**Ministerul Educației și Cercetării**

|  |
| --- |
| **Universitatea Tehnica a Moldovei** |

**Facultatea Calculatoare, Informatica si Microelectronica**

**Structuri de date și Algoritmi**

**Raport**

|  |
| --- |
| Lucrarea de Laborator nr. 2  Tema: Algoritmi de prelucrare a fișierelor binare |

|  |  |
| --- | --- |
| A efectuat stundent(ul/a) grupei: | SI-212 |
| Numele si prenumele elev(ului/ei): | Vozian Vladimir |
|  |  |
| Profesorul: | Mititelu Vitalie |

|  |
| --- |
| Chișinău 2022 |

**Scopul lucrării:**

Programarea algoritmilor de prelucrare a fișierelor binare prin utilizarea structurilor, funcțiilor, pointerilor, alocării dinamice a memoriei în limbajul C.

**Sarcina:**

Pentru fișierele binare cu elemente de tip structură (conform variantelor) să se afișeze la ecran următorul meniu de opțiuni:

1. Crearea unui fișier binar nou, introducerea câmpurilor structurilor de la tastatură și înscrierea lor în acest fișier.

2. Afișarea elementelor fișierului binar la ecran.

3. Adăugarea unei structuri noi la sfârșit de fișier.

4. Modificarea unei structuri a fișierului.

5. Căutarea după un câmp al structurii în fișier.

6. Sortarea structurilor fișierului după un careva câmp.

7. Eliminarea unei structuri din fișier.

8. Ștergerea fișierului de pe disc (la dorință)

0. Ieșire din program.

Să se elaboreze funcțiile pentru implementarea opțiunilor meniului.

**Varianta:**

16. Structura Canal TV cu câmpurile: denumirea, țara, telefonul, genul, ratingul.

**Rezumat**

Un fişier este o structură dinamică, situată în memoria secundară (pe disk-uri). Limbajul C permite operarea cu fişiere:

* de **tip text** - un astfel de fişier conţine o succesiune de **linii**, separate prin new line ('\n')
* de **tip binar** - un astfel de fişier conţine o succesiune de octeti, **fără nici o structură**.

Prelucrarea unui fişier presupune asocierea acestuia cu un canal de I/E (numit flux sau stream). Există trei canale predefinite, care se deschid automat la lansarea unui program:

* **stdin** - fişier de intrare, text, este intrarea standard - tastatura
* **stdout** - fişier de iesire, text, este ieşirea standard - ecranul monitorului.
* **stderr** – fişier de iesire, text, este ieşirea standard unde sunt scris mesajele de eroare - ecran.

Pentru a prelucra un fişier, trebuie parcurse următoarele etape:

* **se defineşte** o variabilă de tip **FILE**\* pentru accesarea fişierului; **FILE** este un tip structură definit în <stdio.h>, care conţine informaţii referitoare la fişier şi la tamponul de transfer de date între memoria centrală şi fişier (adresa, lungimea tamponului, modul de utilizare a fişierului, indicator de sfârsit, de poziţie în fişier). Puteți citi mai multe [aici](http://stackoverflow.com/questions/5672746/what-exactly-is-the-file-keyword-in-c).
* **se deschide fişieru**l pentru un anumit **mod de acces**, folosind funcţia de bibliotecă **fopen**, care realizează şi asocierea între variabila fişier şi numele extern al fişierului
* **se prelucrează fişierul** în citire/scriere cu **funcţiile specifice**
* **se închide fişierul** folosind funcţia de bibliotecă **fclose**

#### **Funcții**

Mai jos se prezintă restul funcţiilor de prelucrare a fişierelor. Pentru documentația oficială puteți citi [aici](http://www.cplusplus.com/reference/cstdio/).

##### fopen

FILE \*[fopen](http://www.opengroup.org/onlinepubs/009695399/functions/fopen.html)(const char \*filename, const char \*mod);

deschide fişierul cu numele **filename** pentru acces de tip **mod**.

Returnează **pointer la fişier** sau **NULL** dacă fişierul nu poate fi deschis; valoarea returnată este memorată în variabila fişier, care a fost declarată pentru accesarea lui.

Modul de deschidere poate fi:

* “**r**” - **readonly** , este permisă doar citirea dintr-un fişier existent
* “**w**” - **write**, crează un nou fişier, sau dacă există deja, distruge vechiul continut
* “**a**” - **append**, deschide pentru scriere un fişier existent ( scrierea se va face în continuarea

informaţiei deja existente în fişier, deci pointerul de acces se plasează la sfârşitul fişierului )

* “**+**” - permite scrierea şi citirea - **actualizare** (ex: “r+”, “w+”, “a+”). Între read şi write trebuie repoziţionat cursorul de acces printr-un apel la **fseek**.
* “**b**” - specifică fişier de tip **binar**
* “**t**” - specifică fişier de tip **text** (implicit), la care se face automat conversia CR-LF(“\n\f”) în sau din CR ('\n').

##### fclose

int [fclose](http://www.opengroup.org/onlinepubs/009695399/functions/fclose.html)(FILE \*pFile);

**închide fişierul** asociat cu variabila **pFile** şi eliberează zona tampon; returnează 0 la succes, EOF (end of file) la eroare

##### fseek

int [fseek](http://www.opengroup.org/onlinepubs/009695399/functions/fseek.html)(FILE \*pFile, long offset, int whence);

**repoziţionează pointerul** asociat fişierului **pFile**; offset - numărul de octeţi între poziţia dată de whence şi noua poziţie.

whence - are una din cele trei valori posibile:

* SEEK\_SET = 0 - Căutarea se face de la începutul fişierului
* SEEK\_CUR = 1 - Căutare din poziţia curentă
* SEEK\_END = 2 - Căutare de la sfârşitul fişierului

##### ftell

long [ftell](http://www.opengroup.org/onlinepubs/009695399/functions/ftell.html)(FILE \*pFile);

**întoarce poziţia curentă** în cadrul fișierului asociat cu pFile.

##### fgetpos

int [fgetpos](http://www.opengroup.org/onlinepubs/009695399/functions/fgetpos.html)(FILE \*pFile, fpos\_t \*ptr);

această funcţie **memorează poziţia curentă** în variabila ptr în cadrul fişierului asociat cu pFile (ptr va putea fi folosit ulterior cu funcţia fsetpos).

##### fsetpos

int [fsetpos](http://www.opengroup.org/onlinepubs/009695399/functions/fsetpos.html)(FILE \*pFile, const fpos\_t \*ptr);

această funcţie **setează poziţia curentă** în fişierul asociat cu pFile la valoarea ptr, obţinută anterior prin funcţia fgetpos.

##### feof

int [feof](http://www.opengroup.org/onlinepubs/009695399/functions/feof.html)(FILE \*fis);

returnează 0 dacă nu s-a detectat **sfârşit de fişier** la ultima operaţie de citire, respectiv o valoare nenulă (adevărată) pentru sfârşit de fişier.

##### freopen

FILE\* [freopen](http://www.opengroup.org/onlinepubs/009695399/functions/freopen.html)(const char \*filename, const char \*mode, FILE \*fp);

se închide fişierul fp, se deschide fişierul cu numele filename în modul mode şi acesta se asociază la fp; se întoarce fp sau NULL în caz de eroare.

##### fflush

int [fflush](http://www.opengroup.org/onlinepubs/009695399/functions/fflush.html)(FILE \*fp);

Această funcţie se utilizează pentru fişierele deschise pentru scriere şi are ca efect scrierea în fişier a datelor din bufferul asociat acestuia, care înca nu au fost puse în fişier.

**Codul deplin al programului:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct CanalTV{

    char denumire[20];

    char tara[10];

    unsigned long telefon;

    char gen[15];

    float rating;

}CTV;

CTV \*pCanale = NULL;

int size = 0;

int firstRun = 1;

char \*fisier = "canale.dat";

int Meniu(){

    if(firstRun)

        firstRun = 0;

    else{

        system("pause");

        system("cls");

    }

    printf("Alocare memorie...............1\n");

    printf("Citire canale.................2\n");

    printf("Afisare lista danale..........3\n");

    printf("Eliberare memorie.............4\n");

    printf("Adaugare canal................5\n");

    printf("Eliminare canal...............6\n");

    printf("Sortare lista de canale.......7\n");

    printf("Salvare pointer in fisier.....8\n");

    printf("Incarcare fisier in pointer...9\n");

    printf("Iesire din program............0\n");

    int menu;

    scanf("%d", &menu);

    return menu;

}

void AlocMem(){

    char rasp;

    if(size != 0){

        printf("Lista canalelor va fi stearsa irevocabil. Continuati?[d/n]\n");

        rasp = getche();

        if(rasp == 'd')

            ElibMem();

        else{

            printf("\nBine, datele nu vor fi sterse.\n");

            return;

        }

    }

    do{

        printf("Introduceti numarul canalelor: ");

        scanf("%d", &size);

    }while(size <= 0);

    pCanale = (CTV \*)malloc(size \* sizeof(CTV));

    if(pCanale == NULL){

        printf("\nNu s-a reusit alocarea memorie. Se va realiza iesirea din program!\n");

        exit(1);

    }else

        printf("\nA fost alocata memorie pentru %d canale.\n", size);

}

void ElibMem(){

    if(pCanale){

        free(pCanale);

        pCanale = NULL;

        size = 0;

        printf("A fost eliberata memoria alocata pentru pointerul la canale.\n");

    }

    else

        printf("Nu a fost alocata memorie, nu este ce elibera!\n");

}

void Citire(){

    if(size == 0)

        printf("Nu a fost alocata memorie. Nu este posibil de citit datele despre canale.\n");

    else{

        printf("Introduceti datele despre %d canale: \n", size);

        for(int i=0; i<size; i++){

            printf("Canalul %d:\n", i+1);

            printf("Denumirea: "); scanf("%s", pCanale[i].denumire);

            printf("Tara: "); scanf("%s", &pCanale[i].tara);

            printf("Telefon: "); scanf("%lu", &pCanale[i].telefon);

            printf("Gen: "); scanf("%s", &pCanale[i].gen);

            printf("Rating: "); scanf("%f", &pCanale[i].rating);

            if (!(0<=pCanale[i].rating && pCanale[i].rating<=10)){

                while(!(0<=pCanale[i].rating && pCanale[i].rating<=10)){

                printf("Rating gresit!\nRating:");

                scanf("%f" ,&pCanale[i].rating);

                }

            }

        }

        printf("Au fost citite datele despre %d canale.\n", size);

    }

}

void Afisare(){

    if(size == 0)

        printf("Nu a fost alocata memorie. Nu este posibil de afisat datele despre canale.\n");

    else{

        printf("Lista canalelor: \n");

        for(int i=0; i<size; i++){

            printf("%2d.| %15s.| Tara: %10s.| Telefon: %10lu.| Gen: %15s.| Rating: %4.2f.|\n", i+1, pCanale[i].denumire, pCanale[i].tara, pCanale[i].telefon, pCanale[i].gen, pCanale[i].rating);

        }

    }

}

void AdaugCanal(){

    size++;

    pCanale = (CTV \*)realloc(pCanale, size \* sizeof(CTV));

    printf("Introduceti datele despre canalul nou: \n", size);

    printf("Denumirea: "); scanf("%s", pCanale[size - 1].denumire);

    printf("Tara: "); scanf("%s", &pCanale[size - 1].tara);

    printf("Telefon: "); scanf("%lu", &pCanale[size - 1].telefon);

    printf("Gen: "); scanf("%s", &pCanale[size - 1].gen);

    printf("Rating: "); scanf("%f", &pCanale[size - 1].rating);

    //verificarea ratingului

    if (!(0<=pCanale[size - 1].rating && pCanale[size - 1].rating<=10)){

        while(!(0<=pCanale[size - 1].rating && pCanale[size - 1].rating<=10)){

            printf("Rating gresit!\nRating:");

            scanf("%f" ,&pCanale[size - 1].rating);

            }

        }

    printf("Au fost citite datele canalului nou.\n");

}

void ElimCanal(){

    char denumire[20];

    if(size == 0)

        printf("Nu ati alocat memorie! Nu avem ce elimina!\n");

    else if(size == 1){

        printf("Lista contine doar un canal (%s). Dupa eliminarea acestuia va deveni vida!\n", pCanale->denumire);

        ElibMem();

    }

    else{

        Afisare();

        int n;

        printf("Introduceti numarul canalului pe care il eliminati: ");

        scanf("%d", &n);

        n--;

        if(n < 0 || n >= size){

            printf("Numar incorect!\n");

            return;

        }

        strcpy(denumire, pCanale[n].denumire);

        for(int i=n; i<size-1; i++)

            pCanale[i] = pCanale[i+1];

        size--;

        pCanale = (CTV \*)realloc(pCanale, size \* sizeof(CTV));

        printf("Canalul \"%s\" a fost eliminat!\n", denumire);

    }

}

void SortByDenumire(){

    if(size == 0){

        printf("Nu ati alocat memorie! Nu avem ce sorta!\n");

        return;

    }

    CTV temp;

    int sortat;

    do{

        sortat = 1;

        for(int i=0; i<size-1; i++)

            if(strcmp(pCanale[i].denumire, pCanale[i+1].denumire) > 0){

                sortat = 0;

                temp = pCanale[i];

                pCanale[i] = pCanale[i+1];

                pCanale[i+1]= temp;

            }

    }while(sortat == 0);

    printf("Lista canalelor a fost sortata in ordine alfabetica dupa denumire.\n");

    Afisare();

}

void SortByRatingDesresc(){

    if(size == 0){

        printf("Nu ati alocat memorie! Nu avem ce sorta!\n");

        return;

    }

    CTV temp;

    int sortat;

    do{

        sortat = 1;

        for(int i=0; i<size-1; i++)

            if(pCanale[i].rating < pCanale[i+1].rating){

                sortat = 0;

                temp = pCanale[i];

                pCanale[i] = pCanale[i+1];

                pCanale[i+1] = temp;

            }

    }while(sortat == 0);

    printf("Lista canalelor a fost sortata in ordine crescatoare.\n");

    Afisare();

}

void SortByRatingCresc(){

    if(size == 0){

        printf("Nu ati alocat memorie! Nu avem ce sorta!\n");

        return;

    }

    CTV temp;

    int sortat;

    do{

        sortat = 1;

        for(int i=0; i<size-1; i++)

            if(pCanale[i].rating > pCanale[i+1].rating){

                sortat = 0;

                temp = pCanale[i];

                pCanale[i] = pCanale[i+1];

                pCanale[i+1] = temp;

            }

    }while(sortat == 0);

    printf("Lista canalelor a fost sortata in ordine descrescatoare.\n");

    Afisare();

}

void PointToFis(){

    FILE \*f;//0. Declararea unei variabile de tip pointer la fisier

    int i;

    f = fopen(fisier, "wb");//1. Deschiderea fisierului binar pentru a scrie in el

    for(i=0; i<size; i++)

        fwrite(&pCanale[i], sizeof(CTV), 1, f);//2. Scrierea datelor din pointer in fisierul binar

    fclose(f);//3. Inchiderea fisierului

    printf("Datele despre %d canale au fost inscrise in fisier.\n", i);

}

void FisToPoint(){

    FILE \*f;

    CTV stud;

    int i = 0;

    f = fopen(fisier, "rb");//1. Deschiderea fisierului binar pentru a citi din el

    if(f == NULL){

        printf("Fisierul nu exista! Mai intai creati-l!\n");

        return;

    }

    //printf("Lista studentilor: \n");

    while(fread(&stud, sizeof(CTV), 1, f))//2. Citirea datelor despre canal din fisier

        i++;

    size = i;

    ElibMem();

    pCanale = (CTV \*)malloc(size \* sizeof(CTV));

    fseek(f, 0, 0);//Ne intoarcem la inceputul fisierului

    i = 0;

    while(fread(&pCanale[i], sizeof(CTV), 1, f))

        i++;

    fclose(f);//3. Inchiderea fisierului

    printf("Datele despre %d canale au fost transferate din fisier in pointer!\n", i);

}

int subMeniu(){

    int m;

    printf("Dupa denumire....1\n");

    printf("Dupa rating....2\n");

    scanf("%d", &m);

    return m;

}

int main()

{

    int menu, smenu, sm\_rasp;

    char rasp;

    printf("Doriti sa incarcati datele din fisier in pointer?[d/n]-> ");

    rasp = getche();

    printf("\n");

    if(rasp == 'd')

        FisToPoint();

    do{

        menu = Meniu();

        switch(menu){

            case 1: AlocMem(); //break;

            case 2: Citire(); break;

            case 3: Afisare(); break;

            case 4: ElibMem(); break;

            case 5: AdaugCanal(); break;

            case 6: ElimCanal(); break;

            case 7:

                    smenu = subMeniu();

                    if(smenu == 1) SortByDenumire();

                    else if(smenu == 2){

                        printf("Cum doriti sa ordonati lista?\n1. Crescator;\n2. Descrescator;\n");

                        scanf("%d", &sm\_rasp);

                        if(sm\_rasp == 1) SortByRatingCresc();

                        else if(sm\_rasp == 2) SortByRatingDesresc();

                        else printf("Nu este asa optiune.\n");

                    }

                    else printf("Criteriu incorect!\n");

            break;

            case 8: PointToFis(); break;

            case 9: FisToPoint(); break;

            case 0:

                printf("Salvati datele din pointer in fisier?[d/n]-> ");

                rasp = getche();

                printf("\n");

                if(rasp == 'd')

                    PointToFis();

                ElibMem();

                break;

            default: printf("Optiune incorecta! Mai incercati!\n");

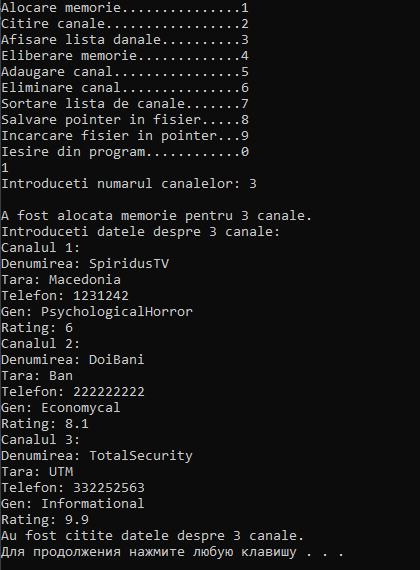
        }

    }while(menu != 0);

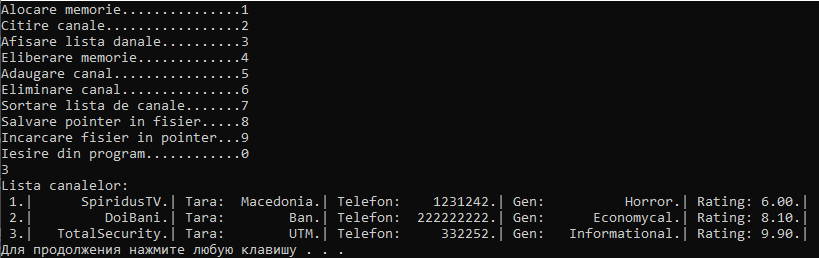
    return 0;

}

**Exemple de execuție a programului:**



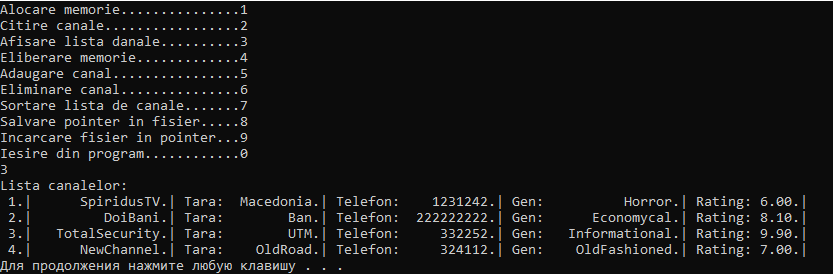
Aici este arătată alocarea memoriei și citirea de la tastatura.



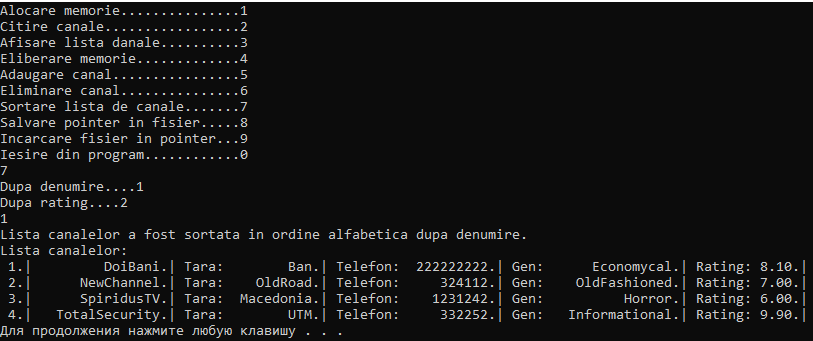
Afișarea elementelor



Adaugarea canalului.

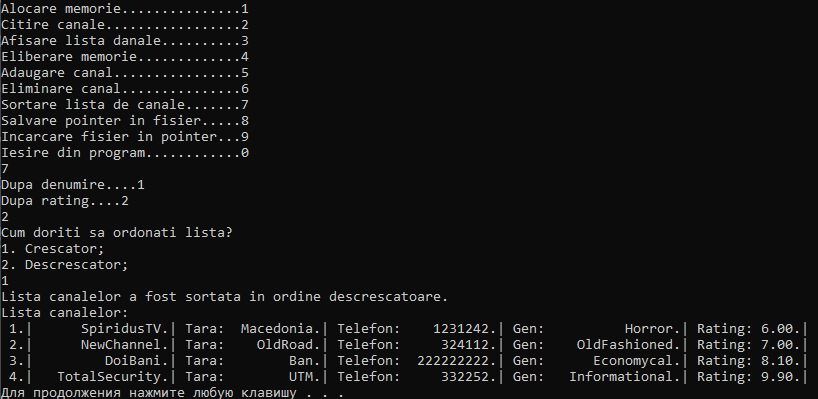


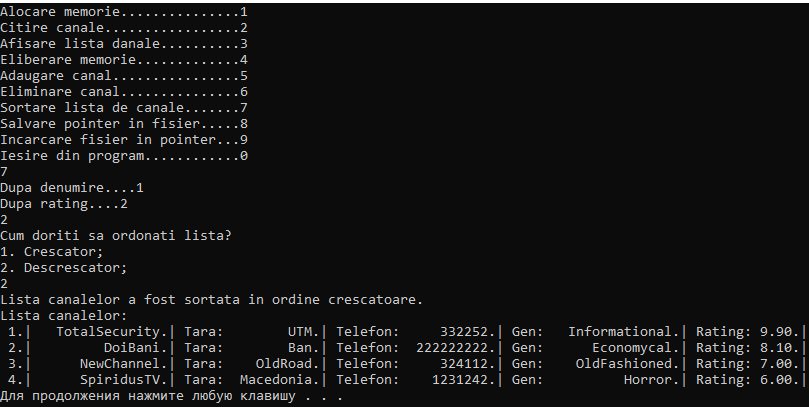
Afisarea canalelor impreuna cu cel adaugat.

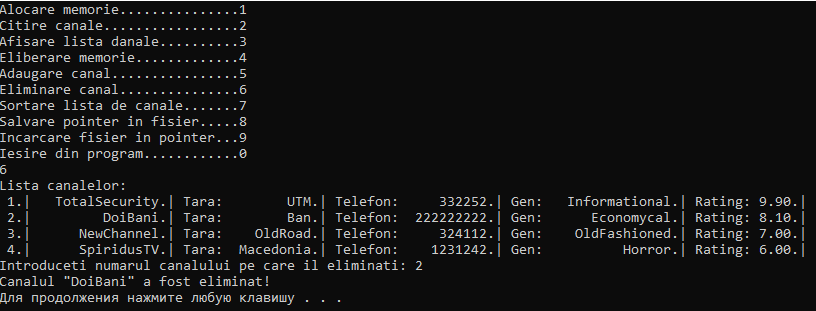


Sortarea sa impartit in 2 variante la alegere, dupa alfabet si dupa rating, aici este varianta sortarii dupa denumire.

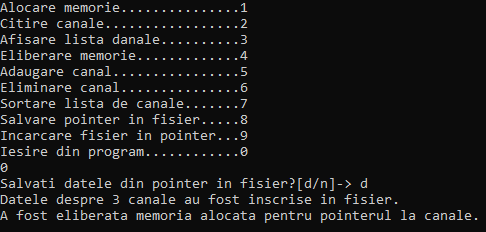
Sortarea deja după rating se poate alege in 2 moduri, crescator si descrescator. Și ambele exemple.





Eliminarea canalului:

Și deja salvarea datelor în fișier la ieșirea din program.



**Concluzii pe baza efectuării lucrării de laborator:**

* Lucrul cu fișiere și implementarea lor în program este foarte utilă datorită posibilității stocării datelor prelucrate, introduse, modificate într-un file care salveaza și conține informația pana la următoarea folosire a file-ului.
* Datele nu sunt deja de o singură folosință ca înainte, și operarea cu informația nu mai este limitată de timp, și nu are necesitate ca programul sa fie mereu deschis pentru stocarea datelor.
* Deci introducerea în lucrul cu fișiere este foarte util pentru a scoate limitele de operare cu informația, la fel fișierele sunt utile pentru multiple testări pentru a modifica ușor programul, și având deja informația in fișier se pot face testarile cum se schimba datele cu schimbul programului.
* Acest program a fost creat cu ajutorul programelor metodice a profesorului Mititelu Vitalie, fiind refăcută și performată.

**Surse**

<https://ocw.cs.pub.ro/courses/programare/laboratoare/lab12>