FIŞIERE

8

10.6. POZIŢIONAREA ÎNTR-UN FIŞIER

Pe lângă mecanismul de poziţionare implicit (asigurat prin operaţiile de citire şi scriere) se pot folosi şi operaţiile de poziţionare explicită.

- **Funcţia `fseek`**

```
int fseek(FILE *pf, long deplasament, int referinţa);
```

Funcţia deplasează capul de citire/scriere al discului, în vederea prelucrării înregistrărilor fişierului într-o ordine oarecare. Funcţia setează poziţia curentă în fluxul de date la n octeţi faţă de `referinţă`):

`deplasament` – defineşte numărul de octeţi peste care se va deplasa capul discului;

`referinţa` – poate avea una din valorile:

0 - începutul fişierului (`SEEK_SET`);

1 - poziţia curentă a capului (`SEEK_CUR`);

2 - sfârşitul fişierului (`SEEK_END`).

Funcţia returnează valoarea zero la poziţionarea corectă şi o valoare diferită de zero în caz de eroare (prototip în **stdio.h**).

- **Funcţia `lseek`**

```
int lseek(int indicator, long n, int referinta);
```

Seteaza poziţia curentă de citire/scriere în fişier la n octeţi faţa de `referinţă`. Returnează valoarea 0 în caz de succes şi diferită de zero în caz de eroare (prototip în **io.h**).

- **Funcţia `fgetpos`**

```
int fgetpos(FILE *flux_date, fpos_t *poziţie);
```

Determină poziţia curentă (pointer către o structură, `fpos_t`, care descrie această poziţie în fluxul de date). Înscrie valoarea indicatorului în variabila indicată de `poziţie`. Returnează 0 la determinarea cu succes a acestei poziţii sau valoare diferită de zero în caz de eşec. Structura care descrie poziţia poate fi transmisă ca argument funcţiei `fsetpos` (prototip în **stdio.h**).

- **Funcţia `fsetpos`**

```
int fsetpos(FILE *flux_date, const fpos_t *poziţie);
```

Setează poziţia curentă în fluxul de date (atribuie indicatorului valoarea variabilei indicate `poziţie`), la o valoare obţinută printr apelul funcţiei `fgetpos`. Returnează valoarea 0 în caz de succes, sau diferită de 0 în caz de eşec (prototip în **stdio.h**).

Există funcţii pentru modificarea valorii indicatorului de poziţie şi de determinare a poziţiei curente a acestuia.

- **Funcţia `ftell`**

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

Indică poziţia curentă a capului de citire în fişier. Funcţia returnează o valoare de tip long int care reprezintă poziţia curentă în fluxul de date (deplasamentul în octeţi a poziţiei capului faţă de începutul fişierului) sau -1L în caz de eroare (prototip în **stdio.h**).

- **Funcţia `tell`**

```
long tell(int indicator);
```

Returnează poziţia curentă a capului de citire/scriere în fişier (exprimată în număr de octeţi faţă de începutul fişierului), sau -1L în caz de eroare (prototip în **io.h**).

- **Funcţia `rewind`**

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

Poziţionează indicatorul la începutul fluxului de date specificat ca argument (prototip în **stdio.h**).

10.7. FUNCŢII UTILITARE PENTRU LUCRUL CU FIŞIERE

Funcţii ***de testare a sfârşitului de fişier***

- **Funcţia `feof`**

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

Returnează o valoare diferită de zero în cazul întâlnirii sfârşitului de fişier sau 0 în celelalte cazuri (prototip în **stdio.h**).

- **Funcţia `eof`**

```
int eof(int indicator);
```

Returnează valoarea 1 dacă poziţia curentă este sfârşitul de fişier, 0 dacă indicatorul este poziţionat în altă parte, sau -1 în caz de eroare (prototip în **io.h**).

Funcţii ***de golire a fluxurilor de date***

- **Funcţia `fflush`**

```
int fflush(FILE *flux_date);
```

Goleşte un fluxul de date specificat ca argument. Returnează 0 în caz de succes şi -1 (`EOF`) în caz de eroare (prototip în **stdio.h**).

- **Funcţia `flushall`**

```
int flushall(void);
```

Goleşte toate fluxurile de date existente, pentru cele de scriere efectuând şi scrierea în fişiere. Returnează numărul de fluxuri asupra cărora s-a efectuat operaţia (prototip în **stdio.h**).

10.8. ALTE OPERAŢII CU FIŞIERE

Funcţii care ***permit operaţii ale sistemului de operare asupra fişierelor***

- **Funcţia `remove`**

```
int remove(const char *nume_fişier);
```

Şterge un fişier. Returnează valoarea 0 pentru operaţie reuşită şi -1 pentru operaţie eşuată (prototip în **stdio.h**).

- **Funcţia `rename`**

```
int rename(const char *nume_vechi, const char *nume_nou);
```

Redenumeşte un fişier. Returnează 0 pentru operaţie reuşita şi -1 în cazul eşecului (prototip în **stdio.h**).

- **Funcţia `unlink`**

```
int unlink(const char *nume_fişier);
```

Şterge un fişier. Returnează 0 la operaţie reuşită şi -1 la eşec; dacă fişierul are permisiune read-only, funcţia nu va reuşi operaţia (prototip în **io.h**, **stdio.h**).

Funcţii care ***permit manipularea aceluiaşi fişier prin două indicatoare de fişier independente***

- **Funcţia `dup`**

```
int dup(int indicator);
```

Duplică un indicator de fişier. Returnează noul indicator de fişier pentru operaţie reuşită sau -1 în cazul eşecului (prototip în **io.h**).

- **Funcţia `dup2`**

```
int dup2(int indicator_vechi, int indicator_nou);
```

Duplică un indicator de fişier la valoarea unui indicator de fişier deja existent. Returnează 0 în caz de succes şi -1 în caz de eşec (prototip în **io.h**).

Funcţii pentru ***aflarea sau modificarea dimensiunii în octeţi a fişierelor***

- **Funcţia `chsize`**

```
int chsize(int indicator, long lungime);
```

Modifică dimensiunea unui fişier, conform argumentului `lungime`. Returnează 0 pentru operaţie reuşită sau -1 în caz de eşec (prototip în **stdio.h**).

- **Funcţia `filelength`**

```
long filelength(int indicator);
```

Returnează lungimea unui fişier (în octeţi) sau -1 în caz de eroare (prototip în **io.h**).

Funcţii de ***lucru cu fişiere temporare*** care oferă facilităţi de lucru cu fişiere temporare prin generarea de nume unice de fişier în zona de lucru.

- **Funcţia `tmpfile`**

```
FILE *tmpfile(void);
```

Deschide un fişier temporar, ca flux de date, în mod binar (`w+b`). Returnează pointerul către fişierul deschis în cazul operaţiei reuşite, sau `NULL` în caz de eşec (prototip în **stdio.h**).

- **Funcţia `tmpnam`**

```
char *tmpnam(char *sptr);
```

Crează un nume unic pentru fişierul temporar (prototip în **stdio.h**).

- **Funcţia `creattemp`**

```
int creattemp(char *cale, int attrib);
```

Crează un fişier unic ca nume, cu atributele specificate în argumentul `attrib` (prin `_fmode`,`O_TEXT` sau `O_BINARY`), în directorul dat în argumentul `cale`. Returnează indicatorul (handler-ul) către fişierul creat sau -1 (şi setarea `errno`) în cazul eşecului (prototip în **io.h**).

Exemplu: Să se creeze un fişier binar, care va conţine informaţiile despre angajaţii unei întreprinderi: nume, marca, salariu. Să se afişeze apoi conţinutul fişierului.

```
#include<iostream.h>
#include <stdio.h>
#include <ctype.h>
typedef struct
   { char nume[20];int marca;double salariu;
    }angajat;
union
{angajat a;char sbinar[sizeof(angajat)];}buffer;

int main()
{angajat a; FILE *pf; char cont;char *nume_fis;
cout<<"Nume fisier care va fi creat:"; cin>>nume_fis;
      if ((pf= fopen(nume_fis, "wb")) == NULL)
         {  cout<<"\nEroare creare fişier "<<nume_fis<<"!\n";
            return 1;   }
do
  {cout<<"Marca : ";cin>>a.marca;
   cout<<"Nume : ";cin>>a.nume;
   cout<<"Salariu :";cin>>a.salariu;
```

```
   buffer.a=a;
   fwrite(buffer.sbinar,1,sizeof(angajat),pf);
   cout<<"Continuati introducerea de date (d/n) ?";
   cin>>cont;
  } while(toupper(cont)!='N');
 fclose(pf);
```

//citirea informatiilor

```
      if ((pf= fopen(nume_fis, "rb")) == NULL)
         {  cout<<"\nEroare citire fişier "<<nume_fis<<"!\n";
            return 1;   }
 for(;;)
  {
   fread(buffer.sbinar,1,sizeof(a),pf);
   a=buffer.a1;
   if(feof(pf))  exit(1);
   cout<<" Marca : "<<a.marca;
   cout<<" Numele : "<<a.nume<<'\n';
   cout<<" Salariul : "<<a.salariu<<'\n';
  }
fclose(pf);
}
```

Exemplu: Aplicaţie pentru gestiunea materialelor dintr-un depozit. Aplicaţia va avea un meniu principal şi va permite gestiunea următoarelor informaţii: codul materialului (va fi chiar "numărul de ordine"), denumirea acestuia, unitatea de măsură, preţul unitar, cantitatea contractată şi cea recepţionată (vectori cu 4 elemente). Memorarea datelor se va face într-un fişier de date (un fişier binar cu structuri), numit "material.dat". Aplicaţia conţine următoarele funcţii:

1. `help()` - informare privind opţiunile programului
2. Funcţii pentru fişierele binare, care să suplinească lipsa funcţiilor standard pentru organizarea directă a fişierelor binare:
   - `citireb()` - citire în acces direct din fişier;
   - `scrieb()` - scriere în acces direct în fişier;
   - `citmat()` - citirea de la terminal a informaţiilor despre un material;
   - `afismat()` - afişarea informaţiilor despre un material (apelată de `list`);
   - `lungfisis()` - determinarea lungimii fişierului existent;
   - `crefis()` - creare fişier.
3. Funcţii pentru adaugarea, modificarea, ştergerea şi listarea de materiale.

```
#include <process.h>
#include <iostream.h>
#include <stdio.h>
#include <ctype.h>
typedef struct material
       { int codm,stoc,cant_c[4],cant_r[4];
         char den_mat[20],unit_mas[4];
         float preţ;
};
material mat;
FILE *pf;
void crefis(),adaug(),modif(),sterg(),list(),help();
void main()
 {
  char opţiune;
  do         //afişarea unui meniu de opţiuni şi selecţia opţiunii
   {
    cout<<'\n'<<"Opţiunea Dvs. de lucru este"<<'\n'
           <<"(c|a|m|s|l|e|h pentru help) : ";
    cin>>opţiune;
    switch(opţiune)
```

```
        {
         case 'c':case 'C':crefis();break;
         case 'a':case 'A':adaug();break;
         case 'm':case 'M':modif();break;
         case 's':case 'S':şterg();break;
         case 'l':case 'L':list();break;
         case 'h':case 'H':help();break;
         case 'e':case 'E':       break;
         default:help();          break;
        }
   }while(toupper(opţiune)!='E');
 }
void help()          // afişare informaţii despre utilizarea meniului şi opţiunile acestuia
 {cout<<"Opţiunile de lucru sunt :"<<'\n';
  cout<<"    C,c-creare fisier"<<'\n';
  cout<<"    A,a-adaugare"<<'\n';
  cout<<"    M,m-modificare"<<'\n';
  cout<<"    L,l-listare"<<'\n';
  cout<<"    S,s-ştergere"<<'\n';
  cout<<"    H,h-help"<<'\n';
  cout<<"    E,e-exit"<<'\n';
 }
long int lungfis(FILE *f)       // returnează lungimea fişierului
 {long int posi,posf;
  posi=ftell(f); fseek(f,0,SEEK_END);
  posf=ftell(f); fseek(f,posi,SEEK_SET);
  return posf;
 }
void scrieb(int nr,void *a,FILE *f) //scriere în fişierul binar
 {long depl=(nr-1)*sizeof(material);
  fseek(f,depl,SEEK_SET);
  if(fwrite(a,sizeof(material),1,f)!=1)
      {cout<<"Eroare de scriere in fişier !"<<'\n';
       exit(1); }
 }
void citireb(int nr,void *a,FILE *f)        //citire din fişierul binar
 {long depl=(nr-1)*sizeof(material);
  fseek(f,depl,SEEK_SET);
  if(fread(a,sizeof(material),1,f)!=1)
      {cout<<"Eroare de citire din fişier !"<<'\n';
       exit(2);   }
 }
void afismat(material *a)       //afişarea informaţiilor despre un anumit material
 {
  int i;
  if(a->codm)
      {cout<<"Cod material     : "<<a->codm<<'\n';
       cout<<"Denumire material: "<<a->den_mat<<'\n';
       cout<<"Cantităţi contractate:"<<'\n';
       for(i=0;i<4;i++)
         cout<<"Contractat  "<<i<<"   : "<<a->cant_c[i]<<'\n';
       cout<<"Cantităţi recepţionate:"<<'\n';
       for(i=0;i<4;i++)
         cout<<"Receptionat "<<i<<"  : "<<a->cant_r[i]<<'\n';
       cout<<"Stoc             : "<<a->stoc<<'\n';
       cout<<"Unitate de masura: "<<a->unit_mas<<'\n';
       cout<<"Preţ unitar      : "<<a->preţ<<'\n';
       }
   else     cout<<"Acest articol a fost şters !"<<'\n';
 }
void citmat(material *a)        //citirea informaţiilor despre un anumit material
 {
```

```
  int i;float temp;
  cout<<"Introduceti codul materialului (0=End): ";cin>>a->codm;
  if(a->codm==0) return;
  cout<<"Introduceţi denumirea materialului : ";cin>>a->den_mat;
  cout<<"Introduceţi unitatea de măsură : ";cin>>a->unit_mas;
  cout<<"Introduceţi preţul : ";cin>>temp;a->preţ=temp;
  cout<<"Introduceţi cantitaţile contractate : "<<'\n';
  for(i=0;i<4;i++)
     {cout<<"Contractat "<<i+1<<"  : ";cin>>a->cant_c[i]; }
  cout<<"Introduceţi cantitaţile recepţionate : "<<'\n';
  for(i=0;i<4;i++)
     {cout<<"Receptionat "<<i+1<<"  : ";cin>>a->cant_r[i]; }
 }
void crefis()       //deschidere fisier
 {
  if((pf=fopen("material.dat","r"))!=NULL)
        cout<<"Fişierul exista deja !"<<'\n';
      else
        pf=fopen("material.dat","w");
  fclose(pf);
 }
void adaug()        //adăugare de noi materiale
 {
  int na;  pf=fopen("material.dat","a");//deschidere pentru append
  na=lungfis(pf)/sizeof(material);
  do
   {citmat(&mat);
    if(mat.codm) scrieb(++na,&mat,pf);
   } while(mat.codm);
  fclose(pf);
 }

void modif()        //modificarea informaţiilor despre un material existent
 {
  int na;  char ch;  pf=fopen("material.dat","r+");
  do
   {cout<<"Numarul articolului de modificat este (0=END): ";cin>>na;
    if(na)
      {citireb(na,&mat,pf);
       afismat(&mat);
       cout<<"Modificaţi articol (D/N) ? :";
       do
       {    cin>>ch;
            ch=toupper(ch);
       } while(ch!='D' && ch!='N');
       if(ch=='D')
         {citmat(&mat);
          scrieb(na,&mat,pf);
         }
      }
   }while(na);
  fclose(pf);
 }
void sterg()        //ştergerea din fişier a unui material
 { int n;long int na; pf=fopen("material.dat","r+");
  mat.codm=0; na=lungfis(pf)/sizeof(material);
  do
   {
    do
     {cout<<"Numarul articolului de şters este (0=END): ";cin>>n;
      if(n<0||n>na) cout<<"Articol eronat"<<'\n';
     }while(!(n>=0 && n<=na));
```

```
    if(n) scrieb(n,&mat,pf);
   }while(n);
  fclose(pf);
 }
void list()          //afişare informaţii despre un anumit material
 {
  int na;  pf=fopen("material.dat","r");
  do
    {cout<<"Numarul articolului de listat este (0=END): ";cin>>na;
     if(na)
        {citireb(na,&mat,pf);
         afismat(&mat);
         cout<<'\n';
        }
     }while(na);
  fclose(pf);
 }
```

ÎNTREBĂRI ŞI EXERCIŢII

Chestiuni practice

1. Scrieţi un program de tipărire a conţinuturilor mai multor fişiere, ale căror nume se transmit ca parametri către funcţia main. Tipărirea se face pe ecran (lungimea paginii = 22) sau la imprimantă (lungimea paginii = 61). Conţinutul fiecărui fişier va începe pe o pagină nouă, cu un titlu care indică numele fişierului. Pentru fiecare fişier, paginile vor fi numerotate (cu ajutorul unui contor de pagini).
2. Scrieţi un program care citeşte un fişier text. Pornind de la conţinutul acestuia, se va crea un alt fişier, prin înlocuirea spaţiilor consecutive cu unul singur. Se vor afişa pe ecran conţinutul fişierului de la care s-a pornit şi conţinutul fişierului obţinut.
3. Să se consulte conţinutul unui fişier şi să se afişeze următoarele informaţii statistice: numărul de cuvinte din fişier, numărul de caractere, numărul de linii, numărul de date numerice (nu cifre, numere!).
4. Scrieţi un program care să compare conţinutul a două fişiere, şi afişaţi primele linii care diferă şi poziţia caracterelor diferite în aceste linii.
5. Scrieţi un program care citeşte conţinutul unui fişier sursă scris în limbajul C şi afişează în ordine alfabetică fiecare grup al numelor de variabile care au primele n caractere identice (n este citit de la tastatură).
6. Scrieţi un program care consultă un fişier text şi afişează o listă a tuturor cuvintelor din fişier. Pentru fiecare cuvânt se vor afişa şi numerele liniilor în care apare cuvântul.
7. Scrieţi un program care citeşte un text introdus de la tastatură şi afişează cuvintele distincte, în ordinea crescătoare a frecvenţei lor de apariţie. La afişare, fiecare cuvânt va fi precedat de numărul de apariţii.
8. Scrieţi un program care citeşte un text introdus de la tastatură, ordonează alfabetic liniile acestuia şi le afişează.
9. Scrieţi o aplicaţie pentru gestiunea informatiilor despre cărţile existente într-o bibliotecă. Aplicaţia va avea un meniu principal care va permite:

a) Memorarea datelor într-un fişier (un fişier binar cu structuri), al cărui nume se introduce de la tastatură. Fişierul va contine informaţiile: nume carte, autor, editura, anul apariţiei, preţ. Pentru fiecare carte, se va genera o cotă (un număr unic care să constituie cheia de căutare).
b) Adaugărea de noi cărţi;
c) Afişarea informaţiilor despre o anumită carte;
d) Căutarea titlurilor după un anumit autor;
e) Modificarea informaţiilor existente;
f) Lista alfabetică a tuturor autorilor;
g) Ştergerea unei cărţi din bibliotecă;
h) Ordonarea descrescătoare după anul apariţiei;
i) Numele celei mai vechi cărţi din bibliotecă;
j) Numele celei mai scumpe cărţi din bibliotecă;
k) Numele autorului cu cele mai multe cărţi;
l) Valoarea totală a cărţilor din bibliotecă.