C: arrays

Vasilios Papaliakos, < >

Hit the space bar or swipe left for next slide

Σκοποί μαθήματος

Στο τέλος του μαθήματος, θα μπορείτε:

- να εξηγείτε τι είναι οι πίνακες και γιατί είναι χρήσιμοι
- να δημιουργείτε πίνακες και να τους χρησιμοποιείτε
- να περιγράφετε τη συσχέτιση θέσεων πίνακα και διευθύνσεων μνήμης
- να αρχικοποιείτε ένα πίνακα χωρίς χρήση βρόγου
- να αναφέρετε το πρόβλημα υπέρβασης και τις επιπτώσεις του

Πίνακες (arrays) - Τι είναι

- μια συλλογή από μεταβλητές του **ιδίου τύπου**
  - *αναφερόμαστε σε όλη τη συλλογή **με ένα όνομα***
  - *όλες οι τιμές ενός πίνακα βρίσκονται σε **διαδοχικές θέσεις μνήμης***
  - *ο **αριθμητικός δείκτης (index)** δείχνει τη σειρά κάθε στοιχείου μέσα στον πίνακα*
- οι πίνακες είναι πολύ σημαντικοί στον προγραμματισμό:
  - *πολλοί αλγόριθμοι στηρίζονται σε αυτούς*
  - *χρησιμεύουν στην ταξινόμηση, αναζήτηση και διαχείριση σχετιζόμενων δεδομένων*
- στη C υποστηρίζονται πίνακες πολλαπλών διαστάσεων
  - *εδώ θα δούμε κυρίως τους μονοδιάστατους πίνακες και αυτούς των δύο διαστάσεων*

Ενα απλό παράδειγμα

- Η γενική μορφή δήλωσης ενός μονοδιάστατου πίνακα:

```
τύπος όνομα_πίνακα[πλήθος_στοιχείων];
```

  - *τύπος: ίδιος για όλα τα στοιχεία του πίνακα*
  - *όνομα_πίνακα: όπως το όνομα μιας μεταβλητής*
  - *πλήθος_στοιχείων: ακέραια σταθερά (από 0 έως n-1)*

- παράδειγμα πίνακα 10 ακεραίων:

```
int arr[10];
```

  - *πρώτο στοιχείο είναι το arr[0] και τελευταίο το στοιχείο arr[9]*
  - *έστω οι εντολές (αναθέσεις):*

```
arr[2] = 5;
arr[5] = 12;
arr[8] = 6;
```

  - *θα γέμιζαν τον πίνακα arr ως εξής:*

		5			12			6	

Διευθύνσεις μνήμης

- Ας δούμε αναλυτικότερα το προηγούμενο παράδειγμα:

arr(0)	arr(1)	arr(2)	arr(3)	arr(4)	arr(5)	arr(6)	arr(7)	arr(8)	arr(9)
		5			12			6	
100	104	108	112	116	120	124	128	132	136

  - *στην πάνω γραμμή φαίνονται οι θέσεις του πίνακα*
  - *στη μεσαία γραμμή φαίνονται οι τιμές του πίνακα*
  - *στην κάτω γραμμή φαίνονται οι διευθύνσεις μνήμης (θεωρούμε ότι οι ακέραιες τιμές καταλαμβάνουν 4 bytes και ότι η διεύθυνση του πρώτου στοιχείου είναι 100)*

```c
/* print array addresses */

for (int i=0; i < 4; i++) {
    printf("the address of arr[%d] is %p\n", i, &arr[i]);
}

Output

the address of arr[0] is 0x7ffc98bbf690
the address of arr[1] is 0x7ffc98bbf694
the address of arr[2] is 0x7ffc98bbf698
the address of arr[3] is 0x7ffc98bbf69c
```

- Μπορούμε να δούμε εύκολα ότι οι διευθύνσεις μνήμης είναι συνεχόμενες

Αρχικοποίηση πινάκων

- δήλωση μεγέθους και ανάθεση τιμών ταυτόχρονα

```
char c[5] = {'a','e','i','o','u'};
```

  - *η λίστα με τις τιμές είναι μέσα σε άγκιστρα {}*
  - *οι τιμές χωρίζονται με κόμμα*

```
char c[] = {'a','e','i','o','u'};
```

  - *το μέγεθος του πίνακα μπορεί να παραληφθεί*
  - *ο compiler το υπολογίζει από τα στοιχεία*
  - *οι ισοδύναμες εντολές θα ήταν*

```
char c[5];

c[0] = 'a';
c[1] = 'e';
c[2] = 'i';
c[3] = 'o';
c[4] = 'u';
```

- αρχικοποίηση με μια τιμή (πχ μηδέν), χωρίς βρόγχο

```
int arr[500] = {0};
```

Υπέρβαση περιοχής μνήμης

- στο παράδειγμα μας ορίσαμε ένα πίνακα 10 θέσεων
  και αναθέσαμε τιμές σε 3 μόνο θέσεις
- οι υπόλοιπες θέσεις του πίνακα περιέχουν "σκουπίδια"!
  - *σε κάθε θέση του πίνακα αντιστοιχεί κάποια διεύθυνση μνήμης*
  - *εφόσον δεν έχουμε αναθέσει τιμές, η μνήμη σε αυτή τη διεύθυνση μπορεί να περιέχει ο,τιδήποτε*
  - *γιαυτό είναι καλό να αρχικοποιούμε τους πίνακες μας*

arr(0)	arr(1)	arr(2)	arr(3)	arr(4)	arr(5)	arr(6)	arr(7)	arr(8)	arr(9)
		5			12			6	
100	104	108	112	116	120	124	128	132	136

- Προσοχή! Η C μας επιτρέπει να ξεπεράσουμε τις θέσεις που ορίσαμε στον πίνακα

```
for (int i=0; i < 14; i++) {
    //printf("the address of arr[%d] is %p\t", i, &arr[i]);
    printf(" - the value of arr[%d] is %d\n", i, arr[i]);
}
```

```
 - the value of arr[0] is 0
 - the value of arr[1] is 0
 - the value of arr[2] is 5
 - the value of arr[3] is 0
 - the value of arr[4] is 0
 - the value of arr[5] is 12
 - the value of arr[6] is 0
 - the value of arr[7] is 0
 - the value of arr[8] is 6
 - the value of arr[9] is 22054
 - the value of arr[10] is 15775231
 - the value of arr[11] is 0
 - the value of arr[12] is 5
 - the value of arr[13] is 0
```

- *βλέπουμε ότι ο υπολογιστής ξεπέρασε τις 10 θέσεις που είχαμε ορίσει (οι θέσεις κάθε πίνακα υπολογίζονται ως θέσεις μνήμης)*
- *βλέπουμε επίσης ότι οι θέσεις που δεν αναθέσαμε τιμές μπορεί να περιέχουν ό,τιδήποτε*

Μονοδιάστατοι πίνακες

- οι πίνακες χρησιμεύουν (μεταξύ άλλων) στη διαχείριση τιμών (ιδίου τύπου) που χρειάζεται να κρατήσουμε στη μνήμη
- παράδειγμα αντιστροφής

```c
#include <stdio.h>

int main() {
    int arr[5];

    for (int i=0; i < 5; i++) {
        printf("arr[%d] = ", i);
        scanf("%d", &arr[i]);
    }

    printf("\nIn reverse order: ");
    for (int i=4; i >= 0; i--)
        printf("%d\t", arr[i]);

    return 0;
}
```

```
arr[0] = 45
arr[1] = 5
arr[2] = 87
arr[3] = 62
arr[4] = 3

In reverse order: 3     62      87      5       45
```

Σημειώσεις:

- *για την προσπέλαση των στοιχείων ενός πίνακα χρησιμοποιούμε συνήθως τον βρόγχο for*
- *ο μετρητής του for (συνήθως το i:index) είναι ο δείκτης του πίνακα*

- παραδείγματα στατιστικής
  - *υπολογισμός μέσης ή ενδιάμεσης τιμής, μεγίστου ή ελαχίστου*
  - *εκτύπωση ραβδογραμμάτων*
  - *κλπ*

Πίνακες 2 διαστάσεων

- αποτελούνται από γραμμές και στήλες*
- ένας πίνακας 2 διαστάσεων είναι ισότιμος με ένα μονοδιάστατο που κάθε στοιχείο του είναι κι αυτό ένας πίνακας!
- Η γενική μορφή δήλωσης ενός πίνακα 2 διαστάσεων:

```
τύπος όνομα_πίνακα[πλήθος_γραμμών][πλήθος_στηλών];
```

- Παράδειγμα πίνακα 3x4:

```
int arr[3][4];
```

	j = 0	j = 1	j = 2	j = 3
i = 0				
i = 1				arr[1][3]
i = 2				

- Προσπέλαση στοιχείων πίνακα:
  - *με χρήση ένθετων βρόγχων for:*

```
int i, j;
int arr[3][4];

for (i=0; i<3; i++){
    for (j=0; j<4; j++){
        arr[i][j] = i*j;
    }
}
```

	j = 0	*j = 1*	*j = 2*	*j = 3*
i = 0	*0*	*0*	*0*	*0*
i = 1	*0*	*1*	*2*	*3*
i = 2	*0*	*2*	*4*	*6*

  - *πόσες φορές θα εκτελεστεί η εντολή `arr[i][j] = i*j;`;*
    - a. 3
    - b. 4
    - c. 7
    - d. 12