



PROGRAMARE PROCEDURALĂ

Bogdan Alexe

bogdan.alexe@fmi.unibuc.ro

Secția Informatică, anul I,

2016-2017

Cursul 10

Cursul trecut

1. Siruri de caractere: funcții specifice de manipulare

Programa cursului

- | | |
|---|---|
| <ul style="list-style-type: none">□ Introducere<ul style="list-style-type: none">• Algoritmi.• Limbaje de programare.• Introducere în limbajul C. Structura unui program C.• Complexitatea algoritmilor.
□ Fundamentele limbajului C<ul style="list-style-type: none">• Tipuri de date fundamentale. Variabile. Constante. Operatori. Expresii. Conversii.• Instrucțiuni de control• Directive de preprocesare. Macrodefiniții.• Funcții de citire/scriere.• Etapele realizării unui program C.
□ Tipuri deriveate de date<ul style="list-style-type: none">• Tablouri. Siruri de caractere.• Structuri, uniuni, câmpuri de biți, enumerări.• Pointeri.
□ Funcții (1)<ul style="list-style-type: none">• Declarare și definire. Apel. Metode de trasmitere a parametrilor.• Pointeri la funcții. | <ul style="list-style-type: none">□ Tablouri și pointeri<ul style="list-style-type: none">▪ Legătura dintre tablouri și pointeri▪ Aritmetică pointerilor▪ Alocarea dinamică a memoriei▪ Clase de memorare
□ Siruri de caractere<ul style="list-style-type: none">▪ Funcții specifice de manipulare.
□ Fișiere text și fișiere binare<ul style="list-style-type: none">▪ Funcții specifice de manipulare.
□ Structuri de date complexe și autoreferite<ul style="list-style-type: none">▪ Definire și utilizare
□ Funcții (2)<ul style="list-style-type: none">▪ Funcții cu număr variabil de argumente.▪ Preluarea argumentelor funcției main din linia de comandă.▪ Programare generică.
□ Recursivitate |
|---|---|
-

Cursul de azi

1. Fișiere: noțiuni generale
2. Fișiere text: funcții specifice de manipulare
3. Fișiere binare: funcții specifice de manipulare.
4. Funcții de poziționare în fișiere text și binare

Fișiere

- **fișier = sir de octeți (colecție de date) memorat pe suport extern (magnetic, optic) și identificat printr-un nume.**

- programe sursă: .c, .cpp
- executabile
- imagini: .jpeg, .jpg, .png, .bmp
- documente: .pdf, .dvi, .eps
- audio: .mp3
- video: .avi, .mp4

- pentru fiecare tip de fișier binar este necesar un program care să cunoască și să interpreze corect datele din fișier

Fișiere

- **fișier = sir de octeți (colecție de date) memorat pe suport extern (magnetic, optic) și identificat printr-un nume.**

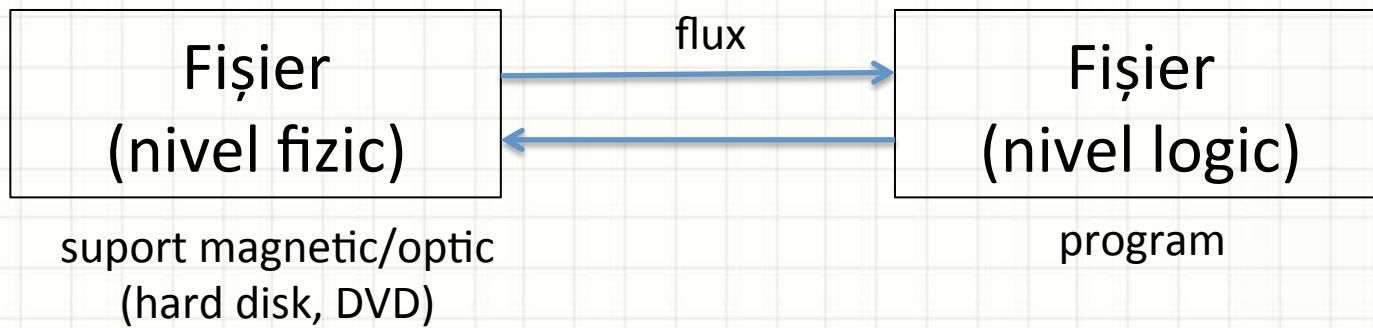
- fișierele sunt entități ale sistemului de operare.
- operațiile cu fișiere se realizează de către sistemul de operare, compilatorul de C traduce funcțiile de acces la fișiere în apeluri ale funcțiilor de sistem; alte limbiage de programare fac același lucru;

- noțiunea de fisier este mai generală
- **fișier = flux de date (stream) = transfer de informație binară (sir de octeți) de la o sursă spre o destinație:**
 - citire: flux de la tastatură (sursă) către memoria internă (destinație)
 - afișare: flux de la memoria internă (sursă) către periferice (monitor, imprimantă)

Fluxuri automate asociate unui program

- **stdin (standard input)** – flux de intrare (citire).
 - asociat implicit cu tastatura.
- **stdout (standard output)** – flux de ieșire (afișare).
 - asociat implicit cu ecranul.
- **stderr (standard error)** – flux de ieșire (afișare) pentru erori.
 - asociat implicit cu ecranul.

Fișiere



- ne referim la un fișier fizic într-un program C printr-o variabilă;
- asocierea dintre numele extern al un fișier (un sir de caractere) și o variabilă dintr-un program C se realizează la deschiderea unui fișier, printr-o funcție standard.

Fișiere

- **tipuri de fișiere:**

- **fișiere text: octeții (caractere ASCII) sunt organizați pe linii.**

Caracterele terminatorii de linii sunt:

- Unix: caracterul line feed (LF) = '\n' – cod ASCII 10
 - Mac OS vechi : caracterul carriage return (CR) = '\r' – cod ASCII 13
 - Windows: CR + LF = '\r\n'
 - un fișier text poate fi terminat printr-un caracter terminator de fișier (EOF = CTRL-Z). Acest terminator nu este obligatoriu. Sfârșitul unui fișier disc poate fi detectat și pe baza lungimii fișierului (număr de octeți), memorată pe disc.

- **fișiere binare: octeții nu sunt organizați în nici un fel (nu există noțiunea de linie)**

Fișiere

□ tipuri de fișiere:

- **fișiere text: octeții (caractere ASCII) sunt organizați pe linii.**

Caracterele terminatorii de linii sunt:

- Unix: caracterul line feed (LF) = '\n' – cod ASCII 10
- Mac OS vechi : caracterul carriage return (CR) = '\r' – cod ASCII 13
- Windows: CR + LF = '\r\n'
- un fișier text poate fi terminat printr-un caracter terminator de fișier (EOF = CTRL-Z). Acest terminator nu este obligatoriu. Sfârșitul unui fișier disc poate fi detectat și pe baza lungimii fișierului (număr de octeți), memorată pe disc.

Acesta este
un
exemplu
fișier fizic

Windows

În Windows dimensiunea fizică a unui fișier = dimensiunea logică dacă avem o singură linie

Acesta este\r\nun\r\n\r\nexemplu
fișier logic

Lucrul cu fișiere

- În biblioteca stdio.h este definită o structură **FILE**. Această structură conține (în membri ei) informații referitoare la un fișier: nume, adresa, adresa bufferului intern în care se procesează (citire/scriere) octetii din fișier, indicator de sfârșit de fișier, indicator de poziție în fișier, etc.
- Lucrul cu fișiere presupune declararea unui pointer la structura **FILE** în vederea realizării legăturii dintre nivelul logic (variabila fișier) și nivelul fizic (numele extern al fișierului) :

FILE * f;

- etapele pentru lucrul cu fișiere:
 - deschiderea unui fișier – funcția **fopen** (legătura nivel logic-fizic);
 - prelucrarea fișierului (citiri/scrieri);
 - închiderea fișierului – funcția **fclose**;

Lucrul cu fișiere

- **deschiderea unui fișier = stabilirea unui flux către acel fișier.** Se realizează folosind funcția fopen:
 - **sintaxa**

FILE *fopen(char *nume_fisier, char *mod_deschidere)

unde

- nume_fisier = numele fisierului
- mod_deschidere = sir de caracter (1-3 caractere) ce precizează tipul de acces la fișier:
 - citire "r", scriere "w", adăugare la sfârșitul fisierului "a";
 - "+" permite scrierea și citirea "r+", "w+", "a+";
 - t (implicit) sau b: specifică tipul de fisier (text sau binar).
- funcția **fopen** întoarce un pointer la o structura **FILE** sau în caz de eroare (fișierul nu se poate deschide) întoarce NULL.

Lucrul cu fișiere

- **deschiderea unui fișier = stabilirea unui flux către acel fișier.** Se realizează folosind funcția fopen:
- **sintaxa**

FILE *fopen(char *nume_fisier, char *mod_deschidere)

unde mod_deschidere = sir de caracter (1-3 caractere) ce precizează tipul de acces la fișier:

Mod	Semnificație
r	Deschide un fișier tip text pentru a fi citit
w	Creează un fișier tip text pentru a fi scris
a	Adaugă într-un fișier tip text
rb	Deschide un fișier de tip binar pentru a fi citit
wb	Creează un fișier de tip binar pentru a fi scris
ab	Adaugă într-un fișier de tip binar
r+	Deschide un fișier tip text pentru a fi citit/scris
w+	Creează un fișier tip text pentru a fi citit/scris
a+	Adaugă în sau creează un fișier tip text pentru a fi citit/scris
r+b	Deschide un text în binar pentru a fi citit/scris
w+b	Creează un fișier de tip binar pentru a fi citit/scris
a+b	Adaugă sau creează un fișier de tip binar pentru a fi citit/scris

Lucrul cu fișiere

- **Închiderea unui fișier = Închiderea unui flux către acel fișier.** Se realizează folosind funcția fclose:
 - **sintaxa**
int *fclose(FILE *f)

unde f = pointer la structura de tip FILE care realizează legătura cu fișierul pe care vreau să-l închid

- funcția **fopen** întoarce valoarea 0 dacă închiderea s-a efectuat cu succes și EOF în caz de eroare. Toate fișierele în care s-a scris trebuie să fie închise. Dacă se realizează doar citirea dintr-un fișier, acesta nu trebuie neapărat închis.
- **tastatura și imprimanta** sunt considerate fișiere text. Ele nu trebuie să fie deschise și închise.

Lucrul cu fișiere

□ exemplu:



```
001 #include<stdio.h>
002
003 int main()
004 {
005     FILE *f;
006     char *numeFisier = "/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/iarna.txt";
007     f = fopen(numeFisier, "r");
008     if(f==NULL)
009     {
010         printf("Eroare la deschiderea fisierului");
011         exit(0);
012     }
013     else
014         printf("Am deschis fisierul %s \n",numeFisier);
015     fclose(f);
016
017     return 0;
018 }
```

Am deschis fisierul /Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/iarna.txt

Pentru Windows, atentie la folosirea lui "\\" pentru a da calea fișierului:
f = fopen("C:\\work\\\\test.txt"); // in Windows

Lucrul cu fișiere

- ❑ **detectarea sfârșitului de fișier.** Se poate realiza și folosind funcția `feof(find end of file)` :
- ❑ **sintaxa**

`int feof(FILE *f)`

unde f = pointer la structura **FILE** corespunzătoare fișierului pe care îl prelucrez.

- ❑ funcția **feof** returnează 0 dacă nu s-a ajuns la sfârșitul fișierului la ultima operație de citire sau o valoare nenulă dacă s-a ajuns la sfârșitul fișierului.

ATENTIE:

```
while (!feof(f))  
    citeste x din f;  
    scrie x
```

se scrie un x (= -1) în plus

varianta corecta

```
while (1)  
    citeste x din f;  
    if (feof(f)) break;  
    scrie x
```

Cursul de azi

1. Fișiere: noțiuni generale
2. Fișiere text: funcții specifice de manipulare
3. Fișiere binare: funcții specifice de manipulare.
4. Funcții de poziționare în fișiere text și binare

Fișiere text

- accesul la fișierele text se poate face la nivel de **șir de caractere** (linie) sau la nivel de **caracter** (octet).

- un fișier text se accesează ca o succesiune de linii de text de lungime variabilă (încheiate cu un terminator de linie : '\n') utilizând un set dedicat de funcții din biblioteca standard.

- funcțiile de citire sau de scriere cu format din/în fișiere text realizează conversia automată din:
 - format extern (șir de caractere) în format intern (binar) - la citire
 - format intern (binar) în format extern (șir de caractere), la scriere pentru numere întregi sau reale.

Functii de citire/scriere la nivel de caracter

- **int fgetc(FILE *f)** întoarce codul ASCII al caracterului citit din fișierul f.
 - dacă s-a ajuns la finalul fișierului sau a avut loc o eroare la citire întoarce EOF (= -1).
- **int fputc(int c, FILE *f)** scrie caracterul cu codul ASCII c în fișierul f.
 - întoarce EOF (= -1) în caz de eroare sau codul ASCII al caracterului scris în caz de succes.

Functii de citire/scriere la nivel de caracter

Exemplu: copierea unui fisier text

exempluCitirecaracter.c

```
1 #include<stdio.h>
2
3 int main()
4 {
5     FILE *fin, *fout;
6     char *numeFisier = "/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/iarna.txt";
7     fin = fopen(numeFisier, "r");
8
9     fout = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/iarna_copie.txt", "w");
10
11    if ((fin == NULL) || (fout == NULL))
12    {
13        printf("Eroare la deschiderea fisierelor");
14        exit(0);
15    }
16
17    printf("Incepem copierea fisierului\n");
18
19    char c;
20    while (c=fgetc(fin) != EOF)
21        fputc(c, fout);
22    printf("Am terminat copierea fisierului\n");
23
24    if ((fclose(fin) != 0) || (fclose(fout) != 0))
25    {
26        printf("Eroare la inchiderea fisierelor");
27        exit(0);
28    }
29
30    return 0;
```

P/curs10/exempluCitirecaracter
Incepem copierea fisierului
Am terminat copierea fisierului
- . . . - . . . -

Functii de citire/scriere la nivel de caracter

Exemplu: copierea unui fisier text

The screenshot shows a code editor window titled "exempluCitirecaracter.c". The code is a C program that reads from a file named "iarna.txt" and writes to a file named "iarna_copie.txt". It includes error handling for file opening and displays messages indicating the start and end of the copy process.

```
#include<stdio.h>

int main()
{
    FILE *fin, *fout;
    char *numeFisier = "/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/iarna.txt";
    fin = fopen(numeFisier,"r");

    fout = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/iarna_copie.txt","w");

    if ((fin == NULL) || (fout == NULL))
    {
        printf("Eroare la deschiderea fisierelor");
        exit(0);
    }

    printf("Incepem copierea fisierului\n");

    char c;
    while (c=fgetc(fin) != EOF)
        fputc(c,fout);
    printf("Am terminat copierea fisierului\n");
}
```

P/curs10/exempluCitirecaracter
Incepem copierea fisierului
Am terminat copierea fisierului

Name	Date Modified	Size	Kind
.DS_Store	Today, 6:18 PM	8 KB	Document
iarna_copie.txt	Today, 6:55 PM	147 KB	Plain Text
iarna.txt	Today, 6:51 PM	147 KB	Plain Text

Functii de citire scriere la nivel de linie

- `char* fgets(char *sir, int m, FILE *f)`
 - citește maxim $m-1$ caractere sau până la '\n' și pune șirul de caractere în sir (adaugă la sfârșit '\0').
 - returnează adresa șirului citit.
 - dacă apare vreo eroare întoarce NULL.

- `int fputs(char *sir, FILE *f)`
 - scrie șirul sir în fișierul f, fără a pune '\n' la sfârșit.
 - întoarce numarul de caractere scrise, sau EOF in caz de eroare.

Functii de citire scriere la nivel de linie

- Exemplu: aflarea numărului de linii al unui fișier

The screenshot shows a code editor window titled "exempluCitireLinii.c". The code is a C program that reads a file and counts the number of lines. The code uses standard input-output functions like `fopen`, `fgets`, and `fclose`. It includes error handling for file opening. The output window below the editor shows the result of running the program on a file named "iarna.txt".

```
#include<stdio.h>
int main()
{
    FILE *f;
    char linie[1000];
    int numarLinii = 0;
    f = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/iarna.txt","r");
    if (f == NULL)
    {
        printf("Eroare la deschiderea fisierului");
        exit(0);
    }
    while (fgets(linie,1000,f) != NULL)
        numarLinii++;
    printf("Fisierul are %d liniile\n",numarLinii);
    fclose(f);
    return 0;
}
```

P/curs10/exempluCitireLinii
Fisierul are 242 linii

Functii de citire scriere cu format

- `int fscanf(FILE *f, char *format)`
 - citește din fisierul f folosind un format (analog cu scanf)
- `int fprintf(FILE *f, char *format)`
 - scrie în fișierul f folosind un format (analog cu printf)

Functii de citire scriere cu format

exempluCitireScriereFormat.c

```
1 #include<stdio.h>
2
3 int main()
4 {
5     FILE * f;
6     int x;
7
8     f = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/numere.txt", "w");
9
10    if (f==NULL)
11    {
12        fprintf(stdin,"Nu pot deschide fisierul! \n");
13        exit(1);
14    }
15
16    while(fscanf(stdin,"%d",&x))
17        fprintf(f,"%d \n",x);
18
19    fclose(f);
20
21    f = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/numere.txt", "r");
22    while(fscanf(f,"%d",&x) != EOF)
23        fprintf(stdout,"%d \n",x);
24
25
26    return 0;
27
28 }
```



numere.txt

```
23
-1000
0
11
```

```
23
-1000
0
11
```

Lucrul cu fișiere

- **detectarea sfârșitului de fișier.** Se poate realiza și folosind funcția feof(find end of file) :
- **sintaxa**

int feof(FILE *f)

unde f = pointer la fișierul pe care îl prelucrez.

- funcția feof returnează 0 dacă nu s-a ajuns la sfârșitul fișierului la ultima operație de citire sau o valoare nenulă dacă s-a ajuns la sfârșitul fișierului.

ATENTIE:

```
while (!feof(f))  
    citeste x din f;  
    scrie x
```

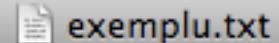
se scrie un x (= -1) în plus

varianta corecta

```
while (1)  
    citeste x din f;  
    if (feof(f)) break;  
    scrie x
```

Lucrul cu fișiere

- detectarea sfârșitului de fișier. Se poate realiza și folosind funcția feof(find end of file) :



```
exempluFeof.c
```

```
1
2
3     int main()
4     {
5         FILE *f;
6         int c;
7
8         f = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/exemplu.txt", "r");
9
10        if (f==NULL)
11        {
12            fprintf(stdin,"Nu pot deschide fisierul! \n");
13            exit(1);
14        }
15
16        printf("Incepem citirea din fisier \n");
17
18
19        while (!feof(f))
20        {
21            c = fgetc(f);
22            printf("%d%c",c,c);
23        }
24
25        printf("\n");
26        fclose(f);
27
28        return 0;
29    }
```

```
A
B
C
```

```
Incepem citirea din fisier
65A10
66B10
67C-1?
```

Lucrul cu fișiere

- detectarea sfârșitului de fișier. Se poate realiza și folosind funcția feof(find end of file) :

The screenshot shows a C IDE interface with the following components:

- Title Bar:** exempluFeof.c
- Code Editor:** The code is as follows:

```
1  int main()
2  {
3      FILE *f;
4      int c;
5
6      f = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/exemplu.txt", "r");
7
8      if (f==NULL)
9      {
10         fprintf(stdin,"Nu pot deschide fisierul! \n");
11         exit(1);
12     }
13
14     printf("Incepem citirea din fisier \n");
15
16     while (1)
17     {
18         c = fgetc(f);
19         if(feof(f))
20             break;
21         printf("%d%c",c,c);
22     }
23
24     printf("\n");
25     fclose(f);
26
27     return 0;
28 }
```

- Output Window:** The output shows the program's execution:

- Line A: Incepem citirea din fisier
- Line B: 65A10
- Line C: 66B10
- Line D: 67C

Cursul de azi

1. Fișiere: noțiuni generale
2. Fișiere text: funcții specifice de manipulare
3. Fișiere binare: funcții specifice de manipulare.
4. Funcții de poziționare în fișiere text și binare

Fișiere binare

- **fișier binar = sir de octeți neformatat pe linii care este stocat pe suport magnetic/optic. Octeții nu sunt considerați ca fiind coduri de caractere.**
- un fișier binar este format în general din articole de lungime fixă, fără separatori între articole. Un articol poate conține:
 - un singur octet
 - un număr binar (pe 2, 4 sau 8 octeți)
 - structură cu date de diferite tipuri
- un fișier binar se accesează ca o succesiune de octeți, cărora funcțiile de citire și scriere din fișier nu le dau nici o interpretare.
- un fișier text se accesează ca o succesiune de linii de text de lungime variabilă (încheiate cu un terminator de linie : '\n') utilizând un set dedicat de funcții din biblioteca standard.

Fisiere binare

- **fișier binar = sir de octeți neformatat pe linii care este stocat pe suport magnetic/optic. Octeții nu sunt considerați ca fiind coduri de caractere.**
- **FILE *fopen(char *nume_fisier, char *mod_deschidere)**
 - nume_fisier = numele fisierului
 - mod_deschidere = sir de caracter ce precizează tipul de acces la fisier:

Mod	Semnificație
r	Deschide un fisier tip text pentru a fi citit
w	Creează un fisier tip text pentru a fi scris
a	Adaugă într-un fisier tip text
rb	Deschide un fisier de tip binar pentru a fi citit
wb	Creează un fisier de tip binar pentru a fi scris
ab	Adaugă într-un fisier de tip binar
r+	Deschide un fisier tip text pentru a fi citit/scris
w+	Creează un fisier tip text pentru a fi citit/scris
a+	Adaugă în sau creează un fisier tip text pentru a fi citit/scris
r+b	Deschide un text în binar pentru a fi citit/scris
w+b	Creează un fisier de tip binar pentru a fi citit/scris
a+b	Adaugă sau creează un fisier de tip binar pentru a fi citit/scris

Fisiere binare

- ❑ fișier binar = sir de octeți neformatat pe linii care este stocat pe suport magnetic/optic. Octeții nu sunt considerați ca fiind coduri de caractere.

exempluMacWin.c

```
1 #include<stdio.h>
2
3 int main()
4 {
5     FILE *f;
6     int c;
7
8     f = fopen("./Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExame", "rb");
9
10    if (f==NULL)
11    {
12        printf("Eroare la deschiderea fisierului");
13        exit(1);
14    }
15
16    while ((c=fgetc(f))!=EOF)
17        printf("%d\n", c);
18
19    fclose(f);
20
21    return 0;
22}
23
```

A B C

exemplu.txt

output pe MAC

```
65
10
66
10
67
Process returned 0 (0x0)    execut
Press ENTER to continue.

C:\Users\Student\D
65
13
10
66
13
10
67
Process returned
Press any key to
```

In Windows output-ul va arăta aşa încrât CR+LF ('\r' + '\n') este terminator de linie (nu se translatează într-un LF ('\n'))

Functii de citire scriere

- `int fwrite(void *tablou, int dim_element, int nr_elem, FILE *f)`
 - scrie în fișierul referit de f cel mult nr_elem elemente de dimensiune dim_element de la adresa tablou;
- `int fread(void *tablou, int dim_element, int nr_elem, FILE *f)`
 - citește cel mult nr_elem elemente de dimensiune dim_element din fisierul referit de f la adresa tablou.

Functii de citire scriere

Exemplul 1

exempluFisierBinar.c

```
1 #include<stdio.h>
2
3 int main()
4 {
5
6     FILE *f,*g;
7     int x = 1094861636;
8     f = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/numarBinar.out","wb");
9     g = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/numarText.out","w");
10    if ((f==NULL) || (g==NULL))
11    {
12        printf("Eroare la deschiderea fisierelor");
13        exit(1);
14    }
15
16    fwrite(&x,sizeof(int),1,f);
17    fprintf(g,"%d",x);
18
19    fclose(f);
20    fclose(g);
21
22
23    return 0;
24
25 }
```

Functii de citire scriere

Exemplul 1

exempluFisierBinar.c

```
1 #include<stdio.h>
2
3 int main()
4 {
5
6     FILE *f,*g;
7     int x = 1094861636;
8     f = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/numarBinar.out","wb");
9     g = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/numarText.out","w");
10    if ((f==NULL) || (g==NULL))
11    {
12        printf("Eroare la deschiderea fisierelor");
13        exit(1);
14    }
15
16    fwrite(&x,sizeof(int),1,f);
17    fprintf(g,"%d",x);
18
19    fclose(f);
20    fclose(g);
21
22
23    return 0;
24 }
```

Scrierea lui 1094861636 in baza 2:

0 1 0 0 0 0 0 1

0 1 0 0 0 0 1 0

0 1 0 0 0 0 1 1

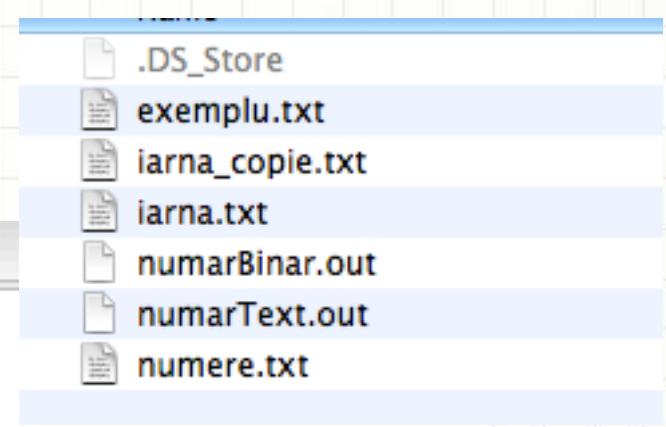
0 1 0 0 0 1 0 0

65 = A

66 = B

67 = C

68 = D



numarText.out



numarBinar.out

Octetii se scriu de la cel mai putin semnificativ la cel mai semnificativ

Functii de citire scriere

□ Exemplul 2

exempluFisierBinar2.c

```
1 #include<stdio.h>
2
3 int main()
4 {
5
6     FILE *f;
7     int x;
8     f = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/numarBinar.out", "rb");
9     if (f==NULL)
10    {
11        printf("Eroare la deschiderea fisierului");
12        exit(1);
13    }
14    while(fread(&x,sizeof(int),1,f)==1)
15        printf("x=%d\n",x);
16    fclose(f);
17
18    char c;
19    f = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/numarBinar.out", "rb");
20    while(fread(&c,sizeof(char),1,f)==1)
21        printf("c=%d\n",c);
22    fclose(f);
23
24    return 0;
25
26 }
```

```
P/curs10/exempluFisierBinar2
x=1094861636
c=68
c=67
c=66
c=65
```

Functii de citire scriere

Exemplul 3

The screenshot shows a C IDE interface. On the left is a code editor window titled "exempluFisierBinar3.c" containing the following C code:

```
1 #include<stdio.h>
2
3 int main()
4 {
5
6     FILE *f1,*f2,*f3;
7     f1 = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/numere1.txt","wb");
8     f2 = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/numere2.txt","wb");
9     f3 = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/numere3.txt","wb");
10    if ((f1==NULL) || (f2==NULL) || (f3==NULL))
11    {
12        printf("Eroare la deschiderea fisierelor");
13        exit(0);
14    }
15    int v[5] = {33,40,50,10,80},i;
16    //varianata 1
17    for(i=0;i<5;i++)
18    {
19        fwrite(&v[i],sizeof(int),1,f1);
20    }
21    //varianata 2
22    fwrite(v,sizeof(int),5,f2);
23    //varianata 3
24    fwrite(v,5*sizeof(int),1,f3);
25
26    fclose(f1);
27    fclose(f2);
28    fclose(f3);
29    return 0;
30
31 }
```

To the right of the code editor are three terminal windows showing the output of the program:

- The first terminal window shows the output for file "numere1.txt":

```
!|(2
P|
```
- The second terminal window shows the output for file "numere2.txt":

```
!|(2
P|
```
- The third terminal window shows the output for file "numere3.txt":

```
!|(2
P|
```

Below the terminals, the text "Coduri ASCII:" is displayed in red, followed by the conversion table:
33 = !, 40 = (, 50 = 2, 10 = \n, 80 = P

Functii de citire scriere

Exemplul 4

exempluFisierBinar4.c

```
int main()
{
    FILE *f1,*f2,*f3;
    f1 = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/numere1.txt","rb");
    f2 = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/numere2.txt","rb");
    f3 = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/numere3.txt","rb");
    if ((f1==NULL) || (f2==NULL) || (f3==NULL))
    {
        printf("Eroare la deschiderea fisierelor");
        exit(0);
    }
    int v[5],i;
    for(i=0;i<5;i++)
    {
        fread(&v[i],sizeof(int),1,f2); printf("%d ",v[i]);
    }
    printf("\n");
    //varianata 2
    fread(v,sizeof(int),5,f3);
    for(i=0;i<5;i++)
        printf("%d ",v[i]);
    printf("\n");
    //varianata 3
    fread(v,5*sizeof(int),1,f1);
    for(i=0;i<5;i++)
        printf("%d ",v[i]);
    printf("\n");

    fclose(f1);
    fclose(f2);
    fclose(f3);
```

P/curs10/exempluFisierBinar4

33 40 50 10 80

33 40 50 10 80

33 40 50 10 80

----- *1----- M----- D----- L-----

Functii de citire scriere

□ Exemplul 5: scrierea unei structuri

exempluFisierBinar5.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 typedef struct
5 {
6     int varsta;
7     char nume[20];
8 } student;
9
10 int main()
11 {
12     FILE *f;
13     student st;
14     int i;
15     f = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/date.in","wb");
16     if (f==NULL)
17     {
18         printf("Fisierul date.in nu se poate crea \n");
19         exit(0);
20     }
21     printf("Nume student = ");scanf("%s",&st.nume);
22     printf("Varsta student = ");scanf("%d",&st.varsta);
23     fwrite(&st,sizeof(st),1,f);
24
25     fclose(f);
26     return 0;
27 }
28
```

Functii de citire scriere

Exemplul 5: scrierea unei structuri

exempluFisierBinar5.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 typedef struct
5 {
6     int varsta;
7     char nume[20];
8 } student;
9
10 int main()
11 {
12     FILE *f;
13     student st;
14     int i;
15     f = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/date.in","wb");
16     if (f==NULL)
17     {
18         printf("Fisierul date.in nu se poate crea \n");
19         exit(0);
20     }
21     printf("Nume student = ");scanf("%s",&st.nume);
22     printf("Varsta student = ");scanf("%d",&st.varsta);
23     fwrite(&st,sizeof(st),1,f);
24
25     fclose(f);
26     return 0;
27 }
```

```
Nume student = Ioana
Varsta student = 20
```

```
Process returned 0 (0x0)    execution time : 2.
Press ENTER to continue.
```

date.in

Ioana

Caracterul cu codul ASCII = 20 e
neprintabil

```
Nume student = Ioana
Varsta student = 45
```

```
Process returned 0 (0x0)    execution time : 3.
Press ENTER to continue.
```

date.in

Ioana

Functii de citire scriere

□ Exemplul 5: citirea unei structuri

exempluFisierBinar6.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 typedef struct
5 {
6     int varsta;
7     char nume[20];
8 } student;
9
10 int main()
11 {
12     FILE *f;
13     student st;
14     int i,n;
15     f = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/date.in","rb");
16     if (f==NULL)
17     {
18         printf("Fisierul date.in nu se poate deschide \n");
19         exit(0);
20     }
21     while(fread(&st,sizeof(st),1,f))
22         printf("%s are %d ani\n",st.nume,st.varsta);
23     fclose(f);
24     return 0;
25 }
```

P/curs10/exempluFisierBinar6
Ioana are 45 ani

Functii de citire scriere

- Exemplul 6: manipularea cheii de sortare și a poziției unei structuri într-un fișier.

exempluFisierBinar7.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 typedef struct
5 {
6     int varsta;
7     char nume[20];
8 } student;
9
10 int main()
11 {
12     FILE *f;
13     student st;
14     int i,n;
15     f = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/date.in","wb");
16     if (f==NULL)
17     {
18         printf("Fisierul date.in nu se poate crea \n");
19         exit(0);
20     }
21     printf("n=");
22     scanf("%d",&n);
23     for(i=0;i<n;i++)
24     {
25         printf("Nume student = ");scanf("%s",&st.nume);
26         printf("Varsta student = ");scanf("%d",&st.varsta);
27         fwrite(&st,sizeof(st),1,f);
28     }
29     fclose(f);
30 }
```

```
n=5
Nume student = Ioana
Varsta student = 19
Nume student = Alexandra
Varsta student = 20
Nume student = Bogdan
Varsta student = 22
Nume student = Vali
Varsta student = 25
Nume student = George
Varsta student = 34
```



Vreau să sorteze elementele structurii după câmpul nume în ordine lexicografică

Functii de citire scriere

- Exemplul 6: manipularea cheii de sortare și a poziției unei structuri într-un fișier.

The screenshot shows a code editor window titled "exempluFisierBinar8.c". The code is a C program that reads student data from a binary file and sorts it based on name. The sorted data is then written to another binary file. The code uses a struct to represent a student, an array of students, and arrays for indices and auxiliary variables. It includes file operations like fopen and fwrite, and sorting logic using strcmp and swapping indices.

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

typedef struct
{
    int varsta;
    char nume[20];
} student;

int main()
{
    FILE *f,*h;
    student st[5];
    int i,j,aux,index[5];
    f = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/date.in","rb");
    for(i=0;i<5;i++)
        {fread(&st[i],sizeof(st[i]),1,f); index[i] = i;}
    for (i=0;i<5;i++)
        for (j=i+1;j<5;j++)
            if(strcmp(st[index[i]].nume,st[index[j]].nume)>0)
                {aux = index[i];index[i] = index[j]; index[j] = aux;}
    for(i=0;i<5;i++)
        printf("%s\n",st[index[i]].nume);
    h = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/date_sortate.in","wb");
    for(i=0;i<5;i++)
        fwrite(&st[index[i]],sizeof(st[index[i]]),1,h);
    fclose(f);
    fclose(h);
    return 0;
}
```

P/curs10/exempluFisierBinar8

Alexandra
Bogdan
George
Ioana
Vali

Cursul de azi

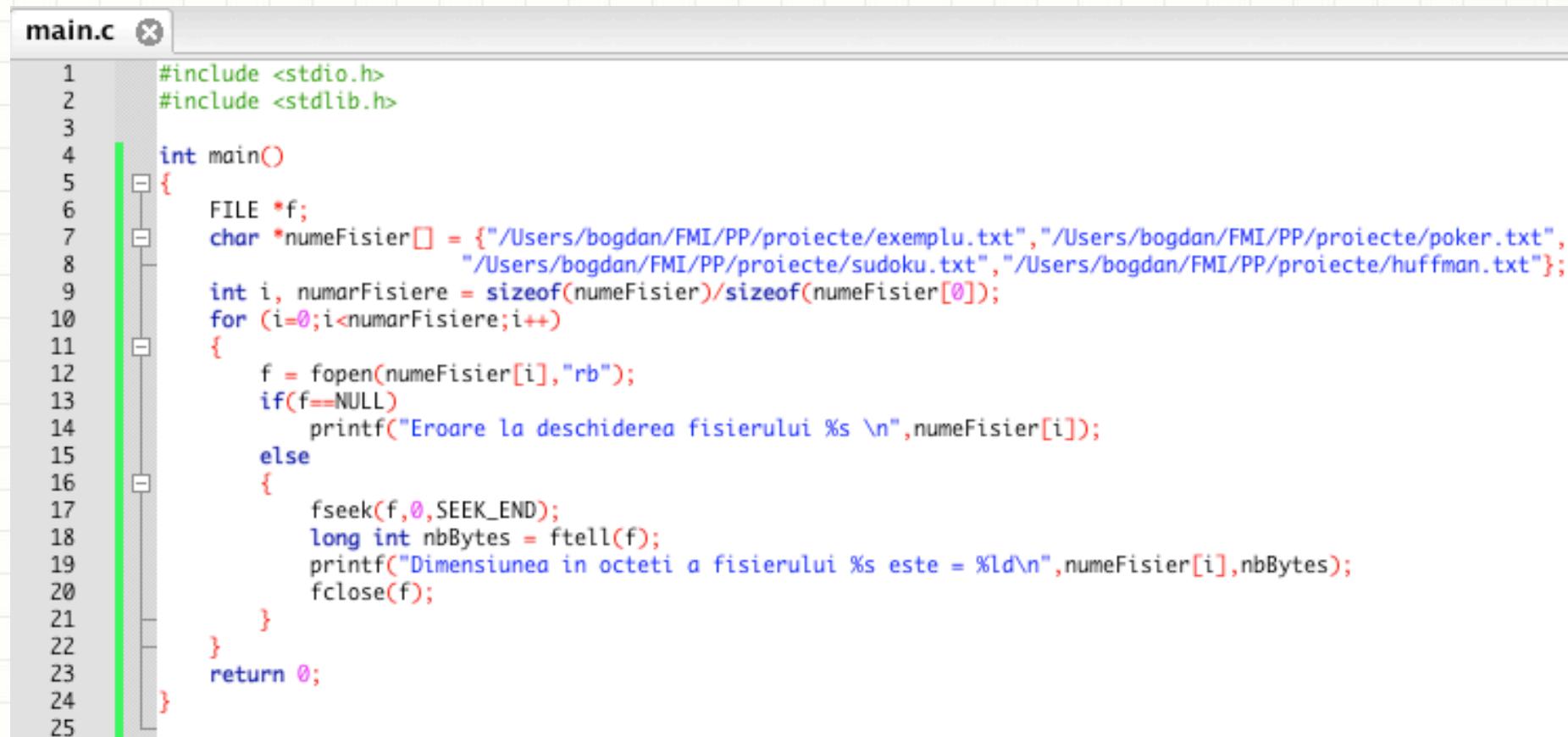
1. Fișiere: noțiuni generale
2. Fișiere text: funcții specifice de manipulare
3. Fișiere binare: funcții specifice de manipulare.
4. Funcții de poziționare în fișiere text și binare

Functii de pozitionare intr-un fisier

- În C ne putem poziționa pe un anumit octet din fișier.
Functiile care permit poziționarea (cele mai importante) sunt:
- **long int ftell(FILE *f)**
 - Întoarce numărul octetului curent față de începutul fișierului;
 - (dimensiunea maximă a unui fișier în C este de $2^{31}-1$ octeți ~ 2GB)
- **int fseek(FILE *f, int nr_octeti, int origine)**
 - mută pointerul de fișier f pe octetul numărul nr_octeti in raport cu origine
 - origine – 3 valori posibile:
 - SEEK_SET (= 0) - început de fișier
 - SEEK_CUR (=1) – poziția curentă
 - SEEK_END (=2) – sfârșit de fișier

Functii de pozitionare într-un fișier

Exemplu: aflarea dimensiunii unui fișier



```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     FILE *f;
7     char *numeFisier[] = {"/Users/bogdan/FMI/PP/proiecte/exemplu.txt", "/Users/bogdan/FMI/PP/proiecte/poker.txt",
8                           "/Users/bogdan/FMI/PP/proiecte/sudoku.txt", "/Users/bogdan/FMI/PP/proiecte/huffman.txt"};
9     int i, numarFisiere = sizeof(numeFisier)/sizeof(numeFisier[0]);
10    for (i=0;i<numarFisiere;i++)
11    {
12        f = fopen(numeFisier[i],"rb");
13        if(f==NULL)
14            printf("Eroare la deschiderea fisierului %s \n",numeFisier[i]);
15        else
16        {
17            fseek(f,0,SEEK_END);
18            long int nbBytes = ftell(f);
19            printf("Dimensiunea in octeti a fisierului %s este = %ld\n",numeFisier[i],nbBytes);
20            fclose(f);
21        }
22    }
23    return 0;
24 }
```

Functii de pozitionare intr-un fisier

Exemplu: aflarea dimensiunii unui fisier

```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     FILE *f;
7     char *numeFisier[] = {"Users/bogdan/FMI/PP/proiecte/exemplu.txt", "Users/bogdan/FMI/PP/proiecte/poker.txt",
8                           "/Users/bogdan/FMI/PP/proiecte/sudoku.txt", "/Users/bogdan/FMI/PP/proiecte/huffman.txt"};
9     int i, numarFisiere = sizeof(numeFisier)/sizeof(numeFisier[0]);
10    for (i=0;i<numarFisiere;i++)
11    {
12        f = fopen(numeFisier[i],"rb");
13        if(f==NULL)
14            printf("Eroare la deschiderea fisierului %s \n",numeFisier[i]);
15        else
16        {
17            fseek(f,0,SEEK_END);      Dimensiunea in octeti a fisierului /Users/bogdan/FMI/PP/
18            long int nbBytes = ftell(f) proiecte/exemplu.txt este = 5
19            printf("Dimensiunea in octe Dimensiunea in octeti a fisierului /Users/bogdan/FMI/PP/
20                  fclose(f);          proiecte/poker.txt este = 30000
21            }
22        }
23        return 0;
24    }
25 }
```

Dimensiunea in octeti a fisierului /Users/bogdan/FMI/PP/
proiecte/exemplu.txt este = 5
Dimensiunea in octeti a fisierului /Users/bogdan/FMI/PP/
proiecte/poker.txt este = 30000
Dimensiunea in octeti a fisierului /Users/bogdan/FMI/PP/
proiecte/sudoku.txt este = 4899
Dimensiunea in octeti a fisierului /Users/bogdan/FMI/PP/
proiecte/huffman.txt este = 5560714

Process returned 0 (0x0) execution time : 0.004 s
Press ENTER to continue.

Functii de pozitionare intr-un fisier

- Exemplul 6: manipularea cheii de sortare și a poziției unei structuri într-un fișier.

The screenshot shows a code editor window titled "exempluFisierBinar8.c". The code is a C program that reads student data from a binary file and writes it sorted to another binary file. The code includes a struct definition for "student" and logic for reading data from "date.in", sorting it based on name, and writing it to "date_sortate.in". The right side of the editor shows the output of the program, which lists student names: Alexandra, Bogdan, George, Ioana, and Vali.

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

typedef struct
{
    int varsta;
    char nume[20];
} student;

int main()
{
    FILE *f,*h;
    student st[5];
    int i,j,aux,index[5];
    f = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/date.in","rb");
    for(i=0;i<5;i++)
        {fread(&st[i],sizeof(st[i]),1,f); index[i] = i;}
    for (i=0;i<5;i++)
        for (j=i+1;j<5;j++)
            if(strcmp(st[index[i]].nume,st[index[j]].nume)>0)
                {aux = index[i];index[i] = index[j]; index[j] = aux;}
    for(i=0;i<5;i++)
        printf("%s\n",st[index[i]].nume);
    h = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/date_sortate.in","wb");
    for(i=0;i<5;i++)
        fwrite(&st[index[i]],sizeof(st[index[i]]),1,h);
    fclose(f);
    fclose(h);
    return 0;
}
```

P/curs10/exempluFisierBinar8
Alexandra
Bogdan
George
Ioana
Vali

Functii de pozitionare într-un fișier

- Exemplul 6: manipularea cheii de sortare și a poziției unei structuri într-un fișier.

```
exempluPozitionare.c X
1
2
3
4
5
6
7     char nume[20];
8 } student;
9
10 int main()
11 {
12     FILE *f,*h;
13     student st[5];
14     int i,j,aux,index[5];
15     f = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/date.in","rb");
16     for(i=0;i<5;i++)
17         {fread(&st[i],sizeof(st[i]),1,f); index[i] = i;}
18     for (i=0;i<5;i++)
19         for (j=i+1;j<5;j++)
20             if(strcmp(st[index[i]].nume,st[index[j]].nume)>0)
21                 {aux = index[i];index[i] = index[j]; index[j] = aux;}
22     for(i=0;i<5;i++)
23         printf("%s\n",st[index[i]].nume);
24     h = fopen("/Users/bogdan/FMI/PP/Bogdan/2016_2017/SubiecteExamen/date_sortate2.in","wb");
25     student st_aux;
26     for(i=0;i<5;i++)
27     {
28         fseek(f,index[i]*sizeof(student),SEEK_SET);
29         fread(&st_aux,sizeof(student),1,f);
30         fwrite(&st_aux,sizeof(student),1,h);
31     }
32     fclose(f);
33     fclose(h);
34     return 0;
35 }
```

Alte funcții pentru lucrul cu fișierele

- ❑ **void rewind (FILE *f)**
 - ❑ reposiționarea pointerului asociat fișierului la începutul său.
- ❑ **int remove(char * nume_fisier);**
 - ❑ șterge fișierul cu numele = nume_fisier. Întoarce 0 în caz de succes, 1 în caz de eroare;
- ❑ **int rename(char *nume_vechi,char *nume_nou);**
 - ❑ redenumește fișierul cu numele = nume_vechi cu nume_nou. Întoarce 0 în caz de succes, 1 în caz de eroare;
- ❑ **char *tmpnam(char* nume_fisier)**
 - ❑ furnizează un nume de fisier pe care îl pune în nume_fisier care nu există în directorul curent