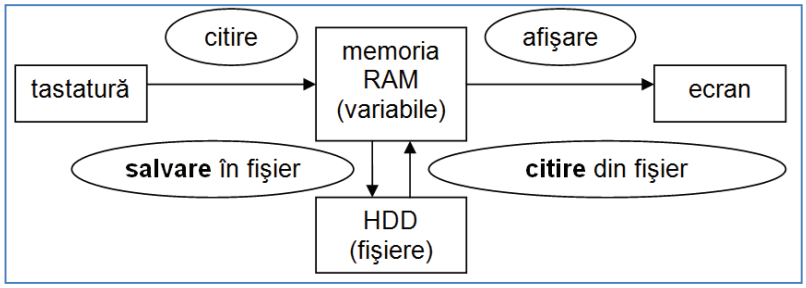
1. Fişiere text

1.Obiective

- Utilizarea principalelor funcţii specifice fişierelor text;
- Implementarea unei aplicaţii de salvare/citire a datelor matriceale în/din fişierele text;
- Generarea fişierelor html şi a fişierelor istoric în cadrul aplicaţiei.

2.Aspecte teoretice

Memoria internă este limitată atât din punctul de vedere al **persistenţei datelor** cât şi din punctul de vedere al capacităţii. Posibilitatea de a stoca şi accesa cantităţi mari de date este oferită de dispozitivele de **stocare externe** (hard-disk, DVD, flash). Din punct de vedere logic, aceste dispozitive organizează datele sub forma **fişierelor**. Transferul datelor din memoria internă într-un fişier poartă denumirea de **salvarea** datelor. Transferul invers presupune **citirea din fişier.**



Funcţii noi			
deschidere	`fopen`		
închidere	`fclose`		
	caracter	şir de caractere	date formatate
scriere	`fputc`	`fgets`	`fscanf`
citire	`fgetc`	`fputs`	`fprintf`

3.Exemple

Exemplul 1 – salvarea/citirea unui vector în/din fişier text

Un program minimal ce operează cu şiruri de numere întregi trebuie să implementeze funcţiile de citire de la tastatură şi afişare pe ecran. Utilizând fişiere se pot implementa încă două funcţii, una care să realizeze salvarea unui vector într-un fişier iar alta citirea unui vector din fişier.

Trebuie respectat un anumit **format al fişierului**. Convenim ca să scriem pe prima linie a fişierului numărul de elemente ale şirului iar pe linia următoare elementele, separate prin spaţiu. De exemplu, după salvarea vectorului `{2, -3, 0, 8}`, fişierul trebuie să conţină:

```
4
2 -3 0 8
```

Funcţia de salvare a vectorului are ca argumente numărul de elemente - **n**, vectorul - **v** şi numele fişierului în care se doreşte salvarea vectorului - **numeFisier**. Această funcţie returnează valoarea **1 (unu)** dacă nu au apărut erori la deschiderea fişierului şi **0 (zero)** în caz contrar.

```
int salvareVector(int n, int v[NMAX], char* numeFisier){
      FILE *pf; //pf este o variabila de tip pointer catre un fisier
      int i;
      //se deschide fisierul in mod scriere (write)
      pf = fopen (numeFisier, "wt");
      //daca fisierul nu poate fi deschis, se returneaza fals logic
      if (pf == NULL)
            return 0;
      //daca pf este un pointer valid atunci se pot scrie datele
      fprintf (pf, "%d\n",n); //se scrie numarul de elemente
      for (i=0;i<n;i++)       //se scriu succesiv elementele
            fprintf (pf, "%d ",v[i]);
      fclose (pf); //inchidere fisier
      return 1;    //salvarea datelor s-a incheiat cu succes
}
```

Funcţia de citire din fişier a vectorului are ca argumente un pointer la numărul de elemente - **n**, vectorul - **v** şi numele fişierului din care se doreşte citirea vectorului - **numeFisier**. Returnează valoarea **1 (unu)** dacă nu au apărut erori la deschiderea fişierului şi **0 (zero)** în caz contrar.

```
int citireVectorFisier(int* pn, int v[NMAX], char* numeFisier){
      FILE *pf;
      int i;
      //se deschide fisierul in mod citire (read)
      pf = fopen (numeFisier, "rt");
      if (pf == NULL)
            return 0;
      fscanf (pf, "%d",pn); //se citeste numarul de elemente
      for (i=0;i<*pn;i++)   //se citesc succesiv elementele
            fscanf (pf, "%d",&v[i]);
      fclose (pf); //inchidere fisier
      return 1;    //citirea datelor din fisier s-a incheiat cu succes
}
```

Exemplul 2 – copierea conţinutului unui fişier text în altul

În cadrul acestui exemplu se va prezenta o funcţie care copie conţinutul unui fişier în altul. Funcţia primeşte ca argument numele fişierelor sursă şi destinaţie (în această ordine). Returnează valoarea **0 (zero)** dacă a apărut eroare la deschiderea fişierelor şi **1 (unu)** în caz contrar.

```
int copiereFisier1 (char* numeSursa, char* numeDest){
      FILE *pfSursa, *pfDest; // pointeri la cele doua fisiere
      char c;                 // variabila de "manevra"
      if ((pfSursa=fopen(numeSursa, "rt"))==NULL) //deschidere sursa
            return 0;
      if ((pfDest=fopen(numeDest, "wt"))==NULL)   //deschidere destinatie
            return 0;
      /*se citesc caractere din fisierul sursa si se scriu in fisierul
      destinatie pana cand functia fgetc() returneaza EOF*/
      while ((c=fgetc(pfSursa))!=EOF)
            fputc(c,pfDest);
      fclose (pfSursa);
      fclose (pfDest);
      return 1;
}
```

Varianta prezentată realizează copierea caracter cu caracter. În cazul fişierelor text poate fi mai performantă o copiere linie cu linie. În continuare este prezentată secvenţa de copiere pentru acest mod de implementare. Pentru dimensiunea tabloului în care se memorează o linie s-a utilizat constanta **MAX_LINIE.** Cu alte cuvinte, presupunem că numărul maxim de caractere de pe o linie din fişierul sursă este **MAX_LINIE - 1** (la sfârşitul şirului trebuie adăugat şi caracterul **'\0'**).

```
int copiereFisier2 (char* numeSursa, char* numeDest){
      FILE *pfSursa, *pfDest;
      char buf[MAX_LINIE]; // variabila de manevra
      if ((pfSursa=fopen(numeSursa, "rt"))==NULL)
            return 0;
      if ((pfDest=fopen(numeDest, "wt"))==NULL)
            return 0;
      //cand ajunge la sfarsitul fisierului fgets() returneaza NULL
      while (fgets(buf,MAX_LINIE-1,pfSursa)!=NULL)
            fputs(buf,pfDest);
      fcloseall();  // se inchid toate fisierele deschise
      return 1;
}
```

Exemplul 3 – testarea existenţei unui fişier

Funcţia care testează existenţa unui fişier are ca argument numele acestuia. Se bazează pe faptul că **fopen()** returnează **NULL** dacă se încearcă deschiderea în citire a unui fişier inexistent.

```
int existaFisier (char* numeFisier){
      FILE *pf;
      //daca fisierul poate fi deschis pentru scriere atunci el exista
      return (pf=fopen (numeFisier,"r"))==NULL ? 0 : (fclose(pf),1);
}
```

4. Temă de realizat în timpul laboratorului

Exerciţiul 1 – matrice şi fişiere text

Să se realizeze o aplicaţie care să afişeze un meniu cu următoarele opţiuni:

```
C - citire matrice de la tastatura
A - afisare matrice pe ecran
S - salvare matrice in fisier
R - citire matrice din fisier
```

Referitor la **formatul datelor în fişier**, pe prima linie se vor scrie numărul de linii şi numărul de coloane separate prin spaţiu. Apoi se vor scrie pe linii succesive liniile matricei (elementele separate prin spaţiu). Conţinutul fişierului după salvarea matricei **{{4,9,2},{0,6,1}}** va fi:

```
2 3
4 9 2
0 6 1
```

Exerciţiul 2 – generare fişier backup

Să se adauge o opţiune care să permită copierea unui fişier în altul (**B** - creare fisier backup pentru un fisier creat la optiunea S).

Exerciţiul 3 – generare fişiere HTML

Din ce în ce mai multe aplicaţii îşi prezintă datele în format **html**. Fişierele html sunt în acest caz **generate** pe baza unor date necunoscute iniţial. Ele se mai numesc fişiere generate ”**on the fly**” (din zbor). Aplicaţia anterioară poate fi extinsă în acest sens prin adăugarea unei opţiuni (**H – generare html**). Accesarea ei trebuie să determine generarea unui fişier html care, deschis cu ajutorul unui browser, să afişeze matricea sub forma unui tabel ca în exemplul următor:

Matrice: 2 linii, 3 coloane

4	9	2
0	6	1

```
<html>
<head><title>Matrice</title></head>
<body>Matrice: 2 linii, 3 coloane.<br>
<table border='1'>
      <tr><td>4</td><td>9</td><td>2</td></tr>
      <tr><td>0</td><td>6</td><td>1</td></tr>
</table></body></html>
```