

15. FIȘIERE

15.1. Operații de intrare / ieșire

În limbajul C nu există instrucțiuni de intrare / ieșire. Operațiile de intrare / ieșire sunt realizate prin apelul unor funcții ale sistemului de operare. Acestea sunt implementate prin funcții, sub o formă compatibilă pentru diversele sisteme de operare (sunt portabile).

Un *fișier* este o colecție ordonată de *articole* (înregistrări) păstrate pe un suport extern de memorie și identificate printr-un nume.

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

Pentru *fișierul standard de ieșire*, rezultatele sunt afișate pe terminalul standard de ieșire.

Mesajele de eroare se afișează în *fișierul standard de eroare*.

Fișierul are un articol care marchează *sfârșitul fișierului*. Pentru fișierul standard de intrare de la tastatură, *sfârșitul de fișier*, pentru sistemele de operare DOS și Windows se generează prin **Ctrl-Z** (pentru Unix – prin **Ctrl-D**).

Operațiile specifice prelucrării fișierelor sunt:

- deschiderea unui fișier;
- închiderea unui fișier;
- creerea unui fișier;
- citirea de articole din fișier (consultarea fișierului);
- actualizarea (sau modificarea) fișierului;
- adăugare de articole la sfârșitul fișierului;
- poziționarea în fișier;
- ștergerea unui fișier;
- schimbarea numelui unui fișier.

Prelucrarea fișierelor se face pe două niveluri:

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

Funcțiile de pe nivelul superior nu asigură o independență totală față de sistemul de operare.

Funcțiile standard de intrare / ieșire au prototipurile în fișierul antet `<stdio.h>`.

15.2. Fișiere text și fișiere binare

Într-un fișier text, toate datele sunt memorate ca șiruri de caractere, organizate pe linii, separate între ele prin marcajul sfârșit de linie '\n' .

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

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

La fișierele text *marcajul de sfârșit de fișier* (caracterul **0x1A**) există fizic în fișier. La întâlnirea acestui caracter funcția **fgetc()** întoarce **EOF (-1)** . Marcajul de sfârșit de fișier se generează de la tastatură prin **Ctrl-Z** .

În cazul *fișierelor binare*, marcajul de sfârșit de fișier nu există fizic în fișier, ci este generat de funcția **fgetc()** .

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

- < redirectează intrarea standard către fișierul specificat;
- > redirectează ieșirea standard către fișierul specificat.

Fișierul standard de eroare nu poate fi redirectat.

Fișierul specificat poate fi:

- con** – pentru consola sistem (tastatura, respectiv ecranul);
- prn** – pentru imprimanta paralelă;
- com1** – pentru interfața serială de date;
- nume_fișier** – pentru un fișier disc;
- NUL** – pentru perifericul nul.

Exemple:

- > **test.exe > prn** redirectează ieșirea programului la imprimantă
- > **test.exe < f1.dat > f2.dat** redirectează atât intrarea cât și ieșirea programului

15.3. Accesul la fișiere

Fișierele disc și fișierele standard sunt gestionate prin pointeri la structuri specializate **FILE**, care se asociază fiecărui fișier pe durata prelucrării.

Fișierele standard au pointerii predefiniți: **stdin**, **stdout**, **stderr**, **stdprn**, **stdaux** .

Declararea unui pointer la fișier se face prin:

```
FILE *pf;
```

1. deschiderea unui fișier:

Înainte de a fi prelucrat, un fișier trebuie să fie deschis. Prin deschidere:

- se asociază unui *nume de fișier* un *pointer la fișier*;
- se stabilește un *mod de acces* la fișier.

Pentru deschiderea unui fișier se folosește funcția cu prototipul:

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

Prin deschiderea unui fișier se stabilește o conexiune logică între fișier și variabila pointer și se alocă o zonă de memorie (buffer) pentru realizarea mai eficientă a operațiilor de intrare / ieșire.

Funcția întoarce:

- un pointer la fișier, în caz că deschiderea fișierului se face în mod corect;
- **NULL** dacă fișierul nu poate fi deschis.

Vom considera mai întâi două *moduri de acces*:

- *citire* (sau consultare) "**r**" – citirea dintr-un fișier inexistent va genera eroare;
- *scriere* (sau creare) "**w**" – dacă fișierul există deja, el va fi șters.

Fișierele standard nu trebuiesc deschise.

Redirectarea unui fișier deschis poate fi realizată cu:

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

Fișierul deschis este închis și este deschis un nou fișier având ca sursă fluxul, numele și modul de acces specificați ca parametri. O utilizare importantă o constituie redirectarea fluxului standard de intrare: datele vor fi citite din fișierul specificat, fără a face nici o modificare în program.

2. închiderea unui fișier

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

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

- funcția întoarce 0 la închidere normală și **EOF** la producerea unui incident;
- fișierele standard nu se închid de către programator;
- în cazul unui fișier de ieșire, se scriu datele rămase nescrise din buffer în fișier, așa că operația de închidere este obligatorie;
- în cazul unui fișier de intrare, datele necitite din bufferul de intrare sunt abandonate;
- se eliberează bufferele alocate;
- se întrerupe conexiunea pointer – fișier.

V-246

Programare în C

Valeriu Iorga

1. UN TUR DE ORIZONT ÎN LIMBAJUL C

- 1.1. Structura unui program C foarte simplu
- 1.2. Elemente necesare scrierii unor programe C foarte simple
 - 1.2.1. Directiva **define**
 - 1.2.2. Tipuri
 - 1.2.3. Definiții și declarații de variabile
 - 1.2.4. Atribuirea
 - 1.2.5. Decizia
 - 1.2.6. Ciclul
 - 1.2.7. Afișarea valorii unei expresii (descriptori)
 - 1.2.8. Citirea valorilor de la terminal
- 1.3. Structura unui program

2. ELEMENTELE FUNDAMENTALE ALE LIMBAJULUI C

- 2.1. Alfabetul limbajului
- 2.2. Atomi lexicali
 - 2.2.1. Identificatori
 - 2.2.2. Cuvinte cheie
 - 2.2.3. Literali
 - 2.2.4. Șiruri de caractere
 - 2.2.5. Comentarii
 - 2.2.6. Terminatorul de instrucțiune
 - 2.2.7. Constante
- 2.3. Ciclul de dezvoltare al unui program
 - 2.3.1. Definierea problemei de rezolvat
 - 2.3.2. Identificarea pașilor necesari pentru rezolvarea problemei
 - 2.3.3. Proiectarea algoritmului
 - 2.3.4. Scrierea programului folosind un limbaj de programare
 - 2.3.5. Implementarea programului: editare, compilare, editare de legături, execuție
 - 2.3.6. Testarea și depanarea programului (**debugging**)

3. TIPURI ȘI VARIABILE

- 3.1. Introducere
- 3.2. Tipuri fundamentale
 - 3.2.1. Caracterele (tipul **char**)
 - 3.2.2. Întregii (tipul **int**)
 - 3.2.3. Realii (tipurile **float** și **double**)

- 3.2.4. Definiri de tip cu **typedef**
- 3.2.5. Tipuri enumerate
- 3.2.6. Tipul vid (**void**)
- 3.3. Tipuri derivate
- 3.4. Declararea variabilelor
- 3.5. Echivalența tipurilor

4. OPERATORI ȘI EXPRESII

- 4.1. Conversii de tip
 - 4.1.1. Conversii implicite de tip
 - 4.1.2. Conversii aritmetice
 - 4.1.3. Conversiile de tip explicite (**cast**)
- 4.2. Operatorii aritmetici
- 4.3. Operatorii de atribuire
- 4.4. Operatorii relaționali
- 4.5. Operatorii booleeni
- 4.6. Operatorii binari (la nivel de biți)
- 4.7. Operatorul condițional
- 4.8. Operatorul secvență
- 4.9. Operatori unari
- 4.10. Ordinea evaluării operanzilor

5. INSTRUCȚIUNI

- 5.1. Instrucțiunea expresie
- 5.2. Instrucțiunea compusă (blocul)
- 5.3. Instrucțiunea vidă
- 5.4. Instrucțiunea **if**
- 5.5. Instrucțiunea **switch**
- 5.6. Instrucțiunea **while**
- 5.7. Instrucțiunea **do..while**
- 5.8. Instrucțiunea **for**
- 5.9. Instrucțiunea **continue**
- 5.10. Instrucțiunea **break**
- 5.11. Instrucțiunea **goto**
- 5.12. Instrucțiunea **return**
- 5.13. Operații de intrare / ieșire
- 5.14. Probleme rezolvate
- 5.15. Probleme propuse

6. FUNCȚII (1)

- 6.1. Apelarea funcțiilor
- 6.2. Definiții de funcții

- 6.3. Comunicarea între funcții prin variabile externe. Efecte laterale ale funcțiilor
- 6.4. Funcții care apelează alte funcții
- 6.5. Programe cu mai multe fișiere sursă
- 6.6. Fișiere antet
- 6.7. Funcții matematice uzuale
- 6.8. Probleme rezolvate
- 6.9. Probleme propuse

7. VARIABLE

- 7.1. Variabile externe (globale) și variabile interne (locale)
- 7.2. Domenii de vizibilitate ale variabilelor
- 7.3. Clase de memorare
 - 7.3.1. Variabile și funcții statice
 - 7.3.2. Variabile regiștri
 - 7.3.3. Inițializări

8. TABLOURI

- 8.1. Tablouri cu o dimensiune (vectori)
- 8.2. Probleme rezolvate
- 8.3. Probleme propuse

9. POINTERI

- 9.1. Operatori specifici pointerilor
- 9.2. Pointeri generici (pointeri `void`)
- 9.3. Pointeri constanți și pointeri la constante
- 9.4. Operații aritmetice cu pointeri
- 9.5. Legătura între pointeri și tablouri
- 9.6. Parametri tablouri

10. ȘIRURI DE CARACTERE

- 10.1. Generalități
- 10.2. Funcții de intrare / ieșire relative la șiruri de caractere
- 10.3. Tablouri de pointeri
- 10.4. Parametrii liniei de comandă
- 10.6. Probleme propuse (Șiruri de caractere)

11. ALOCAREA DINAMICĂ A MEMORIEI

- 11.1. Funcții pentru gestiunea dinamică a memoriei
- 11.2. Probleme rezolvate

12. FUNCȚII (2)

- 12.1. Mecanisme de transfer ale parametrilor

- 12.2. Funcții care întorc pointeri
- 12.3. Pointeri la funcții
- 12.4. Declarații complexe (șarade)

13. TABLOURI ȘI POINTERI (2)

- 13.1. Pointeri la pointeri
- 13.2. Tablouri multidimensionale
- 13.3. Probleme propuse

14. STRUCTURI

- 14.1. Definirea tipurilor structurilor și declararea variabilelor structuri
- 14.2. Inițializarea structurilor
- 14.3. Accesul la câmpurile structurilor
- 14.4. Pointeri la structuri
- 14.5. Atribuirii de structuri
- 14.6. Structuri și funcții
- 14.7. Uniuni
- 14.8. Probleme rezolvate
- 14.9. Probleme propuse

15. FIȘIERE

- 15.1. Operații de intrare / ieșire
- 15.2. Fișiere text și fișiere binare
- 15.3. Accesul la fișiere
- 15.4. Operații de intrare – ieșire
- 15.5. Probleme rezolvate
- 15.6. Probleme propuse

16. FUNCȚII (3)

- 16.1. Funcții recursive
 - 16.1.1. Exemple de definiții recursive
 - 16.1.2. Recursivitate liniară
 - 16.1.3. Recursivitate binară
 - 16.1.4. Exemple de probleme rezolvate recursiv
- 16.2. Funcții cu număr variabil de parametri
- 16.3. Funcții polimorfice
- 16.4. Probleme rezolvate
- 16.5. Probleme propuse

17. PREPROCESORUL

- 17.1. Definirea de constante simbolice
- 17.2. Substituirea textuală (definirea de macroinstrucțiuni)
- 17.3. Includerea de fișiere

17.4. Compilarea condiționată

18. DEZVOLTAREA PROGRAMELOR MARI

18.1. Compilare separată

18.2. Utilitarul **make**