Διάλεξη 8η: Πίνακες (arrays)

Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Εισαγωγή στην Επιστήμη Υπολογιστών

Βασίζεται σε διαφάνειες του Κ. Παναγιωτάκη

Εισαγωγή

- Έστω ότι θέλουμε να αποθηκεύσουμε 100 ονόματα φοιτητών και τους βαθμούς τους. Πως θα το κάναμε αυτό με μεταβλητές;
- Πως θα μπορούσαμε να πούμε με αυτό το τρόπο “ταξινόμησε τους βαθμούς και τύπωσε τους”;
- Πως θα μπορούσαμε να πούμε “τύπωσε μόνο εκείνους που πέρασαν το μάθημα”;
- ΛΥΣΗ: Πίνακες
  - Συλλογή μεταβλητών ίδιου τύπου
  - Σταθερό μέγεθος
  - N στοιχεία: από 0 έως N-1
  - *Στη C η αρίθμηση ξεκινάει από το 0*

Πίνακες

- Πίνακας
  - Σύνολο από συνεχόμενες θέσεις μνήμης
  - Όλες οι θέσεις περιέχουν δεδομένα του ίδιου τύπου
  - Δήλωση πίνακα

Γενική μορφή

```
type array_name[position_number];
```

- Ένα στοιχείο του πίνακα προσδιορίζεται από
  - Το όνομα του πίνακα
  - Τη θέση του στοιχείου μέσα στον πίνακα
  - Έκφραση που αναφέρεται σε πίνακα

Γενική μορφή

```
array_name[position_number]
```

Πίνακες

- Πίνακας
  - Σύνολο από συνεχόμενες θέσεις μνήμης
  - Όλες οι θέσεις περιέχουν δεδομένα του ίδιου τύπου
  - Δήλωση πίνακα

Ο τύπος δεδομένων του κάθε στοιχείου του πίνακα

Γενική μορφή

```
type array_name[position_number];
```

- Ένα στοιχείο του πίνακα προσδιορίζεται από
  - Το όνομα του πίνακα
  - Τη θέση του στοιχείου μέσα στον πίνακα
  - Έκφραση που αναφέρεται σε πίνακα

Γενική μορφή

```
array_name[position_number]
```

Πίνακες

- Πίνακας
  - Σύνολο από συνεχόμενες θέσεις μνήμης
  - Όλες οι θέσεις περιέχουν δεδομένα του ίδιου τύπου
  - Δήλωση πίνακα

Γενική μορφή

```
type array_name[position_number];
```

Το όνομα του πίνακα

- Ένα στοιχείο του πίνακα προσδιορίζεται από
  - Το όνομα του πίνακα
  - Τη θέση του στοιχείου μέσα στον πίνακα
  - Έκφραση που αναφέρεται σε πίνακα

Γενική μορφή

```
array_name[position_number]
```

Πίνακες

- Πίνακας
  - Σύνολο από συνεχόμενες θέσεις μνήμης
  - Όλες οι θέσεις περιέχουν δεδομένα του ίδιου τύπου
  - Δήλωση πίνακα

Γενική μορφή

```
type array_name[position_number];
```

Ο αριθμός στοιχείων (ή *μήκος*) του πίνακα

- Ένα στοιχείο του πίνακα προσδιορίζεται από
  - Το όνομα του πίνακα
  - Τη θέση του στοιχείου μέσα στον πίνακα
  - Έκφραση που αναφέρεται σε πίνακα

Γενική μορφή

```
array_name[position_number]
```

Πίνακες

- Πίνακας
  - Σύνολο από συνεχόμενες θέσεις μνήμης
  - Όλες οι θέσεις περιέχουν δεδομένα του ίδιου τύπου
  - Δήλωση πίνακα

Γενική μορφή

```
type array_name[position_number];
```

- Ένα στοιχείο του πίνακα προσδιορίζεται από
  - Το όνομα του πίνακα
  - Τη θέση του στοιχείου μέσα στον πίνακα
  - Έκφραση που αναφέρεται σε πίνακα

Γενική μορφή

```
array_name[position_number]
```

Πίνακες

- Πίνακας
  - Σύνολο από συνεχόμενες θέσεις μνήμης
  - Όλες οι θέσεις περιέχουν δεδομένα του ίδιου τύπου
  - Δήλωση πίνακα

Γενική μορφή

```
type array_name[position_number];
```

- Ένα στοιχείο του πίνακα προσδιορίζεται από
  - Το όνομα του πίνακα
  - Τη θέση του στοιχείου μέσα στον πίνακα
  - Έκφραση που αναφέρεται σε πίνακα

Γενική μορφή

```
array_name[position_number]
```

Το όνομα του πίνακα

Πίνακες

- Πίνακας
  - Σύνολο από συνεχόμενες θέσεις μνήμης
  - Όλες οι θέσεις περιέχουν δεδομένα του ίδιου τύπου
  - Δήλωση πίνακα

Γενική μορφή

```
type array_name[position_number];
```

- Ένα στοιχείο του πίνακα προσδιορίζεται από
  - Το όνομα του πίνακα
  - Τη θέση του στοιχείου μέσα στον πίνακα
  - Έκφραση που αναφέρεται σε πίνακα

Γενική μορφή

```
array_name[position_number]
```

Ο *δείκτης θέσης* (index) του στοιχείου

Πίνακες (2)

- Αρίθμηση των στοιχείων
  - Πρώτο στοιχείο στο δείκτη θέσης (index) 0
  - Για πίνακα μεγέθους N, τελευταίο στοιχείο στη θέση N - 1
    - `c[0]`, `c[1]`, ..., `c[n − 1]`
- Δήλωση πολλών πινάκων ίδιου τύπου
  - Όπως και οι μεταβλητές
  - `int b[100], x[27];`
- Τα στοιχεία του πίνακα χρησιμοποιούνται σαν συνηθισμένες μεταβλητές
  - `c[0] = 3;`
  - `if (x[0] == 3) { ... }`
  - `printf("%d", c[0]);`
- Μπορούμε να κάνουμε πράξεις μέσα στο δείκτη
  - `c[5 - 2] == c[rand() % 42]`
  - `c[N - i - 1] = c[i]`
- Εμφωλιασμένοι δείκτες: `c[c[c[1]]]`
- *Προσοχή:*
  - Ο δείκτης πίνακα (index) είναι πάντα ακέραιος
    - `c[i]` μόνο άν `int i;`

Υπολογισμός στοιχείου πίνακα

- Η αναφορά σε πίνακα είναι έκφραση στη C `array_name[expression]`
  - Πρώτα υπολογίζεται η τιμή του `expression`
  - Η τιμή του `expression` πρέπει να είναι ακέραια (int)
  - Η τιμή ολόκληρης της έκφρασης `array_name[expression]` είναι το στοιχείο του πίνακα `array_name` που αντιστοιχεί στην συγκεκριμένη θέση που δίνεται από το `expression`
  - ο τύπος ολόκληρης της έκφρασης είναι ο τύπος των στοιχείων του πίνακα

Παράδειγμα

```
double array[100];
int i, x = 5;

for (i = 0; i < 100; ++i) {
  scanf("%lf", &(array[i]));
}
i = (int) (array[10 * x]);
```

Υπολογισμός στοιχείου πίνακα

- Η αναφορά σε πίνακα είναι έκφραση στη C `array_name[expression]`
  - Πρώτα υπολογίζεται η τιμή του `expression`
  - Η τιμή του `expression` πρέπει να είναι ακέραια (int)
  - Η τιμή ολόκληρης της έκφρασης `array_name[expression]` είναι το στοιχείο του πίνακα `array_name` που αντιστοιχεί στην συγκεκριμένη θέση που δίνεται από το `expression`
  - ο τύπος ολόκληρης της έκφρασης είναι ο τύπος των στοιχείων του πίνακα

Παράδειγμα

```
double array[100];
int i, x = 5;

for (i = 0; i < 100; ++i) {
  scanf("%lf", &(array[i]));
}
i = (int) (array[10 * x]);
```

Το `i` παίρνει τιμές από 0 έως 99

Υπολογισμός στοιχείου πίνακα

- Η αναφορά σε πίνακα είναι έκφραση στη C `array_name[expression]`
  - Πρώτα υπολογίζεται η τιμή του `expression`
  - Η τιμή του `expression` πρέπει να είναι ακέραια (int)
  - Η τιμή ολόκληρης της έκφρασης `array_name[expression]` είναι το στοιχείο του πίνακα `array_name` που αντιστοιχεί στην συγκεκριμένη θέση που δίνεται από το `expression`
  - ο τύπος ολόκληρης της έκφρασης είναι ο τύπος των στοιχείων του πίνακα

Παράδειγμα

```
double array[100];
int i, x = 5;

for (i = 0; i < 100; ++i) {
  scanf("%lf", &(array[i]));
}
i = (int) (array[10 * x]);
```

Το `array[i]` είναι το `i`-οστό στοιχείο του πίνακα

Υπολογισμός στοιχείου πίνακα

- Η αναφορά σε πίνακα είναι έκφραση στη C `array_name[expression]`
  - Πρώτα υπολογίζεται η τιμή του `expression`
  - Η τιμή του `expression` πρέπει να είναι ακέραια (int)
  - Η τιμή ολόκληρης της έκφρασης `array_name[expression]` είναι το στοιχείο του πίνακα `array_name` που αντιστοιχεί στην συγκεκριμένη θέση που δίνεται από το `expression`
  - ο τύπος ολόκληρης της έκφρασης είναι ο τύπος των στοιχείων του πίνακα

Παράδειγμα

```
double array[100];
int i, x = 5;

for (i = 0; i < 100; ++i) {
  scanf("%lf", &(array[i]));
}
i = (int) (array[10 * x]);
```

Το `&(array[i])` είναι η διεύθυνση του `i`-οστού στοιχείου του πίνακα

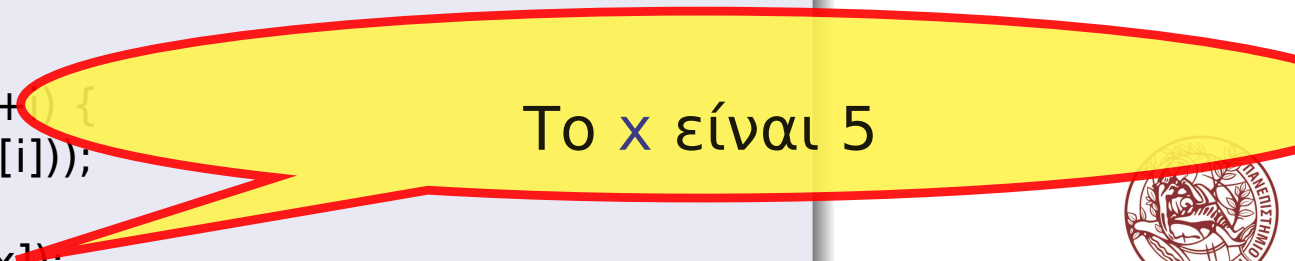
Υπολογισμός στοιχείου πίνακα

- Η αναφορά σε πίνακα είναι έκφραση στη C `array_name[expression]`
  - Πρώτα υπολογίζεται η τιμή του `expression`
  - Η τιμή του `expression` πρέπει να είναι ακέραια (int)
  - Η τιμή ολόκληρης της έκφρασης `array_name[expression]` είναι το στοιχείο του πίνακα `array_name` που αντιστοιχεί στην συγκεκριμένη θέση που δίνεται από το `expression`
  - ο τύπος ολόκληρης της έκφρασης είναι ο τύπος των στοιχείων του πίνακα

Παράδειγμα

```
double array[100];
int i, x = 5;

for (i = 0; i < 100; ++i) {
  scanf("%lf", &(array[i]));
}
i = (int) (array[10 * x]);
```



Υπολογισμός στοιχείου πίνακα

- Η αναφορά σε πίνακα είναι έκφραση στη C `array_name[expression]`
  - Πρώτα υπολογίζεται η τιμή του `expression`
  - Η τιμή του `expression` πρέπει να είναι ακέραια (int)
  - Η τιμή ολόκληρης της έκφρασης `array_name[expression]` είναι το στοιχείο του πίνακα `array_name` που αντιστοιχεί στην συγκεκριμένη θέση που δίνεται από το `expression`
  - ο τύπος ολόκληρης της έκφρασης είναι ο τύπος των στοιχείων του πίνακα

Παράδειγμα

```
double array[100];
int i, x = 5;

for (i = 0; i < 100; ++i) {
  scanf("%lf", &(array[i]));
}
i = (int) (array[10 * x]);
```

Το `10 * x` είναι έκφραση που πολλαπλασιάζει 2 ακέραιους

Υπολογισμός στοιχείου πίνακα

- Η αναφορά σε πίνακα είναι έκφραση στη C `array_name[expression]`
  - Πρώτα υπολογίζεται η τιμή του `expression`
  - Η τιμή του `expression` πρέπει να είναι ακέραια (int)
  - Η τιμή ολόκληρης της έκφρασης `array_name[expression]` είναι το στοιχείο του πίνακα `array_name` που αντιστοιχεί στην συγκεκριμένη θέση που δίνεται από το `expression`
  - ο τύπος ολόκληρης της έκφρασης είναι ο τύπος των στοιχείων του πίνακα

Παράδειγμα

```
double array[100];
int i, x = 5;

for (i = 0; i < 100; ++i) {
  scanf("%lf", &(array[i]));
}
i = (int) (array[10 * x]);
```

Το `10 * x` είναι ακέραια έκφραση που όταν υπολογιστεί θα δώσει τιμή 50

Υπολογισμός στοιχείου πίνακα

- Η αναφορά σε πίνακα είναι έκφραση στη C `array_name[expression]`
  - Πρώτα υπολογίζεται η τιμή του `expression`
  - Η τιμή του `expression` πρέπει να είναι ακέραια (int)
  - Η τιμή ολόκληρης της έκφρασης `array_name[expression]` είναι το στοιχείο του πίνακα `array_name` που αντιστοιχεί στην συγκεκριμένη θέση που δίνεται από το `expression`
  - ο τύπος ολόκληρης της έκφρασης είναι ο τύπος των στοιχείων του πίνακα

Παράδειγμα

```
double array[100];
int i, x = 5;

for (i = 0; i < 100; ++i) {
  scanf("%lf", &(array[i]));
}
i = (int) (array[10 * x]);
```

Το `array[10 * x]` είναι σωστή έκφραση που έχει τύπο `double` και θα δώσει τιμή ίση με τα περιεχόμενα του 50-στού στοιχείου

Υπολογισμός στοιχείου πίνακα

- Η αναφορά σε πίνακα είναι έκφραση στη C `array_name[expression]`
  - Πρώτα υπολογίζεται η τιμή του `expression`
  - Η τιμή του `expression` πρέπει να είναι ακέραια (int)
  - Η τιμή ολόκληρης της έκφρασης `array_name[expression]` είναι το στοιχείο του πίνακα `array_name` που αντιστοιχεί στην συγκεκριμένη θέση που δίνεται από το `expression`
  - ο τύπος ολόκληρης της έκφρασης είναι ο τύπος των στοιχείων του πίνακα

Παράδειγμα

```
double array[100];
int i, x = 5;

for (i = 0; i < 100; ++i) {
  scanf("%lf", &(array[i]));
}
i = (int) (array[10 * x]);
```

Η τιμή αυτή θα μετατραπεί σε ακέραιο και θα αποθηκευτεί στο `i`

Δήλωση πινάκων

- Όταν δηλώνουμε πίνακες ως μεταβλητές καθορίζουμε:
  - Τύπο
  - Όνομα
  - Αριθμό στοιχείων

Παραδείγματα

```
int c[30];
float myArray[31337];
```

- Όταν δηλώνουμε πίνακες ως ορίσματα στον *ορισμό* συναρτήσεων καθορίζουμε:
  - Όνομα
  - Τύπο
  - `int myfunction(int c[]){ ... }`
- Όταν δηλώνουμε πίνακες ως ορίσματα στη *δήλωση* συναρτήσεων καθορίζουμε:
  - Τύπο
  - `int myfunction(int []);`

Δήλωση πινάκων

- Όταν δηλώνουμε πίνακες ως μεταβλητές καθορίζουμε:
  - Τύπο
  - Όνομα
  - Αριθμό στοιχείων

Το c είναι πίνακας 30 ακεραίων

Παραδείγματα

```
int c[30];
float myArray[31337];
```

- Όταν δηλώνουμε πίνακες ως ορίσματα στον *ορισμό* συναρτήσεων καθορίζουμε:
  - Όνομα
  - Τύπο
  - `int myfunction(int c[]){ ... }`
- Όταν δηλώνουμε πίνακες ως ορίσματα στη *δήλωση* συναρτήσεων καθορίζουμε:
  - Τύπο
  - `int myfunction(int []);`

Δήλωση πινάκων

- Όταν δηλώνουμε πίνακες ως μεταβλητές καθορίζουμε:
  - Τύπο
  - Όνομα
  - Αριθμό στοιχείων

Παραδείγματα

```
int c[30];
float myArray[31337];
```

Το `myArray` είναι πίνακας 31337 αριθμών κινητής υποδιαστολής

- Όταν δηλώνουμε πίνακες ως ορίσματα στον *ορισμό* συναρτήσεων καθορίζουμε:
  - Όνομα
  - Τύπο
  - `int myfunction(int c[]){ ... }`
- Όταν δηλώνουμε πίνακες ως ορίσματα στη *δήλωση* συναρτήσεων καθορίζουμε:
  - Τύπο
  - `int myfunction(int []);`

Δήλωση πινάκων

- Όταν δηλώνουμε πίνακες ως μεταβλητές καθορίζουμε:
  - Τύπο
  - Όνομα
  - Αριθμό στοιχείων

Παραδείγματα

```
int c[30];
float myArray[31337];
```

- Όταν δηλώνουμε πίνακες ως ορίσματα στον *ορισμό* συναρτήσεων καθορίζουμε:
  - Όνομα
  - Τύπο
  - `int myfunction(int c[]){ ... }`
- Όταν δηλώνουμε πίνακες ως ορίσματα στη *δήλωση* συναρτήσεων καθορίζουμε:
  - Τύπο
  - `int myfunction(int []);`

Δήλωση πινάκων

- Όταν δηλώνουμε πίνακες ως μεταβλητές καθορίζουμε:
  - Τύπο
  - Όνομα
  - Αριθμό στοιχείων

Παραδείγματα

```
int c[30];
float myArray[31337];
```

- Όταν δηλώνουμε πίνακες ως ορίσματα στον *ορισμό* συναρτήσεων καθορίζουμε:
  - Όνομα
  - Τύπο
  - `int myfunction(int c[]){ ... }`
- Όταν δηλώνουμε πίνακες ως ορίσματα στη *δήλωση* συναρτήσεων καθορίζουμε:
  - Τύπο
  - `int myfunction(int []);`

Το όνομα επιτρέπεται αλλά είναι προεραιτικό

Αρχικοποίηση

- Αρχικοποίηση
  - `int n[5] = { 1, 2, 3, 4, 5 };`
  - Αν δεν υπάρχουν αρκετά στοιχεία για αρχικοποίηση, τότε τα υπολοιπόμενα γίνονται 0
    - Όλα τα στοιχεία 0: `int n[5] = { 0 };`
    - `for (i = 0; i < 5; ++i) { n[i] = 0; }`
  - Αν βάλουμε πιο πολλά στοιχεία από όσα έχει ο πίνακας τότε θα πάρουμε λάθος
- Αν αρχικοποιούμε τον πίνακα μπορούμε να μη δώσουμε μέγεθος και ο πίνακας θα είναι όσος και τα στοιχεία που δίνουμε
  - `int n[] = { 1, 2, 3, 4, 5 };`
  - 5 στοιχεία, οπότε πίνακας 5 στοιχείων

Παράδειγμα: ιστόγραμμα

histogram.c

```c
#include <stdio.h>

/* The size of the array
 * storing the grades */
#define HIST_SIZE 11

/* The size of the array
 * storing the histogram */
#define STUDENTS 30

/*
 * Computes the histogram n of array
 * a[]. Array a[] has sizeA elements.
 */
void histogram(int n[], int a[], int sizeA)
{
  int i;

  for (i = 0; i < sizeA; ++i) {
    n[a[i]]++;
  }
}

int main()
{
  int hist[HIST_SIZE] = {0};
  int grades[STUDENTS], i, j;

  for (i = 0; i < STUDENTS; ++i) {
    scanf("%d", &(grades[i]));
  }

  histogram(hist, grades, STUDENTS);

  for (i=0; i < HIST_SIZE; i++) {
    printf("%7d%13d%8c", i, hist[i], ' ');
    for (j = 0; j < hist[i]; j++) {
      putchar('*');
    }
    printf("\n");
  }
  return 0;
}
```

Όρια

- Στη C δεν υπάρχει έλεγχος ορίων!
  - Αν `int c[10];` τότε ο πίνακας `c` έχει 10 στοιχεία
  - Τα όρια του πίνακα `c` είναι ανάμεσα στο `c[0]` και `c[9]`
- Τι γίνεται αν βγούμε έξω από τον πίνακα;
  - `c[-1];`
  - `c[10];`
  - `c[10000000];`

Παράδειγμα: crash!

outofbounds.c

```
#include <stdio.h>

int main()
{
  int a = 5, b = 6, c;
  int n[10];

  printf("a = %d, b = %d\n", a, b);

  /* Here we get out of bounds. What's going to
     happen depends on the system. */
  for (c = -3; c < 10; c++) {
    n[c] = 10;
  }

  /* Here we get out of bounds. What's going to
     happen depends on the system. */
  n[10] = 0;

  printf("a = %d, b = %d\n", a, b);
  return 0;
}
```

sizeof

- Ο τελεστής `sizeof` μπορεί να χρησιμοποιηθεί με
  - Ονόματα μεταβλητών
  - Ονόματα τύπων
  - Σταθερές
- Χρήση του `sizeof`
  - Επιστρέφει το μέγεθος σε bytes
  - Έχει τύπο `size_t`: μέγεθος σε bytes ως ακέραιος
  - Για πίνακες, επιστρέφει το μέγεθος του 1 στοιχείου επί το πλήθος
  - Αν `sizeof(int)` είναι 4 bytes, τότε το παρακάτω πρόγραμμα θα τυπώσει “40”

sizeof.c

```
int my_array[10];
printf("\%d", sizeof(my_array));
```

Υπολογισμός Διεύθυνσης Στοιχείων

- Στη γλώσσα C, το όνομα ενός πίνακα είναι δείκτης στην αρχή του πίνακα
  - Το όνομα του πίνακα αντιστοιχεί σε μια σταθερή διεύθυνση μνήμης
- Έστω ο πίνακας `double a[100];`, που αρχίζει στην διεύθυνση `a`
- Κάθε στοιχείο του πίνακα πιάνει χώρο `sizeof(double)`
- Πού βρίσκεται στη μνήμη το στοιχείο `a[index]`;
  - Το πρώτο στοιχείο (`index == 0`) έχει διεύθυνση `&a[0]`, που αντιστοιχεί στη θέση μνήμης `a`
  - Το δεύτερο στοιχείο (`index == 1`) έχει διεύθυνση `&a[1]`, που αντιστοιχεί στη θέση μνήμης `a + sizeof(double)`
  - Το τρίτο στοιχείο (`index == 2`) έχει διεύθυνση `&a[2]`, που αντιστοιχεί στη θέση μνήμης `a + 2 * sizeof(double)`
  - κλπ.
- Η διεύθυνση (σε Bytes) στη μνήμη είναι η:
  `a + sizeof(double) * index`

Ο Πίνακας ως Δείκτης

- Έστω ο πίνακας `int b[100];`
  - Η έκφραση `b[0]` είναι το 1ο στοιχείο του πίνακα
  - Η έκφραση `&b[0]` είναι η διεύθυνση του 1ου στοιχείου του πίνακα
  - Η έκφραση `&b[0]` είναι ισοδύναμη με την έκφραση `b` (το όνομα του πίνακα)
- Αντί να γράφουμε `&b[0]` απλά γράφουμε `b`
- Το `b` είναι διεύθυνση μνήμης: δείκτης στο πρώτο στοιχείο
  - Μπορούμε να χρησιμοποιούμε το `b` ακριβώς όπως ένα δείκτη
  - Αργότερα θα δούμε ότι μπορούμε να χρησιμοποιούμε και δείκτες ως πίνακες!
- Ο τύπος του `b` είναι σταθερός δείκτης σε ακέραιο

Παράδειγμα: Addresses

array-memory.c

```
#include <stdio.h>

/* Maximum array size */
#define SIZE 10

int main()
{
  int array[SIZE] =
      {5, 42, 9, 123, 0, 1821, 31337, 1729, -1, -232};
  int n;

  for (n = 0; n < SIZE ; n++) {
    printf("array[%2d] = %5d,"
           "at memory address %ld\n",
           n, array[n], (long) &array[n]);
  }
  return 0;
}
```

Πίνακες ως ορίσματα συναρτήσεων

- Δίνουμε το όνομα του πίνακα ως όρισμα
  - Συνήθως περνάμε το μέγεθος του πίνακα ως επιπλέον όρισμα

array-argument.c

```
float mean(int array[], int length)
{
  int i;
  float sum = 0;

  for (i = 0; i < length; i++) {
    sum += array[i];
  }
  return sum / length;
}
```

```
int main(void)
{
  int my_array[24], i;

  i = mean(myArray, 24);
}
```

Πίνακες ως ορίσματα συναρτήσεων (2)

- Συνάρτηση με όρισμα πίνακα αντιστοιχεί με call-by-reference
  - Το όνομα του πίνακα είναι και δείκτης στο 1ο στοιχείο
  - Η συνάρτηση έχει πρόσβαση στη μνήμη του πίνακα
  - Η συνάρτηση μπορεί να αλλάξει τα περιεχόμενα του πίνακα

array-argument.c

```
void flip(int array[], int length)
{
  for(i = 0; i < length / 2; i++) {
    swap(&array[i], &array[length -i -1]);
  }
}
```

- Τα στοιχεία του πίνακα είναι τιμές
  - Αν είναι ορίσματα συνάρτησης, περνούν κανονικά με call-by-value
  - `printf("%d", myArray[3]);`

Παράδειγμα

side-effects.c

```c
#include <stdio.h>

/* function declaration */
void f(int a[], int size, int b);

/* function definitions */
int main()
{
  int c[10] = {0};
  int n;
  int q = 30;

  /* Print the values of the array
     before the function call */
  for (n = 0; n < 10; n++) {
    printf("%2d, %3d\n", n, c[n]);
  }
  printf("\nq = %d\n\n", q);

  /* call function f */
  f(c, 10, q);

  /* Print the values of the array
     after the function call. */
  for (n = 0; n < 10; n++) {
    printf("%2d, %3d\n", n, c[n]);
  }
  printf("\nq = %d\n", q);

  return 0;
}

/* Definition of function f
 * What does f do?
 */
void f(int a[], int size, int b)
{
  while(size) {
    a[size-1] = size--;
  }
  b = 100;
}
```