

PROGRAMAREA CALCULATOARELOR

Cursul 6

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, șiruri 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)

- Declarare și definire. Apel. Metode de transmitere a paramerilor. Pointeri la funcții.

❑ Tablouri și pointeri

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

❑ Șiruri 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.

CUPRINSUL CURSULUI DE AZI

1. Fișiere: noțiuni generale
2. Fișiere text: funcții specifice de manipulare.

CUPRINSUL CURSULUI DE AZI

1. Fișiere: noțiuni generale
2. Fișiere text: funcții specifice de manipulare.

FIȘIERE

- ❑ **fișier = șir 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 limbaje de programare fac același lucru;
- ❑ noțiunea de fișier este mai generală
- ❑ fișier = flux de date (stream) = transfer de informație binară (șir 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 care se deschid automat in 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

❑ 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

- ❑ **fișiere text: octeții (caractere ASCII) sunt organizați pe linii. Caracterele terminatorii de linii sunt:**
 - ❑ 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.

LUCRUL CU FIȘIERE

- în biblioteca stdio.h este definită o structură **FILE**. Exemple:

```
typedef struct _iobuf
{
    char* _ptr;
    int _cnt;
    char* _base;
    int _flag;
    int _file;
    int _charbuf;
    int _bufsiz;
    char* _tmpfname;
} FILE;
```

```
typedef struct {
    int level; /* fill/empty level of buffer */
    unsigned flags; /* File status flags */
    char fd; /* File descriptor */
    unsigned char hold; /* Ungetc char if no buffer */
    int bsize; /* Buffer size */
    unsigned char *buffer; /* Data transfer buffer */
    unsigned char *curp; /* Current active pointer */
    unsigned istemp; /* Temporary file indicator */
    short token; /* Used for validity checking */
} FILE;
```

LUCRUL CU FIȘIERE

- ❑ declararea unui **pointer la structura FILE** = realizarea legăturii dintre nivelele logic (variabila fișier) și fizic (numele extern al fișierului) :

FILE * f;

Cererea de deschidere a unui fisier:

- ❑ Fisierul a putut fi deschis:
 - ❑ Pointer-ul la **FILE** nu este **NULL**;
 - ❑ Se prelucreaza fisierul si se inchide;
- ❑ Fisierul nu a putut fi deschis:
 - ❑ Pointer-ul la **FILE** este **NULL**;
 - ❑ Nu se poate continua cu prelucrarea;
 - ❑ Nu este necesara inchiderea ;

Exemplu FILE*

Untitled20.c

```
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <conio.h>
#define CLOSE_FILE 1
void verify_file(FILE* f)
{
    if (f != NULL)
    {
        printf("%p\n", f);
        if (CLOSE_FILE)
            fclose(f);
    }
    else
    {
        printf("Nu am putut deschide fisierul! Eroare: %s\n", strerror(errno));
    }
}

int main()
{
    FILE *f1, *f2;
    f1 = fopen("Untitled20.c", "r");
    verify_file(f1);
    f1 = fopen("Untitled30.c", "r");
    verify_file(f1);
    f2 = fopen("Untitled10.c", "r");
    verify_file(f2);

    printf("Nr maxim fisiere deschise: %d\n", FOPEN_MAX);

    getch();
}
```

C:\Users\traian\Documents\Untitled20.exe

```
76492960
Nu am putut deschide fisierul! Eroare: No such file or directory
76492960
Nr maxim fisiere deschise: 20
```

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_acces)
- ❑ nume_fisier = numele fisierului
- ❑ mod_acces= șir de 1-3 caractere ce indica tipul de acces :
 - ❑ citire "r", scriere "w", adăugare la sfârșitul fișierului "a";
 - ❑ "+" permite scrierea și citirea "r+", "w+", "a+";
 - ❑ t (implicit) sau b: specifică tipul de fișier (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

Moduri de prelucrare a fișierelor text

Mod	Descriere
r	Deschiderea fișierului pentru citire. Fișierul trebuie să existe!
w	Crearea unui fișier pentru scriere. Dacă fișierul există, conținutul acestuia este șters în totalitate!
a	Deschiderea sau crearea (dacă nu există) unui fișier pentru adăugarea de conținut numai la sfârșitul acestuia
r+	Deschiderea unui fișier pentru actualizarea conținutului (citire și scriere). Fișierul trebuie să existe!
w+	Deschiderea unui fișier pentru actualizarea conținutului (citire și scriere). Dacă fișierul există, conținutul acestuia este șters în totalitate!
a+	Deschiderea unui fișier pentru citirea conținutului și adăugarea de conținut numai la sfârșitul acestuia. Dacă fișierul nu există, acesta este creat.

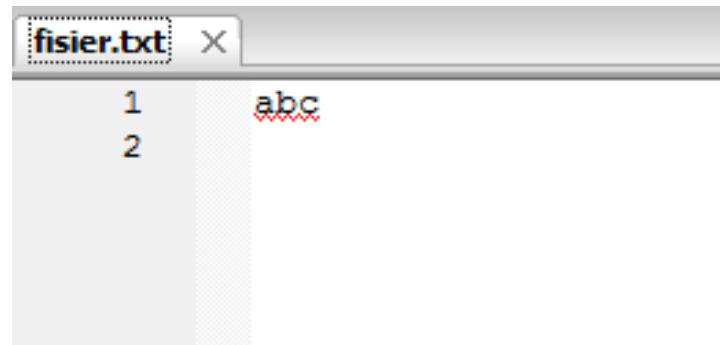
```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    FILE *g = fopen ("fisier.txt", "r+");
    char sir[5];

    if (g == NULL)
        printf("eroare!");

    fgets(sir, 5, g);
    printf("%s", sir);

    return 0;
}
```



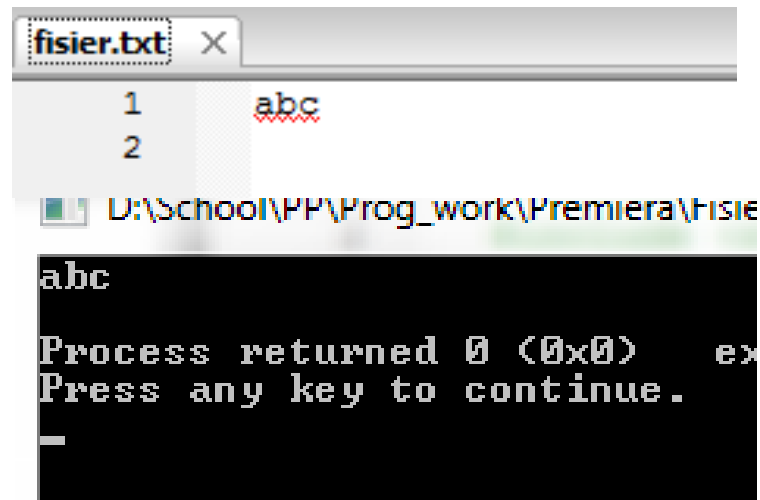
```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    FILE *g = fopen ("fisier.txt", "r+");
    char sir[5];

    if (g == NULL)
        printf("eroare!");

    fgets(sir, 5, g);
    printf("%s", sir);

    return 0;
}
```



The screenshot shows a Windows file explorer window titled 'fisier.txt' with a list of two lines: '1 abc' and '2'. Below the file explorer is a console window with a black background. The console displays the text 'abc' on the first line, followed by 'Process returned 0 (0x0) ex' and 'Press any key to continue.' on the next line. A cursor is visible on the line 'Press any key to continue.'

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

```
int main()
```

```
{
```

```
FILE *g = fopen ("fisier.txt", "r+");
```

```
char c;
```

```
if (g == NULL)
```

```
    printf("\n Nu s-a putut deschide! \n");
```

```
c = fgetc(g);
```

```
printf("%c", c);
```

```
fputc('z', g);
```

```
fflush(g);
```

```
if (fclose(g) != 0)
```

```
    printf("\n Probleme la inchiderea fisierelor!\n");
```

```
return 0;
```

```
}
```

fisier.txt X

1	abc
2	


```

#include <stdlib.h>

int main()
{
    FILE *g = fopen ("fisier.txt", "r+");
    char c;

    if (g == NULL)
        printf("\n Nu s-a putut deschide! \n");

    c = fgetc(g);
    printf("%c", c);

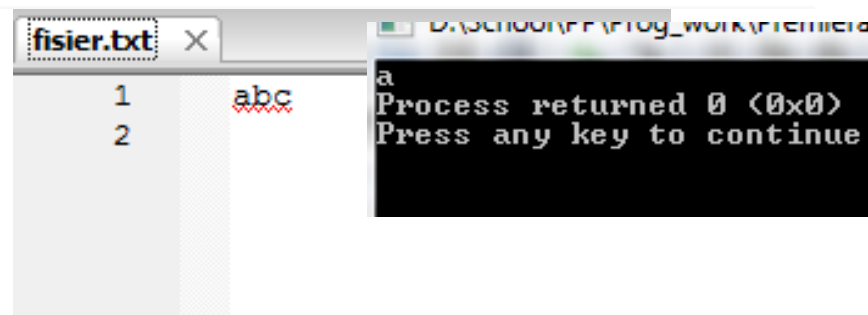
    fputc('z', g);
    fflush(g);

    if (fclose(g) != 0)
        printf("\n Probleme la inchiderea fisierelor!\n");

    return 0;
}

```

Rezultat: fisierul
ramane neschimbat?!



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

int main()
{
    FILE *g = fopen ("fisier.txt","r+");
    char c;

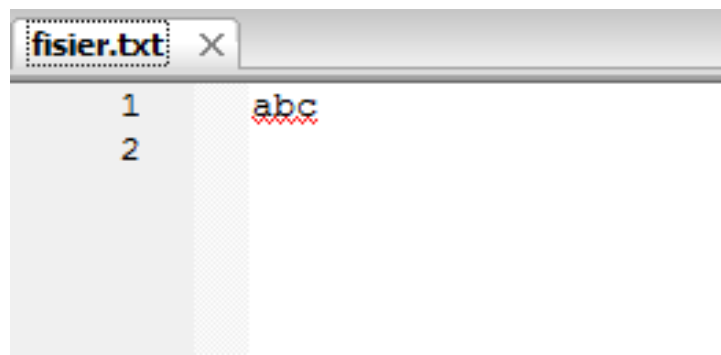
    if (g == NULL)
        printf("\n Nu s-a putut deschide! \n");

    c = fgetc(g);
    printf("%c",c);
    fclose(g);

    g = fopen ("fisier.txt","r+");
    fputc('z',g);
    fflush(g);

    if (fclose(g) != 0)
        printf("\n Probleme la inchiderea fisierelor!\n");

    return 0;
}
```



```

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

int main()
{
    FILE *g = fopen ("fisier.txt", "r+");
    char c;

    if (g == NULL)
        printf("\n Nu s-a putut deschide! \n");

    c = fgetc(g);
    printf("%c", c);
    fclose(g);

    g = fopen ("fisier.txt", "r+");
    fputc('z', g);
    fflush(g);

    if (fclose(g) != 0)
        printf("\n Probleme la inchiderea fisierelor!\n");

    return 0;
}

```

Rezultat: suprascrierea
primului caracter

fisier.txt	
1	zbc
2	

Redeschiderea fisierului implica resetarea cursorului in punctul de inceput al fisierului!

- **Adaugand un + dupa modifierul de acces (e.g. w+,r+,a+), fisierul se deschide cu permisiuni de citire/scriere.**

Cu toate acestea:

- a) **Dupa ce s-a citit din fisier, va trebui apelata o functie de pozitionare in fisier (fseek, fsetpos, rewind)**
- b) **Dupa scriere, va trebui apelata fflush() sau o functie de pozitionare in fisier, inainte de a citi.**

=> Ce facem daca dorim scrierea la pozitia curenta a cursorului?

=> Ce facem daca dorim scrierea la pozitia curenta a cursorului?

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

int main()
{
    FILE *g = fopen ("fisier.txt", "r+");
    char c;

    if (g == NULL)
        printf("\n Nu s-a putut deschide! \n");

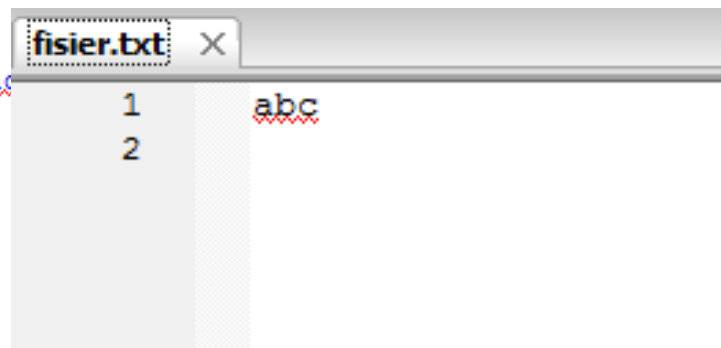
    c = fgetc(g);
    printf("%c", c);

    fseek(g, 0, SEEK_CUR);

    fputc('z', g);
    fflush(g);

    if (fclose(g) != 0)
        printf("\n Probleme la inchidere");

    return 0;
}
```



=> Ce facem daca dorim scrierea la pozitia curenta a cursorului?

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

int main()
{
    FILE *g = fopen ("fisier.txt", "r+");
    char c;

    if (g == NULL)
        printf("\n Nu s-a putut deschide! \n");

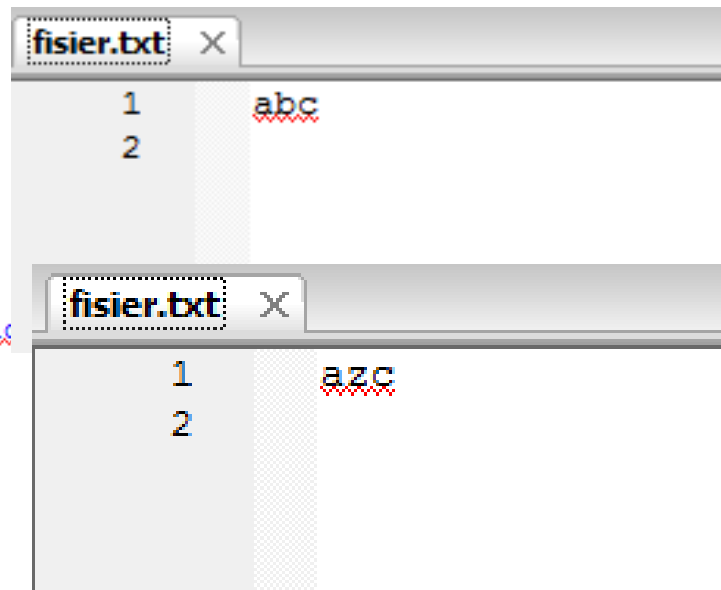
    c = fgetc(g);
    printf("%c", c);

    fseek(g, 0, SEEK_CUR);

    fputc('z', g);
    fflush(g);

    if (fclose(g) != 0)
        printf("\n Probleme la inchidere");

    return 0;
}
```



Schimbare caracter de la pozitia curenta!

LUCRUL CU FIȘIERE

- ❑ Închiderea unui fișier = închiderea unui flux către acel fișier.

Se realizează folosind funcția `fclose`:

- ❑ sintaxa

`int *fclose(FILE *f)`

`f` = pointer la `FILE` care realizează legătura cu fișierul pe care vreau să-l închid

- ❑ `fclose` întoarce 0 dacă închiderea s-a efectuat cu succes și EOF în caz de eroare.
- ❑ Toate fișierele în care s-a scris trebuie închise (dacă se realizează doar citire, fișierul nu trebuie închis).
- ❑ tastatura și imprimanta sunt considerate fișiere text. Ele nu trebuie deschise și închise.

LUCRUL CU FIȘIERE

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

int main()
{
    FILE *g = fopen ("fisier.txt", "w");
    int a;

    if (g == NULL)
        printf("Nu s-a putut deschide!");

    fputc('a', g);

    a = fclose(g);
    printf("%d", a);

    return 0;
}
```


LUCRUL CU FIȘIERE

□ exemplu:

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

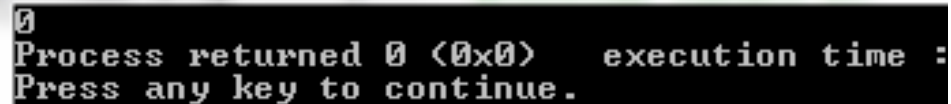
int main()
{
    FILE *g = fopen ("fisier.txt", "w");
    int a;

    if (g == NULL)
        printf("Nu s-a putut deschide!");

    fputc('a', g);

    a = fclose(g);
    printf("%d", a);

    return 0;
}
```



```
0
Process returned 0 (0x0) execution time :
Press any key to continue.
```

LUCRUL CU FIȘIERE

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

- ❑ **sintaxa**

`int feof(FILE *f)`

`f` = pointer la 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.

LUCRUL CU FIȘIERE

■ Detectarea sfârșitului de fișier

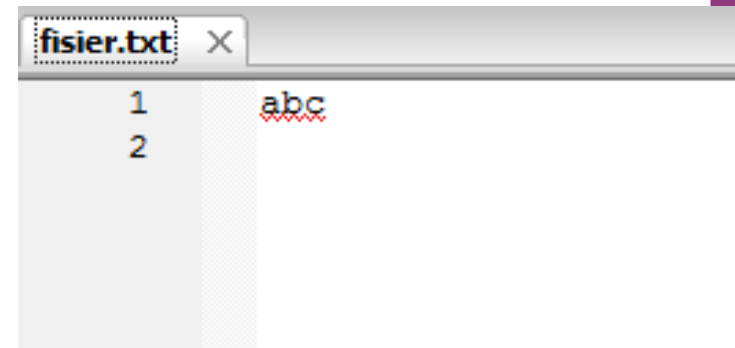
```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    FILE *g = fopen ("fisier.txt", "r+");
    char c;

    if (g == NULL)
        printf("eroare!");

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

    return 0;
}
```



1	abc
2	

LUCRUL CU FIȘIERE

■ Detectarea sfârșitului de fișier

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

```
int main()
```

```
{
```

```
FILE *g = fopen ("fisier.txt", "r+");
```

```
char c;
```

```
if (g == NULL)
```

```
    printf("eroare!");
```

```
while (!feof(g))
```

```
{
```

```
    c = fgetc(g);
```

```
    printf("a<97> b<98> c<99>
```

```
<10> <-1>
```

```
Process returned 0 (0x0)   execution time : 0.01
```

```
Press any key to continue.
```

```
return 0;
```

```
}
```

fisier.txt

1	abc
2	

CUPRINSUL CURSULUI DE AZI

1. Funcții de citire/scriere.
2. Fișiere: noțiuni generale
3. Fișiere text: funcții specifice de manipulare.

FUNCȚII DE CITIRE/SCRIERE LA NIVEL DE CARACTER

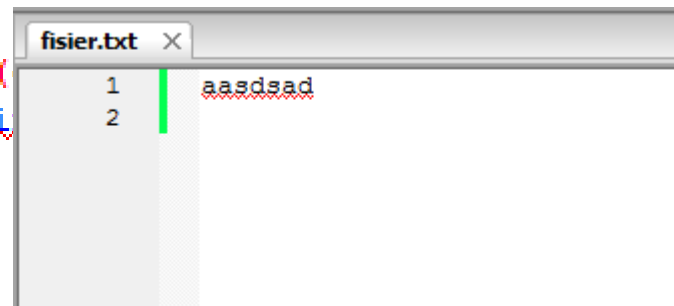
- ❑ `int fgetc(FILE *f)` returneaza codul ASCII al caracterului citit din fișierul `f`.
 - ❑ dacă s-a ajuns la finalul fișierului sau a avut loc o eroare la citire întoarce EOF (= -1).
- ❑ `int fputc(int c, FILE *f)` scrie caracterul cu codul ASCII `c` în fișierul `f`.
 - ❑ întoarce EOF (= -1) în caz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

FUNCTII DE CITIRE/SCRIERE LA NIVEL DE CARACTER

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

```
int main()
```

```
{  
    char nume[30]="fichier_copy.txt";  
    FILE *f = fopen ("fichier.txt", "r");  
    FILE *g = fopen (nume, "w");  
    char c;  
  
    if (g == NULL)  
        printf("\n Nu s-a putut deschide! \n");  
  
    while ((c = fgetc(f))!=EOF)  
        fputc(c,g);  
  
    if ((fclose(f)!=0) || (fclose(  
        printf("\n Probleme la i  
  
    return 0;  
}
```

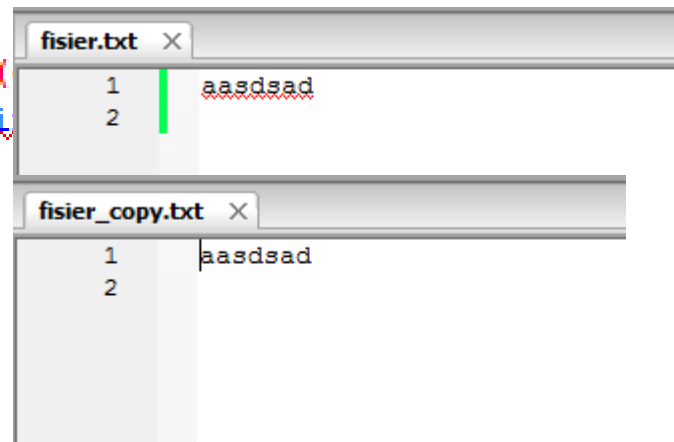


FUNCTII DE CITIRE/SCRIERE LA NIVEL DE CARACTER

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

```
int main()
```

```
{  
    char nume[30]="fichier_copy.txt";  
    FILE *f = fopen ("fichier.txt", "r");  
    FILE *g = fopen (nume, "w");  
    char c;  
  
    if (g == NULL)  
        printf("\n Nu s-a putut deschide! \n");  
  
    while ((c = fgetc(f))!=EOF)  
        fputc(c,g);  
  
    if ((fclose(f)!=0) || (fclose(  
        printf("\n Probleme la i  
  
    return 0;  
}
```



FUNCTII DE CITIRE/SCRIERE LA NIVEL DE CARACTER

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

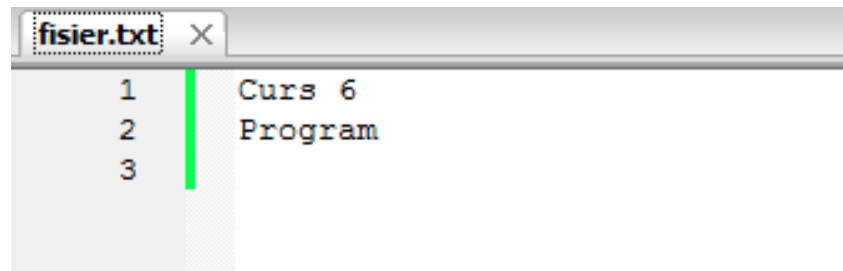
int main()
{
    FILE *f = fopen ("fisier.txt", "r");
    char c;

    if (f == NULL)
        printf("\n Nu s-a putut deschide! \n");

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

    if (fclose(f) != 0)
        printf("\n Probleme la inchiderea fisierelor!\n");

    return 0;
}
```



FUNCTII DE CITIRE/SCRIERE LA NIVEL DE CARACTER

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

int main()
{
    FILE *f = fopen ("fisier.txt", "r");
    char c;

    if (f == NULL)
        printf("\n Nu s-a putut deschide! \n");

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

        if (fclose(f) != 0)
            printf("\n Probleme la inchiderea fisierului");

        return 0;
    }
}
```

fisier.txt X

1	Curs 6
2	Program
3	

```
G(67) u(117) r(114) s(115) (32) 6(54)
(10) P(80) r(114) o(111) g(103) r(114) a(97) m(109)
(10) (-1)
Process returned 0 (0x0) execution time : 0.031 s
Press any key to continue.
```

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 numărul de caractere scrise, sau EOF in caz de eroare.

FUNCTII DE CITIRE/SCRIERE LA NIVEL DE LINIE

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

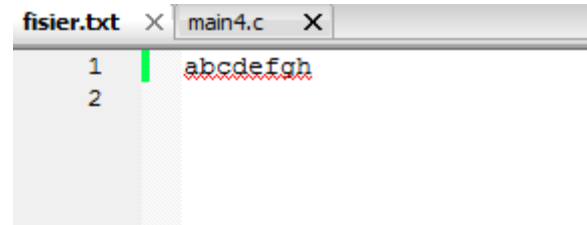
int main()
{
    FILE *g = fopen ("fisier.txt", "r+");
    char sir[30];

    if (g == NULL)
        printf("\n Nu s-a putut deschide! \n");

    fgets(sir, 20, g);
    printf("%s", sir);

    if (fclose(g) != 0)
        printf("\n Probleme la inchiderea fisierelor!\n");

    return 0;
}
```



The screenshot shows a text editor window with two tabs: 'fisier.txt' and 'main4.c'. The 'fisier.txt' tab is active, displaying a file with two lines of text. Line 1 contains 'abcdefgh' and line 2 is empty. A green cursor is positioned at the start of line 1.

Line	Content
1	abcdefgh
2	

FUNCTII DE CITIRE/SCRIERE LA NIVEL DE LINIE

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

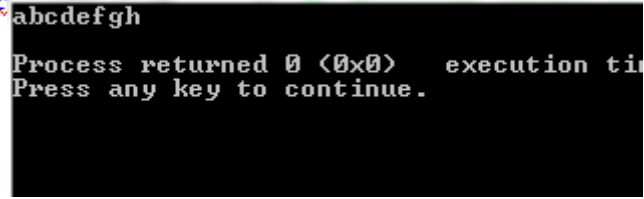
int main()
{
    FILE *g = fopen ("fisier.txt", "r+");
    char sir[30];

    if (g == NULL)
        printf("\n Nu s-a putut deschide! \n");

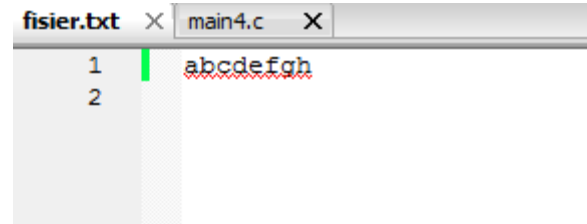
    fgets(sir, 20, g);
    printf("%s", sir);

    if (fclose(g) != 0)
        printf("\n Probleme la inchiderea fisierelor");

    return 0;
}
```



```
abcdefgh
Process returned 0 (0x0) execution time: 0.000 sec.
Press any key to continue.
```



```
fisier.txt X main4.c X
1 abcdefgh
2 abcdefgh
```

FUNCȚII 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

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

int main()
{
    FILE *f = fopen ("NUMAR.txt", "w");
    int x;

    if (f == NULL)
        fprintf(stdin, "\n Nu s-a putut deschide! \n");

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

    fclose(f);

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

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

    fclose(f);
    return 0;
}
```

FUNCTȚII DE POZIȚIONARE ÎNTR-UN FIȘIER

- ❑ în C ne putem poziționa pe un anumit octet din fișier. Funcțiile 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 în 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 POZIȚIONARE ÎNTR-UN FIȘIER

□ Exemplu: aflarea dimensiunii unui fișier

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

int main()
{
    FILE *g = fopen ("fisier.txt", "r+");
    char nr_linii = 0;
    long int nr_octeti;

    if (g == NULL)
        printf("\n Nu s-a putut deschide! \n");

    fseek(g, 0, SEEK_END);
    nr_octeti = ftell(g);
    printf("%ld", nr_octeti);

    if (fclose(g) != 0)
        printf("\n Probleme la inchiderea fisierelor!\n");

    return 0;
}
```

fisier.txt		X
1	abcdefgh	
2		

FUNCTII DE POZITIONARE ÎNTR-UN FIȘIER

□ Exemplu: aflarea dimensiunii unui fișier

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

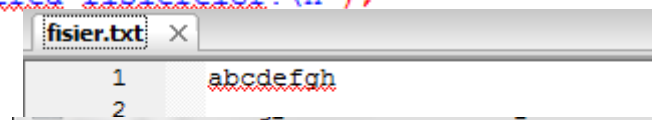
int main()
{
    FILE *g = fopen ("fisier.txt","r+");
    char nr_linii = 0;
    long int nr_octeti;

    if (g == NULL)
        printf("\n Nu s-a putut deschide! \n");

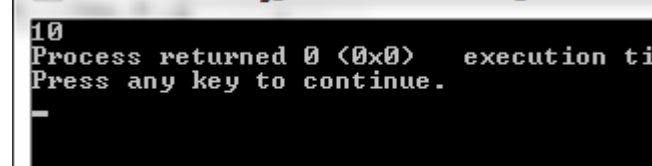
    fseek(g,0,SEEK_END);
    nr_octeti = ftell(g);
    printf("%ld",nr_octeti);

    if (fclose(g)!=0)
        printf("\n Probleme la inchiderea fisierelor!\n");

    return 0;
}
```



1	abcdefghijkl
2	



```
10
Process returned 0 (0x0)   execution time: 0.000 sec
Press any key to continue.
```

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);**

- ❑ redenumeste fișierul cu numele = nume_vechi cu nume_nou. Întoarce 0 în caz de succes, 1 în caz de eroare;