Δ.Π.Θ Δομημένος Προγραμματισμός

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ & ΔΙΟΙΚΗΣΗΣ

Δυναμική Παραχώρηση Μνήμης

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

Στατική Μνήμη

- Χρησιμοποιείται όταν οι απαιτήσεις σας για τη χρήση μνήμης, είναι γνωστές κατά τη μεταγλώττιση ενός προγράμματος.
- Μετά τη μεταγλώττιση, ο μεταγλωττιστής, μπορεί να προβλέψει ακριβώς την ποσότητα της μνήμης που θα χρειαστούμε.
- Κατά την εκτέλεση ενός προγράμματος, η ποσότητα μνήμης που μας έχει παραχωρηθεί, δεν μπορεί ούτε να αυξηθεί, ούτε να μειωθεί.
- Η διαχείρισής της γίνεται αυτόματα. Δεν απαιτούνται ενέργειες από τον προγραμματιστή.

Δυναμική Μνήμη

- Χρησιμοποιείται όταν οι απαιτήσεις μας σε μνήμη, δεν είναι γνωστές κατά τη φάση της μεταγλώττισης.
- Παραχωρείται κατά τη διάρκεια της εκτέλεσης ενός προγράμματος.
- Κατά την εκτέλεση ενός προγράμματος, η ποσότητα μνήμης που μας έχει παραχωρηθεί, μπορεί και να αυξηθεί, αλλά και να μειωθεί.
- Η διαχείρισή της δεν γίνεται αυτόματα. Ο προγραμματιστής αναλαμβάνει αυτό το ρόλο.

Διαχείριση Κύριας Μνήμης

Μνήμη RAM

Σωρός
(Heap)

Στοίβα
(Stack)

Καθολικές μεταβλητές (global)

Συγγραφή κώδικα (text)

Δυναμική Μνήμη

Στατική Μνήμη

Διαχείριση Δυναμικής Μνήμης

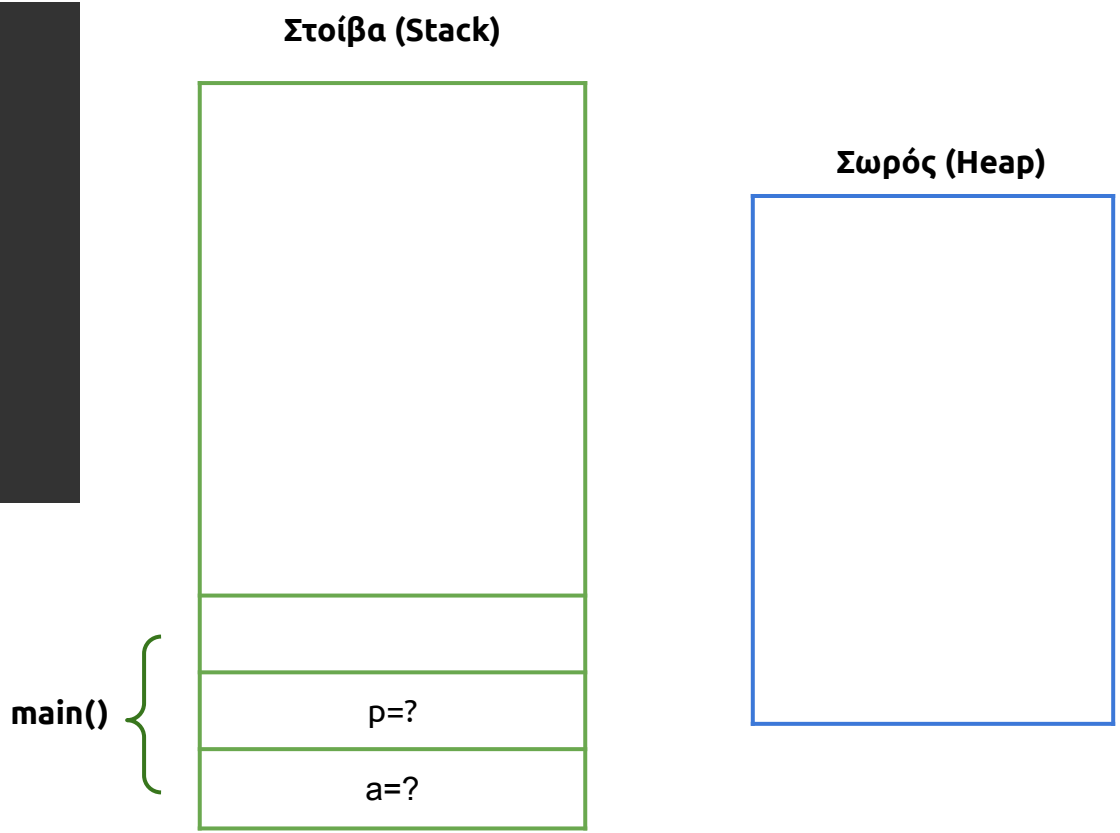
Συναρτήσεις

- `malloc, calloc`: για να δεσμεύσουμε δυναμικά μια ποσότητα μνήμης.
- `free`: για να ελευθερώσουμε μια ποσότητα μνήμης, που δεν χρειαζόμαστε πλέον.
- `realloc`: για να αυξήσουμε ή να μειώσουμε το μέγεθος ενός ήδη δεσμευμένου μπλοκ μνήμης.

Βρίσκονται στη βιβλιοθήκη: `stdlib.h`

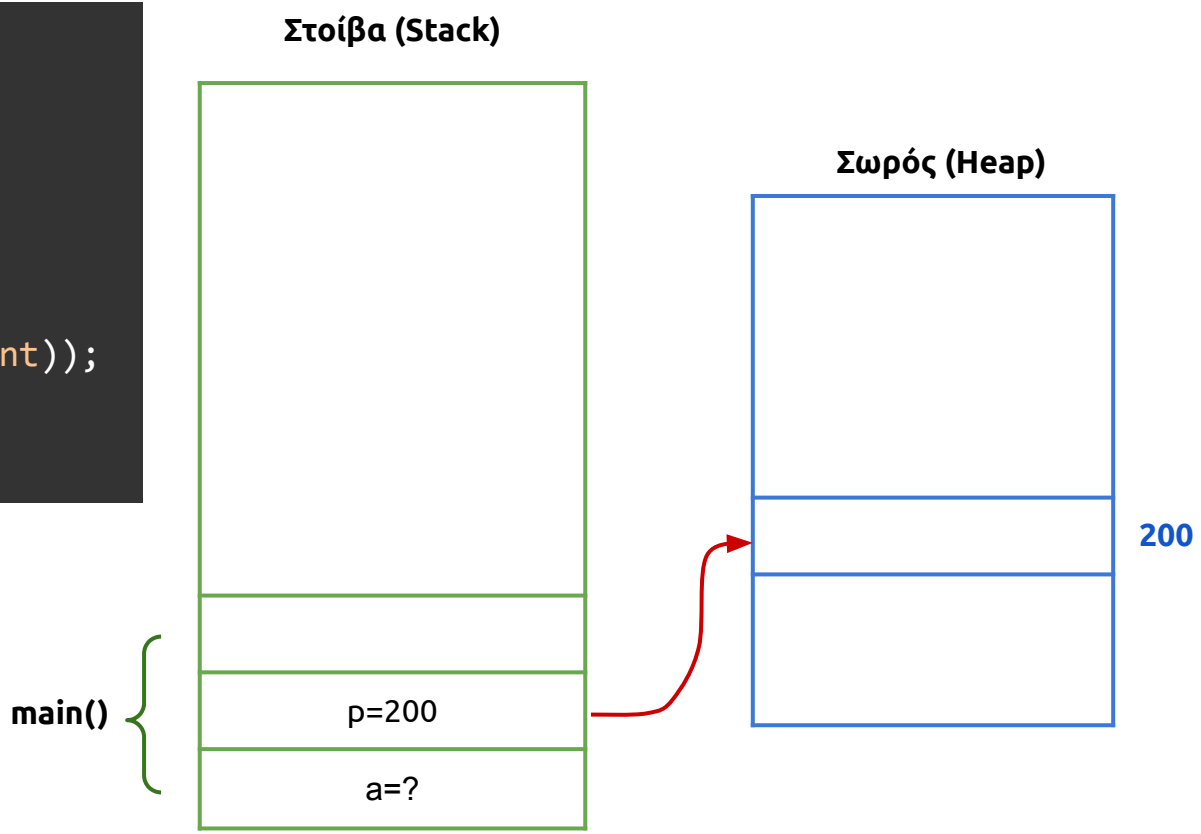
Δυναμική Παραχώρηση Μνήμης


```c
#include <stdio.h>
#include <stdlib.h>
int main(){

    int a;
    int *p;

}
```



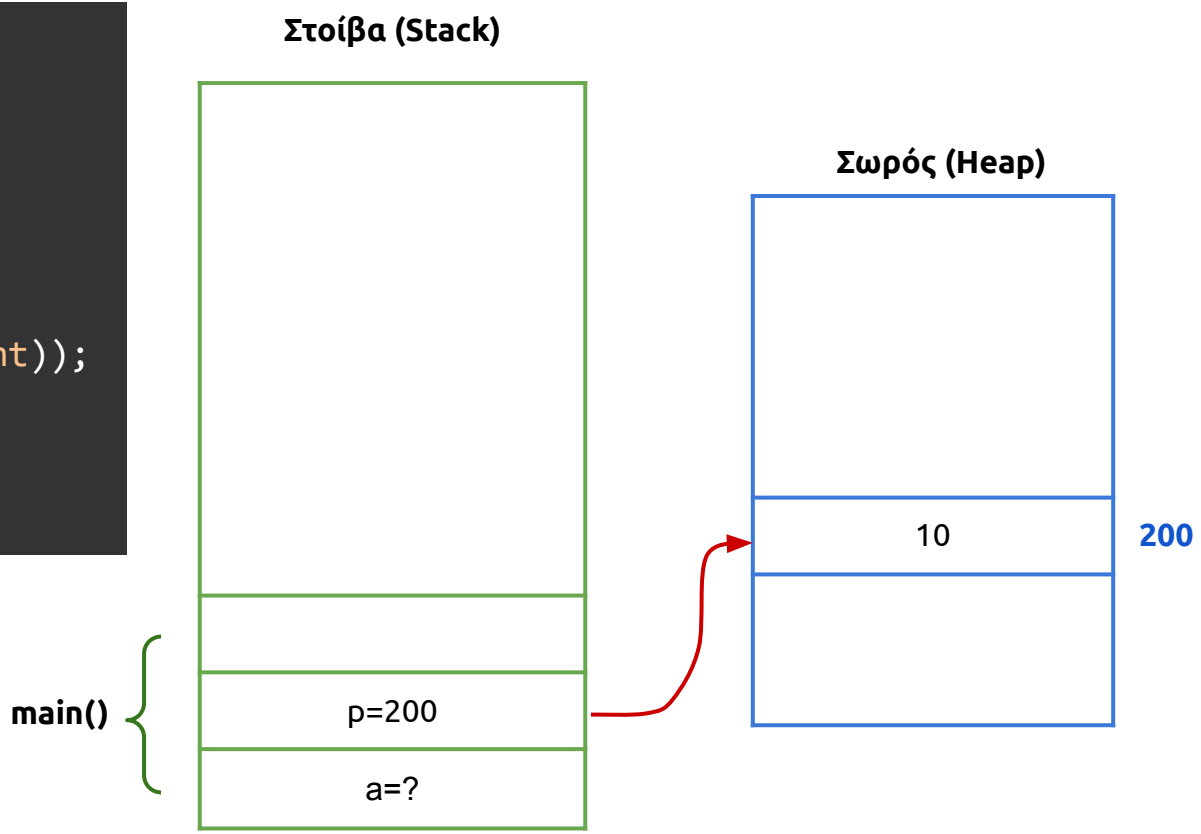
Δυναμική Παραχώρηση Μνήμης


```c
#include <stdio.h>
#include <stdlib.h>
int main(){

    int a;
    int *p;
    p = (int*)malloc(sizeof(int));
}
```



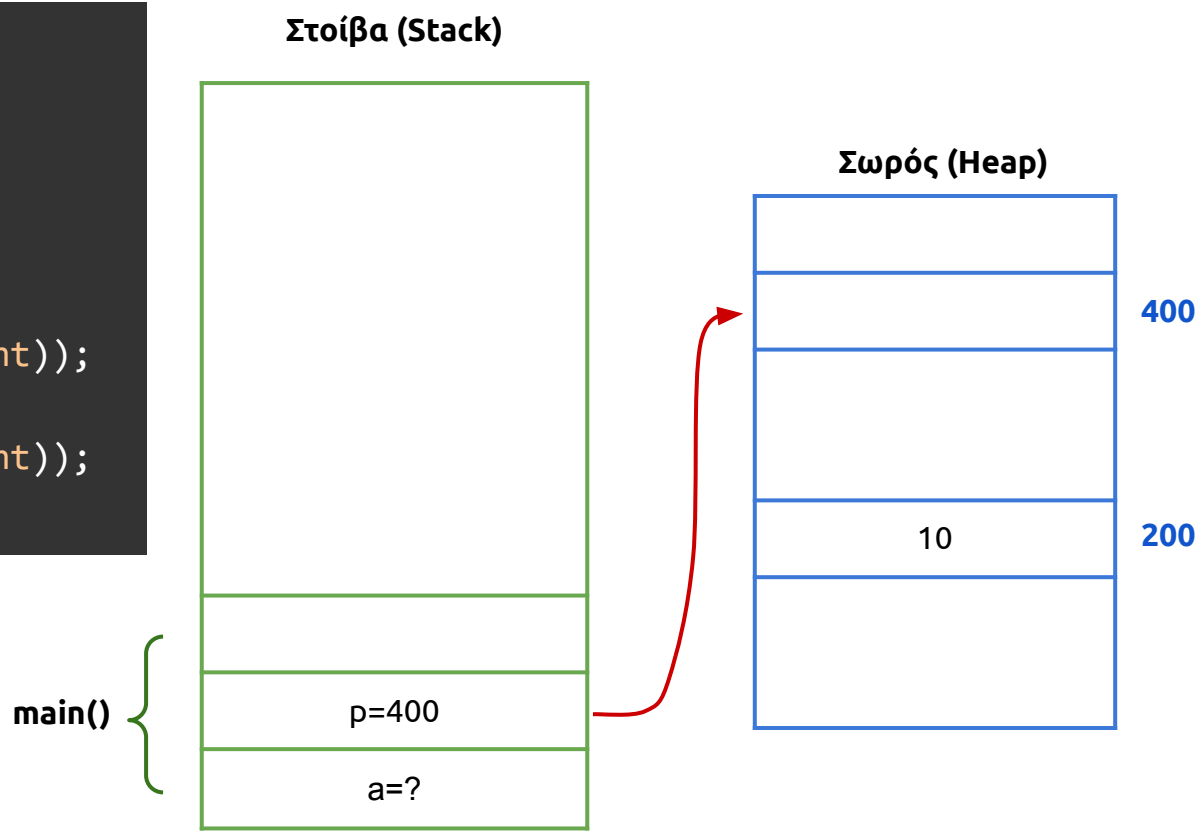
Δυναμική Παραχώρηση Μνήμης


```c
#include <stdio.h>
#include <stdlib.h>
int main(){

    int a;
    int *p;
    p = (int*)malloc(sizeof(int));
    *p = 10;
}
```



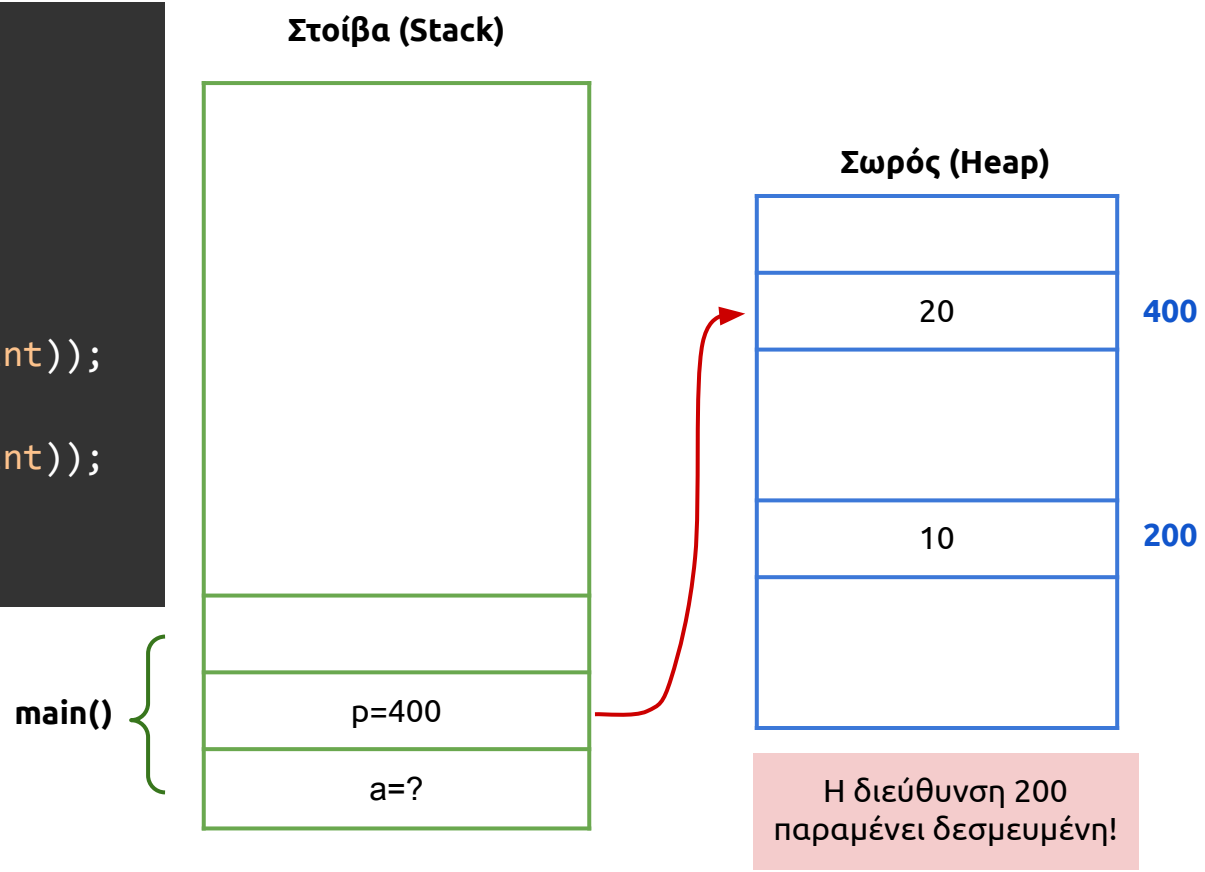
Δυναμική Παραχώρηση Μνήμης


```c
#include <stdio.h>
#include <stdlib.h>
int main(){

    int a;
    int *p;
    p = (int*)malloc(sizeof(int));
    *p = 10;
    p = (int*)malloc(sizeof(int));
}
```



Δυναμική Παραχώρηση Μνήμης

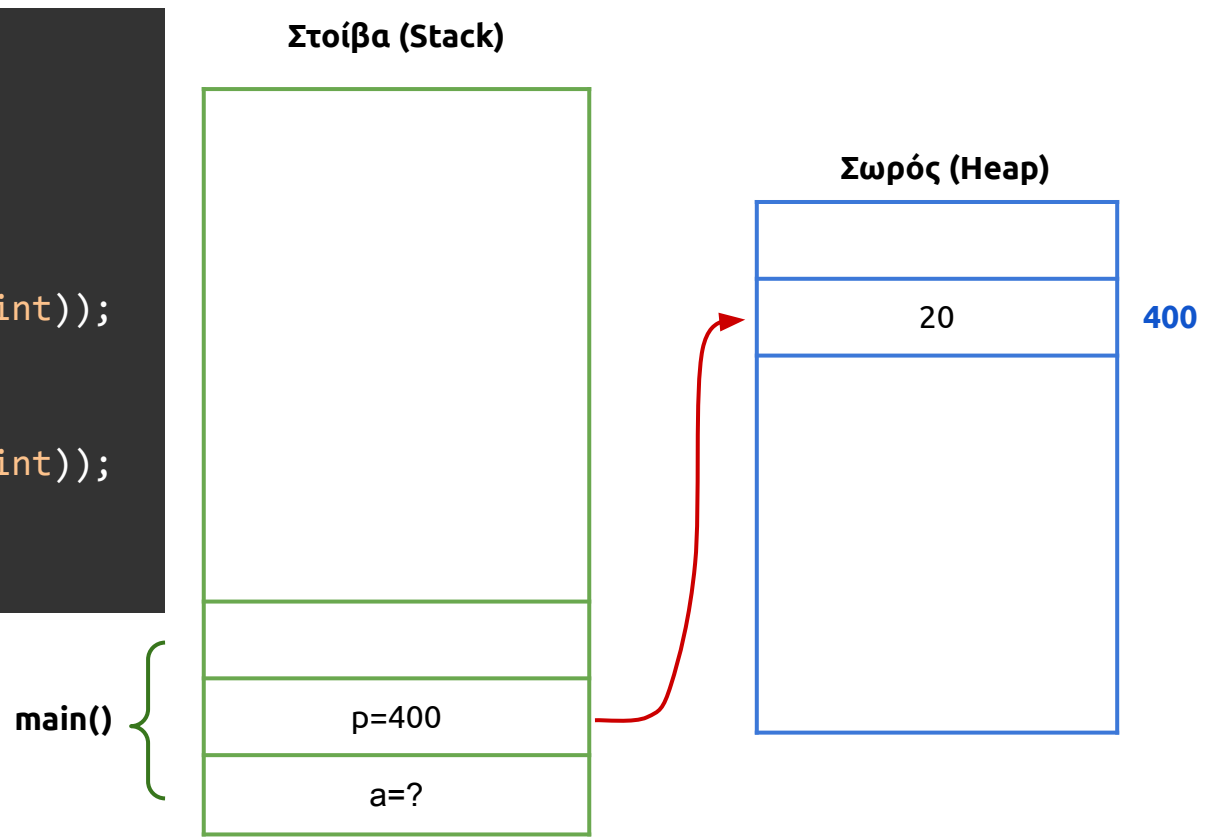

```c
#include <stdio.h>
#include <stdlib.h>
int main(){

    int a;
    int *p;
    p = (int*)malloc(sizeof(int));
    *p = 10;
    p = (int*)malloc(sizeof(int));
    *p = 20;
}
```



Δυναμική Παραχώρηση Μνήμης

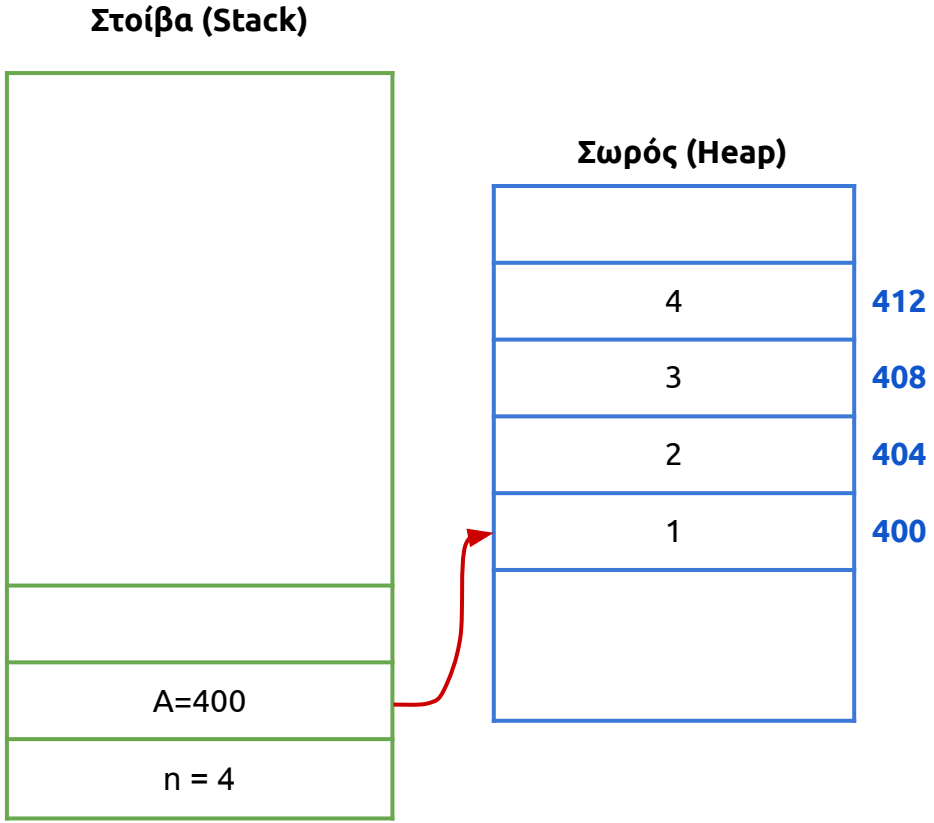

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    int a;
    int *p;
    p = (int*)malloc(sizeof(int));
    *p = 10;
    free(p);
    p = (int*)malloc(sizeof(int));
    *p = 20;
}
```

Ο προγραμματιστής πρέπει να καθαρίζει τη μνήμη Heap



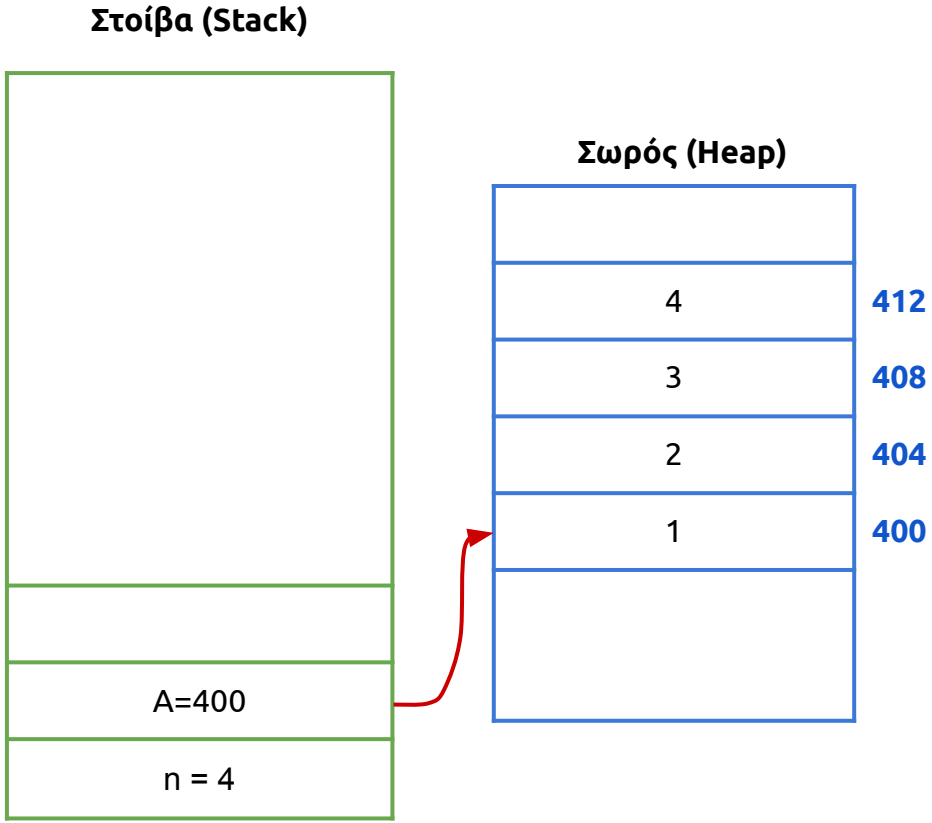
Δυναμικοί Πίνακες - malloc


```c
#include <stdio.h>
#include <stdlib.h>
int main(){

 int n;
 printf("Δώστε το μέγεθος του πίνακα: ");
 scanf("%d", &n);

 int *A = (int*)malloc(n * sizeof(int));
 for(int i = 0; i < n; i++)
    A[i] = i + 1;

 for(int i = 0; i < n; i++)
    printf("%d ", A[i]);
}
```

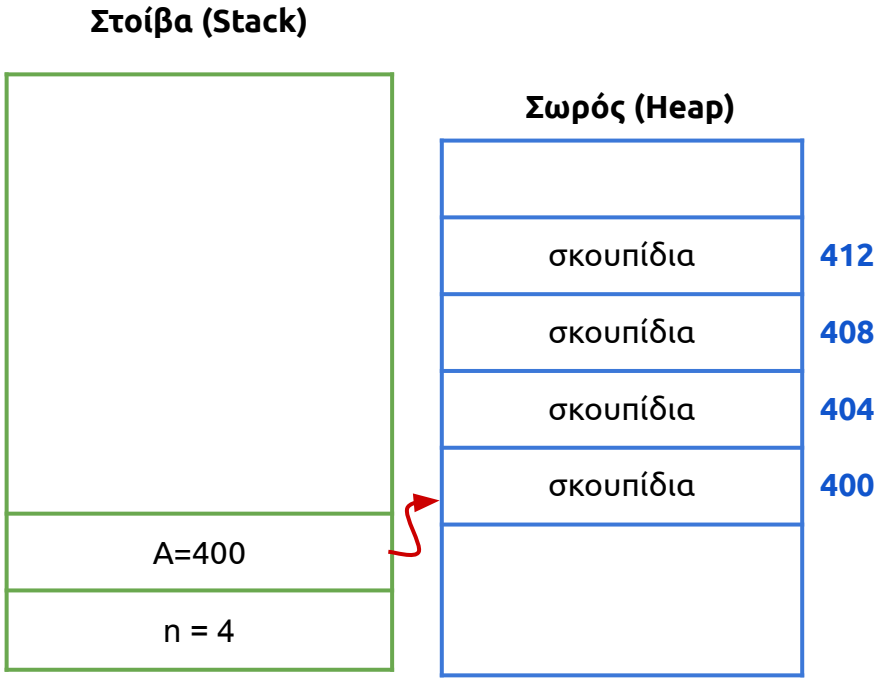


Δυναμικοί Πίνακες - calloc

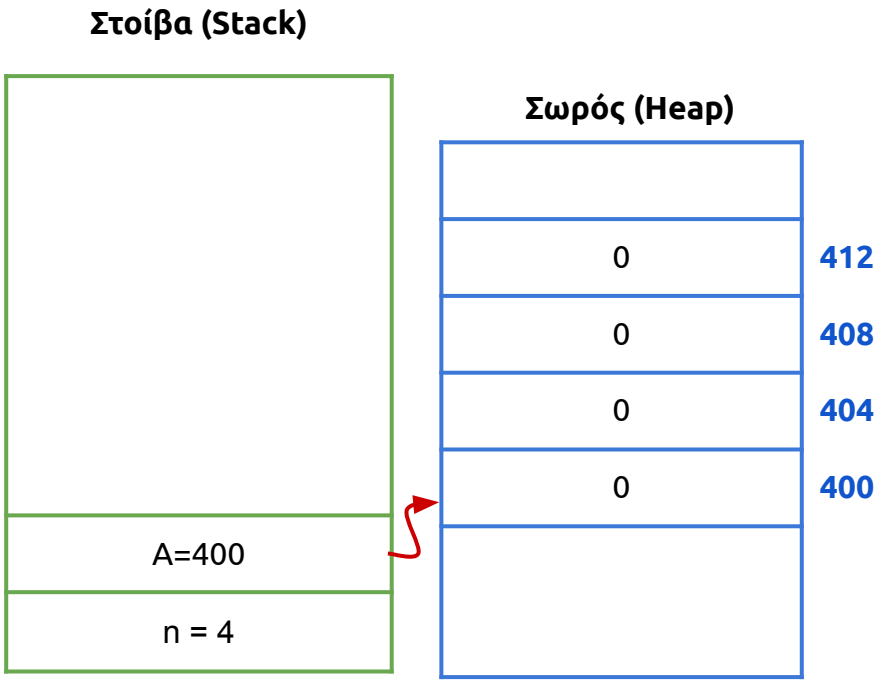

```c
#include <stdio.h>
#include <stdlib.h>
int main(){

  int n;
  printf("Δώστε το μέγεθος του πίνακα: ");
  scanf("%d", &n);

  int *A = (int*)calloc(n, sizeof(int));
  for(int i = 0; i < n; i++)
     A[i] = i + 1;

  for(int i = 0; i < n; i++)
     printf("%d ", A[i]);
}
```



Διαφορά malloc - calloc



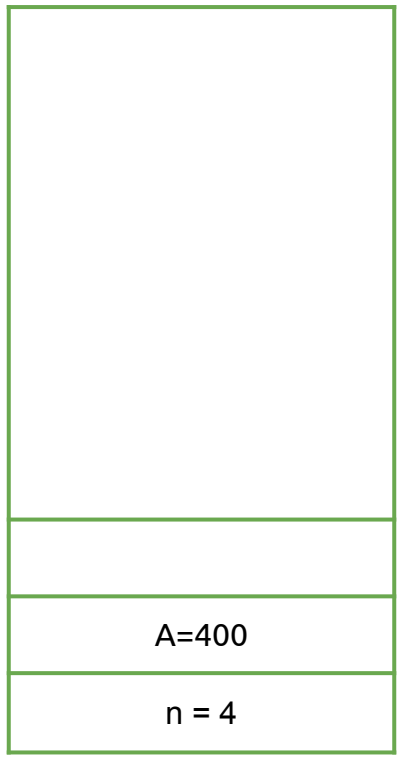

```
int *A = (int*)malloc(n * sizeof(int));
```

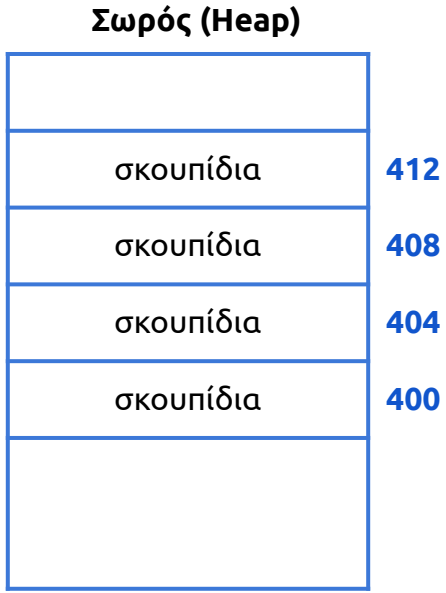



```
int *A = (int*)calloc(n, sizeof(int));
```

Δυναμικοί Πίνακες - free

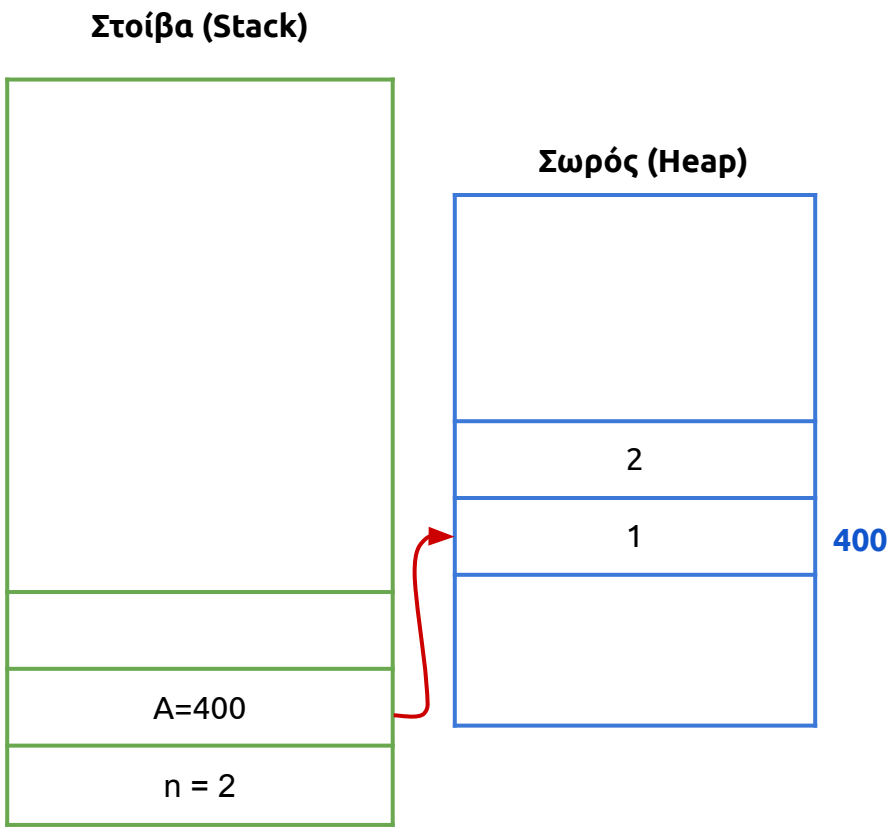

```c
#include <stdio.h>
#include <stdlib.h>
int main(){

  int n;
  printf("Δώστε το μέγεθος του πίνακα: ");
  scanf("%d", &n);

  int *A = (int*)malloc(n * sizeof(int));
    for(int i = 0; i < n; i++)
       A[i] = i + 1;

    free(A);
    for(int i = 0; i < n; i++)
       printf("%d ", A[i]);
}
```





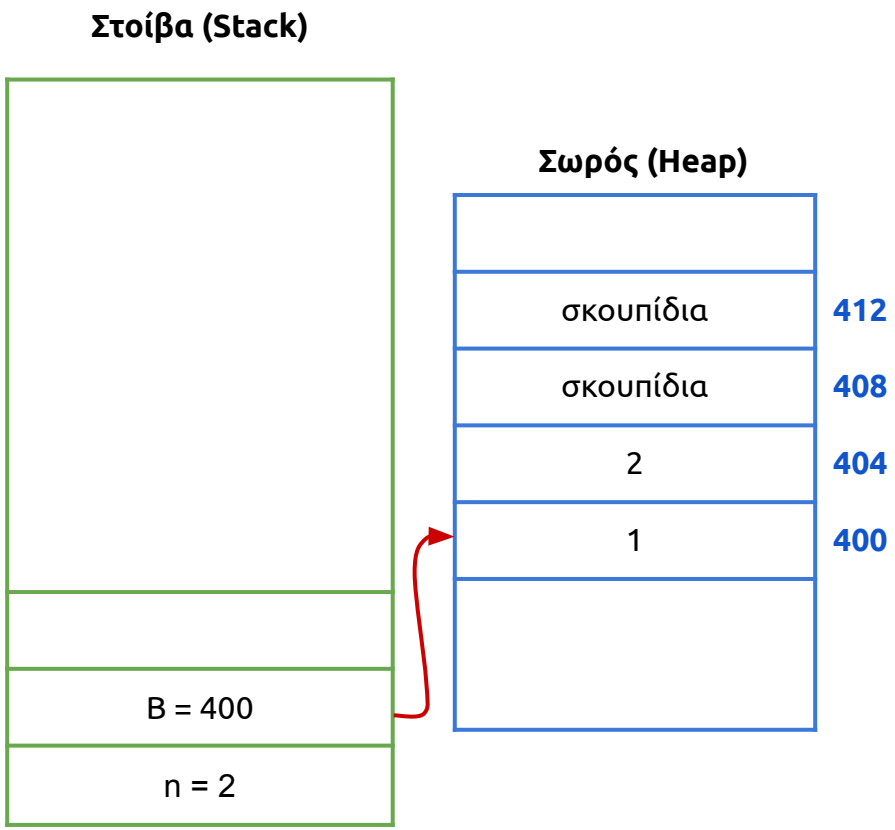
Δυναμικοί Πίνακες - realloc


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

int main(){

int n;
printf("Δώστε το μέγεθος του πίνακα: ");
scanf("%d", &n);

int *A = (int*)malloc(n * sizeof(int));
for(int i = 0; i < n; i++)
    A[i] = i + 1;
int *B = (int*)realloc(A, 2 * n * sizeof(int));
for(int i = 0; i < 2 * n; i++)
    printf("%d\n", B[i]);
}
```



Δυναμικοί Πίνακες - realloc


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

int main(){

int n;
printf("Δώστε το μέγεθος του πίνακα: ");
scanf("%d", &n);

int *A = (int*)malloc(n * sizeof(int));
for(int i = 0; i < n; i++)
    A[i] = i + 1;
int *B = (int*)realloc(A, 2 * n * sizeof(int));
for(int i = 0; i < 2 * n; i++)
    printf("%d\n", B[i]);
}
```



Παράδειγμα 1

Τι θα εκτυπώσει το παρακάτω πρόγραμμα;

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

int main() {
    int *ptr = (int *)malloc(sizeof(int));
    *ptr = 10;
    free(ptr);
    printf("%d\n", *ptr);
    return 0;
}
```

Παράδειγμα

Τι θα εκτυπώσει το παρακάτω πρόγραμμα;

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

int main() {
    int *ptr = (int *)malloc(sizeof(int));
    *ptr = 10;
    free(ptr);
    printf("%d\n", *ptr);
    return 0;
}
```

σκουπίδια

Τι συμβαίνει όταν προσπαθούμε να προσπελάσουμε μνήμη που έχει αποδεσμευτεί;

Άσκηση 4.1

Να γραφεί πρόγραμμα που να βρίσκει και να εμφανίζει το μεγαλύτερο από `n` σε πλήθος στοιχείων ενός πίνακα, χρησιμοποιώντας δυναμική παραχώρηση μνήμης.

Η τιμή του `n` να καθορίζεται κατά τη διάρκεια της εκτέλεσης του προγράμματος μέσω της scanf, με τον περιορισμό `n > 0`.

Να υλοποιήσετε τη λύση χρησιμοποιώντας:

(α) τη συνάρτηση `malloc()` και
(β) τη συνάρτηση `calloc()`.

Άσκηση 4.1 - Λύση


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

int main(){
   int n, max;
   do{
     printf("Δώστε το μέγεθος του πίνακα: ");
     scanf("%d", &n);
   }while(n <= 0);

   Int *A = (int*)malloc(n * sizeof(int));

   printf("Δώστε τα στοιχεία του πίνακα:\n");
   for(int i = 0; i < n; i++)
      scanf("%d", A+i);
   max = A[0];

   for(int i = 1; i < n; i++){
      if (A[i] > max)
         max = A[i];
   }
   printf("Max = %d\n", max);
   free(A);
   return 0;
}
```

```c
int *A = (int*)calloc(n, sizeof(int));
```

Έλεγχος Παραχώρησης Μνήμης

Η παρακάτω εντολή, ενδεχομένως να μην μπορεί να εκτελεστεί αν η αιτούμενη ποσότητα μνήμης δεν είναι διαθέσιμη:

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

Ασφαλής τερματισμός του προγράμματος:

```
if (A == NULL) {
        printf("Πρόβλημα παραχώρησης μνήμης.\n");
        exit(0);
}
```

ή

```
if (!A) {
        printf("Πρόβλημα παραχώρησης μνήμης.\n");
        exit(0);
}
```

Άσκηση 4.2

Να γραφεί συνάρτηση που να δημιουργεί έναν πίνακα με τυχαίους ακέραιους αριθμούς στο διάστημα `[50-100]`.

Το μέγεθος του πίνακα να καθορίζεται κατά την εκτέλεση του προγράμματος από τη συνάρτηση `main()`, ελέγχοντας παράλληλα ότι είναι θετικός αριθμός.

Η συνάρτηση main(), θέλουμε να εμφανίζει τους τυχαίους αριθμούς.

Άσκηση 4.2 - getRandomNumbers


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

void getRandomNumbers(int size, int low, int high, int *arr) {

    for (int i = 0; i < size; i++)
        arr[i] = rand() % (high - low + 1) + low;
}
```

Άσκηση 4.2 - main


```c
int main() {
    int n;
    do {
        printf("Δώστε το μέγεθος του πίνακα: ");
        scanf("%d", &n);
    } while (n <= 0);

    srand(time(0));
    int *A = (int *)malloc(n * sizeof(int));
    if (A == NULL) {
        printf("Πρόβλημα παραχώρησης μνήμης!\n");
        return 1;
    }
    getRandomNumbers(n, 50, 100, A);
```

```c
    for (int i = 0; i < n; i++)
        printf("%d ", A[i]);
    printf("\n");

    free(A);
    return 0;
}
```

Άσκηση 4.2 - 2η Λύση


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

int *getRandomNumbers(int size, int low, int high) {

    int *arr = (int *)malloc(size * sizeof(int));
    if (!arr) {
        printf("Πρόβλημα παραχώρησης μνήμης!\n");
        return NULL;
    }

    for (int i = 0; i < size; i++)
        arr[i] = rand() % (high - low + 1) + low;
    return arr;
}
```

Άσκηση 4.2 - 2η Λύση


```c
int main() {
    int n;
    do {
        printf("Δώστε το μέγεθος του πίνακα: ");
        scanf("%d", &n);
    } while (n <= 0);
    srand(time(0));

    int *A = getRandomNumbers(n, 50, 100);
    if (A == NULL)
        return 1;
    for (int i = 0; i < n; i++)
        printf("%d ", A[i]);
    free(A);
    return 0;
}
```

Άσκηση 4.3

Να γραφεί συνάρτηση `void remove_min` που να διαγράφει το μικρότερο ή τα μικρότερα σε τιμή στοιχεία ενός μονοδιάστατου αριθμητικού πίνακα `N` θέσεων (`N` = γνωστό).

Στη συνέχεια να γραφεί κατάλληλο πρόγραμμα που να κάνει χρήση αυτής της συνάρτησης.

Άσκηση 4.3 - remove_min


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

int* remove_min(int arr[], int* size)
{
    int min = arr[0];
    int count = 1;

    for (int i = 1; i < *size; i++) {
        if (arr[i] < min) {
            min = arr[i];
            count = 1;
        }
        else if (arr[i] == min)
           count++;
    }
```

```c
int newSize = *size - count;

int* newArr = (int*)malloc(newSize * sizeof(int));

 int index = 0;
 for (int i = 0; i < *size; i++) {
     if (arr[i] != min){
        newArr[index] = arr[i];
        index++;
     }
 }

 *size = newSize;

 return newArr;
}
```

Άσκηση 4.3 - main


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

    int size = sizeof(arr) / sizeof(arr[0]);

    int* newArr = remove_min(arr, &size);

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

    free(newArr);
    return 0;
}
```

Άσκηση 4.4

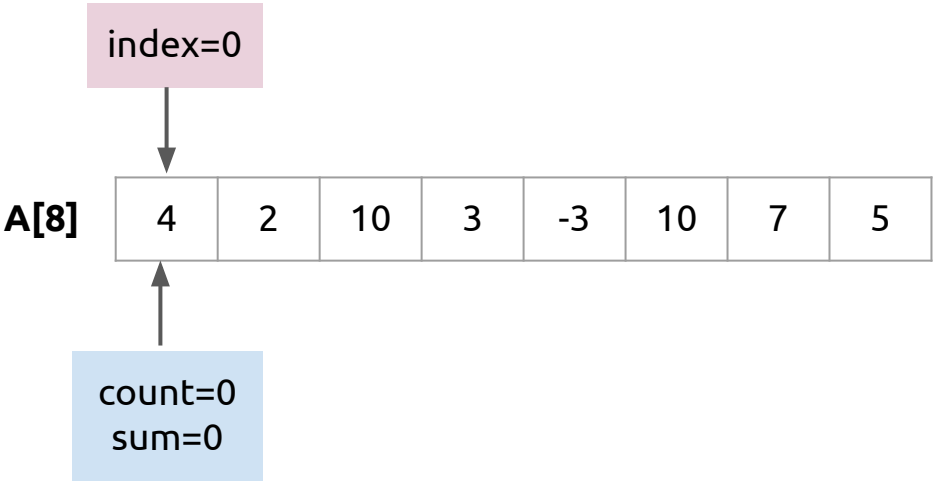
Χρησιμοποιώντας δυναμική παραχώρηση μνήμης, να γραφεί πρόγραμμα που:

1. δημιουργεί έναν μονοδιάστατο, μη ταξινομημένο πίνακα a, μεγέθους n. Η τιμή του n θα καθορίζεται κατά τη διάρκεια της εκτέλεσης του προγράμματος μέσω της scanf, με τον περιορισμό `n > 0`.
2. διαβάζει μία τιμή, έστω s.
3. βρίσκει και εμφανίζει έναν υποπίνακα b του a, (που αποτελείται από συνεχόμενα στοιχεία του a), το άθροισμα των οποίων θα ισούται με s.

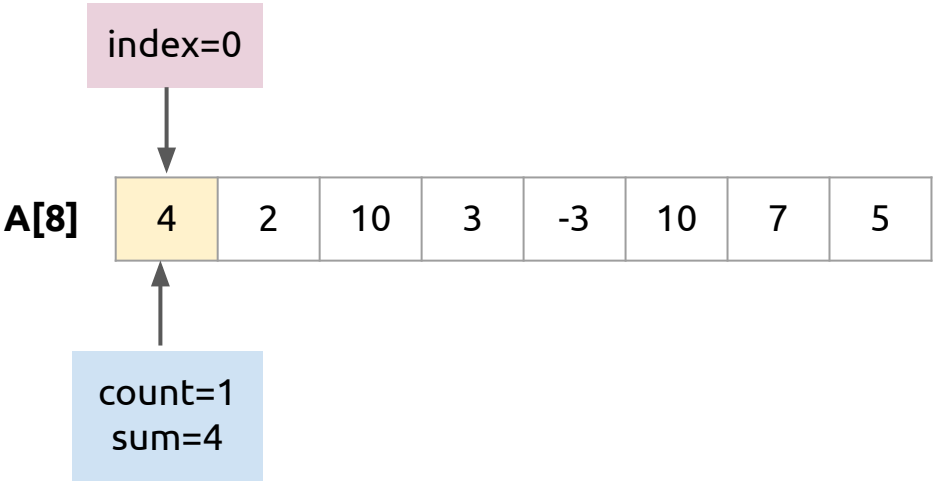
Παράδειγμα:

```
n=8, s=20
a={4, 2, 10, 3, -3, 10, 7, 5}
b={10, 3, -3, 10}
```

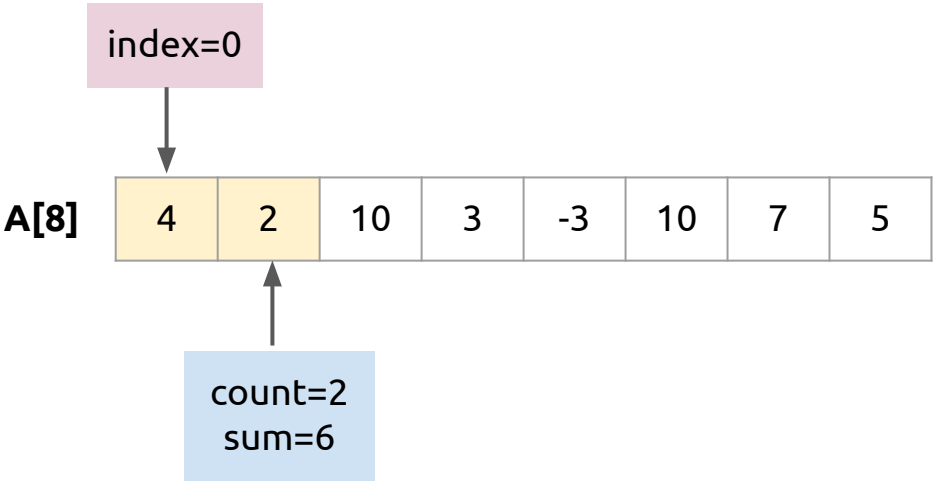
Άσκηση 4.4



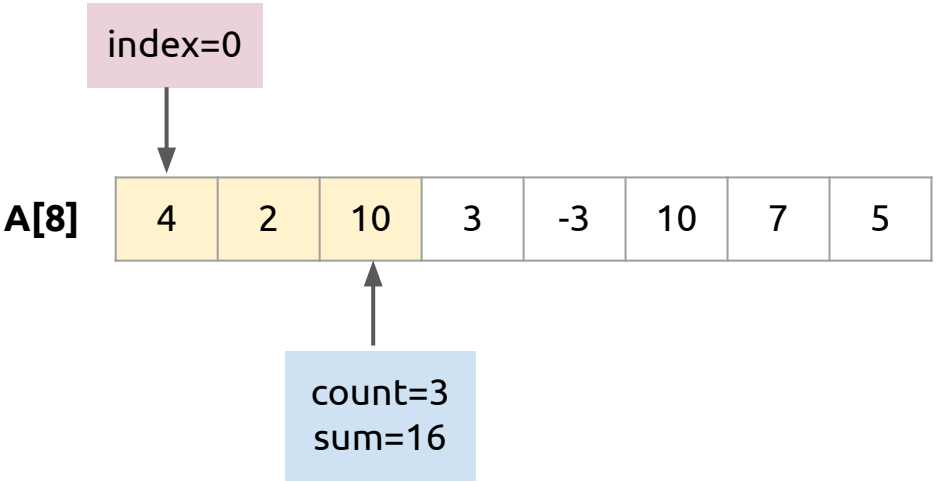
Άσκηση 4.4



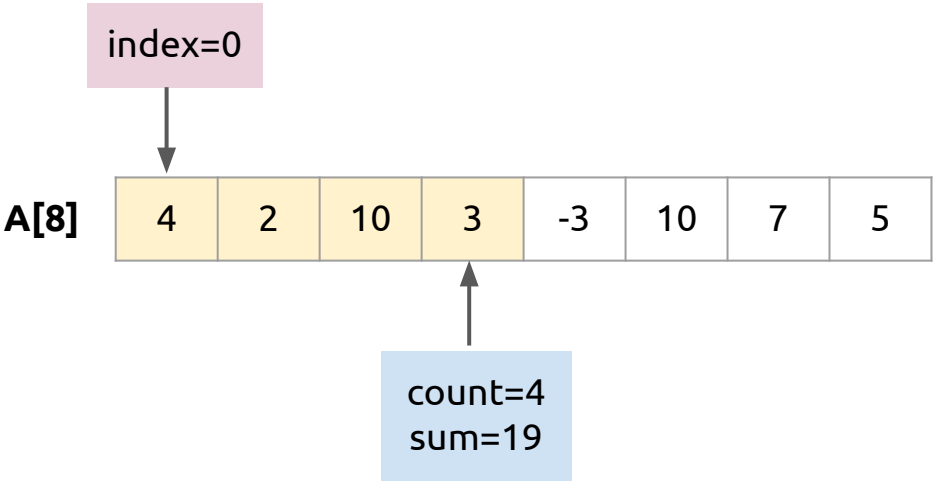
Άσκηση 4.4



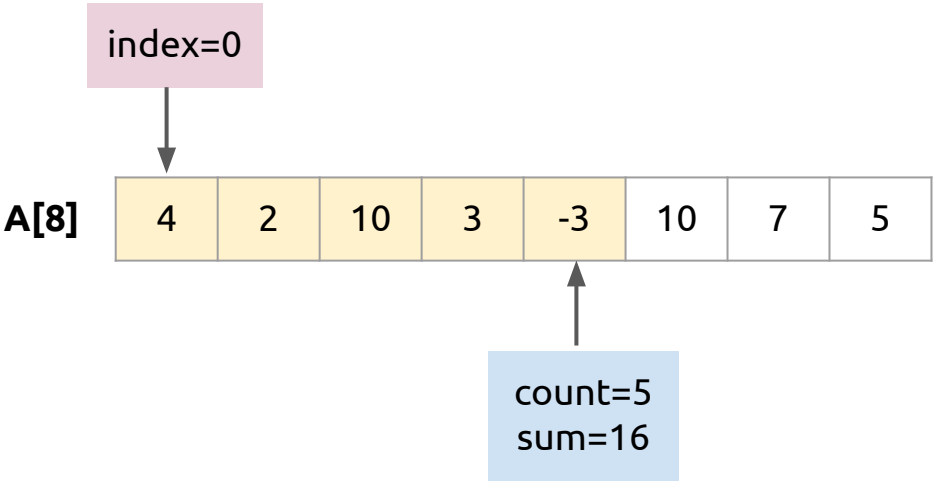
Άσκηση 4.4



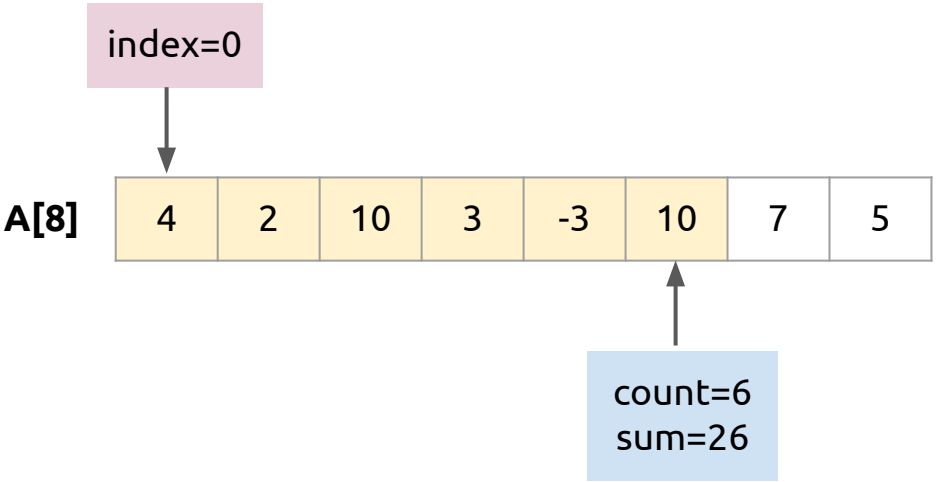
Άσκηση 4.4



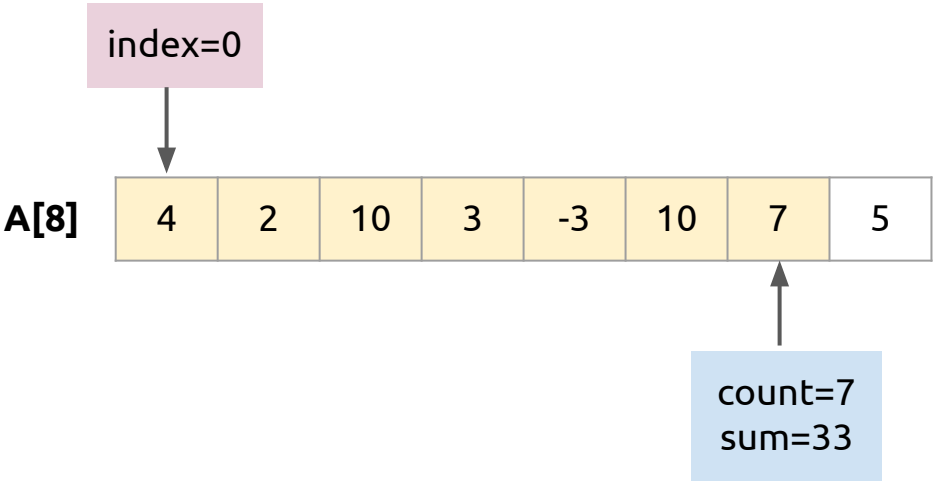
Άσκηση 4.4



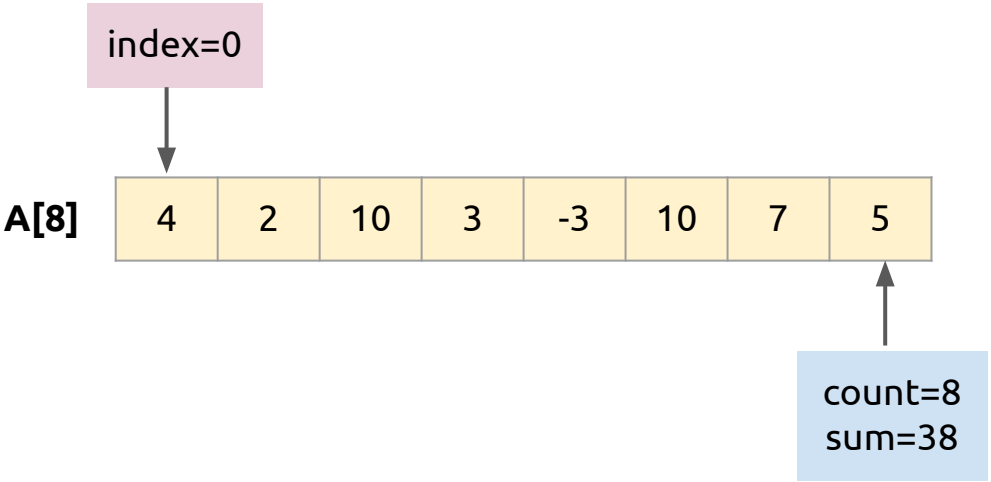
Άσκηση 4.4



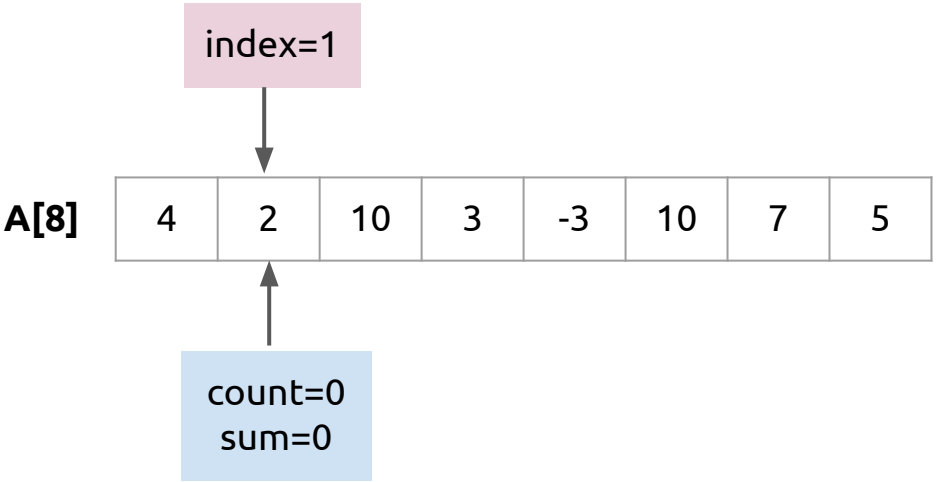
Άσκηση 4.4



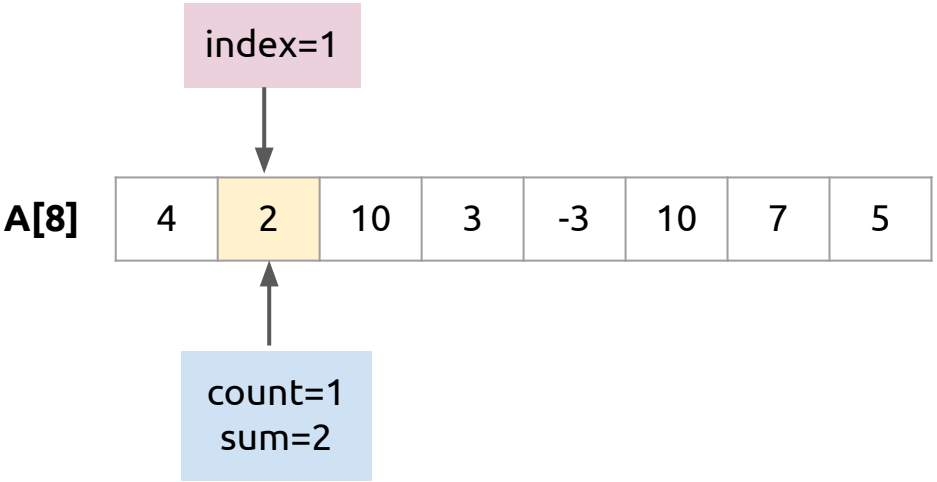
Άσκηση 4.4



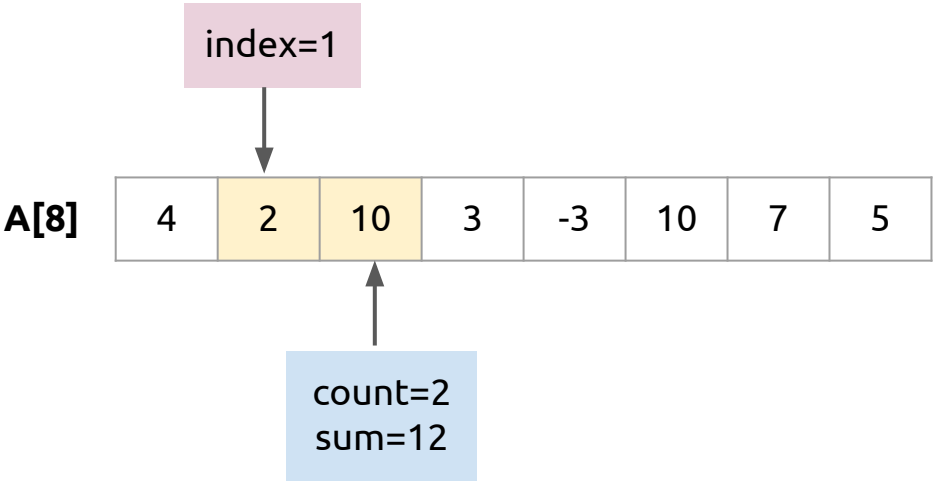
Άσκηση 4.4



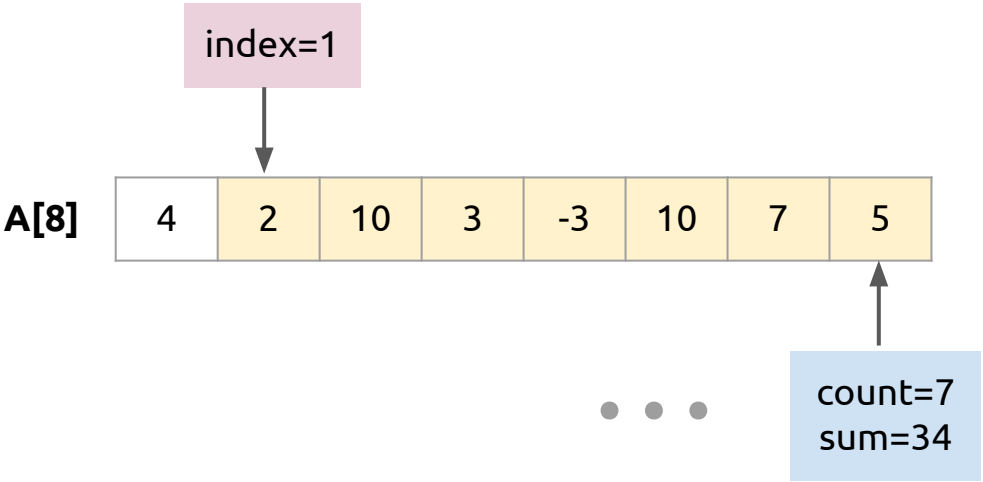
Άσκηση 4.4



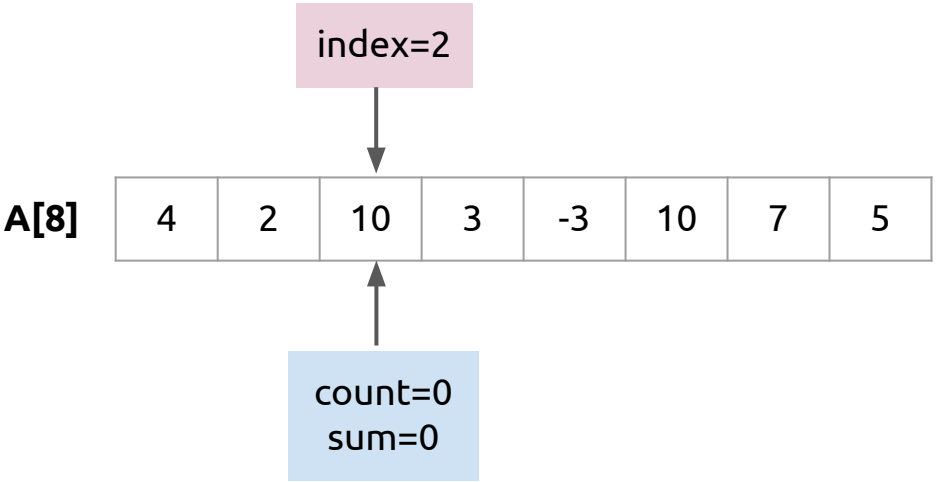
Άσκηση 4.4



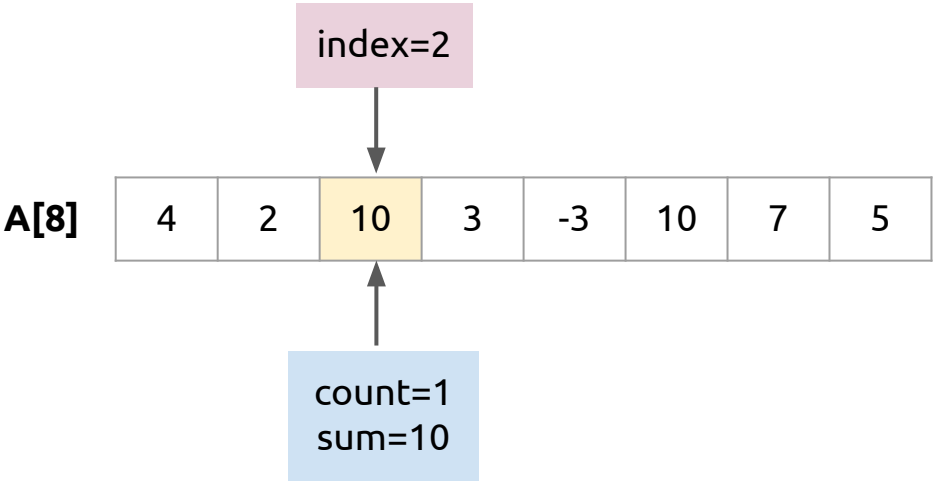
Άσκηση 4.4



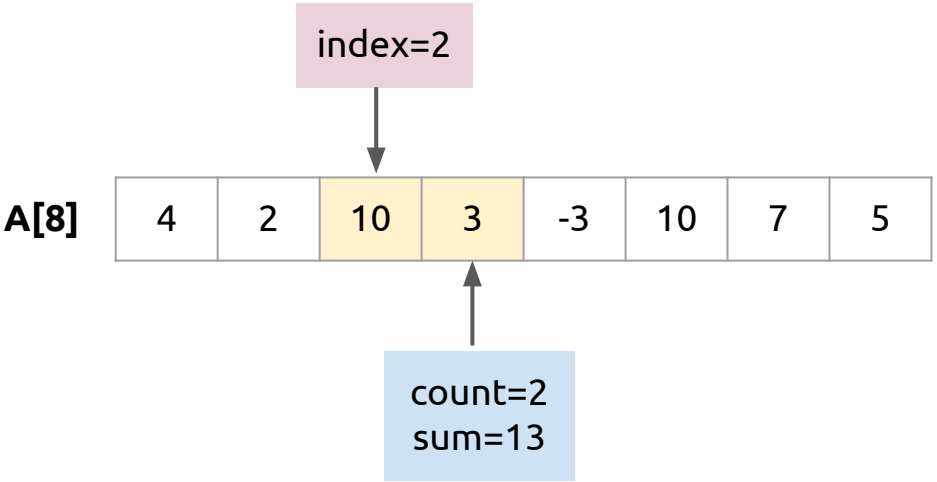
Άσκηση 4.4



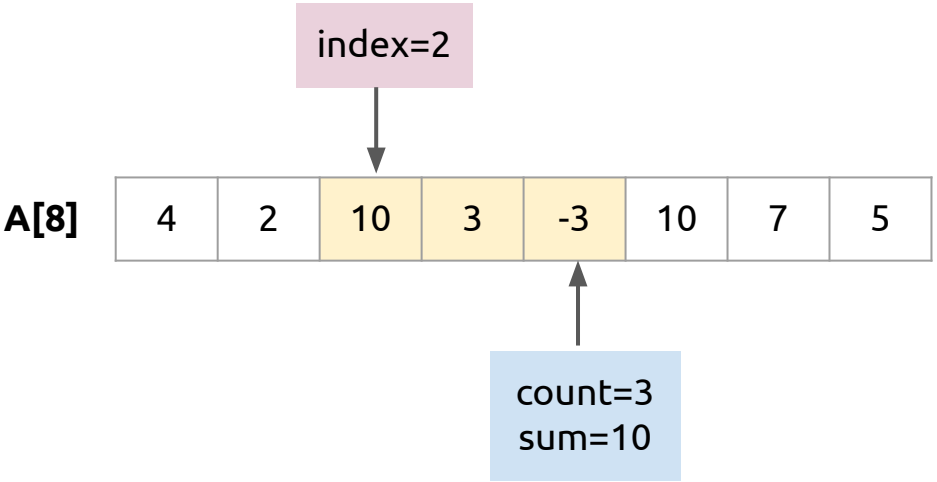
Άσκηση 4.4



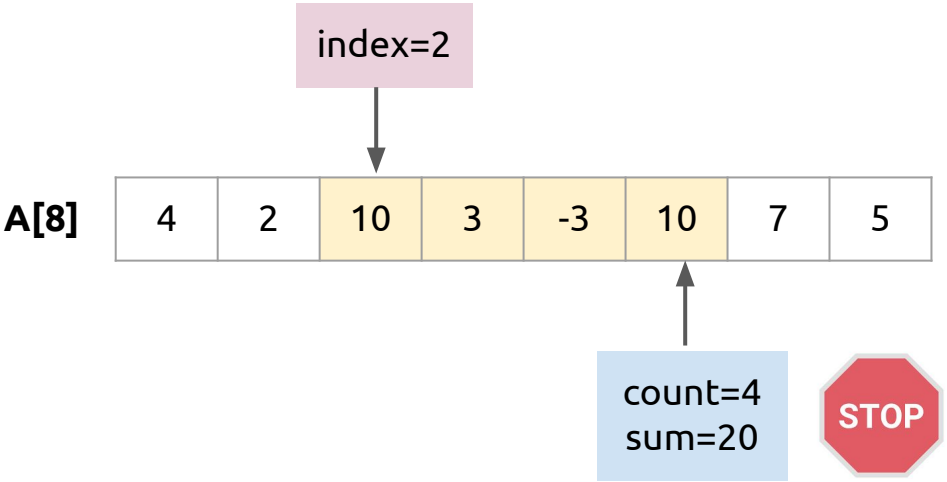
Άσκηση 4.4



Άσκηση 4.4



Άσκηση 4.4



Για να περιγράψω το τμήμα του πίνακα που έχει sum=20, χρειάζομαι μόνο τις μεταβλητές **index** και **count**

Άσκηση 4.4 - subarray


```c
#include <stdio.h>
#include <stdlib.h>
int subarray(int *A, int n, int s, int *count){
    int sum;
    for(int i = 0; i < n; i++){
        sum = 0;
        *count = 0;
        for(int j = i; j < n; j++){
            sum += A[j];
            *count = *count + 1;
            if(sum == s)
               return i;
        }
    }
    return -1;
}
```

Άσκηση 4.4 - main


```c
int main(){
   int n, s, index, count;

   do{
    printf("Δώστε το μέγεθος του πίνακα: ");
    scanf("%d", &n);
   }while(n <= 0);

   int *A = (int*)malloc(n * sizeof(int));

   printf("Δώστε τα στοιχεία του πίνακα:\n");
   for(int i = 0; i < n; i++)
      scanf("%d", A+i);

   printf("Δώστε το s: ");
   scanf("%d", &s);

  index =  subarray(A, n, s, &count);

  int *B = (int*)malloc(count * sizeof(int));
  int j = 0;
  for(int i = index; i < count+index; i++){
      B[j] = A[i];
      j++;
  }

  if (index != -1){
      for(int i = 0; i < count; i++)
         printf("%d ", B[i]);
  }
  else printf("Δεν βρέθηκαν στοιχεία. \n");

}
```

Άσκηση 4.5 (1/3)

Θεωρείστε έναν μονοδιάστατο πίνακα ακεραίων θετικών αριθμών.
Ο πίνακας περιέχει `N`, (`N > 200`) ζεύγη ακεραίων θετικών τιμών.
Η τιμή του N καθορίζεται κατά τη διάρκεια της εκτέλεσης της `main()`.

Κάθε ζεύγος τιμών, αντιστοιχεί:

- στον αύξοντα αριθμό εβδομάδας (επιτρεπτές τιμές `1-52`, όλες οι εβδομάδες αφορούν το ίδιο ημερολογιακό έτος) και
- στις ώρες εκτός λειτουργίας, εντός της συγκεκριμένης εβδομάδας, μιας οποιασδήποτε εργαλειομηχανής σε μια βιομηχανική μονάδα.

Κάθε αύξων αριθμός εβδομάδας, μπορεί να υπάρχει περισσότερες από μια φορές και ο πίνακας δεν είναι ταξινομημένος ως προς τον αύξοντα αριθμό εβδομάδας, δηλ. τα ζεύγη υπάρχουν με τυχαία σειρά.
Ενδέχεται επίσης, να μην υπάρχουν όλοι οι δυνητικοί αύξοντες αριθμοί εβδομάδας στον πίνακα.

Άσκηση 4.5 (2/3)

Να γραφεί μια συνάρτηση με όνομα `weeks_off` που να δέχεται ως όρισμα εισόδου αυτό τον πίνακα. Η συνάρτηση θέλουμε να επιστρέφει στη `main()` τα παρακάτω:

1. Ένα νέο μονοδιάστατο πίνακα που να περιλαμβάνει σε αύξουσα σειρά (χωρίς να χρησιμοποιηθεί διαδικασία ταξινόμησης), όλους τους υπάρχοντες αύξοντες αριθμούς εβδομάδας, καθέναν από μια φορά.
2. Ένα νέο μονοδιάστατο πίνακα που να περιλαμβάνει σε αύξουσα σειρά (χωρίς να χρησιμοποιηθεί διαδικασία ταξινόμησης), ζεύγη θετικών τιμών: (α/α εβδομάδας, συνολικός χρόνος εκτός λειτουργίας για την εβδομάδα, για όλες τις εργαλειομηχανές), για όλους τους υπάρχοντες α/α εβδομάδας του αρχικού πίνακα εισόδου.

Σημείωση: η συνάρτηση ΔΕΝ θα πρέπει να περιέχει εντολές `printf`.

Άσκηση 4.5 (3/3)

Παράδειγμα με N=8:

Αρχικός Πίνακας:

13	2	19	4	33	8	8	11	13	9	8	5	19	4	50	50

Νέος πίνακας μόνο με τους κωδικούς:

8	13	19	33	50

Νέος πίνακας ζευγών:

8	16	13	11	19	8	33	8	50	50

Στη συνέχεια να γραφεί πρόγραμμα που:

- να δίνει τιμές στα ορίσματα εισόδου της συνάρτησης `weeks_off`, εντός των αποδεκτών ορίων τιμών με κατάλληλη χρήση της συνάρτησης δημιουργίας τυχαίων αριθμών `rand()`.
- να καλεί τη συνάρτηση `weeks_off`, και να εμφανίζει τα σχετικά αποτελέσματα.

Άσκηση 4.5 - myRand


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

int myRand(int lower, int upper) {
   int rand_num;
    rand_num = rand() % (upper - lower + 1) + lower;
   return rand_num;
 }
```

Άσκηση 4.5 - weeks_off (1/2)


```
int *weeks_off( int *A, int n, int *size){

   int weeksLookup[53] = {0};

   for(int i = 0; i < n; i+=2){
       int week = A[i];
       weeksLookup[week] = weeksLookup[week] + A[i+1];
   }
   *size = 0;

   for (int i = 0; i < 53; i++){
       if(weeksLookup[i] > 0)
          (*size)++;
   }
```

Άσκηση 4.5 - weeks_off (2/2)


```
    int* New = (int*)malloc(*size*2 * sizeof(int));
    if (New == NULL) {
        printf("Πρόβλημα παραχώρησης μνήμης!\n");
        exit(1);
    }
    int index = 0;
    for(int i = 0; i < 53; i++){
        if(weeksLookup[i] > 0){
            New[index] = i;
            New[index+1] = weeksLookup[i];
            index +=2;
        }
    }
    *size = *size*2 ;
    return New;
}
```

Άσκηση 4.5 - main (1/2)


```c
int main(){

   int n;
   do{
      printf("Δώστε το μέγεθος του πίνακα: ");
      scanf("%d", &n);
   }while(n <= 8 || (n % 2) != 0);

   int *A = (int*)malloc(n * sizeof(int));
   if (A == NULL) {
     printf("Πρόβλημα παραχώρησης μνήμης!\n");
    return 1;
   }
```

```c
   printf("Αρχικός Πίνακας: ");

   for(int i = 0; i < n; i+=2){
      A[i] = myRand(1, 52);
      A[i+1] = myRand(1, 100);
      printf("%d %d ", A[i], A[i+1]);
   }
   printf("\n");
```

Άσκηση 4.5 - main (2/2)


```
    int size = 0;

    int *New = weeks_off(A, n, &size);
    printf("Εβδομάδες: ");
    for(int i = 0; i < size; i+=2)
        printf("%d ", New[i]);

    printf("\n");
    for(int i = 0; i < size; i++)
        printf("%d ", New[i]);
  free(A);
  free(New);
  return 0;
}
```