Κεφάλαιο 8.6

Πίνακες II
(Διάλεξη 16)

Διδάσκων: Δημήτρης Ζεϊναλιπούρ

Πίνακες - Επανάληψη

- Στην προηγούμενη διάλεξη κάναμε μια εισαγωγή στην δομή δεδομένων Πίνακας
- Σε ένα πίνακα ένα σύνολο αντικειμένων του **ιδίου τύπου** αποθηκεύονται **σε σειρά**, π.χ.

```
int md[12]=  {31,28,31,30,31,30,31};
```

τύπος

όνομα

Μέγεθος (πλήθος στοιχείων

md[0]	31
md[1]	28
md[2]	31
…	
…	
md[11]	31

τιμές

Εφαρμογές Πινάκων (Arrays)

Σήμερα θα δούμε τις εξής εφαρμογές Πινάκων:

1. Παράλληλοι Πίνακες
2. Γραμμική Αναζήτηση (Linear Search)
3. Γραμμική Αναζήτηση & Ενημέρωση
4. Μέτρηση Στοιχείων που ικανοποιούν κάποια συνθήκη.
5. Αλγόριθμος Ταξινόμησης Πίνακα Selection-Sort

1) Παράλληλοι Πίνακες

Περισσότεροι από ένα μονοδιάστατους πίνακες, όπου κάθε δείκτης i αναφέρετε στα στοιχεία ενός κοινού αντικειμένου.

2 παράλληλοι πίνακες

π.χ

```
#define STUDENT_NUM      55
int student_id[STUDENT_NUM];
float student_grade[STUDENT_NUM];
```

- int student_id[i] περιέχει αρ. ταυτότητας του i
- float student_grade[i] περιέχει τον βαθμό του φοιτητή i.

1) Παράλληλοι Πίνακες

Δηλαδή η αποθήκευση των πληροφοριών σε αυτούς τους πίνακες μοιάζει ως εξής:

student_id

0	12345
1	37349
2	9995
...	
54	20001

student_grade

0	2.12
1	6.14
2	4.56
...	
54	7.8

1) Παράλληλοι Πίνακες

```
#include <stdio.h>
#define STUDENT_NUM  3

int main() {
    int i;
    int student_id[]={12345, 37349, 9995};
    float student_grade[]={90.3,78.2,45.3};

    for(i=0;i<STUDENT_NUM;i++)
        printf("Student with id: %d, grade: %f\n", student_id[i],
    student_grade[i]);
}
```

Εκτυπώνει

Student with id: 12345, grade: 90.300003
Student with id: 37349, grade: 78.199997
Student with id: 9995, grade: 45.299999

1) Παράλληλοι Πίνακες

Θα μπορούσα να είχα περισσότερους από 2 πίνακες. π.χ.

```
#define STUDENT_NUM 55
int student_id[STUDENT_NUM];
float student_grade[STUDENT_NUM];
int student_age[STUDENT_NUM];
int student_year[STUDENT_NUM];
char student_sex[STUDENT_NUM];
…..
```

2) Γραμμική αναζήτηση (Linear Search)

- **Γράψετε τμήμα προγράμματος που αναζητά μέσα στον πίνακα ακέραιων στοιχείων *student_id* την θέση που περιέχει την τιμή *z*.**
- Δηλαδή ψάχνουμε να βρούμε αν υπάρχει μια συγκεκριμένη ταυτότητα στον πίνακα ταυτοτήτων.
- Το μέγεθος του πίνακα ορίζεται με την σταθερά ***STUDENT_NUM***.
- Αν δεν βρεθεί το στοιχείο το πρόγραμμα να εκτυπώνει μήνυμα λάθους, αν βρεθεί επιστρέφει την θέση i στην οποία βρέθηκε.

2) Γραμμική αναζήτηση (Linear Search)

- Τι πρέπει να γίνει;
  - Αναζήτηση
- Ψευδοκώδικας

  για κάθε στοιχείο του πίνακα

    εάν είναι ίσο με z

      φύλαξε θέση

      τερμάτισε επανάληψη
- Κόστος
  - στην χειρότερη περίπτωση εξέταση όλων των στοιχείων του πίνακα
  - Στην καλύτερη περίπτωση βρίσκουμε το στοιχείο στην πρώτη θέση.

	student_id
0	12345
1	37349
2	9995
...	
54	20001

2) Γραμμική αναζήτηση (Linear Search) - Υλοποίηση

```
#include <stdio.h>

#define STUDENT_NUM 3    // Εδώ χρησιμοποιούμε μόνο 3 φοιτητές αντί 54

int main() {
    int i;
    int student_id[]={12345, 37349, 9995};
    int z = 98995;

    for(i=0; i<STUDENT_NUM; i++)
        if (student_id[i]==z)
            break;

    if (i==STUDENT_NUM)
        printf("Not FOUND");
    else
        printf("Found at position:%d", i);
}
```

Ψάχνει τον φοιτητή με την ταυτότητα z. Αν υπάρχει τέτοιος φοιτητής εκτυπώνει την θέση του στον πίνακα, ειδ' αλλιώς μήνυμα λάθους.

3) Γραμμική Αναζήτηση & Ενημέρωση

- Υποθέστε ύπαρξη δυο παράλληλων πινάκων με ίδιο μέγεθος (student_id και student_grade)

- Γράψετε προγραμμα που αναζητά μέσα στον student_id τον φοιτητή με αριθμό ταυτότητας id και ενημερώνει στον student_grade την βαθμολογία (του φοιτητή id) με την τιμή 100

- Αν δεν υπάρχει ο συγκεκριμένος φοιτητής δώστε το κατάλληλο μήνυμα λάθους

3) Γραμμική Αναζήτηση & Ενημέρωση

```
#include <stdio.h>
#define STUDENT_NUM 3              // Εδώ χρησιμοποιούμε μόνο 3 φοιτητές αντί 54
int main() {
      int i;  int student_id[]={12345, 37349, 9995};   float student_grade[]={90.3,78.2,45.3};
      int id = 9995;

      for(i=0; i<STUDENT_NUM; i++) {
            if (student_id[i]==id) {
                  student_grade[i]=100;
                  break;
            }
      }

      if (i==STUDENT_NUM)      {        printf("Not FOUND\n");                          }
      else                     {        printf("Record Updated at position:%d\n", i);   }

      for(i=0;i<STUDENT_NUM;i++)
            printf("Student with id: %d, grade: %f\n", student_id[i], student_grade[i]);
}
```

3) Γραμμική Αναζήτηση & Ενημέρωση

Αποτέλεσμα του προγράμματος της προηγούμενης διαφάνειας:

Record Updated at position:2
Student with id: 12345, grade: 90.300003
Student with id: 37349, grade: 78.199997
Student with id: 9995, grade: 100.000000

Τι θα αλλάζαμε αν θέλαμε να δώσουμε σε όλους με βαθμό από 93 και πανω την τιμή 100.

4) Μέτρηση Στοιχείων

- Γράψετε ένα πρόγραμμα που υπολογίζει και επιστρέφει των αριθμών φοιτητών που παίρνουν βαθμό από 80 και πάνω.
- Οι βαθμοί είναι αποθηκευμένοι σε πίνακα.

```
#define SIZE 3
float student_grade[]={90.3,78.2,45.3};
```

- Το μέγεθος του πίνακα περιέχεται στην παράμετρο SIZE.

4) Μέτρηση Στοιχείων

```
#include <stdio.h>
#define STUDENT_NUM 3

int main() {

    int i;
    float student_grade[]={90.3,78.2,45.3};
    int count = 0;

    for(i=0; i<STUDENT_NUM; i++) {
        if (student_grade[i]>=80) {
            count++;
        }
    }

    printf("Υπάρχουν %d φοιτητές με 80 και πάνω", count);
}
```

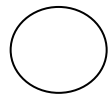
5) Αλγόριθμος Ταξινόμησης Πίνακα SelectioSort

- Μας δίδεται ένας πίνακας αριθμών. Θέλουμε να τον ταξινομήσουμε.
- Υπάρχουν πολλοί αλγόριθμοι. Ένας τέτοιος αλγοριθμος είναι ο SelectionSort.

- Η *SelectionSort* βασίζεται στα ακόλουθα τρία βήματα:
  1. επιλογή του ελάχιστου στοιχείου
  2. ανταλλαγή με το i-οστό στοιχείο (i είναι μια μεταβλητή που αυξάνεται κατά ένα).
  3. επανάληψη των βημάτων 1 και 2 για τα υπόλοιπα στοιχεία.

5) Παράδειγμα Selection Sort

Θέση	0	1	2	3	4	5
Αρχικός Πίνακας	*34*	***8***	*64*	*51*	*32*	*33*
Με i=0	8	34	64	51	**32**	33
Με i=1	8	32	64	51	34	**33**
Με i=2	8	32	33	51	**34**	64
Με i=3	8	32	33	34	**51**	64
Με i=4	8	32	33	34	51	**64**
Τελικός Πίνακας	**8**	**32**	**33**	**34**	**51**	**64**



Στοιχείο που θα εισαχθεί στην θέση i

5) Υλοποίηση Selection Sort

```
#include <stdio.h>
#define SIZE 1

int main() {
    int A[]={34, 8, -21, 64, 51, 32, 33};
    int pos, temp;
    int i,j;

    for (i=0; i<SIZE-1; i++) {
        // θέση επόμενου μικρότερου στοιχείου
        pos=i;

        // βρές το μικρότερο στοιχείο
        for (j = i+1; j < SIZE; j++) {
            if (A[j]<A[pos])
                pos=j;          // θέση μικρότερου
        }

        // αντάλλαξε το A[i] με το  A[pos]
        temp = A[i];
        A[i] = A[pos];
        A[pos] = temp;
    }
```

Στο τέλος μπορούμε να εκτυπώσουμε τον ταξινομημένο πίνακα

```
// Εκτύπωση Πίνακα
for (i=0; i<SIZE; i++) {
    printf("%d, ", A[i]);
}
```

5) Εκτέλεση Selection Sort

BEFORE:[8,4,8,43,3,5,2,1,10,]

Swapping 8 <-> 1
[1,**4,8,43,3,5,2,8,10**,]
Swapping 4 <-> 2
[1,2,8,43,3,5,4,8,10,]
Swapping 8 <-> 3
[1,2,3,43,8,5,4,8,10,]
Swapping 43 <-> 4
[1,2,3,4,8,5,43,8,10,]
Swapping 8 <-> 5
[1,2,3,4,5,8,43,8,10,]
Swapping 8 <-> 8
[1,2,3,4,5,8,43,8,10,]
Swapping 43 <-> 8
[1,2,3,4,5,8,8,43,10,]
Swapping 43 <-> 10
[1,2,3,4,5,8,8,10,43,]

AFTER:[1,2,3,4,5,8,8,10,43,]