# 5ο Φροντιστήριο Δομημένου Προγραμματισμού

## Άσκηση 1 – Αντιστροφή δύο τιμών (swap) με δείκτες

Να γραφτεί πρόγραμμα που διαβάζει δύο αριθμούς από τον χρήστη και ανταλλάσσει τις τιμές τους χρησιμοποιώντας μία εξωτερική συνάρτηση: void swap(int *a, int *b);

```
#include <stdio.h>

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main() {
    int x, y;
    printf("Δώσε δύο αριθμούς: ");
    scanf("%d %d", &x, &y);

    swap(&x, &y);

    printf("Μετά το swap: x = %d, y = %d\n",x,y);
    return 0;
}
```

**Επεξήγηση**

- Η συνάρτηση λαμβάνει διευθύνσεις των x και y.
- Ο δείκτης *a προσπελαύνει την τιμή της μεταβλητής x.
- Η ανταλλαγή γίνεται απευθείας στη μνήμη — όχι αντίγραφα.
- Η swap(&x, &y) περνάει τη διεύθυνση των μεταβλητών.

## Άσκηση 2 – Εύρεση μέγιστου με δείκτες

Να γραφτεί πρόγραμμα που διαβάζει 3 αριθμούς και μέσω συνάρτησης:
int *findMax(int *a, int *b, int *c);
επιστρέφει δείκτη προς τον μεγαλύτερο αριθμό.

```
#include <stdio.h>

int* findMax(int *a, int *b, int *c) {
    int *max = a;
    if (*b > *max) max = b;
    if (*c > *max) max = c;
    return max;
}

int main() {
    int x, y, z;
    printf("Δώσε τρεις αριθμούς: ");
    scanf("%d %d %d", &x, &y, &z);

    int *maxPtr = findMax(&x, &y, &z);
    printf("Μέγιστος αριθμός: %d\n", *maxPtr);

    return 0;
}
```

**Επεξήγηση**

- Η συνάρτηση συγκρίνει τις τιμές που δείχνουν οι δείκτες.
- Επιστρέφει δείκτη σε όποια μεταβλητή είναι μεγαλύτερη.
- Στο τέλος τυπώνεται ο μέγιστος μέσω *maxPtr.

## Άσκηση 3 – Μετατροπή πεζών σε κεφαλαία σε string

Να γραφτεί συνάρτηση που λαμβάνει ένα string και μετατρέπει όλα τα γράμματα σε κεφαλαία με χρήση pointer:
void toUpper(char *str);

```
#include <stdio.h>
#include <ctype.h>

void toUpper(char *str) {
    while (*str != '\0') {
        *str = toupper(*str);
        str++;
    }
}

int main() {
    char text[50];
    printf("Δώσε κείμενο: ");
    fgets(text, 50, stdin);

    toUpper(text);
    printf("Με κεφαλαία: %s", text);

    return 0;
}
```

**Επεξήγηση**

- Ο pointer str κινείται χαρακτήρα-χαρακτήρα.
- Η toupper() μετατρέπει το γράμμα.
- Το while (*str != '\0') τερματίζει στο τέλος του string.

## Άσκηση 4 – Υπολογισμός μέσου όρου με δείκτες και πίνακα

Να γραφτεί πρόγραμμα που διαβάζει Ν αριθμούς σε πίνακα και υπολογίζει τον μέσο όρο μέσω pointer arithmetic.

```
#include <stdio.h>

double average(int *arr, int n) {
    int sum = 0;
    for(int i = 0; i < n; i++)
        sum += *(arr + i);
    return (double)sum / n;
}

int main() {
    int n;
    printf("Πόσα στοιχεία; ");
    scanf("%d", &n);

    int nums[n];
    printf("Δώσε %d αριθμούς: ", n);
    for(int i = 0; i < n; i++)
        scanf("%d", &nums[i]);

    printf("Μέσος όρος = %.2f\n", average(nums,
n));
    return 0;
}
```

**Επεξήγηση**

- Το *(arr + i) χρησιμοποιεί pointer arithmetic.
- Ο pointer μετακινείται i θέσεις δεξιά.
- Η average επιστρέφει double.

## Άσκηση 5 – Αντιστροφή πίνακα με δείκτες

Να γίνει συνάρτηση που αντιστρέφει τα στοιχεία ενός πίνακα in-place:
void reverse(int *arr, int n);

```
#include <stdio.h>

void reverse(int *arr, int n) {
    int *start = arr;
    int *end = arr + n - 1;

    while (start < end) {
        int temp = *start;
        *start = *end;
        *end = temp;
        start++;
        end--;
    }
}

int main() {
    int arr[5] = {1,2,3,4,5};
    reverse(arr, 5);

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

    return 0;
}
```

**Επεξήγηση**

- Δημιουργούμε δύο δείκτες: αρχή και τέλος.
- Τους φέρνουμε προς το κέντρο, ενώ ανταλλάσσουμε τιμές.

## Άσκηση 6 – Εύρεση χαρακτήρα σε string

Να δημιουργηθεί συνάρτηση που βρίσκει την πρώτη εμφάνιση ενός χαρακτήρα σε string και επιστρέφει pointer προς αυτόν.

```
#include <stdio.h>

char* findChar(char *str, char ch) {
    while (*str != '\0') {
        if (*str == ch)
            return str;
        str++;
    }
    return NULL;
}

int main() {
    char text[50] = "hello world";
    char *pos = findChar(text, 'o');

    if (pos)
        printf("Βρέθηκε στη θέση: %ld\n", pos - text);
    else
        printf("Δεν βρέθηκε.\n");

    return 0;
}
```

**Επεξήγηση**

- Αν το βρει, επιστρέφει pointer στη θέση του χαρακτήρα.
- Η θέση υπολογίζεται με pos - text.

## Άσκηση 7 – Κυκλική μετατόπιση πίνακα δεξιά

Να γίνει πρόγραμμα που μετατοπίζει έναν πίνακα 1 θέση δεξιά:
[1,2,3,4] → [4,1,2,3]
με χρήση pointers.

```
#include <stdio.h>

void rotateRight(int *arr, int n) {
    int last = arr[n-1];
    for (int i = n-1; i > 0; i--)
        arr[i] = arr[i-1];
    arr[0] = last;
}

int main() {
    int arr[4] = {1,2,3,4};

    rotateRight(arr, 4);

    for(int i=0;i<4;i++)
        printf("%d ", arr[i]);

    return 0;
}
```

**Επεξήγηση**

- Αποθηκεύουμε το τελευταίο στοιχείο.
- Μετακινούμε όλα τα στοιχεία μία θέση δεξιά.
- Το πρώτο στοιχείο γίνεται το αρχικό τελευταίο.

## Άσκηση 8 – Μετρητής λέξεων με pointers

Να γίνει συνάρτηση που μετρά πόσες λέξεις υπάρχουν σε ένα string.
Λέξη = ακολουθία χαρακτήρων ανάμεσα σε κενά.

```
#include <stdio.h>
#include <ctype.h>

int countWords(char *str) {
    int count = 0;
    int inWord = 0;

    while (*str != '\0') {
        if (!isspace(*str) && !inWord) {
            inWord = 1;
            count++;
        }
        else if (isspace(*str)) {
            inWord = 0;
        }
        str++;
    }
    return count;
}

int main() {
    char text[100] = "Hello world this is C";
    printf("Λέξεις: %d\n", countWords(text));
    return 0;
}
```

**Επεξήγηση**

- Ο pointer σαρώνει χαρακτήρα-χαρακτήρα.
- Όταν εντοπιστεί μη κενός χαρακτήρας και δεν είμαστε ήδη σε λέξη → νέα λέξη.
- isspace() εντοπίζει κενά.