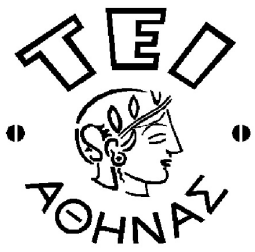

Τεχνολογικό Εκπαιδευτικό Ίδρυμα Αθήνας
Τμήμα Μηχανικών Πληροφορικής Τ.Ε
Προγραμματισμός Υπολογιστών (Εργαστήριο)
Εργασία 07

Στέφανος Στεφάνου
Αριθμός Μητρώου : 161118
Τμήμα Α3(Τετάρτη 8:00-10:00)

Τα παρακάτω προγράμματα γραφτήκανε και δοκιμαστήκανε κάτω απο ZorinOS 64bit(Ubuntu based) και gcc έκδοση 5.4.0 κάτω απο το πρότυπο c99(ISO/IEC 9899:1999)

Άσκηση 1)

Να γραφεί πρόγραμμα το οποίο θα διαβάζει τα στοιχεία Ν φοιτητών από την standard είσοδο και στην συνέχεια θα τα γράφει σε αρχείο κειμένου.

Κώδικας Άσκησης 1)

```
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#define FAIL NULL
#define SUCCESS 1
#define ERR 2
#define FOPEN_ERROR -1
/*
 * Συνάρτηση openfile()
 * Ανοίγει το αρχείο σε κατάσταση append ("a")
 * @:arg
 *      δεν λαμβάνει καμια παράμετρο
 * @:returns
 *      NULL σε περίπτωση αποτυχίας ανοίγματος αρχείου
 *      εναν δείκτη σε δεδομένα τύπου FILE (struct) assosiated με ενα αρχείο ,
το οποίο δείχνει στο τέλος (κατάσταση append)
 */
FILE *openfile() { return fopen("data.txt","a"); }
/*
 * Συνάρτηση writeintoopenedfile(FILE **,char **)
 * Γράφει πάνω στο αρχείο /Stream το οποίο δίνεται ως πρώτη παράμετρος , τα
δεδομένα που δίνονται στον buffer (δευτερη παράμετρος)
 * Προσοχή! πρέπει να έχει ήδη ανοιχτεί το αρχειο /Stream πριν την κλήση της
writeintofile και να εχε δεσμευτεί δυναμικα
 * ο buffer
 * @:arg
 *      FILE ** file_descriptor -> Εναν pointer προς pointers τύπου FILE
 *      char ** Buffer -> η περιοχή ενδιάμεσης μνήμης προς εγραφή
 * @:returns ακαίρεο (int)
 *      SUCCESS (1) Σε περίπτωση επιτυχίας
 *      ERR(2) Σε διαφορετική περίπτωση
 */
int writeintoopenedfile(FILE **file_descriptor,char **Buffer){
```

```
    fflush(*file_descriptor);
    if(fprintf(*file_descriptor,*Buffer)){
        return SUCCESS;
    }
    else return ERR;
}
/*
 * Συνάρτηση INIT_MEMOPY(char **Buffer)
 * Κάνει τις απαραίτητες δεσμεύσεις στην μνήμη για την περιοχή ενδιάμεσης
αποθήκευσης Buffer
 * @:arg
 *      char ** Buffer , εναν διπλο pointer προς την περιοχή ενδιάμεσης
αποθήκευσης
 * @:no_returns (void)
 */
void INIT_MEMORY(char **Buffer){
    *Buffer=(char *)malloc(200*sizeof(char));
    if(*Buffer==FAIL){
        fprintf(stderr,"cant allocate memory!");
        exit(EXIT_FAILURE);
    }
    memset(*Buffer,0,200*sizeof(char));
}
/*
 * Σύνάρτηση EXIT_PROCCESS(File **file_pointer,char **Buffer)
 * Κάνει την απαραίτηση απελευθέρωση μνήμης και κλείνει το ρεύμα εξόδου που
ανοίχτηκε με την openfile()
 * @:arg
 *        FILE **file_pointer -> ενα ανοιχτό stream προς αρχείο
 *        char **Buffer -> εναν pointer που δείχνει σε δυναμικα δεσμευμένο
πίνακα!!!!
 * @:no_returns!
 *
 * @:warn
 *      το ρεύμα θα πρέπει να είναι ανοιγμένο με μια κληση της openfile()
 *      Η μνήμη πρέπει να είναι δυναμικά δεσμευμένη με μια κλήση της
INIT_MEMORY
 *
 */
void EXIT_PROCCESS(FILE **file_pointer,char **Buffer){
    fclose(*file_pointer);
    free(*Buffer);
}
void Call_UI(char **Buffer,FILE **file_pointer){
    int num,i;
    size_t buffsize=200;//Δήλωση μεγέθους buffer (το size_t είναι ενα εσωτερικό
typedef της c -> unsigned int)
    printf("Δώσε αριθμό μαθητών");
    scanf("%d",&num);
    for (i = 0; i <=num; ++i) {
        if(i==0){
            fflush(*file_pointer);
            getline(Buffer,&buffsize,stdin);
            continue;
        }
        printf("%d)Δώσε στοιχεία φοιτητη\n",i);
        fflush(stdin);
```

```
        getline(Buffer,&buffsize, stdin);
        if(writeintoopenedfile(file_pointer,Buffer)==SUCCESS){
            printf("%s -> [Success into data.txt]\n",*Buffer);
        }
        else{
            printf("%s -> [Error into data.txt\n",*Buffer);
        }
    }
}
int main() {
    char *Buffer;                                   //Δήλωση Buffer
    FILE *file_pointer=openfile();                  //Άνοιγμα αρχείου
    INIT_MEMORY(&Buffer);                           //Αρχικοποίηση Μνήμης
    Call_UI(&Buffer,&file_pointer);                 //Κλήση συνάρτησης που υλοιποιεί το
UserInterface
    EXIT_PROCCESS(&file_pointer,&Buffer);   //Εξοδος
}
```

Άσκηση 2)

Να γραφεί πρόγραμμα το οποίο θα διαβάζει από αρχείο κειμένου τα στοιχεία όλων των φοιτητών που βρίσκονται σε αυτό και στη συνέχεια θα τυπώνει στην standard έξοδο τα στοιχεία του φοιτητή με τον μεγαλύτερο μέσο όρο.

```
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#define FAIL NULL
#define SUCCESS 1
#define ERR 2
#define AM_Size 6
#define NAME_Size 40
#define GIVE_ME_MAX 1
/*
 * Struct Record
 * Αποθηκεύει τα δεδομένα του κάθε φοιτητη
 *
 */
struct record{
    char AM[AM_Size+1];
    char NAME[NAME_Size+1];
    char SURNAME[NAME_Size+1];
    int semester;
    int  Marks[5];
};
/*
 * Συνάρτηση opefile
 * Ανοίγει το αρχείο data.txt προς αναγνωση (reading mode "r")
 *
 * @:no_args
 *
 * @returns
 *       FAIL(NULL) σε περίπτωση αποτυχίας
 *       FILE * fp -> εναν file pointer που δείχνει το αρχειο data.txt
 *
 */
```

```
FILE *openfile(){
    FILE *fp = fopen("data.csv","r");
    if(fp==FAIL){
        fprintf(stderr,"[CantOpenFileException :P -> ]Cant open file,check if
exists,give me sudo permissions maybe? :P ");
        return FAIL;
    }
    else return fp;
}
/*
 * Συνάρτηση EXIT_PROCCESS(FILE **,char **)
 * Αποδεσμεύει τον δυναμικά δεσμευμένο Buffer και κλείνει το ρεύμα εξοδου
 * @:arg
 *      Εναν δείκτη σε δείκτες τύπου FILE -> το ανοιχτό ρεύμα εξόδου που
πρόκειται να κλείσει
 *      Εναν δείκτη προς τον Buffer Εισόδου -> ο Buffer που θα αποδεσμευτεί !
 */
void EXIT_PROCCESS(FILE **file_pointer,char **Buffer){
    fclose(*file_pointer);
    free(*Buffer);
}
/*
 * Συνάρτηση CalculateMo(struct record *TempRawData)
 * @:returns double
 *              Τον αριθμο που αντιπροσωπεύει τον μέσο όρο !
 * @:args
 *        struct record *TempRawData -> ενας δείκτης σε δομή που περιέχει
στοιχεία φοιτητή
 *
 */
double CalculateMO(struct record *TempRawData){
    int i,temp=0;
    for(i=0;i<5;i++){
        temp+=TempRawData->Marks[i];
    }
    return temp/5;
}
/*
 * Συνάρτηση HandleStudentRawData(struct record *,struct record *)
 * Χειρίζεται μια νεά εισαγωγή δεδομένων απο το αρχείο σε μορφη struct record
 * Ελένχει αν είναι ο μεγαλυτερος μεσος όρος μεχρι στιγμης και προβαίνει στις
κατάλληλες ενεργειες
 *
 *
 * @:no_return
 *
 *
 * @:args
 *        struct record *TempRawData -> ενας δείκτης σε δομή που περιέχει
στοιχεία φοιτητή
 *        struct record *MaxStudent -> ενας δείκτης σε δομή η οποία περιέχει τα
στοιχεία του καλύτερου φοιτητη μεχρι στιγμής
 *
 */
void HandleStudentRawData(struct record *TempRawData,struct record
*MaxStudent){
    static double maxMO=0;
```

```
    double tempMO=CalculateMO(TempRawData);
    if(tempMO>maxMO){
        maxMO=tempMO;
        *MaxStudent=*TempRawData;
    }
}
/*
 * Συνάρτηση show_results (struct record *)
 *
 * Εκτυπωνει τα στοιχεία ενος φοιτητή
 *
 * @:arg
 *       struct record *PrintRecord -> τα στοιχεία ενος φοιτητή προς εκτύπωση!
 * @:no_return
 */
void show_results(struct record *PrintRecord){
    int i;
    printf("ID : %s\n",PrintRecord->AM);
    printf("NAME : %s\n",PrintRecord->NAME);
    printf("SURNAME : %s\n",PrintRecord->SURNAME);
    printf("SEMESTER : %d\n",PrintRecord->semester);
    for(i=0;i<5;i++){
        printf("LESSON %d WITH MARK %d\n",i,PrintRecord->Marks[i]);
    }
}
/*
 * Συνάρτηση INIT_MEMORY(char ** )
 * Αρχικοποιεί τον Buffer εισόδου (200 bytes)
 * @:args
 *       char **Buffer -> ο δείκτης που θα δείχνει στον buffer εισόδου
 * @:returns
 *         FAIL(NULL) -> Σε περίπτωση σφαλματος στην δεσμεύση χώρου
 *         SUCCESS(1)  ->Σε περίπτωση επιτυχίας αρχικοποιησης
 */
int INIT_MEMORY(char **Buffer,struct record *MaxStudent,struct record
*TempStudent){
    *Buffer = (char *)malloc(sizeof(char)*250);
    if(*Buffer==NULL){
        return FAIL;
    }
    memset(*Buffer,0,250);
    memset(MaxStudent,0,sizeof(struct record));
    memset(TempStudent,0,sizeof(struct record));
}
int main(){
    struct record MaxStudent,TempStudent;               //Δημιουργία μεταβλητών
-δομών για μεγιστο και προσωρινά δεδομένα
    int maxMarkMO=0,num,i;                              //Μεγιστος μεσος
όρος,αριθμος γραμμών και μετρητής βρόνχου
    char *Buffer;                                       //pointer στον Buffer
    INIT_MEMORY(&Buffer,&MaxStudent,&TempStudent);      //Αρχικοποίηση
μεταβλητών
    FILE *file_pointer=openfile();                      //Δημιουργια του
file_pointer
    size_t BufferSize=250;                              //Μέγεθος Buffer
    fscanf(file_pointer,"%d\n",&num);                   //Διάβασμα γραμμών απο
το αρχειο και αγνόηση ενος \n
```

```
    for(i=0;i<num;i++){
        fflush(stdin);
        getline(&Buffer,&BufferSize,file_pointer);      //Φόρτωση στην μνήμη
της κάθε γραμμης
                                                        //Ξεδιάλεξη Δεδομένων
        sscanf(Buffer,"%6[^,],%s
%[^,],%d,%d,%d,%d,%d,%d",TempStudent.AM,TempStudent.NAME,TempStudent.SURNAME,&T
empStudent.semester,&TempStudent.Marks[0],&TempStudent.Marks[1],&TempStudent.Ma
rks[2],&TempStudent.Marks[3],&TempStudent.Marks[4]);
        HandleStudentRawData(&TempStudent,&MaxStudent); //Σύγριση των δεδομένων
    }
    show_results(&MaxStudent); //Εκτύπωση μεγαλύτερου
}
```