

Programarea calculatoarelor

FMI

Secția Calculatoare și tehnologia informației, anul I

Cursul 9 / 02.12.2024

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: pointeri, tablouri, șiruri de caractere, structuri, uniuni, câmpuri de biți, enumerări
- 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. Recapitulare. Lucrul cu fișiere**
2. Fișiere binare
3. Deschiderea. Funcții de citire/ scriere
4. Alte funcții

Recapitulare. Lucrul cu fișiere

- ❑ **Fișier = o colecție de date (de obicei de același tip), stocate pe suport extern.**
- ❑ Fișierele sunt entități ale sistemului de operare: numele lor respectă convențiile sistemului, fără legătură cu un anumit limbaj de programare.
- ❑ Conținutul unui fișier este variabil (numărul de elemente poate fi chiar nul) și poate fi format din: **texte, numere, informații binare** (executabile, numere, imagini sau sunete în format binar).
- ❑ Fișierele se folosesc în principal pentru păstrarea permanentă a unor date importante, dar și pentru memorarea unor date inițiale (de intrare) sau date de ieșire numerice voluminoase, de interes pentru anumite programe.
- ❑ Noțiunea de fișier include orice flux de date (**I/O stream**) din exterior spre memorie și viceversa.

Recapitulare. Lucrul cu fișiere

- ❑ Operațiunile pentru prelucrarea fișierelor sunt:
 - ❑ crearea fișierului;
 - ❑ asignarea fișierului intern (logic) la unul extern (fizic);
 - ❑ deschiderea fișierului;
 - ❑ operații de acces la date;
 - ❑ închiderea fișierului.

Recapitulare. Lucrul cu fișiere

- ❑ Există fișiere standard, care sunt gestionate automat de sistem, dar asupra cărora se poate interveni și în mod explicit:
 - ❑ fișierul standard de intrare **stdin** (STanDard INput);
 - ❑ fișierul standard de ieșire **stdout** (STanDard OUTput);
 - ❑ fișierul standard scriere mesaje de eroare **stderr** (STanDard ERRor);

Categorii de funcții folosite pentru prelucrarea fișierelor

- ❑ **funcții de prelucrare generală**: se aplică tuturor fișierelor, indiferent de tipul informației conținute
- ❑ **funcții de citire/scriere cu format**: pentru citirea/scrierea fișierelor text care conțin informație de tip text (linii de text, separate prin perechea CR/LF, iar la sfârșit se găsește caracterul CTRL-Z)
- ❑ **funcții de citire/scriere fără format**: pentru citirea/scrierea fișierelor binare
- ❑ **funcții de citire/scriere pentru caractere**: pentru transferul unui singur caracter între un fișier text și memorie
- ❑ **funcții de citire/scriere pentru șiruri de caractere**: pentru transferul unui șir de caractere între un fișier text și memorie

Funcțiile de citire/scriere deplasează pointerul de citire/scriere al fișierului spre sfârșitul acestuia, cu un număr de octeți egal cu numărul de octeți transferați fără a trece de sfârșitul fișierului.

Recapitulare. Lucrul cu fișiere

- ❑ Deschiderea unui fișier existent, precum și crearea unui fișier nou se fac cu ajutorul funcției **fopen**, care are următorul prototip:

FILE *fopen (const char *cale_nume, const char *mod);

- ❑ unde:
 - ❑ cale_nume este un pointer spre un șir de caractere care definește calea de nume a fișierului;
 - ❑ mod este un pointer spre un șir de caractere care definește modul de prelucrare a fișierului deschis

Recapitulare. Lucrul cu fișiere

□ Cum se definește modul de prelucrare a unui fișier deschis:

“r” – citire în mod text (read);

“w” – scriere în mod text (write);

“a” – adăugare în mod text (append);

“r+” – citire/scriere în mod text (modificare);

“w+” – citire/scriere în mod text (creare);

“rb” – citire în mod binar;

“wb” – scriere în mod binar;

“r+b” – citire/scriere în mod binar (modificare);

“w+b” – citire/scriere în mod binar (creare);

“ab” – adăugare de înregistrări în mod binar

Cuprinsul cursului de azi

1. Recapitulare. Lucrul cu fișiere
- 2. Fișiere binare**
3. Deschiderea. Funcții de citire/ scriere
4. Alte funcții

Fișiere binare

- ❑ **fișier binar** = șir 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)
 - ❑ o structură cu date de diferite tipuri



Fișier binar vs fișier text

- ❑ **fișier binar** = șir de octeți neformatat pe linii care este stocat pe suport magnetic/optic. Octeții nu sunt considerați ca fiind coduri de caractere
- ❑ Diferența fișier binar – fișier text:
 - ❑ un **fișier binar** se accesează ca o **sucesiune de octeți**, cărora funcțiile de citire și scriere din fișier nu le dau nici o interpretare
 - ❑ un **fișier text** se accesează ca o **sucesiune de linii de text** de lungime variabilă (încheiate cu un terminator de linie : ‘\n’) utilizând un set dedicat de funcții din biblioteca standard

Cuprinsul cursului de azi

1. Recapitulare. Lucrul cu fișiere
2. Fișiere binare
3. **Deschiderea. Funcții de citire/ scriere**
4. Alte funcții

Deschiderea fișierelor

- **FILE *fopen(char *nume_fisier, char *mod_deschidere)**
 - nume_fisier = numele fișierului
 - mod_deschidere = șir de caracter 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

Crearea/ închiderea unui fișier binar

- ❑ se definește un pointer la tipul FILE,
 - ❑ se apelează funcția **fopen** cu parametru mod “**wb**”,
 - ❑ se atribuie valoarea returnată pointerului la tipul FILE
-
- ❑ Dacă valoarea returnată este pointerul NULL operația de creare a eșuat.

Crearea/ închiderea unui fișier binar

main.c

```
1  #include <stdio.h>
2
3  int main()
4  {
5      FILE *f; // pointer la tipul FILE
6      char fname[30]; // identificatorul (numele extern) fisierului
7      printf(" Nume fisier: ");
8      scanf("%s",fname);
9      if((f=fopen(fname,"wb")) == NULL)
10         printf(" Fisierul %s nu a fost creat \n",fname);
11     else
12         printf(" Fisierul %s a fost creat \n",fname);
13     fclose(f); // inchidere fisier
14     return 0;
15 }
16
```



input

Nume fisier:

Funcții de citire/ scriere

`int fwrite(void *tablou, int dim_elem, int nr_elem, FILE *f)`

- **scrie** în fișierul referit de `f` cel mult `nr_elem` elemente de dimensiune `dim_elem` de la adresa `tablou`;

`int fread(void *tablou, int dim_elem, int nr_elem, FILE *f)`

- **citește** cel mult `nr_elem` elemente de dimensiune `dim_elem` din fișierul referit de `f` la adresa `tablou`.

Funcții de citire/ scriere

Ex.1. Scriere

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

int main()
{
    FILE *f, *g;
    int x = 1936880995;

    f = fopen("fisier_nrb.out", "wb");
    g = fopen("fisier_nrt.out", "w");

    fwrite(&x, sizeof(int), 1, f);
    fprintf(g, "%d", x);

    printf("Cod ASCII c = %d\n", 'c');
    printf("Cod ASCII u = %d\n", 'u');
    printf("Cod ASCII r = %d\n", 'r');
    printf("Cod ASCII s = %d\n", 's');
    //printf("%d\n", (99<<24) + (117<<16) + (114<<8) + 115);
    printf("%d\n", (115<<24) + (114<<16) + (117<<8) + 99);

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

Funcții de citire/ scriere

Ex.1. Scriere

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

int main()
{
    FILE *f, *g;
    int x = 1936880995;

    f = fopen("fisier_nrb.out", "wb");
    g = fopen("fisier_nrt.out", "w");

    fwrite(&x, sizeof(int), 1, f);
    fprintf(g, "%d", x);

    printf("Cod ASCII c = %d\n", 'c');
    printf("Cod ASCII u = %d\n", 'u');
    printf("Cod ASCII r = %d\n", 'r');
    printf("Cod ASCII s = %d\n", 's');
    //printf("%d\n", (99<<24) + (117<<16) + (114<<8) + 115);
    printf("%d\n", (115<<24) + (114<<16) + (117<<8) + 99);

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

```
Cod ASCII c = 99
Cod ASCII u = 117
Cod ASCII r = 114
Cod ASCII s = 115
1936880995
```

Funcții de citire/ scriere

Ex.1. Scriere

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

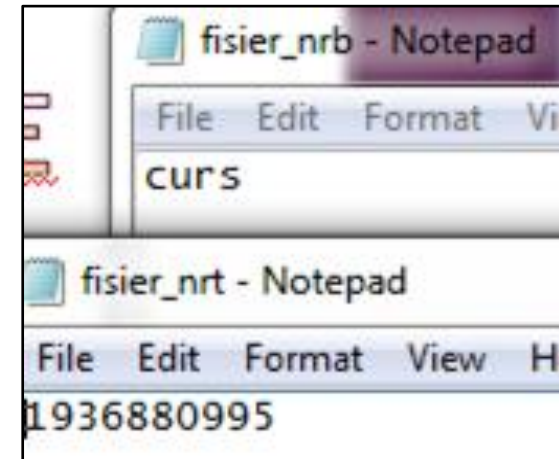
int main()
{
    FILE *f, *g;
    int x = 1936880995;

    f = fopen("fisier_nrb.out", "wb");
    g = fopen("fisier_nrt.out", "w");

    fwrite(&x, sizeof(int), 1, f);
    fprintf(g, "%d", x);

    printf("Cod ASCII c = %d\n", 'c');
    printf("Cod ASCII u = %d\n", 'u');
    printf("Cod ASCII r = %d\n", 'r');
    printf("Cod ASCII s = %d\n", 's');
    //printf("%d\n", (99<<24) + (117<<16) + (114<<8) + 115);
    printf("%d\n", (115<<24) + (114<<16) + (117<<8) + 99);

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



```
Cod ASCII c = 99
Cod ASCII u = 117
Cod ASCII r = 114
Cod ASCII s = 115
1936880995
```

Funcții de citire/ scriere

Ex.2. Citire

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

int main()
{
    FILE *f;
    int x; char c;
    f = fopen("fisier_nrb.out", "rb");

    if (f == NULL)
    {printf("Deschidere esuata!"); exit(1);}

    while (fread(&x, sizeof(int), 1, f)==1)
        printf("x = %d\n", x);
    fclose(f);

    f = fopen("fisier_nrb.out", "rb");

    while (fread(&c, sizeof(char), 1, f)==1)
        printf("c = %d\n", c);
}
```

Funcții de citire/ scriere

Ex.2. Citire

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


int main()
{
    FILE *f;
    int x; char c;
    f = fopen("fisier_nrb.out", "rb");

    if (f == NULL)
    {printf("Deschidere esuata!"); exit(1);}

    while (fread(&x, sizeof(int), 1, f)==1)
        printf("x = %d\n", x);
    fclose(f);

    f = fopen("fisier_nrb.out", "rb");

    while (fread(&c, sizeof(char), 1, f)==1)
        printf("c = %d\n", c);
```



```
x = 1936880995
c = 99
c = 117
c = 114
c = 115
```

Funcții de citire/ scriere

Ex.3. Scriere

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

int main()
{
    FILE *f1, *f2, *f3;
    f1 = fopen("fb1.txt", "wb");
    f2 = fopen("fb2.txt", "wb");
    f3 = fopen("fb3.txt", "wb");

    if ((f1 == NULL) || (f2 == NULL) || (f3 == NULL))
    {printf("Deschidere esuata!"); exit(1);}

    int v[4] = {50, 51, 52, 53}, i;

    for (i=0; i<4; i++)
        fwrite(&v[i], sizeof(int), 1, f1);

    fwrite(v, sizeof(int), 4, f2);

    fwrite(v, 4*sizeof(int), 1, f3);

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

    return 0;
}
```

Funcții de citire/scriere

Ex.3. Scriere

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

int main()
{
    FILE *f1, *f2, *f3;
    f1 = fopen("fb1.txt", "wb");
    f2 = fopen("fb2.txt", "wb");
    f3 = fopen("fb3.txt", "wb");

    if ((f1 == NULL) || (f2 == NULL) || (f3 == NULL))
    {printf("Deschidere esuata!"); exit(1);}

    int v[4] = {50, 51, 52, 53}, i;

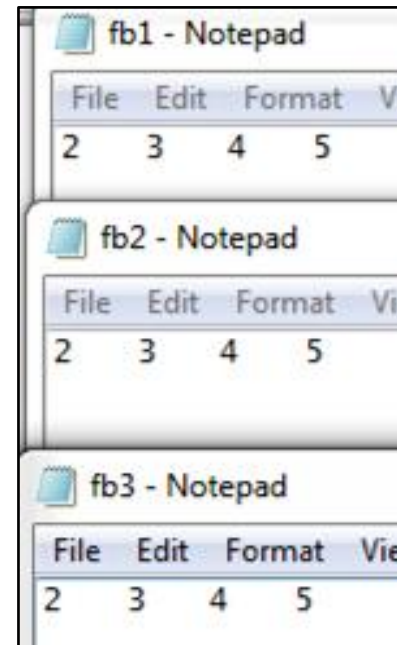
    for (i=0; i<4; i++)
        fwrite(&v[i], sizeof(int), 1, f1);

    fwrite(v, sizeof(int), 4, f2);

    fwrite(v, 4*sizeof(int), 1, f3);

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

    return 0;
}
```



Funcții de citire/ scriere

Ex.3. Scriere

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

int main()
{
    FILE *f1, *f2, *f3;
    f1 = fopen("fb1.txt", "wb");
    f2 = fopen("fb2.txt", "wb");
    f3 = fopen("fb3.txt", "wb");

    if ((f1 == NULL) || (f2 == NULL) || (f3 == NULL))
    {printf("Deschidere esuata!"); exit(1);}

    int v[4] = {50, 51, 52, 53}, i;

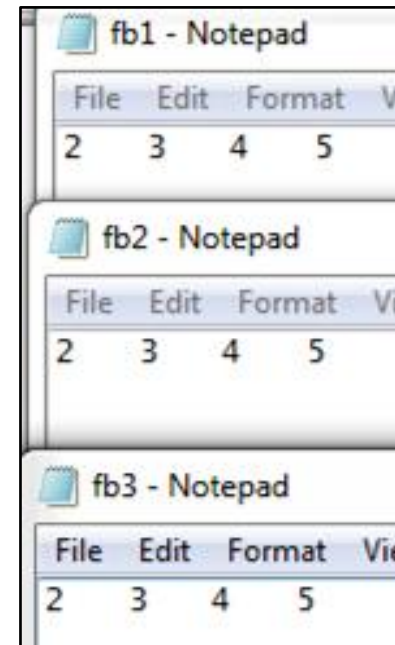
    for (i=0; i<4; i++)
        fwrite(&v[i], sizeof(int), 1, f1);

    fwrite(v, sizeof(int), 4, f2);

    fwrite(v, 4*sizeof(int), 1, f3);

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

    return 0;
}
```



	Dec	Hx	Oct	Html	Chr
32	20	040	 	Space	
33	21	041	!	!	
34	22	042	"	"	
35	23	043	#	#	
36	24	044	$	\$	
37	25	045	%	%	
38	26	046	&	&	
39	27	047	'	'	
40	28	050	((
41	29	051))	
42	2A	052	*	*	
43	2B	053	+	+	
44	2C	054	,	,	
45	2D	055	-	-	
46	2E	056	.	.	
47	2F	057	/	/	
48	30	060	0	0	
49	31	061	1	1	
50	32	062	2	2	
51	33	063	3	3	
52	34	064	4	4	
53	35	065	5	5	

Funcții de citire/ scriere

Ex.4. Citire

```
void afisare(int* v, int dim)
{
    int i;
    for (i = 0; i<dim; i++)
        printf("%d ", v[i]);
    printf("\n");
}

int main()
{
    FILE *f1, *f2, *f3;
    f1 = fopen("fb1.txt", "rb");
    f2 = fopen("fb2.txt", "rb");
    f3 = fopen("fb3.txt", "rb");

    if ((f1 == NULL) || (f2 == NULL) || (f3 == NULL))
        {printf("Deschidere esuata!"); exit(1);}

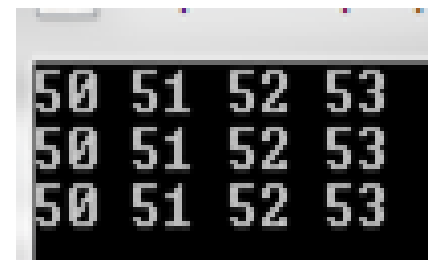
    int v[5], i;

    for (i=0; i<4; i++)
        fread(&v[i], sizeof(int), 1, f1);

    afisare(v, 4);

    fread(v, sizeof(int), 4, f2);
    afisare(v, 4);

    fread(v, 4*sizeof(int), 1, f3);
    afisare(v, 4);
}
```



50	51	52	53
50	51	52	53
50	51	52	53

Funcții de citire/ scriere

□ Scrierea unei structuri

```
#include <stdio.h>
#include <stdlib.h>
typedef struct
{
    char nume[20];
    float medie;
} student;

int main()
{
    FILE *f;
    f = fopen("fisier_struct.out", "wb");

    if (f == NULL)
    {printf("Deschidere esuata!"); exit(1);}

    student s;
    strcpy(s.nume, "Alexandru");
    s.medie = 9.5;
    fwrite(&s, sizeof(s), 1, f);

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

Funcții de citire/ scriere

❑ Scrierea unei structuri

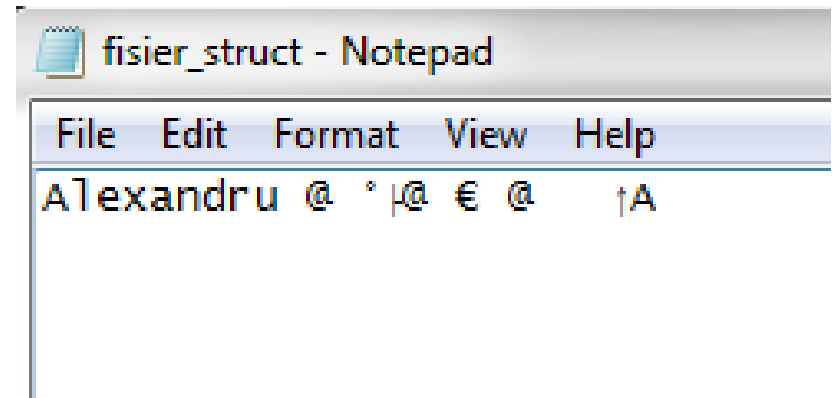
```
#include <stdio.h>
#include <stdlib.h>
typedef struct
{
    char nume[20];
    float medie;
} student;

int main()
{
    FILE *f;
    f = fopen("fisier_struct.out", "wb");

    if (f == NULL)
    {printf("Deschidere esuata!"); exit(1);}

    student s;
    strcpy(s.nume, "Alexandru");
    s.medie = 9.5;
    fwrite(&s, sizeof(s), 1, f);

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



Funcții de citire/ scriere

□ Citirea unei structuri

```
#include <stdlib.h>
typedef struct
{
    char nume[20];
    float medie;
} student;

int main()
{
    FILE *f;
    f = fopen("fisier_struct.out", "rb");

    if (f == NULL)
    {printf("Deschidere esuata!"); exit(1);}

    student s;
    fread(&s, sizeof(s), 1, f);
    printf("Student cu numele: %s si media %f \n", s.nume, s.medie);

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

Funcții de citire/ scriere

❑ Citirea unei structuri

```
#include <stdlib.h>
typedef struct
{
    char nume[20];
    float medie;
} student;

int main()
{
    FILE *f;
    f = fopen("fisier_struct.out", "rb");

    if (f == NULL)
    {printf("Deschidere esuata!"); exit(1);}

    student s;
    fread(&s, sizeof(s), 1, f);
    printf("Student cu numele: %s si media %f \n", s.nume, s.medie);

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

Student cu numele: Alexandru si media 9.500000

Process returned 0 (0x0) execution time : 0.008 s
Press any key to continue.

Exemplu: Scrieți un program C care sa creeze un fișier cu numele **student.txt** pentru citirea numelui si notei unui număr de studenți stabilit de utilizator.

main.c

```
1  #include <stdio.h>
2
3  int main()
4  {
5      char name[50];
6      int nota,i,n;
7      printf("Numarul de studenti: ");
8      scanf("%d",&n);
9      FILE *f;
10     f=(fopen("F:\\temp\\student.txt","w"));
11     if(f==NULL)
12         printf("Err!");
13     else
14         for(i=0;i<n;++i)
15         {
16             printf("Pentru studentul %d\nInserati numele: ",i+1);
17             scanf("%s",name);
18             printf("Inserati nota: ");
19             scanf("%d",&nota);
20             fprintf(f,"\\nNume: %s \\nnota=%d \\n",name,nota);
21         }
22     fclose(f);
23     return 0;
24 }
25
```

Exemplu: Scrieți un program C care sa creeze un fișier cu numele **student.txt** pentru citirea numelui si notei unui număr de studenți stabilit de utilizator.

```
main.c
1  #include <stdio.h>
2
3  int main()
4  {
5      char name[50];
6      int nota,i,n;
7      printf("Numarul de studenti: ");
8      scanf("%d",&n);
9      FILE *f;
10     f=(fopen("F:\\temp\\student.txt","w"));
11     if(f==NULL)
12         printf("Err!");
13     else
14         for(i=0;i<n;++i)
15         {
16             printf("Pentru studentul %d\\nInserati numele: ",i+1);
17             scanf("%s",name);
18             printf("Inserati nota: ");
19             scanf("%d",&nota);
20             fprintf(f,"\\nNume: %s \\nnota=%d \\n",name,nota);
21         }
22     fclose(f);
23     return 0;
24 }
25
```

```
main.c  F:\temp\student.txt
1
2  Nume: Pop
3  nota=9
4
5  Nume: Albu
6  nota=10
7

Numarul de studenti: 2
Pentru studentul 1
Inserati numele: Pop
Inserati nota: 9
Pentru studentul 2
Inserati numele: Albu
Inserati nota: 10
```

Reluarea rulării are ca efect pierderea datelor din fișier.

Exemplu: Scrieți un program C care să creeze un fișier cu numele **student.txt** pentru citirea numelui și notei unui număr de studenți stabilit de utilizator. Dacă fișierul există, să se adauge informațiile în fișierul existent.

```
main.c  F:\temp\student.txt  Ctrl+S
1  #include <stdio.h>
2
3  int main()
4  {
5      char nume[50];
6      int nota,i,n;
7      printf("Numarul de studenti: ");
8      scanf("%d",&n);
9      FILE *f;
10     f=(fopen("F:\\temp\\student.txt","a"));
11     if(f==NULL)
12         printf("Eroare!");
13     else
14         for(i=0;i<n;++i)
15         {
16             printf("Pentru studentul %d\nInserati numele: ",i+1);
17             scanf("%s",nume);
18             printf("Inserati nota: ");
19             scanf("%d",&nota);
20             fprintf(f,"\nNume: %s \nnota=%d \n",nume,nota);
21         }
22     fclose(f);
23     return 0;
24 }
```

```
main.c  F:\temp\student.txt
1
2  Nume: Andrei
3  nota=9
4
5  Nume: Matei
6  nota=8
7

Numarul de studenti: 2
Pentru studentul 1
Inserati numele: Andrei
Inserati nota: 9
Pentru studentul 2
Inserati numele: Matei
Inserati nota: 8
```

Exemplu: Scrieți un program C care sa creeze un fișier cu numele **student.txt** pentru citirea numelui si notei unui număr de studenți stabilit de utilizator. Daca fișierul există, să se adauge informațiile in fișierul existent.

```
main.c F:\temp\student.txt Ctrl+S
1  #include <stdio.h>
2
3  int main()
4  {
5      char nume[50];
6      int nota,i,n;
7      printf("Numarul de studenti: ");
8      scanf("%d",&n);
9      FILE *f;
10     f=(fopen("F:\\temp\\student.txt","a"));
11     if(f==NULL)
12         printf("Eroare!");
13     else
14         for(i=0;i<n;++i)
15         {
16             printf("Pentru studentul %d\nInserati numele: ",
17             scanf("%s",nume);
18             printf("Inserati nota: ");
19             scanf("%d",&nota);
20             fprintf(f,"\nNume: %s \nnota=%d \n",nume,nota);
21         }
22     fclose(f);
23     return 0;
24 }
```

```
main.c F:\temp\student.txt
1
2 Nume: Andrei
3 nota=9
4
5 Nume: Matei
6 nota=8
7
8 Nume: Ana
9 nota=9
10
11 Nume: Maria
12 nota=10
13
14 Nume: Ionel
15 nota=8
16
```

```
Numarul de studenti: 3
Pentru studentul 1
Inserati numele: Ana
Inserati nota: 9
Pentru studentul 2
Inserati numele: Maria
Inserati nota: 10
Pentru studentul 3
Inserati numele: Ionel
Inserati nota: 8
```

Cuprinsul cursului de azi

1. Recapitulare. Lucrul cu fișiere
2. Fișiere binare
3. Deschiderea. Funcții de citire/ scriere
4. **Alte funcții**

Funcții de poziționare într-un fișier

- ❑ *Modificarea poziției indicatorului de citire/ scriere*
- ❑ *Poziționarea indicatorului de citire/ scriere la începutul fișierului*
- ❑ *Testarea sfârșitului de fișier*
- ❑ *Determinarea poziției curente a indicatorului de citire/ scriere*
- ❑ *Poziționarea absolută a indicatorului de citire/ scriere*

Modificarea poziției indicatorului de citire/ scriere

□ Funcția fseek:

int fseek(FILE* f, long deplasare, int origine);

- parametrul **deplasare** reprezintă numărul de octeți cu care se deplasează indicatorul în fișierul **f**,
- parametrul **origine** reprezintă referința față de care se deplasează indicatorul de citire / scriere.

□ parametrul **origine** poate avea valorile:

- SEEK_SET (=0) poziționare față de începutul fișierului;
- SEEK_CUR (=1) poziționare față de poziția curentă;
- SEEK_END (=2) poziționare față de sfârșitul fișierului.

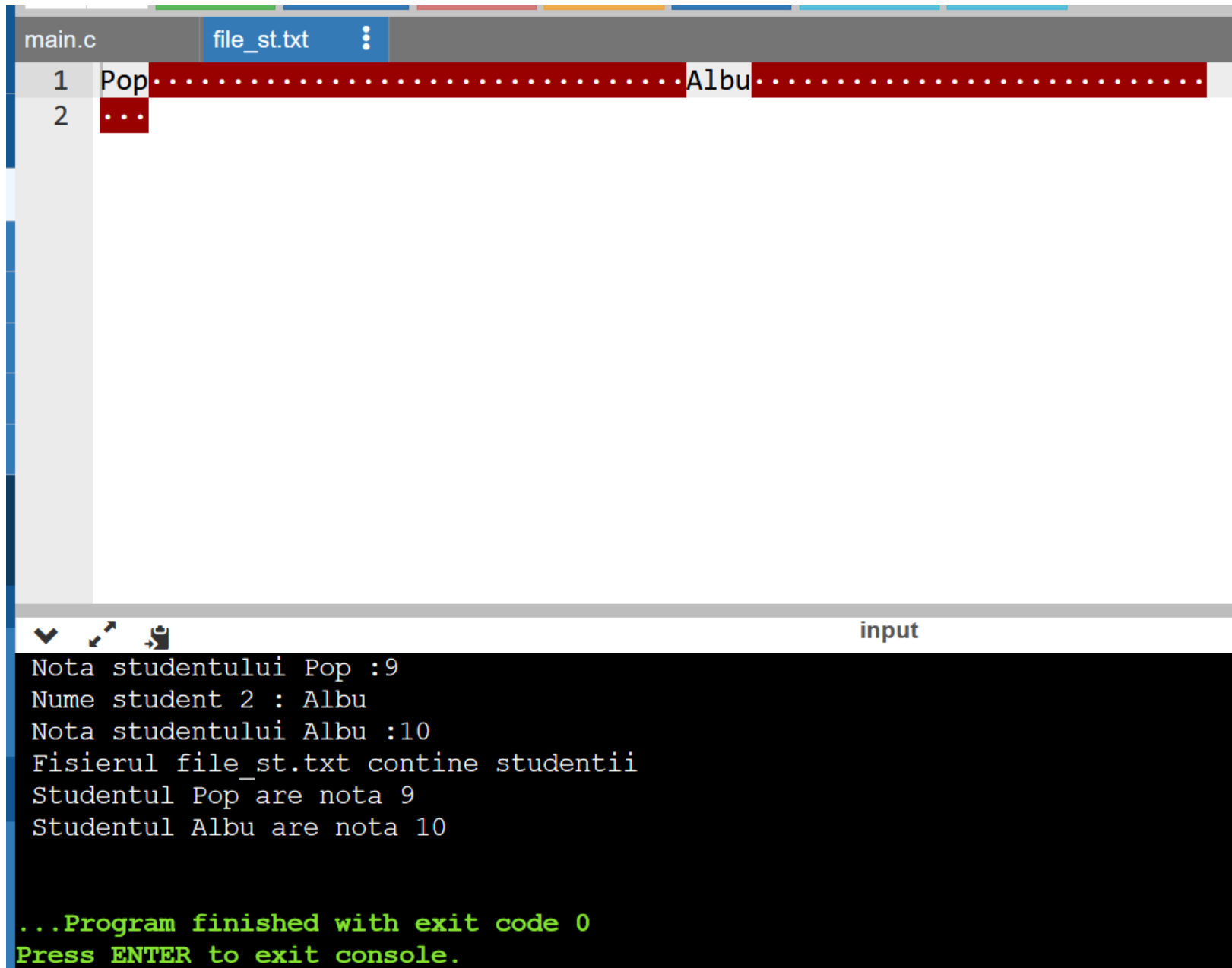
□ Funcția returnează valoarea 0 în caz de succes (uneori și în caz de eșec).

□ Se semnalează eroare prin returnarea unei valori nenule numai în cazul în care fișierul cu descriptorul **f** nu este deschis.

Exemplu:

```
main.c  F9 st.txt  Ctrl+S
1  #include <stdio.h>
2
3  struct student {
4      char nume[30];int nota;}s;
5
6  int main()
7  {
8      FILE *f; int n,i; char idf[50];
9      printf(" Identificator fisier : ");
10     scanf("%s",idf);
11     if((f=fopen(idf,"wb"))==NULL) // creare fisier binar
12         printf(" Eroare deschidere fisier \n");
13     else
14         printf(" Numar studenti : "); // preluare date de la tastatura
15         scanf("%d",&n);
16         for(i=1;i<=n;i++)
17         {
18             printf(" Nume student %d : ",i); scanf("%s",s.nume);
19             printf(" Nota studentului %s :",s.nume);scanf("%d",&s.nota);
20             fwrite(&s,sizeof(s),1,f); // scriere in fisier
21         }
22         fclose(f); // inchidere fisier
23         if((f=fopen(idf,"rb"))==NULL) // deschidere fisier binar
24             printf(" Eroare deschidere sursa \n");
25         else
26             printf(" Fisierul %s contine studentii \n",idf);
27         for(i=1;i<=n;i++)
28         {
29             fread(&s,sizeof(s),1,f); // citire din fisier
30             printf(" Studentul %s are nota %d \n",s.nume,s.nota);
31         }
32         fclose(f);
33     return 0;
34 }
35
```

Exemplu:



The image shows a C program named `main.c` that reads student data from a file named `file_st.txt`. The program's logic is as follows:

- Line 1: Reads a student's name and grade from the file. The output shows "Pop" with a grade of 9.
- Line 2: Reads another student's name and grade. The output shows "Albu" with a grade of 10.

The console output displays the following information:

```
Nota studentului Pop :9
Nume student 2 : Albu
Nota studentului Albu :10
Fisierul file_st.txt contine studentii
Studentul Pop are nota 9
Studentul Albu are nota 10
```

The program concludes with the message: "...Program finished with exit code 0" and "Press ENTER to exit console."

Poziționarea indicatorului de citire/ scriere la începutul fișierului

- ❑ Funcția rewind:

void rewind(FILE *f);

- ❑ Funcția are un singur parametru de apel și anume descriptorul fișierului implicat.
- ❑ Executarea funcției are ca efect:
 - ❑ poziționarea la începutul fișierului **f** deschis anterior,
 - ❑ resetarea indicatorului de sfârșit de fișier și
 - ❑ resetarea indicatorilor de eroare.
- ❑ După apelul lui rewind poate urma o operație de scriere sau citire din fișier.

Testarea sfârșitului de fișier

□ se realizează prin apelul macro-definiției feof:

int feof(FILE* f);

- macro-ul furnizează valoarea indicatorului de sfârșit de fișier asociat lui **f**.
- valoarea indicatorului este stabilită la fiecare operație de citire din fișier.
- valoarea returnată este:
 - 0, dacă indicatorul are valoarea sfârșit de fișier sau
 - o valoare diferită de zero, în caz contrar.
- apelul lui **feof** trebuie să fie precedat de apelul unei funcții de **citire** din fișier.
- După atingerea sfârșitului de fișier, toate încercările de citire vor eșua, până la apelul funcției **rewind** sau **închiderea și apoi redeschiderea** fișierului.

Determinarea poziției curente a indicatorului de citire/ scriere

- se poate face cu funcția `fgetpos`:

`int fgetpos(FILE* f, fpos_t * pozitie);`

- după apel, la adresa **`pozitie`** se află poziția indicatorului de citire/ scriere din fișierul **`f`**, ca număr relativ al octetului curent.
- primul octet are numărul 0.
- valoarea returnată poate fi folosită pentru poziționare cu funcția **`fsetpos`**.
- funcția returnează:
 - valoarea 0, în caz de succes ,
 - o valoare nenulă, în caz de eroare.

Determinarea poziției curente a indicatorului de citire/ scriere

- ❑ se poate face cu funcția `ftell`:

`long int ftell(FILE* f);`

- ❑ după apel, la adresa poziție se află poziția indicatorului de citire
- ❑ funcția returnează:
 - ❑ poziția în fișierul `f` a indicatorului de citire/ scriere în caz de succes
 - ❑ `-1L` în caz contrar.
- ❑ dacă fișierul este binar, poziția este dată în număr de octeți față de începutul fișierului (dimensiunea maximă a unui fișier în C este de $2^{31}-1$ octeți $\sim 2\text{GB}$)
- ❑ valoarea poate fi folosită pentru poziționare cu funcția **`fseek`**.

Poziționarea absolută a indicatorului de citire/ scriere

- se face cu funcția `fsetpos`:

`int fsetpos(FILE* f, const fpos_t pozitie);`

- Indicatorul de citire/ scriere se mută în fișierul cu descriptorul **`f`** la octetul cu numărul indicat de parametrul **`pozitie`** (care poate fi o valoare obținută prin apelul lui **`fgetpos`**).
- Ambele funcții (**`fgetpos`** și **`fsetpos`**) resetează indicatorul de sfârșit de fișier.

Alte funcții pentru lucrul cu fișiere

- *Redenumirea sau mutarea unui fișier*
- *Ștergerea unui fișier*
- *Furnizare unui nume de fișier*

Redenumirea sau mutarea unui fișier existent

- ❑ se realizează prin apelul funcției cu prototipul:

int rename(const char* n_vechi, const char* n_nou);

- ❑ unde **n_vechi** este vechiul nume al fișierului, iar **n_nou**=numele nou.
- ❑ dacă numele vechi conține numele discului (de exemplu C:), numele nou trebuie să conțină același nume de disc.
- ❑ dacă numele vechi conține o cale, numele nou **nu** este obligat să conțină aceeași cale. *Folosind o altă cale se obține mutarea fișierului pe disc.*
- ❑ folosind aceeași cale (sau nefolosind calea) se obține redenumirea fișierului.
- ❑ **nu** sunt permise caracterele (?, *) în cele două nume.
- ❑ funcția returnează valoarea:
 - ❑ 0, în caz de succes
 - ❑ -1, în caz de eroare.

Ștergerea unui fișier existent

- ❑ se poate realiza prin apelul funcției remove, care are următorul prototip:

int remove(const char* identificador);

- ❑ parametrul identificador reprezintă numele extern al fișierului și care poate să conțină calea de căutare a fișierului.

Furnizare unui nume de fișier

- Funcția

char *tmpnam(char* nume_fisier)

- furnizează un nume de fisier pe care îl pune în nume_fisier care nu există în directorul curent

BUFFERED STREAMS

- ❑ Operațiile cu HDD sunt mult mai încete decât cele cu memoria (RAM). Drept urmare, se folosește un buffer pentru a citi/ scrie blocuri mai mari în memorie înainte de a reciti/rescrie efectiv buffer-ul pe HDD
- ❑ Toate stream-urile deschise cu `fopen()` folosesc buffere dacă se știe că nu se referă la un dispozitiv interactiv
- ❑ Tipuri de buffering: full buffer, line buffer, no buffer
- ❑ Bufferul se poate schimba cu:

`void setbuf (FILE * stream, char * buffer);`

- ❑ Tipul de buffering se poate schimba cu:

`int setvbuf (FILE * stream, char * buffer, int mode, size_t size);`

Golirea explicită a zonei tampon a unui fișier

- se realizează prin apelul funcției cu următorul prototip:

int fflush(FILE* f);

- dacă fișierul cu descriptorul **f** are asociat un buffer de ieșire, funcția scrie în fișier toate informațiile din acesta, la poziția curentă.
- dacă fișierul are asociat un buffer de intrare, funcția îl golește. În caz de succes returnează valoarea zero, iar în caz de eroare valoarea constantei simbolice EOF (definită în stdio.h).
- înainte de a citi un șir de caractere de la tastatură, **buffer-ul trebuie golit** pentru a preveni citirea unui șir vid (datorită unei perechi CR/LF rămase în buffer de la o citire anterioară a unei valori numerice).
- Ștergerea se realizează prin apelul: fflush(stdin);

BUFFERED STREAMS - FFLUSH

int fflush (FILE * stream);

Pentru a forța scrierea buffer-ului stream-ului în fișier

```
#include <stdio.h>

int main()
{
    FILE *f = fopen("myfile2.txt", "w");
    if (f){
        fputc('a', f);
        sleep(10000);
        fflush(f);
        sleep(10000);
        fclose(f);
    }
    return 1;
}
```

Cursul 9

Fișiere binare. Funcții specifice de manipulare

Cursul 10

Structuri de date complexe și autoreferite