Capitolul 10 – Fişiere

Obiectiv: Familiarizarea cu memorarea pe suport extern.

Activităţi:
- Prezentarea unor noţiuni generale legate de lucru cu fişiere (fişiere text şi fişiere binare);
- Prezentarea principalelor funcţii folosite în lucrul cu fişiere;

10.1 Introducere în lucrul cu fişiere

Un *fişier* reprezintă o secvenţă de date stocată în memoria externă a unui calculator (pe harddisk-uri, dischete, CD-ROM, etc). Memoria externă furnizează o capacitate de stocare mult mai mare decât memoria internă, nu este volatilă (nu-şi pierde conţinutul la întreruperea alimentării cu energie electrică), dar are ca principal dezavantaj viteza de transfer a datelor mult mai redusă decât memoria internă. De asemenea, memoriile externe nu dispun de conexiune directă cu procesorul calculatorului.

Pentru a putea prelucra informaţiile din fişiere, programele trebuie să aducă informaţiile din fişiere în memoria internă (RAM) şi apoi în variabilele programului pentru efectuarea prelucrărilor necesare asupra informaţiilor respective, iar apoi să transfere rezultatele înapoi în fişiere.

În limbajul C, operaţiile cu fişierele se realizează prin intermediul unui set de funcţii din biblioteca standard de intrare/ieşire (I/O), care necesită includerea fişierului header *stdio.h*.

Primul lucru care trebuie făcut atunci când vrem să lucrăm cu fişiere este să ne definim un pointer către o structură de tipul FILE. O structură de tipul FILE este o structură de control a stream-urilor. Ea păstrează informaţii cum ar fi adresa de început şi lungimea zonei tampon a datelor, poziţia indicatorului de fişier, poziţia indicatorului de sfârşit a fişierului, etc. Un stream este un canal logic de comunicaţie între program şi fişierul de pe hard. Toate operţiile de citire, şi de scriere din / în fişier se realizează prin intermediul acestui canal de comunicaţie. Deschiderea fişierului, respectiv crearea stream-ului se realizează cu ajutorul funcţiei fopen. Funcţia deschide fişierul pe care îl specificăm prin nume şi cale în modul pe care îl precizăm şi creează canalul de comunicaţie cu fişierul respectiv. În C putem lucra cu 2 tipuri de fişiere: text sau binare.

Pentru a se putea accesa informaţiile dintr-un fişier, trebuie efectuate următoarele operaţii:
- deschiderea fişierului;
- citirea/scrierea conţinutului fişierului;
- închiderea fisierului;

Deschiderea fişierului este necesară datorită faptului că sistemul de operare trebuie să aloce resurse interne, pentru a se putea accesa conţinutul fisierului. Aceste resurse sunt eliberate în momentul închiderii fişierului respectiv.

În momentul deschiderii unui fişier, programatorul trebuie să indice sistemului de operare trei elemente:
- numele fişierului;
- modul de accesare al fişierului (citire, scriere, adaugare la sfârşit, actualizare - citire şi scriere);
- tipul fişierului: text sau binar;

Referitor la tipul fişierului, trebuie facută distincţia între fişierele binare şi fişierele text, atunci când programatorul doreşte să acceseze (şi să modifice) conţinutul unui fişier. Diferenţele apar la nivelul funcţiilor de bibliotecă pentru citirea şi scrierea conţinutului fişierelor.

Un *fişier binar* se accesează ca o succesiune de octeţi, cărora funcţiile de citire şi scriere din fişier nu le dau nici o interpretare.

Un *fişier text* se accesează ca o succesiune de linii de text de lungime variabilă, utilizând un set dedicat de funcţii din biblioteca standard. Aceste funcţii inspectează toate caracterele citite din fişier şi încheie o operaţie de citire în momentul în care a fost preluat un caracter '\n' - new line (linie nouă).

10.2 Funcţii folosite în lucrul cu fişiere

10.2.1 Prezentare generală

În limbajul C există o serie de funcţii specializate în lucrul cu fişiere. Pentru a se putea folosi aceste funcţii, este necesară includerea fişierului header *stdio.h*

Principalele funcţii folosite în lucrul cu fişiere sunt următoarele:

Nume funcţie	Descriere funcţie
fopen ()	Deschide un fişier
fclose ()	Închide un fişier
fputch ()	Scrie un caracter într-un fişier
fgetch ()	Citeşte un caracter dintr-un fişier
fseek ()	Caută un anumit octet într-un fişier
fprintf ()	Echivalentul lui printf (), folosit pentru un fişier
fscanf ()	Echivalentul lui scanf (), folosit pentru un fişier
feof ()	Returnează o valoare dif. de 0 dacă se ajunge la sfârşitul fişierului
ferror ()	Returnează o valoare dif. de 0 dacă apare o eroare
ftell ()	Returnează poziţia cursorului într-un fişier

Tabel 10.1 – Principalele funcţii folosite în lucrul cu fişiere

Pentru a se putea utiliza un fişier, este necesară folosirea unui pointer, care să facă legătura între fişierul respectiv şi sistemul I/O din C. Pointer-ul respectiv este un pointer către anumite informaţii despre fişierul respectiv – nume, starea şi poziţia in care se află poinetrul de fişier, dimensiune etc.

```
FILE * fisier;
```

10.2.2 Deschiderea şi închiderea fişierelor

Deschiderea unui fişier se face cu funcţia *fopen ()*, care are următorul prototip:

```
FILE *fopen (char *nume_fişier, char *mod)
```

Primul parametru *(filename)* reprezintă numele fişierului care urmează să se dechidă.

Al doilea parametru *(mod)* este un şir de caractere care indică modul de accesare şi tipul fişierului. Valorile posibile pentru acest parametru sunt următoarele:

Valoarea lui *mod*	Tipul fişierului	Modul de accesare
"rt"	Text	doar citire; dacă fişierul nu există se semnalează eroare
"rb"	Binar	
"wt"	Text	doar scriere; dacă fişierul nu există, va fi creat, iar dacă există, va fi suprascris
"wb"	Binar	
"at"	Text	doar scriere; dacă fişierul nu există va fi creat, iar dacă există se va adauga la sfârşitul lui
"ab"	Binar	
"r+t"	Text	citire şi scriere; dacă fişierul nu există se semnalează eroare
"r+b"	Binar	
"w+t"	Text	citire şi scriere; dacă fişierul nu există va fi creat, iar dacă fişierul există va fi suprascris
"w+b"	Binar	
"a+t"	Text	citire şi scriere; dacă fişierul nu există va fi creat, iar dacă fişierul există se va adăuga la sfârşitul său
"a+b"	Binar	

Tabel 10.2 –Moduri de accesare a fişierelor

Dacă operaţia de deschidere eşuează, funcţia returnează valoarea NULL. Dacă deschiderea reuşeşte, funcţia returnează o valoare de tip *FILE* *, care se foloseşte în toate operaţiile ulterioare asupra fişierului respectiv.

Exemplu: crearea (deschiderea) unui fişier şi setarea atributului "w" (pentru a se putea scrie în fişierul respectiv):

```
FILE *fisier;
fisier = fopen ("exemplu.txt", "wt");
```

O versiune mai corectă a exemplului anterior este:

```
FILE *fisier;
fisier = fopen ("exemplu.txt", "wt");

if (fisier == NULL)
{
     printf ("Fisierul nu poate fi creat.");
}
else
{
   printf ("Fisierul a fost creat cu succes");
}
```

Această variantă este recomandată, deoarece utilizatorului i se confirmă crearea fişierului dorit, iar în caz contrar (protecţie la scriere, disc plin etc.) se va afişa un mesaj.

După ce sunt folosite, fişierele trebuie închise. Atunci când execuţia programului se încheie cu succes (fie prin încheierea execuţiei funcţiei main (), fie prin apelarea funcţiei exit()), toate fişierele deschise în momenul respectiv sunt automat închise. Dacă rularea unui program este întreruptă forţat (prin blocarea execuţiei sau prin rularea funcţiei abort()), fişierele deschise în momentul respectiv nu sunt închise.

Închiderea unui fişier se face cu funcţia *fclose ()*, care are următorul prototip:

```
int fclose (FILE *fisier)
```

Parametrul *fişier* trebuie să fie valoarea returnată de funcţia *fopen ()*. Apelarea funcţiei va avea ca rezultat şi scrierea oricărei date rămasă în buffer-ul discului pe care se află fişierul, executând de asemenea închiderea fişierului la nivel de sistem de operare.

Funcţia *fclose ()* returnează 0 dacă operaţia de închidere a fişierului s-a realizat cu succes şi EOF dacă a intervenit o eroare.

Dacă un fişier nu este închis corespunzător pot apare diverse situaţii nedorite – pierderi de date, fişiere corupte sau/şi distruse, posibilitatea apariţiei de erori în programul în care se foloseşte fişierul (sau fişierele) respectiv(e). De asemenea, funcţia *fclose ()* distruge asocierea dintre fişierul respectiv şi variabila FILE * corespunzătoare, astfel încât variabila respectiva poate fi reutilizata.

10.2.3 Determinarea poziţiei / setarea cursorului de fişier

Informaţiile dintr-un fişier sunt accesate secvenţial, iterativ. Orice fişier, odată deschis, are asociat un cursor care indică poziţia curentă în fişierul respectiv (iniţial - începutul fişierului respectiv). Funcţiile de scriere şi citire din fişiere operează începand de la poziţia curentă a cursorului respectivului fişier. Totodată, orice operaţie de scriere sau citire avansează automat acest cursor cu numărul de octeţi citiţi/scrişi. Spre exemplu, daca poziţia curentă este pe octetul 7, scrierea sau citirea a 5 octeţi avansează cursorul pe octetul 12.

Poziţia curentă a cursorului dintr-un fişier se poate afla folosind funcţia *ftell ()*:

```
long ftell (FILE *fişier)
```

Funcția *ftell ()* returnează poziția (în octeți) față de începutul fişierului în care se află cursorul de parcurgere a fişierului.

Poziționarea cursorului pe o anumita poziție din fişier se poate face, de asemenea, folosind funcția *fseek ()*:

```
int fseek (FILE *fişier, long pozitie, int referinta)
```

Parametrul *fişier* trebuie să fie o valoare returnată de funcția *fopen ()*. Parametrul *pozitie* indică deplasarea în octeți față de *referința* unde se va poziționa cursorul. Ca referință (al treilea parametru), se pot folosi următoarele valori:

SEEK_SET – față de începutul fişierului;
SEEK_CUR – față de poziția curentă a cursorului respectivului fişier
SEEK_END – față de sfârşitul fişierului

Exemplu: program care determină şi afişează dimensiunea în octeți a unui fişier:

```
#include <stdio.h>

long get_file_size (FILE *fisier)
{
    long pos;
    long size;

    /* salvam pozitia curenta a cursorului */
    pos = ftell (fisier);

    /* pozitionare la sfarsitul fisierului */
    fseek (fisier, 0, SEEK_END);

    /* aflam dimensiunea in octeti */
    size = ftell (fisier);

    /* refacem pozitia initiala a cursorului */
    fseek (fisier, pos, SEEK_SET);

    return size;
}

int main (void)
{
    FILE *fisier;
    long size;
    char nume [80];

    printf ("Introduceti numele fisierului: ");
    gets (nume);

    /* deschidem fisierul pentru citire, binar */
    fisier = fopen (nume, "rb");

    if (fisier == NULL)
    {
        printf ("Nu se poate deschide fisierul %s! \n",
                 nume);
```

```
    }
    else
    {
        size = get_file_size (fisier);
        printf ("Fisierul %s are %d octeti. \n", nume,
                 size);

        fclose (fisier);
    }
    return 0;
}
```

10.2.4 Citirea şi scrierea din/în fişier

În cazul în care se lucrează cu **fişiere text** se folosesc în mod uzual următoarele funcţii**:**

- pentru scriere: sunt folosite funcţiile *fputs ()* şi *fprintf ()*;
- pentru citire: sunt folosite funcţiile *fgets ()* şi *fscanf ()*;

Funcţiile recomandate pentru lucrul cu fişiere binare sunt urmatoarele:

- pentru scriere: se foloseşte funcţia *fwrite ()*;
- pentru citire: se foloseşte funcţia *fread ()*;

Observaţie: atunci când cursorul ajunge la sfârşitul fişierului, scrierea în fişierul respectiv are ca efect creşterea dimensiunii fişierului cu numărul de octeţi scrişi.

Funcţiile fputs () şi fgets ()

Funcţiile *fputs ()* şi *fgets ()* sunt folosite pentru citirea şi scrierea de şiruri de caractere în sau dintr-un fişier.

Prototipurile acestor funcţii sunt următoarele:

```
int fputs (char *sir, FILE *fisier);
char *fgets (char *sir, int lungime, FILE *fişier);
```

Funcţia *fputs ()* scrie în fişierul specificat şirul spre care indică pointer-ul *sir*. Dacă apare vreo eroare, funcţia *fputs ()* returnează EOF.

Funcţia *fgets ()* citeşte un şir de caractere din fişierul specificat, până când este întâlnit un caracter de linie nouă sau atunci când au fost citite *lungime-1* caractere. Dacă este citit un caracter de linie nouă, acesta va fi inclus în şir. Funcţia va returna şirul de caractere *şir* dacă citirea s-a făcut corect şi returnează valoarea NULL dacă apare o eroare în timpul citirii. Şirul rezultat (în urma citirii) se termină cu '\0'.

Funcţiile fscanf () şi fprintf ()

Funcţiile *fscanf ()* şi *fprintf ()* funcţionează identic cu funcţiile *scanf ()* şi *printf ()*, cu deosebirea că citirea respectiv scrierea se face operează asupra unui fişier specificat ca prim parametru, permiţând folosirea şirurilor şi a caracterelor de formatare.

```
int fscanf(FILE *f,char *format,..adresele variabilelelor...);
int fprintf (FILE *f,char *format, ...valorile variabilelor...);
```

Funcţiile fread () şi fwrite ()

Funcţiile *fread ()* şi *fwrite ()* sunt folosite pentru scrierea datelor în fişierele binare, fără a da nici o interpretare conţinutului. Astfel, funcţiile *fread ()* şi *fwrite ()* sunt folosite pentru citirea şi scrierea de blocuri de date structurate, fişierul rezultat fiind o copie a conţinutului blocului de memorie din care se face scrierea în fişier (în cazul *fwrite ()*) sau o imagine a conţinutului fişierului (în cazul *fread ()*).

Prototipurile funcţiilor *fread ()* şi *fwrite ()* sunt următoarele:

```
size_t fread (void *ptr,size_t noct,size_t nr,FILE *f);
size_t fwrite (void *ptr,size_t noct,size_t nr,FILE *f);
```

În cazul funcţiei *fread ()*, pointer-ul *ptr* este un pointer către o regiune de memorie care va primi datele din fişierul respectiv, iar în cazul funcţiei *fwrite ()* pointer-ul *ptr* este un pointer către informaţiile care vor fi scrise în fişierul respectiv.

Valoarea *nr* specifică numărul de elemente citite sau scrise, fiecare dintre elementele respectiver având dimensiunea in octeţi specificata in parametrul *noct*.

Funcţia *fwrite ()* returnează numărul de elemente scrise în fişier, dacă nu apare nici o eroare în timpul scrierii (valoarea returnată va fi egală cu *nr* dacă nu intervine nici o eroare).

Funcţia *fread ()* returnează numărul de elemente citite, putând fi mai mică decât valoarea lui *nr*, dacă se ajunge la sfârşitul fişierului sau dacă apare vreo eroare.

Exemplu de folosire a funcţiei *fwrite ()*:

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

typedef struct
{
    int varsta;
    char nume[20];
}student;

int main(void)
{
    FILE *f;
    student st;

    f = fopen ("fis_binr.bin", "wb");

    if (f == NULL)
       printf ("Fisierul nu se poate crea!");

    else
    {
       st.varsta = 19;
       strcpy (st.nume, "Andrei");

       /* scrierea structurii in fisier */
       fwrite (&st, sizeof(st), 1, f);
       fclose (f);
    }
    return 0;
}
```

Exemplu de folosire a funcţiei *fread ()* (pentru fişierul creat în exemplul anterior):

```
#include <stdio.h>

typedef struct
{
    int varsta;
    char nume[20];
}student;

int main(void)
{
    FILE *f;
    student st;
    int n;

    f = fopen ("fis_binr.bin", "rb");
    if (f == NULL)
       printf ("Fisierul nu se poate deschide!");
    else
    {
       n = fread (&st, sizeof (st), 1, f);

       if (n == 1)
          printf ("%s are %d ani \n", st.nume, st.varsta);
       else
          printf ("Nu s-a putut citi inregistrarea din
                   fis.");
       fclose (f);
    }
    return 0;
}
```

10.3 Exemple rezolvate

Exemplul 1: program care citeşte un fişier text şi îl afişează linie cu linie:

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

int main (void)
{
    FILE *fis;
    char *sir, *rez;

    sir = malloc (50 * sizeof (char));
    /*iniţializarea pointerului */

    fis = fopen ("exemplu.txt", "rt");

    if (fis == NULL)
       printf ("Fisierul nu se poate deschide!");
    else
    {
       while (!feof (fis))
       {
          rez = fgets (sir, 50, fis);
```

```
            if (rez == NULL)
               printf ("Nu s-a putut efectua citirea!");
            else
               printf ("%s", sir);
         }
         fclose (fis);
      }
      free (sir); /*eliberarea memoriei alocate*/
      return 0;
}
```

Exemplul 2: program care citeşte de la tastatură un text de mai multe linii şi îl scrie într-un fişier text:

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

void cit_sir (char *sir)
{
     do
     {
            gets (sir);
     }
     while (strlen (sir) == 0);
}

int main (void)
{
     FILE *fis;
     char sir[50];
     int n, i, b;

     fis = fopen ("2.txt", "wt");

     if (fis == NULL)
     {
        printf ("Fisierul nu se poate crea ! \n");
     }
     else
     {
        printf ("Cate linii doriti sa introduceti? ");
        scanf ("%d", &n);

        for (i = 0; i < n; i++)
        {
           printf ("Linia %d: ", i + 1);
           cit_sir (sir);
           strcat (sir, "\n");
           b = fputs (sir, fis);
           if (b == EOF)
              printf ("Nu se poate face scrierea ! \n");
        }

        fclose (fis);
     }
     return 0;
}
```

Exemplul 3: program care citeşte de la tastatură câteva înregistrări de tip structură (ex. informații despre studenți - numele şi vârsta) şi le scrie într-un fişier:

```
#include <stdio.h>
#include <string.h>
#define DIM 100

typedef struct
{
    int varsta;
    char nume[20];
}student;

student tab[DIM];
int n;

void cit_sir (char *sir)
{
    do
    {
       gets (sir);
    }while (strlen (sir) == 0);
}
int main (void)
{
    FILE *f;
    int i;

    printf (“Cate inregistrari doriti? “);
    scanf (“%d”, &n);
    for (i = 0; i < n; i++)
    {
       printf (“Numele persoanei %d: “, i + 1);
       cit_sir (tab[i].nume);

       printf (“Varsta persoanei %d: “, i + 1);
       scanf (“%d”, &tab[i].varsta);
    }
    /*se creează un fişier binar*/
    f = fopen ("ex_binar.bin", "wb");

    if (f == NULL)
       printf ("Fisierul nu se poate crea!");
    else
    {
    /*TOATE inregistrarile sunt scrise printr-o apelare*/
        fwrite (tab, sizeof (student), n, f);
        fclose (f);
    }
    return 0;
}
```

Exemplul 4: program care citeşte fişierul generat de exemplul anterior si afişează înregistrările citite:

```
#include <stdio.h>
#define DIM 100

typedef struct
{
    int varsta;
    char nume[20];
}student;

student tab[DIM];
int n;

int main (void)
{
    FILE *f;
    int i;

    f = fopen ("ex_binar.bin", "rb");
    if (f == NULL)
       printf ("Fisierul nu se poate deschide!");
    else
    {
       n = fread (tab, sizeof (student), DIM, f);
       for (i = 0; i < n; i++)
       {
          printf ("Numele %d: %s\n", i+1, tab[i].nume);
          printf ("Varsta %d: %d\n", i+1, tab[i].varsta);
       }
       fclose (f);
    }
    return 0;
}
```

10.4 Probleme propuse

1. Să se realizeze un program care copiază dintr-un fişier text într-un alt fişier text doar liniile care conţin un anumit cuvânt, introdus de la tastatură.
2. Se dă un fişier text. Să se determine numărul de linii din fişier. Să se creeze un alt fişier, cu liniile din primul aparând în ordine inversă.
3. Să se realizeze un program care şterge un anumit student (al cărui nume este introdus de la tastatură) din baza de date creată cu penultimul exemplu rezolvat (Exemplul 3).

Indicaţie: Intrucat nu există nici o modalitate de a micşora dimensiunea unui fişier, pentru a şterge o înregistrare din fişier se poate proceda în două moduri:

a) se încarcă informaţiile din fişier în tabloul de structuri, se inchide fişierul, se elimină din tablou elementul şters, se suprascrie fişierul prin deschiderea lui cu atributul "w", operaţie urmată de scrierea tabloului de structuri în fişier şi închiderea fişierului;
b) se genereaza un alt fişier care să conţină toate înregistrarile din cel iniţial, mai putin cea care trebuie ştearsă. Apoi se poate utiliza funcţia *remove* pentru a sterge fisierul initial si *rename* pentru a redenumi fisierul generat la numele iniţial:

```
    int remove (char *filename);
    int rename (char *old_name, char *new_name)
```

La funcţia *remove*, parametrul *filename* reprezintă numele fişierului care se va şterge.

La funcţia *rename*, parametrul *old_name* şi *new_name* reprezintă numele iniţial respectiv cel final al fişierului).

În momentul aplicării funcţiilor, fişierele respective trebuie să fie închise, alfel funcţiile eşuează! Funcţiile returnează valoarea 0 dacă operaţia a reuşit sau o valoare diferită de 0 în caz de eşec.

4. Un client lansează o precomandă la o firmă speificând:
   - nume client (30 de caractere)
   - produs comandat (30 de caractere)
   - cantitatea comandată (întreg)

   Pentru a onora precomanda firma caută produsul comandat în depozit, şi trimite clientului care a lansat comanda un raspuns de forma:
   - adresa client
   - mesaj de forma: *Produsul … există şi costă* ….sau *Produsul … nu există* pe stoc.
   - Pentru a trimite aceste raspunsuri firma foloseşte o agendă conţinand perechi:
     - nume client
     - adresă client

   Scrieţi un program care foloseşte fisierele binare comenzi, depozit şi agenda şi creează fişierul text răspunsuri.

10.5 Intrebari recapitulative

1. Care sunt tipurile de fisiere cu care se poate lucra in C? Care este diferenta dintre ele?
2. Care sunt functiile care permit deschiderea respectiv inchiderea unui fisier? Cum se utilizeaza?
3. Care sunt modurile in care poate fi deschis un fisier?
4. Precizati modul de utilizare al functiilor fwrite / fread.
5. Precizati modul de utilizare al functiilor fputs, fgets.