13. Fişiere text. Fişiere binare. Fișiere în acces direct.

(Text files. Binary files. Files in direct access.)

1. Obiective:

- Scrierea de programe folosind alte date utilizator şi fişiere text.
- Înţelegerea modalităţii de lucru cu fişiere binare
- Înţelegerea accesului aleator în fişiere
- Scrierea de programe folosind fişiere binare

1'. Objectives:

- Writing programs using other data types and text files.
- Understanding how the binary files work
- Understanding the random access files
- Writing programs that use binary files

2. Rezumat:

Flux-uri (Stream-uri) de date

Sistemul de intrări/ieşiri separă programatorul de echipament printr-o aşa-zisă *abstractizare* a datelor care circulă în permanenţă între maşina de lucru şi program.

Această formă abstractă de comunicare se numeşte *flux(stream)*, iar instrumentul efectiv de comunicare se numeşte *fişier(file)*. Stream-ul asigură un canal de comunicaţie între două componente hardware sau software. Una din componente e numită sursă, ea transmite date pe acest canal, iar cealaltă destinaţie, ea recepţionând datele din canal.

Stream-uri de tip text

Acest tip de stream este practic o secvenţă de caractere ASCII sau UNICODE. Standardul ANSI C permite ca un stream de tip text (caractere ASCII) să fie organizat pe linii, fiecare linie fiind terminată cu un caracter de control, de *linie nouă* , *LF(Line Feed)*, având codul ASCII *0x0A*. Utilizarea acestui caracter este opţională, folosirea sa depinzând de programator şi de modul de implementare a aplicaţiei respective. La fişiere text caracterul *LF* este translatat într-o secvenţă de două caractere, *CR* şi *LF* (*0x0D 0x0A*). În acest mod nu va exista o corespondenţă totală între ceea ce este transmis şi ceea ce conţine fişierul.

Stream-uri standard

Noţiunea de fişier în limbajul C/C++ este privită într-un mod mai larg. Se poate asocia un stream unui fişier prin operaţia de deschidere, urmând apoi a fi transmise datele între program şi celălalt capăt al conexiunii logice. La lansarea unei aplicaţii C/C++ automat devin active mai multe fluxuri standard şi sunt închise automat la încheierea execuţiei. Acestea sunt:

- *stdin:* intrare standard; stream de tip text;
- *stdout:* ieşire standard; stream de tip text;
- *stderr:* iesire standard erori; stream de tip text;

După atingerea scopului pentru care a fost folosit fişierul respectiv, urmează eliberarea conexiunii logice(si a resurselor alocate) prin efectuarea operaţiei de închidere. În general,

înaintea închiderii propriu-zise se goleşte stream-ul asociat (flushing) pentru a fi siguri că nu rămân date blocate în bufferul de disc asociat conexiunii. În cazul în care nu se realizează închiderea explicită prin program a unui fişier, el este închis automat în momentul terminării execuţiei programului care a deschis respectivul fişier. Dacă execuţia programului este terminată prematur, există posibilitatea ca stream-urile respective să ramana deschise.

Funcţii de intrare-ieşire la nivel consolă- definite în <*conio.h*> (lucreaza fara buffer). Aceste functii nu mai sunt recunoscute de majoritatea compilatoarelor C/C++ moderne ce nu mai permit accesul direct de la consola.

- *int getch(void);* preia un caracter de la tastatură; //*VC++ int _getch( ) ;*
- *int getche(void);* la fel ca *getch( )*, dar îl şi afişează pe ecran (preluare cu ecou); //*VC++ int _getche( ) ;*
- *int putch(int caracter)*; scrie caracterul indicat pe ecran, valoarea returnată fiind fie caracterul de afişat (în caz de succes), fie caracterul EOF în caz de eşec. Codul ASCII al EOF e definit în <*stdio.h*> ca valoare -1, care in C2, pe un octet, este *0xFF*.

Funcţii standard de intrare-ieşire- definite în <*stdio.h*> (lucreaza cu buffer)

- *char* gets(char* sir)*; extrage un şir de caractere din *stdin* până la întâlnirea caracterului LF '\n', pe care îl depune în şirul de octeti *sir*, urmat de '\0'. In acest mod se pot citi siruri de caractere care pot contine si spatii. Functia returnează adresa şirului citit în caz de succes sau un pointer NULL în caz contrar;
- *int puts(const char* sir);* trimite şirul de octeti *sir* la fluxul *stdout* şi returnează o valoare non-negativă în caz de succes şi EOF în caz contrar;
- *int getchar(void);* preia un caracter din fluxul *stdin*, în caz de succes returnează caracterul citit sau EOF în caz contrar; (e o macrofunctie)
- *int putchar(int caracter);* insereaza caracterul de afişat în fluxul de ieşire *stdout* si returnează caracterul inserat în caz de succes sau EOF în caz contrar. (macrofunctie)

Platforma Microsoft Visual Studio compilatoarele *VC++1y/2z* ofera functia *gets_s()* ce permite optimizarea/securizarea introducerii sirurilor de caractere. Aceasta functie foloseste un parametru suplimentar, lungimea sirului. La citiri combinate de siruri de caractere si alte tipuri de date e necesar a ignora ultimul caracter din buffer ce se face apeland *getchar()* sau *fflush(stdin)* sau *cin.get()* functie de context;

Tratarea fişierelor la nivel inferior

Acest mod de abordare a fişierelor este dependentă de sistemul de operare şi este prin consecinţă, mult mai puţin folosită. Funcţiile de lucru la acest nivel sunt descrise în fişierele header: <*io.h*>, <*stat.h*>, <*fcntl.h*>.
Principalele funcţii de la acest nivel sunt:

- *int open(const char* path, int access [, unsigned mode]);* unde *path* reprezintă calea spre fişier, iar parametrul *access* specifica unul din modurile de deschidere *r, w, r+w, a*, etc. Returnează un descriptor de fişier, *file handle*, argument pt. *read(), write()*

- *int creat(const char *path, int amode);* unde *path* reprezintă calea spre noul fişier ce va fi creat în unul din modurile date de *amode*;
  Exemplu: *handle = creat("FILENAME.EXT", S_IREAD | S_IWRITE);*

- *int read(int handle, void* buf, unsigned len);* se încearcă citirea a *len* octeţi în bufferul de memorie *buf* din fişierul asociat cu *handle*;

- *int write(int handle, void *buf, unsigned len);* semnificaţia variabilelor este aceeaşi ca şi în cazul de mai sus ;

- *long lseek(int handle, long offset, int fromwhere);* poziţionează cursorul în fişierul *handle* la distanţa *offset* octeţi, începând de la *fromwhere*. Ultima variabilă are aceleaşi posibile valori ca şi în cazul funcţiei *fseek()*;

- *int close(int handle);* închide fişierul descris de *handle* şi returnează –1 în caz de eroare sau 0 dacă operaţia s-a efectuat cu succes.
  Exemplu : *close(handle);*

Tratarea fişierelor la nivel superior

Pentru ca funcţiile specifice lucrului cu fişiere să fie recunoscute în program, trebuie inclus fişierul header <*stdio.h*>. Acesta asigură prototipurile pentru funcţiile de I/O şi defineşte următoarele tipuri de date: *size_t, fpos_t* şi *FILE*. Primele două reprezintă varietăţi de întreg fără semn, iar tipul FILE este o structura avand ca elemente atributele fisierului.

Macrourile pentru lucrul cu fişiere, definite de *stdio.h*, sunt următoarele:

- **NULL**, defineşte un pointer nul;
- **EOF**, este definit în general ca fiind –1 şi reprezintă valoarea returnată când o funcţie încearcă să citească peste sfârşitul fişierului;
- **FOPEN_MAX** defineşte numărul maxim de fişiere care pot fi deschise simultan;
- **SEEK_SET, SEEK_CUR, SEEK_END** sunt folosite împreună cu funcţia *fseek()* şi ajută la poziţionarea în fişier la început (SEEK_SET), la poziţia curentă (SEEK_CUR) sau la sfârşit (SEEK_END).

Pointer-ul fişierului este legătura dintre fişier şi sistemul de I/O definit de standardul ANSI C. Prin intermediul acestuia se vehiculează toate informaţiile în- şi dinspre fişier.

Exemplu: *FILE* fp;*

Funcţii standard C/C++ specifice lucrului cu fişiere în mod text:

Nume funcţie	Efect
fopen()	Deschide un fişier
fclose()	Închide un fişier
fputc()	Scrie un caracter în fişier
fgetc()	Citeşte un caracter din fişier
fputs()	Scrie un şir de caractere în fişier
fgets()	Citeşte un şir de caractere din fişier
fseek()	Poziţionează cursorul la un anumit octet în fişier
fprintf()	Scrie date formatate în fişier
fscanf()	Citeşte date formatate din fişier
ftell()	Permite citirea indicatorului de poziţie
fgetpos()	Citeşte valoarea poziţiei curente
fsetpos()	Seteaza valoarea indicatorului de pozitie
feof()	Returnează *true* dacă se ajunge la sfârşitul fişierului
ferror()	Returnează *true* dacă a apărut o eroare
rewind()	Readuce indicatorul de poziţie la începutul fişierului
remove()	Şterge un fişier
fflush()	Goleşte un stream asociat unui fişier

Deschiderea unui fişier folosind standardul C/C++
Prototipul funcţiei de deschidere a unui fişier, fopen(), este următorul:

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

unde *file_name* reprezintă numele fişierului care va fi deschis în modul de acces definit de *mod*. În tabelul următor sunt prezentate modurile de deschidere ale unui fişier:

Mod de deschidere	***Semnificaţie***
r	Deschide un fişier text pentru citire
w	Deschide un fişier text pentru scriere
a	Adaugă într-un fişier text
rb	Deschide un fişier binar pentru citire
wb	Deschide un fişier binar pentru scriere
ab	Adaugă într-un fişier binar
r+	Deschide un fişier text pentru citire/scriere (nu creaza daca nu exista, nu distruge daca exista)
w+	Crează un fişier text pentru citire/scriere (creaza daca nu exista, distruge daca exista)
a+	Adaugă sau crează un fişier text pentru citire/scriere (creaza daca nu exista, adauga la sfarsit daca exista)
r+b	Deschide un fişier binar pentru citire/scriere
w+b	Crează un fişier binar pentru citire/scriere
a+b	Adaugă sau crează un fişier binar pentru citire/scriere

Închiderea unui fişier se realizează prin apelul funcţiei

int fclose(FILE fp);*

şi are ca efect închiderea fluxului deschis în prealabil către fişierul respectiv.

Operaţia de închidere trebuie efectuată numai într-un context adecvat (după ce toate procesele de citire/scriere au fost încheiate), pentru că în caz contrar, pot apărea pierderi de date din fişier, poate fi distrus fişierul sau chiar pierdut.

După închiderea unui fişier, se eliberează blocul de control al acestuia, fiind disponibil pentru a fi reutilizat.

Platforma Visual Studio, compilatoarele *Visual C++1y/2z* au introdus o alta sintaxa pentru functia de deschidere a unui fisier *fopen_s();*
http://msdn.microsoft.com/en-us/library/z5hh6ee9.aspx

errno_t fopen_s(FILE** pFile, const char *filename, const char *mode);

-*pFile* –e un pointer catre file pointer-ul ce va primi pointer-ul fisierului deschis.
-*filename* – nume fisier
-*mode* –tip-ul de acces permis care permite si considerarea de fisiere UNICODE
-*errno*, flag eroare (*int*), daca e = 0, e OK, daca nu, eroare deschidere

Scrierea în fişier se realizează prin intermediul următoarelor funcţii in mod standard:

- *int fputc(int c, FILE* stream);*
  Scrie caracterul ***c*** în fişierul identificat de pointerul *stream.*
  Funcţia returnează caracterul *c* în caz de reuşită sau EOF în caz contrar.
- *int fputs(const char* s, FILE* stream);*
  Scrie şirul pointat de *s* în fişierul *stream*. Returnează o valoare pozitivă în caz de reuşită sau EOF în caz contrar.
- *int fprintf(FILE* stream, const char* format[, argument, ...]);*
  Are parametrii similari cu *printf()*, singura deosebire fiind apariţia pointer-ului *stream* la fişierul unde va avea loc scrierea. Returnează numărul de octeţi scrişi sau, în caz de eşec,

EOF. În specificatorul de format trebuie folosite şi caracterele de formatare dorite cum ar fi '\n'.

Citirea din fişier are drept suport următoarele funcţii in mod standard:

- *int fgetc(FILE* stream);* citeşte un caracter de la poziţia curentă din fişierul indicat de *stream*. În caz de succes, returnează caracterul citit, altfel EOF.
- *char* fgets(char* s, int n, FILE* stream);* depune in în şirul *s* un şir de caractere de lungime *n* citit din fişierul *stream*. Returnează şirul în caz de succes sau NULL în caz de sfârşit prematur de fişier sau în caz de eroare.
- *int fscanf(FILE* stream, const char* format[, address, ...]);* citeşte din fişierul *stream* date formatate, analog cu funcţia *scanf()*.

Platforma Microsoft Visual Studio compilatoarele *VC++1y/2z* ofera functia *fscanf_s(),* ca un mecanism de securitate/optimizare in cadrul platformei. Folosind directiva:

#define _CRT_SECURE_NO_WARNINGS

se pot folosi functiile standard existente in C/C++ cu fisiere.

Funcţiile *feof()* şi *ferror()*

- *int feof(FILE* fp);* verifică in structura FILE asociata fisierului la deschidere, dacă la accesul anterior s-a atins sau nu sfârşitul de fişier. Returnează o valoare diferită de zero dacă s-a ajuns la sfârşitul fişierului sau zero in caz contrar.
- *int ferror(FILE* fp);* returnează o valoare pozitivă dacă s-a produs o eroare la citire/scriere sau 0 în caz de operaţie reuşită.

Stream-uri binare

Un stream binar este o secvenţă de octeţi aflată într-o corespondenţă biunivocă cu cei de la echipamentul extern. La nivele mai jos aceste secvente de octeţi sunt completate ("padding"), sau segmentate, pentru a aduce pachetul de date la o lungime standard cerută de protocolul de comunicare, (de exemplu un disc are 512 octeţi/sector sau un cadru IP transporta 1500 octeti "payload"), dar aceste operatii sunt transparente pentru utilizator. În cazul fluxurilor binare nu apar translatări de date (te tipul 0A <−−−> 0D 0A, cum erau la fluxuri de tip text).

Lucrul cu fişiere în mod binar

În cazul fişierelor binare, informaţia stocată în interiorul acestora nu este lizibilă. Dacă totuşi cineva forţează deschiderea unui fişier binar cu un editor de texte (de ex. *Notepad*), "caracterele" care vor fi afişate pe ecran par un amestec de caractere alfanumerice fără sens, incluzand caractere semigrafice (cod ASCII > 7Fh) şi caractere de control.

Citirea/scrierea fişierelor binare se face cu următoarele două funcţii, declarate în <stdio.h>:

size_t fread(void **ptr**, size_t **size**, size_t **n**, FILE* **stream**);*

size_t fwrite(const void **ptr**, size_t **size**, size_t n, FILE* **stream**);*

ptr -- un pointer către un buffer de memorie care va primi datele din/spre fişier,
size -- dimensiunea în octeţi a unui *articol* de tipul celor care vor fi citite/scrise din/în fişier.
Articol în acest context înseamnă un pachet de octeţi, de o dimensiune ce depinde de aplicaţie. Poate fi de exemplu un element al unui tablou de structuri, numită *înregistrare*(în cazul unei baze de date), un *cadru*(în cazul transmiterii datelor la distanţă) etc.
n -- numărul articolelor care urmează să fie citite/scrise
stream -- specifică fişierul deschis în prealabil, este acel "file-handle" returnat de *fopen()*

Deschiderea şi închiderea fişierelor binare se face cu aceleaşi funcţii ca şi la fişierele de tip text, cu *fopen()/fopen_s()* şi *fclose()*.

Funcţii specifice accesului aleator

Accesul la un octet înregistrat poate fi *secvenţial* (ca de exemplu la "tape drive", cu organizare unidimensională) sau direct, numit *aleator* (de exemplu, prin coordonate cilindru, cap magnetic, pistă, sector, etc.). Pentru sistemul de fişiere conţinutul fişierului e transparent. Programatorul însă va utiliza funcţii diferite pentru a accesa conţinutul fişierelor text sau a celor binare.

- *long int ftell(FILE* fp);*

–Permite citirea *indicatorului de poziţie*, care contine offset-ul octetului de inceput pt. urmatorul acces. Primul octet de date este la offset zero. Indicatorul de pozitie se actualizeaza automat dupa fiecare access.
–Funcţia returnează poziţia curentă din fişierul specificat de pointerul *fp* în caz de reuşită
sau valoarea -1L (valoarea lui EOF, -1, extinsă cu semn pe 4 octeţi, în C2), în caz de eşec

- *int fgetpos(FILE* fp, long int* poz);*

–Citeşte valoarea poziţiei curente şi o înscrie în variabila *poz*
–Returnează valoarea 0 în caz de succès, sau una diferita de zero in caz de eroare, setand variabila globala *errno* la ovaloare pozitiva, care poate fi interpretata cu functia *perror()*

- *int fsetpos(FILE* fp, const long int* poz);*

–Se atribuie valoarea variabilei *poz* la indicatorul de poziţie asociat fişierului indicat de *fp*. Altfel spus pozitioneaza cursorul in fisier.
–Se returnează valoarea 0 în caz de succes, ca si *fgetpos()*.

- *int fseek(FILE* fp, long nr_oct, int origine);*

–*fseek()* deplasează indicatorul de poziţie din poziţia *origine* cu un număr de octeţi *nr_oct* numit şi "offset" (număr cu semn)
–returnează valoarea 0 în caz de reuşită sau o valoare nenulă în caz de eşec
–Poziţia de referinţă "*origine*" poate avea trei valori:
0 = SEEK_SET : început de fişier
1 = SEEK_CUR : poziţia curentă a cursorului
2 = SEEK_END : sfârşit de fişier
unde SEEK_SET etc. sunt constante simbolice de limbaj, definite în <stdio.h>

- *void rewind(FILE *fp);*

–Permite poziţionarea indicatorului de poziţie la începutul fişierului
–Se şterge şi indicatorul de eroare asociat fişierului

Alte funcţii referitoare la fişiere

*FILE * freopen(const char *filename, const char *mode, FILE *stream);*
//Asociază un nou fişier generic la fluxul-sursă deschis (de exemplu redirectează *stdout* la *stdprn* sau către un fişier pe disc)

*int rename(const char *oldname, const char *newname);* // redenumeste un fişier prin program

*void clearerr(FILE *fp);* // sterge indicatorii de eroare si de sfârşit-fişier

3. Exemple:

Fişiere text

Exemplul F1

```
/*Copierea unui fişier caracter cu caracter.

Experimentul 1.
Rulare executabil din IDE, fisierul fis1.txt se creeaza prin program.

Experimentul 2.
Rulare executabil prin IDE, fis1.txt se creeaza manual.

Se decomenteaza liniile de sub titlul "//creare fisier-sursa prin program",
care creeaza fisierul-sursa si scriu continutul "ttt" in el. Se selecteaza in IDE
cele cinici linii de cod-sursa, si <Ctrl><k> apoi <Ctrl><c>

Se sterg fisierele fis1.txt si fis2.txt din subdirectorul proiectului,
unde IDE-ul pastreaza si fiserul sursa.cpp, de exemplu, daca  numele proiectului
este L13, atunci calea la acest subdirector va fi
C:\...\L13\L13

Se creeaza manual un nou fisier fis1.txt din linia de c-da. Pentru aceasta se
navigheaza cu File-manager-ul lui Windows in directorul C:\...\L13\L13
se da <Shift><Clic-dreapta> in zona libera a ferestrei si din meniul-context se
alege "Open PowerShell window here". La prompt se tasteaza cmd<Enter>, care
lanseaza o noua instanta a Command Prompt in fereastra curenta.

Se vor introduce cateva comenzi DOS, ca mai jos.
PowerShell nu mai cunoaste denumirea DOS pt. fluxul CON: (consola)

C:\...\L13\L13>copy CON: fis1.txt
qwertyuio
12345678^Z
C:\...\L13\L13>dir fis*
C:\...\L13\L13>type fis1.txt

Dupa rularea acestui program din IDE, cu <Ctrl><F5>,
se va verifica, cu comanda dir de mai sus,
daca a aparut in directorul curent si fisierul-text fis2.txt
Se poate afisa continutul lui cu c-da type.
Putem verifica si identitatea continuturilor cu c-da DOS "file compare", fc:
C:\...\L13\L13>fc fis1.txt fis2.txt

PowerShell ofera aceste functionalitati DOS, dar in alt fel
PS C:\...\L13\L13> "qwertyuio
>> 12345678" | out-file fis1.txt
si
PS C:\...\L13\L13> Compare-object (get-content fis1.txt) (get-content fis2.txt)

Experimentul 3.
Rulare manuala executabil din linia de c-da
In acest caz se deshide un terminal PowerShell in directorul
C:\...\L13\Debug
Fisierul fis1.txt se creeaza fie prin program,
fie manual, urmand comenzile de la experimentul 2
Se lanseaza executabilului din linia de c-da
PS C:\...\L13\Debug\L13.exe
De data aceasta executabilul nostru va crea fis2.txt in directorul lui curent,
C:\...\L13\Debug
Dupa terminarea experimentelor se inchide fereastra-terminal cu comanda "exit"

OBS.
MS VS creeaza fisierul .exe cu numele proiectului, adica "nume_proiect.exe"
Daca se ruleaza acest executabil din IDE,
el va cauta fis1.txt si va crea fis2.txt
in directorul unde e si sursa *.cpp
```

```
daca insa se lanseaza  manual executabilul,
din linia de c-da in terminalul-DOS deschis in dir.-ul Debug,
unde compilatorul il depune,
el va cauta fis1.txt si va crea fis2.txt in acel director.
 */

#include <stdio.h>
#include <stdlib.h>
//fopen_s – incepand din VS 2010

int main(void){
	FILE *fps, *fpd;
	char c;
	char fis_s[] = "fis1.txt", fis_d[] = "fis2.txt";//current directory
	errno_t err;

	// creare fisier-sursa prin program
	err=fopen_s(&fps, fis_s, "w");
	if(err!=0){puts("Eroare la creare fisierului sursa !");
	exit(1); }
	fprintf(fps,"ttt");
	fclose(fps);

	err=fopen_s(&fps, fis_s, "r");
	if(err!=0){
		//fis1.txt va fi in directorul unde se afla fisierul sursa *.cpp
		printf("Eroare la deschiderea fisierului sursa %s\n", fis_s);
		exit(1);}

	// deschidere fisier destinatie
	err=fopen_s(&fpd, fis_d, "w");
	if(err!=0) { puts("Eroare la deschiderea fisierului destinatie !");
	exit(1);
	}
	while((c = fgetc(fps)) != EOF){ //citire caractere din fisierul sursa *.cpp
= getc(fps) e la fel
		fputc(c, fpd); // scriere caractere in fisierul destinatie// putc(c,
fpd) e la fel
		putc(c, stdout); // in paralel se afiseaza caracterele copiate si in
fer.-terminal.
		//fputc(c, stdout); e la fel
	}
	// inchidere fisiere
	fclose(fps);
	fclose(fpd);
	printf("\nCopiere terminata.\n\
Fisierul destinatie, %s, e salvat in directorul curent\n",fis_d);
	return 0;
}
/* rulare Experimentul 1
ttt
Copiere terminata.
Fisierul destinatie, fis2.txt, e salvat in directorul curent
Press any key to continue . . .
*/

//Varianta C/C++ standard (se foloseste fopen() in loc de fopen_s()
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
int main(void) {
	FILE* fps, * fpd;
	char c;
	char fis_s[] = "fis1.txt", fis_d[] = "fis2.txt";
	if ((fps = fopen(fis_s, "r")) == NULL) {
```

```
	//fis1.txt este in directorul unde se afla fisierul sursa *.cpp.
//Se poate crea in proiectul VC++ cu Source Files>Add->New Item->Utility->TextFile
		puts("Eroare la deschiderea fisierului sursa !");
		exit(1);
	}
	// deschidere fisier destinatie
	if ((fpd = fopen(fis_d, "w")) == NULL) {
		puts("Eroare la deschiderea fisierului destinatie !");
		exit(1);
	}

	while ((c = getc(fps))! = EOF) // citire caractere din fisierul sursa
		putc(c, fpd); // scriere caractere in fisierul destinatie

	 // inchidere fisiere
	fclose(fps);
	fclose(fpd);
	puts("Copiere terminata in fis2.txt");
	return 0;
}
```

Exemplul F2

```
// citirea fisierului in bucla infinita, cu iesire la atingerea sfarsitului de
fisier. Flag-ul EOF din structura pointata de fp e actualizat doar dupa o tentativa
de acces, de exemplu cu fgetc()
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

int main() {
	FILE* fp;
	int c;
	fp = fopen("file.txt", "r");
//file.txt se creaaza in prealabil in acelasi director cu fis. sursa *.cpp
	if (fp == NULL) {
		perror("Error in opening 'file.txt'"); // afiseaza si motivul erorii
		exit(1); }
	while (1) {
		c = fgetc(fp);
		if (feof(fp))
			break;
		printf("%c", c);
	}
	fclose(fp);
	return 0;
}//main
```

Exemplul F3

```
/*Copierea unui fișier linie cu linie. Cale absoluta si relativa.
Acest fisier-sursa sa-l adaugati la proiect ca "source.cpp"
Se creeaza si un mic fisier text, tot Source.cpp in directorul e.g. CPP de pe
discul E:
(sau un alt disc/director la care aveti drept de scriere de pe calculatorul
propriu.)
Pentru a-i vedea continutul, dati pe el <Clic-dreapta> si selectati din meniu
<Edit>
Incercati si varianta in care specificam pentru numele fisierului sursa doar
„source.cpp”,
caz in care in "fis3.txt” ar trebui sa regasiti o copie al acestui fiser-
sursa*/

#include <stdio.h>
#include <stdlib.h>
#define		DIML	81
```

```
int main(void) {
    FILE* fps, * fpd;
    errno_t err;
    char buf[DIML], * p;
    char fiss[] = "E:/CPP/Source.cpp", fisd[] = "E:/CPP/fis3.txt";
       // se va urmari aparitia noului fis. pe drive-ul E:, dir CPP

    //char fiss[] = "Source.cpp", fisd[] = "E:/CPP/fis3.txt";
       // putem observa schimbarea lungimii si a continutului lui fis3.txt

    // deschidere fisier sursa
    if ((err = fopen_s(&fps, fiss, "r") != 0)) {
        puts("Eroare la deschiderea fisierului sursa !");
        exit(1);
    }

    // deschidere fisier destinatie
    if ((err = fopen_s(&fpd, fisd, "w") != 0)) {
        puts("Eroare la deschiderea fisierului destinatie !");
        exit(1);
    }

    //citire date din fisierul sursa
    while((p = fgets(buf, DIML, fps)) != NULL)//prioritate operatori:!= 6, = 13
              fputs(buf, fpd);// scriere date destinatie

    // inchidere fisiere
    fclose(fpd);
    fclose(fps);
    puts("Copiere terminata");
    return 0;
}
```

Exemplul F4

```
/* Exemplu fprintf() -- "Write formatted data to stream" si
 fscanf_s() cu _countof() */
#include <stdio.h>
#include <stdlib.h>
int main() {
       char str[80];
       float f;
       FILE* pFile;
       int err;
       err = fopen_s(&pFile, "myfile.txt", "w");
       if (err != 0) {
              perror("\nError to create the file 'myfile.txt'");
              exit(1);//Error to create the file 'myfile.txt': Permission denied
       }
       fprintf(pFile, "%f %s", 3.1416f, "PI-Omega");
       fclose(pFile);

       err = fopen_s(&pFile, "myfile.txt", "r");
       if (err == 0) {
              fscanf_s(pFile, "%f", &f);
              fscanf_s(pFile, "%s", str, _countof(str));
              fclose(pFile);
              printf("I have read: %g and %s \n", f, str);
              return 0;
       }
       else {
              puts("\nError to open the file 'myfile.txt'");
              exit(1); }
```

```
}
/* rulare
I have read: 3.1416 and PI-Omega
Press any key to continue . . .
*/
```

Exemplul F5

```
/* Aplicatie care citeste valorile intregi din fisierul test.txt, creeat "manual",
afiseaza radacina patrata extrasa din cele pozitive,
si adauga la sfarsitul fisierului numarul valorilor pozitive gasite in fisier*/

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main() {
	FILE* fp;
	int x, contor = 0;
	fp = fopen("test.txt", "r");
	if (!fp) {
		perror("\n EROARE la deschiderea fisierului...");
		exit(1);
	}
	printf("\n Extragem radical din valorile pozitive citite din fisier:\n");
	while (fscanf(fp, "%d", &x) != EOF)
		if (x > 0) {
			contor++;
	printf("valoare citita= %d, radacina patrata=%.2lf\n", x, sqrt((float)x));
		}
	fclose(fp);
	fp = fopen("test.txt", "a");
	if (!fp) {
	printf("\n EROARE la deschiderea fisierului in vederea adaugarii...\n");
	exit(1);
	}
	fprintf(fp, " %d", contor);
	fclose(fp);
	return 0;
}//main
/*
Pentru continutul lui test.txt
3 -2
-9 5 -1
programul afiseaza:

 Extragem radical din valorile pozitive citite din fisier:
valoare citita= 3, radacina patrata=1.73
valoare citita= 5, radacina patrata=2.24

E:\...\L13ex5\Debug\L13ex5.exe (process 9276) exited with code 0.
Press any key to close this window . . .
*/
```

Exemplul F6

```
// Program care genereaza numere aleatoare, le scrie într-un fisier binar,
// apoi le citeste din fisier si le  afiseaza pe ecran.
// Numele fisierului si numarul valorilor de generat se introduc de la tastatura.

//#define _CRT_SECURE_NO_WARNINGS  //pt. C standard, fopen()
#include <stdio.h>
#include <stdlib.h>
#include <time.h> // functii de biblioteca din limbaj pt. ceasul in timp real
```

```
FILE* generare();// functia creaza un fisier binar, scrie numerele in el, lasa
fisierul deschis. Returneaza un descriptor de fisier, numit si “file-handle”

int main() {
	FILE* f1;
	int buf1; // un intreg simplu, buffer pt. afisare cu fread()
	f1 = generare();/* Revenind din functie indicatorul de pozitie sta pe
urmatorul octet de scris. Aici va pune fclose() un EOF, daca nu se mai scrie nimic
in acest fisier. Daca fisierul are N octeti, offsetul acestui octet = N (primul
fiind la offset zero).  Deci ftell() ne returneaza chiar lungimea fisierului.*/
	printf("Lungimea fisierului binar = %d[octeti]\n", ftell(f1));

	// Urmeaza sa afisam continutul fisierului
	fseek(f1, 0L, SEEK_SET);  // aducem pointerul de fisier pe primul octet.
	printf("Valorile scrise in fisier sunt: ");
	/*while (1) {
		fread(&buf1, sizeof(int), 1, f1);
		if (feof(fp))break;
		printf("%4d", buf1);
	}
	*/
	while (fread(&buf1, sizeof(int), 1, f1) == 1)
		printf("%4d", buf1);

	fclose(f1);
	puts("\n...fisierul e inchis, apasti o tasta pt. a termina.\n");
	return 0;
}

FILE* generare(){ // returnam identificatorul pentru fisierul creat in functie
	int n, i;
	char sp[10];  // tablou de caractere, numele fisierului de creat
	int buf2;  // loc de manevra pt. un singur numar intreg, de scris in fisier
	FILE* pf;  // declaram fp un pointer la structura de control-stream tip FILE
	int OFFSET = 1;
	int RANGE_MAX = 100;

	printf("Numele fisierului binar(!max.10 car.)<Enter>: ");
	gets_s(sp, _countof(sp));
	printf("Cate valori? : "); // cu un break-point aici vedem cum gets()
//a substituit terminatorul de rand ‘\n’ cu caracterul cu ‘\0’ (octetul null)
	scanf_s("%d", &n);
	errno_t err;
	err = fopen_s(&pf, sp, "w+b");
// Efect mod de acces w+b: daca exista un fis. cu acest nume, va fi suprascris
	if (err != 0) {
		puts("\n N-am putut deschide fisierul.");
		exit(1);
	}
	else {
	    srand((unsigned)time(NULL));
	    for (i = 0; i < n; i++) {// n = cate numere scriem in fisierul binar
	// formatului double ofera 53 biti pt. mantisa si 11 biti pt. exponent.
		buf2 =(int)(((double)rand()/RAND_MAX)*RANGE_MAX + OFFSET);
	//RAND_MAX = constanta predefinita in limbaj
		fwrite(&buf2, sizeof(int), 1, pf);
	    }
	}
	printf("...gata, toate numerele s-au scris in %s, dar fisierul e inca
deschis\n", sp);
	return pf;
}

/*
exemplu de rulare:
```

```
Numele fisierului binar(!max.10 car.)<Enter>: f7.bin
Cate valori? : 7
...gata, toate numerele s-au scris in f7.bin, dar fisierul e inca deschis
Lungimea fisierului binar = 28[octeti]
Valorile scrise in fisier sunt:   11  60  50  88  55  45  45
...fisierul e inchis, apasti o tasta pt. a termina.

E:\...\L13\Debug\L13.exe (process 11696) exited with code 0.
Press any key to close this window . . .
*/
```

Exemplul F7:

```
// Pentru a testa acest program putem utiliza fisierul binar creat cu Exemplul F6
// Programul determina lungimea unui fisier binar,\
folosind functii pt. accesul aleator, fseek(), ftell().
// Numele fisierului se preia prin program, “runtime” .

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

int main() {
	char c[255]; // numele/calea dir. a fisierului de citit, max. 254 caractere
	FILE* fp; //declarare pointer la strutura de control-flux de tip predefinit\
Fluxul specificat de fp va fi conectat la fisierul nostru de catre fopen()

	printf("Introduceti numele fisierului: ");
	gets_s(c); // preia un  sir de caractere de la KBD, le depune in variabila-
tablou c, adaugand automat la sfarsit terminatorul de sir '\0'

	errno_t err;
	if ((err = fopen_s(&fp, c, "rb")) != 0){ //deschidem fis. binar, pt. citire
		printf("\n Fisierul nu poate fi deschis!");
		exit(1);
	}
	//fseek() permite accesul aleator(direct) la orice octet din fisier
	fseek(fp, 0L, SEEK_END); // ne-am pozitionat la sfarsitul fisierului

//ftell() ne returneaza offset-ul pozitiei_curente fata de inceputul fis., [octeti]
printf("Fis. %s are %d octeti,\nverificati cu file-mangerul SO.\n", c, ftell(fp));
	fclose(fp);    //inchidem fisierul
	return 0;
}
/* exemplu de rulare:
Introduceti numele fisierului: f7.bin
Fis. f7.bin are 28 octeti,
verificati cu file-mangerul SO.
...
================================================
Deschidem un terminal PowerShell in subdirectorul-sursa al proiectului
PS E:\...\212\exemple_L13A\exemple_L13A> dir .\f7.bin

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----       12/31/2020   9:19 PM             28 f7.bin

PS E:\...\212\exemple_L13A\exemple_L13A>
*/
```

Exemplul F8:

```
/* Programul concateneaza “n” fisiere binare intr-un nou fisier destinatie,
numele fisierelor sunt preluate prin program. Ca si un program de arhivare.
Pentru a testa acest program utilizam cateva fis.-e binare create cu ex. F6. */
```

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

typedef unsigned char DATE;  //tip definit de utilizator

int main(void) {
	DATE Buff; // pt. stocarea temporara a octetului de transferat
	int nr; // numarul fisierelor de concatenat
	int cont_planif, cont_realiz; // variabile contor pt. fisiere-sursa
	char nmf_s[100], nmf_d[100]; // nume fisier sursa si dest.
	FILE* fs, * fd;   //declararea pointerilor asociati fisierelor
	printf("Numele fisierul destinatie este: "); // f_out.bin
	scanf("%s", nmf_d);
	errno_t err;
	if ((err = fopen_s(&fd, nmf_d, "wb")) != 0) {
		//deschidem fisierul destinatie pentru scriere, in mod binar
		printf("\n Fisierul %s nu poate fi scris !", nmf_d);
		exit(1);
	} // fisierul dest. apare deja si in File Manager, dar cu lungime zero
	printf("Numarul fisierelor binare de concatenat este: ");
	//scanf("%d",&nr);
	fscanf(stdin, "%d", &nr); // echivalenta cu linia precedenta
	for(cont_planif = 0, cont_realiz = 0; cont_planif < nr; cont_planif++){
		printf("Dati numele fisierului sursa #%d: ", cont_planif + 1);
		scanf("%s", nmf_s);
		errno_t err; //desch. fis.-rul sursa actual in mod binar, citire
		if ((err = fopen_s(&fs, nmf_s, "rb")) != 0) {
			printf(":-( Fisierul %s nu poate fi deschis\n", nmf_s);
			continue; // acum cont_realiz nu este incrementat
		}// if()
// luam unul cate unul octetii din fis.-sursa, alipindu-le la fisierul dest.
		while (1) {/*feof() returneaza nonzero doar daca o operatie
anterioara, ca fread() sau fseek() etc., a setat bitul EOF in structura FILE, o
struct. de control al fluxului atasat acestui fisier.*/
			fread(&Buff, sizeof(DATE), 1, fs);
// dupa fread() indicatorul de Pozitie_curenta sta pe octetul urmator de citit.
			if (feof(fs))// acum octetul din pozitia curenta este EOF
				break;		// parasim bucla infinita
			fwrite(&Buff, sizeof(DATE), 1, fd);
		}// while()
		fclose(fs);		//inchidem fisierul sursa actual
		cont_realiz++;			// succes
		printf("pozitia curenta in %s = %d\n", nmf_d, ftell(fd));
 // Afisam lungimea fisierului dest., ca un indicator de progres.
	}// for()
	fclose(fd);   // in sfarsit inchidem si fisierul destinatie
	printf("Am concatenat %d fisiere binare in %s\n", cont_realiz, nmf_d);
	return 0;
}
/*
Iata un exemplu de rulare.

Numele fisierul destinatie este: f_out.bin
Numarul fisierelor binare de concatenat este: 3
Dati numele fisierului sursa #1: f2.bin
pozitia curenta in f_out.bin = 8
Dati numele fisierului sursa #2: f3.bin
pozitia curenta in f_out.bin = 20
Dati numele fisierului sursa #3: f7.bin
pozitia curenta in f_out.bin = 48
Am concatenat 3 fisiere binare in f_out.bin
```

```
E:\...\212\exemple_L13A\Debug\exemple_L13A.exe(process 398) exited with code 0.
Press any key to close this window . . .

Pregatiri:

Facem doua rulari al ex6.exe pt. a obtine inca doua fisiere binare f2.bin si
f3.bin, pt. experimentul cu concatenare (unul il avem deja, f7.bin, de la
rularea initiala a lui ex6.exe)

f2.bin va contine doi intregi(2x4octeti), si f3.bin trei, iar f7.bin are 7,
------------------------------------------------------------------------
Numele fisierului binar(!max.10 car.)<Enter>: f2.bin
Cate valori? : 2
...gata, toate numerele s-au scris in f2.bin, dar fisierul e inca deschis
Lungimea fisierului binar = 8[octeti]
Valorile scrise in fisier sunt:   29  27
...fisierul e inchis, apasti o tasta pt. a inchide fereastra-terminal.
------------------------------------------------------------------------
Numele fisierului binar(!max.10 car.)<Enter>: f3.bin
Cate valori? : 3
...gata, toate numerele s-au scris in f3.bin, dar fisierul e inca deschis
Lungimea fisierului binar = 12[octeti]
Valorile scrise in fisier sunt:   30  12  14
...fisierul e inchis, apasti o tasta pt. a inchide fereastra-terminal.
========================================================
Afisam lungimile fisierelor de concatenat
PS E:\...\212\exemple_L13A\exemple_L13A>  dir *.bin
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----       12/31/2020   9:49 PM              8 f2.bin
-a----       12/31/2020   9:51 PM             12 f3.bin
-a----       12/31/2020   9:19 PM             28 f7.bin
PS E:\...\212\exemple_L13A\exemple_L13A>
============== rulam ex8.exe(vezi mai sus)
Verificam rezultatul prin vizualizarea continutului fisierelor binare.
Afisam continuturi de fisier cu "cmdlet"-ul "format-hex" din Win10PowerShell

PS E:\...\212\exemple_L13A\exemple_L13A> format-hex f2.bin
00000000   1D 00 00 00 1B 00 00 00

PS E:\...\212\exemple_L13A\exemple_L13A> format-hex f3.bin
00000000   1E 00 00 00 0C 00 00 00 0E 00 00 00              ............

PS E:\...\212\exemple_L13A\exemple_L13A> format-hex f7.bin
00000000   0B 00 00 00 3C 00 00 00 32 00 00 00 58 00 00 00  ....<...2...X...
00000010   37 00 00 00 2D 00 00 00 2D 00 00 00              7...-...-...

PS E:\...\212\exemple_L13A\exemple_L13A> format-hex f_out.bin
00000000   1D 00 00 00 1B 00 00 00 1E 00 00 00 0C 00 00 00  ................
00000010   0E 00 00 00 0B 00 00 00 3C 00 00 00 32 00 00 00  ........<...2...
00000020   58 00 00 00 37 00 00 00 2D 00 00 00 2D 00 00 00  X...7...-...-...

PS E:\...\212\exemple_L13A\exemple_L13A>
*/
```

Exemplul F9:

```
/* Scrierea si citirea elementelor unei structuri intr-un fisier.
Se va creea in prealabil un director CPP pe un disc valid de pe calculatorul
propriu */
```

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

typedef struct {
	char nume[10];
	char pren[10];
	int varsta;
} PERS;

int main() {
	int i, np = 3; // np=numar persoane
	FILE* fp;
	PERS tab[3]; // tablou de structuri (Alocare statica, pt. 3 pers.)
	PERS* p_pers; // pointer de tipul structurii
	PERS x; // struct. de manevra pentru afisarea datelor citite din fisier
	//char nm_f[] = "E:/CPP/baza_date.bin";
	char nm_f[] = "baza_date.bin";
	p_pers = tab
	for (i = 0; i < np; i++) {
		printf("\n Numele persoanei %c: ", char('a' + i));
		// in loc de "pers. 1, 2,..." le putem numi "pers. a, b..."
		scanf("%s", tab[i].nume);
		printf("\n Preumele persoanei %c: ", char('a' + i));
		scanf("%s", (p_pers + i)->pren); // in loc de tab[i].pren putem
accesa campul si prin pointer
		printf("\n Varsta persoanei %c: ", char('a' + i));
		scanf("%d", &tab[i].varsta);
	}
	errno_t err;
	err = fopen_s(&fp, nm_f, "wb");//se creeaza si se deschide fis
	if (err != 0) {
		printf("\n EROARE la crearea fisierului...\n");
		exit(1);
	}

	for (i = 0; i < np; i++)
		fwrite(&tab[i], sizeof(PERS), 1, fp);
	// deoarece alocarea tablourilor in memorie este contigua,
	// bucla for(...) de mai sus se poate inlocui cu linia urmatoare:
	// fwrite(tab, sizeof(PERS), np, fp); // un nume de tablou e un
pointer...
	fclose(fp);

	err = fopen_s(&fp, nm_f, "rb");
	if (err != 0) {
		printf("\n EROARE la deschiderea fisierului ...\n");
		exit(1);
	}
	p_pers = &x;
	while (fread(p_pers, sizeof(PERS), 1, fp)) {
		// citim un singur "articol", de marimea structurii noastre
		// if(x.nume[0] == 'A') //  am putea filtra doar anumite
inregistrari
		printf("\n Numele: %s, prenumele: %s\n", x.nume, x.pren);
	}
	fclose(fp);
	return 0;
}//main
/* Pentru curiosi, iata o rulare sub Linux.  -- fopen_s() <-- fopen()
Afisam continutul fisierului binar cu comanda-terminal od, "octal dump"
```

```
# g++ ex9od.cpp
# ./a.out
Numele persoanei a: Num_A
Preumele persoanei a: Pa
Varsta persoanei a: 3

Numele persoanei b: ABC
Preumele persoanei b: abc
Varsta persoanei b: 128

Numele persoanei c: 123
Preumele persoanei c: 456
Varsta persoanei c: 63

 Numele: Num_A, prenumele: Pa
 Numele: ABC, prenumele: abc
 Numele: 123, prenumele: 456
#
# od -atx1 baza_date.bin
0000000   N   u   m   _   A nul nul nul  ht nul   P   a nul nul nul nul
         4e  75  6d  5f  41  00  00  00  09  00  50  61  00  00  00  00
0000020 dle del del del etx nul nul nul   A   B   C nul nul nul nul nul
         90  ff  ff  ff  03  00  00  00  41  42  43  00  00  00  00  00
0000040 soh nul   a   b   c nul nul nul  gs  nl   @ nul nul nul nul nul
         01  00  61  62  63  00  00  00  1d  0a  40  00  80  00  00  00
0000060   1   2   3 nul   $ del nul nul nul nul   4   5   6 nul nul nul
         31  32  33  00  a4  7f  00  00  00  00  34  35  36  00  00  00
0000100   P  ht   @ nul   ? nul nul nul
         d0  09  40  00  3f  00  00  00
0000110
#
OBS.
Se vad frumos intregii (4 octeti, Little-Endian) si terminatorii de sir '\0'
EOF nu apare in listing. Valorile 0x0000000, 0x0000020, ... reprezinta offset-
ul octetilor(indicatorul de pozitie in fisier. Avem 16 octeti pe un rand.)

Daca acelasi fisier binar il deschidem cu un editor de fisiere-text, ca
Notepad.exe sau Wordpad.exe, ele ne afiseaza:
Num_A ÌÌÌÌPa ÌÌÌÌÌÌÌ_   ABC ÌÌÌÌÌÌabc ÌÌÌÌÌÌ€   123 ÌÌÌÌÌÌ456 ÌÌÌÌÌÌ?
*/
```

Exemplul F10:

```
/* Scrierea si citirea elementelor unei structuri intr-un fisier, urmata de
sortare dupa mai multe campuri */

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
const int dim_char = 20;
const int np = 3;//np=numar persoane
int comp_nume_pren_varsta(const void* a, const void* b);

struct Pers{
	char nume[dim_char];
	char pren[dim_char];
	int varsta;
};

int main() {
	int i;
```

```
	FILE* fp, *fps;
	Pers tab[np]; // tablou de structuri (Alocare statica, pt. np pers.)
	Pers* p_pers; // pointer de tipul structurii
	Pers x; // struct. de manevra pentru datele din fisier
	char nm_f[] = "baza_date.bin"; //director curent
	char nm_f_s[] = "sort_baza_date.bin";
	p_pers = tab;
	printf("\nIntrodu date pentru %d persoane", np);
	for (i = 0; i < np; i++) {
		printf("\n Numele persoanei %c: ", char('a' + i));
		// in loc de "pers. 1, 2,..." le putem numi "pers. a, b..."
		scanf("%s", tab[i].nume);
		printf("\n Preumele persoanei %c: ", char('a' + i));
		scanf("%s", (p_pers + i)->pren); //in loc de tab[i].pren, accesam
prin pointerul p_pers
		printf("\n Varsta persoanei %c: ", char('a' + i));
		scanf("%d", &tab[i].varsta);
	}
	errno_t err;
	err = fopen_s(&fp, nm_f, "wb");//se creeaza si se deschide baza_date.bin
	if (err != 0) {printf("\n EROARE la crearea fisierului...\n");
		exit(1);}
	for (i = 0; i < np; i++)
		fwrite(&tab[i], sizeof(Pers), 1, fp);//articol cu articol
	fclose(fp);
	err = fopen_s(&fp, nm_f, "rb");
	if (err != 0) {printf("\n EROARE la deschiderea fisierului ...\n");
		exit(1);}
	p_pers = &x;
	printf("\nFisierul cu datele initiale este:\n");
	while (fread(p_pers, sizeof(Pers), 1, fp)) {
	printf("\n Numele: %s, prenumele: %s, varsta: %d\n", x.nume, x.pren,
x.varsta);//acces prin x
	}
	fclose(fp);

	err = fopen_s(&fps, nm_f_s, "wb");// sort_baza_date.bin
	if (err != 0) {printf("\n EROARE la crearea fisierului...\n");
		exit(1);}
	printf("\n\nFisierul cu datele sortate este:\n");
	qsort(tab, np, sizeof(Pers), comp_nume_pren_varsta);
	fwrite(tab, sizeof(Pers), np, fps);//scriere intreg tabloul sortat de
dim np
	fclose(fps);
	err = fopen_s(&fps, nm_f_s, "rb");
	if (err != 0) {printf("\n EROARE la deschiderea fisierului ...\n");
		exit(1);}
	while (fread(p_pers, sizeof(Pers), 1, fp)) {
	printf("\n Numele: %s, prenumele: %s, varsta: %d\n", p_pers->nume,
p_pers->pren, p_pers->varsta);//acces prin p_pers
	}
	fclose(fps);
	return 0;
}//main

//sortare dupa nume, prenume si varsta
int comp_nume_pren_varsta(const void* a, const void* b) {
	int fl_nume, fl_pre;
	Pers* pa = (Pers*)a;
	Pers* pb = (Pers*)b;
	if ((fl_nume = strcmp(pa->nume, pb->nume)) == 0)if ((fl_pre = strcmp(pa-
>pren, pb->pren)) == 0)
```

```
        return (pa->varsta - pb->varsta); //ascendent_nume, prenume si varsta
               else return fl_pre;//diferenta prenume
        return fl_nume;//diferenta nume
}
```

4. Întrebări:

- Ce înțelegeți printr-un stream de tip text?
- Ce fluxuri standard se deschid automat la lansarea în execuţie a unui program C/C++?
- Ce funcţii utilizaţi pentru scrierea/citirea unui fişier text, caracter cu caracter?
- Ce funcţii utilizaţi pentru scrierea/citirea unor înregistrări de tip float dintr-un fişier text?
- Ce funcţie utilizaţi pentru citirea unui fişier text linie cu linie?
- Ce este un stream binar?
- Care sunt funcţiile utilizate pentru citirea/scrierea fişierelor binare?
- Cum se realizează accesul aleator la fişiere?

5. Temă:

1. Să se scrie un program care citeşte şi apoi afişează date întregi preluate dintr-un fişier text al cărui nume este citit de la consolă. Creati in prealabil fisierul prin program.
2. Să se scrie un program care citeşte dintr-un fişier text 10 numere întregi (generat in prealabil prin program sau extern). Să se scrie funcţiile care:
   a. aranjează crescător/descrescator şirul si afiseaza rezultatul;
   b. numără câte elemente sunt pare si afiseaza rezultatul.
   Adăugaţi în fişierul original noile rezultate obţinute.
3. Scrieţi un program care citeşte de la consolă *n* numere întregi pe care le scrie într-un fişier text cu numele citit de la tastatură. Citiţi apoi numerele din fişier, determinaţi media lor aritmetică, pe care o adăugaţi la sfârşitul fişierului şi o afişaţi şi pe ecran.
4. Scrieţi un program C/C++ care citeşte de la tastatură un caracter apoi scrie acest caracter într-un fişier text pe *n* linii, câte *n* caractere pe fiecare linie, *n* citit de la consola.
5. Să se scrie o aplicaţie C/C++ care citeşte un fişier text linie cu linie şi îl afişează pe ecran. Se va folosi un fisier existent din sistem sau se va genera in prealabil unul prin program.
6. Scrieţi un program C/C++ care citeşte de la tastatură valori reale în format float, cu confirmare. Valorile citite vor fi scrise într-un fişier text cu numele citit din linia de comandă. Citiţi apoi fişierul şi afişati valorile mai mari decât o valoare dată, citită de la tastatură.
7. Scrieţi o aplicaţie C/C++ care citeşte caracter cu caracter un fişier text şi converteşte primul caracter al fiecărui cuvânt în majusculă.
8. Scrieţi un program care citeşte valori reale dintr-un fişier creat în prealabil si scrie într-un alt fişier partea întreagă a numerelor pozitive citite.
9. Să se scrie o aplicaţie care:
   - citeşte de la consolă un numar întreg n;
   - citeşte de la consolă, cu o funcţie, "n" numere reale, într-un tablou unidimensional, alocat dinamic în memorie;
   - scrie aceste valori din tablou într-un fişier binar, al cărui nume este citit tot de la consolă;
   - citeşte apoi conţinutul fişierului şi afişează numerele din 3 în 3 poziţii, folosind funcţii specifice accesului aleator la fişiere.
10. Să se scrie o aplicaţie care:
    - defineşte o structură numită Student, cu câmpurile numele, prenumele, grupa, media;
    - citeşte de la consolă un număr întreg n (numărul studentilor)

- pentru fiecare înregistrare de tip student, citeşte cu o funcţie datele aferente şi le scrie într-un fişier, cu numele preluat de la consolă;
- citeşte conţinutul fişierului şi afişează studenţii ce au media mai mare decât o valoare citită de la consolă.

11. Să se scrie o aplicaţie care:
- citeşte de la consolă un numar întreg n
- citeşte de la consolă, cu o funcţie, n numere întregi, memorându-le într-un tablou unidimensional, alocat dinamic
- scrie valorile din acest tablou într-un fişier binar, al cărui nume este citit de la consolă;
- citeşte conţinutul fişierului şi afişează conţinutul si offsetul poziţilor pe care s-au găsit numere pare.

12. Să se scrie un program, care:
- preia din linia de comanda doua nume de fisier.
- citeşte prin program 8 numere intregi, pe care le scrie in primul fisier, in mod binar.
- citeşte înapoi valorile din acest fisier si calculează media aritmetica a numerelor mai mari decat 4
- scrie rezultatul in al doilea fisier, sub forma: Media aritmetica a numerelor…, este… unde in locul punctelor de suspensie va scrie valorile a caror medie a fost calculată, respectiv valoarea mediei, cu o precizie de 2 zecimale.

13. Definiti o structura prajitura, cu campurile nume, nr_bucati, pret.
- cititi de la tastatura datele pentru un numar n de prajituri si salvati aceste date intr-un fisier binar.
- cititi apoi inregistrările din fişier şi afişati toate informaţiile despre prăjitura cea mai ieftină.

14. Scrieti intr-un fisier binar câteva numere intregi, citite de la tastatura. Citiţi apoi numerele de pe poziţii impare si afişati pentru fiecare, dacă este valoare pară sau impară.

5'. Homework:

1. Write a program that displays the integer elements read from a text file. The filename is entered from the keyboard. The file has to be created in another program or function.
2. Write a program that reads from a text file 10 integer numbers. The file has to be previously created using a different code or by using the operating system's facilities. Write the functions that:
   - order the integers array in ascending/descending order and displays the result
   - count the number of even numbers in the array and display the result

   Write the generated results into the original file.
3. Write a program that reads from the keyboard *n* integer values and then stores them into a text file. The filename has to be read from the keyboard. Then read the numbers from file, calculate their average value, display-it and append-it to the end of the text-file.
4. Write a program that reads from the keyboard a single character and an integer value *n*. Generate a text file that will contain *n* lines and on each line write the character *n* times, *n* being read from the keyboard.
5. Develop a C/C++ application that will display (line by line) the content of a previously created text file then one created by program.
6. Write a C/C++ program that reads from the keyboard (with confirmation) a series of float values. Write the values into a text file that has the name entered from the command line. Read the file's content and display all the values greater than a given number, read from the keyboard.
7. Write a C/C++ application that reads a text file character by character and converts the first letter of each word into its uppercase equivalent.
8. Write a program that reads real values from a previously created file and writes to another file the entire part of the positive numbers read.
9. Write an application that:
   - reads from the keyboard an integer value n;

- reads from the keyboard (using a function) n floating-point numbers, storing them into a dynamically allocated one-dimensional array;
- writes out the floating-point values into a binary file, the filename being read from the keyboard;
- reads back the file's content, displaying the numbers with indexes 0, 3, 6, etc. using the random file-access methods;

10. Write an application that:
- defines a structure called Student, having fields name, surname, group, average mark ;
- reads from the keyboard an integer value n;
- for each student it reads from the keyboard (in a function) the personal data (all fields),
- stores the information for all the n students and into a binary file, the filename being read from the keyboard;
- reads back the file's contents, displaying the data related to the students who have the average mark >= than a specific value given from the keyboard.

11. Write an application that:
- reads from the keyboard an integer value *n*;
- reads from the keyboard, with a function, *n* integers, storing them into a dynami allocated one-dimensional array;
- writes these values into a binary file (the filename is also read from the keyboard);
- reads the file's content and displays the offset and content of all positions where even numbers are found.

12. Write a C/C++ program that reads from the comand line two file names. The program should ask the user to introduce eight integer values fom the KBD, saving them into the first file, in binary mode. Read back these values from the file, and determine the arithmetical media of the values greather than 4. Write the result into the second file in text mode, using the following format: "The average value of ... is ...". The first space needs to be replaced with the values used for calculating the average, the second with the average value itself, using a two digits precision.

13. Define a structure called *cookie* that contains as variables the *name, no_of_pieces* and *price*. Read from the keyboard the data for *n* cookies and save it into a binary file. Read back the file and display the information belonging to the cheapest cookie.

14. Write into a binary file a series of integer numbers read from the keyboard.
Read back the numbers stored in the file on odd positions displaying wether they are odd or even numbers.