FIȘIERE BINARE

Fișiere:

1. Fișiere binare – un octet nu are o semnificație specială

2. Fișiere text – un octet reprezintă codul ASCII al unui caracter, iar textul este împărțit pe linii folosind caracterele CR+LF (`'\r' + '\n'`)

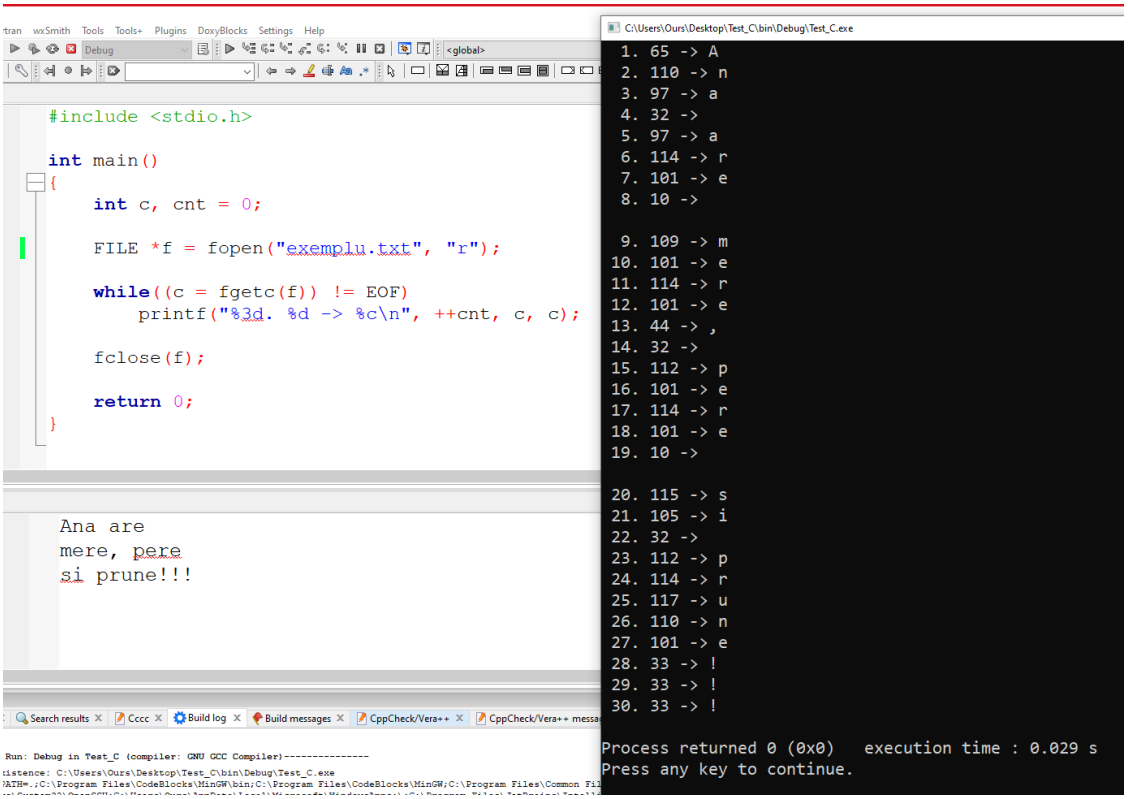
Deschiderea unui fișier binar:

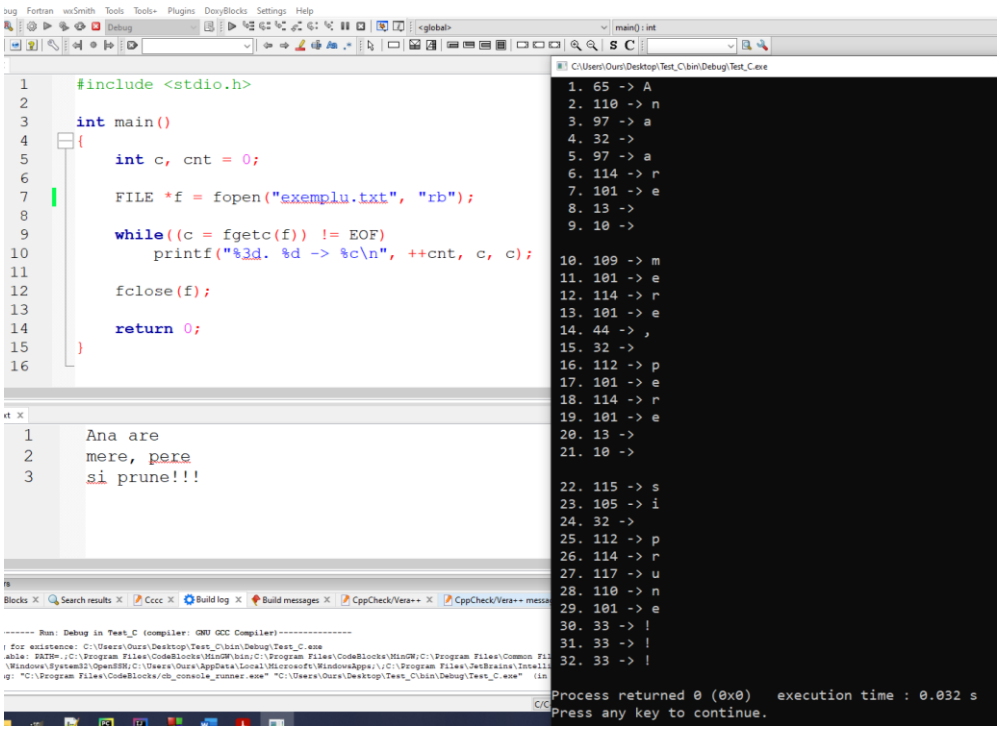
- **Funcția fopen cu modurile simple: "rb", "wb" și "ab" (b = binary)**
- **Funcția fopen cu modurile mixte: "rb+", "wb+" și "ab+"**
- **Mod mixt = prima operație permisă este cea indicată de litera din mod, iar a doua este complementara sa**

Exemple:

- **rb+** = citire (indicată de caracterul 'r') + scriere (operația complementară citirii)
- **wb+** sau **ab+** = scriere (indicată de caracterul 'w') + citire (operația complementară scrierii)

Exemplu: Citirea unui fișier text la nivel text, respectiv la nivel binar (nu mai există octeți "speciali", adică '\r'+'\n' nu se mai transformă automat într-un '\n').





```
#include <stdio.h>

int main()
{
    int c, cnt = 0;

    FILE *f = fopen("exemplu.txt", "rb");

    while((c = fgetc(f)) != EOF)
        printf("%3d. %d -> %c\n", ++cnt, c, c);

    fclose(f);

    return 0;
}
```

```
Ana are
mere, pere
si prune!!!
```

```
 1. 65 -> A
 2. 110 -> n
 3. 97 -> a
 4. 32 ->
 5. 97 -> a
 6. 114 -> r
 7. 101 -> e
 8. 13 ->
 9. 10 ->

10. 109 -> m
11. 101 -> e
12. 114 -> r
13. 101 -> e
14. 44 -> ,
15. 32 ->
16. 112 -> p
17. 101 -> e
18. 114 -> r
19. 101 -> e
20. 13 ->
21. 10 ->

22. 115 -> s
23. 105 -> i
24. 32 ->
25. 112 -> p
26. 114 -> r
27. 117 -> u
28. 110 -> n
29. 101 -> e
30. 33 -> !
31. 33 -> !
32. 33 -> !

Process returned 0 (0x0)   execution time : 0.032 s
Press any key to continue.
```

Închiderea unui fișier binar:

`int fclose(FILE *f)`

Operații de scriere/citire:

a) **`int fwrite(adresa de început a unei zone de memorie, dimensiunea unui bloc în octeți, numărul de blocuri de octeți, FILE* f)`** - funcția returnează numărul blocurilor scrise corect în fișier

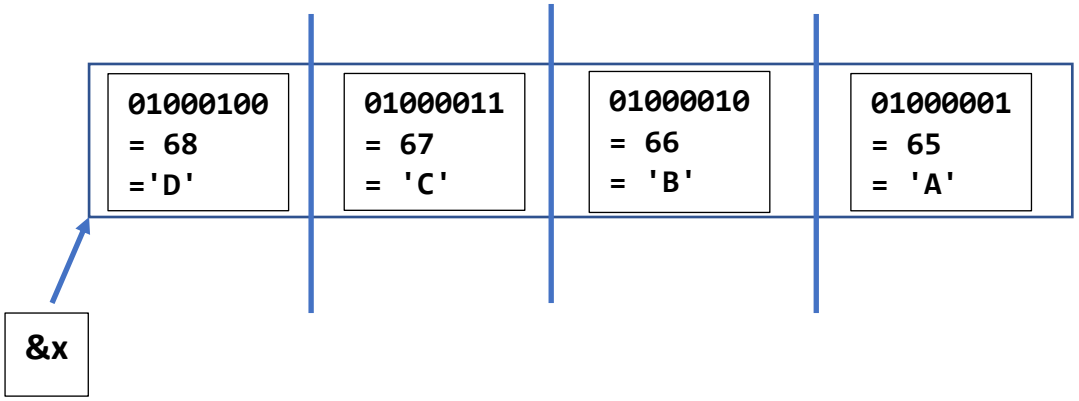
b) **`int fread(adresa de început a unei zone de memorie, dimensiunea unui bloc în octeți, numărul de blocuri de octeți, FILE* f)`** - funcția returnează numărul blocurilor citite corect din fișier

Zonă de memorie formată din 8 octeți							
1	2	3	4	5	6	7	8

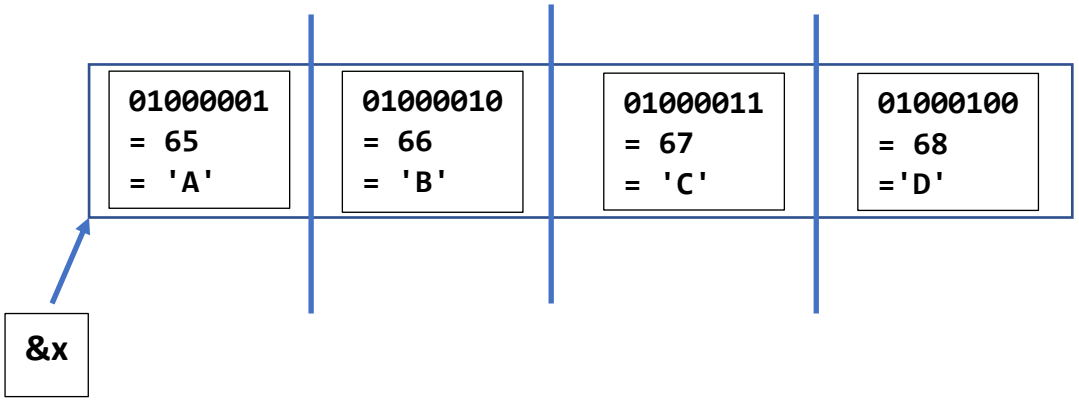
- Nr. blocuri = 8, dim. bloc (în octeți) = 1 <=> 8 valori de tip char
- Nr. blocuri = 2, dim. bloc (în octeți) = 4 <=> 2 valori de tip int / float
- Nr. blocuri = 4, dim. bloc (în octeți) = 2 <=> 4 valori de tip short int
- Nr. blocuri = 1, dim. bloc (în octeți) = 8 <=> 1 valoare de tip double / long long int

Exemplu:

```
int x = 1145258561 -> sizeof(int) = 4 octeți
```



Little endian:




```
x = 1145258561
x = 01000100 | 01000011 | 01000010 | 01000001 -> baza 2
x =       44 |       43 |       42 |       41 -> baza 16
x =       68 |       67 |       66 |       65 -> baza 10
x =      'D' |      'C' |      'B' |      'A' -> ASCII
x =      'A' |      'B' |      'C' |      'D' -> little endian
```

//zona de memorie care începe la adresa &x conține 1 bloc de 4 octeți
fwrite(&x, sizeof(int), 1, f);

//zona de memorie care începe la adresa &x conține 4 blocuri de 1 octet
fwrite(&x, 1, sizeof(int), f);

Exemplu: Scrierea/citirea unui număr întreg la nivel binar

```
#include<stdio.h>

int main()
{
    int x = 1145258561, y, r;

    printf("x = %d\n", x);


    FILE* f = fopen("numar.bin", "wb");
    r = fwrite(&x, sizeof(int), 1, f);
    printf("r = %d\n\n", r);
    fclose(f);

    f = fopen("numar.bin", "rb");
    r = fread(&y, 1, sizeof(int), f);
    printf("r = %d\n", r);
    fclose(f);

    printf("y = %d\n", y);

    return 0;
}
```

Endianess: ordinea în care sunt stocați octeții unei variabile în memoria internă!

Big endian: octetul cel mai puțin semnificativ este memorat ultimul de la stânga la dreapta (adresa cea mai mare)

2134 => cifra cea mai semnificativă (2) este prima

Big endian se utilizează în protocoale de rețea!

213445+
000265

213710

Little endian: octetul cel mai puțin semnificativ este memorat primul de la stânga la dreapta (adresa cea mai mică)

2134 => 4312 în format little endian

Little endian se utilizează în procesoarele de tip Intel!

544312+
562000

017312

Fișierele binare permit, în afara accesului secvențial la octeții săi, și accesul aleatoriu (poziționarea directă pe un anumit octet din fișier)!

Funcții de poziționare:

a) `long ftell(FILE* f)` **–** furnizează numărul de ordine al octetului curent din fișier (cuprins între 0 și 2^{31}-1 = 2147483647) => în limbajul C standard NU pot fi manipulate direct fișiere mai mari de 2GB!!!

```
#include <stdio.h>

int main()
{
    int c;

    FILE *f = fopen("exemplu.txt", "rb");

    while((c = fgetc(f)) != EOF)
        printf("%3ld. -> %c\n", ftell(f), c);

    fclose(f);

    return 0;
}
```

b) `int fseek(FILE* f, int nr_octeți, int origine)` – poziționează pointerul de fișier pe octetul peste numărul de octeți indicat (`nr_octet`) raportându-se la originea respectivă, specificată printr-una din constantele următoare:

- **SEEK_SET** = începutul fișierului
- **SEEK_END** = sfârșitul fișierului
- **SEEK_CUR** = octetul curent din fișier (pointerul de fișier)

Exemplu: funcție care calculează dimensiunea în octeți a unui fișier

```
//Fisierul f trebuie sa fie deschis!!!
int dim_fisier(FILE *f)
{
    //pcrt = pozitia curenta in fisier
    //dimf = dimensiunea fisierului in octeti
    int dimf, pcrt;

    pcrt = ftell(f);

    fseek(f, 0, SEEK_END);
    dimf = ftell(f);

    fseek(f, pcrt, SEEK_SET);

    return dimf;
}
```

Observație: NU se recomandă utilizarea funcțiilor de poziționare în fișiere text!!!

Exemplu: scrierea unui tablou unidimensional într-un fișier binar

```
#include <stdio.h>

int main()
{
    int i;
    int v[] = {1, 2, 3 , 4, 5, 6, 7, 8, 9, 10};
    int n = sizeof(v) / sizeof(int);

    FILE *f = fopen("tablou.bin", "wb");

    //Scrierea tabloului
    //Varianta 1 - element cu element
    for(i = 0; i < n; i++)
        fwrite(&v[i], sizeof(int), 1, f);
        //fwrite(v+i, sizeof(int), 1, f);

//    //Varianta 2 - tot tabloul
//    //privit ca n blocuri a cate sizeof(int) octeti fiecare
//    fwrite(v, sizeof(int), n, f);
//
//    //Varianta 3 - tot tabloul
//    //privit ca 1 bloc a n*sizeof(int) octeti
//    fwrite(v, n*sizeof(int), 1, f);
```

```
    //Citirea tabloului
    //redeschid fisierul in alt mod (citire)
    freopen("tablou.bin", "rb", f);

    //calculez numarul de numere intregi din fisier
    fseek(f, 0, SEEK_END);
    n = ftell(f) / sizeof(int);
    fseek(f, 0, SEEK_SET);

    //citirea tabloului
    fread(v, sizeof(int), n, f);

    //afisarea tabloului
    printf("Tabloul:\n");
    for(i = 0; i < n; i++)
        printf("%d ", v[i]);

    fclose(f);

    return 0;
}
```

Observație: Pentru a simplifica citirea elementelor tabloului din fișier, se poate scrie, la începutul fișierului, numărul n de elemente din tablou!

Observație: Dacă valorile scrise într-un fișier binar NU sunt toate de același tip, atunci este obligatoriu să scriem în fișier și dimensiunile lor în octeți!