15. FIŞIERE

15.1. Operaţii de intrare / ieşire

În limbajul C nu există instrucţiuni de intrare / ieşire. Operaţiile de intrare / ieşire sunt realizate prin apelul unor funcţii ale sistemului de operare. Acestea sunt implementate prin funcţii, sub o formă compatibilă pentru diversele sisteme de operare (sunt portabile).

Un *fişier* este o colecţie ordonată de ***articole*** (înregistrări) păstrate pe un suport extern de memorie şi identificate printr-un nume.

Pentru ***fişierul standard de intrare***, datele sunt introduse de la tastatură.

Pentru ***fişierul standard de ieşire***, rezultatele sunt afişate pe terminalul standard de ieşire.

Mesajele de eroare se afişează în ***fişierul standard de eroare***.

Fişierul are un articol care marchează ***sfârşitul fişierului***. Pentru fişierul standard de intrare de la tastatură, *sfârşitul de fişier*, pentru sistemele de operare DOS şi Windows se generează prin **`Ctrl-Z`** (pentru Unix – prin **`Ctrl-D`**).

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.

Prelucrarea fişierelor se face pe două niveluri:

- nivelul inferior, care apelează direct la sistemul de operare;
- nivelul superior, care utilizează structuri speciale **`FILE`** .

Funcţiile de pe nivelul superior nu asigură o independenţă totală faţă de sistemul de operare.

Funcţiile standard de intrare / ieşire au prototipurile în fişierul antet **`<stdio.h>`** .

15.2. Fişiere text şi fişiere binare

Într-un fişier text, toate datele sunt memorate ca şiruri de caractere, organizate pe linii, separate între ele prin marcajul sfârşit de linie **'\n'**.

Într-un fişier text spaţiul de memorare pe disc nu este folosit în mod eficient pentru datele numerice (astfel întregul **12345** ocupă 5 octeţi).

Într-un fişier binar, datele sunt păstrate în formatul lor intern (2 octeţi pentru **int**, 4 octeţi pentru **float**, etc).

La fişierele text ***marcajul de sfârşit de fişier*** (caracterul **0X1A**) există fizic în fişier. La întâlnirea acestui caracter funcţia **fgetc()** întoarce **EOF (-1)**. Marcajul de sfârşit de fişier se generează de la tastatură prin **Ctrl-Z**.

În cazul *fişierelor binare*, marcajul de sfârşit de fişier nu există fizic în fişier, ci este generat de funcţia **fgetc()**.

În MSDOS (şi în Unix), la nivelul liniei de comandă intrările şi ieşirile standard pot fi redirectate în fişiere disc, fără a opera nici o modificare la nivelul programului. Astfel:

< redirectează intrarea standard către fişierul specificat;

> redirectează ieşirea standard către fişierul specificat.

Fişierul standard de eroare nu poate fi redirectat.

Fişierul specificat poate fi:

con − pentru consola sistem (tastatura, respectiv ecranul);

prn − pentru imprimanta paralelă;

com1 − pentru interfaţa serială de date;

nume_fişier − pentru un fişier disc;

NUL − pentru perifericul nul.

Exemple:

> test.exe > prn redirectează ieşirea programului la imprimantă

> test.exe < f1.dat > f2.dat redirectează atât intrarea cât şi ieşirea programului

15.3. Accesul la fişiere

Fişierele disc şi fişierele standard sunt gestionate prin pointeri la structuri specializate **FILE**, care se asociază fiecărui fişier pe durata prelucrării.

Fişierele standard au pointerii predefiniţi: **stdin**, **stdout**, **stderr**, **stdprn**, **stdaux**.

Declararea unui pointer la fişier se face prin:

```
FILE *pf;
```

1. deschiderea unui fişier:

Înainte de a fi prelucrat, un fişier trebuie să fie deschis. Prin deschidere:
- se asociază unui *nume de fişier* un *pointer la fişier*;
- se stabileşte un *mod de acces* la fişier.

Pentru deschiderea unui fişier se foloseşte funcţia cu prototipul:

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

Prin deschiderea unui fişier se stabileşte o conexiune logică între fişier şi variabila pointer şi se alocă o zonă de memorie (buffer) pentru realizarea mai eficientă a operaţiilor de intrare / ieşire. Funcţia întoarce:
- un pointer la fişier, în caz că deschiderea fişierului se face în mod corect;
- **NULL** dacă fişierul nu poate fi deschis.

Vom considera mai întâi două ***moduri de acces***:
- ***citire*** (sau consultare) **"r"** − citirea dintr-un fişier inexistent va genera eroare;
- ***scriere*** (sau creare) **"w"** − dacă fişierul există deja, el va fi şters.

Fişierele standard nu trebuiesc deschise.
Redirectarea unui fişier deschis poate fi realizată cu:

```
FILE* freopen(char* nume, char* mod, FILE* flux));
```

Fişierul deschis este închis şi este deschis un nou fişier având ca sursă fluxul, numele şi modul de acces specificaţi ca parametri. O utilizare importantă o constituie redirectarea fluxului standard de intrare: datele vor fi citite din fişierul specificat, fără a face nici o modificare în program.

2. închiderea unui fişier

După terminarea prelucrărilor asupra unui fişier, acesta trebuie închis. Un fişier este închis automat la apelarea funcţiei **exit()**.

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

- funcţia întoarce 0 la închidere normală şi **EOF** la producerea unui incident;
- fişierele standard nu se închid de către programator;
- în cazul unui fişier de ieşire, se scriu datele rămase nescrise din buffer în fişier, aşa că operaţia de închidere este obligatorie;
- în cazul unui fişier de intrare, datele necitite din bufferul de intrare sunt abandonate;
- se eliberează bufferele alocate;
- se întrerupe conexiunea pointer – fişier.

V-246

Programare în C

Valeriu Iorga

12.2. Funcţii care întorc pointeri
12.3. Pointeri la funcţii
12.4. Declaraţii complexe (şarade)

13. TABLOURI ŞI POINTERI (2)
13.1. Pointeri la pointeri
13.2. Tablouri multidimensionale
13.3. Probleme propuse

14. STRUCTURI
14.1. Definirea tipurilor structurilor şi declararea variabilelor structuri
14.2. Iniţializarea structurilor
14.3. Accesul la câmpurile structurilor
14.4. Pointeri la structuri
14.5. Atribuiri de structuri
14.6. Structuri şi funcţii
14.7. Uniuni
14.8. Probleme rezolvate
14.9. Probleme propuse

15. FIŞIERE
15.1. Operaţii de intrare / ieşire
15.2. Fişiere text şi fişiere binare
15.3. Accesul la fişiere
15.4. Operaţii de intrare – ieşire
15.5. Probleme rezolvate
15.6. Probleme propuse

16. FUNCŢII (3)
16.1. Funcţii recursive
16.1.1. Exemple de definiţii recursive
16.1.2. Recursivitate liniară
16.1.3. Recursivitate binară
16.1.4. Exemple de probleme rezolvate recursiv
16.2. Funcţii cu număr variabil de parametri
16.3. Funcţii polimorfice
16.4. Probleme rezolvate
16.5. Probleme propuse

17. PREPROCESORUL
17.1. Definirea de constante simbolice
17.2. Substituirea textuală (definirea de macroinstrucţiuni)
17.3. Includerea de fişiere