Εισαγωγή στον Προγραμματισμό

Πανεπιστήμιο Θεσσαλίας

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ

Πίνακες (Arrays)

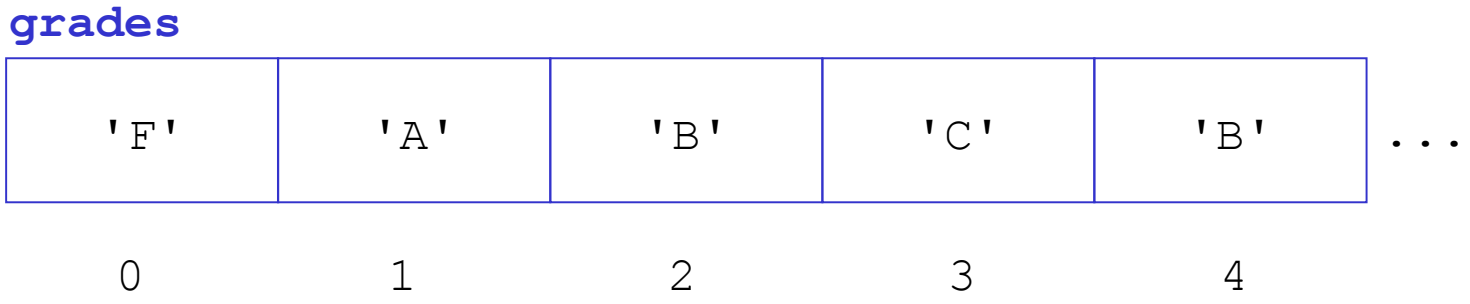
- Πρόβλημα:
  - Πολλές φορές χρειαζόμαστε μεγάλο αριθμό μεταβλητών για να αποθηκεύσουμε όμοιες ποσότητες.
    - πχ. από μια μεταβλητή για το βαθμό κάθε ενός φοιτητή
    - Ακόμη κι αν δηλώναμε τόσες μεταβλητές όσοι είναι οι φοιτητές, κάθε φορά που θα θέλαμε να χρησιμοποιήσουμε το πρόγραμμα για διαφορετικό πλήθος φοιτητών, θα έπρεπε να κάνουμε πολλές αλλαγές.
- Λύση:
  - Πίνακες: μας επιτρέπουν να γκρουπάρουμε κάτω από ένα κοινό όνομα ένα μεγάλο αριθμό μεταβλητών (θέσεων μνήμης).

Πίνακες

- Ενας πίνακας είναι μια ειδική δομή που μας επιτρέπει να αποθηκεύσουμε
  - σε διαδοχικές θέσεις μνήμης,
  - κάτω από κοινό όνομα,
  - ένα συγκεκριμένο αριθμό δεδομένων
  - ιδίου τύπου.

Πίνακες

- Φανταστείτε ένα πίνακα ως μια σειρά από "κουτάκια" στη μνήμη.
  - Σε κάθε κουτί αποθηκεύεται μία τιμή
  - Ολα τα κουτιά περιέχουν τιμές ίδιου τύπου
  - Ολα τα κουτιά έχουν το ίδιο όνομα
  - Επιλέγουμε συγκεκριμένο κουτί χρησιμοποιώντας τον αύξοντα αριθμό του.
  - Το μέτρημα ξεκινά από το μηδέν.



Πίνακες

- Για να μπορούμε να χρησιμοποιήσουμε ένα πίνακα πρέπει πρώτα να τον δηλώσουμε.
- Στη δήλωση προσδιορίζουμε τον **τύπο** των στοιχείων που αποθηκεύονται σε αυτόν, το **όνομα** του πίνακα, και το **πλήθος** των στοιχείων.

```
char grades[5];
```

- Μετά τη δήλωση έχει δημιουργηθεί το:

τα περιεχόμενα των κουτιών είναι "σκουπίδια" - δεν έχει γίνει αρχικοποίηση

Μέγεθος πινάκων

- Το μέγεθος ενός πίνακα δεν αλλάζει.
- Τίθεται κατά τη δήλωση
- Πρέπει να είναι συγκεκριμένη ακέραια τιμή *
- Συχνά το ορίζουμε με #define

```
#define SIZE 5

int main () {
    char grades[SIZE];
    ...
}
```

όλες οι εμφανίσεις του SIZE μέσα στο πρόγραμμα αντικαθίστανται από το 5.

* Λεπτομέρειες στο μάθημα

Χρήση πινάκων

- Για να προσπελάσουμε ένα στοιχείο πίνακα χρησιμοποιούμε το όνομα του πίνακα και τον αύξοντα αριθμό του σε αγκύλες.
- Κατά τα άλλα, το στοιχείο χρησιμοποιείται ακριβώς σαν μια μεταβλητή:

```
#define SIZE 5

int main () {
    char grades[SIZE];
    grades[0] = 'A';
    printf("%d\n", grades[0]);
    ...
}
```

Σε αυτό το παράδειγμα έχει ανατεθεί βαθμός μόνο στη θέση 0 του πίνακα. Στις άλλες θέσεις υπάρχουν "σκουπίδια".

Χρήση πινάκων

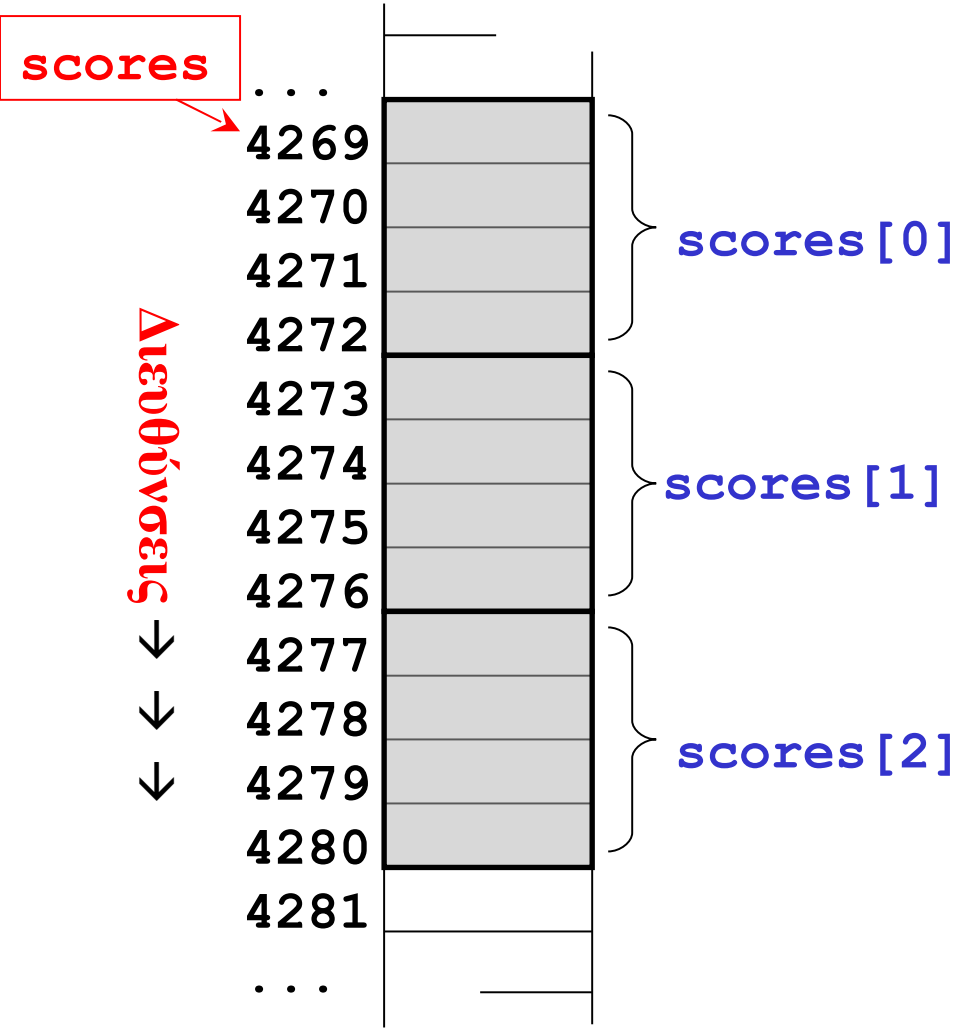
- Εφόσον το "μέτρημα" ξεκινά από 0, η τελευταία θέση ενός πίνακα μεγέθους SIZE έχει αύξοντα αριθμό SIZE - 1.
- ΠΡΟΣΟΧΗ! Ο compiler ΔΕΝ ανιχνεύει προσπέλαση εκτός των ορίων του πίνακα.
  - Για παράδειγμα, δε θα πει τίποτα αν γράψουμε grades[-3] ή grades[4385]
  - Προφανώς κάτι τέτοιο οδηγεί (αν είμαστε τυχεροί) σε προσπέλαση θέσης μνήμης που δεν έχουμε πρόσβαση ή (αν είμαστε άτυχοι) σε λάθος αποτελέσματα.

Χρήση πινάκων

- Το όνομα ενός πίνακα από μόνο του αναπαριστά τη διεύθυνση στη μνήμη όπου ξεκινά η αποθήκευση των στοιχείων του πίνακα.
- Ο αύξων αριθμός είναι ουσιαστικά μετατόπιση (offset) από την αρχή του πίνακα (με την έννοια του πόσα στοιχεία προηγούνται)

Πίνακες

```
int scores[3];
```



Το scores[i] είναι αποθηκευμένο στη διεύθυνση:

scores + i * sizeof(int)

αρχή του πίνακα

μετατόπιση

μέγεθος στοιχείου πίνακα (σε bytes)

Πίνακες

```
#include<stdio.h>

#define SIZE 5

int main () {
    int scores[SIZE], i;
    printf("scores: %p\n", scores);
    for (i=0; i < SIZE; i++) {
        printf("%d: %p\n", i, &scores[i]);
    }
    return 0;
}
```

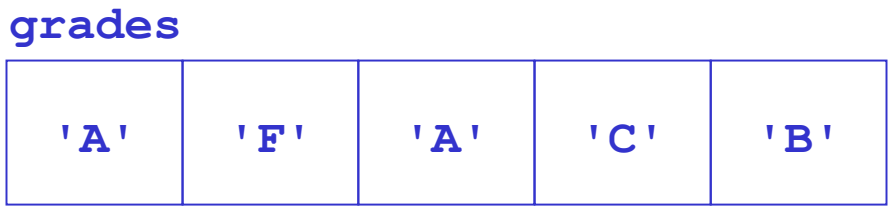
```
scores: 0x7fff5170ab70
0: 0x7fff5170ab70
1: 0x7fff5170ab74
2: 0x7fff5170ab78
3: 0x7fff5170ab7c
4: 0x7fff5170ab80
```

Αρχικοποίηση πινάκων

- Καλό είναι να αρχικοποιούμε ένα πίνακα πριν τον χρησιμοποιήσουμε. Υπάρχουν διάφοροι τρόποι:

  1. Κατά τη δήλωση:

```
char grades[5] = {'A', 'F', 'A', 'C', 'B'};
```



ΠΡΟΣΟΧΗ: Αυτού του είδους η αρχικοποίηση μπορεί να γίνει μόνο κατά τη δήλωση.

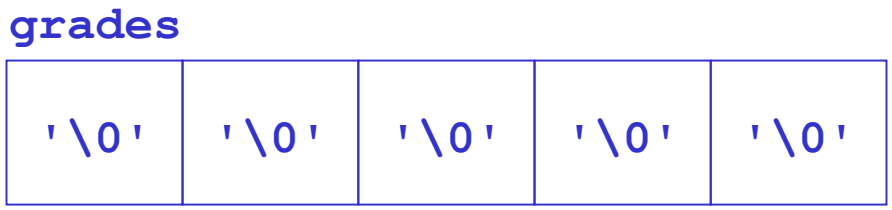
```
char grades[5];
grades = {'A', 'F', 'A', 'C', 'B'};
```

Αρχικοποίηση πινάκων

- Καλό είναι να αρχικοποιούμε ένα πίνακα πριν τον χρησιμοποιήσουμε. Υπάρχουν διάφοροι τρόποι:

2. Κατά τη δήλωση, σε μηδέν:

```
char grades[5] = {'\0'};
```



ΠΡΟΣΟΧΗ: Αυτού του είδους η αρχικοποίηση λειτουργεί "σωστά" μόνο για το μηδέν.

```
char grades[5] = {'A'};
```
δεν τα αρχικοποιεί όλα σε 'A', αλλά μόνο το πρώτο!

Αρχικοποίηση πινάκων

- Καλό είναι να αρχικοποιούμε ένα πίνακα πριν τον χρησιμοποιήσουμε. Υπάρχουν διάφοροι τρόποι:

  3. Με επανάληψη:

```
int numbers[SIZE];
for (i = 0; i < SIZE; i++) {
    scanf("%d", &numbers[i]);
}
```

ή

```
for (i = 0; i < SIZE; i++) {
    numbers[i] = 0;
}
```

Αρχικοποίηση πινάκων

- Απαγορεύεται να γίνει ανάθεση τιμής σε ολόκληρο τον πίνακα χρησιμοποιώντας μόνο το όνομά του:
  - int scores[5];

    scores = 18; // **ΛΑΘΟΣ!**

  - int nums[5], vals[5];

    nums = vals; // **ΛΑΘΟΣ!**

Παραδείγματα

- Γράψτε ένα πρόγραμμα που διαβάζει τους βαθμούς 20 φοιτητών (ακέραιοι αριθμοί), τους αποθηκεύει σε ένα πίνακα και υπολογίζει το μέσο όρο τους

Παραδείγματα

- Γράψτε ένα πρόγραμμα που διαβάζει μια σειρά από χαρακτήρες μέχρι να διαβάσει παύλα (οπότε και σταματά την ανάγνωση), τους αποθηκεύει σε ένα πίνακα, και μετά τους αντιστρέφει. Ο πίνακας θα έχει μέγεθος 20, αλλά δεν είναι υποχρεωτικό να διαβαστούν 20 χαρακτήρες.
  - Χωρίς βοηθητικό πίνακα!

Παραδείγματα

- Γράψτε ένα πρόγραμμα που διαβάζει 20 πραγματικούς αριθμούς και τους αποθηκεύει σε ένα πίνακα. Μετά, διαβάζει έναν ακόμη αριθμό και αφαιρεί από τον πίνακα όσους είναι μικρότεροι από αυτόν.
  - Χωρίς βοηθητικό πίνακα!