**Lezione 13/03/2024**

Testing dei programmi e uso dei file.

**Testing**

consiste nel testare il programma con dati attesi, cioè con dati i quali sappiamo già la risposta.

* Oracle = output atteso.

Non possiamo dire che il programma sia perfetto, poiché con infiniti input possiamo avere infiniti output.

Bugging -> individuare il difetto è correggerlo, errori di analisi o di logica sono i peggiori, poiché non sono errori che riesci a risolvere facilmente,

* Quando troviamo un bug dobbiamo ripetere tutti i test da capo, perché potremmo aver riparato una cosa, rompendone un’altra
  + Per questo utilizziamo i file per dare in input i file.
* Potrebbe succedere che anche se il testing, dia dei buoni risultati ma ci sia ancora un errore.

Non potendo rappresentare tutti gli elementi, quindi le ragruppiamo in classi di input.

* **Test suite**

Dati principali di testing.

Primo ragionamento tagli dell’input:

* Numero di elementi array n > 1
* Caso specifico n = 1

Secondo come sono fatti questi dati di input:

* Array non orinato
* Array ordinato crescente
* Array ordinato decrescente

Nel caso specifico i vari casi si sovrappongono: quindi i **test case** sono 4:

* 3 nel cas n > 1
* 1 nel caso n = 1

**Flussi(Stream)**

Sorgente di input, o di una destinazione per l’output.

**Aprire (e chiudere) un File**

Usiamo la macro **FILE con fopen**  che restituisce un puntatore a file, in una certa mode

* r : read
* w : write
* rw : write and read
* rb : leggi in binario
* wb : scrivi in binario
* a+ : accodamento (append)
* w+ : tronca il file
* r+ : il file deve esistere.

**Const ->** Variabile costante: Non può essere modificata

**fclose(Puntatore a file)** smette di leggere il file, se non si chiude il file alla fine del programma potrebbe esserci dei problemi, con il file. -> Return 0, in caso di successo.

**Fgets** -> legge riga per riga, fino al \n oppure fino alla fine del limite dei caratteri inserti dalla stringa, se abbiamo letto tutto il file allora il valore di ritorno sarà **null.**

**Fscanf** -> legge ed immagina da stream da un certò strem. Return dati letti e scritti con successo

**Fprintf ->** Stampa su un certo steam.

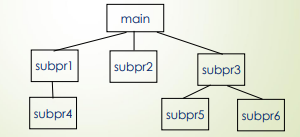
Invece ci sono delle funzioni che lavorano sulle stringhe:

**sscanf** possiamo leggere l’input da una stringa

**sprinf** possiamo stampare elementi da una stringa, possiamo spezzare una stringa, e stampare come variabili.

**Testing**

* **Big-Bang:** Testiamo l’intero programma, senza testare le funzioni nel loro individualità, è quando troviamo un errore allora non sapremo dove trovarlo
* **Strategia incrementali:** testare e integrare un sottoprogramma alla volta.



Testiamo prima le funzioni **chiamate (Callie) e poi le funzioni chiamanti (Caller)**