Α' Εξάμηνο
ΕΙΣΑΓΩΓΗ ΣΤΟ ΔΟΜΗΜΕΝΟ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

Εργαστήριο – 8η εβδομάδα.
Κοζάνη, 26 Νοεμβρίου 2008.

Οι ασκήσεις αυτής της σειράς βασίζονται σε παραδείγματα από το βιβλίο του Β. Σεφερίδη "C για αρχάριους" και αφορούν τη χρησιμοποίηση αρχείων κειμένου. Έχουν υποστεί μερικές αλλαγές εμπνευσμένες από άλλα παραδείγματα στο βιβλίο των Kernighan και Ritchie "Η γλώσσα προγραμματισμού C".

Άσκηση p7-1
Αυτό το πρόγραμμα επιτρέπει τη δημιουργία ενός σειριακού αρχείου κειμένου όπου αποθηκεύουμε εγγραφές με το όνομα και τον πληθυσμό διαφόρων χωρών (φυσικά, είναι εύκολο να το τροποποιήσουμε για να αποθηκεύσουμε άλλες και περισσότερων ειδών πληροφορίες).

Επισημαίνουμε τη χρήση της "δικής μας" συνάρτησης get_string για την ανάγνωση αλφαριθητικών από την πρότυπη είσοδο, εμπνευσμένης από την υλοποίηση της fgets που παρουσιάζεται στο βιβλίο των K & R, σελ. 229. Στο βιβλίο του Σεφερίδη χρησιμοποιείται η συνάρτηση βιβλιοθήκης gets. Αυτή όμως, σήμερα θεωρείται κακή επιλογή από πλευράς ασφάλειας επειδή δεν ελέγχει το μέγεθος της εισόδου που δέχεται. Αυτό σημαίνει "τρύπα ασφάλειας" (security hole) για ένα σύστημα που χρησιμοποιεί τη gets για να πάρει δεδομένα από το χρήστη. Έτσι, μπορεί να δεχτεί αυθαίρετα μεγάλη είσοδο που θα οδηγήσει, στην καλύτερη περίπτωση σε κατάρρευση (crash) λόγω παραβίασης των επιτρεπτών ορίων της μνήμης και στη χειρότερη, σε διείσδυση λογισμικού που θα προσβάλλει το σύστημα, θα υποκλέψει πληροφορίες κλπ.

Η συνάρτηση scanf από την άλλη, αν και γενικά είναι αρκετά ευέλικτη, στην περίπτωση των αλφαριθμητικών βρίσκεται κάπως πιο δύσχρηστη, ίσως επειδή ακριβώς είναι πολύ "έξυπνη". Π.χ αν θέλουμε να εισάγουμε μία φράση με κενά διαστήματα, όπως "Ενωμένα Αραβικά Εμιράτα", το πρώτο κενό διάστημα θα εκληφθεί ως τέλος του αλφαριθμητικού και θα περάσει μόνο η λέξη "Ενωμένα". Συναρτήσεις όπως η fgets και η get_string που παρουσιάζεται εδώ, διαβάζουν συγκεκριμένο αριθμό χαρακτήρων που διαβάζονται όλοι εκτός από τον '\n' που σηματοδοτεί και τον τερματισμό της ανάγνωσης, οπότε το αλφαριθμητικό τερματίζεται με τον κενό χαρακτήρα '\0'. Έτσι, μπορούμε να δώσουμε αλφαριθμητικά με κενά, ενώ με απλό πάτημα του Enter χωρίς άλλους χαρακτήρες, μπορούμε να δώσουμε και το κενό αλφαριθμητικό που με τη σειρά του μπορεί να χρησιμοποιηθεί ως συνθήκη εξόδου, για την επαναρχικοποίηση ενός buffer κλπ.

Ακόμη επισημαίνουμε ότι το πρόγραμμα μπορεί να μετατραπεί με μια ελάχιστη αλλαγή (στην εντολή ανοίγματος) ώστε να ανοίγει το αρχείο για γράψιμο (δηλαδή να καταργεί τα όποια προϋπάρχοντα περιεχόμενα) ή προσθήκη (δηλαδή να προσθέτει νέες εγγραφές χωρίς να σβήνει τις παλιές).

Γράψτε το ακόλουθο πρόγραμμα σε ένα αρχείο με το όνομα π.χ. p7_1.c, μεταγλωττίστε το και εκτελέστε το:

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

```
#define SIZE 40

char *get_string(char *, int );

int main(int argc, char **argv)
{
    FILE *fp;
    int population;
    char filename[SIZE], country[SIZE], tempstr[SIZE];

    printf("Δώσε όνομα αρχείου: ");
    get_string(filename,SIZE);
/*    fp=fopen(filename, "w"); */
    fp=fopen(filename, "a");

    while(1) {
        printf("Δώσε όνομα κράτους (ENTER για τέλος): ");
        if(!get_string(country,SIZE))
            break;
        printf("Δώσε πληθυσμό σε εκατομμύρια: ");
        get_string(tempstr, SIZE);
        population = atoi(tempstr);
        fprintf(fp, "%s %d\n", country, population);
        fflush(fp); /* you need this if, e.g.,
                       you interrupt execution by Ctrl-C */
    }
    fclose(fp);

    return 0;
}

char *get_string(char *s, int n)
{ /* adapted from K & R's fgets */
    int c;
    char *cs;

    cs = s;
    while(--n > 0) {
        if((c = getchar()) == '\n')
            break;
        *cs++ = c;
```

```
    }
    *cs = '\0';
    fflush(stdin);
    return (cs == s) ? NULL : s;
}
```

Ερωτήσεις:
Τι έξοδο παίρνετε;
Τι σημαίνει η οδηγία #define;
Τι κάνει η εντολή while(1) { ... } και πώς φεύγουμε από αυτή τη δομή επανάληψης;
Τι κάνει η συνάρτηση atoi ;
Τι κάνει ο τριαδικός τελεστής (? :) στην get_string;

Άσκηση p7-2
Αφού γράψαμε τα δεδομένα σε ένα αρχείο θα θέλαμε να είμαστε σε θέση να τα διαβάσουμε. Αυτό το πρόγραμμα διαβάζει τα δεδομένα από το αρχείο που δημιούργησε ή ενημέρωσε το προηγούμενο και απλώς τα εμφανίζει στην οθόνη (φυσικά, θα μπορούσαμε να κάνουμε, άλλη, πιο πολύπλοκη επεξεργασία στις εγγραφές που διαβάζονται με αυτό τον τρόπο).

Γράψτε το ακόλουθο πρόγραμμα σε ένα αρχείο με το όνομα π.χ. p7_2.c, και μεταγλωττίστε το. Με γαλάζιο είναι ο κώδικας που παραμένει ίδιος με πριν και με κόκκινο τα νέα στοιχεία ή οι αλλαγές.

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

#define SIZE 40

char *get_string(char *, int );

int main(int argc, char **argv)
{
    FILE *fp;
    int population;
    char filename[SIZE], country[SIZE];

    printf("Δώσε όνομα αρχείου: ");
    get_string(filename,SIZE);
    fp=fopen(filename, "r");

    while(fscanf(fp, "%s %d", country, &population) != EOF)
        printf("%s %d εκατομμύρια\n", country, population);

    fclose(fp);

    return 0;
```

```
}

char *get_string(char *s, int n)
{ /* adapted from K & R's fgets */
    int c;
    char *cs;

    cs = s;
    while(--n > 0) {
        if((c = getchar()) == '\n')
            break;
        *cs++ = c;
    }
    *cs = '\0';
    fflush(stdin);
    return (cs == s) ? NULL : s;
}
```

Μετά εκτελέστε το δίνοντάς του το όνομα του αρχείου που δημιουργήσατε με το προηγούμενο πρόγραμμα.

Ερωτήσεις:
Τι έξοδο παίρνετε;
Πώς ερμηνεύετε την παρουσία της fscanf στη συνθήκη της while;

Άσκηση p7-3
Με το πρόγραμμα p7-1, φτιάξαμε ένα αρχείο με εγγραφές ποικίλου μήκους που διαχωρίζονται μεταξύ τους από την αλλαγή γραμμής (χαρακτήρας '\n'). Αυτό το αρχείο μπορεί να προσπελαστεί σειριακά. Αν όμως έχουμε να κάνουμε με πολύ μεγάλα αρχεία, π.χ. μια βάση δεδομένων με χιλιάδες εγγραφές, θα θέλαμε να είμαστε σε θέση να βρίσκουμε πιο γρήγορα όποια εγγραφή μας ενδιαφέρει, χωρίς να διαβάσουμε πρώτα όλες τις προηγούμενες. Δηλαδή, θέλουμε να φτιάξουμε ένα αρχείο τυχαίας προσπέλασης. Αυτό θα το πετύχουμε πολύ εύκολα κάνοντας όλες τις εγγραφές να έχουν το ίδιο μήκος.

Αντιγράψτε το πρόγραμμα *p7_1.c* σε ένα αρχείο με το όνομα π.χ. p7_1.c, και μεταγλωττίστε το. Με γαλάζιο είναι ο κώδικας που παραμένει ίδιος *με το πρώτο πρόγραμμα p7_1c*, αυτής της σειράς και με κόκκινο τα νέα στοιχεία ή οι αλλαγές. Ουσιαστικά αλλάζει μόνο το αλφαριθμητικό μορφοποίησης στην εντολή fprintf, έτσι ώστε τα ονόματα να έχουν 20 διαστήματα πλάτος και στοίχιση αριστερά, ενώ οι πληθυσμοί να έχουν πλάτος 15 διαστημάτων.

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

#define SIZE 40

char *get_string(char *, int );
```

```
int main(int argc, char **argv)
{
    FILE *fp;
    int population;
    char filename[SIZE], country[SIZE], tempstr[SIZE];

    printf("Δώσε όνομα αρχείου: ");
    get_string(filename,SIZE);
/*    fp=fopen(filename, "w"); */
    fp=fopen(filename, "a");

    while(1) {
        printf("Δώσε όνομα κράτους (ENTER για τέλος): ");
        if(!get_string(country,SIZE))
            break;
        printf("Δώσε πληθυσμό σε εκατομμύρια: ");
        get_string(tempstr, SIZE);
        population = atoi(tempstr);
        fprintf(fp, "%-20s%15d\n", country, population);
        fflush(fp); /* you need this if, e.g.,
                       you interrupt execution by Ctrl-C */
    }
    fclose(fp);

    return 0;
}

char *get_string(char *s, int n)
{ /* adapted from K & R's fgets */
    int c;
    char *cs;

    cs = s;
    while(--n > 0) {
        if((c = getchar()) == '\n')
            break;
        *cs++ = c;
    }
    *cs = '\0';
    fflush(stdin);
    return (cs == s) ? NULL : s;
```

```
}
```

Μετά εκτελέστε το δίνοντας του διαφορετικά αριθμητικά ορίσματα από τη γραμμή εντολών, όπως και προηγουμένως.

Ερωτήσεις:
Τι έξοδο παίρνετε τώρα;

Άσκηση p7-4
Όπως και με το δεύτερο πρόγραμμα, θα θέλαμε να μπορούμε να διαβάσουμε τις εγγραφές του αρχείου τυχαίας προσπέλασης που δημιουργήσαμε. Αυτό θα το πετύχουμε με τη βοήθεια της fseek, αφού ξέρουμε ότι οι εγγραφές έχουν σταθερό μήκος (συγκεκριμένα, ίσο με 36).

Γράψτε το επόμενο πρόγραμμα σε ένα αρχείο με το όνομα π.χ. p7_4.c, και μεταγλωττίστε το. Με γαλάζιο είναι ο κώδικας που παραμένει ίδιος *σε σχέση με το δεύτερο πρόγραμμα, p7_2.c*, και με κόκκινο τα νέα στοιχεία ή οι αλλαγές.

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

#define SIZE 40
#define LENGTH 36

char *get_string(char *, int );

int main(int argc, char **argv)
{
    FILE *fp;
    int population, rec_no;
    char filename[SIZE], country[SIZE], tempstr[SIZE];

    printf("Δώσε όνομα αρχείου: ");
    get_string(filename,SIZE);
    fp=fopen(filename, "r");

    while(1) {
        printf("Ποια εγγραφή θέλετε να δείτε; (0 για τέλος)\n");
        scanf("%d", &rec_no);
        if(rec_no == 0)
            break;
        fseek(fp, LENGTH * (rec_no -1), SEEK_SET);
        if(fscanf(fp, "%20s%15d\n", country, &population) != EOF)
             printf("Χώρα:              %s\nΠληθυσμός (εκ.) %d\n",
country, population);
        else
```

```
            printf("Εκτός ορίων του αρχείου!\n");
    }
    fclose(fp);

    return 0;
}

char *get_string(char *s, int n)
{ /* adapted from K & R's fgets */
    int c;
    char *cs;

    cs = s;
    while(--n > 0) {
        if((c = getchar()) == '\n')
            break;
        *cs++ = c;
    }
    *cs = '\0';
    fflush(stdin);
    return (cs == s) ? NULL : s;
}
}
```

Μετά εκτελέστε το δίνοντας του διαφορετικά αριθμητικά ορίσματα από τη γραμμή εντολών, όπως και προηγουμένως.

Ερωτήσεις:
Τι έξοδο παίρνετε τώρα;
Γιατί η παράμετρος LENGTH της fseek έχει οριστεί να έχει την τιμή 36;

Άσκηση p7-5
Οι δυνατότητες που μας παρέχουν τα αρχεία τυχαίας προσπέλασης φαίνονται καλύτερα στο επόμενο παράδειγμα όπου μπορούμε να διορθώσουμε ή ακόμη και να διαγράψουμε εγγραφές από το αρχείο μας. Πρέπει να σημειωθεί βέβαια ότι πρόκειται για *λογικές* και όχι *φυσικές* διαγραφές. Δηλαδή η εγγραφή γίνεται κενή ή μηδενίζεται, αλλά εξακολουθεί να καταλαμβάνει την αρχική μνήμη. Για πλήρη, φυσική διαγραφή, θα έπρεπε όλες οι επόμενες εγγραφές να μετακινηθούν κατά μία θέση προς τα πάνω.

Γράψτε το επόμενο πρόγραμμα σε ένα αρχείο με το όνομα π.χ. p7_5.c, και μεταγλωττίστε το. Οι διαφορές με κόκκινο είναι σε σχέση με το προηγούμενο πρόγραμμα, p7_4.c.

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

```
#define SIZE 40
#define LENGTH 36

char *get_string(char *, int );

int main(int argc, char **argv)
{
    FILE *fp;
    int population, rec_no, field_no;
    char filename[SIZE], country[SIZE];

    printf("Δώσε όνομα αρχείου: ");
    get_string(filename,SIZE);
    fp=fopen(filename, "r+");

    printf("Ποια εγγραφή θέλετε να δείτε; (0 για τέλος)\n");
    scanf("%d", &rec_no);

    fseek(fp, LENGTH * (rec_no -1), SEEK_SET);
    fscanf(fp, "%20s%15d\n", country, &population);
     printf("Χώρα:              %s\nΠληθυσμός (εκ.) %d\n", country,
population);
       printf("Ποιο πεδίο θέλετε να διορθώσετε; 1 ή 2; (0 για
διαγραφή)\nΕπιλογή:");
    scanf("%d", &field_no);
    getchar();
    switch(field_no) {
        case 0:
            fseek(fp, -LENGTH, SEEK_CUR);
            fprintf(fp, "%-20s%15d", NULL, 0);
            break;
        case 1:
            printf("Δώστε το νέο όνομα: ");
            get_string(country, SIZE);
            fseek(fp, -LENGTH, SEEK_CUR);
            fprintf(fp, "%-20s", country);
            break;
        case 2:
            printf("Δώστε το νέο πληθυσμό: ");
            scanf("%d", &population);
            fseek(fp, -LENGTH+20, SEEK_CUR);
            fprintf(fp, "%15d", population);
```

```
            break;
    }
    fclose(fp);

    return 0;
}

char *get_string(char *s, int n)
{ /* adapted from K & R's fgets */
    int c;
    char *cs;

    cs = s;
    while(--n > 0) {
        if((c = getchar()) == '\n')
            break;
        *cs++ = c;
    }
    *cs = '\0';
    fflush(stdin);
    return (cs == s) ? NULL : s;
}
```

Μετά εκτελέστε το δοκιμάζοντας διαφορετικές περιπτώσεις. Κάθε φορά ανοίγετε το αρχείο για να δείτε το αποτέλεσμα (ή δείτε τα περιεχόμενά του χρησιμοποιώντας το προηγούμενο πρόγραμμα).

Ερωτήσεις:
Πώς ερμηνεύετε τα ορίσματα της fseek στις διάφορες εμφανίσεις της;