1 - RECAP DEL C

Recap dei passaggi per compilare il codice sorgente:

Il **compiler** verifica la correttezza sintattica del codice sorgente e costituisce un file oggetto che viene salvato su disco. Gli errori **logici** non possono essere individuati dal compilatore, come la verifica se un programma *x* andrà in loop o meno.
Il **linker** collega i vari file oggetto costruiti dal compilatore e unisce eventuali librerie esterne, al fine di generare il file eseguibile.
Il **loader** carica in memoria ed esegue l'eseguibile compilato. Il processo è preso in carico dalla CPU che esegue sequenzialmente le istruzioni ed eventualmente alloca della memoria per creare variabili.

PROBLEM SOLVING

Prima di scrivere un codice bisogna attuare un **paradigma**, ovvero ricercare una corretta soluzione tramite un implementazione concettuale, la quale avviene:

- Determinando un **analisi**
- Creare con cura l'approccio risolutivo (Top-down, bottom up, ecc...)
- Produrre uno pseudo-code o flowchart
  Mentre si scrive il programma invece bisogna:
- Comprendere quali parti di codice riprendere e sono state già risolte
- Seguire la procedura standard (Inizializzazione, Elaborazione, Visualizzazione)

PROGRAMMAZIONE STRUTTURATA

***TEOREMA DI BOHEM JACOPINI**:**
Ogni programma può essere scritto usando tre strutture di controllo fondamentali:

- **SEQUENZA**
- **SELEZIONE**
- **ITERAZIONE**

PADDING

Il padding è una tecnica utilizzata per allineare i dati in memoria, tramite l'inserimento di uno o più byte (indirizzi) vuoti, inseriti (o lasciati vuoti) a loro volta tra gli indirizzi di memoria assegnati per gli altri membri della struttura durante l'allocazione della memoria. L'architettura di un processore del computer è tale da poter leggere 1 word (4 byte nel processore a 32 bit) dalla memoria alla volta. Per sfruttare questo vantaggio del processore, i dati sono sempre allineati come un pacchetto da 4 byte che porta a inserire indirizzi vuoti tra l'indirizzo di un altro membro, portando un occupazione diversa di byte a seconda dell'ordine di dichiarazione

In linguaggi come C il padding viene aggiunto automaticamente dal compilatore per allineare i dati in memoria. Ad esempio:

```
struct Esempio {
    char a;      // 1 byte
    int b;       // 4 byte
    short c;     // 2 byte
};
```

Una struttura comunque ha un valore diverso di byte a seconda dell'ordine in cui viene dichiarata