Modulo: Approfondimenti sul linguaggio C: file [07]

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 file 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*:** **c:\dati\elenco.txt**

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

Tipo di file
- **binario** (modo **b**)
- **testo** (default)

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)`

- nome_file: stringa
- modo:

"r"	read	"r+"	r & w
"w"	write	"w+"	r & w
"a"	append	"a+"	r & a

restituisce

`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 <stdio.h>
#include <string.h>
void main()
{char *testo = "Nel mezzo del cammin di nostra vita\n"
                                                 … … ";
 int i;char *cp, nomefile[13];
 FILE *fp;
 printf("nome del file (max 8 chars)=");
 scanf("%s",&nomefile); strcat(nomefile,".txt");
 printf("nomefile=%s\n",nomefile);
 fp = fopen(nomefile,"w");
 cp = testo;
 while(*cp != '\0')
      {putc(*cp,fp); cp++;
      }
 fclose(fp);}
```

dichiara **fp** puntatore a file

apre il file associando **fp** al file

scrive un carattere sul file puntato da **fp**

chiude il file

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
`fread(…)`	`fwrite(…)`

I/O su file binario in C

Input

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

Output

tipe_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.

Esempio

```
#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: *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

puntatore

uscita per errore del programma

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)
```

- `fp`: file pointer
- `offset`: spostarsi num byte da origin
- `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

Esempio [cont]

posiziona fp all'inizio del file

ricerca sequenziale

sostituisce

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

torna indietro di un record per sovrascrivere

Esempio [cont]

posiziona fp all'inizio del file

visualizza il file

```
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  }
70
```

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)

```c
#include <stdio.h>

int main()
{
    FILE *fp; long pos; int vet[10];

    // fbinario.dat contiene 20 valori di tipo int
    if( (fp = fopen("fbinario.dat", "rb")) != NULL )
    {
        // legge 10 valori
        fread(vet, sizeof(vet[0]), 10, fp);

        pos = ftell(fp);

        printf("\nposizione dopo fread(): %ld\n", pos);
        fclose(fp);
    }
return 0;
}
```

```
posizione dopo fread(): 40
```

byte già letti

I/O su file testo in C

Input		Output	
non formattato	formattato	non formattato	formattato

`getc(…)`
`gets(…)`
`fgets(…)`
	`fscanf(…)`	`putc(…)`
		`puts(…)`
		`fputs(…)`
			`fprintf(…)`

```
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`**)

I/O stringa di caratteri

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

numero caratteri da leggere

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**.

Analogamente a **gets(…)**, per leggere da un file

fgets(char *p_str, int n, FILE *fp)

Esempio

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

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"); gets(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 rinova la paura!
```

```
0       Nel mezzo del cammin di nostr
0       a vita mi ritrovai per una se
0       lva oscura che la diritta via
0        era smarrita. Ah quanto a di
0       r qual era e` cosa dura esta
0       selva selvaggia e aspra e for
0       te che nel pensier rinova la
16      paura!
```

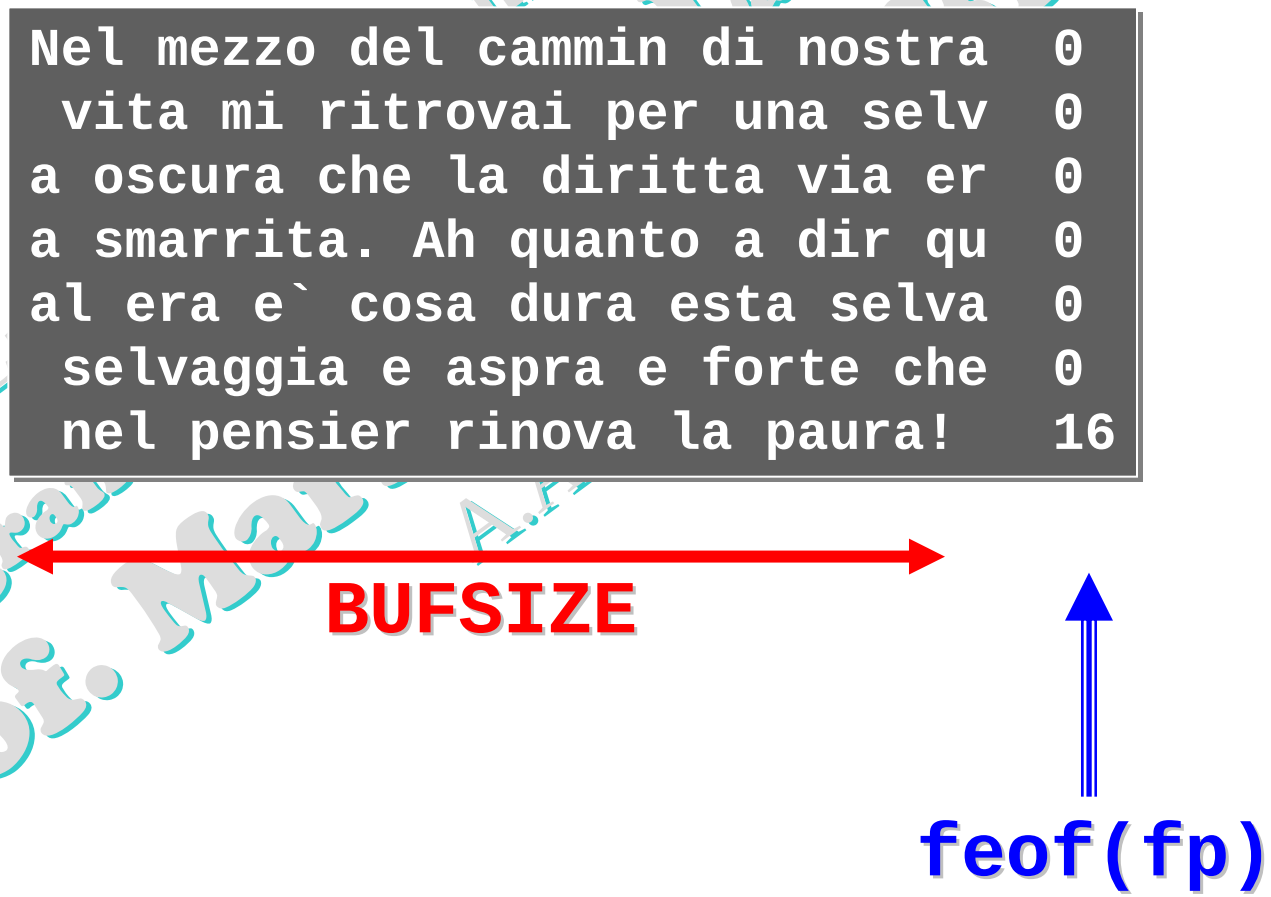
feof(fp)

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"); gets(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 rinova la paura!
```



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/ccccccc)
- numero degli esami superati (short)
- media pesata* degli esami (float)
- crediti acquisiti (short).

Help!

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. [liv.2]

Come si calcola la Media pesata degli esami?

$Esame_1, cfu_1, voto_1$
$Esame_2, cfu_2, voto_2$
...
$Esame_k, cfu_k, voto_k$
...
$Esame_N, cfu_N, voto_N$

voto pesato

$$voto_k \times \frac{cfu_k}{\sum_{k=1}^{N} cfu_k}$$

peso

$$Media_p = \frac{cfu_1 * voto_1 + cfu_2 * voto_2 + cfu_3 * voto_3 + \ldots + cfu_N * voto_N}{cfu_1 + cfu_2 + cfu_3 + \ldots + cfu_N}$$

CFU totali

Come si aggiorna la Media pesata?

$$OLD\_Media_p = \frac{cfu_1 * voto_1 + cfu_2 * voto_2 + cfu_3 * voto_3 + \ldots + cfu_N * voto_N}{OLD\_CFU\_tot}$$

$Esame_{N+1}$, cfu_{N+1}, $voto_{N+1}$ →

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