Input/Output su File

Premessa importante

- Queste dispense si intendono unicamente come riferimento rapido per consultare il significato e I prototipi delle funzioni per l'accesso ai file.
- Per avere esempi concreti sulle modalita' di utilizzo delle stesse e' necessaria la visione dei sorgenti creati a lezione, disponibili su:

  `www.diit.unict.it/users/dpatti/FONDINF/STUFF/src_current.zip`

La struttura FILE

- Necessita' di memorizzare dati in maniera non volatile
- In C (libreria <stdio.h>) è definito un tipo FILE che:
  - astrae il concetto di sequenza di dati contenuti nella memoria di massa
  - Facilita l'accesso ai file mediante un insieme di funzioni predefinite
- Esempio di dichiarazione: `FILE *fp;`
- Le variabile fp e' un puntatore ad un tipo FILE

Apertura file: fopen

- NB:La dichiarazione di una variabile di tipo FILE* file non implica la creazione di un file, poiche' il puntatore non e' inizializzato.
- Utilizzo della funzione fopen:

```
FILE * fopen(char * name, char * mode);
```

  - name: stringa che specifica il nome del file
  - mode: stringa che specifica la modalità di trasferimento dati: “r”: lettura (il file deve già esistere) “w”: scrittura (viene creato un file di nome file name) “a”: aggiunta in coda.

Errori in apertura

- Il risultato di fopen( ) può essere NULL, una costante che indica un puntatore non valido. Questo accade se:
  - il file aperto in lettura non esiste o non si hanno I permessi di lettura
  - il file aperto in scrittura è protetto da scrittura oppure è protetta da scrittura l'unità di memoria su cui si trova
- Controllare quindi sempre il risultato di fopen( )

Esempio di apertura in lettura con fopen

```
char nome[30];
   FILE * fp;
   printf("\n File da leggere:");
   scanf("%s",nome);
   fp = fopen(nome,"r");
   if ( fp==NULL)
     printf("\n Errore apertura di %s !!",nome);
   else
   {      /* faccio quello che devo...*/
   }
```

Chiusura file: fclose

- I file aperti in C devono essere chiusi al termine dell'esecuzione:
  - I dati eventualmente ancora bufferizzati vengono effettivamente scritti sul disco
  - Si libera il puntatore a FILE, che puo' eventualmente anche essere riutilizzato
- Esempio, chiudi il FILE * fp inizializzato in precedenza con fopen:

```
fclose(fp);
```

I/O a basso livello:getc e putc

- Scrittura e lettura orientata ai caratteri
- Funzioni getc( ) e putc( )
- Prototipi:
  - int getc(FILE *infile);
  - void putc(char c, FILE *outfile);
- Ma se leggiamo caratteri, perche' il valore di ritorno e' int ?
  - Carattere speciale EOF: getc( ) ritorna un int e non un char

I/O formattato: fprintf e fscanf

- Analoghe alle printf e scanf, solo che operano su file invece che su schermo e tastiera
- NB: necessitano di un FILE * precedentemente aperto mediante fopen
- Il valore restituito e' EOF in caso di fine file.
- Vedere esempi di utilizzo su:

`http://www.diit.unict.it/users/dpatti/FONDINF/STUFF/src_current.zip`

I/O di righe: fgets

- Un file di testo, a piu' alto livello, puo' essere visto come un insieme di righe, ognuna della quali e' una sequenza di caratteri che termina con carattere '\n'. Il prototipo della funzone fgets e':

```
char *fgets(char * buffer,int dim,FILE *fp);
```

- legge una stringa da fp e la inserisce in buffer
- legge al max dim caratteri, a meno che non incontri prima un carattere di fine riga
- ritorna l'indirizzo di buffer oppure NULL se la lettura è fallita