FIŞIERE TEXT. FIŞIERE BINARE.
FUNCŢII DE PRELUCRARE A FIŞIERELOR

1. Conţinutul lucrării

În lucrare sunt prezentate funcţiile de prelucrare a fişierelor de nivel superior, utilizând structuri speciale de tip FILE. Principalele operaţii care se pot efectua asupra fişierelor la acest nivel sunt: crearea, deschiderea, citirea/scrierea unui caracter sau a unui şir de caractere, citirea/scrierea binară a unui număr de articole, poziţionarea într-un fişier, închiderea unui fişier, vidarea zonei tampon a unui fişier.

2. Consideraţii teoretice

La acest nivel, fiecărui fişier i se ataşează un pointer la o structură de tip FILE:

FILE *p;

Tipul FILE şi toate prototipurile funcţiilor de prelucrare se găsesc în fişierul **stdio.h**

2.1. Deschiderea unui fişier

Deschiderea unui fişier existent, precum şi crearea unui fişier nou se fac cu ajutorul funcţiei **fopen**, care are următorul prototip:

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

unde:

- **cale_nume** este un pointer spre un şir de caractere care defineşte calea de nume a fişierului;
- **mod** este un pointer spre un şir de caractere care defineşte modul de prelucrare a fişierului deschis, după cum urmează:

  “r” – citire în mod text (read);
  “w” – scriere în mod text (write);
  “a” – adăugare în mod text (append);
  “r+” – citire/scriere în mod text (modificare);
  “w+” – citire/scriere în mod text (creare);
  “rb” – citire în mod binar;
  “wb” – scriere în mod binar;
  “r+b” – citire/scriere în mod binar (modificare);
  „w+b” – citire/scriere în mod binar (creare);
  “ab” – adăugare de înregistrări în mod binar.

Conţinutul unui fişier existent deschis în scriere „w” ,va fi şters, el considerându-se deschis în creare.

Dacă fişierul este deschis în modul „a”, se vor putea adăuga noi înregistrări după ultima înregistrare existentă în fişier.

Un fişier inexistent deschis în modul „w” sau „a” va fi creat.

Funcţia **fopen** returnează un pointer spre tipul FILE în caz de succes sau pointerul nul în caz de eroare.

Fişierele standard de intrare/ieşire sunt deschise automat la lansarea programului. Pentru ele, pointerii de tipul FILE sunt:

stdin – intrare standard;
stdout – ieşire standard;
stderr – ieşire standard erori.

2.2. Citirea/scrierea unui caracter

Un fişier poate fi prelucrat, în citire sau scriere, caracter cu caracter, folosind funcţiile **fgetc** şi **fputc,** ale căror prototipuri sunt:

```
int fgetc (FILE *pf);
int fputc (int ch, FILE *pf);
```

în care:

- **pf** este pointerul de tipul FILE returnat de funcţia **fopen**;
- **ch** este codul ASCII al caracterului care se scrie.

Funcţia **fgetc** returnează codul ASCII al caracterului scris. În caz de eroare ambele returnează –1.

De exemplu, secvenţa de copiere a intrării standard la ieşirea standard este:

```
while ((c=fgetc(stdin))!=EOF)
        fputc(c, stdout);
```

2.3. Citirea/scrierea unui şir de caractere

Citirea dintr-un fişier a unui şir de caractere se face cu ajutorul funcţiei **fgets**, care are prototipul:

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

în care:

- **s** este pointerul spre zona din memorie unde are loc păstrarea şirului de caractere;
- **n** este numărul de octeţi a zonei în care se citesc caracterele din fişier; citirea se opreşte la întâlnirea caracterului '\n' sau citirea a cel mult n-1 caractere; ultimul caracter în ambele cazuri va fi '\0';
- **pf** este pointerul de tipul FILE.

Funcţia returnează valoarea pointerului **s**. La întâlnirea sfârşitului de fişier funcţia returnează valoarea zero.

Scrierea unui şir de caractere (inclusiv caracterul „\0") se face cu funcţia **fputs,** care are prototipul:

```
int fputs (const char *s, FILE *pf);
```

unde **s** este pointerul spre începutul zonei de memorie care conţine şirul de caractere care se scrie în fişier.

Funcţia **fputs** returnează codul ASCII al ultimului caracter scris în fişier sau –1 în caz de eroare.

2.4. Citirea/scrierea cu format

Citirea/scrierea cu format se poate face cu ajutorul funcţiilor **fscanf/fprintf,** similare cu funcţiile **sscanf/sprintf,** prezentate în lucrarea L02, deosebirea constând în faptul că în cadrul funcţiilor **sscanf/sprintf** se precizează ca prim parametru pointerul zonei unde se păstrează şirul de caractere, iar în cadrul funcţiilor **fscanf/fprintf** se precizează ca prim parametru pointerul de tipul FILE, aşa cum reiese din prototipurile lor:

```
int fscanf (FILE *pf, const char *format, …);
int fprintf (FILE *pf, const char *format, …);
```

Funcţia **fscanf** returnează numărul de câmpuri citite corect; la întâlnirea sfârşitului de fişier funcţia returnează valoarea EOF.

Funcţia **fprintf** returnează numărul caracterelor scrise în fişier sau –1 în caz de eroare.

2.5. Vidarea zonei tampon a unui fişier

Vidarea zonei tampon a unui fişier se face cu ajutorul funcţiei **fflush**, având prototipul următor:

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

unde **pf** este pointerul de tipul FILE.

Dacă fişierul e deschis în scriere, conţinutul zonei tampon se scrie în fişierul respectiv.
Dacă fişierul e deschis în citire, caracterele necitite se pierd.
Funcţia returnează zero în caz de succes şi –1 în caz de eroare.

2.6. Poziţionarea într-un fişier

Poziţionarea într-un fişier pe un anumit octet se poate face cu ajutorul funcţiei **fseek,** care are prototipul:

```
int fseek (FILE *pf, long increment, int origine);
```

în care:

- **pf** este pointerul de tipul FILE;
- **increment** este numărul de octeţi peste care se va poziţiona indicatorul în fişier, ţinând cont de parametrul origine;
- **origine** are una din valorile:
  SEEK_SET – incrementul se consideră faţă de începutul fişierului;
  SEEK_CUR – incrementul se consideră faţă de poziţia curentă a indicatorului în fişier;
  SEEK_END – incrementul se consideră faţă de sfârşitul fişierului.

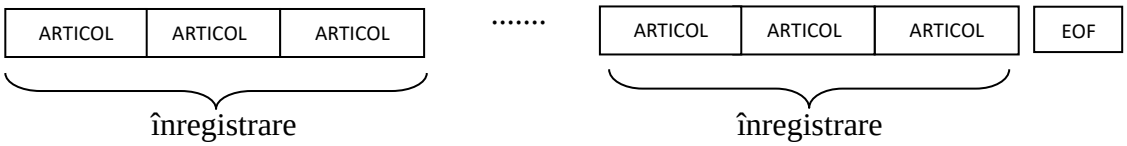
Funcţia **fseek** returnează deplasamentul rezultat faţă de începutul fişierului sau -1 în caz de eroare.

Poziţia curentă a indicatorului într-un fişier, dată prin deplasamentul în octeţi faţă de începutul său este returnată de către funcţia **ftell** de prototip:

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

2.7. Prelucrarea fişierelor binare

În acest caz, fişierele sunt considerate ca o succesiune de înregistrări, fiecare înregistrare conţinând un număr de articole, ca în figura următoare:



Articolele sunt de lungime fixă. Un articol este o dată de un tip predefinit sau definit de utilizator.

Citirea, respectiv scrierea unei înregistrări se face cu ajutorul funcţiilor **fread** şi **fwrite**, care au prototipurile:

```
unsigned fread (void *buf, unsigned dim, unsigned nrart, FILE *pf);
unsigned fwrite (void *buf, unsigned dim, unsigned nrart, FILE *pf);
```

unde:

– **buf** este pointerul spre zona tampon care conţine articolele citite, respectiv cele care se scriu;
– **dim** – este dimensiunea unui articol în octeţi;
– nrart – numărul articolelor dintr-o înregistrare;
– pf – este pointerul de tipul FILE.

Funcţiile returnează numărul articolelor citite, respectiv scrise în caz de succes, sau –1 în caz de eroare.

2.8. Închiderea unui fişier

Închiderea unui fişier se realizează cu ajutorul funcţiei **fclose**, care are prototipul:

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

unde pf este pointerul de tipul FILE returnat de **fopen.**

Funcţia returnează 0 în caz de succes şi –1 în caz de eroare.

2.9. Ştergerea unui fişier

Un fişier închis poate fi şters cu ajutorul funcţiei **remove,** care are prototipul:

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

unde **cale_nume** este un pointer spre un şir de caractere care redă calea de nume a fişierului.

Funcţia returnează 0 în caz de succes şi o valoare negativă în caz de eroare.

2.10. Exemple

În programul L11Ex1 este creat un fişier, caracter cu caracter, citite de la tastatură. Apoi este ilustrat modul de adăugare la sfârşitul fişierului, de data aceasta, a unor şiruri de caractere. La sfârşit fişierul este listat linie cu linie, cu numerotarea lor.

```
/* Programul L11Ex1 */

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

// Programul ilustrează prelucrarea fisierului pe caractere si siruri de caractere

int main()
{
  char ch, s[100], nume_fis[50]="D:\\fisier1.txt";
  int i;
  FILE *pf;
  // crearea fişierului; scrierea caracterelor inclusiv '\n' introduse de la tastatura
  pf=fopen(nume_fis,"w");
  printf("Introduceti textul. Pentru a termina apasati CTRL+Z si ENTER \n");
  while ((ch=fgetc(stdin))!=EOF)
  {
    fputc(ch,pf);
  }
  fclose(pf);
  /*Adaugarea de siruri de caractere*/
  pf=fopen(nume_fis,"r+");
  fseek(pf,0l,SEEK_END);
  printf("Introduceti sirurile de caractere terminate cu ENTER. Pentru a termina\
        apasati CTRL+Z si ENTER\n");
  while(fgets(s,100,stdin)!=(char*)0)
  {
    fputs(s,pf);
  }
  fclose(pf);
  /*Afisarea continutului */
  printf("CONTINUTUL FISIERULUI cu NUMEROTAREA LINIILOR:\n");
  i=0;
  pf=fopen(nume_fis,"r");
  while(fgets(s,100,pf)!=(char *)0)
  {
    printf("%d %s", i, s);
    i++;
  }
  fclose(pf);
  printf("Dupa apasarea unei taste fisierul va fi sters!\n");
  getch();
  remove(nume_fis);
  return 0;
}
```

Programul L11Ex2 ilustrează modul de prelucrare binară a unui fişier. Programul conţine crearea fişierului şi afişarea conţinutului său.

```
/* Programul L11Ex2 */

#include <stdio.h>
/* Programul ilustreaza prelucrarea binara a unui fisier */
typedef struct {
                    char nume[40];
                    long suma;
      } ARTICOL;

 void afisare(char *nume_fis)  {
    FILE *pf;
    ARTICOL buf;
    int i;
    pf=fopen(nume_fis,"rb");
    printf("NR.CRT.     SUMA   NUMELE-PRENUMELE\n");
    i=0;
    while(fread(&buf,sizeof(ARTICOL),1,pf)>0)
    {
       printf("%6d %10ld   %-40s\n",i,buf.suma,buf.nume);
       i++;
    }
    fclose(pf);
 }

 int main()  {
   FILE *pf;
   ARTICOL buf;
   int i,n;
   char nume_fis[40]="D:\\fisier2.dat";
   /*Crearea fisierului */
   printf("Introduceti numarul de persoane: ");
   scanf("%d",&n);
   pf=fopen(nume_fis,"wb");
   for(i=1;i<=n;i++)
   {
      fflush(stdin);
      printf("Numele persoanei: ");
      fgets(buf.nume,40,stdin);
      printf("Suma = ");
      scanf("%ld",&buf.suma);
      fwrite(&buf,sizeof(ARTICOL),1,pf);
   }
   fclose(pf);
   printf("\nCONTINUTUL FISIERULUI\n");
   afisare(nume_fis);
   return 0;
 }
```

3. Mersul lucrării

3.1. Se vor analiza şi executa exemplele din lucrare și din materialul de curs.

3.2. Să se creeze un fişier care să conţină produsele unui magazin. Un produs este reprezentat printr-o structură ce conţine codul produsului, denumirea, unitatea de măsură, cantitatea, preţul unitar. Plecând de la acest fişier, să se obţină un fişier sortat după codul produsului.

3.3. Având creat fişierul sortat la punctul 3.2. se vor scrie funcţii de intrare şi de ieşire a produselor din magazin.

3.4. Se va scrie un program pentru admiterea la facultate în anul I. Programul va cuprinde crearea fişierului cu candidaţii înscrişi. În final trebuie să se obţină fişierele cu candidaţii admişi pe secţii şi cei respinşi pe baza mediei generale obţinute calculată astfel: 0.2*media_bacalaureat + 0.8*test_matematica.

3.5. Se consideră un director de fişiere. Fiecare intrare în director conţine numele (8 caractere) şi extensia (3 caractere) fişierului, numărul de octeți alocați pentru el, data şi ora ultimei actualizări (zi, lună, an, oră, minut, secundă).

Se cere:

a) crearea directorului de fişiere;
b) afişarea directorului în ordine alfabetică a numelor fişierelor;
c) afişarea directorului în ordine crescătoare a ultimei actualizări.

3.6. Se citeşte de la tastatură un text care se scrie într-un fişier "poveste.txt". Să se afişeze apoi conţinutul fişierului, fiecare linie fiind precedată de numărul de ordine al ei.

3.7. De la tastatură se citesc partea reală şi partea imaginară pentru n numere complexe. Să se creeze atât un fişier text cât și un fişier binar care conţin numerele complexe citite (partea reală și partea imaginară) împreună cu modulele și argumentele lor.

3.8. Două firme îşi păstrează informaţiile referitoare la stocul de mărfuri (cod produs, denumire, cantitate, preţ unitar) în fişierele binare "marfa1.dat" şi respectiv "marfa2.dat", ordonate crescător după cod. Prin fuzionarea celor două firme, rezultă un stoc comun care trebuie memorat în fişierul binar "marfa.dat", ordonat după cod.

a) Să se creeze fişierele iniţiale, pe baza datelor introduse de la tastatură şi apoi să se creeze fişierul binar de stoc comun "marfa.dat". Pentru mărfuri cu cod comun, se consideră că denumirea şi preţul unitar corespund;

b) Fişierul "marfa.dat" se va parcurge secvenţial, tipărind pentru fiecare articol denumirea şi cantitatea;

c) Pentru o componentă dorită dată prin numărul de ordine, se va modifica direct preţul său unitar în fișierul "marfa.dat".

3.9. Să se scrie programul pentru concatenarea mai multor fişiere ce conţin numere reale. Numele fișierelor sunt trimise ca și argumente la execuția programului. Se va tipări informaţia din fişierul rezultat.