FIŞIERE

Iulia IACOB

Generalităţi

- ***Fişier*** = colecţie ordonată de ***date memorate*** (înregistrări) păstrate pe un *suport extern* de memorie şi identificate printr-un *nume*.

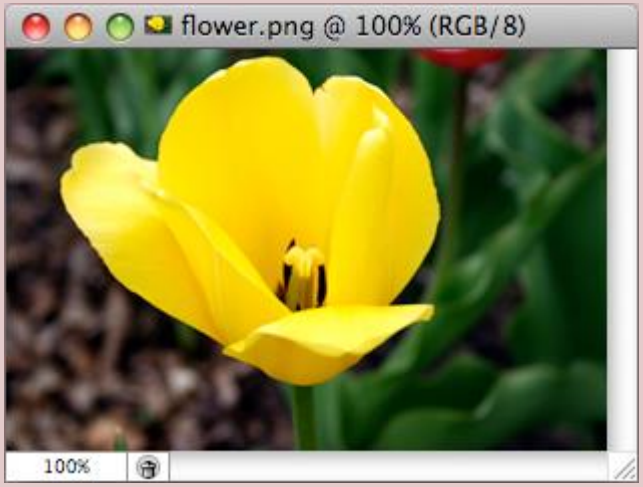
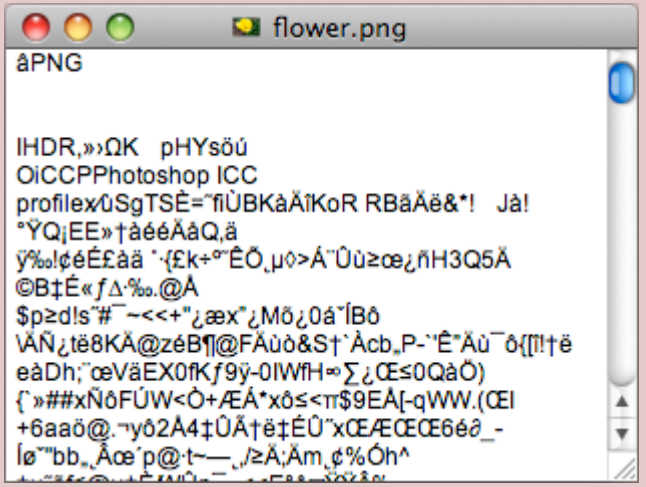
Operaţiile specifice prelucrării fişierelor sunt.

- 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

FIŞIERE TEXT vs FIŞIERE BINARE

Fişiere text	Fişiere binare
➢ datele sunt memorate ca şiruri de caractere ➢ conţin o succesiune de linii, separate prin NewLine (**'\n'** – sau **'\r' şi '\n'**) ➢ fiecare linie are 0 sau mai multe caractere (pot fi citite)	➢ datele sunt păstrate în formatul lor intern ➢ conţin o succesiune de octeţi

FIŞIERE TEXT vs FIŞIERE BINARE

Image Viewer	Text Editor
	

Accesul la fisiere

Fişierele sunt gestionate prin pointeri la structuri **`FILE`**, asociate fişierelor pe durata prelucrării.

`FILE` - structură ce conține informațiile necesare controlării unui fișier.

Declararea unui pointer la fişier: **`FILE *f;`**

Deschiderea unui fisier

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

sau

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

Deschiderea unui fişier

- stabileste o conexiune logică între fişier şi variabila pointer
- alocă o zonă de memorie pentru realizarea mai eficientă a operaţiilor de intrare / ieşire

Exemplu:

```
FILE *f, *g;
f=fopen("student.txt","rt");
```

- se deschide fisierul **`student.txt`**, din directorul curent, in modul citire text.

```
g=fopen("d:\\PC\\test.bin","wt");
```

- se deschide fisierul **`test.bin`** având calea **`d:\\PC\\test.bin`** în modul scriere text.

Moduri de acces la fisier

Mod acces	Descriere
"w"	▪ se creează un fișier pentru scriere ▪ dacă există deja un fișier cu același nume, conținutul lui este șters
"w+"	▪ se creează un fișier atât pentru scriere, cât și pentru citire ▪ dacă există deja un fișier cu același nume, conținutul lui este șters
"r"	▪ fişierul se deschide pentru citire, ▪ fişierul trebuie să existe deja
"r+"	▪ fişierul se deschide pentru citire şi scriere ; ▪ fişierul trebuie să existe deja
"a"	▪ dacă fişierul nu există atunci el va fi creat ▪ dacă fişierul există fişierul se deschide pentru adăugare, ▪ poziţionarea se face la sfârşitul fişierului
"a+"	▪ fişierul se deschide pentru citire şi adăugare la sfârșit; ▪ citirea: se poate face de la orice poziție din fișier, ▪ scrierea: se face doar la sfârșit ▪ dacă fişierul nu există atunci el va fi creat
"t" sau "b"= tip fişier ("text", "binary"), implicit este "t“	

Închiderea unui fişier

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

- returnează **0** la închidere normală
- returnează **EOF** în caz de eroare
- închide fişierul şi se eliberează bufferele alocate
- se întrerupe conexiunea pointer – fişier

Închiderea unui fişier este absolut necesară pentru fişierele în care s-a scris ceva

Citirea şi scrierea fişierelor

Tip fisier	Conversie	Unitate transferata	Funcţii folosite
text	fără	caracter	**fgetc()** **fputc()**
		linie	**fgets()** **fputs()**
	cu	linii	**fscanf()** **fprintf()**
binar	fără	articol	**fread()** **fwrite()**

Operaţii de scriere / citire la nivel de caracter

```
int fputc(char c, FILE *f);
```

- scrie caracterul **c** în fişierul **f**
- ✘ la eroare returnează **EOF**

```
FILE *f;
char c;

f = fopen („litere.txt","wt"); ....
if (f!=NULL)
{
      for (c = 'A' ; c <= 'Z' ; c++)
      fputc ( c , f );

      fclose (f);
}
```

A se vede >> fputc.c

Operaţii de scriere / citire la nivel de caracter

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

- ✓ returnează următorul caracter citit din fişier convertit în întreg fără semn
- ✗ la eroare returnează **EOF**

```
FILE *f;
char c;
int n=0;

....

do {
  c = fgetc (f);
             if (c=='a') n++;
} while (c!= EOF);
....
```

A se vede >> fgetc.c

Operaţii de scriere / citire pentru şiruri de caractere

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

- copiază şirul **s** în fişierul **f**
- nu copiază şi terminatorul de şir '\0'
- ✓ returnează o valoare non-negativa (**≥ 0**)
- ✗ returnează **EOF** la apariţia unei erori

```
...
fputs("Mesaj demonstartiv pentru scriere in fisier!", f);
...
```

A se vede >> fputs.c şi fputs_fseek.c

Operaţii de scriere / citire pentru şiruri de caractere

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

- citeşte caractere din fişierul **f**, şi le memorează în şirul **s** până la întâlnirea primului caracter **'\n'** sau până au fost citite cel mult **n-1** caractere;
- ✓ returnează adresa de început a şirului
- ✗ returnează **NULL** la întâlnire **sfârşit de fişier** sau la **eroare**

```
char *c, buffer[200],numeFisier[25];
FILE *f;

...
while (fgets(buffer,200,f)!=NULL) {
            printf("%s", buffer);
}
...
//sau
...
do {
   c=fgets(buffer,100,f);
   if(c) printf("%s",buffer);
} while(c);
```

A se vede >> fgets.c

Operaţii de scriere / citire cu format

```
int fprintf(FILE *f, char *format, lista_parametri);
```

- transferă în fişierul specificat **f**, valorile **lista_parametri**, convertite potrivit **format**
- ✓ returnează numărul de caractere scrise
- ✘ returnează o valoare negativă dacă s-a produs o eroare

```
FILE *f; char numeFisier[25], expresie[100];
...
f=fopen(numeFisier,"wt");
...
for (i=0;i<n;i++){
            printf ("Introduceti expresia %d: ", i+1); gets (expresie);
            fprintf (f, "Expresie %d: %s\n",i,expresie);
        }
...
fclose(f);
```

A se vede >> fprintf.c

http://www.cplusplus.com/reference/cstdio/fprintf/

Operaţii de scriere / citire cu format

```
int fscanf(FILE *f, char *format, lista_adrese_parametri);
```

- se citesc date din fişierul **f**, sub controlul formatului, iniţializându-se parametrii din listă
- ✓ returnează numărul de câmpuri (componente) citite
- ✘ returnează **EOF** în caz de producere a unui incident la citire sau la întâlnirea marcajului de sfârşit de fişier.

```
FILE *f; char numeFisier[25], s[20]; int d;
...
f=fopen(numeFisier,"wt+");
...
fprintf (f, "%d %s", 2014, "Anul");
fseek(f,0,SEEK_SET);
fscanf (f, "%d %s", &d, s);
...
fclose(f);
printf ("Datele citie din fiser sunt: %s - %d", s, d);
```

A se vede >> fscanf.c

http://www.cplusplus.com/reference/cstdio/fscanf/

Citirea şi scrierea fişierelor

Tip fisier	Conversie	Unitate transferata	Funcţii folosite
text	fără	caracter	**fgetc()** **fputc()**
		linie	**fgets()** **fputs()**
	cu	linii	**fscanf()** **fprintf()**
binar	fără	articol	**fread()** **fwrite()**

Operaţii de scriere / citire în modul de acces binar

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

Operaţii de scriere / citire în modul de acces binar

```
size_t fwrite ( const void * ptr, size_t size, size_t na, FILE * stream );
```

- scrie **na** articole de lungime **size**, din **ptr** în fişierul **stream**
- ✓ returnează numărul de articole scrise
- ✘ returnează **0** dacă s-a produs o eroare

```
struct abonat {
      char nume[20],prenume[20],telefon[20];int varsta;
      }
...
FILE *f; struct abonat a;
f=fopen(numeFisier,"wb");
...
strcpy(a.nume,"Popescu"); strcpy(a.prenume,"Andrei");
      strcpy(a.telefon,"0215111222"); a.varsta=20;
      fwrite(&a,sizeof(a),1,f);
...
fclose(f);
```

A se vede >> fwrite_fread.c

http://www.cplusplus.com/reference/cstdio/fwrite/

Operaţii de scriere / citire în modul de acces binar

`size_t fread ( void * ptr, size_t size, size_t na, FILE * stream );`

- citeşte cel mult **`na`** articole, de lungime **`size`** fiecare, din fişierul **`stream`** în **`ptr`**
- ✓ returnează numărul de înregistrări citite
- ✘ returnează **0** dacă s-a produs o eroare sau s-a ajuns la sfârşit de fişier

```
struct abonat {
  char nume[20],prenume[20],telefon[20];int varsta;
  }
...
FILE *f; struct abonat b;
f=fopen(numeFisier, "rb");
...
fseek(f,0,SEEK_SET);
fread(b.nume,sizeof(b.nume),1,f); printf("NUME: \t\t%s\n", b.nume);
   fread(b.prenume,sizeof(b.prenume),1,f); printf("PRENUME: \t%s\n",
   b.prenume); fread(b.telefon,sizeof(b.telefon),1,f); printf("Telefon:
   \t%s\n", b.telefon); fread(&b.varsta,sizeof(b.varsta),1,f);  printf("Varsta:
   \t%d\n",b.varsta);
fclose(f);
```

A se vede >> fwrite_fread.c

http://www.cplusplus.com/reference/cstdio/fread/

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

- ✓ returnează poziţia curentă în fişier, exprimată prin numărul de octeţi faţă de începutul fişierului
- ✗ returnează -1L dacă s-a produs o eroare

Poziţionarea în fişier

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

- realizează o poziţionare la începutul fişierului, fiind echivalent cu:

```
fseek(f,0,SEEK_SET); sau fseek(f,0,0);
```

Poziţionarea în fişier

```
int fseek(FILE *f, long deplasare, int origine);
```

- modifică poziţia curentă în fişierul cu pointerul **f** cu **deplasare** octeţi relativ la cel de-al treilea parametru **origine**, astfel:
  - dacă **origine = 0 sau origine = SEEK_SET** deplasarea se face faţă de începutul fişierului
  - dacă **origine = 1 sau origine = SEEK_CUR** deplasarea se face faţă de poziţia curentă
  - dacă **origine = 2 sau origine = SEEK_END** deplasarea se face faţă de sfârşitul fişierului

✓ returnează **0** pentru o poziţionare corectă

✘ returnează **≠0** în caz de eroare

Poziţionarea în fişier

Funcţia **fseek()** - folosită în general pentru fişierele binare

- DAR, poate fi folosită şi pentru fişiere text

```
FILE *f;

fputs("Mesaj pentru a testa
pozitionarea in fisiere text", f);
```

În fişier

Mesaj pentru a testa pozitionarea in fisiere text

```
FILE *f;

fseek(f,13,SEEK_SET);
fputs("a vedea", f);
```

Mesaj pentru a vedea pozitionarea in fisiere text

A se vede >> fputs_fseek.c

Poziţionarea în fişier

Funcţia **ftell() -** folosită în general pentru fişierele binare

- DAR, poate fi folosită şi pentru fişiere text

```
FILE *f; numeFisier[25];

long df = ftell(f);
printf("Fisierul %s are %d Bytes\n",
numeFisier, df);
```

Pe ecran:

```
Fisierul demo.txt are 0
Bytes
```

```
FILE *f; numeFisier[25];

fseek(f, 0, SEEK_END);
long df = ftell(f);
printf("Fisierul %s are %d Bytes\n",
numeFisier, df);
```

```
Fisierul demo.txt are
49 Bytes
```

A se vede >> fseek_ftell.c

Tratarea erorilor

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

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

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

- returnează o valoare diferită de **0**, dacă s-a detectat o eroare în cursul operaţiei de intrare / ieşire