



# Laurea triennale in Informatica

*modulo (CFU 6) di*

## Programmazione II e Lab.

# prof. Mariarosaria Rizzardi

Centro Direzionale di Napoli – Isola C4

stanza: n. 423 – IV piano Lato Nord

tel.: 081 547 6545

email: mariarosaria.rizzardi@unipARTHENOPE.it

## Unità didattica: gestione di file sequenziali in C

[1-AC]

**Titolo:** File sequenziali (file testo e file binari) in C

Argomenti trattati:

- ✓ Dichiarazione di file in C e file-pointer
- ✓ Esempi d'uso di un file testo e di un file binario
- ✓ Funzioni C per l'input/output su file formattato e non formattato
- ✓ Come si determina la fine del file
- ✓ Lettura da file mediante "buffer"

Prerequisiti richiesti: fondamenti della programmazione C

# File in C

**Attenzione !!!** al nome (completo di path) di un file in C

*Esempio Windows:*

c:\dati\elenco.txt

nel programma C: `char *nomefile = "c:\\dati\\elenco.txt";`  
input da tastiera: c :\dati\elenco.txt

**binario (modo b)**

**testo (default)**

**Tipo di file**

Un **file testo** è un insieme di righe di caratteri: ogni riga contiene 0 o più caratteri (al massimo 255) e termina con un carattere speciale (es.: \n).

Un file, prima di essere usato, va **aperto** specificando il modo in cui verrà usato:

in C

**fp=fopen(nome\_file, modo)**

restituisce

stringa

"r" read  
"w" write  
"a" append

"r+" r & w  
"w+" r & w  
"a+" r & a

**FILE \*fp** (file pointer)

È un puntatore ad una struttura contenente informazioni sul file. Serve per indirizzare tutte le operazioni sul file.

Un file, dopo averlo usato, va **chiuso**:

in C

**fclose(fp)**

# Esempio: scrittura di un file di caratteri

```

/* file di caratteri: copia i caratteri del testo
   nel file specificato in ingresso */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int main()
{char *testo = "Nel mezzo del cammin di nostra vita\n"
             ... ... ";
char *cp, nomefile[13];
FILE *fp; dichiara fp puntatore a file
printf("nome del file (max 8 chars)=");
fgets(nomefile,9,stdin); nomefile[strlen(nomefile)-1]='\0';
strcat(nomefile,".txt");
printf("nomefile=%s\n", nomefile);
fp = fopen(nomefile, "w"); apre il file in scrittura associando fp al file
cp = testo;
while(*cp != '\0') if (fp == NULL)
    exit(1);
{
    putc(*cp,fp); scrive un carattere sul file puntato da fp
    cp++;
}
fclose(fp); chiude il file
return 0;
}

```

# File binario

in C

```
FILE *fp; char nome_file[20];  
...  
fp=fopen(nome_file, "rb")
```

Apre in lettura (**r**) il file binario (**b**)

## Input/Output su file binario in C

Lettura/scrittura di un intero  
blocco di dati in formato  
interno (così com'è in memoria)

Input	Output
<b>fread(...)</b>	<b>fwrite(...)</b>

# I/O su file binario in C

## Input

```
type_i fread(void *buffer, type_i size, type_i count, FILE *fp)
```

## Output

```
type_i fwrite(void *buffer, type_i size, type_i count, FILE *fp)
```

Sono trasferite fino a **count** voci, ciascuna di **size** bytes, tra l'unità specificata da **fp** e l'area di memoria puntata da **buffer**. Il puntatore sul file avanza del numero di byte relativo (max **count** × **size**).

Le due funzioni restituiscono il numero di voci completamente trasferite.

```
#include <stdio.h>
...
{int j,vet1[LUNG],vet2[LUNG]; FILE *fp;
...
if (fwrite(vet1,sizeof(int),LUNG,fp)!=LUNG)
...
if (fread(vet2,sizeof(int),LUNG,fp)!=LUNG)
...
```

## Esempio

# Esempio: uso di file binario (lettura e scrittura)

```
#include <stdlib.h>
#include <stdio.h>
#define LUNG 20
void main()
{int j,vet1[LUNG],vet2[LUNG]; FILE *fp;
 for (j=0; j<LUNG; j++) vet1[j]=2*j;
 if ((fp=fopen("fbinario.dat","wb"))==NULL)
 {puts("Errore apertura file"); exit(1);}
 if(fwrite(vet1,sizeof(int),LUNG,fp)!=LUNG)
 {puts("Errore scrittura file"); exit(1);}
 fclose(fp);
 if ((fp=fopen("fbinario.dat","rb"))==NULL)
 {puts("Errore apertura file"); exit(1);}
 if (fread(vet2,sizeof(int),LUNG,fp)!=LUNG)
 {puts("Errore lettura file"); exit(1);}
 for (j=0; j<LUNG; j++)
 printf("%2d\t%2d\n",vet1[j],vet2[j]);
 fclose(fp);
}
```

lunghezza in byte  
singolo elemento

numero elementi  
nel blocco

uscita per errore  
del programma

puntatore

# Lettura/Scrittura su file binario in C

uso di **fseek()** in **<stdio.h>**

## posizionamento all'interno di un file

```
int fseek(FILE *fp, long offset, int origin)
```

file  
pointer

spostarsi  
num byte  
da origin

posizione  
iniziale

restituisce 0 in caso di successo

origin:	SEEK_CUR	Current position of file pointer
	SEEK_END	End of file
	SEEK_SET	Beginning of file

# Esempio

Ricerca il valore di key all'interno del file e se lo trova lo sostituisce

Apre il file in lettura/scrittura

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #define LUNG 20
4 void main()
5 {    int j, dato, key, new_dato;
6     FILE *fp;
7
8     // "r+": Apre il file per leggere / scrivere
9     if ((fp=fopen("fbinario.dat","r+b")) == NULL)
10    {    puts("\nErrore apertura file"); exit(1); }
11
12    // Legge il file binario: 1 dato alla volta
13    puts("Legge il file binario 1 record alla volta");
14    for (j=0;j<LUNG: j++)
15    {    if (fread(&dato,sizeof(int),1,fp) != 1)
16        // attenzione: &vet_comp e non vet_comp !!!
17        {    puts("\nErrore lettura file"); exit(1); }
18        printf("\ndato(%2d) =%5d", j+1,dato);
19    }
20    puts("\n");
21 }
```

visualizza il file

## posiziona fp all'inizio del file

# Esempio [cont]

```

22     printf("Valore da cercare: "); scanf("%d", &key);
23     printf("Sostituirlo con : "); scanf("%d", &new_dato);
24
25     if ((fseek(fp, 0, SEEK_SET)) != 0)
26     {
27         puts("\nErrore posizionamento file"); exit(1);
28     }
29     else
30         puts("\nPosizionato all'inizio del file\n");
31
32     // Legge il file binario: 1 dato alla volta per cercare key
33     puts("\nLegge il file binario 1 record alla volta");
34     for (j=0; j<LUNG; j++)
35     {
36         if (fread(&dato, sizeof(int), 1, fp) != 1)
37             // attenzione: &dato e non dato !!!
38             { puts("\nErrore lettura file"); exit(1); }
39         if (dato == key)
40             {
41                 printf("\nTrovato key: in dato(%2d) =%5d      =====", j+1, dato)
42
43                 // torna indietro di un dato sul file
44                 if ((fseek(fp, -(int)sizeof(dato), SEEK_CUR)) != 0)
45                 {
46                     puts("\nErrore posizionamento file"); exit(1);
47                 }
48                 else
49                     {
50                         if (fwrite(&new_dato, sizeof(int), 1, fp) != 1)
51                             { puts("\nErrore scrittura file"); exit(1); }
52                         puts("\nDato sostituito!");
53                         break;
54                     }
55                 else
56                     printf("\nLegge dato(%2d) =%5d", j+1, dato);
57             }
58         puts("\n");

```

**ricerca sequenziale**

**sostituisce**

**torna indietro di un record per sovrascrivere**

# Esempio [cont]

posiziona fp all'inizio del file

matematica  
ario di ...  
rdin

```
52     puts ("\n");
53
54     if (fseek(fp, 0, SEEK_SET) != 0)
55     {   puts ("\nErrore posizionamento file"); exit(1); }
56     else
57         puts ("\nPosizionato all'inizio del file\n");
58
59 // Legge il file binario: 1 dato alla volta
60 puts("Legge il file binario 1 record alla volta");
61 for (j=0; j<LUNG; j++)
62 {   if (fread(&dato, sizeof(int), 1, fp) != 1)
63     {   puts ("\nErrore lettura file"); exit(1); }
64     printf ("\ndato(%2d) =%5d", j+1, dato);
65   }
66   puts ("\n");
67
68 fclose(fp);
69 }
```

visualizza il file

# RUN esempio

Legge il file binario 1 record alla volta

```
dato( 1) = 0  
dato( 2) = 2  
dato( 3) = 4  
dato( 4) = 6  
dato( 5) = 8  
dato( 6) = 10  
dato( 7) = 12  
dato( 8) = 14  
dato( 9) = 16  
dato(10) = 18  
dato(11) = 20  
dato(12) = 22  
dato(13) = 24  
dato(14) = 26  
dato(15) = 28  
dato(16) = 30  
dato(17) = 32  
dato(18) = 34  
dato(19) = 36  
dato(20) = 38
```

Valore da cercare: 10

Sostituirlo con : 999

Posizionato all'inizio del file

Legge il file binario 1 record alla volta

```
legge dato( 1) = 0  
legge dato( 2) = 2  
legge dato( 3) = 4  
legge dato( 4) = 6  
legge dato( 5) = 8
```

**trovato key: in dato( 6) = 10 =====**

dato sostituito!

Posizionato all'inizio del file

Legge il file binario 1 record alla volta

```
dato( 1) = 0  
dato( 2) = 2  
dato( 3) = 4  
dato( 4) = 6  
dato( 5) = 8  
dato( 6) = 999  
dato( 7) = 12  
dato( 8) = 14
```

...

# Altre funzioni di stream I/O

void **rewind(FILE \*fp)** riposiziona fp all'inizio del file

long **ftell(FILE \*fp)** restituisce la posizione corrente (byte)

```
1 #include <stdio.h>
2
3 int main()
4 {
5     FILE *fp; long pos; int vet[10];
6
7     // fbinario.dat contiene 20 valori di tipo int
8     if( (fp = fopen("fbinario.dat", "rb")) != NULL )
9     {
10         // legge 10 valori
11         fread(vet, sizeof(vet[0]), 10, fp);
12
13         pos = ftell(fp);
14
15         printf("\nposizione dopo fread(): %ld\n", pos);
16         fclose(fp);
17     }
18     return 0;
19 }
```

posizione dopo fread(): 40 ← byte già letti

# I/O su file testo in C

Input		Output	
non formattato	formattato	non formattato	formattato
getc(...)	fscanf(...)	putc(...)	fprintf(...)
gets(...)		puts(...)	
fgets(...)		fputs(...)	

**fscanf(FILE \*, "formato", variabili)**



**fprintf(FILE \*, "formato", variabili)**

# I/O singolo carattere (come int)

```
int getc(FILE * )
int getchar(void)
int putc(intero, FILE *)
int putchar(intero)
```

intero: qualsiasi tipo intero, ma solo gli 8 bit meno significativi saranno trasferiti

Unità standard di I/O  
**(stdin, stdout)**

numero caratteri da leggere

# I/O stringa di caratteri

```
char *gets(char *)
char *fgets(char *, int, FILE *)
int puts(char *)
int fputs(char *, FILE *)
```

deprecata **gets(char \*p\_str)**

legge i caratteri da **stdin** finchè non incontra  
**\n** oppure la fine della riga.

Il carattere **\n** è poi sostituito da **NULL (\0)** e la  
stringa è memorizzata nella locazione puntata da  
**p\_str**.

Invece di **gets(...)**, per leggere da un file  
**fgets(char \*p\_str, int n, FILE \*fp)**

Legge anche il carattere **\n** che va sostituito da  
**NULL (\0)**.

# Esempio

```
/* legge i caratteri dal file specificato in ingresso  
e li visualizza sullo schermo */  
#include <stdlib.h>  
#include <stdio.h>  
#include <string.h>  
int main()  
{  
    char ch, *cp, nomefile[13]; /* 13=8+4+'\0' */  
    FILE *fp;  
    printf("nome del file (max 8 chars)");  
    fgets(nomefile,9,stdin); nomefile[strlen(nomefile)-1]='\0';  
    strcat(nomefile,".txt");  
    fp=fopen(nomefile,"r");  
    do {  
        ch=getc(fp); putchar(ch);  
    } while(ch != EOF);  
    fclose(fp);  
    return 0;  
}
```

**EOF = end of file**

La ***fine di un file testo*** può essere individuata:

- confrontando il carattere letto con **EOF**;
- oppure
- tramite la function **C**

**int feof(FILE \*)**

che restituisce **0 (falso)** se non è stata raggiunta la fine del file; un valore **≠0 (vero)** altrimenti.

La ***fine di un file binario*** può essere individuata **solo** tramite la function **C**

**int feof(FILE \*)**

# Esempio

```
/* lettura "bufferizzata" di file  
   1 stringa-buffer alla volta */  
  
#include <stdlib.h>  
#include <stdio.h>  
#define BUFSIZE 30  
  
void main()  
{char buffer[BUFSIZE], nomefile[60]; FILE *fp;  
 puts("nome del file testo"); fgets(nomefile, ...  
 if ((fp=fopen(nomefile,"r"))==NULL)  
 { puts("Errore apertura file"); exit(1);  
 }  
 while (!feof(fp))  
 {fgets(buffer,BUFSIZE,fp);  
 printf("%d\t%s\n",feof(fp),buffer);  
 }  
 fclose(fp);  
}
```

# File di lettura (senza \n)

Nel mezzo del cammin di nostra vita mi ritrovai per una selva oscura che la diritta via era smarrita. Ah quanto a dir qual era e` cosa dura esta selva selvaggia e aspra e forte che nel pensier rinnova la paura!

0  
0  
0  
0  
0  
0  
0  
16

Nel mezzo del cammin di nostra vita mi ritrovai per una selva oscura che la diritta via era smarrita. Ah quanto a dir qual era e` cosa dura esta selva selvaggia e aspra e forte che nel pensier rinnova la paura!



**feof(fp)**

← →

**BUFSIZE-1**



# Esempio

```
/*lettura "bufferizzata" da file:  
    1 carattere alla volta */  
  
#include <stdlib.h>  
#include <stdio.h>  
#define BUFSIZE 30  
  
void main()  
{int j;char ch,buffer[BUFSIZE],nomefile[60];FILE *fp;  
puts("nome del file testo"); fgets(nomefile, ...  
if ((fp=fopen(nomefile,"r"))==NULL)  
{puts("Errore apertura file"); exit(1);}  
while (!feof(fp))  
{j=-1;  
 while (j<BUFSIZE-1 && !feof(fp))  
 {ch=(char)getc(fp);  
 j++; buffer[j]=ch;  
 putchar(buffer[j]);  
 } printf("\t\t%d\n",feof(fp));  
}  
}
```



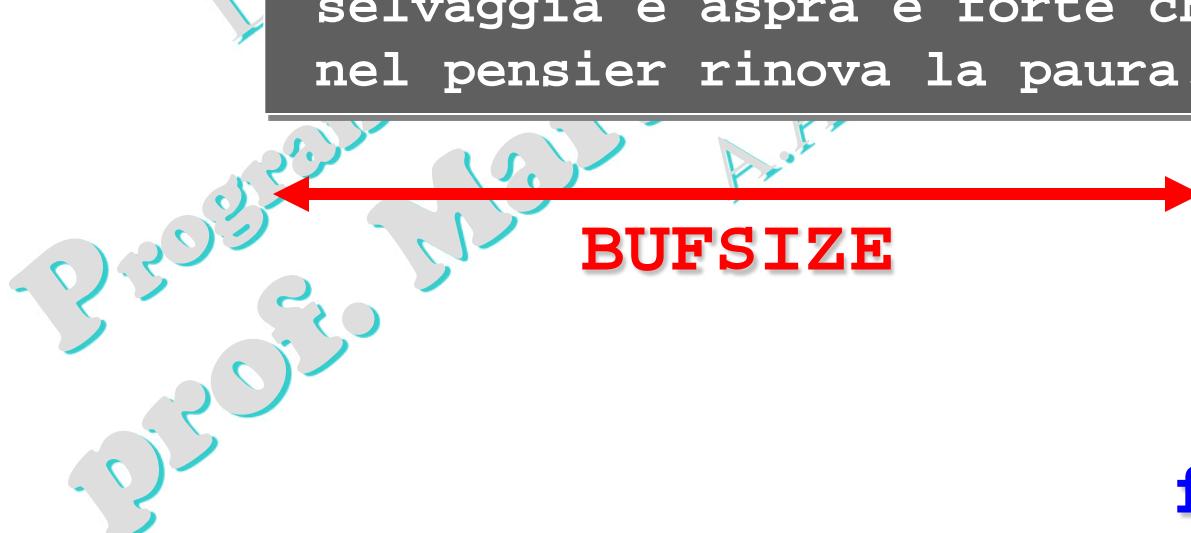
# File di lettura (senza \n)

Nel mezzo del cammin di nostra vita mi ritrovai per una selva oscura che la diritta via era smarrita. Ah quanto a dir qual era e` cosa dura esta selva selvaggia e aspra e forte che nel pensier rinnova la paura!

Nel mezzo del cammin di nostra vita mi ritrovai per una selv a oscura che la diritta via er a smarrita. Ah quanto a dir qu al era e` cosa dura esta selva selvaggia e aspra e forte che nel pensier rinnova la paura!	0 0 0 0 0 0 0 16
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------

**BUFSIZE**

**feof(fp)**



# Lettura da file binario: errore

Il file "fbinario.dat" contiene 20 interi: 0, 2, 4, ..., 38.

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

int main()
{
    FILE *fp;
    int j, vet_comp;

    /* APRE IL FILE BINARIO */
    if ((fp=fopen("fbinario.dat","rb")) == NULL)
    {
        puts("Errore apertura file"); exit(1);
    }

    /* LEGGE 1 RECORD ALLA VOLTA DAL FILE */
    j=-1;
    while ( !feof(fp) )
    {
        if (fread(&vet_comp,sizeof(int),1,fp) != 1)
        {
            puts("Errore lettura file"); exit(1);
        }
        j++;
        printf("vet2[%2d] = %3d\n", j,vet_comp);
    }
    fclose(fp);
    return 0;
}
```

vet2[ 0 ] = 0
vet2[ 1 ] = 2
vet2[ 2 ] = 4
vet2[ 3 ] = 6
vet2[ 4 ] = 8
vet2[ 5 ] = 10
vet2[ 6 ] = 12
vet2[ 7 ] = 14
vet2[ 8 ] = 16
vet2[ 9 ] = 18
vet2[10] = 20
vet2[11] = 22
vet2[12] = 24
vet2[13] = 26
vet2[14] = 28
vet2[15] = 30
vet2[16] = 32
vet2[17] = 34
vet2[18] = 36
vet2[19] = 38

???

Errore lettura file

La funzione `feof(fp)` restituisce "VERO" solo se il file è posizionato oltre la sua fine.

# Lettura da file binario: soluzione 1

Il file "fbinario.dat" contiene 20 interi: 0, 2, 4, ..., 38

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

int main()
{
    FILE *fp;
    int j, vet_comp;

    /* APRE IL FILE BINARIO */
    if ((fp=fopen("fbinario.dat","rb")) == NULL)
    {
        puts("Errore apertura file"); exit(1);
    }

    /* LEGGE 1 RECORD ALLA VOLTA DAL FILE */
    j=-1;
    while ( fread(&vet_comp,sizeof(int),1,fp) == 1 )
    {
        j++;
        printf("vet2[%2d] = %3d\n", j,vet_comp);
    }
    if (!feof(fp))
    {
        puts("Errore lettura file"); exit(1);
    }
    else
        puts("Fine del file");
    fclose(fp);
    return 0;
}
```

vet2[ 0 ] = 0
vet2[ 1 ] = 2
vet2[ 2 ] = 4
vet2[ 3 ] = 6
vet2[ 4 ] = 8
vet2[ 5 ] = 10
vet2[ 6 ] = 12
vet2[ 7 ] = 14
vet2[ 8 ] = 16
vet2[ 9 ] = 18
vet2[10] = 20
vet2[11] = 22
vet2[12] = 24
vet2[13] = 26
vet2[14] = 28
vet2[15] = 30
vet2[16] = 32
vet2[17] = 34
vet2[18] = 36
vet2[19] = 38

!!!

Fine del file

# Lettura da file binario: soluzione 2

Il file "fbinario.dat" contiene 20 interi: 0, 2, 4, ..., 38

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

int main()
{
    FILE *fp;
    int j, vet_comp;

    /* APRE IL FILE BINARIO */
    if ((fp=fopen("fbinario.dat","rb")) == NULL)
    {
        puts("Errore apertura file"); exit(1);
    }

    /* LEGGE 1 RECORD ALLA VOLTA DAL FILE */
    j=-1;

    do {
        if ( fread(&vet_comp,sizeof(int),1,fp) == 1 )
        {
            j++;
            printf("vet2[%2d] = %3d\n", j,vet_comp);
        }
    } while ( !feof(fp) )

    fclose(fp);
    return 0;
}
```

vet2[ 0] =	0
vet2[ 1] =	2
vet2[ 2] =	4
vet2[ 3] =	6
vet2[ 4] =	8
vet2[ 5] =	10
vet2[ 6] =	12
vet2[ 7] =	14
vet2[ 8] =	16
vet2[ 9] =	18
vet2[10] =	20
vet2[11] =	22
vet2[12] =	24
vet2[13] =	26
vet2[14] =	28
vet2[15] =	30
vet2[16] =	32
vet2[17] =	34
vet2[18] =	36
vet2[19] =	38

Fine del file

!!!

# Esercizi

1

Scrivere *function C* che legga, mediante buffer di 200char, un file testo e lo visualizzi sullo schermo 40char/riga e 25righe/schermata.

2

Scrivere *function C* che crei un file binario “**studente.dat**” contenente le seguenti informazioni:

- cognome e nome (30c) c = char
- matricola (ccc/cccccc)
- numero degli esami superati (short)
- media pesata\* degli esami (float)
- crediti acquisiti (short).

Il file contiene le informazioni **ordinate per matricola**.

Scrivere inoltre *function C* che, **aggiorni studente.dat** a partire da un file relativo ad un certo esame (per es. **esameProg2.dat**) contenente gli studenti che hanno superato l'esame in una certa data ed i relativi voti.

**Help!**

# Come si calcola la Media pesata degli esami?

Esame<sub>1</sub>, cfu<sub>1</sub>, voto<sub>1</sub>

Esame<sub>2</sub>, cfu<sub>2</sub>, voto<sub>2</sub>

...

Esame<sub>k</sub>, cfu<sub>k</sub>, voto<sub>k</sub>

...

Esame<sub>N</sub>, cfu<sub>N</sub>, voto<sub>N</sub>

## voto pesato

$$\text{voto}_k \times \frac{\text{cfu}_k}{\sum_{k=1}^N \text{cfu}_k}$$

↑  
peso

$$\text{Media}_p = \frac{\text{cfu}_1 * \text{voto}_1 + \text{cfu}_2 * \text{voto}_2 + \text{cfu}_3 * \text{voto}_3 + \dots + \text{cfu}_N * \text{voto}_N}{\text{cfu}_1 + \text{cfu}_2 + \text{cfu}_3 + \dots + \text{cfu}_N}$$

CFU totali

# Come si aggiorna la Media pesata?

$$\text{OLD\_Media}_p = \frac{cfu_1 * voto_1 + cfu_2 * voto_2 + cfu_3 * voto_3 + \dots + cfu_N * voto_N}{\text{OLD\_CFU\_tot}}$$

Esame<sub>N+1</sub>, cfu<sub>N+1</sub>, voto<sub>N+1</sub>



$$\text{NEW\_Media}_p = \frac{\text{OLD\_Media}_p * \text{OLD\_CFU\_tot} + cfu_{N+1} * voto_{N+1}}{\underbrace{\text{OLD\_CFU\_tot} + cfu_{N+1}}_{\text{NEW\_CFU\_tot}}}$$