File e puntatori a file

Vitoantonio Bevilacqua
vitoantonio.bevilacqua@poliba.it

La libreria “stdio.h” mette a disposizione delle funzioni che hanno la peculiarità di trattare con questa nuova tipologia di dato, ovvero dati di tipo puntatore a FILE:
Per fare ciò è necessario individuare dove risiede il file nella memoria di massa (hard disk, ecc.) utilizzando un “puntatore a file” la cui dichiarazione è:

```
FILE *fp; //fp = File Pointer = Puntatore a file
```

Successivamente è possibile aprire un determinato file mediante una funzione che ha la seguente sintassi:

```
fp = fopen("C:\\...\\Nome_file.txt", "Modo_apertura");
```

Ricordando che nel primo campo bisogna indicare il percorso completo del file con doppi “backslash” (‘\\’), e come modalità di apertura una delle seguenti:

Modo	Significato **NB esistono altre modalità che saranno presentate in seguito**
“r”	“read” : apre un file di testo in modalità di sola lettura; se si verifica un errore (es. il file non esiste) la funzione fopen ritorna il codice di errore NULL
“w”	“write” : apre un file di testo in modalità di sola scrittura. Se il file esiste, verrà sovrascritto, altrimenti verrà creato automaticamente.

Prima di eseguire una qualsiasi operazione conviene verificare che il file esista, è possibile quindi inserire nel codice una condizione di robustezza:

```
if(fp==NULL)
      printf( “File inesistente” );
```

In seguito è possibile importare il flusso di dati (stream) presente sul file mediante la funzione fscanf:

```
fscanf (fp, “%specificatore_di_tipo” , argomenti);
```

Il cui compito è di “leggere” ciò che è presente nel file (indicato dal puntatore fp) e copiarlo nella variabile indicata (argomenti); in questo modo scriviamo temporaneamente nella memoria RAM le informazioni dal file, salvato nella memoria di massa.
Un’altra funzione è la fprintf il cui compito è di scrivere su file, la sua sintassi è:

```
fprintf (fp, “%specificatore_di_tipo” , argomenti);
```

Con questa riga di codice si scrive nel file, salvato nella memoria di massa, indicato da fp, le informazioni che risiedono temporaneamente nella memoria RAM (nella variabile argomenti) .
Quando il file diventa non più necessario, è possibile chiuderlo tramite il seguente comando:

```
fclose (fp);
```

dove fp è il puntatore al file da chiudere.

- **OSSERVAZIONE :**

Esistono quindi migliori versioni per l’acquisizione e la stampa di informazioni: “fscanf” ed “fprintf”; queste funzioni sono una versione generale delle funzioni “scanf” e “printf”. Infatti le prime due funzioni (“fscanf” ed “fprintf”) svolgono lo stesso lavoro delle ultime due semplicemente utilizzando come primo argomento ‘stdin’ e ‘stdout’, i quali virtualizzano il concetto di tastiera e schermo.

ESEMPIO 1: *utilizzo delle funzioni fscanf e fprintf*

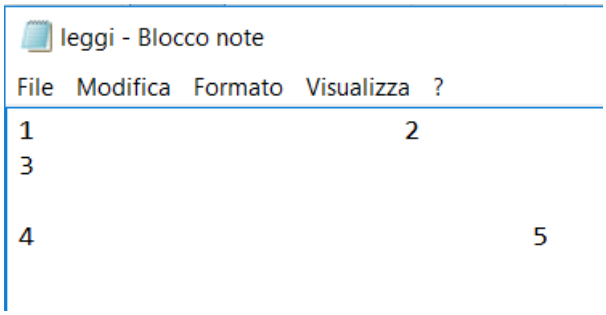
```c
#include <stdio.h>
int main()
{
    char nome[20];
    char cognome[20];
    fprintf(stdout,"Inserisci nome e cognome: "); /* Utilizzo della fprintf indicando come file
                                                     "stdout" così trasferire dati effettuando
                                                     un'operazione di scrittura su schermo */

    fscanf(stdin,"%s %s", nome, cognome);    /* Utilizzo della fscanf indicando il
                                                puntatore a file che virtualizza la tastiera */

    fprintf(stdout,"Cognome: %s\t\tNome: %s", cognome, nome); /* Stampa a schermo del
                                                                 cognome e nome inseriti */

    return 0;
}
```

ESEMPIO 2: *acquisizione e stampa su file*

```c
#include <stdio.h>

void leggi_file(FILE *,int *);
void scrivi_file(FILE *,int *);
void stampa(int *);

int main()
{
    FILE *fp;
    int V[5];
    fp=fopen("C:\\...\\leggi.txt","r"); //si presuppone l'esistenza di un file che
                                        //contenga 5 numeri disposti casualmente
    if(fp==NULL)    printf("il file non esiste!\n");
    else{
        leggi_file(fp,V);
        fclose(fp);
        stampa(V);
        fp=fopen("C:\\...\\scrivi.txt","w");
        scrivi_file(fp,V);
        fclose(fp);
    }
       return 0;
}

/* La funzione legge da file degli interi e li copia in un vettore */
void leggi_file(FILE *A, int *B)
{
    int i;
    for(i=0; i<5; i++)
        fscanf(A," %d" , &B[i]);

}

/* La funzione stampa a schermo i valori interi contenuti nel vettore, e se la funzione precedente è
   andata a buon fine, i valori saranno gli stessi contenuti nel file */
void stampa(int *A)
{
    int i;
    for(i=0; i<5; i++)
        fprintf(stdout,"%d\t\t", A[i]);
}
```

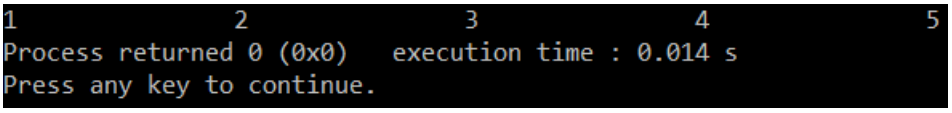
```c
/* La funzione stampa su file i numeri che sono stati copiati nel vettore, in modo "ordinato" */
void scrivi_file(FILE *A, int *B)
{
    int i;
    for(i = 0; i < 5; i++)
        fprintf(A, "%d\t\t" , B[i]);

}
```

- **INPUT :**

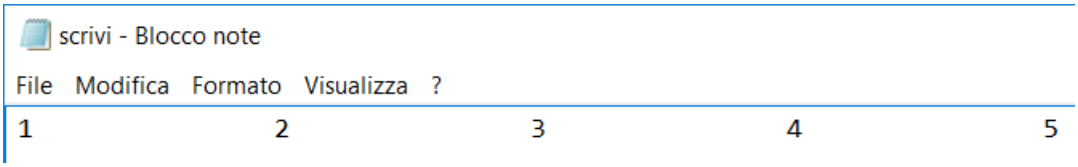


INPUT File – Screen del file "leggi.txt" (aperto con Blocco Note). Si noti come i valori sono inseriti in maniera quasi casuale, c'è solo l'accortezza di averli inseriti separati da spazio o dall'invio.

- **OUTPUT:**



OUTPUT Schermo – Screen della stampa a video del vettore che contiene i dati acquisiti dal file.



OUTPUT File – Screen del file "scrivi.txt" (aperto con Blocco Note). Si noti come i valori siano stati stampati "in ordine".

ESEMPIO 3: *acquisizione da file di informazioni riguardanti una classe di 10 studenti, e stampa su un file i bocciati in ordine crescente di voto, e su un altro file i promossi in ordine decrescente di voto.*

```c
#include <stdio.h>

struct studente //Dichiarazione della struttura di uno studente
{
	char cognome[20];
	char nome[20];
	int voto;
};

void leggi(FILE*, struct studente*);
void ordina(struct studente*);
void stampaPromossi(FILE*, struct studente*);
void stampaBocciati(FILE*, struct studente*);

int main()
{
	struct studente classe[10]; //La classe, di fatto risulta un vettore di studenti
	FILE* fp;

	fp = fopen( "C:\\...\\Classe.txt" , "r");
	if (fp == NULL) printf( "Il file è inesistente!" ); //Condizione di robustezza
	else
	{
		leggi(fp, classe);
		fclose(fp);

		ordina(classe);

	fp = fopen( "C:\\...\\Promossi.txt" , "w"); //Apre in modalità scrittura il file,
                                                //quindi se inesistente, lo crea
	stampaPromossi(fp, classe);
	fclose(fp);

	fp = fopen( "C:\\...\\Bocciati.txt" , "w"   //Apre in modalità scrittura il file,
                                               //quindi se inesistente, lo crea
	stampaBocciati(fp, classe);
	fclose(fp);
	}
	return 0;
}

/* La funzione acquisisce da file Cognome, Nome e Voto di ciascuno studente */
void leggi(FILE* A , struct studente* B)
{
	int i;
	for (i = 0; i < 10; i++)
	{
		fscanf(A, "%s %s %d", B[i].cognome, B[i].nome, &B[i].voto);
	}
}

/* Ordina tramite algoritmo di SELECTION SORT il vettore classe, in ordine crescente di voto */
void ordina(struct studente* A)
{
	struct studente B;
	int i, j;
	for (i = 0; i < 9; i++)
	{
```

```
			int imin = i;
			for (j = i + 1; j < 10; j++)
			{

				if (A[j].voto < A[imin].voto)
				{
					imin = j;
				}
			}
			B = A[i];
			A[i] = A[imin];
			A[imin] = B;
		}
}

/* Stampa in ordine decrescente di voto tutti gli studenti promossi */
void stampaPromossi(FILE* A, struct studente* B)
{
		int i;
		for (i = 9; i > 0; i--)
		{
			if (B[i].voto >= 18)
			{
				fprintf(A, "%s\t\t%s\t\t%d\n", B[i].cognome, B[i].nome, B[i].voto);
			}
		}
}

/* Stampa in ordine crescente di voto tutti gli studenti bocciati */
void stampaBocciati(FILE* A, struct studente* B)
{
		int i;
		for (i = 0; i < 10;i++)
			if (B[i].voto < 18)
			{
				fprintf(A, "%s\t\t%s\t\t%d\n", B[i].cognome, B[i].nome, B[i].voto);
			}
}
```