Laborator 8
Fișiere. Stream-uri

În acest capitol sunt prezentate consideraţii teoretice și probleme rezolvate privind definirea , clasificarea și utilizarea fișierelor respectiv a stream-urilor.

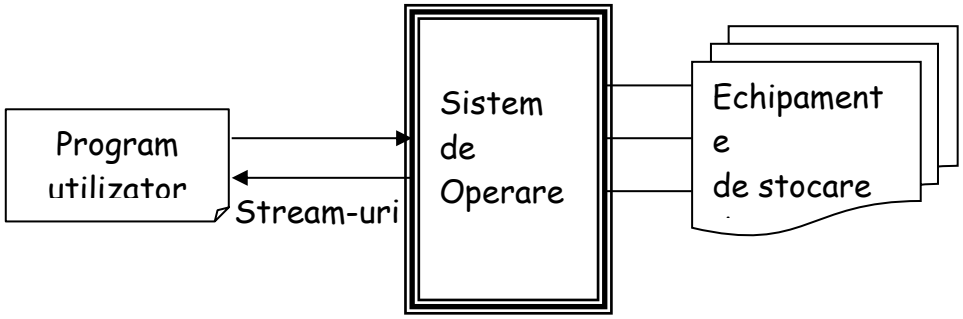
Consideraţii teoretice

Fișierul este o colecție ordonată de înregistrări (articole) aflată pe un suport magnetic (instrument fizic efectiv) în care se stochează datele citite/scrise în programul utilizator. Fișierele în C++ au forma unei structuri numită FILE, care conține informații despre un anumit stream .

Mediile de programare C/C++ conțin funcțiile de I/O destinate manipulării structurii FILE, însă pentru utilizarea ei este necesară declararea unui pointer la această structură. Majoritatea funcțiilor specifice fișierelor sunt conținute în <stdio.h> și au un "f" adăugat în fața numelui . Ex. fscanf(), fprintf(), etc.

Streamul (flux) reprezintă o formă abstractă, independentă de tipul echipamentelor utilizate (terminale, drivere de disc, drivere de unități fizice, etc) care realizează legătura dintre programul utilizatorului și fișierele de I/O utilizate de acesta.

Prin intermediul stream-urilor se realizează operațiile de citire/scriere din/în fișiere. Stream-urile pot fi de tip text sau binare. Stream-urile realizează legătura dintre sistemul de operare și programul utilizator așa cum este ilustrat în figura de mai jos.



Fișierele (și streamurile) se clasifică în două mari categorii:

- **Fișiere (stream-uri) tip text** = secvență de caractere ASCII, organizată pe linii, terminată opțional de caracterul linie nouă. Ex. fiș. cu extensia .txt, .c, etc.
- **Fișiere (stream-uri) binare** = secvență ordonată de caractere (octeti). Ex. fiș. executabile cu extensia .exe, .bat, .com, etc.

Stream-urile standard utilizate în C si C++ sunt

- stdin: streamer de intrare (intrare standard) de tip text. Ex. tastatura
- stdout: streamer de ieșire (ieșire standard) de tip text. Ex.monitorul
- stderr: streamer de ieșire erori (ieșire standard erori) de tip text.

La lansarea în execuție a unui program C/C++ se deschid automat stream-urile specificate mai sus , iar la închiderea programului , acestea sunt închise.

Accesul la informația conținută în fișiere se obține prin intermediul următoarelor tipuri de operații:

- deschiderea fișierului respectiv
- citirea/scrierea informației
- căutarea unei anumite informații, alte operații
- închiderea fișierului

Indicatorul de poziție , la deschiderea fișierului se poziționează pe începutul de fișier, apoi se incrementează/decrementează pe parcursul parcurgerii fișierului iar în final este poziționat pe sfârșitul de fișier.

Modurile de tratare a fișierelor sunt:

a. la nivel inferior (low level):

- se utilizează mai rar pentru că depinde de sistemul de operare
- funcțiile utilizate la acest nivel sunt incluse în <io.h>, <stat.h>, <fcntl.h>. Funcții uzuale: open, creat, read, write, lseek, close

b. la nivel superior (high level):

- se utilizează mai frecvent
- funcțiile utilizate la acest nivel sunt incluse în <stdio.h> pentru C și <fstream.h> pentru C++ . Funcții uzuale: fopen, fclose, fputc, fgetc,fseek, fprintf, fscanf, feof, ferror, rewind, remove, fflush

a) Funcții la nivel inferior:

Funcția **open()** are următorul prototip:

int open(const char *path, int access [,unsigned mode]);

unde **path** = reprezintă calea spre fișier

access = variabila de tip întreg care indică modul de deschidere a fișierului.

Opțiunile și semnificația acestora sunt prezentate în tabelul de mai jos:

access	Semnificatie
O_RDONLY	(numai) Permisiune de **citire** din fișier
O_WRONLY	(numai) Permisiune de **scriere** în fișier
O_RDWR	Permisiune de **citire și scriere** din/în fișier
O_APPEND	Permisiune de **adăugare** în fisier
O_CREAT	Permisiune de **creare** fișier. Dacă fișierul există nu are efect; dacă nu există este creat
O_TRUNC	Permisiune de **trunchiere** fișier . Dacă fișierul există lungimea sa este trunchiată la 0
O_BINARY	Permisiune de **deschidere** a fișierului în mod **binar**
O_TEXT	Permisiune de **deschidere** a fișierului în mod **text**

mode = este un parametru opțional și poate lua una dintre valorile specificate în tabelul de mai jos:

mode	Semnificatie
S_IWRITE	Permisiunea de **scriere** în fișier este autorizată
O_IREAD	Permisiunea de **citire** din fișier este autorizată

S_IREAD|S_IWRITE	Permisiune de **citire** și **scriere** din/în fișier autorizată

Funcția **creat()** are următorul prototip:

int creat(const char *path, int amode);

unde **path** = calea spre noul fișier ce va fi creat

amode = variabila de tip întreg care indică modul de creare a fișierului și poate lua una din valorile specificate în tabelul de mai jos:

amode	Semnificatie
S_IWRITE	Permisiunea de **scriere** în fișier este autorizată
O_IREAD	Permisiunea de **citire** din fișier este autorizată
S_IREAD|S_IWRITE	Permisiune de **citire** și **scriere** din/în fișier autorizată

Funcția **read()** are următorul prototip:

int read(int handle, void *buf, unsigned len);

unde **len** = nr de octeți ce se încearcă a fi citiți în bufferul de memorie **buf** din fișierul asociat cu **handle**

Exemplu:

```
void *buf;int handle, bytes;
...
handle=open("test.exe",O_RDONLY|O_BINARY,S_IWRITE|S_IREAD)
bytes=read(handle,buf,10);
...
```

Funcția **write()** are următorul prototip:

int write(int handle, void *buf, unsigned len);

unde **len** = nr de octeți ce se încearcă a fi scriși din bufferul de memorie **buf** în fișierul asociat cu **handle.**

Funcția **lseek()** are următorul prototip:

int lseek(int handle, long offset, int fromwhere);

Se poziționează cursorul în fișierul asociat cu **handle** la distanța **offset,** octeții începând de la **fromwhere.**

Funcția **close()** are următorul prototip:

int close(int handle);

Funcția close() închide fișierul asociat cu **handle**, returnând valoarea –1 în caz de eroare și 0 dacă operația s-a efectuat cu succes.

Un **pointer la fișier** este un pointer la informațiile despre fișierul respectiv: numele, starea și poziția curentă .

Declararea unui pointer la un fișier se realizează după formatul :

FILE *fp ;

unde: **FILE** este un cuvânt rezervat în C, și este numele unui tip structură prin care se definește un fișier , iar **fp** este numele variabilei pointer la fișierul respectiv.

b) Funcții la nivel superior:

Aceste funcții sunt prezentate în tabelul de mai jos:

Funcție	Semnificație
fopen()	**deschiderea** unui fișier
fclose()	**închiderea** unui fișier
fputc(),putc()	**scrierea unui caracter** într-un fișier
fputs()	**scrierea unui șir de caractere** într-un fișier
fprintf()	**scrierea datelor formatate** într-un fișier
fgetc(),getc()	**citirea unui caracter** dintr-un fișier
fgets()	**citirea unui șir de caractere** dintr-un fișier
fscanf()	**citirea datelor formatate** dintr-un fișier
fseek()	**poziționarea cursorului** la un anumit octet într-un fișier
feof()	**determinarea sfârșitului de fișier**. Funcția returnează adevărat (≠0) dacă se determină sfârșit de fișier
ferror()	**determinarea unei erori** . Funcția returnează adevarat dacă a apărut o eroare
rewind()	**readucerea indicatorului de poziție la începutul fișierului**
remove()	**ștergerea** unui fișier
fflush()	**golirea streamului** asociat unui fișier

Funcțiile uzuale utilizate pentru **deschiderea** și **închiderea fișierelor** sunt:

- deschiderea unui fișier **fopen()**
- închiderea unui fișier: **fclose()**

Funcțiile uzuale pentru **scrierea în fișiere** sunt:

- scrierea unui caracter în fișier: fputc
- scrierea unui sir de caractere în fișier: fputs
- scrierea datelor formatate în fișier: fprintf

Funcțiile utilizate pentru **citirea din fișiere** sunt:

- citirea unui caracter dintr-un fișier: fgetc
- citirea unui șir de caractere dintr-un fișier: fgets
- citirea datelor formatate dintr-un fișier: fscanf

Funcția fopen() realizează deschiderea fișierelor . Prototipul funcției este:

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

- **numefisier** este numele fișierului ce va fi deschis în modul mod, și poate include opțional și o cale (director)
- **mod** este un șir de caractere care indică modul în care va fi deschis fișierul respectiv

mod	Semnificație

r	deschiderea unui fișier text pentru **citire**
w	deschiderea/creearea unui fișier text pentru **scriere**
a	deschiderea unui fișier text pentru **adăugare**
rb	deschiderea unui **fișier binar** pentru **citire**
wb	deschiderea (creearea) unui **fișier binar** pentru **scriere**
ab	deschiderea unui **fișier binar** pentru **adăugare**
r+	deschiderea unui fișier text pentru **citire/scriere**
w+	deschiderea(creearea) unui fișier text pentru **citire/scriere**
a+	deschiderea (crearea) unui fișier text pentru **citire,scriere și adăugare**
r+b	deschiderea un **fișier binar** pentru **citire/scriere**
w+b	deschiderea (crearea) unui **fișier binar** pentru **citire/scriere**
a+b	deschiderea (crearea) unui **fișier binar** pentru **citire/scriere**

Funcția **fopen()** returnează un pointer null în caz de eroare la deschiderea fișierului, și un pointer la fișierul deschis în caz de succes.

Funcția **fclose()** se utilizează pentru închiderea unui fișier (și a stream-ului asociat) deschis cu fopen() și are următorul prototip:

int fclose(FILE *fp) ;

unde **fp** este pointerul la fișierul deschis cu fopen()

Funcția returnează 0 în caz de succes, și EOF în caz de eroare și este inclusă în biblioteca <stdio.h>.

Operația de închidere trebuie efectuată după încheierea operațiilor de citire/scriere în fișier, în caz contrar pot rezulta pierderi de date din fișier și chiar distrugerea fișierului respectiv.

Exemplu : deschiderea/închiderea unui fișier

```
FILE *fp;
…
fp=fopen(”test.txt”,”w”); //deschiderea fisierului
….                        //operatii de scriere in fisier
fclose(fp);               //inchiderea fisierului
```

Funcția **fputc()** este utilizată pentru scrierea unui singur caracter într-un fișier. Prototipul funcției este:

int fputc(int ch, FILE *fp) ;

unde : **fp** este pointerul returnat la deschiderea fișierului cu fopen() și specifică în ce fișier va fi scris caracterul **ch**

ch este caracterul scris în fișier, este de tip int dar se folosește numai octetul inferior

Funcția este inclusă în biblioteca <stdio.h>.

Exemplu: scrierea unui caracter în fișier

```
FILE *fp;
```

```
char ch=“A”;
…
fp=fopen(”test.txt”,”w”); //deschiderea fisierului
ch=fputc(fp);      //operatie de scriere in fisier
fclose(fp);                  //inchiderea fisierului
```

Funcția **fgetc()** este utilizată pentru citirea unui singur caracter dintr-un fișier. Prototipul funcției este:

int fgetc(FILE *fp) ;

unde **fp** este pointerul returnat la deschiderea fișierului cu fopen()

Funcția returnează un întreg (octetul superior) și EOF pentru eroare, fiind inclusă în biblioteca <stdio.h>.

Funcția **feof()** este utilizată pentru determinarea sfârșitului de fișier și are umătorul prototip:

int feof(FILE *fp) ;

unde ***fp** este pointerul returnat la deschiderea fișierului cu fopen()

Funcția returnează 0 dacă s-a ajuns la sfârșitul fișierului sau o valoare pozitivă în caz contrar.

Funcția este inclusă în biblioteca <stdio.h>.

Funcția **fflush()** este utilizată pentru golirea streamului atașat unui fișier sau golirea tuturor fișierelor deschise pentru scriere și are următorul prototip:

int fflush(FILE *fp);

unde ***fp** pointer la fișier

Funcția scrie conținutul datelor din stream(buffer) în fișierul asociat lui fp. Dacă fp este null atunci vor fi golite toate fișierele deschise pentru ieșiri.

Funcția returnează 0 în caz de succes, altfel returnează EOF, și este inclusă în biblioteca <stdio.h>.

Funcția **fputs()** se utilizează pentru scrierea unui șir de caractere într-un fișier și are următorul prototip:

int fputs(const char *sir, FILE *fp) ;

unde funcția scrie șirul de caractere pointat de ***sir** în fișierul specificat prin pointerul ***fp**
Funcția returnează o valoare pozitivă, sau EOF în caz de eroare și este inclusă în <stdio.h>.

Exemplu :

```
...
char sir[80]=“orice text” ;
FILE *fp;
…
fputs(sir,fp);
…
```

Funcția **fgets()** este utilizată pentru citirea unui șir de caractere și are următorul prototip

char *fgets(char *sir, int n, FILE *fp)

unde fgets() citește un șir de lungime **n** caractere din fișierul specificat prin ***fp**, în șirul pointat de ***sir**.

Funcția returnează un pointer la șirul citit din fișier sau un pointer NULL în caz de eroare și este definită în biblioteca <stdio.h>.

Funcția **fprintf()** realizează scrierea în fișiere a datelor formatate și are următorul prototip:

int fprintf(FILE *fp, const char *sir_control,…);

***fp** este un pointer de fișier, returnat de o apelare a funcției fopen() și operația de scriere se efectuează asupra acestui fișier.

Funcția este definită în <stdio.h>.

Exemplu : scrierea într-un fișier a 2 variabile, tip șir și respectiv tip întreg

```
FILE *fp;
char s[80];
int t;
….
fprintf(fp,”Sirul este:%s, n=%d”, s,t ); //scrie in fisier
….
```

Funcția **fscanf()** este utilizată pentru citirea datelor formatate și are următorul prototip:

int fscanf(FILE *fp, const char *sir_control,…);

unde ***fp** este un pointer de fișier, returnat de o apelare a funcției fopen() și operația de citire se efectuează asupra acestui fișier.

Funcția este definită în <stdio.h>.

Funcția rewind() este utilizată pentru repoziționarea indicatorului de poziție la începutul fișierului și are următorul prototip:

void rewind(FILE *fp);

unde ***fp** pointer la fișierul în care se realizează repoziționarea indicatorului de poziție

Funcția este definită în <stdio.h>.

Funcția **ferror()** este utilizată pentru determinarea unei erori detectate la efectuarea unor operații asupra unui fișier și are următorul prototip:

int ferror(FILE *fp);

Funcția returnează o valoare pozitivă dacă a apărut o eroare în timpul ultimei operații asupra fișierului deschis prin pointerul *fp, și 0 în caz de reușita. Funcția este inclusă în biblioteca <stdio.h>

Funcția **remove()** este utilizată pentru ștergerea fișierelor din programul C/C++ și are următorul prototip:

int remove(const char *numefisier);

Funcția șterge fișierul specificat și returnează 0 în caz de succes al operației de ștergere, altfel o valoare diferită de 0.

Funcțiile specifice citirii/scrierii blocurilor de date din/în fisiere binare sunt: fwrite(), fread().

Funcția **fread()** are prototipul:

size_t fread(void*buffer,size_t nrocteti,size_t numara,FILE *fp);

unde ***buffer** este un pointer către o regiune de memorie în care se vor memora date primite de la fișier

numara corespunde numărului de elemente citite, cu dimeniunea egală cu **nrocteti**
size_t este definit în <stdio.h> și este aproximativ echivalent cu întreg fără semn

Funcția returnează nr. de elemente citite din fișier; această valoare poate fi mai mică decât **numara** dacă se ajunge la sfârșitul fișierului sau dacă apare o eroare.

Funcția **fwrite()** are prototipul:
size_t fwrite(const void *buffer, size_t nrocteti, size_t numara, FILE *fp);
unde ***buffer** este un pointer către datele care vor fi scrise în acel fișier

numara corespunde numărului de elemente scrise, având dimensiunea în octeți egală cu **nrocteti**

Funcția returnează nr. de elemente scrise în fișier; această valoare va fi egală cu **numara** dacă nu apare o eroare. Ambele funcții sunt definite în <stdio.h>.

Funcția **fseek()** realizează accesul aleator la un fișier și are prototipul:

int fseek(FILE *fp, long numocteti, int origine);

unde ***fp** este un pointer pentru fișier, returnat de o apelare a funcției fopen();

numocteti corespunde numărului de octeți de la origine care va deveni noua poziție curentă
origine este una din următoarele definiâii macro din <stdio.h>:

- Început fișier SEEK_SET
- Poziție curentă SEEK_CUR
- Sfârșit fișier SEEK_END

Funcția returnează 0 pentru operație încheiată cu succes, și valoare diferită de 0 pentru eroare. Funcția se utilizează pentru citire/scriere aleatoare și este definită în <stdio.h>.

Utilizarea intrării/ieșirii și erorii standard se realizează prin stream-urile standard: stdin,stdout,stderr. La lansarea în execuție a unui program C/C++ se deschid automat următoarele stream-uri:

- **stdin**: streamer de intrare (intrare standard) de tip text. Ex. tastatura
- **stdout**: streamer de ieșire (iesire standard) de tip text. Ex.monitorul
- **stderr**: streamer de ieșire erori (iesire standard erori) de tip text.

La închiderea programului C/C++ stream-urile sunt automat închise.

Exemplu : scriere la ieșirea standard

```
putchar(char c)
{return putc(c, stdout);}
```

Exemplu : citire de la intrarea standard

```
FILE *fp
char s[80];
int t;
fscanf(stdin, “%s%d”,s,&t);
```

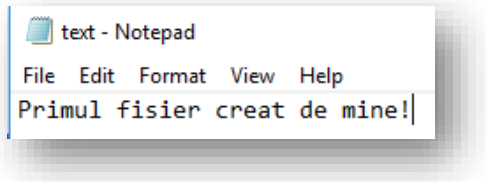
PROBLEME REZOLVATE:

Ex.1. Programul ilustrează crearea, deschiderea și scrierea într-un fișier numit “text.txt” utilizând funcțiile de nivel inferior: open, close, write. Să se verifice conținutul fișierului text.txt.

Varianta in C		**Varianta in C++**

#include <stdio.h> //pentru printf
#include <string.h> //pentru strlen
#include <fcntl.h> //pentru O_CREAT si O_RDWR
#include <io.h> //pentru open, write si close
int main()
{int test;
char msg[]="Primul fisier creat de mine!";
if ((test=open("text.txt",O_CREAT|O_RDWR))==-1)
{printf("eroare la deschidere fisier!\n");}
else {printf("Fisierul a fost creat. Verificati!\n");}
write(test,msg,strlen(msg)); close(test);
return 0;
}
		#include <iostream>
		#include <string.h> //pentru strlen
		#include <fstream>
		using namespace std;
		int main()
		{ofstream test("text.txt");
		char msg[]="Primul fisier creat de mine!";
		if (!test.is_open())
		{cout<<"eroare la deschidere fisier!"<<endl;}
		else {cout<<"Fisierul a fost creat. Verificati!"<<endl;}
		test<<msg;test.close();
		return 0;
		}

Rezultate:



Aplicație:

Să se modifice programul astfel încât să se adauge în fișier încă un șir de caractere introdus de la tastatură.

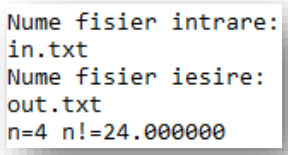
Ex.2. Programul realizează calculul factorialului: se citește n dintr-un fișier fis1.txt și tiparește n! într-un fișier fis2.txt. Atenție, fișierul de intrare în care se introduce valoarea lui n trebuie creat de către utilizator.După executarea programului să se verifice dacă fișierul de ieșire a fost creat și dacă conține rezultatul n! .

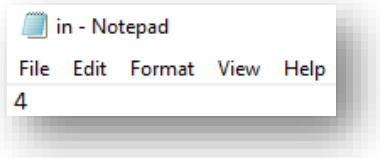
Varianta in C		**Varianta in C++**

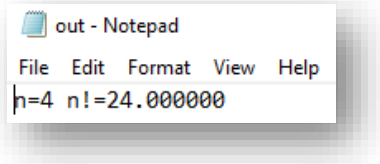
#include <stdio.h>
#include <stdlib.h>
int main (void)
{ FILE *fis1;
FILE *fis2;
char s1[12], s2[12];
int n,i;
double fact;
printf("Nume fisier intrare: \n");
scanf("%s",s1);
		#include <iostream>
		#include<fstream>
		#include <stdlib.h>
		using namespace std;
		int main (void)
		{ ifstream fis1;
		ofstream fis2;
		char s1[12], s2[12];
		int n,i;
		double fact;

printf("Nume fisier iesire: \n");
scanf("%s",s2);
fis1=fopen(s1,"r");
if (fis1==NULL) {printf("ERROR la deschidere");exit(1);}
fscanf(fis1,"%d",&n);
fflush(stdin);
fis2=fopen(s2,"w");
if (fis2==NULL) {printf("ERROR");
exit(1);}
/* citeste n si calculeaza n!*/
fact=1.; i=2;
for (i=2;i<=n;i++) fact=fact*i;
printf("n=%d n!=%lf \n", n, fact);
fprintf(fis2,"n=%d n!=%lf \n", n, fact);
return 0;
}
		cout<<"Nume fisier intrare: "<<endl;
		cin.get(s1,12);
		cin.ignore();
		cout<<"Nume fisier iesire: "<<endl;
		cin.get(s2,12);
		fis1.open(s1);
		if (fis1.fail()) {cerr<<"ERROR la
		deschidere"<<endl;exit(1);}
		fis1>>n;
		fis2.open(s2);
		/* citeste n si calculeaza n!*/
		fact=1.; i=2;
		for (i=2;i<=n;i++) fact=fact*i;
		printf("n=%d n!=%lf \n", n, fact);
		fis2<<"n="<<n<<" n!="<<fact<<endl;
		fis2.close();
		return 0;}

Rezultate:
-fișierul de intrare trebuie să se afle în rădăcina directorului care se creează pentru proiect (nu în bin, nu în bin/debug)
-dacă fișierul de intrare nu există execuția se termină cu eroare
fișierul de ieșire e creat automat la execuție







Aplicație:

Să se modifice programul astfel încât să se scrie în fișierul de ieșire rezultatul lui $\sqrt{n^3}$.

Ex.3.Programul realizează ștergerea unui fișier specificat în linia de comandă sub forma:
>remove numefisier.extensie

Varianta in C		**Varianta in C++**
#include <stdio.h> #include <stdlib.h> #include <ctype.h> int main(int argc,char *argv[])		#include <iostream> #include <stdlib.h> #include <stdlib.h> #include <cstdio>//pentru remove

{ char s[50];
  if (argc!=2){ printf("corect remove fisier.extensie\n");exit(1);}
  printf("Sterg %s ? (Y/N):",argv[1]); gets(s);
  if(toupper(*s)=='Y') {if(remove(argv[1])) {
  printf("fisierul nu se poate sterge\n"); exit(1);}
  else printf("Chiar l-am sters !");
  return 0; }}
		using namespace std;
		int main(int argc,char *argv[])
		{ char s[50];
		  if (argc!=2){ cout<<"corect remove fisier.extensie"<<endl;exit(1);}
		  cout<<"Sterg "<<argv[1]<<"? (Y/N):"; cin.get(s,50);
		  if(toupper(*s)=='Y') { if(remove(argv[1])) {
		  cout<<"fisierul nu se poate sterge"<<endl; exit(1);}
		  else cout<<"Chiar l-am sters !";} }

Rezultate:

Obs:

- fișierul executabil poate fi redenumit remove.exe sau proiectul va fi numit remove
- execuția trebuie realizată dintr-o linie de comandă
- înainte de execuție trebuie creat un fișier text (de ex cu Notepad)

Aplicație:

Să se verifice dacă fișierul a fost efectiv șters

```
 Directory of D:\laura\codeblocks\ex83\bin\Debug

02/12/2019  02:18 PM    <DIR>          .
02/12/2019  02:18 PM    <DIR>          ..
02/12/2019  02:18 PM                19 in.txt
02/12/2019  02:16 PM            30,035 remove.exe
               2 File(s)         30,054 bytes
               2 Dir(s)  386,563,194,880 bytes free

D:\laura\codeblocks\ex83\bin\Debug>remove in.txt
Sterg in.txt ? (Y/N):y
Chiar l-am sters !
D:\laura\codeblocks\ex83\bin\Debug>dir
 Volume in drive D is Local Disk
 Volume Serial Number is 4E57-A59E

 Directory of D:\laura\codeblocks\ex83\bin\Debug

02/12/2019  02:19 PM    <DIR>          .
02/12/2019  02:19 PM    <DIR>          ..
02/12/2019  02:16 PM            30,035 remove.exe
               1 File(s)         30,035 bytes
               2 Dir(s)  386,563,194,880 bytes free
```

Ex. 4. Programul realizează scrierea/citirea unor variabile tip int și double (nu caractere!) într-un /dintr-un fișier binar. Să se verifice dacă fișierului binar "test" a fost creat și dacă conține variabilele tipărite.

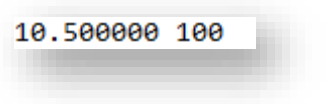
Varianta in C		Varianta in C++

#include <stdio.h>
#include <stdlib.h>
int main(void)
{FILE *fp;
double d=10.5; int i=100;
if((fp=fopen ("test","wb+"))==NULL) {
  printf("nu se poate deschide fisierul\n");
exit(1); }
		#include <stdio.h>
		#include <iostream>
		#include <stdlib.h>
		using namespace std;
		int main(void)
		{FILE *fp;
		double d=10.5; int i=100;
		if((fp=fopen ("test","wb+"))==NULL) {

```
fwrite(&d, sizeof(double),1,fp);
fwrite(&i, sizeof(int),1,fp); rewind(fp);
fread(&d, sizeof(double),1,fp);
fread(&i, sizeof(int),1,fp);
printf("%lf %d ",d,i);
fclose(fp);
return 0;
}
```

```
        printf("nu se poate deschide fisierul\n");
exit(1);}
fwrite(&d, sizeof(double),1,fp);
fwrite(&i, sizeof(int),1,fp); rewind(fp);
fread(&d, sizeof(double),1,fp);
fread(&i, sizeof(int),1,fp);
cout<<d<<"  "<<i;
fclose(fp);
return 0;}
```

Rezultate:



Aplicație:

Să se modifice programul astfel încât să se citească de la tastatură două valori pentru variabilele d și i și apoi să se scrie pătratul lor în fișierul test

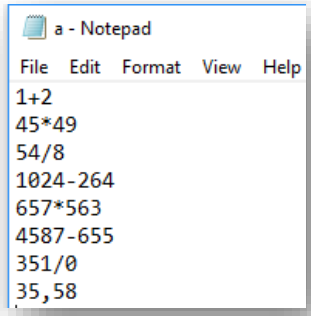
Ex.5: Programul citește pe rând câte o operație din fișierul de intrare a.txt, sub forma op1 op op2 și se tipărește rezultatul operației în fișierul ab.txt

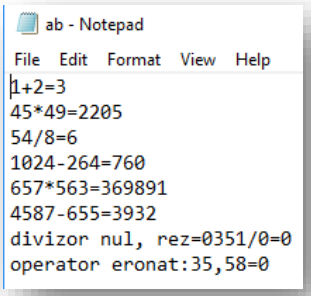
```
#include <stdio.h>
#include <stdlib.h>
int main (void)
{FILE *fis1, *fis2;
int op1,op2,rez; char op;
fis1=fopen("a.txt","r");
fis2=fopen("ab.txt","wa");
while
(fscanf(fis1,"%d%c%d\n",&op1,&op,&op2)!=EOF)
{          switch (op) {
case '+': rez=op1+op2; break;
case '-': rez=op1-op2; break;
case '*': rez=op1*op2; break;
case '/': if (op2==0) { rez=0;fprintf(fis2,"divizor nul,
rez=%d", rez); }
else rez=op1/op2;  break;
default: rez=0; fprintf(fis2,"operator eronat:");}
fprintf(fis2,"%d%c%d=%d\n",op1,op,op2,rez);}
printf ("Succes");
   return 0;}
```

```
#include <iostream>
#include <stdlib.h>
#include<fstream>
using namespace std;
int main (void)
{ifstream fis1;
ofstream fis2;
int op1,op2,rez; char op;
fis1.open("a.txt");
fis2.open("ab.txt");
if (fis1.fail()) {cerr<<"EROARE la
deschidere"<<endl;exit(1);}
while (!fis1.eof())
{          fis1>>op1>>op>>op2;
   switch (op) {
case '+': rez=op1+op2; break;
case '-': rez=op1-op2; break;
case '*': rez=op1*op2; break;
case '/': if (op2==0) { rez=0;fis2<<"divizor nul, rez=
"<<rez; }
else rez=op1/op2;  break;
default: rez=0; fis2<<"operator eronat:";}
fis2<<op1<<op<<op2<<"="<<rez<<endl;}
cout<<"Succes";
   return 0;}
```

Rezultate:

Aplicație:





Să se modifice programul astfel încât să se adauge și alte operații matematice (pow, sqrt,sin, log, etc) utilizând funcții din <math.h>

Ex.6: Programul copiază un fișier text specificat în alt fișier specificat de către utilizator. Trebuie să creați fișierul de intrare și să salvați un text oarecare în acest fișier, iar apoi după executarea programului să verificați dacă fișierul copiat a fost creat și dacă conține același text ca și fișierul de intrare.

Varianta in C		Varianta in C++

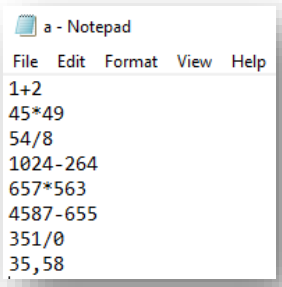
#include <stdio.h>
#include <stdlib.h>
int main()
{char in_name[25], out_name[25];
FILE *in_file, *out_file, *fopen (); int c;
printf("Numele fisierului ce trebuie copiat:\n");
scanf("%24s", in_name);
printf("Numele fisierului in care se copiaza:\n");
scanf("%24s", out_name);
in_file = fopen (in_name, "r");
if(in_file == NULL)
printf("Nu pot deschide %s pentru citire.\n",
in_name);
else { out_file = fopen (out_name, "w");
if(out_file == NULL)
printf("Nu pot deschide %s pentru scriere.\n",
out_name);
else { while((c = getc(in_file)) != EOF)
putc (c, out_file); putc (c, out_file);
/* copy EOF */
printf("Fisierul a fost copiat.\n");
fclose (out_file); } fclose (in_file); }
return 0;}
		#include <iostream>
		#include <stdlib.h>
		#include<fstream>
		using namespace std;
		int main()
		{char in_name[25], out_name[25];
		ifstream in_file;
		ofstream out_file; char c;
		cout<<"Numele fisierului ce trebuie copiat:"<<endl;
		cin.get(in_name,25);
		cin.ignore();
		cout<<"Numele fisierului in care se copiaza:"<<endl;
		cin.get(out_name,25);
		cin.ignore();
		in_file.open(in_name);
		if(in_file.fail())
		cerr<<"Nu pot deschide "<<in_name<<" pentru
		citire."<<endl;
		else { out_file.open (out_name);
		if(out_file.fail())
		cerr<<"Nu pot deschide "<<out_name<<" pentru
		scriere."<<endl;
		else { while (in_file.get(c)) out_file<<c;
		cout<<"Fisierul a fost copiat.";
		in_file.close(); } out_file.close();} return 0;}

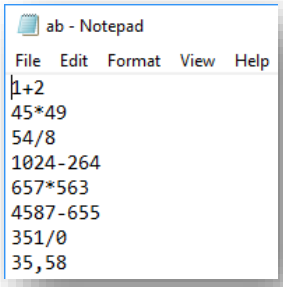
Rezultate:

```
Numele fisierului ce trebuie copiat:
a.txt
Numele fisierului in care se copiaza:
ab.txt
Fisierul a fost copiat.
```

Aplicație:

Să se modifice programul astfel încât să se adauge la sfârșitul fișierului copiat textul "STOP".





Ex.7 Fișierul int.txt se inițializează prin program cu numere de la 0 la 1000 iar dublul acestor numere se tipărește în fișierul out.txt.

Varianta in C		Varianta in C++

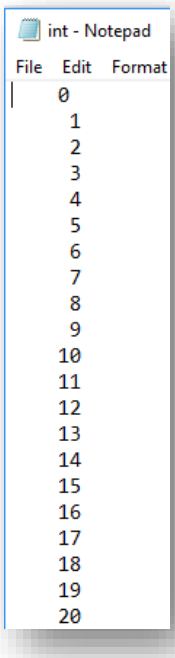
#include <stdio.h>
#include <stdlib.h>
#define max 2000
int main(void)
{FILE *fis1, *fis2;
int i,j=0,a[max], *p;
printf ("\nInitializarea sirului cu nr. de la 0 la 1000 folosind un pointer si un index\n");
fis1=fopen("int.txt","w"); fis2=fopen("out.txt","w");
for (p=a,i=0;i<=1000;j++,i++) p[i]=j;
for (p=a,i=0;i<=1000;i++)
{printf("%4d", p[i]);
fprintf(fis1,"%5d \n ", p[i]);
fprintf(fis2,"%5d \n", p[i]*2);}
fclose (fis1);fclose (fis2);
return 0;}
		#include <iostream>
		#include <stdlib.h>
		#define max 2000
		#include <fstream>
		using namespace std;
		int main(void)
		{ofstream fis1, fis2;
		int i,j=0,a[max], *p;
		cout<<endl<<"Initializarea sirului cu nr. de la 0 la 1000 folosind un pointer si un index";
		fis1.open("int.txt"); fis2.open("out.txt");
		for (p=a,i=0;i<=1000;j++,i++) p[i]=j;
		for (p=a,i=0;i<=1000;i++)
		{cout<< p[i]<<" ";
		fis1<<p[i]<<endl;
		fis2<<p[i]*2<<endl;}
		fis1.close();fis2.close();
		return 0;}

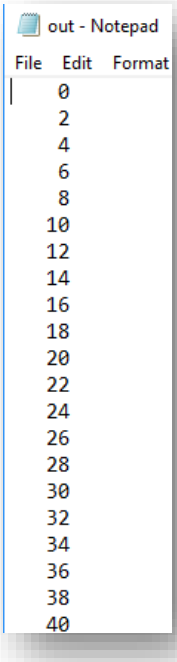
Rezultate: ***Aplicatie:***

secvență din pagina de rezultate:

```
Initializarea sirului cu nr. de la 0 la 1000 folosind un pointer si un index
   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  2
5  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50
51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76
 77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102
 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 12
8 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 1
54 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179
180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205
 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 23
1 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 2
57 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282
283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308
 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 33
4 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 3
60 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385
386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411
 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 43
7 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 4
63 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488
489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514
 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 54
0 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 5
66 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591
592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617
```

Modificați programul astfel încât fișierul de ieșire să conțină valoarea polinomului $2x^3+7x^2-x+5$, unde x sunt valorile din fișierul int.txt





Ex.8 Programul realizează calculul mediilor studenților ale căror date sunt introduse în fișierul int.txt sub forma (tab între campuri):

Popa Mircea	1111	10.	9.5	9.5	8.
Munteanu Ioan	1323	6.5	7.5	8.5	6.5
Adam Mihai	1121	7.5	6.	7.	8.
Cristea Mihaela	1122	8.5	9.	10.	7.5
Alisie Mirela	2121	6.5	8.5	7.5	9.
Laza Sergiu	2222	10.	10.	10.	10.
Vaida Maria	2525	9.5	7.	6.	8.

Fișierul care conține articolele de tip student cu mediile calculate este med.txt. Programul calculează mediile studenților în fișierul med.txt și apoi ordonează descrescător în ordinea mediilor studenții în fișierul sort.txt .

Varianta in C

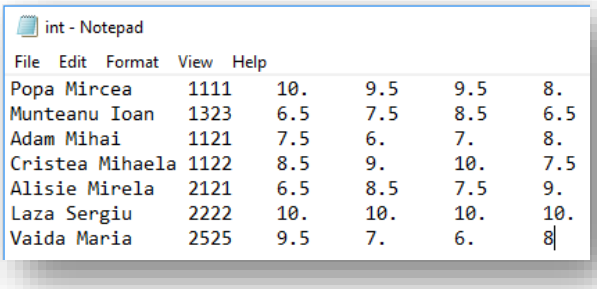
```
#include <stdio.h>
#include <stdlib.h>
struct student{
                    char name[20];
                    char prenume[20];
                    int id;     //cod grupa
                    double nota1;
                    double nota2;
                    double nota3;
                    double nota4;
                    double media;} s[100], aux[100];
FILE *fis1;                 //fisier I cu notele studentilor int.txt
FILE *fis2;                 //fisier O cu mediile studentilor med.txt
FILE *fis3;                 //fisier O cu studentii ordonati dupa medii sort.txt
int i=0,n=0, k=0;
char ch;
void sortmed(struct student s[100]);
int main()
{fis1=fopen("int.txt","r");
if (fis1==NULL) {printf("ERROR");  exit(1);}
while (fscanf(fis1, "\n%14s\t%14s\t%d\t%lf\t%lf\t%lf\t%lf", s[i].name, s[i].prenume,&s[i].id,
&s[i].nota1,&s[i].nota2,&s[i].nota3,&s[i].nota4)!=EOF)
{i++;}n=i;
printf("n=%d\n", n);
fis2=fopen("med.txt", "w+");
for (i=0;i<n;i++)
        {s[i].media = (s[i].nota1+s[i].nota2+s[i].nota3+s[i].nota4)/4;
        fprintf(fis2, "%14s\t%14s\t%d\t%lf\t%lf\t%lf\t%lf\t%lf\n", s[i].name, s[i].prenume,s[i].id,
s[i].nota1,s[i].nota2,s[i].nota3,s[i].nota4,s[i].media);}
printf("Fisierul sort.txt este ordonat descrecator in ordinea mediilor");
for (i=0;i<n;i++)  {sortmed(s);}
fis3=fopen("sort.txt","w+");
fprintf(fis3,"\nLista studentilor in ordinea mediilor\n");
for (i=0; i<n; i++)
{fprintf(fis3,"\n%d.\nNume : %-14s\nPrenume: %-14s\ncod grupa:
%d\nmedia:%lf\n",(i+1),s[i].name,s[i].prenume,s[i].id,s[i].media);
    fprintf(fis3, "**********************************\n"); }
    return 0;}
void sortmed(struct student s[100])
{int j;
for (j=0;j<=n;j++)
        {if (s[j].media<s[j+1].media)
            { aux[j]=s[j] ;
              s[j]=s[j+1];
              s[j+1]=aux[j];k++;printf("k=%d\n",k);}  }
}
```

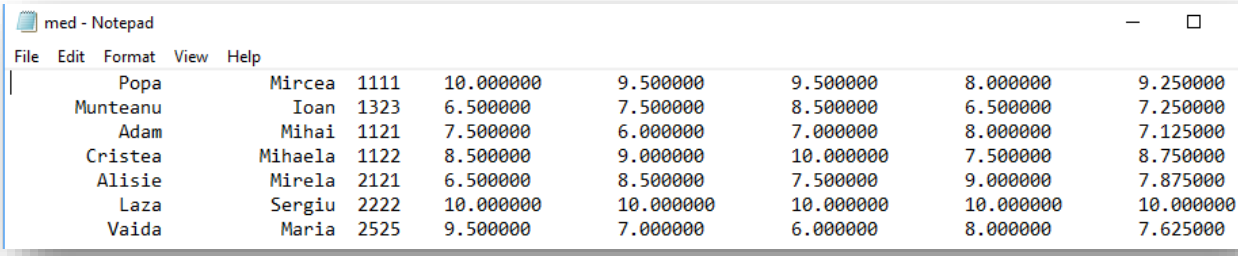
Varianta C++

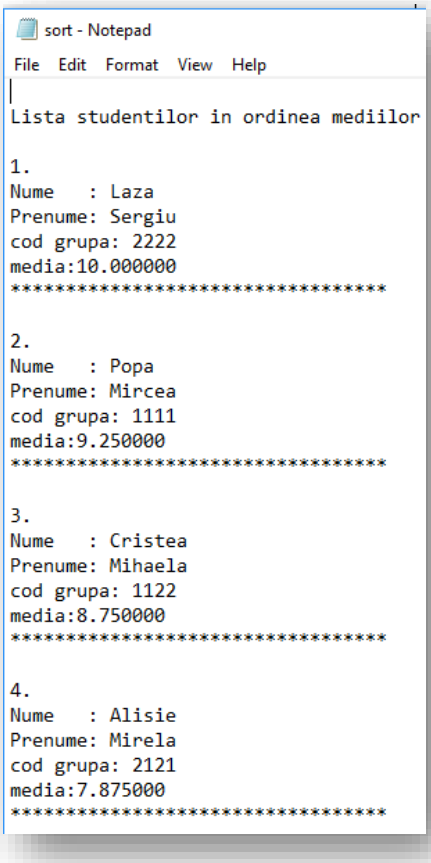
```
#include <iostream>
#include <stdlib.h>
#include<fstream>
```

```
using namespace std;
struct student{
                    char name[20];
                    char prenume[20];
                    int id;     //cod grupa
                    double nota1;
                    double nota2;
                    double nota3;
                    double nota4;
                    double media;} s[100], aux[100];
ifstream fis1;              //fisier I cu notele studentilor int.txt
ofstream fis2;              //fisier O cu mediile studentilor med.txt
ofstream fis3;              //fisier O cu studentii ordonati dupa medii sort.txt
int i=0,n=0, k=0;
char ch;
void sortmed(struct student s[100]);
int main()
{fis1.open("int.txt");
if( fis1.fail())
   {cerr<<"eroare"<<endl; exit(1);}
while (!fis1.eof())
{
  fis1>>s[i].name>>s[i].prenume>>s[i].id>>s[i].nota1>>s[i].nota2>>s[i].nota3>>s[i].nota4;
i++;}
n=i;
cout<<"n="<<n<<endl;
fis2.open("med.txt");
for (i=0;i<n;i++)
          {s[i].media = (s[i].nota1+s[i].nota2+s[i].nota3+s[i].nota4)/4;
          fis2<<s[i].name<<"\t"<<s[i].prenume<<"\t"<<s[i].id<<"\t"<<s[i].nota1<<"\t"<<s[i].nota2<<"\t"<<s
[i].nota3<<"\t"<<s[i].nota4<<"\t"<<s[i].media<<endl;}
cout<<"Fisierul sort.txt este ordonat descrecator in ordinea mediilor";
for (i=0;i<n;i++)  {sortmed(s);}
fis3.open("sort.txt");
fis3<<"\nLista studentilor in ordinea mediilor"<<endl;
for (i=0; i<n; i++)
{fis3<<"\n"<<i+1<<"\nNume : "<<s[i].name<<"\nPrenume: "<<s[i].prenume<<"\ncod grupa:
"<<s[i].id<<"\nmedia: "<<s[i].media<<endl;
     fis3<<"**********************************"<<endl; }
     return 0;}
void sortmed(struct student s[100])
{int j;
for (j=0;j<=n;j++)
          {if (s[j].media<s[j+1].media)
              { aux[j]=s[j] ;
                s[j]=s[j+1];
                s[j+1]=aux[j];k++;cout<<"k="<<k<<endl;}  }
}
```

Rezultate:







Aplicație: *Modificați programul astfel încât să sortați fișierul în ordine alfabetică.*

Probleme propuse

1. ***Să se scrie un program care permite introducerea liniilor de text de la tastatură într-un fișier specificat în linia de comandă. Dacă programul se numește scrie.c atunci în linia de comandă se introduce de ex: >scrie text.txt***
   și un text oarecare scris pe mai multe linii fiecare linie incheiată cu Enter și terminat prin caracterul "." Fișierul text.txt va conține liniile de text introduse de la tastatură după linia de comandă.

2. ***Să se scrie programul care:***
   - ***citește datele angajaților dintr-o firmă de la tastatură sub forma: nume, prenume, adresa,cod numeric,salar brut***
   - ***creează un fișier în care se scriu aceste date***
   - ***calculează salarul net și impozitul (după formula impozit =0.4*salarbrut) pentru fiecare angajat***
   - ***crează un fișier în care sunt tipăriți într-un tabel tip stat de plată toți angajații cu sumele corespunzătoare salarului brut, salarului net și impozitului.***

3. ***Să se scrie programul care: citeste dint-un fisier un sir de numere reale x[100] si afiseaza in alt fisier valorile calculate ale functiei:*** $f(x) = \begin{cases} x^2 - 1, x > 0 \\ -5, x < 0 \end{cases}$

4. ***Să se scrie programul care: citeste dint-un fisier un sir de numere reale x[100] si afiseaza in alt fisier valorile calculate ale expresiei :E(x)=*** $\sum_{i=1}^{n}(x^2 + 1)$