313CA
IORDACHE MĂDĂLINA GABRIELA

17.02.2021

Examen P.C.

1.

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

struct *data_set {
        char nume_produs[32];
        int aparitii;
// Funcția main
int main ( int argc, char *argv[] )
{
    // numărul de fișiere
    int n = atoi(argv[0]);
    // stocul limită
    int K = atoi(argv[1]);
    struct data_set my_data_set[1024]
        // maximum 1024 de produse definite

    File *fp;
    int counter = 0;  // numărul de elemente din vector

  for(int i=0; i<n; ++i) {
    char nume_fișier[32];  // 32 - dimensiunea numelui fișierului

    strcpy(nume_fișier, argv[2+n]; // 2+n - sărim peste primele două argumente

    fp = fopen(nume_fișier, "rt");

    char line[50];  // 50 - dimensiunea unei linii
```

```
    while (fgets(fp, 50, line) {
        char nume_produs[32];
        int aparitii = 0;
        sscanf(line, "%s %d", nume_produs, &aparitii)
          // citim numele produsului si aparitiile lui
        put_in_vector(nume_produs, aparitii, my_data_set,
                      &counter)
    }
}

fclose(fp);  // Inchidem fisier
// deschidem fisier de iesire
File *out_f = fopen("inventar.txt", "wt");
  // iteram prin produse
for (int i=0; i<1024; i++) {
    // cantitate mai mica decat K
    if (my_data_set[i].aparitii < K) {
        printf(out_f, "%s %d \n", my_data_set[i],
               nume_produs, aparitii);
        // printam pe ecran
    }
    fprintf(out_f, "%s %d", my_data_set[i].nume_fisier,
            my_data_set[i].aparitii);

}
fclose(out_f);  // inchidem fisier

return 0;

}
```

```
void pun_in_vector(char nume[32], int aparitii, struct data_set,
                   my_data_set[1024], int *counter);
{
// Adauga in vector produsul cu cantitatea respectiva
int found = 0;
for(int i=0; i<=*counter; i++) {
    // verificam daca gasim deja produsul
    if(strcmp(my_data_set[i], nume_produs, nume)==0) {
        my_data_set[i].aparitii += aparitii;
        found = 1;
    } break; // iesim fiindca nu ne intereseaza urmatoarele
               date
}
if(!found) {
    // nu l-am gasit si il adaugam
    strcpy(my_data_set[i].nume_produs, nume);
    my_data_set[i].aparitii = aparitii;
    (*counter)++;
}
```

2.

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

typedef struct {
    int nv;
    peer * values;
} data_set;

typedef struct {
    long timestamp;
    double value;
} peer;

int main(int argc, char * argv[])
{
    File * file_in;
    file_in = fopen(argv[0], "rb"); // deschidem fișierul binar
    char line[1024]; // linia pentru comandă
    char command[5]; // pentru show = 5 variabile
    char values[1019]; // pentru valorile pos

    fgets(stdin, 1024, line);
    sscanf(line, "%s %s", command, values);
    // aici am împărțit comanda în două
    // prima parte comanda show, iar a doua valorile pozițiilor
    int counter = 0;
    int values_pos[1024]; // considerăm maximum 1024 de poziții
    for(int i = 0; i < strlen(values); ++i) {
        if(values[i] != " ") {
            values_pos[counter++] = atoi(values[i]);
        }
    }
```

```
// am reținut valorile într-un vector
// trebuia pusă la început dar am uitat - scuze.
for (int i=0; i < counter ; i++) {
    int pos = values - pos[i];
    // verificare fișier
    if (! file_in) {
        printf ("Nu am putut deschide fișier");
        exit(-1);
    }

/* Mai trebuie vreo în afara forului, dar nu am unde
 să scriu așa să las aici
*/
// Afla dimensiunea fișierului
int file_len = 0;
fseek (file_in , 0, SEEK_END); //mă duc la final
file_len = ftell (file);
fseek ( file ,0, SEEK_SET ); //mă duc la început
if ( pos > file_len) {
    printf ("Datele nu se află în fișier \n" )
} else {
    struct data_set data ;
    data . values = (struct peer *) malloc (512, sizeof(peer));
    // 512 valori de tip peer în vector
    for (int i=0, i< pos , i++ ) {
        fseek (file_in ,0, SEEK_CUR );
            // mă poziționez în fișier exact după ce s
            citesc datele ca să nu calculez dimensiunea
            datelor.
        fread (& data , sizeof (data ,1 , file_in) ;
    }
```

```
// Am ajuns unde mă interesează ;
memset ( data , 0 , size of ( data ) ;
// getim datele necesare
fread ( & data , sizeof ( data ) , 1 , file_in ) ;
printf ( " Numărul valorilor monitorizate % d \n ", data . nv );
for ( int i = 0 , i < data . nv ; i++ ) {
    printf (" timp măsurare % d \n", values [i]. time_stamp)
    printf (" valoare măsurată % d \n , values [i] . value )
    free ( data . values );
}
    // Ne ducem la începutul fișierului pentru
    a citi următoarele date ;
    fseek ( file_in , 0 , seek_set ) ;
}   // închidem for-ul mare
fclose ( file_in ) ;    // închidem fișierul
return 0 ;
}
```

3. // Nu avem destinație char * my_strcpy (char * dest, char* src){
if (dest == NULL) {
return NULL
}
char *out = dest; // return value
// folosim ca să nu returnăm date moarte,
și doar să semnalăm să nu avem erori
while (*dest++ = src++);
return out;
}
// Ce facem aici e să mergem byte cu byte la fiecare
char *; în spate compilatorul reține v[i] ca fiind *(v+i)
unde v este adresa de start pentru v. În while cu
operatorul "=" atribuim valoarea lui dest cu src până
când operația nu va da fail (SRC != NULL).