθότητα των παρακάτω εκφράσεων:

```
a[i] == (*(a + i))    // Σωστό
a[0] == (*(a + 0))    // Σωστό
a[0] == *(a + 0)      // ???
```

Παράδειγμα

Διερευνήστε την ορθότητα των παρακάτω εκφράσεων:

```
a[i] == (*(a + i))    // Σωστό
a[0] == (*(a + 0))    // Σωστό
a[0] == *(a + 0)      // Σωστό
a[0] == *a            // ???
```

Παράδειγμα

Διερευνήστε την ορθότητα των παρακάτω εκφράσεων:

```
a[i] == (*(a + i))    // Σωστό
a[0] == (*(a + 0))    // Σωστό
a[0] == *(a + 0)      // Σωστό
a[0] == *a            // Σωστό
&a[i] == &(*(a + i)) // ???
```

Παράδειγμα

Διερευνήστε την ορθότητα των παρακάτω εκφράσεων:

```
a[i] == (*(a + i))    // Σωστό
a[0] == (*(a + 0))    // Σωστό
a[0] == *(a + 0)      // Σωστό
a[0] == *a            // Σωστό
&a[i] == &(*(a + i)) // Σωστό
&a[i] == a + i        // ???
```

Παράδειγμα

Διερευνήστε την ορθότητα των παρακάτω εκφράσεων:

```
a[i] == (*(a + i))    // Σωστό
a[0] == (*(a + 0))    // Σωστό
a[0] == *(a + 0)      // Σωστό
a[0] == *a            // Σωστό
&a[i] == &(*(a + i)) // Σωστό
&a[i] == a + i        // Σωστό
&a[i] == &a[0] + i    // ???
```

Παράδειγμα

Διερευνήστε την ορθότητα των παρακάτω εκφράσεων:

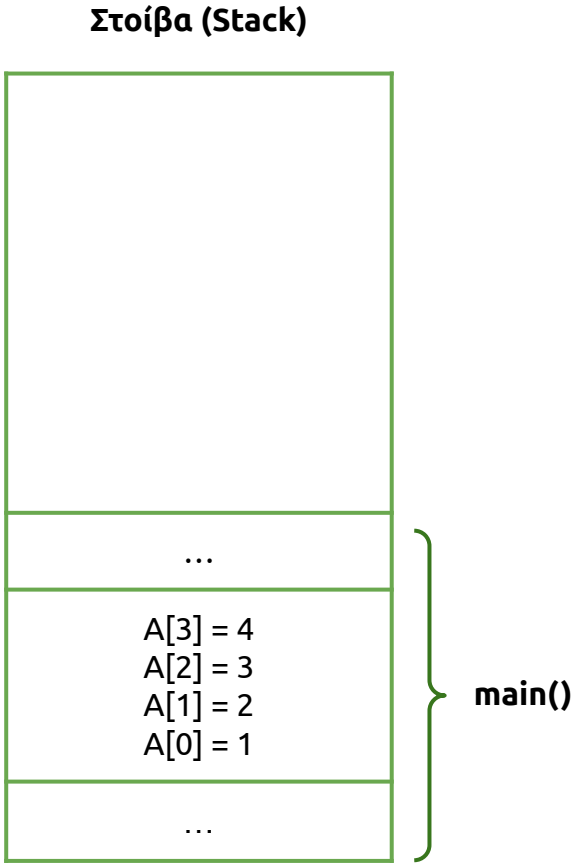
```
a[i] == (*(a + i))   // Σωστό
a[0] == (*(a + 0))   // Σωστό
a[0] == *(a + 0)     // Σωστό
a[0] == *a           // Σωστό
&a[i] == &(*(a + i)) // Σωστό
&a[i] == a + i       // Σωστό
&a[i] == &a[0] + i   // Σωστό
```

Πίνακες και Συναρτήσεις


```c
#include <stdio.h>

int sumOfElements(int A[]){
   int sum = 0;
   for(int i = 0; i < 5; i++){
       sum += A[i];
   }
   return sum;
}

int main() {
   int A[] = {1, 2, 3, 4, 5};
   int total = sumOfElements(A);
   printf("sum: %d \n", total);
   return 0;
}
```

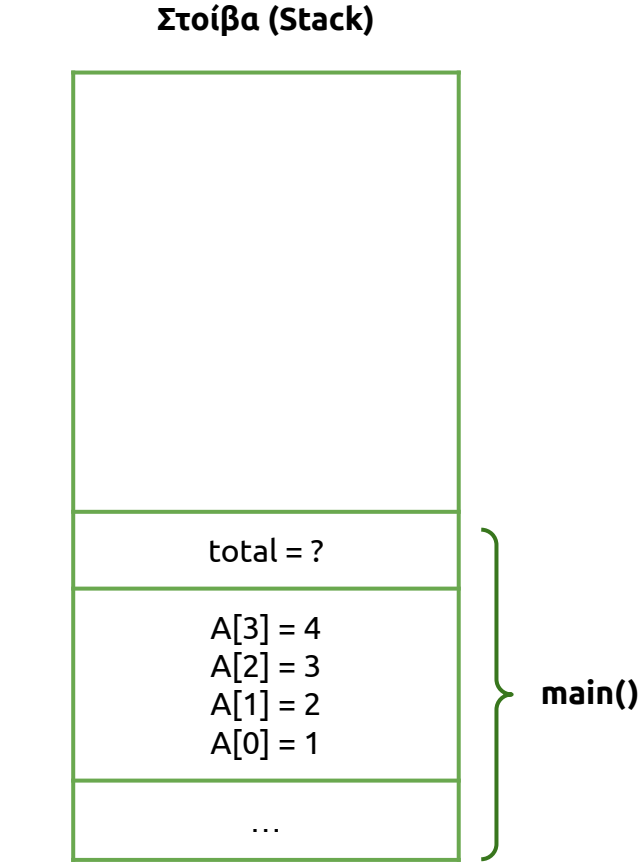
Πίνακες και Συναρτήσεις


```c
#include <stdio.h>

int sumOfElements(int A[]){
    int sum = 0;
    for(int i = 0; i < 4; i++){
        sum += A[i];
    }
    return sum;
}

int main() {
    int A[] = {1, 2, 3, 4};
    int total = sumOfElements(A);
    printf("sum: %d \n", total);
    return 0;
}
```



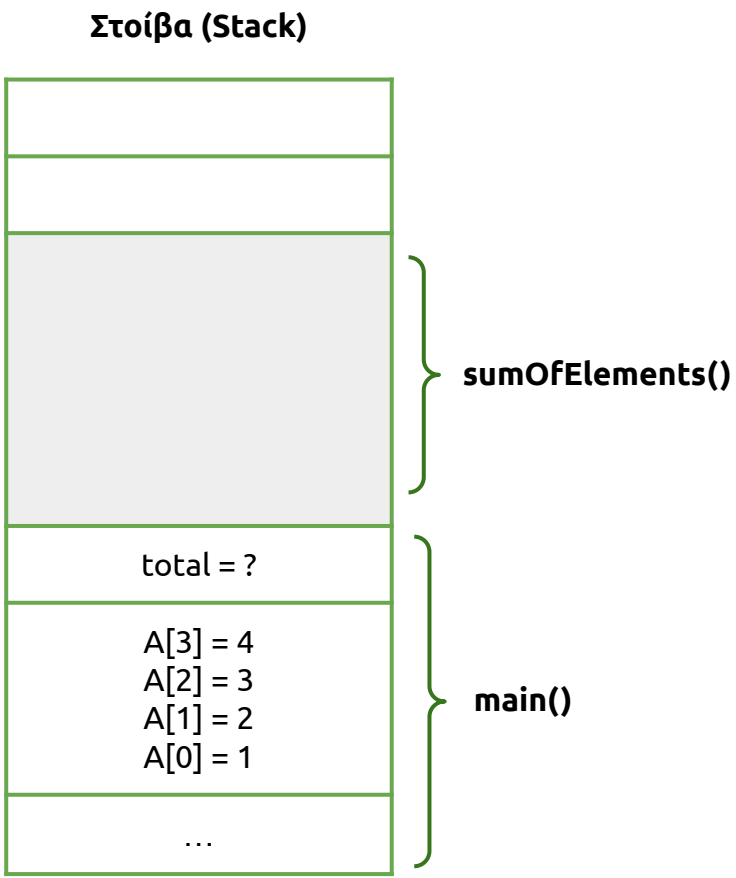
Πίνακες και Συναρτήσεις


```c
#include <stdio.h>

int sumOfElements(int A[]){
   int sum = 0;
   for(int i = 0; i < 4; i++){
       sum += A[i];
   }
   return sum;
}

int main() {
   int A[] = {1, 2, 3, 4};
   int total = sumOfElements(A);
   printf("sum: %d \n", total);
   return 0;
}
```



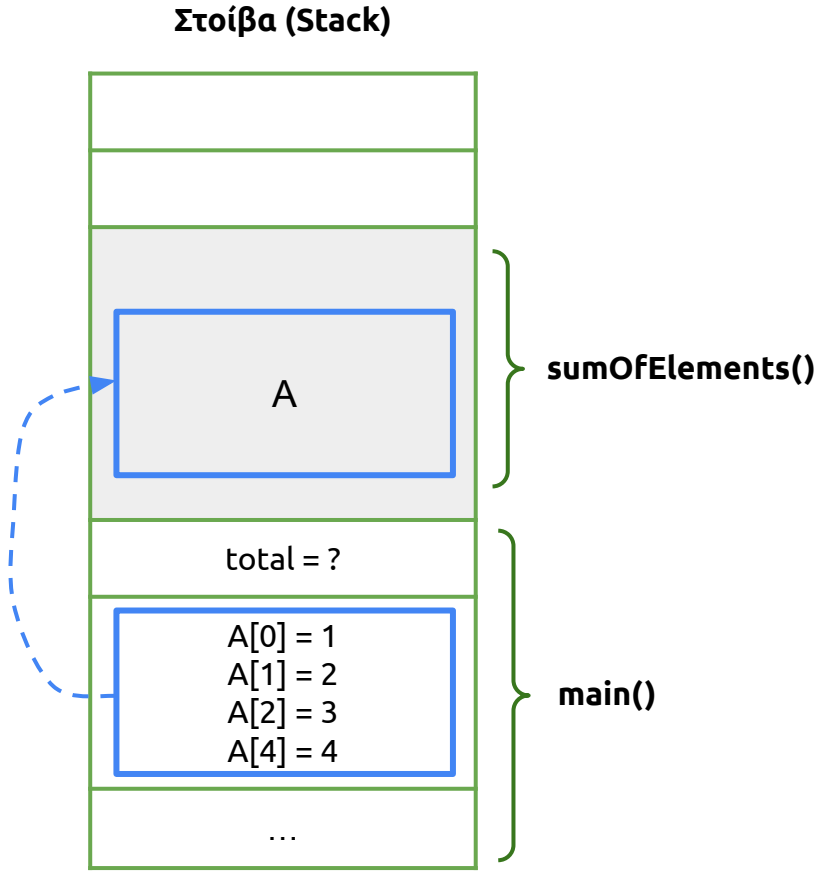
Πίνακες και Συναρτήσεις


```c
#include <stdio.h>

int sumOfElements(int A[]){
    int sum = 0;
    for(int i = 0; i < 4; i++){
        sum += A[i];
    }
    return sum;
}

int main() {
    int A[] = {1, 2, 3, 4};
    int total = sumOfElements(A);
    printf("sum: %d \n", total);
    return 0;
}
```



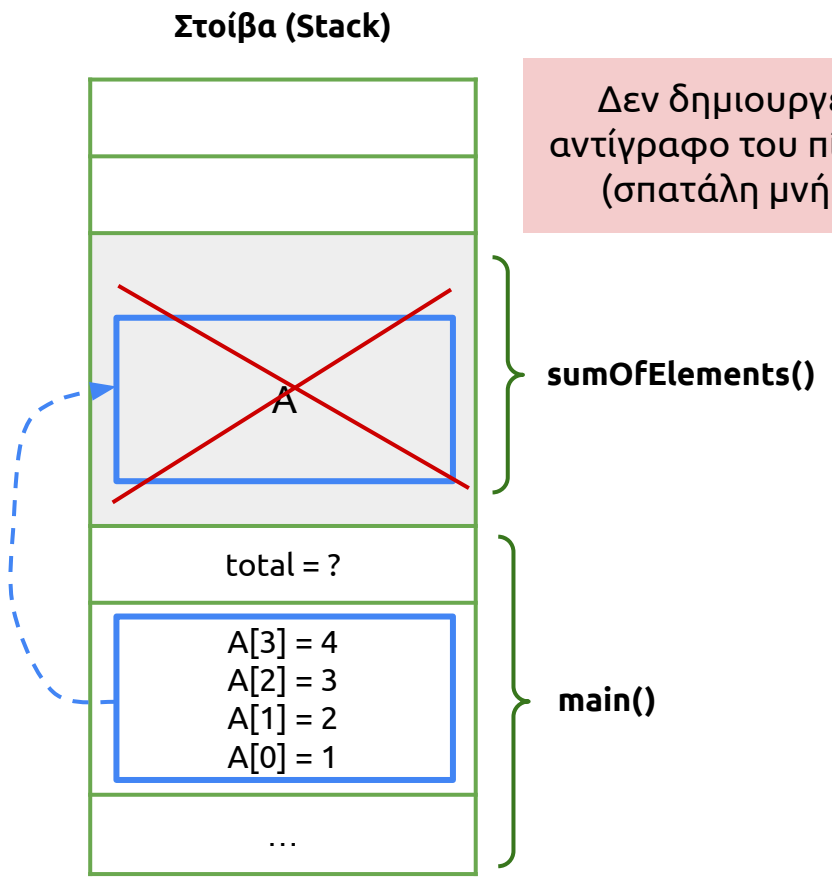
Πίνακες και Συναρτήσεις


```c
#include <stdio.h>

int sumOfElements(int A[]){
    int sum = 0;
    for(int i = 0; i < 4; i++){
        sum += A[i];
    }
    return sum;
}

int main() {
    int A[] = {1, 2, 3, 4};
    int total = sumOfElements(A);
    printf("sum: %d \n", total);
    return 0;
}
```



Πίνακες και Συναρτήσεις

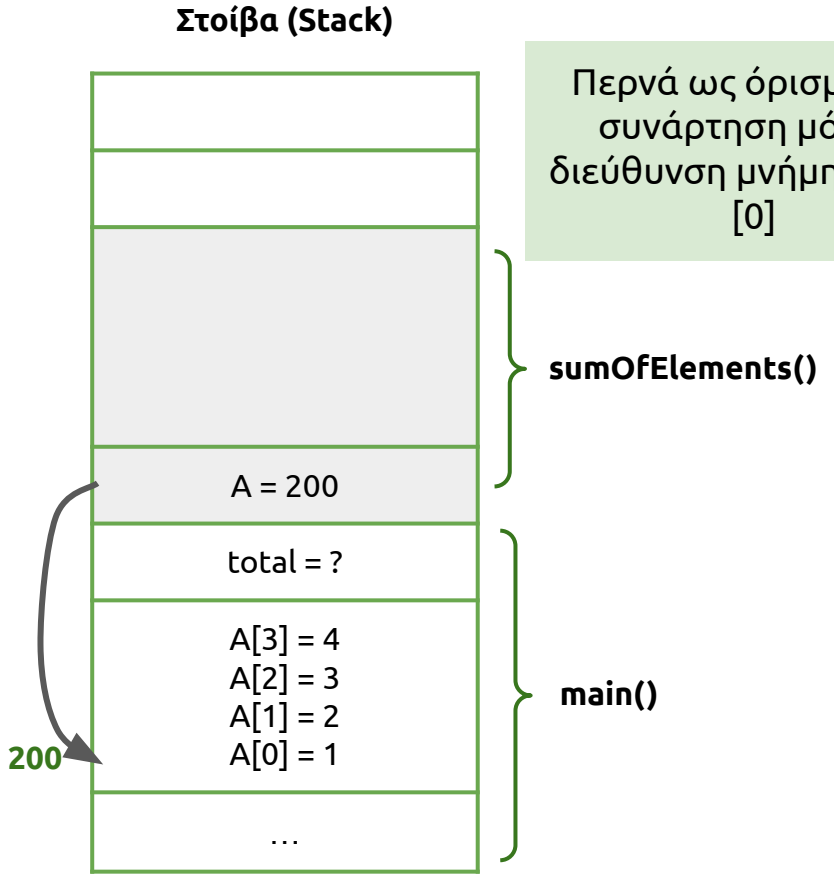

```c
#include <stdio.h>

int sumOfElements(int A[]){
    int sum = 0;
    for(int i = 0; i < 4; i++){
        sum += A[i];
    }
    return sum;
}

int main() {
    int A[] = {1, 2, 3, 4};
    int total = sumOfElements(A);
    printf("sum: %d \n", total);
    return 0;
}
```



Δεν δημιουργείται αντίγραφο του πίνακα Α (σπατάλη μνήμης)

Πίνακες και Συναρτήσεις

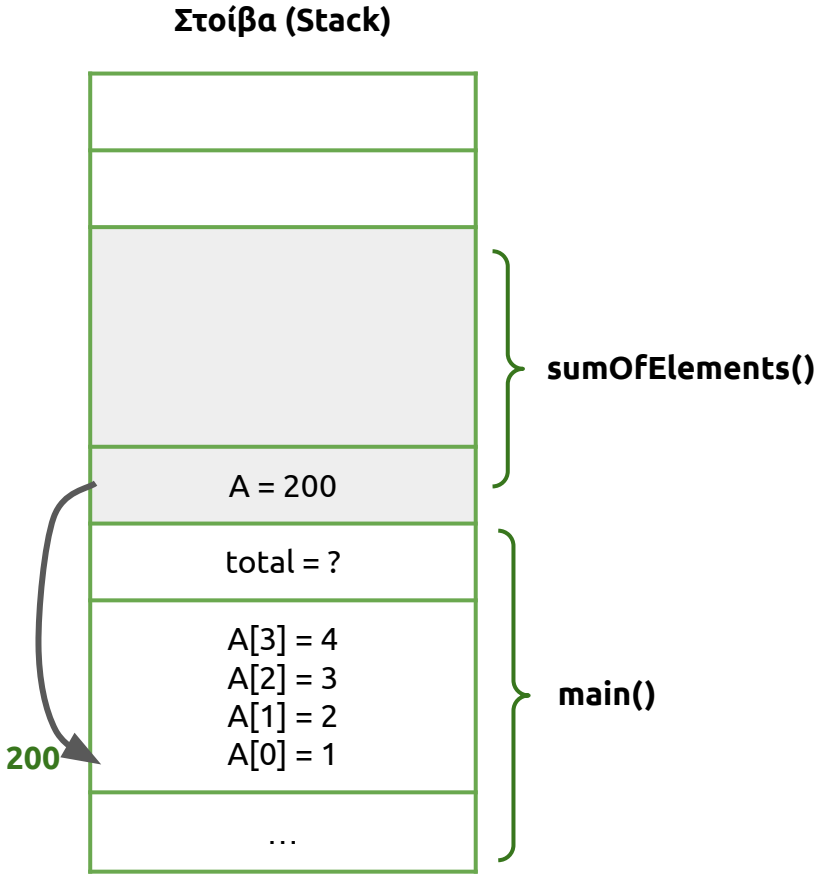

```c
#include <stdio.h>

int sumOfElements(int A[]){
    int sum = 0;
    for(int i = 0; i < 4; i++){
        sum += A[i];
    }
    return sum;
}

int main() {
    int A[] = {1, 2, 3, 4};
    int total = sumOfElements(A);
    printf("sum: %d \n", total);
    return 0;
}
```



Περνά ως όρισμα στη συνάρτηση μόνο η διεύθυνση μνήμης του A [0]

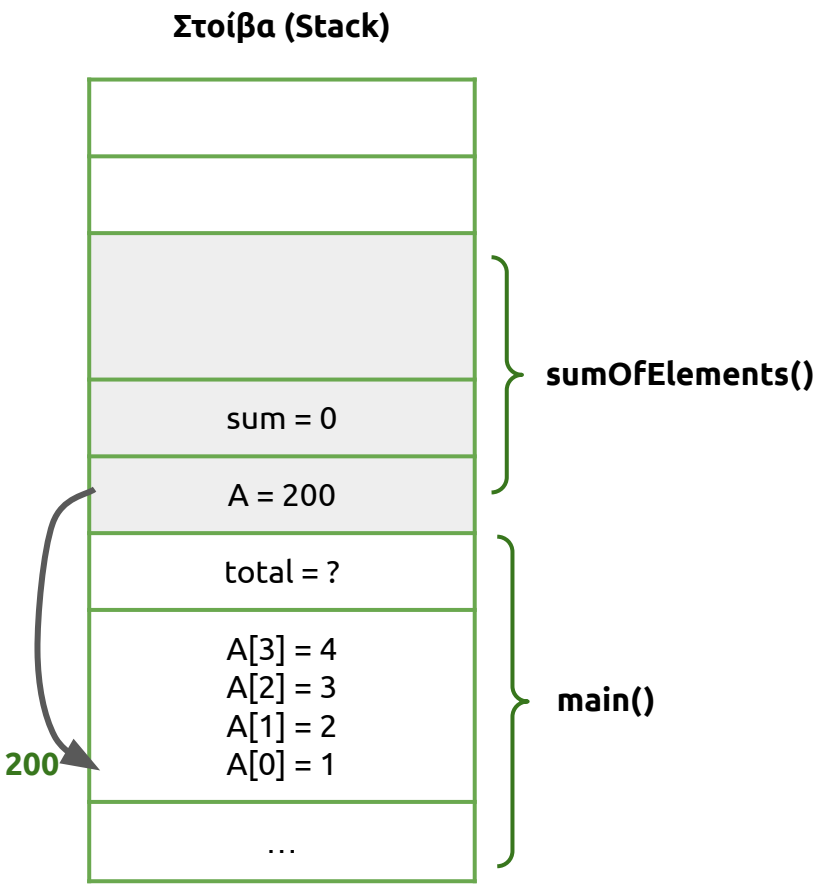
Πίνακες και Συναρτήσεις


```c
#include <stdio.h>

int sumOfElements(int *A){
    int sum = 0;
    for(int i = 0; i < 4; i++){
        sum += A[i];
    }
    return sum;
}

int main() {
    int A[] = {1, 2, 3, 4};
    int total = sumOfElements(A);
    printf("sum: %d \n", total);
    return 0;
}
```



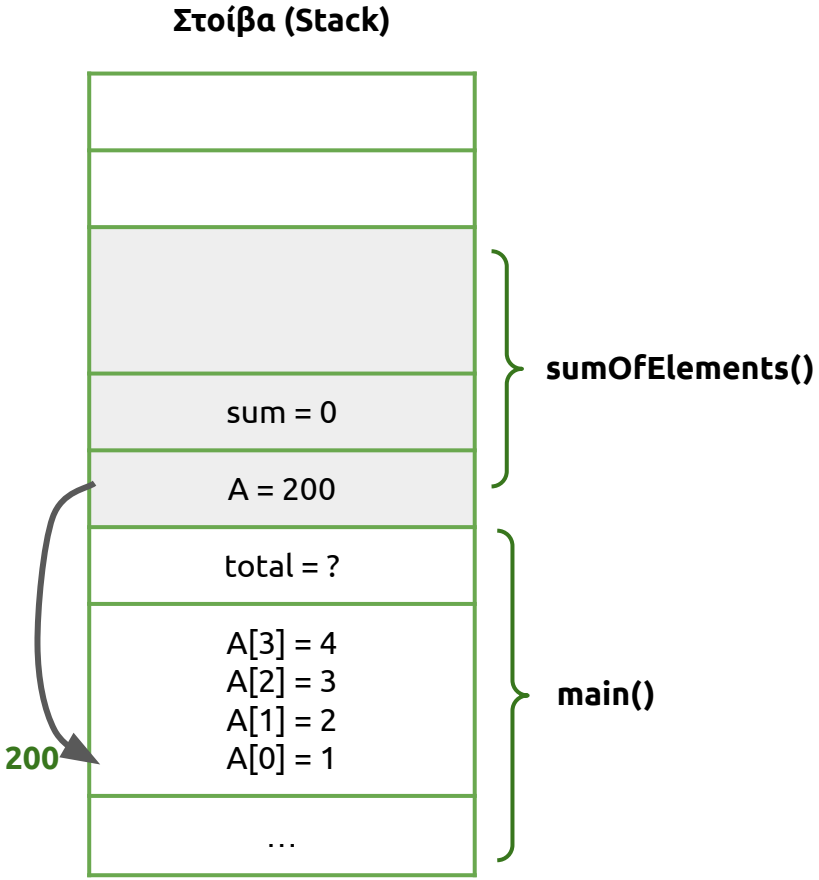
Πίνακες και Συναρτήσεις


```c
#include <stdio.h>

int sumOfElements(int *A){
    int sum = 0;
    for(int i = 0; i < 4; i++){
        sum += A[i];
    }
    return sum;
}

int main() {
    int A[] = {1, 2, 3, 4};
    int total = sumOfElements(A);
    printf("sum: %d \n", total);
    return 0;
}
```



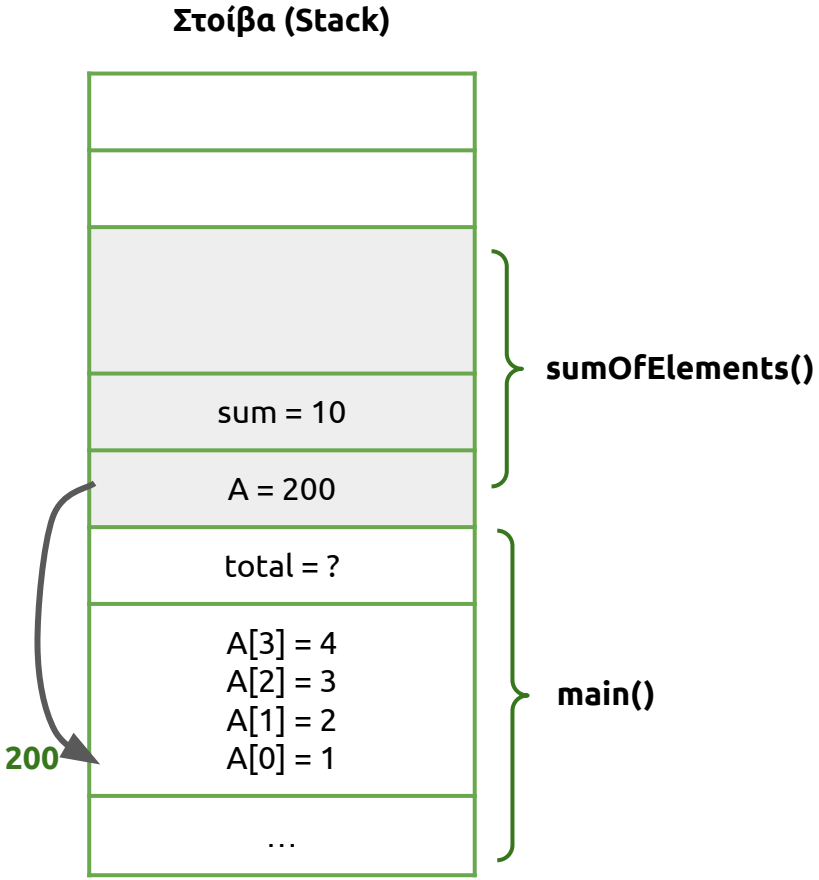
Πίνακες και Συναρτήσεις


```c
#include <stdio.h>

int sumOfElements(int *A){
    int sum = 0;
    for(int i = 0; i < 4; i++){
        sum += *(A + i);
    }
    return sum;
}

int main() {
    int A[] = {1, 2, 3, 4};
    int total = sumOfElements(A);
    printf("sum: %d \n", total);
    return 0;
}
```



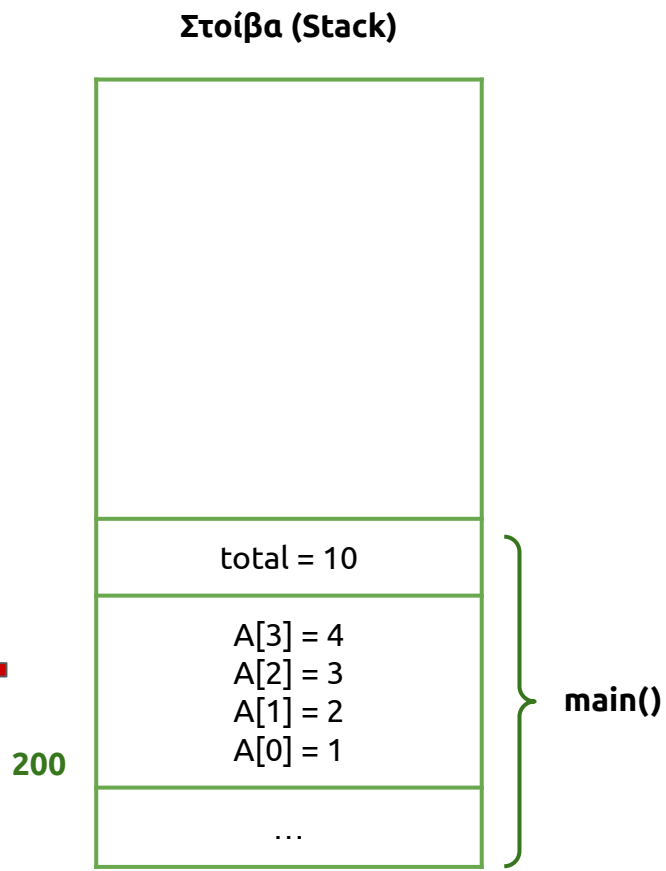
Πίνακες και Συναρτήσεις


```c
#include <stdio.h>

int sumOfElements(int *A){
    int sum = 0;
    for(int i = 0; i < 4; i++){
        sum += *(A + i);
    }
    return sum;
}

int main() {
    int A[] = {1, 2, 3, 4};
    int total = sumOfElements(A);
    printf("sum: %d \n", total);
    return 0;
}
```



Πίνακες και Συναρτήσεις


```c
#include <stdio.h>

int sumOfElements(int *A){
    int sum = 0;
    for(int i = 0; i < 4; i++){
        sum += *(A + i);
    }
    return sum;
}

int main() {
    int A[] = {1, 2, 3, 4};
    int total = sumOfElements(A);
    printf("sum: %d \n", total);
    return 0;
}
```



Πίνακες και Συναρτήσεις


```c
#include <stdio.h>

int sumOfElements(int *A){
    int sum = 0;
    for(int i = 0; i < 4; i++){
        sum += *(A + i);
    }
    return sum;
}

int main() {
    int A[] = {1, 2, 3, 4};
    int total = sumOfElements(A);
    printf("sum: %d \n", total);
    return 0;
}
```

Η συνάρτηση δεν γνωρίζει το μέγεθος του πίνακα

Πίνακες και Συναρτήσεις

Πιο γενική λύση της άσκησης.
Αν αλλάξω μέγεθος στον πίνακα, δεν θα χρειαστεί να πειράξω καθόλου τη συνάρτηση

```c
#include <stdio.h>

int sumOfElements(int *A, int size){
    int sum = 0;
    for(int i = 0; i < size; i++){
        sum += A[i];
    }
    return sum;
}
```

```c
int main() {
    int A[] = {1, 2, 3, 4};
    int size = sizeof(A) / sizeof(A[0]);

    int total = sumOfElements(A, size);

    printf("sum: %d \n", total);

    return 0;
}
```

Έλεγχος Ταξινόμησης σε Πίνακα

A[5]

1
3
7
12
13

Θέλω να ελέγξω αν ο πίνακας A[5] είναι ταξινομημένος κατά αύξουσα σειρά.

Έλεγχος Ταξινόμησης σε Πίνακα

A[5]

1
3
7
12
13

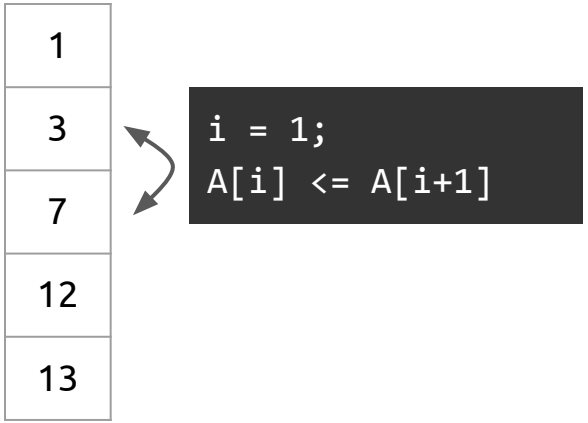
```
i = 0;
A[i] <= A[i+1]
```

True: θα πρέπει να συνεχίσω για i=1

False: Σταματάω. Ο πίνακας δεν είναι ταξινομημένος με αύξουσα σειρά

Έλεγχος Ταξινόμησης σε Πίνακα

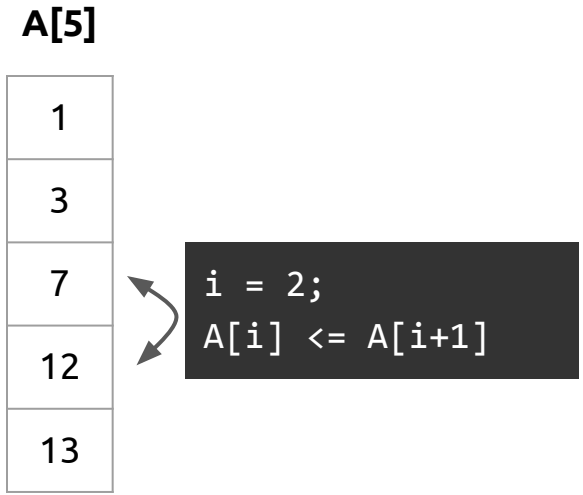
A[5]



True: θα πρέπει να συνεχίσω για i=2

False: Σταματάω. Ο πίνακας δεν είναι ταξινομημένος με αύξουσα σειρά

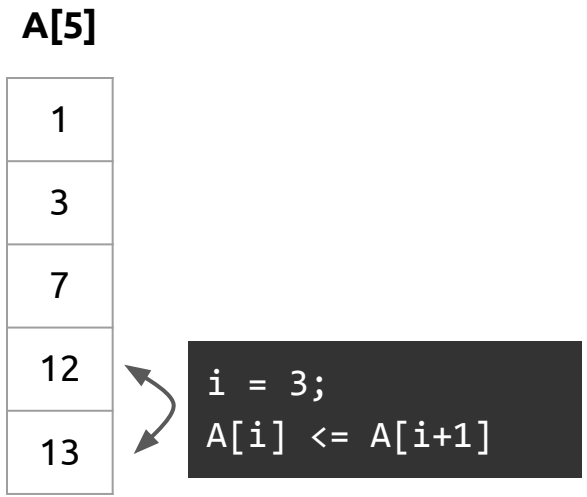
Έλεγχος Ταξινόμησης σε Πίνακα



True: θα πρέπει να συνεχίσω για i=3

False: Σταματάω. Ο πίνακας δεν είναι ταξινομημένος με αύξουσα σειρά

Έλεγχος Ταξινόμησης σε Πίνακα



True: Συμπέρασμα, ο πίνακας είναι ταξινομημένος

False: Σταματάω. Ο πίνακας δεν είναι ταξινομημένος με αύξουσα σειρά

Έλεγχος Ταξινόμησης σε Πίνακα

A[5]

1
3
7
12
13

```
bool sorted = true;

for(int i = 0; i < 4; i++){
    if(A[i] > A[i+1]){
      sorted = false;
      break;
    }
}
if(sorted)
   printf("Είναι Ταξινομημένος");
else
   printf("Δεν είναι Ταξινομημένος");
```

Άσκηση 3.1

Να γράψετε μια συνάρτηση με όνομα `out_of_order` που θα δέχεται 2 παραμέτρους:

- ένα πίνακα με στοιχεία τύπου `float`
- το μέγεθος του πίνακα τύπου `int`

Η συνάρτηση θα ελέγχει αν ο πίνακας είναι ταξινομημένος σε αύξουσα διάταξη (δηλαδή `a[0] ≤ a[1] ≤ a[2] ≤ ...`) και θα επιστρέφει:

- την τιμή `-1` αν όλα τα στοιχεία είναι ταξινομημένα
- τη θέση του πρώτου στοιχείου που είναι εκτός ταξινομημένης διάταξης.

Στη συνέχεια να γράψετε πρόγραμμα που να καλεί τη συνάρτηση `out_of_order` και να εμφανίζει το αποτέλεσμα.

Άσκηση 3.1 - out_of_order


```c
#include <stdio.h>
int out_of_order(float A[], int size) {
    for (int i = 0; i < size - 1; i++) {
        if (A[i] > A[i + 1])
            return i + 1; // Επιστρέφει τη θέση του πρώτου μη ταξινομημένου στοιχείου
    }
    return -1; // Όλα τα στοιχεία είναι ταξινομημένα
}
```

```c
int out_of_order(float *A, int size) {
    for (int i = 0; i < size - 1; i++) {
        if (*(A + i) > *(A + i + 1))
            return i + 1;
    }
    return -1;
}
```

Άσκηση 3.1 - main


```c
int main() {
    float A[] = {1.2, 2.3, 3.5, 2.8, 5.0}; // Παράδειγμα μη ταξινομημένου πίνακα
    int size = sizeof(A) / sizeof(A[0]);

    int result = out_of_order(A, size);

    if (result == -1) {
        printf("Ο πίνακας είναι ταξινομημένος.\n");
    } else {
        printf("Το πρώτο στοιχείο που είναι εκτός σειράς βρίσκεται στη θέση: %d\n", result);
    }

    return 0;
}
```

Συμπληρωματικά στοιχεία Πίνακα

Δύο στοιχεία ενός πίνακα λέγονται **συμπληρωματικά** όταν το άθροισμά τους ισούται με μια συγκεκριμένη σταθερή τιμή: A[i] + A[j] = S

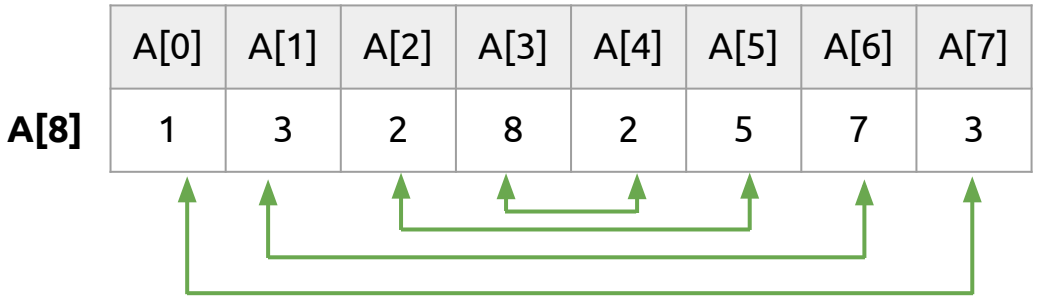
Έστω ο πίνακας:

	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
A[8]	1	3	2	8	2	5	7	3

Ψάχνουμε τα να δούμε αν είναι συμπληρωματικά τα ζευγάρια με άθροισμα το S = 10:

A[0] + A[7]
A[1] + A[6]
A[2] + A[5]
A[3] + A[4]

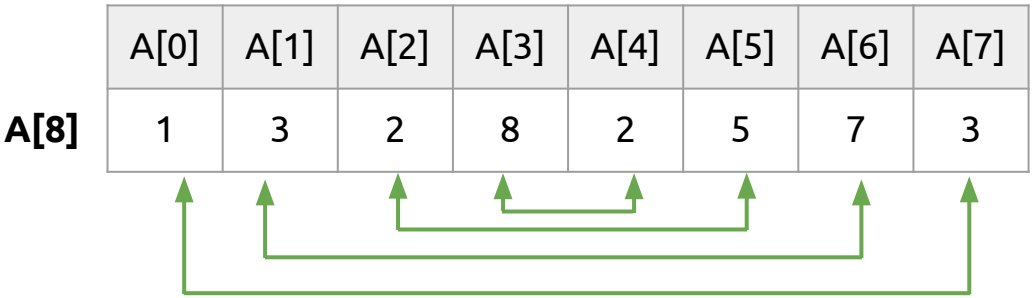
Συμπληρωματικά στοιχεία Πίνακα



Ψάχνουμε τα να δούμε αν είναι συμπληρωματικά τα ζευγάρια με άθροισμα το S = 10:

A[0] + A[7]
A[1] + A[6]
A[2] + A[5]
A[3] + A[4]

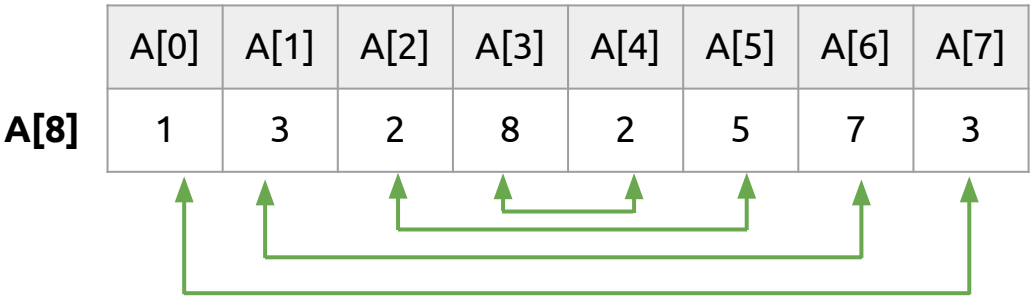
Συμπληρωματικά στοιχεία Πίνακα



Ψάχνουμε τα να δούμε αν είναι συμπληρωματικά τα ζευγάρια με άθροισμα το S = 10:

Πόσους ελέγχους θα κάνουμε;

Συμπληρωματικά στοιχεία Πίνακα

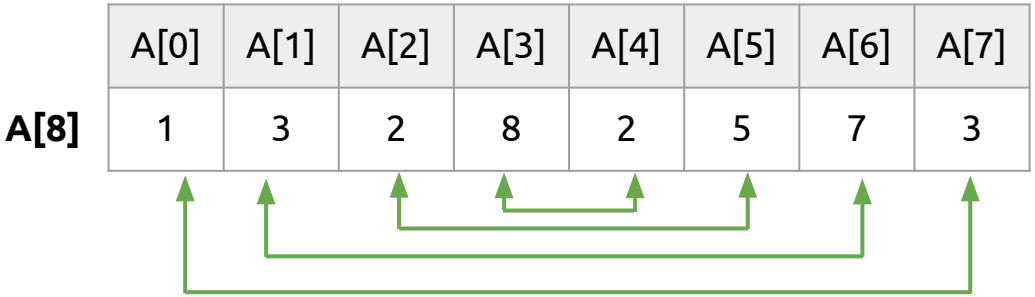


Ψάχνουμε τα να δούμε αν είναι συμπληρωματικά τα ζευγάρια με άθροισμα το S = 10:

Πόσους ελέγχους θα κάνουμε;

A[0] ⟷ A[7]
A[1] ⟷ A[6]
A[2] ⟷ A[5]
A[3] ⟷ A[4]

Συμπληρωματικά στοιχεία Πίνακα



Ψάχνουμε τα να δούμε αν είναι συμπληρωματικά τα ζευγάρια με άθροισμα το S = 10:

A[0] ⟷ A[7]
A[1] ⟷ A[6]
A[2] ⟷ A[5]
A[3] ⟷ A[4]

```
int size = 8;
for (int i = 0; i < size / 2; i++) {
  if (A[i] + A[size - 1 - i] == 10)
    printf("Συμπληρωματικά %d + %d = 10\n", A[i], A[size - 1 - i]);
}
```

Άσκηση 3.2

Να γραφεί συνάρτηση που θα βρίσκει και θα επιστρέφει τα συμπληρωματικά στοιχεία ενός μονοδιάστατου πίνακα `n` θέσεων.

Συμπληρωματικά είναι εκείνα τα στοιχεία του πίνακα που βρίσκονται στις θέσεις `0` και `n-1`, `1` και `n-2`, `2` και `n-3` κλπ. και το άθροισμά τους είναι ίσο του `99`.

Στη συνέχεια να γραφεί πρόγραμμα που θα γεμίζει έναν μονοδιάστατο πίνακα `100` θέσεων με ακέραιους θετικούς αριθμούς από `0-99`
(θα γίνεται έλεγχος κατά την εισαγωγή, εφόσον χρησιμοποιηθεί η `scanf`),
θα καλεί τη συνάρτηση και θα εμφανίζει τα αποτελέσματα.

Άσκηση 3.2 - find_complementary


```c
#include <stdio.h>

void find_complementary(int *arr, int size) {
   printf("Συμπληρωματικά στοιχεία που αθροίζουν στο 99:\n");
   for (int i = 0; i < size / 2; i++) {
       if (arr[i] + arr[size - 1 - i] == 99) {
           printf("Θέσεις %d και %d: %d + %d = 99\n", i, size - 1 - i, arr[i],
                  arr[size - 1 - i]);
       }
   }
}
```

Άσκηση 3.2 - main


```c
int main() {
    int arr[100];

    printf("Εισάγετε 100 θετικούς ακεραίους από 0 έως 99:\n");
    for (int i = 0; i < 100; i++) {
        do {
            printf("Στοιχείο %d: ", i);
            scanf("%d", &arr[i]);
        } while (arr[i] < 0 || arr[i] > 99);
    }

    find_complementary(arr, 100);

    return 0;
}
```

Άσκηση 3.3

Να γραφεί μια συνάρτηση με όνομα `addinnew()` που θα δέχεται ως παραμέτρους 2 μονοδιάστατους πίνακες του ίδιου μεγέθους.
Η συνάρτηση θα προσθέτει τα αντίστοιχα στοιχεία των δύο πινάκων και θα τα τοποθετεί σε τρίτο (νέο) πίνακα της ίδιας διάστασης.

Στη συνέχεια να γραφεί η συνάρτηση main() όπου:

- Θα καταχωρούνται τα στοιχεία σε 2 μονοδιάστατους πίνακες ακεραίων αριθμών `max` (`max`=γνωστό) σε πλήθος θέσεων, με χρήση τυχαίων αριθμών στο διάστημα `1-50`.

- Θα καλείται η συνάρτηση `addinnew()`

Άσκηση 3.3 - addNew


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

#define MAX 10 // Ορισμός μεγέθους πίνακα

void addNew(int *A, int *B, int *result, int size) {
   for (int i = 0; i < size; i++) {
       result[i] = A[i] + B[i];
   }
}
```

Άσκηση 3.3 - main


```c
int main() {
    int A[MAX], B[MAX], result[MAX];
    srand(time(0));

    printf("Πίνακας A: ");
    for (int i = 0; i < MAX; i++) {
        A[i] = rand() % 50 + 1;
        printf("%d ", A[i]);
    }
    printf("\n");
```

```c
    printf("Πίνακας B: ");
    for (int i = 0; i < MAX; i++) {
        B[i] = rand() % 50 + 1;
        printf("%d ", B[i]);
    }
    printf("\n");

    addNew(A, B, result, MAX);

    printf("Νέος Πίνακας: ");
    for (int i = 0; i < MAX; i++) {
        printf("%d ", result[i]);
    }
    printf("\n");
    return 0;
}
```

Άσκηση 3.4

Να γραφεί μια συνάρτηση με όνομα `final` και ορίσματα :

- ένα μονοδιάστατο πίνακα ακεραίων αριθμών `n` θέσεων
- έναν ακέραιο αριθμό `k`

Η συνάρτηση θα δημιουργεί έναν νέο πίνακα που θα περιέχει όλα τα στοιχεία του αρχικού πίνακα των οποίων το άθροισμα, ξεκινώντας από την αρχή του πίνακα, δεν υπερβαίνει τον αριθμό `k` π.χ. αν ο πίνακας περιέχει τα στοιχεία :

11	7	3	17	9	4	20	15

και k = 50,
Τότε ο νέος πίνακας θα περιέχει τα στοιχεία:

11	7	3	17	9

(είναι 11+7+3+17+9+4=51 >50
και το τελευταίο στοιχείο που ελέγχεται είναι το 4 αλλά δεν περιλαμβάνεται στο νέο πίνακα)

Άσκηση 3.4

Στη συνέχεια να γραφεί ένα πρόγραμμα που :

- θα δημιουργεί ένα πίνακα θετικών ακεραίων τυχαίων διψήφιων αριθμών μεγέθους `max` (`max` = γνωστό)
- θα καλεί τη συνάρτηση `final` με παραμέτρους τον πίνακα των τυχαίων αριθμών μεγέθους `max` και μια τιμή για την παράμετρο `k`
- θα εμφανίζει τα αποτελέσματα που θα επιστρέψει η συνάρτηση

Άσκηση 3.4 - final


```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX 10 // Ορισμός μεγέθους πίνακα

void final(int *arr, int size, int k, int *result, int *new_size) {
    int sum = 0;
    *new_size = 0;
    for (int i = 0; i < size; i++) {
        if (sum + arr[i] > k)
            break;
        result[*new_size] = arr[i];
        (*new_size)++;
        sum += arr[i];
    }
}
```

Άσκηση 3.4 - main


```c
int main() {
    int arr[MAX], result[MAX], new_size, k;
    srand(time(0));

    printf("Αρχικός πίνακας: ");
    for (int i = 0; i < MAX; i++) {
        arr[i] = rand() % 90 + 10;
        printf("%d ", arr[i]);
    }
    printf("\nΔώστε τιμή για το k: ");
    scanf("%d", &k);
```

```c
    final(arr, MAX, k, result, &new_size);

    printf("Νέος πίνακας: ");
    for (int i = 0; i < new_size; i++) {
        printf("%d ", result[i]);
    }
    printf("\n");
    return 0;
}
```

Συνάρτηση Εκτύπωσης Πίνακα

Θέλουμε να γράψουμε μια γενική συνάρτηση που θα εκτυπώνει τα στοιχεία ενός πίνακα

```c
#include <stdio.h>

void printArray(int *arr, int size) {
...
}

int main() {
   int arr[] = {1, 2, 3, 4, 5};
   int size = sizeof(arr) / sizeof(arr[0]);
   printArray(arr, size);
   return 0;
}
```

Συνάρτηση Εκτύπωσης Πίνακα

Θέλουμε να γράψουμε μια γενική συνάρτηση που θα εκτυπώνει τα στοιχεία ενός πίνακα

```c
#include <stdio.h>

void printArray(int *arr, int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    printArray(arr, size);
    return 0;
}
```

Συνάρτηση Αναζήτησης σε Πίνακα

Θέλουμε να γράψουμε συνάρτηση που ελέγχει αν ένας αριθμός υπάρχει ήδη σε έναν πίνακα:

```c
#include <stdio.h>

int existsInArray(int *arr, int size, int num) {
     ...
}

int main() {
   int arr[] = {1, 2, 3, 4, 5};
   int size = sizeof(arr) / sizeof(arr[0]);

   printf("%d\n", existsInArray(arr, size, 3)); // 1
   printf("%d\n", existsInArray(arr, size, 6)); // 0
     return 0;
}
```

Συνάρτηση Αναζήτησης σε Πίνακα

Θέλουμε να γράψουμε συνάρτηση που ελέγχει αν ένας αριθμός υπάρχει ήδη σε έναν πίνακα:

```
int existsInArray(int *arr, int size, int num) {

   for (int i = 0; i < size; i++) {
       if (arr[i] == num) {
           return 1; // Υπάρχει ήδη
       }
   }
   return 0; // Δεν υπάρχει
}
```

Συνάρτηση προσθήκης σε Πίνακα

Θέλουμε να γράψουμε συνάρτηση που να προσθέτει έναν αριθμός σε πίνακα, μόνο όταν αυτός δεν υπάρχει ήδη:

```
void addIfNotExists(int *arr, int *size, int num) {
     ...
}

int main() {
   int result[10];
   int size = 0;

   addIfNotExists(result, &size, 2); // Θα αλλάξει το size !
   addIfNotExists(result, &size, 4); // Θα αλλάξει το size !
   addIfNotExists(result, &size, 4); // Δεν θα προστεθεί
   printArray(result, size);
   return 0;
}
```

Συνάρτηση προσθήκης σε Πίνακα

Θέλουμε να γράψουμε συνάρτηση που να προσθέτει έναν αριθμός σε πίνακα, μόνο όταν αυτός δεν υπάρχει ήδη:

```
void addIfNotExists(int *arr, int *size, int num) {

   if (!existsInArray(arr, *size, num)) {
       arr[*size] = num;
       (*size)++;
   }
}
```

Συνάρτηση τυχαίου Πίνακα

Θέλουμε να γράψουμε συνάρτηση που δημιουργεί έναν πίνακα μονοψήφιων τυχαίων αριθμών.:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX 10
void generateRandomArray(int *arr, int size) {
     ...
}
int main() {
   srand(time(0));
   int arr[MAX];
   generateRandomArray(arr, MAX);
   printArray(arr, MAX);
   return 0;
}
```

Συνάρτηση τυχαίου Πίνακα

Θέλουμε να γράψουμε συνάρτηση που δημιουργεί έναν πίνακα μονοψήφιων τυχαίων αριθμών.:

```
void generateRandomArray(int *arr, int size) {

    for (int i = 0; i < size; i++) {
        arr[i] = rand() % 10; // Μονοψήφιοι αριθμοί (0-9)
    }
}
```

Άσκηση 3.5

Να γραφεί μια συνάρτηση με όνομα `disparate` και ορίσματα :

- ένα μονοδιάστατο πίνακα ακεραίων αριθμών `n` θέσεων

Η συνάρτηση :

Θα δημιουργεί ένα νέο πίνακα που θα περιλαμβάνει όλα στοιχεία του αρχικού πίνακα αλλά μόνον μία φορά το καθένα, δηλ. δεν θα περιέχει 2η φορά το ίδιο στοιχείο. Π.χ. αν ο αρχικός πίνακας περιέχει τα στοιχεία `2,4,5,9,4,3,5,2` ο νέος πίνακας θα περιέχει μόνο τα `2,4,5,9,3`.

(ΥΠΟΔΕΙΞΗ : κάθε φορά που ένα στοιχείο πρόκειται να εισαχθεί στον νέο πίνακα να ελέγχετε αν υπάρχει ήδη σε αυτόν).

Στη συνέχεια να γραφεί ένα πρόγραμμα που :

- Θα δημιουργεί ένα πίνακα ακεραίων μονοψήφιων τυχαίων αριθμών μεγέθους `max` (`max` = γνωστό)
- Θα καλεί τη συνάρτηση `disparate` με παράμετρο εισόδου τον πίνακα των τυχαίων αριθμών
- Θα εμφανίζει τα αποτελέσματα που θα επιστρέψει η συνάρτηση.

Άσκηση 3.5 - main


```c
int main() {
    srand(time(0));

    int arr[MAX], result[MAX], new_size;
    generateRandomArray(arr, MAX);

    printf("Αρχικός πίνακας: ");
    printArray(arr, MAX);

    disparate(arr, MAX, result, &new_size);

    printf("Νέος πίνακας (μοναδικές τιμές): ");
    printArray(result, new_size);

    return 0;
}
```

Θα χρειαστούμε επίσης, όλες τις συναρτήσεις από τις προηγούμενες διαφάνειες *(δεν τις ξαναγράφουμε εδώ)*

Άσκηση 3.5 - disparate


```
void disparate(int *arr, int size, int *result, int *new_size) {

    *new_size = 0;

    for (int i = 0; i < size; i++) {
        if (!existsInArray(result, *new_size, arr[i])) {
            result[*new_size] = arr[i];
            (*new_size)++;
        }
    }
}
```