



# PROGRAMARE PROCEDURALĂ

Bogdan Alexe

[bogdan.alexe@fmi.unibuc.ro](mailto:bogdan.alexe@fmi.unibuc.ro)

Secția Informatică, anul I,

2018-2019

Cursul 6



# Facebook @ University of Bucharest!

Wednesday 7th November | 3 - 5pm  
SPIRU HARET Hall

Our very own Engineers will be hosting a workshop for you to learn more about Software Engineering & Production Engineering!

You will also have the opportunity to hear about the internship and graduate recruitment process from our recruiters!

Please click on the link below to secure your spot! We look forward to seeing you there!

<https://facebookeventsinromania.splashthat.com/>

# Recapitulare – cursul trecut

1. Instrucțiuni de control (break, continue, goto, return).
2. Etapele realizării unui program C.
3. Directive de preprocessare. Macrodefiniții.
4. Funcții de citire/scriere.

# Programa cursului

## □ Introducere

- Algoritmi
- Limbaje de programare.

## □ Fundamentele limbajului C

- Introducere în limbajul C. Structura unui program C.
- Tipuri de date fundamentale. Variabile. Constante. Operatori. Expresii. Conversii.
- Tipuri derivate de date: tablouri, siruri de caractere, structuri, uniuni, câmpuri de biți, enumerări, pointeri
- Instrucțiuni de control
- Directive de preprocesare. Macrodefiniții.
- Funcții de citire/scriere.
- Etapele realizării unui program C.

## □ Fișiere text

- Funcții specifice de manipulare.

## □ Funcții (1)

- Declarație și definire. Apel. Metode de transmitere a parametrilor. Pointeri la funcții.

## □ Tablouri și pointeri

- Legătura dintre tablouri și pointeri
- Aritmetică pointerilor
- Alocarea dinamică a memoriei
- Clase de memorare

## □ Siruri de caractere

- Funcții specifice de manipulare.

## □ Fișiere binare

- Funcții specifice de manipulare.

## □ Structuri de date complexe și autoreferite

- Definire și utilizare

## □ Funcții (2)

- Funcții cu număr variabil de argumente.
- Preluarea argumentelor funcției main din linia de comandă.
- Programare generică.

## □ Recursivitate



# Programa cursului

## □ Introducere

- Algoritmi
- Limbaje de programare.

## □ Fundamentele limbajului C

- Introducere în limbajul C. Structura unui program C.
- Tipuri de date fundamentale. Variabile. Constante. Operatori. Expresii. Conversii.
- Tipuri derivate de date: tablouri, siruri de caractere, structuri, uniuni, câmpuri de biți, enumerări, pointeri
- Instrucțiuni de control
- Directive de preprocessare. Macrodefiniții.
- Funcții de citire/scriere.
- Etapele realizării unui program C.

## Fișiere text

- Funcții specifice de manipulare.

## Fișiere binare

- Funcții specifice de manipulare.



## □ Funcții (1)

- Declarație și definire. Apel. Metode de transmitere a parametrilor. Pointeri la funcții.

## □ Tablouri și pointeri

- Legătura dintre tablouri și pointeri
- Aritmetică a pointerilor
- Alocarea dinamică a memoriei
- Clase de memorare

## □ Siruri de caractere

- Funcții specifice de manipulare.

## □ Structuri de date complexe și autoreferite

- Definire și utilizare

## □ Funcții (2)

- Funcții cu număr variabil de argumente.
- Preluarea argumentelor funcției main din linia de comandă.
- Programare generică.

## □ Recursivitate

# Cuprinsul cursului de azi

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

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

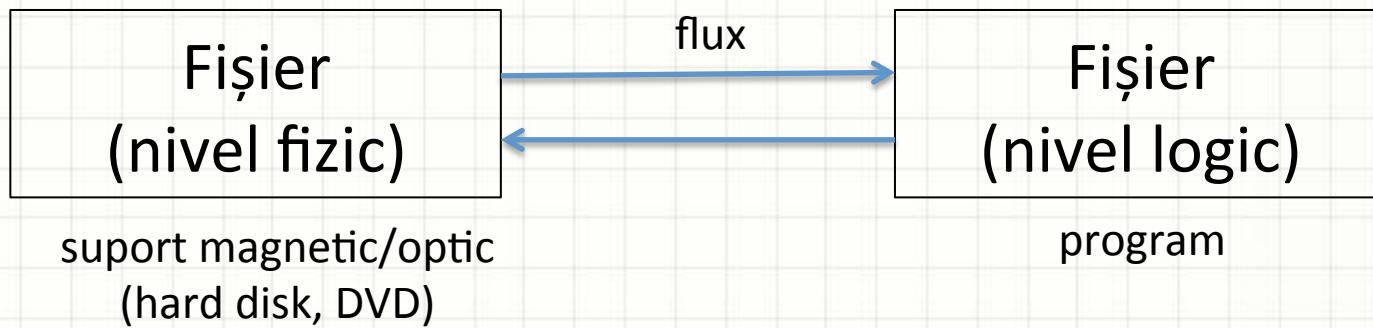
# Fișiere

- **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 (**fopen**).

# Fișiere

- **există 2 tipuri de fișiere:**
  - **fișiere text: fiecare octet este interpretat drept caracter cu codul ASCII dat de octetul respectiv**
    - organizare pe linii (\n are codul ASCII 10)
    - un fișier text în care scriem numărul întreg 123 ocupă trei octeți (codul ASCII pt 1, codul ASCII pt 2, codul ASCII pt 3)
  - **fișiere binare: octeții nu sunt organizați în nici un fel**
    - nu există noțiunea de linie
    - un fișier binar în care scriem numărul întreg 123 ocupă 4 octeți (scrierea binară a lui 123 în baza 2 pentru un int)

# 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 – diferă pentru fișiere text și cele binare) ;
  - 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 **fclose** 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

```
exempluFopen.c x

1 #include<stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     FILE *f;
7     char *numeFisier = "exempluFopen.c";
8     f = fopen(numeFisier,"r");
9     if(f==NULL)
10    {
11        printf("Eroare la deschiderea fisierului");
12        return 0;
13    }
14    else
15        printf("Am deschis fisierul %s in mod text \n",numeFisier);
16    fclose(f);
17    printf("Am inchis fisierul %s deschis in mod text \n",numeFisier);
18
19    f = fopen(numeFisier,"rb");
20    if(f==NULL)
21    {
22        printf("Eroare la deschiderea fisierului");
23        return 0;
24    }
25    else
26        printf("Am deschis fisierul %s in mod binar\n",numeFisier);
27    fclose(f);
28    printf("Am inchis fisierul %s deschis in mod binar\n",numeFisier);
29
30
31    return 0;
32 }
```

# Lucrul cu fișiere

```
exempluFopen.c
```

```
1 #include<stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     FILE *f;
7     char *numeFisier = "exempluFopen.c";
8     f = fopen(numeFisier,"r");
9     if(f==NULL)
10    {
11        printf("Eroare la deschiderea fisierului");
12        return 0;
13    }
14    else
15        printf("Am deschis fisierul %s in mod text \n",numeFisier);
16    fclose(f);
17    printf("Am inchis fisierul %s deschis in mod text \n",numeFisier);
18
19    f = fopen(numeFisier,"rb");
20    if(f==NULL)
21    {
22        printf("Eroare la deschiderea fisierului");
23        return 0;
24    }
25    else
26        printf("Am deschis fisierul %s in mod binar \n");
27    fclose(f);
28    printf("Am inchis fisierul %s deschis in mod binar \n");
29
30
31    return 0;
32 }
```

```
Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ gcc exempluFopen.c
Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ ./a.out
Am deschis fisierul exempluFopen.c in mod text
Am inchis fisierul exempluFopen.c deschis in mod text
Am deschis fisierul exempluFopen.c in mod binar
Am inchis fisierul exempluFopen.c deschis in mod binar
```

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

# Functii pentru lucrul cu fisierele

- se folosesc pentru ambele tipuri de fisiere (text și binare)
- **FILE \*fopen( char \*nume\_fisier, char \*mod)**
  - deschide fisierul cu numele *nume\_fisier* pentru acces de tip *mod*
- **int fclose(FILE \* f);**
  - închide fisierul asociat cu variabile f și eliberează bufferul;
- **int feof(FILE \*f);**
  - returnează 0 dacă nu s-a detectat sfârșit de fisier (EOF) la ultima operație de citire, altfel o valoare nenulă pentru EOF;
- **FILE \* freopen(char \*nume\_fisier, char\* mod, FILE\* f)**
  - se închide fisierul f, se deschide fisierul *nume\_fisier* în modul *mod* și se asociază cu f (de obicei f este stdin sau stdout);
- **int fflush(FILE\* f)**
  - golește bufferul asociat unui fisier; pentru fisierele deschise pentru scriere are ca efect scrierea în fisier a datelor din buffer care încă nu au fost puse în fisier

# Functii pentru lucrul cu fisierele

- **funcții de poziționare într-un fișier**
- În C ne putem poziționa pe un anumit octet din fișier.  
Functiile care permit poziționarea (cele mai importante) sunt:
- **long 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 intr-un fisier

## Exemplu: aflarea dimensiunii unui fisier

exempluCalculeazaDimensiune.c

```
1 #include<stdio.h>
2 #include<stdlib.h>
3
4 int main()
5 {
6
7     FILE *f;
8     int numarLinii = 0;
9     f = fopen("exempluCalculeazaDimensiune.c","r");
10
11    if (f == NULL)
12    {
13        printf("Eroare la deschiderea fisierului");
14        exit(0);
15    }
16
17    fseek(f,0,SEEK_END);
18    long int nbBytes = ftell(f);
19    printf("Fisierul are %ld octeti\n",nbBytes);
20
21    fclose(f);
22    return 0;
23
24 }
```

```
Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ ./a.out
Fisierul are 363 octeti
```

exempluCalculeazaDimensiune.c

```
1 #include<stdio.h>
2 #include<stdlib.h>
3
4 int main()
5 {
6
7     FILE *f;
8     int numarLinii = 0;
9     f = fopen("exempluCalculeazaDimensiune.c","rb");
10
11    if (f == NULL)
12    {
13        printf("Eroare la deschiderea fisierului");
14        exit(0);
15    }
16
17    fseek(f,0,SEEK_END);
18    long int nbBytes = ftell(f);
19    printf("Fisierul are %ld octeti\n",nbBytes);
20
21    fclose(f);
22    return 0;
23
24 }
```

```
Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ ./a.out
Fisierul are 364 octeti
```

# Functii de pozitionare intr-un fisier

## Exemplu: aflarea dimensiunii unui fisier

```
exempluCalculeazaDimensiune.c  x

1 #include<stdio.h>
2 #include<stdlib.h>
3
4 int main()
5 {
6
7     FILE *f;
8     int numarLinii = 0;
9     f = fopen("exempluCalculeazaDimensiune.c","r");
10
11    if (f == NULL)
12    {
13        printf("Eroare la deschiderea fisierului");
14        exit(0);
15    }
16
17    fseek(f,0,SEEK_END);
18    long int nbBytes = ftell(f);
19    printf("Fisierul are %ld octeti\n",nbBytes);
20
21    fclose(f);
22    return 0;
23
24 }
```

Bogdan-Alexes-MacBook-Pro:curs6 bogdan\$ ./a.out  
Fisierul are 363 octeti

```
exempluCalculeazaDimensiune.c  x

1 #include<stdio.h>
2 #include<stdlib.h>
3
4 int main()
5 {
6
7     FILE *f;
8     int numarLinii = 0;
9     f = fopen("exempluCalculeazaDimensiune.c","rb");
10
11    if (f == NULL)
12    {
13        printf("Eroare la deschiderea fisierului");
14        exit(0);
15    }
16
17    fseek(f,0,SEEK_END);
18    long int nbBytes = ftell(f);
19    printf("Fisierul are %ld octeti\n",nbBytes);
20
21    fclose(f);
22    return 0;
23
24 }
```

Bogdan-Alexes-MacBook-Pro:curs6 bogdan\$ ./a.out  
Fisierul are 364 octeti

# Functii pentru lucrul cu fisierele

- **alte funcții de poziționare într-un fișier**
  
- **int fgetpos(FILE \*f, const fpos\_t \*ptr)**
  - memorează poziția curentă în variabila ptr în cadrul fișierului asociat cu f (ptr va fi folosit ulterior cu funcția fsetpos);
  
- **int fsetpos (FILE \*f, const fpos\_t \*ptr)**
  - setează poziția curentă în fișierul asociat cu f la valoarea ptr, obținută anterior prin funcția fgetpos

# Functii de pozitionare într-un fisier

## Exemplu:

```
1 #include <stdio.h>
2
3 int main ()
4 {
5     FILE *f;
6     int c, i;
7     fpos_t pos;
8
9     f = fopen("ABC.txt","r");//ABC.txt contine caracterele A,B,C
10    if (f == NULL)
11    {
12        printf("Eroare la deschiderea fisierului");
13        return 0;
14    }
15    c = fgetc(f);
16    printf("Primul caracter este %c\n",c);
17    fgetpos(f,&pos);
18    for (i=0;i<3;i++)
19    {
20        fsetpos(f,&pos);
21        c = fgetc(f);
22        printf ("Al doilea caracter este %c\n",c);
23        c = fgetc(f);
24        printf ("Al treilea caracter este %c\n",c);
25    }
26    fclose(f);
27    return 0;
28 }
```

# Functii de pozitionare într-un fisier

## Exemplu:

```
1 #include <stdio.h>
2
3 int main ()
4 {
5     FILE *f;
6     int c, i;
7     fpos_t pos;
8
9     f = fopen("ABC.txt","r");//ABC.txt contine caracterele A,B,C
10    if (f == NULL)
11    {
12        printf("Eroare la deschidere\n");
13        return 0;
14    }
15    c = fgetc(f);
16    printf("Primul caracter este %c\n",c);
17    fgetpos(f,&pos);
18    for (i=0;i<3;i++)
19    {
20        fsetpos(f,&pos);
21        c = fgetc(f);
22        printf ("Al doilea caracter este %c\n",c);
23        c = fgetc(f);
24        printf ("Al treilea caracter este %c\n",c);
25    }
26    fclose(f);
27    return 0;
28 }
```

# Alte funcții pentru lucrul cu fișierele

- **void rewind (FILE \*f)**

- repoziționarea 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

# Cuprinsul cursului de azi

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

# Fișiere text

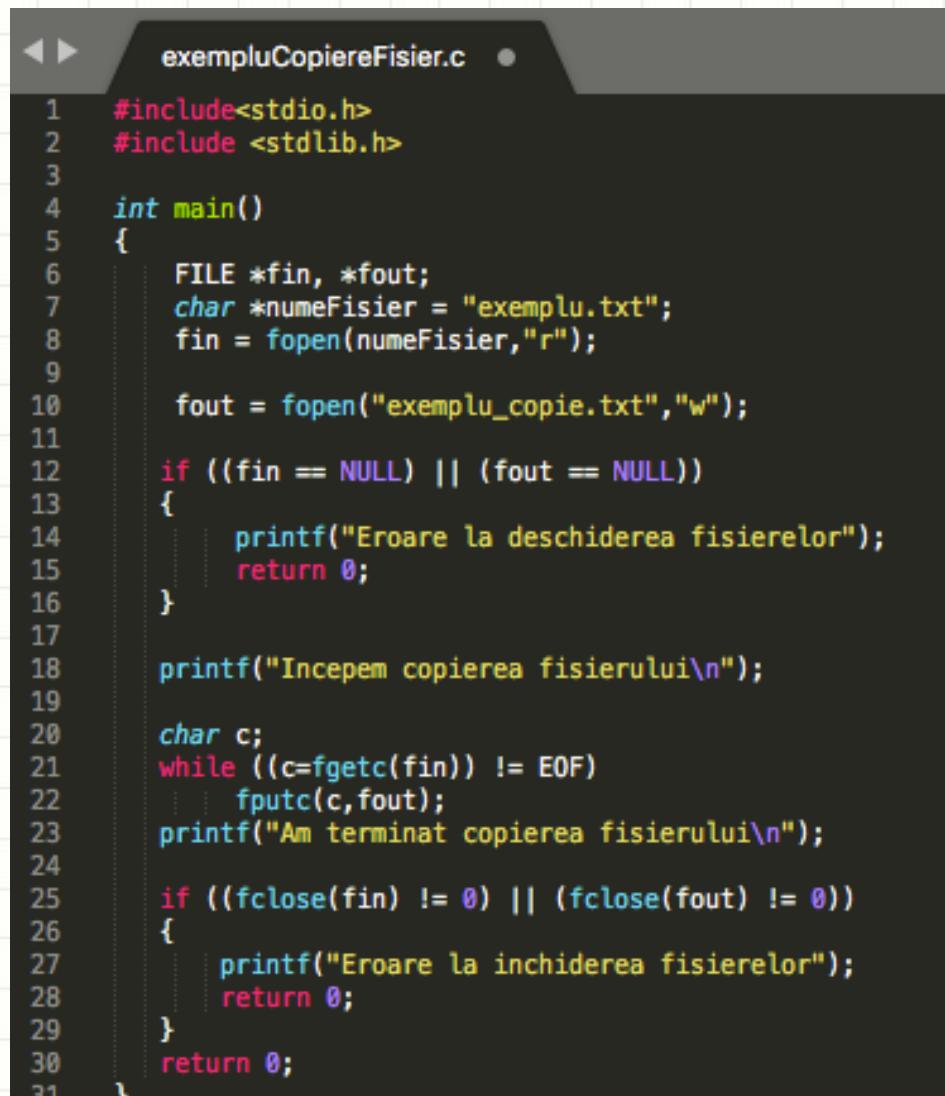
- accesul la fișierele text se poate face la nivel de **șir de caractere** (linie) sau la nivel de **caracter** (octet).
- un fișier text se accesează ca o succesiune de linii de text de lungime variabilă (încheiate cu un terminator de linie : '\n') utilizând un set dedicat de funcții din biblioteca 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 fișier text



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

int main()
{
    FILE *fin, *fout;
    char *numeFisier = "exemplu.txt";
    fin = fopen(numeFisier,"r");

    fout = fopen("exemplu_copie.txt","w");

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

    printf("Incepem copierea fisierului\n");

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

    if ((fclose(fin) != 0) || (fclose(fout) != 0))
    {
        printf("Eroare la inchiderea fisierelor");
        return 0;
    }
    return 0;
}
```

# Functii de citire scriere la nivel de caracter

### □ Exemplu: copierea unui fișier text

```
1 #include<stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     FILE *fin, *fout;
7     char *numeFisier = "exemplu.txt";
8     fin = fopen(numeFisier,"r");
9
10    fout = fopen("exemplu_copie.txt","w");
11
12    if ((fin == NULL) || (fout == NULL))
13    {
14        printf("Eroare la deschiderea fisierelor");
15        return 0;
16    }
17
18    printf("Incepem copierea fisierului\n");
19
20    t
```

```
Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ gcc exempluCopiereFisier.c -o copiazaFisiere
```

[Bogdan-Alexes-MacBook-Pro:curs6 bogdans\$ ./copiaazaFisiere

Incepem copierea fisierului

Am terminat copierea fisierului

```
25     if ((fclose(fin) != 0) || (fclose(fout) != 0))
```

	copiaazaFisiere	Today, 23:26	9 KB	Unix executable
	exemplu_copie.txt	Today, 23:27	5 bytes	Plain Text
	exemplu.txt	Today, 23:26	5 bytes	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

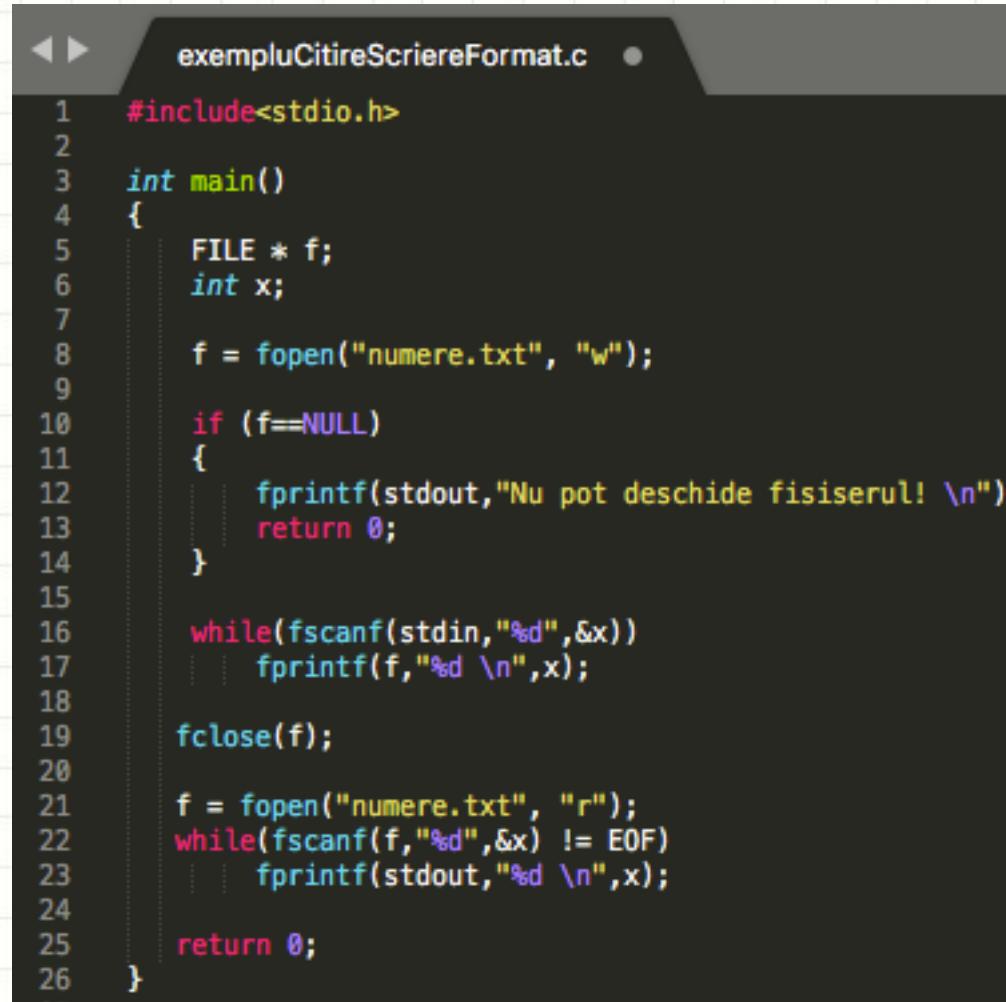
```
1  exempluCitireLinii.c  x
2
3
4  #include<stdio.h>
5  #include<stdlib.h>
6
7  int main()
8  {
9
10     FILE *f;
11     char linie[1000];
12     int numarLinii = 0;
13     f = fopen("test.txt","r");
14
15     if (f == NULL)
16     {
17         printf("Eroare la deschiderea fisierului");
18         exit(0);
19     }
20
21     while (fgets(linie,1000,f) != NULL)
22         numarLinii++;
23
24     printf("Fisierul are %d linii\n",numarLinii);
25
26 }
```

```
[Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ gcc exempluCitireLinii.c
[Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ ./a.out
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



The image shows a code editor window with a dark theme. The title bar says "exempluCitireScriereFormat.c". The code is written in C and performs the following tasks:

- Includes the stdio.h header.
- Defines a main function.
- Creates a FILE pointer f and an integer x.
- Opens a file named "numere.txt" in write mode ("w").
- If the file cannot be opened, it prints an error message and returns 0.
- Enters a loop to read integers from standard input (stdin) using fscanf and write them to the file f using fprintf.
- Closes the file f.
- Opens the same file "numere.txt" in read mode ("r").
- Enters a loop to read integers from the file f using fscanf and print them to standard output (stdout) using fprintf.
- Returns 0 at the end of the program.

```
#include<stdio.h>
int main()
{
    FILE * f;
    int x;

    f = fopen("numere.txt", "w");
    if (f==NULL)
    {
        fprintf(stderr,"Nu pot deschide fisierul! \n");
        return 0;
    }

    while(fscanf(stdin,"%d",&x))
        fprintf(f,"%d \n",x);

    fclose(f);

    f = fopen("numere.txt", "r");
    while(fscanf(f,"%d",&x) != EOF)
        fprintf(stdout,"%d \n",x);

    return 0;
}
```

# Functii de citire scriere cu format

```
1 #include<stdio.h>
2
3 int main()
4 {
5     FILE * f;
6     int x;
7
8     f = fopen("numere.txt", "w");
9
10    if (f==NULL)
11    {
12        fprintf(stdout,"Nu pot deschide fisierul! \n");
13        return 0;
14    }
15
16    while(fscanf(stdin,"%d",&x))
17        fprintf(f,"%d \n",x);
18
19    fclose(f);
20
21 Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ gcc exempluCitireScriereFormat.c
22 Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ ./a.out
23
24 23
25 -1000
26 0
27 11
28 y
29 23
30 -1000
31 0
32 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
```

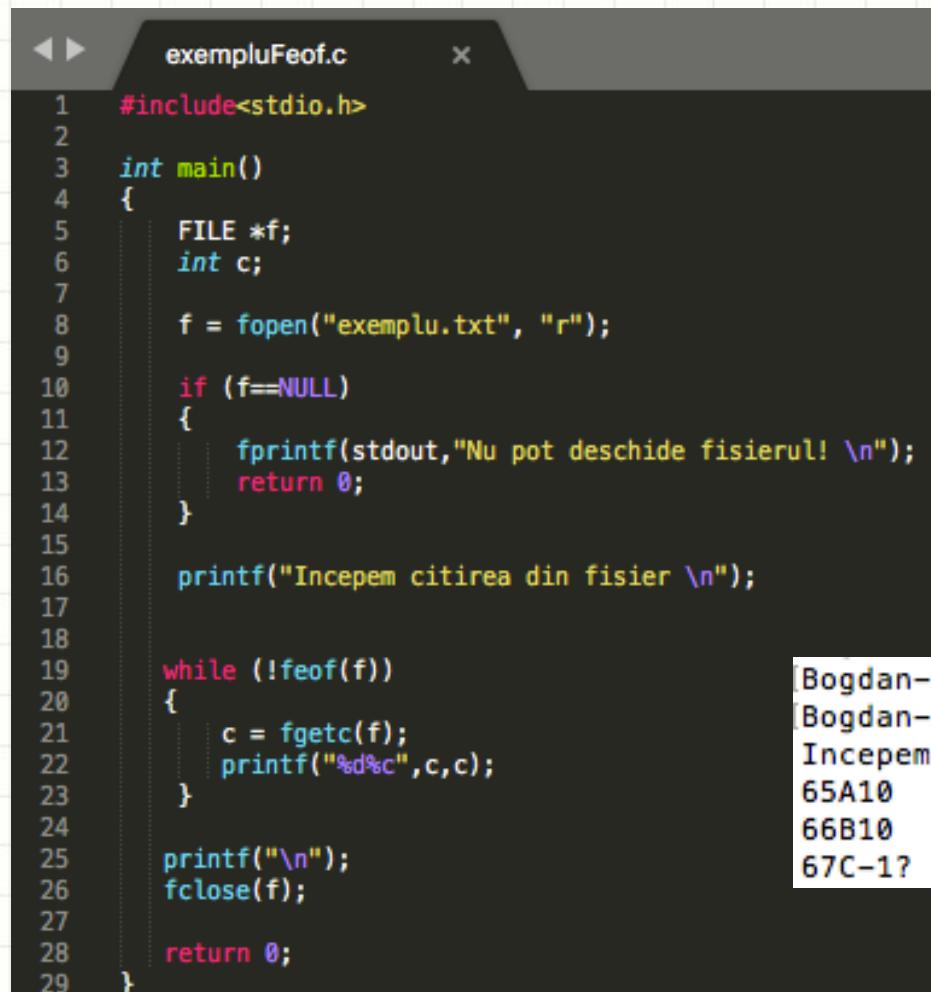
se scrie un x (= -1) în plus

**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) :



```
#include<stdio.h>
int main()
{
    FILE *f;
    int c;

    f = fopen("exemplu.txt", "r");

    if (f==NULL)
    {
        fprintf(stdout,"Nu pot deschide fisierul! \n");
        return 0;
    }

    printf("Incepem citirea din fisier \n");

    while (!feof(f))
    {
        c = fgetc(f);
        printf("%d%c",c,c);
    }

    printf("\n");
    fclose(f);

    return 0;
}
```



```
Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ gcc exempluFeof.c
Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ ./a.out
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) :

```
exempluFeof.c
1 #include<stdio.h>
2
3 int main()
4 {
5     FILE *f;
6     int c;
7
8     f = fopen("exemplu.txt", "r");
9
10    if (f==NULL)
11    {
12        fprintf(stdout,"Nu pot deschide fisierul! \n");
13        return 0;
14    }
15
16    printf("Incepem citirea din fisier \n");
17
18
19    while (1)
20    {
21        c = fgetc(f);
22        if(feof(f))
23            break;
24        printf("%d%c",c,c);
25    }
26
27    printf("\n");
28    fclose(f);
29
30
31 }
```



```
Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ gcc exempluFeof.c
Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ ./a.out
Incepem citirea din fisier
65A10
66B10
67C
```

# Cuprinsul cursului de azi

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

# Fișiere binare

- **fișier binar = sir de octeți neformatat pe linii care este stocat pe suport magnetic/optic.**
- 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.

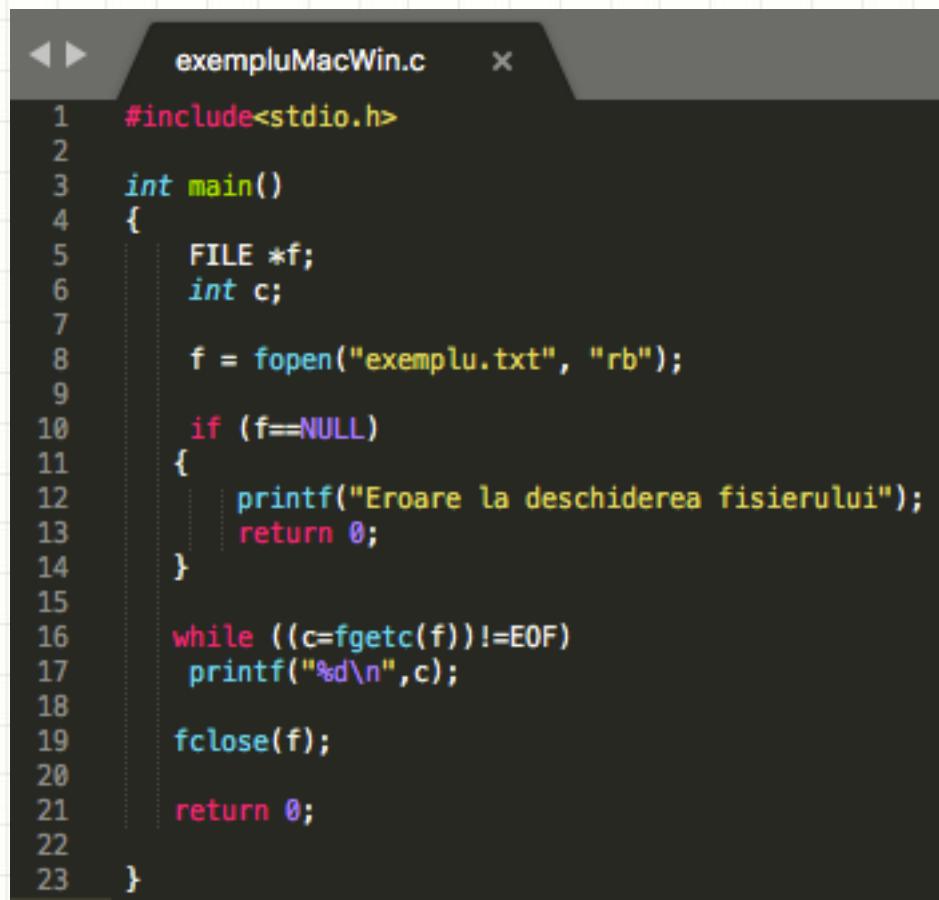
# Fisiere binare

- **fișier binar = sir de octeți neformatat pe linii care este stocat pe suport magnetic/optic.**
  
- **FILE \*fopen( char \*nume\_fisier, char \*mod\_deschidere)**
  - nume\_fisier = numele fisierului
  - mod\_deschidere = sir de caracter ce precizeaza tipul de acces la fisier:

<b>Mod</b>	<b>Semnificație</b>
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.



```
exempluMacWin.c

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

# Fișiere binare

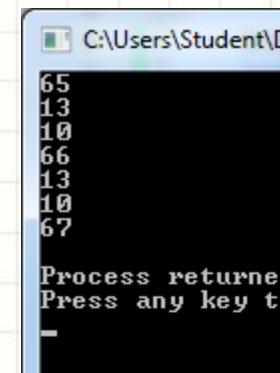
- fișier binar = sir de octeți neformatat pe linii care este stocat pe suport magnetic/optic. Octeții nu sunt considerați ca fiind coduri de caractere.



output pe MAC

```
Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ gcc exempluMacWin.c
[Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ ./a.out
65
10
66
10
67
```

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

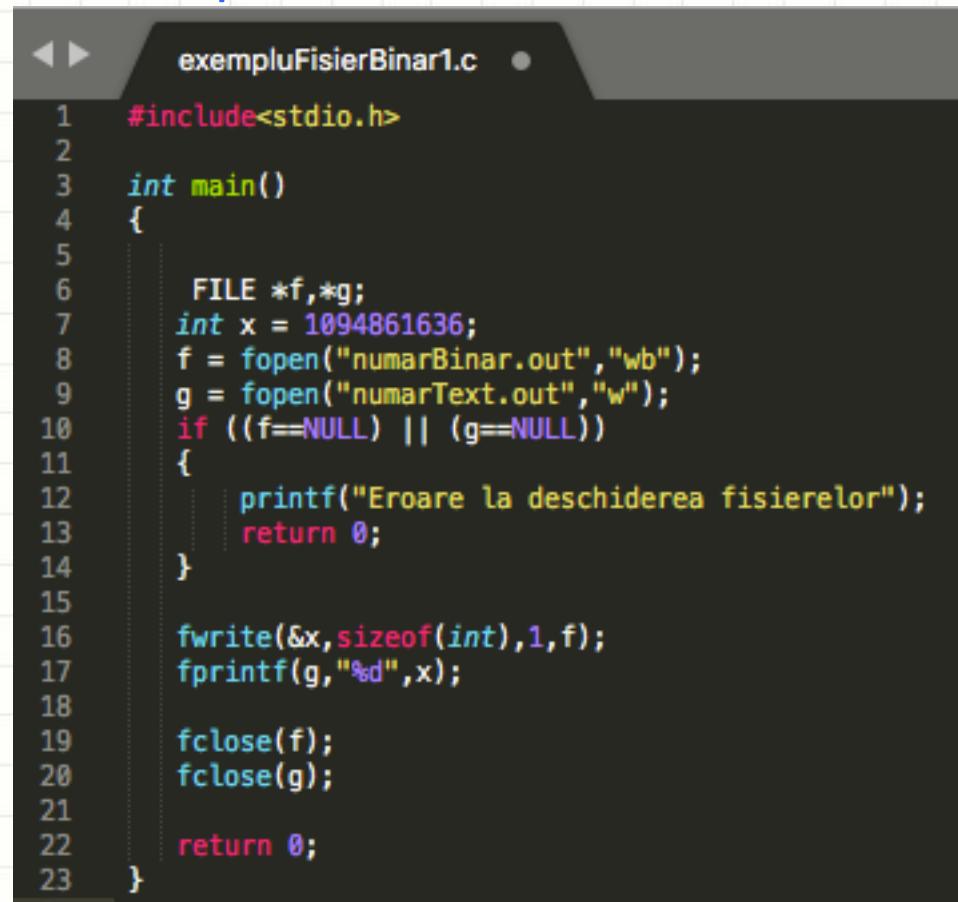


# Functii de citire scriere

- `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



The image shows a screenshot of a code editor window titled "exempluFisierBinar1.c". The code is written in C and demonstrates how to read an integer from a binary file and write it to a text file.

```
#include<stdio.h>
int main()
{
    FILE *f,*g;
    int x = 1094861636;
    f = fopen("numarBinar.out","wb");
    g = fopen("numarText.out","w");
    if ((f==NULL) || (g==NULL))
    {
        printf("Eroare la deschiderea fisierelor");
        return 0;
    }
    fwrite(&x,sizeof(int),1,f);
    fprintf(g,"%d",x);
    fclose(f);
    fclose(g);
    return 0;
}
```

# Functii de citire scriere

## Exemplul 1

```
< > exempluFisierBinar1.c ●

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

Scrierea lui 1094861636 in baza 2:



Octetii se reprezinta de la cel mai putin semnificativ la cel mai semnificativ pe masina mea (little endian)

# Functii de citire scriere

## □ Exemplul 2

```
exempluFisierBinar2.c  x

001 #include<stdio.h>
002
003 int main()
004 {
005
006     FILE *f;
007     int x;
008     f = fopen("numarBinar.out","rb");
009     if (f==NULL)
010     {
011         printf("Eroare la deschiderea fisierului");
012         return 0;
013     }
014     while(fread(&x,sizeof(int),1,f)==1)
015         printf("x=%d\n",x);
016     fclose(f);
017
018     char c;
019     f = fopen("numarBinar.out","rb");
020     while(fread(&c,sizeof(char),1,f)==1)
021         printf("c=%d\n",c);
022     fclose(f);
023
024     return 0;
025
026 }
```

```
Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ gcc exempluFisierBinar2.c
Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ ./a.out
x=1094861636
c=68
c=67
c=66
c=65
```

# Functii de citire scriere

## Exemplul 3

```
exempluFisierBinar3.c

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



Coduri ASCII:

33 = !, 40 = (, 50 = 2, 10 = \n, 80 = P

# Functii de citire scriere

## Exemplul 4

```
exempluFisierBinar4.c

1 #include<stdio.h>
2
3 int main()
4 {
5
6     FILE *f1,*f2,*f3;
7     f1 = fopen("numere1.txt","rb");
8     f2 = fopen("numere2.txt","rb");
9     f3 = fopen("numere3.txt","rb");
10    if ((f1==NULL) || (f2==NULL) || (f3==NULL))
11    {
12        printf("Eroare la deschiderea fisierelor");
13        return 0;
14    }
15    int v[5],i;
16    for(i=0;i<5;i++)
17    {
18        fread(&v[i],sizeof(int),1,f2); printf("%d ",v[i]);
19    }
20    printf("\n");
21    //varianata 2
22    fread(v,sizeof(int),5,f3);
23    for(i=0;i<5;i++)
24        printf("%d ",v[i]);
25    printf("\n");
26    //varianata 3
27    fread(v,5*sizeof(int),1,f1);
28    for(i=0;i<5;i++)
29        printf("%d ",v[i]);
30    printf("\n");
31
32    fclose(f1);
33    fclose(f2);
34    fclose(f3);
35    return 0;
36 }
```

```
[Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ gcc exempluFisierBinar4.c
[Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ ./a.out
33 40 50 10 80
33 40 50 10 80
33 40 50 10 80
```

# Functii de citire scriere

## Exemplul 5: scrierea unei structuri

```
exempluFisierBinar5.c  x

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

# Functii de citire scriere

## Exemplul 5: scrierea unei structuri

The screenshot shows a terminal window with the following output:

```
Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ gcc exempluFisierBinar5.c
Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ ./a.out
Nume student = Popescu
Prenume student = Maria
Varsta student = 20
```

Below the terminal window, there is a file browser interface showing a file named "date.in" containing the text "PopescuMaria".

Characterul cu codul ASCII = 20 e neprintabil

The screenshot shows a terminal window with the following output:

```
Bogdan-Alexes-MacBook-Pro:curs6 bogdan$ ./a.out
Nume student = Popescu
Prenume student = Maria
Varsta student = 45
```

Below the terminal window, there is a file browser interface showing a file named "date.in" containing the text "PopescuMaria-".

# Aplicație: procesarea de imagini

- ❑ o imagine este un fișier binar



# Aplicație: procesarea de imagini

- ❑ o imagine este un fișier binar
- ❑ imagine color RGB
- ❑ fiecare pixel are o culoare dată de un triplet (r,g,b) cu valori pe fiecare canal intre 0 si 255 = 1 octet/canal
- ❑ albastru = (0,0 ,255), negru = (0,0,0),  
rosu = (255, 0, 0), verde = (0,255,0)
- ❑ 1 pixel color = 3 canale = 3 octeți
- ❑ dimensiuni 600 x 800 pixeli
- ❑  $600 \times 800 \times 3$  octeți = 1440000 octeți



# Aplicatie: procesarea de imagini

image.bmp Info

**image.bmp** 1.4 MB

Modified: Today, 00:14

Add Tags...

**General:**

Kind: Windows bitmap image  
Size: 1'440'054 bytes (1.4 MB on disk)  
Where: Macintosh HD ▶ Users ▶ bogdan ▶ FMI ▶ PP ▶ Bogdan ▶ 2017\_0212

Created: Today, 00:14  
Modified: Today, 00:14

Stationery pad  
 Locked

**More Info:**

Where from: <http://cdn.instructables.com/ORIG/FWO/36LM/FLQAM1CS/FWO36LMFLQAM1CS.bmp>, <https://www.google.com/>

Dimensions: 800 × 600  
Color space: RGB  
Alpha channel: No

**Name & Extension:**

image.bmp

Hide extension

Search



**image.bmp**

1.4 MB

Created Today, 00:14  
Modified Today, 00:14  
Last opened Today, 00:14  
Dimensions 800 × 600

[Add Tags...](#)

# Aplicație: procesarea de imagini

- ❑ o imagine este un fișier binar
- ❑ imagine color RGB
- ❑ fiecare pixel are o culoare dată de un triplet (r,g,b) cu valori pe fiecare canal între 0 și 255 = 1 octet/canal
- ❑ 1 pixel color = 3 canale = 3 octeți
- ❑ dimensiuni 600 x 800 pixeli
- ❑  $600 \times 800 \times 3$  octeți = 1440000 octeți
- ❑ 54 de octeți headerul + 144000 octeți pentru pixeli



modificaImagine.c x

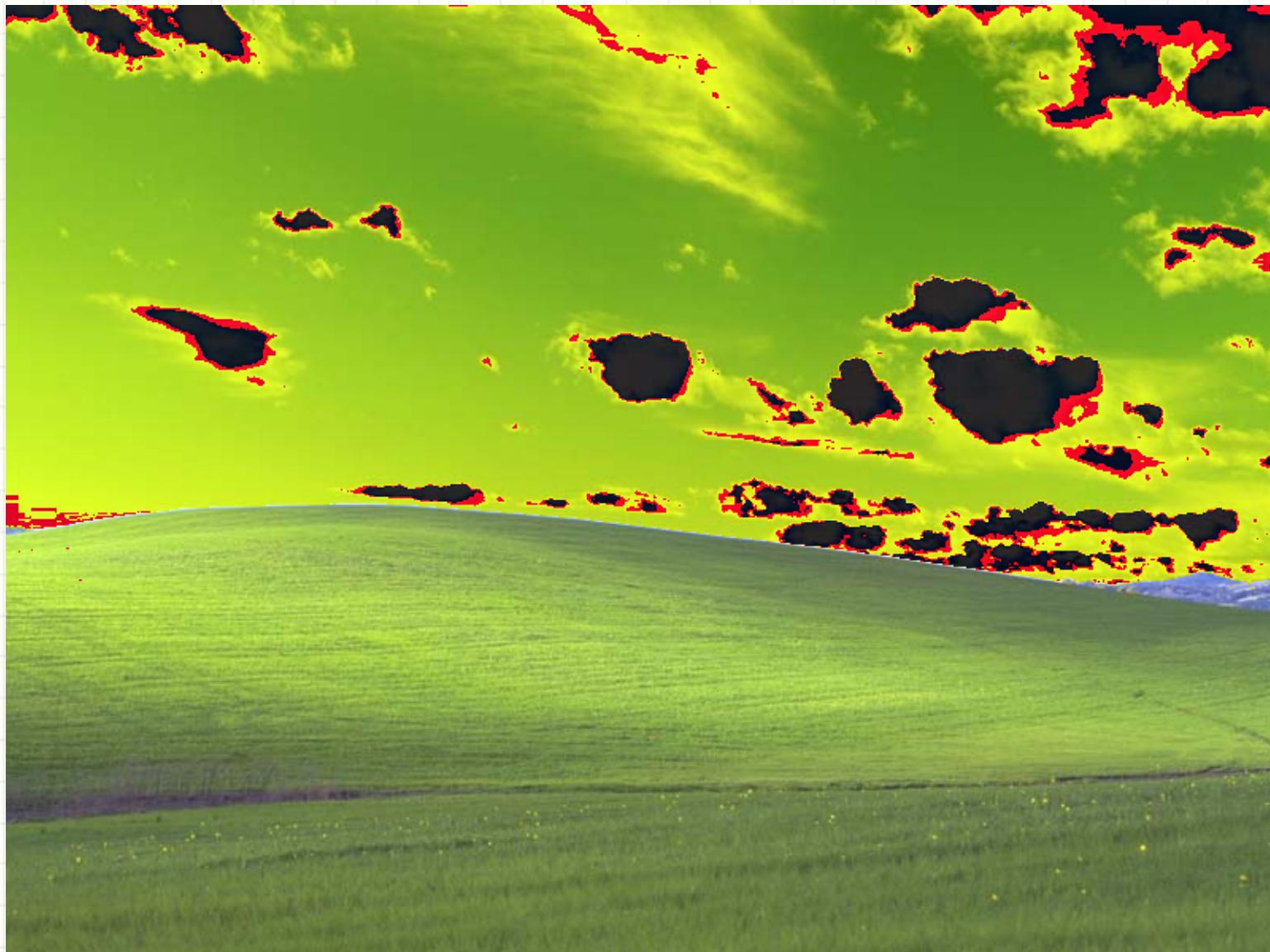
```
1 #include <stdio.h>
2
3 int main()
4 {
5     //pointeri de fisier
6     FILE *fin , *fout;
7
8     //deschid fisierul de intrare in mod binar (pentru citire)
9     fin = fopen("image.bmp" , "rb");
10
11    //deschid fisierul de iesire in mod binar (pentru scriere)
12    fout = fopen("image_modified.bmp" , "wb");
13
14    //copiez headerul
15    int i, x, y;
16    for(i = 0; i < 54; i++)
17    {
18        fread(&x , 1 , 1 , fin);
19        fwrite(&x , 1 , 1 , fout);
20    }
21
22    //citesc octet cu octet din fisierul de intrare
23    while(fread(&x , 1 , 1 , fin) == 1)
24    {
25        y = x + 50;
26        fwrite(&y , 1 , 1 , fout);
27    }
28
29    fclose(fin);
30    fclose(fout);
31
32    return 0;
33 }
```

Adaug 50 de unități la fiecare canal,

Schimb culorile în imagini

Cresc luminozitatea (brightness)

Ce obțin?



- ❑ pixeli negri = (0,0,0)
- ❑ de ce apar pixeli negri?
- ❑  $y = x + 50;$
- ❑ daca  $x > 205$  atunci octetul scris din y va retine o valoare mică (apropiată de 0)



```
modificalmagine.c  x
1 #include <stdio.h>
2
3 int main()
4 {
5     //pointeri de fisier
6     FILE *fin , *fout;
7
8     //deschid fisierul de intrare in mod binar (pentru citire)
9     fin = fopen("image.bmp" , "rb");
10
11    //deschid fisierul de iesire in mod binar (pentru scriere)
12    fout = fopen("image_modified.bmp" , "wb");
13
14    //copiez headerul
15    int i, x, y;
16    for(i = 0; i < 54; i++)
17    {
18        fread(&x , 1 , 1 , fin);
19        fwrite(&x , 1 , 1 , fout);
20    }
21
22    //citesc octet cu octet din fisierul de intrare
23    while(fread(&x , 1 , 1 , fin) == 1)
24    {
25        y = x + 50;
26        if (y > 255)
27            y = 255;
28        fwrite(&y , 1 , 1 , fout);
29    }
30
31    fclose(fin);
32    fclose(fout);
```

Adaug 50 de unități la fiecare canal,

Schimb culorile în imagini

Cresc luminozitatea (brightness)

Ce obțin?





Imagine inițială



Imagine modificată

# Cuprinsul cursului de azi

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

# Functii

- permit modularizarea programelor
  - variabilele declarate în interiorul funcțiilor – variabile locale (vizibile doar în interior)
- parametri funcțiilor
  - permit comunicarea informației între funcții
  - sunt variabile locale funcțiilor
- avantajele utilizării funcțiilor
  - divizarea problemei în subprobleme
  - managementul dezvoltării programelor
  - utilizarea/reutilizarea funcțiilor scrise în alte programe
  - elimină duplicarea codului scris

# Functii

- o funcție = bloc de instrucțiuni care nu se poate executa de sine stătător ci trebuie apelat.

- sintaxa:

**tip\_returnat nume\_functie (lista parametrilor formali)**

```
{     variabile locale  
       instructiuni;  
       return expresie;  
}
```



antetul funcției  
(declarare)

corpus funcției  
(definire)

- lista de parametri formalii poate fi reprezentata de:
  - nici un parametru:
    - **tip\_returnat nume\_functie ()**
    - **tip\_returnat nume\_functie (void)**
  - unul sau mai mulți parametri separați prin virgulă.

# Valoarea returnată de o funcție

- două categorii de funcții:
  - care returnează o valoare: prin utilizarea instrucțiunii **return expresie**;
  - care nu returnează o valoare: prin instrucțiunea **return**; (tipul returnat este void)
- returnarea valorii
  - poate returna orice tip standard (**void**, **char**, **int**, **float**, **double**) sau definit de utilizator (structuri, uniuni, enumerari)
  - declarațiile și instrucțiunile din funcții sunt executate până se întâlnește
    - instrucțiunea **return**
    - accolada închisă **}** - execuția atinge finalul funcției

# Valoarea returnată de o funcție

```
double f(double t)
{
    return t-1.5;
}
```

← definire de funcție

```
float g(int);
```

← declarație de funcție

```
int main()
{
    float a=11.5;
    printf("%f\n", f(a));
    printf("%f\n", g(a));
}
```

**Rezultat afișat**

10.000000

13.000000

```
float g(int z)
{
    return z+2.0;
}
```

← definire de funcție