Laboratorul 12. OPERAȚII DE INTRARE/IEŞIRE. FIŞIERE.

Introducere

Fişierele sunt structuri de date organizate pe suporturi externe.

Un fişier este alcătuit din înregistrări, care pot avea dimensiune fixă sau variabilă. Numărul înregistrărilor dintr-un fişier este limitat de capacitatea de memorare a suportului fizic folosit.

Pe un suport fizic pot fi stocate mai multe fişiere. Pentru a marca sfârşitul unui fişier se utilizează un marcaj special numit eof (end of file).

Clasificarea fişierelor se poate face după diferite criterii, cum ar fi:

- din punct de vedere al conținutului:
  - fişiere text – fişiere care memorează caractere organizate pe linii. O linie constituie o înregistrare şi poate avea dimensiune variabilă. Sfârşitul unei linii, eol (end of line), este marcată prin caracterele <CR><LF> (0D 0A) (revenire la început de rând şi avans la rândul următor). De exemplu, fişierele sursă sunt fişiere text (exemplu.cpp);
  - fişiere cu tip – fişiere care au înregistrări de un anumit tip (elementar sau structurat);
  - fişiere fără tip – fişiere binare.
- din punct de vedere al accesului:
  - cu acces secvențial – informația poate fi prelucrată doar în ordinea în care a fost înregistrată în fişier;
  - cu acces direct – componentele pot fi prelucrate în orice ordine (chiar şi secvențial). Înainte de orice prelucrare, trebuie specificată poziția ocupată de componentă în fişier.
- din punct de vedere al operațiilor de scriere/citire în/din fişier:
  - de intrare – e permisă doar operația de citire – informația e preluată în memorie – de exemplu, fişierul standard Input – asociat cu tastatura;
  - de ieşire – e permisă doar operația de scriere (stocarea informației suport magnetic) – de exemplu, fişierul standard Output – asociat cu ecranul;
  - de intrare/ieşire – fişierele pe suport magnetic pot fi exploatate atât ca fişiere de intrare, cât şi ca fişiere de ieşire, având astfel posibilitatea de a reactualiza informația.

Aşa cum s-a arătat, în C/C++, operațiile de intrare/ieşire se efectuează printr-un set de funcții din biblioteca standard, ceea ce asigură flexibilitate.

Inițierea unui transfer de date este precedată de operația de "deschidere" care creează un stream (de tip text sau binar) şi îl asociază unui dispozitiv sau fişier disc. Practic se foloseşte o variabilă de tip FILE (tipul FILE este declarat în stdio.h) şi se alocă o zonă de memorie tampon prin intermediul căreia se vor efectua transferurile. Se poate controla alocarea de zone tampon prin funcții C.

Pentru dispozitivele care permit acces aleator la date, se poate utiliza un indicator al poziției

curente (numărul de octeți de la începutul fişierului) care este actualizat în urma fiecărui transfer. Prin modificarea acestui indicator, se pot efectua operații de citire/scriere în diferite zone ale fişierului.

La terminarea operațiilor efectuate cu acel stream, este necesară efectuarea operației de "închidere" a acestuia, ceea ce eliberează memoria alocată la deschiderea sa, după ce se goleşte zona tampon.

12.1. Funcții de intrare/ieşire cu caracter general.

- **Deschiderea unui stream**

Funcția fopen() deschide un fişier, îi asociază un stream şi întoarce un pointer către structura FILE creată.

FILE * fopen (const char * *nume_fisier*, const char * *mod*);

unde:

nume fişier este un şir de caractere care reprezintă un nume de fişier valid pentru sistemul de operare (include şi specificarea căii);

mod este un şir de caractere care precizează scopul deschiderii şi este alcătuit din câte un caracter din următoarele grupe:

a. ' r' deschide fişierul, dacă există, pentru citire
'w' creează un fişier nou pentru scriere şi şterge
'a' deschide un fişier (îl creează dacă nu există) pentru scriere la sfârşit (adăugare).

b. '+' asociat caracterelor din grupul (a), indică intenția de a efectua operații de citire şi scriere asupra fişierului deschis în condițiile r, w, a

c. 't' mod text
'b' mod binar.

În caz de eşec, la deschiderea fişierului, funcția fopen() întoarce valoarea NULL. Prin testarea valorii întoarsă de funcția fopen(), se poate determina dacă operația de deschidere a fişierului a reuşit, de exemplu:

```
FILE * f1;
if (!(f1=fopen("fisier.txt", "w+")))
     printf("\neroare la deschiderea fisierului fisier.txt");
else
    printf("\nsucces la deschiderea fisierului fisier.txt");
...
```

- **Închiderea unui stream**

Funcția fclose() închide un stream deschis cu funcția fopen(), după ce goleşte zona tampon de date.

int fclose (FILE * stream_ptr);

unde **stream_ptr** este pointerul întors de funcția fopen().

Funcția returnează valoarea 0 în caz de succes şi EOF (-1) în caz de eşec.

- **Ştergerea unui fişier**

int remove(const char * nume_fisier);

Funcția remove() şterge fişierul cu numele indicat şi întoarce 0 în caz de succes şi -1 în caz de eşec.

- **Testarea sfârşitului fişierului**

Orice stream conține un indicator care este setat atunci când o operație de citire întâlneşte sfârşitul fişierului. Funcția feof() testează indicatorul de sfârşit al fişierului.

int feof(FILE * stream_ptr);

Funcția returnează o valoare nenulă dacă s-a întâlnit sfârşitul fişierului la operația de citire precedentă şi valoarea 0 în caz contrar.

12.2. Operații de citire/scriere fără formatare

Cu un fişier deschis în prealabil, se pot face transferuri de date, fără formatare, la nivel de octet (caracter), de cuvânt (doi octeți) sau de bloc de octeți.

- **Transfer la nivel de octet (caracter)**

int getc(FILE * stream_ptr);

Funcția getc() întoarce caracterul citit din fişier sau EOF dacă întâlneşte sfârşitul fişierului.

int putc(int ch, FILE * stream_ptr);

Funcția putc() scrie caracterul ch în fişier şi întoarce valoarea scrisă în caz de succes şi EOF în caz de eşec. Deşi tipul datelor de transfer este int, nu se utilizează decât octetul mai puțin semnificativ.

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

void main()
{
 FILE * f1;
 char sir[ ]="exemplu text";
 char ch;
 int i;
 if (!(f1=fopen("fis1.txt", "w+")))              // if (!(f1=fopen("fis1.txt", "a+")))
   {     printf("\neroare la deschiderea fisierului fis1.txt");
         getch();
         return;
   }

// se copiaza sir in fis1.txt
 for(i=0; sir[i]!='\0';i++)
   putc(sir[i],f1);
 fclose(f1);
 if ((f1=fopen("fis1.txt", "r+"))==NULL)
   {     printf("\neroare la deschiderea fisierului fis1.txt");
         getch();
         return;
   }
```

```
  // se afiseaza continutul fisierului fis1.txt
  printf("\nContinutul fisierului este :\n");
  while(!feof(f1))
  {   ch=getc(f1);
      if(!feof(f1))
          putch(ch);
  }
  fclose(f1);
}
```

- **Transfer la nivel de cuvânt (2 octeți)**

Transfer la nivel de cuvânt (2 bytes) se poate face folosind funcțiile getw(), respectiv putw().

int getw(FILE * stream_ptr);

int putw(int ch, FILE * stream_ptr);

Funcțiile sunt similare cu getc() şi putc(), cu deosebirea că lucrează cu perechi de octeți, deci valorile sunt int.

- **Transfer de şiruri de caracter**

char * fgets(char * str, int lng, FILE * stream_ptr);

Funcția fgets() citeşte cel mult lng-1 caractere din fişier şi le înscrie în şirul str. Dacă întâlneşte caracterul linie nouă, citirea este oprită şi caracterul e înscris în str. Funcția completează automat str cu terminatorul de şir. Funcția întoarce adresa str în caz de succes sau EOF în caz de eşec.

int fputs(const char * str, FILE * stream_ptr);

Funcția fputs() scrie şirul str în fişier şi întoarce valoarea ultimului caracter înscris în caz de succes şi EOF în caz de eşec.

- **Transfer de blocuri de octeți**

Pentru transfer de blocuri de octeți se folosesc funcțiile fread(), respectiv fwrite() cu prototipurile:

unsigned fread(void * buf, int nr_oct, int cnt, FILE * strem_ptr);

unsigned fwrite(void * buf, int nr_oct, int cnt, FILE * strem_ptr);

unde:

buf = adresa zonei de memorie pentru date transferate

nr_oct = lungimea în octeți a fiecărui bloc;

cnt = numărul de blocuri cu lungimea nr_oct care trebuie citite/scrise;

Funcția **fread()** citeşte din fişier cnt blocuri de nr_oct octeți şi le înscrie în memorie, începând de la adresa buf. Funcția întoarce numărul de blocuri citite efectiv, care poate fi mai mic decât cnt dacă se termină fişierul sau intervine o eroare.

Funcția **fwrite()** scrie în fişier cnt blocuri de nr_oct octeți, preluate de la adresa buf din memorie. Funcția întoarce numărul de blocuri citite efectiv, care poate fi mai mic decât cnt dacă intervine o eroare.

În exemplul următor se înscriu date citite de la tastatură într-o zonă de memorie alocată

dinamic, folosindu-se pointerul lista. Toate datele sunt înscrise într-un fişier, copiindu-se ca blocuri de octeți cu funcția fwrite().

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h>

struct pers {    char nume[15];
                 char prenume[20];
            };

void main()
{ FILE * f1;
  pers * lista;
  int nr_pers, i, nr_inreg;

  if (!(f1=fopen("fis1.txt", "w+")))
    {
    printf("\neroare la deschiderea fisierului fis1.txt");
    getch();
    return;
    }
  printf("\n introduceti numarul persoanelor din lista:");
  scanf("%d", &nr_pers);
  lista=(pers *) malloc(sizeof(pers)*nr_pers);

  //citirea de la tastatura a numelor persoanelor din lista
  for(i=0;i<nr_pers;i++)
    {printf("\nnume=");
      scanf("%14s",lista[i].nume);
      printf("\nprenume=");
      scanf("%19s",lista[i].prenume);
    }

  //inregistrarea listei in fisier
  nr_inreg=fwrite(lista, sizeof(pers), nr_pers, f1);
  if (nr_inreg==nr_pers)
    printf("\nAu fost inregistrate in fisier toate cele %d persoane", nr_pers);
  else
    printf("\nInregistrare incorecta !");
  free(lista);
  fclose(f1);
}
```

12.3. Operații de citire/scriere cu formatare

Ca şi în lucrul cu consola şi în cazul utilizării fişierelor, se pot citi, respectiv scrie, date pentru care se precizează formatul de reprezentare. Prototipurile acestor funcții sunt:

int fscanf(FILE * stream_ptr, char *sir_formatare, …);

int fprintf(FILE * stream_ptr, char *sir_formatare, …);

Funcțiile sunt similare funcțiilor scanf(), respectiv printf(), sir_formatare fiind alcătuit în

mod similar acestora. Apare suplimentar, ca parametru, fişierul din sau în care se preiau/înscriu date.

Funcția fscanf() întoarce ca valoare numărul de valori pentru care citirea, conversia şi memorarea s-au efectuat corect sau EOF dacă s-a întâlnit sfârşitul fişierului.

Funcția fprintf() întoarce ca valoare numărul de octeți transferați sau EOF în caz de eşec.

În exemplul următor datele înscrise în fişier sunt înscrise cu specificarea formatului pentru fiecare valoare în parte folosindu-se funcția fprintf().

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h>

struct pers
{
        char nume[15];
        char prenume[20];
        int an_n;
};

void main()
{
 FILE * f1;
 pers * lista;
 int nr_pers, i, nr_inreg;

 if (!(f1=fopen("fis1.txt", "w+")))      // if (!(f1=fopen("fis1.txt", "a+")))
   {
  printf("\neroare la deschiderea fisierului fis1.txt");
  getch();
  return;
   }
 printf("\n introduceti numarul persoanelor din lista:");
 scanf("%d", &nr_pers);
 lista=(pers *) malloc(sizeof(pers)*nr_pers);

 //citirea de la tastatura a numelor persoanelor din lista
 for(i=0 ; i<nr_pers ; i++)
   {
    printf("\nnume=");
    scanf("%14s",lista[i].nume);
    printf("prenume=");
    scanf("%19s",lista[i].prenume);
    printf("anul nasterii:");
    scanf("%4d",&lista[i].an_n);
   }

 //inregistrarea listei in fisier
 for(i=0 ; i<nr_pers ; i++)
    fprintf(f1, "%14s %19s %4d\n",lista[i].nume, lista[i].prenume, lista[i].an_n);
 free(lista);
 fclose(f1);
}
```

12.4. Operații de citire/modificare a indicatorului de poziție

Anumite dispozitive (de exemplu fişierele disc) permit accesul aleator la date. Fişierul este o listă ordonată de octeți, numerotați începând de la 0, de lungime oarecare.

La deschiderea fişierului se inițializează o variabilă index care specifică poziția în fişier a octetului curent (indicator de poziție). Inițializarea se face cu valoarea 0 dacă deschiderea fişierului s-a făcut în mod 'r' sau 'w' şi cu numărul de octeți în cazul modului 'a'.

Funcțiile de transfer efectuează citirea sau scrierea începând cu octetul din fişier corespunzător valorii curente a indicatorului de poziție. După fiecare operație de transfer, indicatorul avansează cu numărul de octeți citiți sau scrişi.

- **Citirea valorii indicatorului de poziție**

long int ftell(FILE* stream_ptr);

Funcția ftell() întoarce poziția curentă a fişier în caz de succes sau -1 în caz de eşec.

- **Citirea şi modificarea valorii indicatorului de poziție**

int fgetpos(FILE* stream_ptr, const long int * poz);

Funcția fgetpos() înscrie valoarea indicatorului în variabila indicată de poz şi întoarce 0 în caz de succes sau -1 în caz de eşec.

int fsetpos(FILE* stream_ptr, const long int * poz);

Funcția fsetpos() atribuie indicatorului valoarea variabilei indicată de poz şi întoarce 0 în caz de succes sau -1 în caz de eşec.

- **Modificarea valorii indicatorului de poziție**

Indicatorului de poziție se poate muta în interiorul fişierului folosind funcția fseek() care are sintaxa:

int fseek(FILE* stream_ptr, long int * poz, int origine);

unde: origine = poziția de referință la care se adună nr_octeti relativ la poziția precizată care poate avea valorile:

0 = SEEK_SET = început de fişier;
1 = SEEK_CUR = poziție curentă;
2 = SEEK_END = sfârşit de fişier;

Funcția fseek() face o deplasare a indicatorului cu nr_octeti relativ la referința precizată.

- **Poziționarea indicatorului la începutul fişierului**

void rewind(FILE * stream_ptr)

Funcția rewind poziționează indicatorul la începutul fişierului, permițând parcurgerea în mod repetat a fişierului de la începutul său.

În continuare este prezentat un program care deschide un fişier în modul a+ (append) care permite adăugarea de informație. În fişier se înscriu date citite de la tastatură. Pentru afişarea informațiilor din fişier, se face repoziționare a indicatorului folosindu-se funcția rewind().

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

struct ex
        {  char nume[20];
           int nr;
        };

void main()
{
   FILE * f;
   ex tab[20], v;
   int c, i;

   if (!(f=fopen("exemplu.txt","a+")))         // se deschide fişierul exemplu.txt în modul
                                               // append şi se testează reuşita operației
       puts("fisierul nu poate fi deschis");
   else
   {
       do
       {
        printf("citire date:"); // se citesc date de la tastatură
        fflush(stdin);
        printf("\nNume=");
        scanf("%19s ", v.nume);
        printf("\nNr=");
        scanf("%d", &v.nr);
        fprintf(f,"%19s %5d\n",v.nume,v.nr);         // se înscriu datele în fişier
        printf("continui?");
        fflush(stdin);
        c=getchar();
       }
       while((c!='n')&&(c!='N'));     // finalizarea secvenței de citire se
                                      // face la apăsarea tastei n sau N

       rewind(f);                     // repoziționarea la începutul fişierului

      i=0;

      while (!(feof(f)))              // se parcurge fişierul până la întâlnirea eof
      {
          if(fscanf(f,"%19s %5d",tab[i].nume, &tab[i].nr)!=EOF) // se citesc date din
                                                                // fişier
        {
         printf("%s %d\t%ld\n", tab[i].nume, tab[i].nr,ftell(f)); // datele citite din fişier
                                                                  //  se afişează pe ecran
         i++;
        }
      }
   }
  fclose(f);
}
```