

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

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

1. Obiective:

- Scrierea de programe folosind alte date utilizator și fișiere text.
- Înțelegerea modalității de lucru cu fișiere binare
- Înțelegerea accesului aleator în fișiere
- Scrierea de programe folosind fișiere 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/ieșiri separă programatorul de echipament printr-o așa-zisă *abstractizare* a datelor care circulă în permanență între mașina de lucru și program.

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

Stream-uri de tip text

Acest tip de stream este practic o secvență 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 opțională, folosirea sa depinzând de programator și de modul de implementare a aplicației respective. La fișiere text caracterul *LF* este translatat într-o secvență de două caractere, *CR* și *LF (0x0D 0x0A)*. În acest mod nu va exista o corespondență totală între ceea ce este transmis și ceea ce conține fișierul.

Stream-uri standard

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

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

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

înaintea închiderii propriu-zise se golește 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 fișier, el este închis automat în momentul terminării execuției programului care a deschis respectivul fișier. Dacă execuția programului este terminată prematur, există posibilitatea ca stream-urile respective să ramana deschise.

Funcții de intrare-ieșire 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 și afișează pe ecran (preluare cu ecou); //VC++ *int _getche()*;
- *int putch(int character)*; scrie caracterul indicat pe ecran, valoarea returnată fiind fie caracterul de afișat (în caz de succes), fie caracterul EOF în caz de eșec. Codul ASCII al EOF e definit în *<stdio.h>* ca valoare -1, care în C2, pe un octet, este *0xFF*.

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

- *char* gets(char* sir)*; extrage un șir de caractere din *stdin* până la întâlnirea caracterului LF '\n', pe care îl depune în șirul de octeti *sir*, urmat de '\0'. În acest mod se pot citi siruri de caractere care pot contine si spatii. Functia returnează adresa șirului citit în caz de succes sau un pointer NULL în caz contrar;
- *int puts(const char* sir)*; trimite șirul de octeti *sir* la fluxul *stdout* și returnează o valoare non-negativă în caz de succes și 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 character)*; insereaza caracterul de afișat în fluxul de ieșire *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 fișierelor la nivel inferior

Acest mod de abordare a fișierelor este dependentă de sistemul de operare și este prin consecință, mult mai puțin folosită. Funcțiile de lucru la acest nivel sunt descrise în fișierele header: *<io.h>*, *<stat.h>*, *<fcntl.h>*.

Principalele funcții de la acest nivel sunt:

- *int open(const char* path, int access [, unsigned mode])*; unde *path* reprezintă calea spre fișier, iar parametrul *access* specifica unul din modurile de deschidere *r*, *w*, *r+w*, *a*, etc. Returnează un descriptor de fișier, *file handle*, argument pt. *read()*, *write()*
- *int creat(const char *path, int amode)*; unde *path* reprezintă calea spre noul fișier 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* octeți în bufferul de memorie *buf* din fișierul asociat cu *handle*;
- *int write(int handle, void *buf, unsigned len)*; semnificația variabilelor este aceeași ca și în cazul de mai sus ;

- *long lseek(int handle, long offset, int fromwhere);* poziționează cursorul în fișierul *handle* la distanța *offset* octeți, începând de la *fromwhere*. Ultima variabilă are aceleași posibile valori ca și în cazul funcției *fseek()*;
- *int close(int handle);* închide fișierul descris de *handle* și returnează *-1* în caz de eroare sau *0* dacă operația s-a efectuat cu succes.
Exemplu: *close(handle);*

Tratarea fișierelor la nivel superior

Pentru ca funcțiile specifice lucrului cu fișiere să fie recunoscute în program, trebuie inclus fișierul header *<stdio.h>*. Acesta asigură prototipurile pentru funcțiile de I/O și definește următoarele tipuri de date: *size_t*, *fpos_t* și *FILE*. Primele două reprezintă varietăți de întreg fără semn, iar tipul *FILE* este o structura având ca elemente atributele fisierului.

Macrourele pentru lucrul cu fișiere, definite de *stdio.h*, sunt următoarele:

- **NULL**, definește un pointer nul;
- **EOF**, este definit în general ca fiind *-1* și reprezintă valoarea returnată când o funcție încearcă să citească peste sfârșitul fișierului;
- **FOPEN_MAX** definește numărul maxim de fișiere care pot fi deschise simultan;
- **SEEK_SET**, **SEEK_CUR**, **SEEK_END** sunt folosite împreună cu funcția *fseek()* și ajută la poziționarea în fișier la început (**SEEK_SET**), la poziția curentă (**SEEK_CUR**) sau la sfârșit (**SEEK_END**).

Pointer-ul fișierului este legătura dintre fișier și sistemul de I/O definit de standardul ANSI C. Prin intermediul acestuia se vehiculează toate informațiile în- și dinspre fișier.

Exemplu: *FILE* fp;*

Funcții standard C/C++ specifice lucrului cu fișiere în mod text:

Nume funcție	Efect
fopen()	Deschide un fișier
fclose()	Închide un fișier
fputc()	Scrie un caracter în fișier
fgetc()	Citește un caracter din fișier
fputs()	Scrie un șir de caractere în fișier
fgets()	Citește un șir de caractere din fișier
fseek()	Poziționează cursorul la un anumit octet în fișier
fprintf()	Scrie date formate în fișier
fscanf()	Citește date formate din fișier
ftell()	Permite citirea indicatorului de poziție
fgetpos()	Citește valoarea poziției curente
fsetpos()	Setează valoarea indicatorului de poziție
feof()	Returnează <i>true</i> dacă se ajunge la sfârșitul fișierului
ferror()	Returnează <i>true</i> dacă a apărut o eroare
rewind()	Readuce indicatorul de poziție la începutul fișierului
remove()	Șterge un fișier
fflush()	Golește un stream asociat unui fișier

Deschiderea unui fișier folosind standardul C/C++

Prototipul funcției de deschidere a unui fișier, *fopen()*, este următorul:

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

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

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

Închiderea unui fișier se realizează prin apelul funcției

```
int fclose(FILE* fp);
```

și are ca efect închiderea fluxului deschis în prealabil către fișierul respectiv.

Operația 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 fișier, poate fi distrus fișierul sau chiar pierdut.

După închiderea unui fișier, 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 *open_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 fișier se realizează prin intermediul următoarelor funcții in mod standard:

- *int fputc(int c, FILE* stream);*
Scrie caracterul *c* în fișierul identificat de pointerul *stream*.
Funcția returnează caracterul *c* în caz de reușită sau EOF în caz contrar.
- *int fputs(const char* s, FILE* stream);*
Scrie șirul pointat de *s* în fișierul *stream*. Returnează o valoare pozitivă în caz de reușită sau EOF în caz contrar.
- *int fprintf(FILE* stream, const char* format[, argument, ...]);*
Are parametrii similari cu *printf()*, singura deosebire fiind apariția pointer-ului *stream* la fișierul unde va avea loc scrierea. Returnează numărul de octeți scriși sau, în caz de eșec,

EOF. În specificatorul de format trebuie folosite și caracterele de formatare dorite cum ar fi ‘\n’.

Citirea din fișier are drept suport următoarele funcții în mod standard:

- *int fgetc(FILE* stream);* citește un caracter de la poziția curentă din fișierul indicat de *stream*. În caz de succes, returnează caracterul citit, altfel EOF.
- *char* fgets(char* s, int n, FILE* stream);* depune în șirul *s* un șir de caractere de lungime *n* citit din fișierul *stream*. Returnează șirul în caz de succes sau NULL în caz de sfârșit prematur de fișier sau în caz de eroare.
- *int fscanf(FILE* stream, const char* format[, address, ...]);* citește din fișierul *stream* date formate, analog cu funcția *scanf()*.

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

#define _CRT_SECURE_NO_WARNINGS

se pot folosi funcțiile standard existente în C/C++ cu fișiere.

Funcțiile *feof()* și *ferror()*

- *int feof(FILE* fp);* verifică în structura FILE asociată fișierului la deschidere, dacă la accesul anterior s-a atins sau nu sfârșitul de fișier. Returnează o valoare diferită de zero dacă s-a ajuns la sfârșitul fișierului sau zero în caz contrar.
- *int ferror(FILE* fp);* returnează o valoare pozitivă dacă s-a produs o eroare la citire/scriere sau 0 în caz de operație reușită.

Stream-uri binare

Un stream binar este o secvență de octeți aflată într-o corespondență biunivocă cu cei de la echipamentul extern. La nivele mai jos aceste secvențe de octeți 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 octeți/sector sau un cadru IP transporta 1500 octeți “payload”), dar aceste operații sunt transparente pentru utilizator. În cazul fluxurilor binare nu apar translații de date (te tipul 0A <---> 0D 0A, cum erau la fluxuri de tip text).

Lucrul cu fișiere în mod binar

În cazul fișierelor binare, informația stocată în interiorul acestora nu este lizibilă. Dacă totuși cineva forțează deschiderea unui fișier binar cu un editor de texte (de ex. *Notepad*), “caracterele” care vor fi afișate pe ecran par un amestec de caractere alfanumerice fără sens, incluzând caractere semigrafice (cod ASCII > 7Fh) și caractere de control.

Citirea/scrierea fișierelor binare se face cu următoarele două funcții, 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 fișier,

size -- dimensiunea în octeți a unui *articol* de tipul celor care vor fi citite/scrie din/în fișier.

Articol în acest context înseamnă un pachet de octeți, de o dimensiune ce depinde de aplicație. 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 distanță) etc.

n -- numărul articolelor care urmează să fie citite/scrie

stream -- specifică fișierul deschis în prealabil, este acel “file-handle” returnat de *fopen()*

Deschiderea și închiderea fișierelor binare se face cu aceleași funcții ca și la fișierele de tip text, cu *fopen()/fopen_s()* și *fclose()*.

Funcții specifice accesului aleator

Accesul la un octet înregistrat poate fi *secvențial* (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 fișiere conținutul fișierului e transparent. Programatorul însă va utiliza funcții diferite pentru a accesa conținutul fișierelor text sau a celor binare.

- *long int ftell(FILE* fp);*
–Permite citirea *indicatorului de poziție*, care conține offset-ul octetului de început pt. urmatorul acces. Primul octet de date este la offset zero. Indicatorul de pozitie se actualizeaza automat dupa fiecare access.
–Funcția returnează poziția curentă din fișierul specificat de pointerul *fp* în caz de reușită sau valoarea -1L (valoarea lui EOF, -1, extinsă cu semn pe 4 octeți, în C2), în caz de eșec
- *int fgetpos(FILE* fp, long int* poz);*
–Citește valoarea poziției curente și o înscrie în variabila *poz*
–Returnează valoarea 0 în caz de succes, sau una diferita de zero în caz de eroare, setand variabila globala *errno* la o valoare pozitiva, care poate fi interpretata cu functia *perror()*
- *int fsetpos(FILE* fp, const long int* poz);*
–Se atribuie valoarea variabilei *poz* la indicatorul de poziție asociat fișierului indicat de *fp*. Altfel spus poziționeaza cursorul in fisier.
–Se returnează valoarea 0 în caz de succes, ca și *fgetpos()*.
- *int fseek(FILE* fp, long nr_oct, int origine);*
–*fseek()* deplasează indicatorul de poziție din poziția *origine* cu un număr de octeți *nr_oct* numit și “offset” (număr cu semn)
–returnează valoarea 0 în caz de reușită sau o valoare nenulă în caz de eșec
–Poziția de referință “*origine*” poate avea trei valori:
0 = SEEK_SET : început de fișier
1 = SEEK_CUR : poziția curentă a cursorului
2 = SEEK_END : sfârșit de fișier
unde SEEK_SET etc. sunt constante simbolice de limbaj, definite în <stdio.h>
- *void rewind(FILE *fp);*
–Permite poziționarea indicatorului de poziție la începutul fișierului
–Se șterge și indicatorul de eroare asociat fișierului

Alte funcții referitoare la fișiere

*FILE *freopen(const char *filename, const char *mode, FILE *stream);*

//Asociază un nou fișier generic la fluxul-sursă deschis (de exemplu redirecțiază *stdout* la *stdprn* sau către un fișier pe disc)

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

*void clearerr(FILE *fp);* // șterge indicatorii de eroare și de sfârșit-fișier

3. Exemple:

Fișiere text

Exemplul F1

/*Copierea unui fișier caracter cu caracter.

Experimentul 1.

Rulare executabil din IDE, fișierul fis1.txt se creează prin program.

Experimentul 2.

Rulare executabil prin IDE, fis1.txt se creează manual.

Se decommentează liniile de sub titlul "//creare fișier-sursă prin program", care creează fișierul-sursă și scrie conținutul "ttt" în el. Se selectează în IDE cele cinci linii de cod-sursă, și <Ctrl><k> apoi <Ctrl><c>

Se șterg fișierele fis1.txt și fis2.txt din subdirectorul proiectului, unde IDE-ul păstrează și fișierul sursa.cpp, de exemplu, dacă numele proiectului este L13, atunci calea la acest subdirector va fi
C:\...\L13\L13

Se creează manual un nou fișier fis1.txt din linia de comandă. Pentru aceasta se navighează cu File-manager-ul lui Windows în directorul C:\...\L13\L13 se dă <Shift><Click-dreapta> în zona liberă a ferestrei și din meniul-context se alege "Open PowerShell window here". La prompt se tastează cmd<Enter>, care lansează o nouă instanță a Command Prompt în fereastra curentă.

Se vor introduce câteva comenzi DOS, ca mai jos.

PowerShell nu mai cunoaște 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
```

După rularea acestui program din IDE, cu <Ctrl><F5>, se va verifica, cu comanda dir de mai sus, dacă a apărut în directorul curent și fișierul-text fis2.txt. Se poate afișa conținutul lui cu c-da type. Putem verifica și identitatea conținuturilor cu c-da DOS "file compare", fc:
C:\...\L13\L13>fc fis1.txt fis2.txt

PowerShell oferă aceste funcționalități DOS, dar în alt fel
PS C:\...\L13\L13> "qwertyuio
>> 12345678" | out-file fis1.txt
și
PS C:\...\L13\L13> Compare-object (get-content fis1.txt) (get-content fis2.txt)

Experimentul 3.

Rulare manuală executabil din linia de comandă

În acest caz se deschide un terminal PowerShell în directorul

C:\...\L13\Debug

Fișierul fis1.txt se creează fie prin program,

fie manual, urmând comenzile de la experimentul 2

Se lansează executabilul din linia de comandă

PS C:\...\L13\Debug>L13.exe

De data aceasta executabilul nostru va crea fis2.txt în directorul lui curent, C:\...\L13\Debug

După terminarea experimentelor se închide fereastra-terminal cu comanda "exit"

OBS.

MS VS creează fișierul .exe cu numele proiectului, adică "nume_proiect.exe"

Dacă se rulează acest executabil din IDE,

el va căuta fis1.txt și va crea fis2.txt

în directorul unde e și 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 creeaza in prealabil in acelasi director cu fis. sursa *.cpp
    if (fp == NULL) {
        perror("Error in opening 'file.txt'");
        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) {
        perror("\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 %, 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 structura 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 positionat 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. */

```

```

#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

```



```

#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
00000000 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
00000020 dle del del del etx nul nul nul A B C nul nul nul nul
         90 ff ff ff 03 00 00 00 41 42 43 00 00 00 00 00
00000040 soh nul a b c nul nul nul gs nl @ nul nul nul nul
         01 00 61 62 63 00 00 00 1d 0a 40 00 80 00 00 00
00000060 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
00000100 P ht @ nul ? nul nul nul
         d0 09 40 00 3f 00 00 00
00000110
#
OBS.
Se vad frumos intregii (4 octeti, Little-Endian) si terminatorii de sir '\0'
EOF nu apare in listing. Valorile 0x00000000, 0x00000020, ... 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 iiiiPa iiii_ ABC iiiiabc iiii€ 123 iiii456 iiii?
*/

```

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_num_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_num_pren_varsta(const void* a, const void* b) {
    int fl_nume, fl_pren;
    Pers* pa = (Pers*)a;
    Pers* pb = (Pers*)b;
    if ((fl_nume = strcmp(pa->nume, pb->nume)) == 0)if ((fl_pren = strcmp(pa->pren, pb->pren)) == 0)

```

```

    return (pa->varsta - pb->varsta); //ascendent_num, prenume si varsta
        else return fl_pre; //diferenta prenume
    return fl_num; //diferenta nume
}

```

4. Întrebări:

- Ce înțelegeți printr-un stream de tip text?
- Ce fluxuri standard se deschid automat la lansarea în execuție a unui program C/C++?
- Ce funcții utilizați pentru scrierea/citirea unui fișier text, caracter cu caracter?
- Ce funcții utilizați pentru scrierea/citirea unor înregistrări de tip float dintr-un fișier text?
- Ce funcție utilizați pentru citirea unui fișier text linie cu linie?
- Ce este un stream binar?
- Care sunt funcțiile utilizate pentru citirea/scrierea fișierelor binare?
- Cum se realizează accesul aleator la fișiere?

5. Temă:

1. Să se scrie un program care citește și apoi afișează date întregi preluate dintr-un fișier text al cărui nume este citit de la consolă. Creați în prealabil fișierul prin program.
2. Să se scrie un program care citește dintr-un fișier text 10 numere întregi (generat în prealabil prin program sau extern). Să se scrie funcțiile care:
 - a. aranjează crescător/descrescător șirul și afișează rezultatul;
 - b. numără câte elemente sunt și afișează rezultatul.
 Adăugați în fișierul original noile rezultate obținute.
3. Scrieți un program care citește de la consolă n numere întregi pe care le scrie într-un fișier text cu numele citit de la tastatură. Citiți apoi numerele din fișier, determinați media lor aritmetică, pe care o adăugați la sfârșitul fișierului și o afișați și pe ecran.
4. Scrieți un program C/C++ care citește de la tastatură un caracter apoi scrie acest caracter într-un fișier text pe n linii, câte n caractere pe fiecare linie, n citit de la consolă.
5. Să se scrie o aplicație C/C++ care citește un fișier text linie cu linie și îl afișează pe ecran. Se va folosi un fișier existent din sistem sau se va genera în prealabil unul prin program.
6. Scrieți un program C/C++ care citește de la tastatură valori reale în format float, cu confirmare. Valorile citite vor fi scrise într-un fișier text cu numele citit din linia de comandă. Citiți apoi fișierul și afișați valorile mai mari decât o valoare dată, citită de la tastatură.
7. Scrieți o aplicație C/C++ care citește caracter cu caracter un fișier text și convertește primul caracter al fiecărui cuvânt în majusculă.
8. Scrieți un program care citește valori reale dintr-un fișier creat în prealabil și scrie într-un alt fișier partea întreagă a numerelor pozitive citite.
9. Să se scrie o aplicație care:
 - citește de la consolă un număr întreg n ;
 - citește de la consolă, cu o funcție, “ n ” numere reale, într-un tablou unidimensional, alocat dinamic în memorie;
 - scrie aceste valori din tablou într-un fișier binar, al cărui nume este citit tot de la consolă;
 - citește apoi conținutul fișierului și afișează numerele din 3 în 3 poziții, folosind funcții specifice accesului aleator la fișiere.
10. Să se scrie o aplicație care:
 - definește o structură numită Student, cu câmpurile numele, prenumele, grupa, media;
 - citește de la consolă un număr întreg n (numărul studenților)

- pentru fiecare înregistrare de tip student, citește cu o funcție datele aferente și le scrie într-un fișier, cu numele preluat de la consolă;
 - citește conținutul fișierului și afișează studenții ce au media mai mare decât o valoare citită de la consolă.
11. Să se scrie o aplicație care:
 - citește de la consolă un număr întreg n
 - citește de la consolă, cu o funcție, n numere întregi, memorându-le într-un tablou unidimensional, alocat dinamic
 - scrie valorile din acest tablou într-un fișier binar, al cărui nume este citit de la consolă;
 - citește conținutul fișierului și afișează conținutul și offsetul pozițiilor pe care s-au găsit numere pare.
 12. Să se scrie un program, care:
 - preia din linia de comanda doua nume de fisier.
 - citește prin program 8 numere întregi, pe care le scrie în primul fișier, în mod binar.
 - citește înapoi valorile din acest fișier și calculează media aritmetică a numerelor mai mari decât 4
 - scrie rezultatul în al doilea fișier, sub forma: Media aritmetică a numerelor..., este... unde în locul punctelor de suspensie va scrie valorile a caror medie a fost calculată, respectiv valoarea mediei, cu o precizie de 2 zecimale.
 13. Definiți o structură prajitura, cu câmpurile nume, nr_bucati, pret.
 - citiți de la tastatură datele pentru un număr n de prăjituri și salvați aceste date într-un fișier binar.
 - citiți apoi înregistrările din fișier și afișați toate informațiile despre prăjitura cea mai ieftină.
 14. Scrieți într-un fișier binar câteva numere întregi, citite de la tastatură. Citiți apoi numerele de pe poziții impare și afișați 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 \geq 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 dynamically 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 command line two file names. The program should ask the user to introduce eight integer values from 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 greater 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 whether they are odd or even numbers.