14. Fişiere.

14.1. Operaţii de intrare / ieşire.

In limbajul C nu există instrucţiuni de intrare / ieşire. Operaţiile de intrare / ieşire sunt realizate prin apelul unor funcţii ale sistemului de operare. Acestea sunt implementate prin funcţii, sub o formă compatibilă pentru diversele sisteme de operare (sunt portabile).

Un *fişier* este o colecţie ordonată de ***articole*** (înregistrări) păstrate pe un suport extern de memorie şi identificate printr-un nume.
Pentru ***fişierul standard de intrare*** , datele sunt introduse de la tastatură.
Pentru ***fişierul standard de ieşire*** , rezultatele sunt afişate pe terminalul standard de ieşire.
Mesajele de eroare se afişează în ***fişierul standard de eroare*** .

Fişierul are un articol care marchează ***sfârşitul fişierului***. Pentru fişierul standard de intrare de la tastatură, *sfârşitul de fişier*, pentru sistemele de operare DOS şi Windows se generează prin **`Ctrl-Z`** (pentru Unix – prin **`Ctrl-D`**).

Operaţiile specifice prelucrării fişierelor sunt.

- deschiderea unui fişier
- închiderea unui fişier
- creerea unui fişier
- citirea de articole din fişier (consultarea fişierului)
- actualizarea (sau modificarea) fişierului
- adăugare de articole la sfârşitul fişierului
- poziţionarea în fişier.
- ştergerea unui fişier
- schimbarea numelui unui fişier

Prelucrarea fişierelor se face pe două niveluri:
- nivelul inferior, care apelează direct la sistemul de operare.
- nivelul superior, care utilizează structuri speciale **`FILE`**

Funcţiile de pe nivelul superior nu asigură o independenţă totală faţă de sistemul de operare.
Funcţiile standard de intrare / ieşire au prototipurile în fişierul antet **`<stdio.h>.`**

14.2. Fişiere text şi fişiere binare

Într-un fişier text, toate datele sunt memorate ca şiruri de caractere, organizate pe linii, separate între ele prin marcajul sfârşit de linie **`'\n'`** .
Într-un fişier text spaţiul de memorare pe disc nu este folosit în mod eficient pentru datele numerice (astfel întregul **`12345`** ocupă 5 octeţi).

Într-un fişier binar, datele sunt păstrate în formatul lor intern (2 octeţi pentru **`int`**, 4 octeţi pentru **`float`**, etc).

La fişierele text ***marcajul de sfârşit de fişier*** (caracterul **`0X1A`**) există fizic în fişier. La întâlnirea acestui caracter funcţia **`fgetc()`** întoarce **`EOF (-1)`**. Marcajul de sfârşit de fişier se generează de la tastatură prin **`Ctrl-Z`**.

În cazul *fişierelor binare*, marcajul de sfârşit de fişier nu există fizic în fişier, ci este generat de funcția **fgetc()**.

În MSDOS (şi în Unix), la nivelul liniei de comandă intrările şi ieşirile standard pot fi redirectate în fişiere disc, fără a opera nici o modificare la nivelul programului. Astfel:

< redirectează intrarea standard către fişierul specificat
> redirectează ieşirea standard către fişierul specificat

Fişierul standard de eroare nu poate fi redirectat.

Fişierul specificat poate fi:
con – pentru consola sistem (tastatura, respectiv ecranul)
prn – pentru imprimanta paralelă
com1 – pentru interfața serială de date
nume_fişier – pentru un fişier disc
NUL – pentru perifericul nul.

Exemple:
> test.exe > prn redirectează ieşirea programului la imprimantă
> test.exe < f1.dat > f2.dat redirectează atât intrarea cât şi ieşirea programului

14.3. Accesul la fişiere.

Fişierele disc şi fişierele standard sunt gestionate prin pointeri la structuri specializate **FILE**, care se asociază fiecărui fişier pe durata prelucrării.
Fişierele standard au pointerii predefiniți: **stdin, stdout, stderr, stdprn, stdaux.**
Declararea unui pointer la fişier se face prin:

```
FILE *pf;
```

1. ***deschiderea unui fişier*:**

Înainte de a fi prelucrat, un fişier trebuie să fie deschis. Prin deschidere:
- se asociază unui *nume de fişier* un *pointer la fişier*
- se stabileşte un *mod de acces* la fişier

Pentru deschiderea unui fişier se foloseşte funcția cu prototipul:

```
FILE *fopen(char *nume_fisier, char *mod_acces);
```

Prin deschiderea unui fişier se stabileşte o conexiune logică între fişier şi variabila pointer şi se alocă o zonă de memorie (buffer) pentru realizarea mai eficientă a operațiilor de intrare / ieşire. Funcția întoarce:
- un pointer la fişier, în caz că deschiderea fişierului se face în mod corect
- **NULL** dacă fişierul nu poate fi deschis.

Vom considera mai întâi două ***moduri de acces*** :
- ***citire*** (sau consultare) **"r"** – citirea dintr-un fişier inexistent va genera eroare
- ***scriere*** (sau creare) **"w"** - dacă fişierul există deja, el va fi şters

Fişierele standard nu trebuiesc deschise.

Redirectarea unui fişier deschis poate fi realizată cu:

```
FILE* freopen(char* nume, char* mod, FILE* flux));
```

Fişierul deschis este închis şi este deschis un nou fişier având ca sursă fluxul, numele şi modul de acces specificați ca parametri. O utilizare importantă o constituie redirectarea fluxului standard de intrare: datele vor fi citite din fişierul specificat, fără a face nici o modificare în program.

2. închiderea unui fişier

După terminarea prelucrărilor asupra unui fişier, acesta trebuie închis. Un fişier este închis automat la apelarea funcției **`exit()`**.

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

- funcția întoarce 0 la închidere normală şi **`EOF`** la producerea unui incident
- fişierele standard nu se închid de către programator
- în cazul unui fişier de ieşire, se scriu datele rămase nescrise din buffer în fişier, aşa că operația de închidere este obligatorie
- în cazul unui fişier de intrare, datele necitite din bufferul de intrare sunt abandonate
- se eliberează bufferele alocate
- se întrerupe conexiunea pointer – fişier

Secvența următoare deschide un fişier cu un nume dat şi apoi îl închide.

```
FILE *pf = fopen("test1.dat", "w");
fclose(pf);
```

14.4. Operații de intrare – ieşire.

Tabel 14.1. Funcții pentru operații de intrare / ieşire

Tip fişier	Conversie	Unitate transferată	Funcții folosite
text	fără	caracter	**`fgetc(), fputc()`**
		linie	**`fgets(), fputs()`**
	cu	linii	**`fscanf(). fprintf()`**
binar	fără	articol (structură)	**`fread(), fwrite()`**

3.1. operații de intrare / ieşire la nivel de caracter

Scrierea unui caracter într-un fişier se face folosind funcția:

```
int fputc(int c, FILE *pf);
```

Funcția întoarce primul parametru sau `EOF`, în caz de eroare.

Citirea unui caracter dintr-un fişier se face cu funcția:

```
int fgetc(FILE *pf);
```

Funcția întoarce ca rezultat următorul caracter citit din fişier, convertit în întreg fără semn sau **`EOF`** dacă s-a citit sfârşit de fişier sau s-a produs o eroare la citire.

Exemplu : Scrieți un program care realizează copierea unui fişier. Numele celor doua fişiere (sursă şi destinație) sunt citite de la terminal.

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

```
/* copierea unui fisier */
void copiere1(FILE *, FILE *);
int main(){
   char numes[12], numed[12];
   gets(numes);
   gets(numed);
   FILE* s = fopen(numes,"r");
   FILE* d = fopen(numed,"w");
   copiere1(d, s);
   fclose(s);
   fclose(d);
}
void copiere1(FILE *d, FILE *s){
   int c;
   while ((c=fgetc(s)) != EOF)
      fputc(c, d);
}
```

3.2. operații de intrare / ieşire pentru şiruri de caractere

```
char *fgets(char *s, int n, FILE *pf);
```

- citeşte caractere din fişierul cu pointerul **pf**, până la întâlnirea primului caracter **'\n'** (cel mult **n-1** caractere) în tabloul **s**; pune la sfârşit **'\n'** şi **'\0'**
- întoarce **s** sau **NULL** la întâlnire sfârşit de fişier sau la eroare

Exemplu : Scrieți o funcție care simulează funcția **fgets().**

```
char  *fgets(char *s, int n, FILE *pf) {
  char c;
  char *psir = s;
  while (--n > 0 &&(c=fgetc(pf))!=EOF)
    if((*psir++=c)=='\n')
      break;
  *psir='\0';
  return (c==EOF && psir==s) ? NULL: s;
}
```

```
int fputs(char *s, FILE *pf);
```

- copiază şirul în fişierul de ieşire
- nu copiază terminatorul de şir **'\0'**
- întoarce un rezultat nenegativ (numărul de caractere scrise în fişier), sau **EOF** la producerea unei erori

Exemplu : Scrieți o funcție care simulează funcția **fputs().**

```
int fputs(char *s, FILE *pf)
{ int c, n=0;
  while (c = *s++){
    fputc(c, pf);
```

```
    n++;
  }
  return (ferror(pf)) ? EOF: n;
}
```

Copierea unui fişier folosind funcţii orientate pe şiruri de caractere are forma:

```
#define MAX 100
void copiere2(FILE *d, FILE *s){
  char linie[MAX];
  while(fgets(linie, MAX, s))
    fputs(linie, d);
}
```

Revenim acum asupra modurilor de acces la disc. Sunt posibile următoarele situaţii:

1. fişierul nu există; dorim să-l creem şi să punem informaţii în el
– **"w"** - deschidere pentru scriere, noile scrieri se fac peste cele vechi
2. fişierul există deja; dorim să extragem informaţii din el
– **"r"** - deschidere pentru citire, fişierul trebuie să existe deja
– **"r+"** - citire şi scriere ; fişierul trebuie să existe
3. fişierul există deja; dorim să adăugăm informaţii la el, păstrând informaţiile deja existente

– **"a"** - deschidere pentru adăugare, toate scrierile se adaugă la sfârşitul fişierului existent sau nou creat
– **"a+"** - citire şi adăugare; dacă fişierul nu există, el va fi creat
4. fişierul există deja; dorim să punem alte informaţii în el ştergând pe cele existente

– **"w+"** - citire şi scriere; dacă fişierul există deja el este şters

Modul de acces binar se specifică cu sufixul **"b"**. Astfel avem: **"rb", "w+b"**
Modul text este considerat implicit, dar poate fi specificat explicit prin **"t".**

3.3. operaţii de intrare / ieşire cu format

- *scrierea cu format*

```
int fprintf(FILE *pf, char *format, lista_expresii);
```

- transferă în fişierul specificat, valorile expresiilor, convertite, potrivit formatului în caractere
- întoarce numărul de caractere scrise, sau o valoare negativă, dacă s-a produs o eroare.

Un descriptor de conversie din format începe prin % şi poate avea un mai mulţi specificatori opţionali, care preced descriptorul:

```
%[indicator][lăţime][.precizie][spec_lung]descriptor;
```

Indicatorul poate avea una din valorile:

- aliniere stânga
+ afişare numere cu semn
0 completare stânga cu zerouri
  adaugare spaţiu înaintea primei cifre, dacă numărul este pozitiv
%#o - scrie 0 iniţial
  %#x - scrie **0x**
  %#e,f,g,E,G – scrie punctul zecimal şi nu elimină zerourile la sfârşit

Lățimea – număr ce indică lățimea minimă a câmpului în care se face scrierea.
* lățimea este dată de argumentul următor

Precizia este un număr interpretat diferit în funcție de descriptorul folosit.Astfel:
%e, %E, %f – numărul de cifre după punctul zecimal
%s –numărul maxim de caractere afişate
%g, %G – numărul de cifre semnificative
%d, %i – numărul minim de cifre (cu zerouri în față)

Specificarea lungimii se face prin:
H – short l – long L – long double

- ***citirea cu format***

```
int fscanf(FILE *pf, char *format, lista_adrese_variabile);
```

- se citesc date din fişierul pf, sub controlul formatului, inițializându-se variabilele din listă
- funcția întoarce numărul de câmpuri citite sau **EOF** în caz de producere a unui incident la citire sau întâlnire a marcajului de sfârşit de fişier.

3.4. intrări / ieşiri în modul de acces binar

- sunt operații de transfer (citiri / scrieri) fără conversii
- se fac la nivel de articol
- poziția în fişier este actualizată după fiecare citire / scriere

```
unsigned fread(void *zona, unsigned la, unsigned na, FILE *pf);
```

- citeşte cel mult na articole, de lungime la fiecare, din fişierul pf în zona
- întoarce numărul de înregistrări citite sau 0 în caz de eroare sau sfârşit de fişier

```
unsigned fwrite(void *zona, unsigned la, unsigned na, FILE *pf);
```

- scrie na articole de lungime la, din zona în fişierul **pf**
- întoarce numărul de articole scrise.

Pentru a copia un fişier binar (sau text) folosind funcțiile **fread()** şi **fwrite()** vom considera lungimea articolului 1 octet.

```
void copiere3(FILE * d, FILE * s) {
  int noc; /* numarul de octeti cititi */
  char zona[MAX];
  while((noc=fread(zona, 1, MAX, s)) > 0)
    fwrite(zona, 1, noc, d);
}
```

4. Operații pentru fişiere cu acces direct.

4.1. Poziționarea în fişier

```
int fseek(FILE *pf, long depl, int orig);
```

- modifică poziția curentă în fişierul cu pointerul **pf** cu **depl** octeți relativ la cel de-al treilea parametru **orig**, după cum urmează:

- față de începutul fişierului, dacă **orig=0** (sau **SEEK_SET**)

- față de poziția curentă, dacă **orig=1** (sau **SEEK_CUR**)
- față de sfârşitul fişierului, dacă **orig=2** (sau **SEEK_END**)

- întoarce rezultatul 0 pentru o poziționare corectă, şi diferit de 0 în caz de eroare.

```
void rewind(FILE *pf);
```

realizează o poziționare la începutul fişierului, fiind echivalent cu:

```
fseek(pf, 0L, SEEK_SET);
```

```
long ftell(FILE *pf);
```

- întoarce poziția curentă în fişier, exprimată prin numărul de octeți față de începutul fişierului

Exemplu : Scrieți o funcție care determină numărul de octeți ai unui fişier.

```
#include <stdio.h>
long FileSize(FILE *pf)
{ long pozv, noct;
  pozv = ftell(pf); /* salveaza pozitia curenta */
  fseek(pf, 0L, SEEK_END);   /* pozitionare la sfarsit */
  noct = ftell(pf); /* numar de octeti din fisier */
  fseek(pf, pozv, SEEK_SET); /*revenirea la pozitia veche*/
  return noct;
}
```

4. tratarea erorilor

```
int feof(FILE *pf);
```

- întoarce o valoare diferită de 0, dacă s-a detectat marcajul de sfârşit de fişier

```
int ferror(FILE *pf);
```

- întoarce o valoare diferită de 0, dacă s-a detectat o eroare în cursul operației de intrare / ieşire

Exemplul : Fişierul "comenzi.dat" conține articole structuri cu câmpurile:
-denumire produs- un şir de 20 de caractere
-cantitate comandată – o valoare reală.
Fişierul "depozit.dat" este format din articole având câmpurile:
- denumire produs – un şir de 20 de caractere
- stoc şi stoc_minim– valori reale
- preț unitar – valoare reală
Să se actualizeze fişierul de stocuri, prin onorarea comenzilor. O comandă este onorată, dacă prin satisfacerea ei, stocul rămas în magazie nu scade sub stocul minim.
Se va creea un fişier "facturi.dat", conținând pentru fiecare comandă onorată, denumirea produsului comandat şi valoarea comenzii.
Se vor crea de asemeni două fişiere, unul cu comenzi care nu au fost satisfăcute, deoarece produsele erau în cantități insuficiente, celălalt cu comenzi de produse care nu există în depozit.

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

```
typedef struct { char den[20];
                 double cant;} comanda;
typedef struct { char den[20];
                 double stoc, stoc_min, pret;} depozit;
typedef struct { char den[20];
                 double val;} factura;
int main(){
   comanda com;
   depozit dep;
   factura fac;
   double t;
   int gasit, eof;
   FILE *fc, *fd, *ff, *fc1, *fc2;
   /* deschidere fisiere */
   if((fc=fopen("comenzi.dat","rb"))==NULL){
      fprintf(stderr,"eroare deschidere comenzi\n");
      exit(1);
   };
   if((fd=fopen("depozit.dat","r+b"))==NULL){
      fprintf(stderr,"eroare deschidere depozit\n");
      exit(1);
   };
   ff=fopen("facturi.dat","wb");
   fc1=fopen("com1.dat","wb");
   fc2=fopen("com2.dat","wb");
   while(1) { /* ciclu procesare comenzi */
      if(!fread(&com, sizeof(com), 1, fc) break;
      gasit=0;
      rewind(fd);
      do {
         eof=fread(&dep, sizeof(dep), 1, fd)==0;
         if(!eof) {
            if(strcmp(com.den, dep.den)==0){
               gasit = 1;
               if((t=dep.stoc - com.cant)>=dep.stoc_min) {
                  dep.stoc=t;
                  fseek(fd, -sizeof(dep), SEEK_CUR);
                  fwrite(&dep, sizeof(dep), 1, fd);
                  fac.val=com.cant * dep.pret;
                  strcpy(fac.den, com.den);
                  fwrite(&fac, sizeof(fac), 1, ff);
               }
               else
                  fwrite(&com, sizeof(com), 1, fc1);
               break;
            }
         }
      } while(!gasit && !eof);
      if(!gasit)
```

```
            fwrite(&com, sizeof(com), 1, fc2);
      }
      fclose(fc);
      fclose(fd);
      fclose(ff);
      fclose(fc1);
      fclose(fc2);
}
```

Tabel 14.2. Funcţii utilizate în lucrul cu fişiere

Semnătură	Efect
`FILE* fopen(char* nume, char* mod);`	deschide fişierul; întoarce pointerul la fişierul deschis sau **NULL** dacă operaţia eşuează
`int fclose(FILE* pf);`	închide fişierul
`int fgetc(FILE* pf);`	citeşte 1 caracter din fişierş întoarce caracterul citit sau **EOF**
`int fputc(char c, FILE* pf);`	scrie caracterul în fişier
`char* fgets(char* s, int n, FILE* pf);`	citeşte din fişier în **s** cel mult **n–1** caractere,sau până la întâlnire **'\n'**, în locul căruia pune **'\0'**. Întoarce s sau **NULL**, dacă s-a citit **EOF**.
`int fputs(char* s, FILE* pf);`	copiază şirul în fişierul de ieşire.Înlocuieşte terminatorul **'\0'** cu **'\n'**.
`Int fscanf(FILE* pf,char* fmt, lista_adrese);`	citire din fişier sub controlul formatului. Întoarce numărul de câmpuri citite sau **EOF**, în caz de eroare sau sfârşit de fişier.
`int fprintf(FILE* pf, char* fmat,lista_expresii);`	scriere în fişier sub controlul formatului. Întoarce numărul de caractere scrise sau valoare negativă în caz de eroare.
`int fread(char* zona, int la, int na, FILE* pf);`	citeşte din fişier în **zona, na** articole de lungime **la** fiecare.Întoarce numărul de articole efectiv citite.
`int fwrite(char* zona,int la, int na, FILE* pf);`	scrie în fişier din **zona**, **na** articole de lungime **la** fiecare.Întoarce numărul de articole efectiv scrise.
`int fseek(FILE* pf,long depl, int orig);`	poziţionare cu **depl** octeţi faţă de început, poziţia curentă sau sfârşitul fişierului
`void rewind(FILE* pf);`	poziţionare la începutul fişierului
`long ftell(FILE* pf);`	determină poziţia curentă în fişier.
`int feof(FILE* pf);`	întoarce nenul dacă s-a detectat sfârşit de fişier

`int ferror(FILE* pf);`	întoarce nenul dacă s-a detectat o eroare în cursul operației de intrare / ieşire

14.4. Probleme propuse.

1. Să se scrie un program pentru creerea unui fişier binar, având articole structuri cu următoarele câmpuri:

- **`Nume`** depunător - şir de maxim 30 de caractere
- **`Data`** depunerii – o structură având câmpurile întregi: **`zi, lună, an`**.
- **`Suma`** depusă – o valoare reală.

Articolele sunt grupate pe zile în ordine cronologică. Datele se introduc de la consolă, fiecare pe trei linii.

2. Să se scrie un program care folosind fişierul creat în problema 1 calculează şi afişează:

- Suma maximă depusă, împreună cu data şi numele depunătorului.
- Numărul depunerilor din fiecare zi, şi suma totală depusă în fiecare zi, ţinând cont că tranzacțiile dintr-o zi sunt contigue în fişier.

3. Să se scrie un program pentru actualizarea fişierului creat în problema 1, pe baza unor foi de restituire, introduse de la tastatură, conținând numele şi suma solicitată. Programul semnalează la consolă următoarele situații:

- Solicitant inexistent
- Suma solicitată depăşeşte soldul.

Fişierul actualizat este listat la consolă.

4. Să se scrie un program, care folosind fişierul creat în problema 1 actualizează acest fişier prin adăugarea dobânzii la data curentă.
Se precizează următoarele date:

- Data curentă la care se calculează dobânda (an, lună, zi)
- Dobânda anuală

Se va folosi o funcție care determină numărul de zile între data depunerii şi data curent, pentru a calcula dobânda cuvenită.

1. Pentru a sorta elementele unui tablou **`T`**, fără a le deplasa, se creează un nou tablou **`P`**, în care un element $\mathbf{P}_I$ reprezintă poziția pe ar ar avea-o elementul corespunzător din **`T`** în tabloul sortat, adică numărul de elemente care ar trebui să se găsească înaintea fiecărui element din tabloul sortat.:

$$P_i : numaru(T_j \le T_i)$$
$$0 \le j \le n-1$$
$$j \ne i$$

De exemplu:

I	0	1	2	3	4
T	8	2	5	9	6
P	3	0	1	4	2
X	1	2	4	0	3

Pe baza tabloului **P** se obține relativ simplu poziția (indexul) elementelor sortate din tabloul **T**. Cel mai mic element se află în **T** în poziția **k**, astfel încât $P_k=0$, următorul – în poziția corespunzătoare lui $P_k=1$, ultimul element corespunde poziției **k** pentru care $P_k=n-1$.
Dacă tabloul **T** nu are toate elementele distincte, pentru crearea tabloului **P** se face modificarea:

$$P_i = \underset{j<i}{numaru}(T_j \le T_i) + \underset{j>i}{numaru}(T_j < T_i)$$

De exemplu:

I	0	1	2	3	4	5	6	7	8
T	8	2	5	8	5	9	2	6	5
P	6	0	2	7	3	8	1	5	4
X	1	6	2	4	8	7	0	3	5

Problema prezintă interes în cazul în care în locul tabloului **T** avem un fişier cu tip. Sortarea fişierului în raport cu o cheie (unul din câmpurile articolelor fişierului) revine la creearea unui fişier index care reprezintă un fişier de întregi (echivalent tabloului x), în care fiecare element x_I dă poziția celui de.al i.lea element din fişierul sortat în fişierul inițial.

- Să se definească o funcție, care primind ca parametru un fişier binar, creează un fişier index în raport cu o cheie.
- Să se definească o funcție care primind ca parametri un fişier şi un fişier index asociat, afişează articolele fişierului sortate în raport cu indexul dat.

6. Se dă un fişier text.
a) Să se determine numărul de linii din fişier.
b) Să se creeze un nou fişier cu liniile din primul, apărând în ordine inversă.
c) Pe baza fişierului inițial, să se creeze un fişier de caractere, în care nu mai apar caracterele de sfârşit de linie, iar fiecare linie este precedată de lungimea ei (un octet)
d) Se dă un fişier de întregi reprezentând numere de linii din fişierul inițial. Să se afişeze la imprimantă liniile din fişier în ordinea precizată de fişierul de întregi.

7. Fişierul text **prog.c** reprezintă un program sursă C. Să se copieze acest fişier la ieşirea standard suprimând toate comentariile.

8. Se consideră fişierul **abonați.dat** cu articole structuri având câmpurile:

- **Nume** – un şir de 20 de caractere

- **`Adresă`** – un şir de 30 de caractere
- **`Data_expirării`** – o structură cu câmpurile **`an, lună, zi`**.
  Considerăm că data curentă se introduce de la tastatură.

Să se actualizeze fişierul de abonaţi, ştergând pe aceia al căror abonament a expirat la data curentă. Actualizarea se face creind un nou fişier în care se trec numai abonaţii al căror abonament nu a expirat şi care la sfârşit va primi numele fişierului iniţial.
Se va defini şi folosi o funcţie care compară două date (**`d1 şi d2`**) şi întoarce **`1`** , dacă **`d1`** este înaintea lui **`d2`**, şi **`0`** în caz contrar.

Un medicament este specificat printr-o structură care conţine: denumirea comercială internaţională (**dci**) – un şir de 20 caractere şi **cantitate** – o valoare reală.
O reţetă compensată conţine: **nume** pacient – un şir de 20 de caractere, **`n`** - numărul de medicamente prescrise şi **`n`** structuri de tip medicament.
Farmacistul înlocuieşte **dci** cu echivalentul medicamentului produs în ţară, având cel mai mic preţ pe care îl are în stoc şi eliberează un bon care conţine: numele pacientului, numărul de medicamente eliberate, şi pentru fiecare medicament: numele echivalent autohton, valoarea medicamentului şi la final, valoarea totală de plată.
Scrieţi un program care automatizează aceste operaţii. Se precizează următoarele:
- reţetele sunt citite dintr-un fişier binar, cu numele dat ca parametru al comenzii
- echivalenţa dci – medicament autohton este dată într-un fişier binar, ce conţine:
  - **`dci`** – şir de 20 caractere
  - **`ne`** – numărul de medicamente echivalente

şi pentru fiecare medicament echivalent: numele medicamentului – 20 caractere şi preţ – valoare reală.
Dacă **`dci`** nu se găseşte în fişierul de echivalenţe, reţeta nu poate fi onorată, fapt consemnat printr-un mesaj la dispozitivul standard de eroare.
Dacă se găsesc echivalente pentru toate medicamentele din reţetă se crează un bon scris în fişierul de bonuri.
Numele celor 3 fişiere binare: fişierul de reţete, fişierul de echivalenţe şi fişierul de bonuri se dau ca parametri ai comenzii.

O comandă pentru un produs conţine nume **`produs`** (20 caractere) şi **`cantitate`** (real).
Un client este identificat prin: **`nume`** client (30 caractere) şi număr produse comandate **`npc`** (întreg) şi **`npc`** comenzi. Comenzile clienţilor sunt date într-un fişier binar.
Pentru satisfacerea acestor comenzi se consultă un fişier catalog ce conţine articole de forma: nume **`furnizor`** (30 caractere), nume **`produs`** (20 caractere), **`preţ`** unitar (real).
Scrieţi un program care caută să satisfacă comenzile clienţilor folosind produsele din catalog cu preţurile cele mai mici.
Programul va crea un fişier de facturi, în care pentru fiecare comandă apare un articol de forma: nume client, număr produse livrate şi suma totală de plată
Numele fişierelor se preiau ca parametri ai comenzii.

Un **produs** este precizat printr-o structură care conţine: **nume** produs (20 caractere) şi **cantitate** (real)

O firmă efectuează două tipuri de tranzacții: aprovizionări şi vânzări. O tranzacție este specificată prin:

- **tip** tranzacție (un caracter 'A' sau 'V')
- **nume** furnizor sau client (30 caractere)
- **npc** - număr produse comandate (întreg)

şi **npc** structuri **produs**.
In cazul unei aprovizionări mai există un câmp - **preț** unitar, care este dictat de furnizor
Pentru fiecare tranzacție, fiecare produs este căutat după nume într-un fişier de stocuri. Fişierul de stocuri conține articole cu lungime fixată, cu structura: nume produs (20 caractere), stoc (real), stoc minim (real) şi preț unitar (real)
Pentru o aprovizionare, dacă produsul este găsit în fişierul de stocuri, cantitatea este adăugată la cea existentă în stoc, iar prețul întregului stoc se modifică la prețul unitar al produsului din această tranzacție.
Dacă produsul nu este găsit, el este adăugat la sfârşitul fişierului de stocuri, cu stoc minim zero.
Pentru o vânzare, dacă produsul este găsit în fişierul de stocuri, şi este în cantitate suficientă pentru a satisface comanda fără a scădea sub stocul minim, comanda pentru acel produs este satisfăcută, şi stocul este actualizat.
O comandă de vânzare poate fi satisfăcută total, pentru toate produsele solicitate, sau parțial, numai pentru produsele existente în stoc în cantități suficiente.
Folosind un fişier binar de comenzi se cere:
1)să se calculeze câştigul sau datoria rezultată din desfăşurarea acestor tranzacții.
O aprovizionare va apare cu o valoare negativă (o datorie), iar o vânzare cu o valoare pozitivă (un câştig).
2)să se actualizeze fişierul de stocuri conform tranzacțiilor
3)să se creeze un fişier de facturi de încasat pentru tranzacțiile vânzări. Intr-o factură apar: nume client (30 caractere), număr produse livrate (întreg), numele produselor livrate(câte 20 caractere) şi valoare de încasat (real)
Un client are o singură comandă (pentru mai multe produse). Numele fişierelor sunt preluate din linia de comandă

Implementați comanda **medie**, care calculează media aritmetică a unor valori reale, introduse din diferite surse. Comanda poate avea una din formele:

```
medie
medie nf
medie n1  n2  … np
```

În primul caz, datele sunt citite de la tastatură, dintr-o singură linie, fiind separate prin spații albe, iar rezultatul este afişat pe ecran.
În a doua situație, datele sunt preluate dintr-un fişier text, cu numele dat ca argument al comenzii. Media numerelor din fiecare linie este scrisă într-un fişier binar, cu acelaşi nume cu fişierul de intrare, dar cu extensia **`.bin.`**
În ultimul caz, datele sunt preluate ca parametri ai comenzii, iar rezultatul este afişat pe ecran. Menționăm că, în acest ultim caz există cel puțin două argumente.
Se recomandă definirea şi folosirea unei funcții **`double med(char *s);`** care separă numerele reale din şirul de caractere s, le converteşte şi calculează media lor aritmetică. Pentru conversie se poate folosi funcția: **`double atof(char *s);`** din stdlib.h care transformă şirul într-un real.

Utilizând un card de credit putem realiza atât operații de restituire cât şi de depunere. Definiți tipul **card** ca o structură conținînd următoarele câmpuri::
pin –un întreg reprezentând parola de acces a posesorului cardului
operație –un caracter (R = restituire / D = depunere)
suma –un întreg lung reprezentând suma solicitată sau depusă
Scrieți un program care simulează funcționarea unui bancomat, preluând tranzacțiile (restituirile sau depunerile) dintr-un fişier binar de articole de tip card şi actualizează o bază de date conturi ale clienților. Acesta este tot un fişier binar, cu articole conținând: codul pin, contul clientului, datoria maximă pe care o poate face clientul.
Pentru păstrarea confidențialității numele clienților sunt păstrate într-un fişier binar, care stabileşte corespondența cod pin – nume client..
Programul crează şi un bon care notifică tranzacția. Acesta va conține numele clientului, tipul operației, suma primită (sau depusă) şi contul actualizat, şi reprezintă un fişier binar cu articole având câmpurile menționate.
În cazul unei operații de restituire, clientul nu poate retrage o sumă care să facă datoria sa față de bancă, mai mare decât cea specificată. Astfel dacă clientul are în cont 10.000 RON şi poate avea o datorie de 15.000 RON, atunci el nu poate retrage mai mult de 25.000 RON.
Numele celor 4 fişiere (tranzacții, conturi, corespondențe şi bonuri) sunt preluate ca parametri ai comenzii.
Dacă operația nu poate fi efectuată se afişează un mesaj de eroare, în care apare numele clientului.

Dintr-un fişier text se citesc mai multe matrice pătrate. O matrice este reprezentată pe o linie din fişier prin dimensiunea **n** şi cele n^2 elemente reale, considerate în ordinea parcurgerii pe linii a matricei, separate între ele prin spații libere.
Creați un fişier binar, cu acelaşi nume cu fişierul de intrare, dar cu extensia **.ort**, conținând numai matricele ortogonale.(O matrice ortogonală satisface proprietatea $A.A^T=I_n$, unde I_n este matricea unitate de dimensiune **n**).
Numele fişierului text de intrare este dat ca parametru al comenzii.
La o bancă, pentru efectuarea unei operații (depunere sau restituire) clientul completează un formular în care trece: numele, tipul operației şi suma. Pentru păstrarea secretului operațiilor banca foloseşte în locul numelor clienților, coduri. Corespondențele: nume client – cod şi cod – cont sunt păstrate în fişiere binare. Tranzacțiile solicitate de clienți se introduc prin intermediul unui fişier text, în care o linie conține: numele clientului (scris cu nume şi prenume separate prin _), tipul operației se specifică printr-un caracter **R** pentru restituire sau **D** pentru depunere şi suma – o valoare reală. Aceste 3 elemente sunt separate în linie prin spații.
In fişierele binare câmpul nume are 20 de caractere, câmpul cod este un întreg lung, iar câmpul cont este un real.
Scrieți un program care prelucrează tranzacțiile şi actualizează conturile clienților. Pentru o depunere a unui client care nu are cont, i se asociază un cont, se adaugă perechea nume – cod în primul fişier de corespondențe şi perechea cod – sumă în cel de-al doilea fişier binar. O restituire nu este făcută şi se dă un mesaj de eroare în următoarele situații: 1.solicitantul nu există în primul fişier de corespondențe sau 2.suma solicitată depăşeşte contul.
Numele celor două fişiere de corespondențe se dau ca parametri ai comenzii. Codul disponibil asociat unui client nou este păstrat într-o variabilă întreagă statică, şi este incrementat pentru fiecare client nou adăugat.

Implementați comanda **`indent numef`** , care citeşte un fişier XML cu numele **`numef,`** corect sintactic şi-l afişează fără marcaje, cu indentare (textul cuprins între două marcaje va fi afişat pe o linie nouă, decalat la dreapta cu 2 spații libere.
Un fişier XML este un fişier text care conține diverse şiruri de caractere încadrate de marcaje. Un marcaj ("tag") este un şir între parantezele ascuțite '**<**' şi '**>**'. Marcajele se folosesc în perechi: un marcaj de început (ex: **`<a>`**) şi un marcaj de sfârşit (ex: **`</a>`**). Perechile de marcaje pot fi incluse unele în altele.
Exemplu: **`<a> ... <b> ... </b> ... </a>.`** Nu sunt permise construcții de forma:
`<a> ... <b> ... </a> ... </b>`
De exemplu fişierul: **`<a>T1<b>T2</b><c>T3<d>T4</d>T5</c>T6</a>`** va apare afişat ca:

```
T1
  T2
  T3
    T4
  T5
T6
```

Indicație: Se copiază fişierul în memorie. Intr-un ciclu se repetă operațiile:

- caută un marcaj (care începe cu '<')
- dacă este început de marcaj (următorul caracter nu este '/') se decalează poziția de afişare cu 2 poziții la dreapta
- dacă este sfârşit de marcaj se decalează poziția de afişare cu 2 poziții la stânga
- se sare peste marcaj, inclusiv peste >
- se afişează textul până la următorul marcaj (textul cuprins între '<' şi '>')

Se recomandă definirea unei funcții care afişează cu decalaj dat caractere între două adrese şi o funcție care determină lungimea unui fişier text.

Un fişier binar conține mai multe matrice, date prin numărul de linii **`m`**, coloane **`n`** şi cele **`m x n`** elemente reale, considerate în ordinea liniilor.
Se cere să se creeze un nou fişier binar, în care toate matricele de aceeaşi dimensiune să fie înlocuite cu suma lor.
În acest scop, se creează în memorie un tablou de matrice sume: la citirea unei matrice din fişierul binar se caută în acest tablou pentru a vedea dacă mai există o matrice cu aceleaşi dimensiuni – în caz afirmativ se adună la acea matrice, matricea citită din fişier, altfel se creează o nouă intrare în tablou cu matricea citită.
Numele celor două fişiere sunt date ca parametrii ai comenzii. Nu se admite citirea integrală a fişierului de intrare în memorie.

Dintr-un fisier text, să se extragă cuvintele cu lungimea cuprinsă intre m şi n, având cel putin p aparitii, şi să se plaseze într-un fişier binar.
Un articol din fişierul binar va contine următoarele informatii asupra unui cuvânt extras:
- `l` = lungimea cuvântului (întreg) `m <= l <= n <= 15`
- `cuv`= tabloul continând cuvântul extras (tablou de 15 caractere)
- `ap` = numărul de aparitii a cuvântului (întreg) ap >= p

`m, n, p` şi numele fişierului text sunt preluate ca parametri ai comenzii. (numele fişierului text poate avea sau nu extensie).
Fişierul binar creat va avea acelaşi nume ca şi fişierul text, dar extensia `.idx` .

Indicație: Întrucât numărul de aparitii al unui cuvânt este cunoscut numai dupa citirea întregului fişier text, care, în mod categoric, ***nu va putea fi pastrat în memorie***, vom crea un fişier ajutător care va contine articole de forma:

```
struct art{
    int l;         // lungime cuvânt  (m<=l<=15)
    char cuv[15]; //cuvântul extras
};
```

Fiecare cuvânt din fişierul temporar este căutat în fişierul de ieşire.Dacă se află acolo, înseamnă că a fost prelucrat şi se trece mai departe; în caz contrar, se numără aparițiile lui în fişierul temporar şi se trece în fişierul de ieşire, împreună cu numărul de aparții. În final, se scot din fişierul de ieşire articolele cu mai puțin de p aparții.