Apertura di un file:

Si utilizza la funzione `fopen` contenuta in `stdio.h` la cui intestazione è:

```
FILE * fopen(char *nomefile, char *modo);
```

- `FILE` anche questo è definito nella libreria `stdio`, ed è un tipo struct che contiene le informazioni necessarie alla gestione del file.
- Il primo parametro è la stringa nome del file,
- Il secondo parametro è una stringa che contiene un unico carattere e indica la modalità di utilizzo del file.
- Esempio: `FILE *fp; fp = fopen("pippo.dat", "r");`
- L'apertura deve essere effettuata prima di poter operare su un qualsiasi file.

Modalità di apertura di un file

- `"r"` (read) ... sola lettura
- `"w"` (write) ... crea un nuovo file per la scrittura; se il file esiste già ne elimina il contenuto corrente
- `"a"` (append) ... crea un nuovo file, o accoda ad uno esistente per la scrittura
- `"r+"` ... apre un file per l'aggiornamento (lettura e scrittura), a partire dall'inizio del file
- `"w+"` ... crea un nuovo file per l'aggiornamento; se il file esiste già ne elimina il contenuto corrente
- `"a+"` ... crea un nuovo file, o accoda ad uno esistente per l'aggiornamento

- Aprire un file inesistente in scrittura, il file viene creato (ovviamente bisogna avere diritti di scrittura nella directory corrente)
- Aprire un file inesistente in lettura è un errore.
- Il valore calcolato da `fopen` deve essere memorizzato in una variabile e passato a tutte le funzioni che operano sul file.
- se si verifica un errore in fase di apertura, `fopen` restituisce `NULL`
- Esempio:

```
FILE *fp;
if ((fp = fopen("pippo.dat", "r")) == NULL)
    printf("Errore!\n");
    exit(1);
else ...
```

Chiusura di un file:

Si utilizza la funzione `fclose`, contenuta in `stdio.h`, passando il file pointer. L'intestazione della funzione è:

```
int fclose(FILE *fp);
```

- Esempio: `fclose(fp);`
- Se si è verificato un errore, `fclose` restituisce `EOF` (-1).
- La chiusura va sempre effettuata, appena sono terminate le operazioni di lettura/scrittura da effettuare sul file.
- Si consiglia di aprire i file subit rima di effettuare le operazioni di lettura e scrittura e di chiuderlo appena le operazioni sono terminate.
- Si consiglia anche di usare nomi diversi di file pointer per file diversi.

Scrittura su e lettura da file

Consideriamo solo file ad accesso sequenziale:

- si legge e si scrive in sequenza, senza poter accedere agli elementi precedenti o successivi.
- vediamo solo le analoghe di `printf` e `scanf`
- attenzione alla distruzione del contenuto precedente del file, quando si scrive sul file.
- Vediamo un esempio di scrittura:

```
int ris1, ris2;
FILE *fp;
if ((fp = fopen("risultati.dat", "w")) != NULL) {
      ris1 = ...;   ris2 = ...;
      fprintf(fp, "%d %d\n", ris1, ris2);
    fclose(fp);}
```

Scrittura su file:

La funzione `fprintf`

- Intestazione di `fprintf`:

```
int fprintf(FILE *fp, char *formato, ...);
```

- come `printf`, tranne che per il parametro `fp`
- `printf(...);` è equivalente a `fprintf(stdout, ...);`
- scrive sul file identificato da `fp` a partire dalla posizione corrente
- il file deve essere stato aperto in scrittura, append, o aggiornamento
- in caso di errore restituisce la costante simbolica `EOF`(-1 o valore negativo) altrimenti il numero di byte scritti

Lettura da file:

La funzione `fscanf`

- Intestazione di `fscanf`:

```
int fscanf(FILE *fp, char *formato, ...);
```

- come `scanf`, tranne che per `fp`
- `scanf(...);` è equivalente a `fscanf(stdin, ...);`
- legge dal file identificato da `fp` a partire dalla posizione corrente
- il file deve essere stato aperto in lettura o aggiornamento
- restituisce il numero di assegnamenti fatti agli argomenti specificati nell'attivazione dopo la stringa di formato
- se il file termina o si ha un errore prima del primo assegnamento, restituisce la costante simbolica `EOF`

Verifica di fine file:

Si usa la funzione `feof`

- L'intestazione della funzione `feof`

```
int feof(FILE *fp);
```

- Calcola vero (1) se per il file è stato impostato l'indicatore di end of file, ovvero: quando si tenta di leggere oltre la fine del file
- per lo standard input, quando viene digitata la combinazione di tasti
  - `ctrl-z` (per Windows)
  - `return ctrl-d` (per Unix)

Potete anche scrivere i dati direttamente su file e poi farli leggere dal programma, dovete però fare attenzione all'inserimento del fine file.

Calcolare la somma degli interi in un file

```
int somma = 0, n;
FILE *fp;

if ((fp = fopen("pippo.txt", "r")) != NULL) {
  while (fscanf(fp, "%d", &n) == 1)
    somma += n;
  fclose(fp);
}
```

Lettura da file dei dati in un array

```
void readArrayFile (arrayInt a, int size, char nomefile[]) {

  int i,j;
  FILE *flPtr=fopen(nomefile,"r");
  for (i=0;!feof(flPtr);i++) {
      j=fscanf(flPtr,"%d",&a[i]);
      printf("i=%d j= %d a[i] %d ",i,j,a[i]);}
      fclose(flPtr);
      }
int main () {
  arrayInt a;
  readArrayFile(a, SIZE,"dati.txt");
 ...
 }
```