

# C: I File

# I File

Nel C è possibile operare su file solamente in termini di sequenze di byte: **stream**

Non esistono funzioni di alto livello come in altri linguaggi

Usiamo i puntatori. File di **testo** e file **binari**

👉 Attenzione al SO.

Con i file si ha *finalmente* la possibilità di salvare i dati inseriti all'interno del programma

# I File: apertura

Prima di poter scrivere o leggere da file è necessario **aprire** il file stesso

Per far questo è necessario includere la libreria `stdio.h` che contiene il tipo derivato `FILE`

👉 Per leggere e scrivere su file si utilizza un puntatore (a tipo derivato `FILE` )

```
FILE *fileptr; //puntatore al file
```

# I File: apertura e chiusura

1. `fopen()` associa un puntatore al file

👉 Oltre al nome del file occorre specificare il tipo di operazione da eseguire

```
FILE *fileptr;  
fileptr = fopen("my_file", "r"); //r -> in lettura
```

2. `fclose()` chiude un file e permette di salvare i dati

```
fclose(fileptr);
```

# I File: operazioni in apertura

- `r` : solo lettura – se il file non esiste `fopen( )` restituisce errore ( `NULL` )
- `w` : solo scrittura – se il file non esiste sarà creato, altrimenti sovrascritto
- `r+` : lettura e scrittura – se il file non esiste `fopen( )` restituisce errore ( `NULL` )
- `w+` : scrittura e lettura - se il file non esiste sarà creato, altrimenti sovrascritto
- `a` : solo scrittura a fine file (append) – se il file non esiste sarà creato, altrimenti mantenuto
- `a+` : lettura e scrittura a fine file (append) – se il file non esiste sarà creato, altrimenti mantenuto

# I File: funzioni per lettura e scrittura

`fprintf()` e `fscanf()` : simili a `printf()` e `scanf()` ma interagiscono con i file e consentono di agire in modo "formattato"

# I File: esempio `fprintf()`

```
#include <stdio.h>

int main() {
    FILE *fileptr;
    int i;

    fileptr=fopen("file_01.txt","w");
    for(i=1;i<=5;i++)
        fprintf(fileptr,"%d ",i);
    fclose(fileptr);
}
```

# I File: esempio `fscanf()`

```
#include <stdio.h>

int main() {
    FILE *fileptr;
    int i,num;

    fileptr=fopen("file_01.txt","r");
    for(i=1;i<=5;i++) {
        fscanf(fileptr,"%d",&num);
        printf("%d ",num);
    }
    fclose(fileptr);
}
```



# I File: fine file...

```
#include <stdio.h>

int main() {
    FILE *fileptr;
    int num;

    fileptr=fopen("file_01.txt","r");
    while(fscanf(fileptr,"%d",&num)==1) printf("%d ",num);
    fclose(fileptr);
}
```

In C la funzione `feof()` rende un valore diverso da 0 quando arriva a fine file - ma attenzione

# I File: altre funzioni per lettura e scrittura

`fgets()` : legge una riga da un file

`fputs()` : scrive una riga su un file

Il fine riga è individuato dal carattere `\n`

# I File: `fgets()`

```
#include <stdio.h>
#define DIM 100
int main() {
    FILE *fileptr;
    char vettore[DIM];
    fileptr=fopen("file_01.txt","r");
    if(fileptr==NULL) printf("\nFile inesistente\n");
    else {
        fgets(vettore,DIM,fileptr);
        printf("%s",vettore);
        fclose(fileptr);
    }
}
```

## I File: `fgets()`

```
s=fgets(vettore,DIM,fileptr);
```

`s` indirizzo di vettore

`vettore` vettore che conterrà la riga letta

`DIM` dimensione del vettore

`fileptr` puntatore al file da leggere

La funzione `fgets()` aggiunge a fine linea il carattere `\0` di fine stringa!

# I File: `fgets()` e `stdin`

```
#include <stdio.h>
#define DIM 100

int main() {
    FILE *fileptr;
    char vettore[DIM];
    fileptr=fopen("file_02.txt","w");
    printf("Inserisci del testo: ");
    fgets(vettore, DIM, stdin);
    fprintf(fileptr, "%s", vettore);
    fclose(fileptr);
    return 0;
}
```

## I File: `fgets()` e `stdin`

La funzione `fgets()` può quindi essere sfruttata per leggere una riga da tastiera (superando i limiti della `scanf()`) dirottando l'input dal file alla tastiera ( `stdin` )

## I File: `fputs()`

La funzione `fputs()` scrive una riga su un file

```
fputs(vettore, fileptr);
```

# I File: **fputs()**

```
#include <stdio.h>
#define DIM 100
int main() {
    FILE *fileptr;
    char vettore[DIM], invio;
    int n, i;
    fileptr=fopen("file_03.txt", "w");
    printf("\nQuante linee vuoi scrivere: ");
    scanf("%d", &n);
    scanf("%c", &invio);
    for(i=1; i<=n; i++) {
        printf("\nInserisci linea %d: ", i);
        fgets(vettore, DIM, stdin);
        fputs(vettore, fileptr);
    }
    fclose(fileptr);
}
```



# I File: leggere e scrivere un singolo carattere

`fgetc()` e `fputc()`

- `fgetc(fileptr)` : restituisce un intero. Costante simbolica EOF (-1)
- `fputc(c, fileptr)` - Richiede in ingresso il carattere da scrivere

# I File: note

- `fflush(fileptr)` : scarica su disco tutte le scritture contenute nel buffer
- Messaggio a video: `fprintf(stdout, "Messaggio per l'utente");`
- La funzione `gets()` non si deve usare!

# Esercizio

Scrivere un programma che permetta di caricare N numeri interi in un array e successivamente, tramite apposite funzioni consenta di:

- calcolare la media dei valori
- scrive su file i valori maggiore della media

# Soluzione

```
#include <stdio.h>
#define DIM 10

float calcola_media(int *, int);
void scrivi_file(int *, int, float);

int main() {
    int n,v[DIM],i;
    float media;

    do {
        printf("Inserisci dimensione array: \n");
        scanf("%d", &n);
    } while(n<1 || n>DIM);

    printf("Inserisci i %d elementi:\n",n);
    for(i=0;i<n;i++) {
        printf("elemento di indice - %d : ",i);
        scanf("%d",&v[i]);
    }
}
```

# Soluzione

```
media=calcola_media(v, n);  
printf("\nLa media vale: %.1f",media);  
  
for(i=0;i<n;i++)  
    printf("\nelemento di indice - %d: %d ",i, v[i]);  
  
scrivi_file(v, n, media);  
}
```

# Soluzione

```
float calcola_media(int *v, int n) {
    int i;
    float media=0;

    for(i=0; i<n; i++)
        media+=*(v+i);
    return media/n;
}

void scrivi_file(int *v, int n, float media) {
    FILE *fileptr;
    int i;

    fileptr=fopen("my_file.txt", "w");
    for(i=0; i<n; i++) {
        if(*(v+i)>media)
            fprintf(fileptr, "%d ", *(v+i));
    }
}
```

# Esercizio

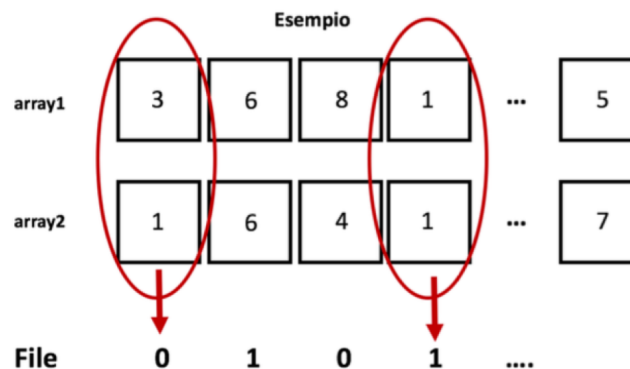
Scrivere un programma in C che:

- legga M numeri interi da un file e li carichi su un array
- legga N numeri interi da un secondo file e li carichi su un secondo array

M e N non sono noti a priori e comunque entrambi sono minori di 100.

Il programma deve permettere, attraverso apposite funzioni, di:

- 1- calcolare (separatamente per ogni array) la media dei numeri interi contenuti in un determinato array
- 2- scrivere un nuovo file che contenga una sequenza di 0 e 1 rispettando la seguente regola: nel nuovo file si scriverà 0 quando i due array in posizione  $i$  hanno un valore tra loro diverso, viceversa si scriverà 1 se i due array in posizione  $i$  hanno un valore uguale. Nello scorrere i due array si presti attenzione al fatto che le loro dimensioni potrebbero essere differenti.



# Soluzione

```
#include <stdio.h>
#define DIM 100

void scrivi_file(int *, int *, int, int);

int main() {
    FILE *fileptr1,*fileptr2;
    int v1[DIM],v2[DIM],i,j;

    fileptr1=fopen("file_01.txt","r");
    if(fileptr1==NULL) printf("\nFile inesistente\n");
    else {
        i=0;
        while(fscanf(fileptr1, "%d", &v1[i])==1) i++;
    }
}
```



# Soluzione

```
fileptr2=fopen("file_02.txt","r");  
if(fileptr2==NULL) printf("\nFile inesistente\n");  
else {  
    j=0;  
    while(fscanf(fileptr2, "%d", &v2[j])==1) j++;  
}
```

# Soluzione

```
if(fileptr1!=NULL && fileptr2!=NULL)
    scrivi_file(v1, v2, i, j);
}
```

# Soluzione

```
void scrivi_file(int *v1, int *v2, int d1, int d2) {  
    FILE *fileptr;  
    int i;  
  
    fileptr=fopen("file_03.txt","w");  
    if(d1>d2) d1=d2;  
    for(i=0;i<d1;i++) {  
        if(*(v1+i) == *(v2+i)) fprintf(fileptr,"%d ",1);  
        else fprintf(fileptr,"%d ",0);  
    }  
}
```

# I File: `fread()` - (low-level reading)

`fread()` permette di leggere un file, traferire (stream) su un vettore e restituire il numero di elementi letti

Se è stato letto tutto il contenuto del file `fread()` restituisce `0`

La lettura avviene in sequenza spostando il puntatore del numero di byte necessario.

# I File: esempio di lettura

```
#include <stdio.h>
#define DIM 100

int main() {
    FILE *fileptr;
    char vettore[DIM];
    int size=1; //in byte di 1 elemento
    int n; //num elementi letti
    int i;

    fileptr=fopen("file01","r"); //path!

    if(fileptr==NULL)
        printf("\nFile inesistente\n");
    else {
        n=fread(vettore,size,DIM,fileptr);
        for(i=0;i<n;i++)
            printf("%c",vettore[i]);
        fclose(fileptr);
    }
}
```

## I File: `fwrite()` - (low-level writing)

`fwrite()` permette di scrivere su un file, trasferire (stream) da un vettore ciò che legge e restituire il numero di elementi scritti

La scrittura avviene in sequenza spostando il puntatore del numero di byte necessario.

# I File: esempio di scrittura

```
#include <stdio.h>
#include <string.h>
#define DIM 100

int main() {

    FILE *fileptr;
    char vettore[DIM];
    int size=1,n,i,len;

    printf("Inserisci del testo: ");
    scanf("%s",vettore);
    len=strlen(vettore);

    fileptr=fopen("file02","w");
    n=fwrite(vettore,size,len,fileptr);
    fclose(fileptr);
}
```

```

#include <stdio.h>
int main() {
    FILE *fileptr;
    int vettore[5],i;
    fileptr=fopen("file_04","wb");
    if(fileptr==NULL) printf("\nFile inesistente\n");
    else {
        for(i=0;i<5;i++)
            scanf("%d",&vettore[i]);
        fwrite(vettore,sizeof(int),5,fileptr);
        fclose(fileptr);
    }
    fileptr=fopen("file_04","rb");
    if(fileptr==NULL) printf("\nFile inesistente\n");
    else {
        fread(vettore,sizeof(int),5,fileptr);
        printf("\nElementi in array:");
        for(i=0;i<5;i++)
            printf("\n%d",vettore[i]);
        fclose(fileptr);
    }
}

```



# Esempio: programma Libretto

Vedi file libretto.c

# I File: *sequenziale vs random*

Posizionare il puntatore: `fseek( )`

Consente di posizionare il puntatore in una qualunque posizione all'interno del file (sia in lettura che in scrittura)

`fseek( fp, n, 0 )` fp viene posizionato sul n° byte a partire dall'inizio del file

`fseek( fp, n, 1 )` fp viene posizionato sul n° byte a partire dalla posizione attuale

`fseek( fp, n, 2 )` fp viene posizionato sul n° byte a partire dalla fine del file

`n=ftell( fp )` //restituisce la posizione attuale del puntatore

```
#include <stdio.h>
int main() {
    FILE *fileptr;
    int i;
    fileptr=fopen("/Users/matteofraschini/Desktop/file03.txt","w");
    for(i=0;i<=9;i++) {
        printf("%ld ",ftell(fileptr));
        fprintf(fileptr,"%d ",i);
    }
    fclose(fileptr);
    fileptr=fopen("/Users/matteofraschini/Desktop/file03.txt","r+");
    fseek(fileptr, 4, 0);
    fprintf(fileptr,"%d",0);
    fclose(fileptr);
}
```