

# Introduzione al corso

---

Prof. Salvatore Venticinque

Prof. Mario Magliulo

# Cosa vuol dire programmare

---

Per realizzare l'esecuzione automatica dell'informazione occorre:

- Disporre di un elaboratore
- Rappresentare l'informazione (dati)
- ***Rappresentare l'elaborazione (programma)***

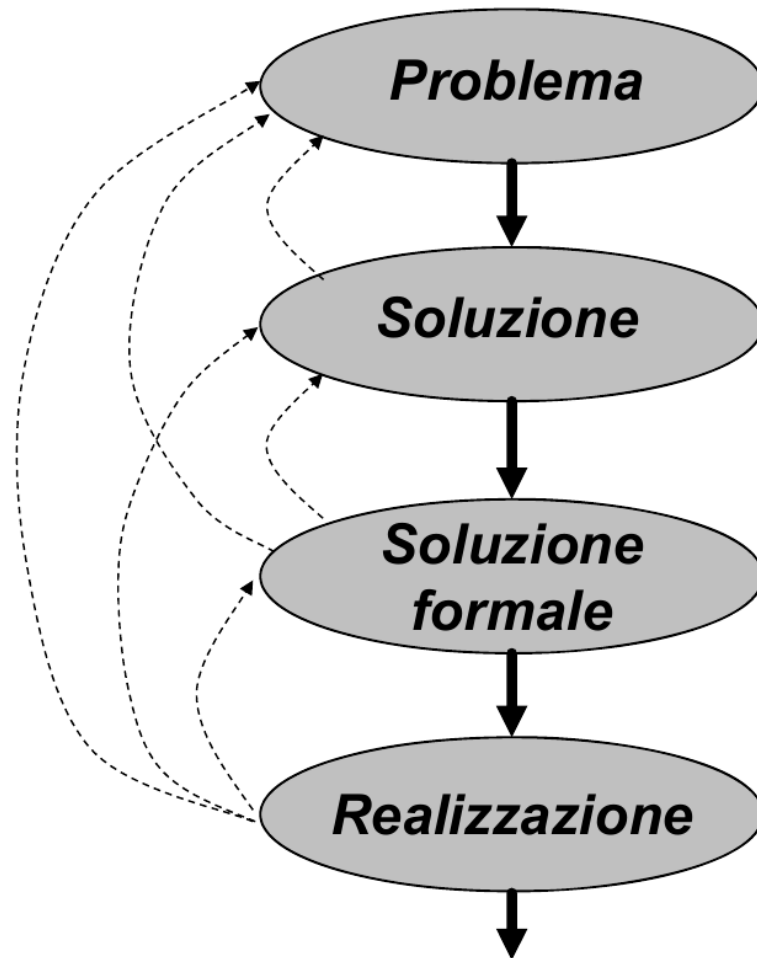
# Cosa vuol dire programmare

---

- Scrivere un programma significa adattare la logica funzionale della macchina alle esigenze operative di un certo problema, secondo uno svolgimento sequenziale.
- Le esigenze operative del problema scaturiscono dalla logica risolutiva idonea al conseguimento dei risultati voluti.
- La logica funzionale di macchina è l'insieme di operazioni elementari che questa è in grado di svolgere.
- Per svolgimento sequenziale si intende la successione nel tempo delle operazioni elementari.

# Progettazione e sviluppo software

---



# Difficoltà

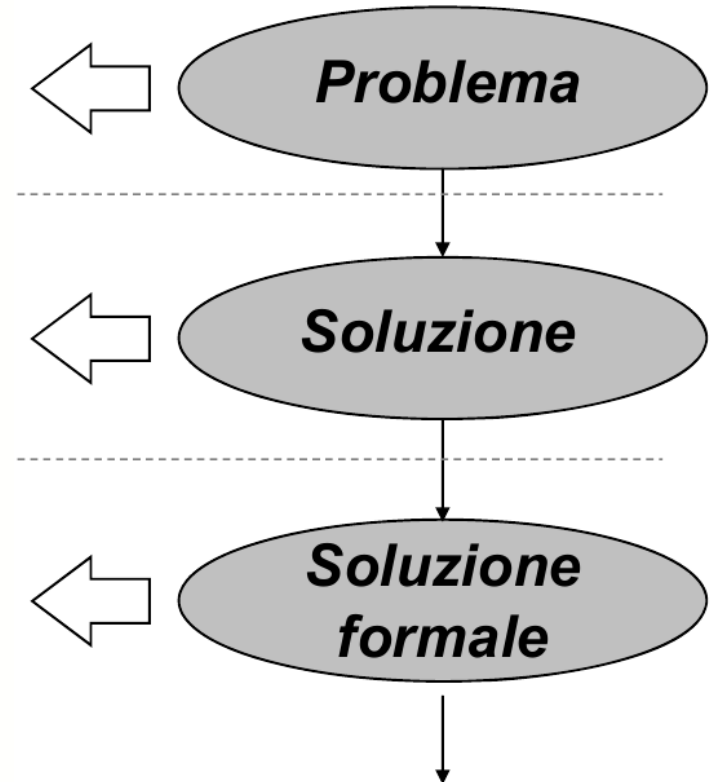
---

I punti critici nello sviluppo di un progetto risiedono essenzialmente in:

- Sviluppo di una soluzione “informale”
- Formalizzazione di una soluzione
  - Permette una più semplice “traduzione” nelle regole di realizzazione
  - La soluzione di un problema passa generalmente attraverso lo sviluppo di un algoritmo

# Esempio di Progettazione

- **Problema:** Calcolo del massimo tra due valori A e B
- **Soluzione:** Il massimo è il più grande tra A e B...
- **Soluzione formale:**
  - 1. inizialmente:  $\text{max} = 0$
  - 2. se  $A > B$  allora  $\text{max} = A$ ; stop
  - 3. altrimenti  $\text{max} = B$ ; stop



# Realizzazione

---

Codifica della soluzione utilizzando:

***un modello eseguibile da un calcolatore***

Ovvero

Codifica della soluzione attraverso:

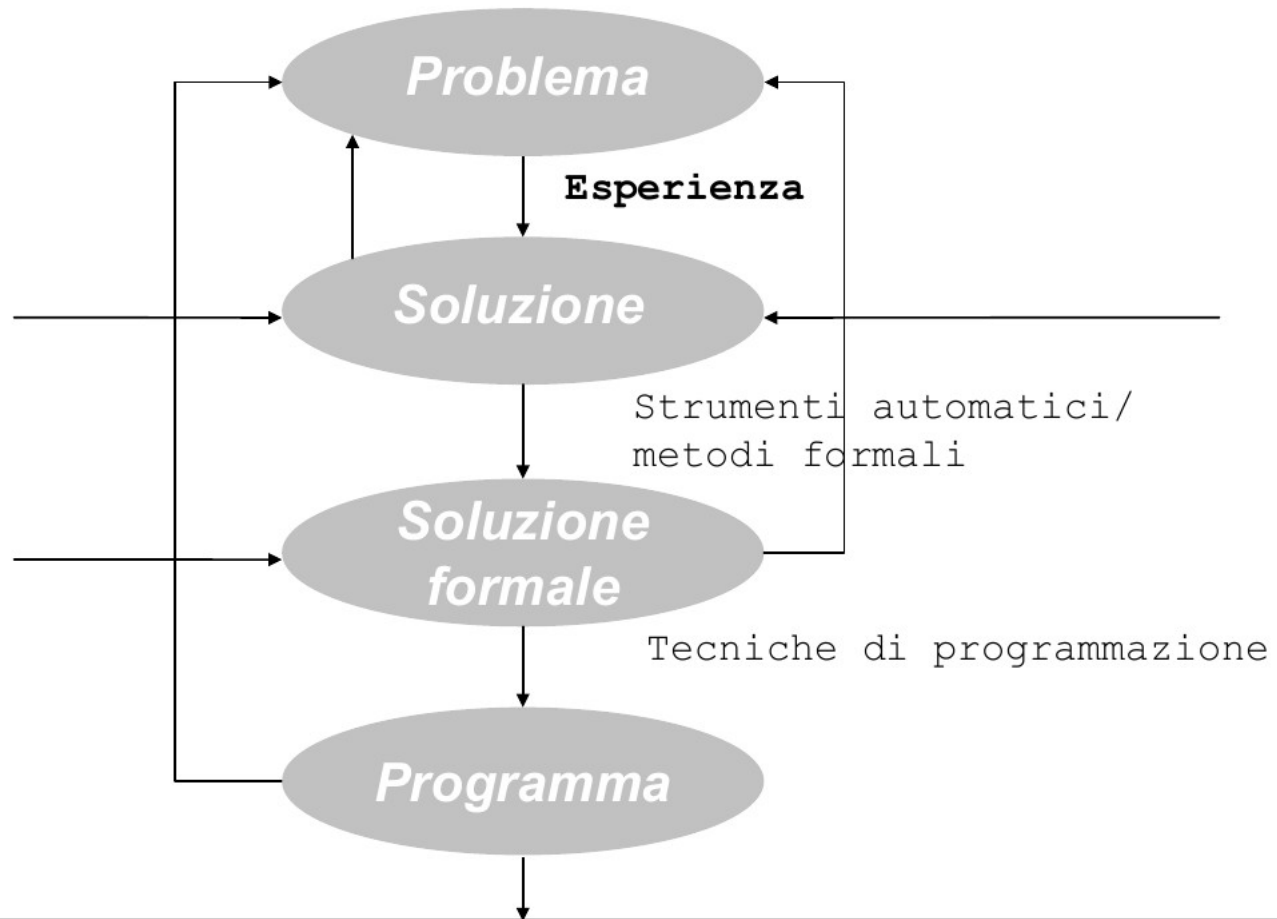
***un linguaggio di programmazione***

*Il risultato dello sviluppo*

***un software: un programma Eseguibile***

# Progettazione Rivista

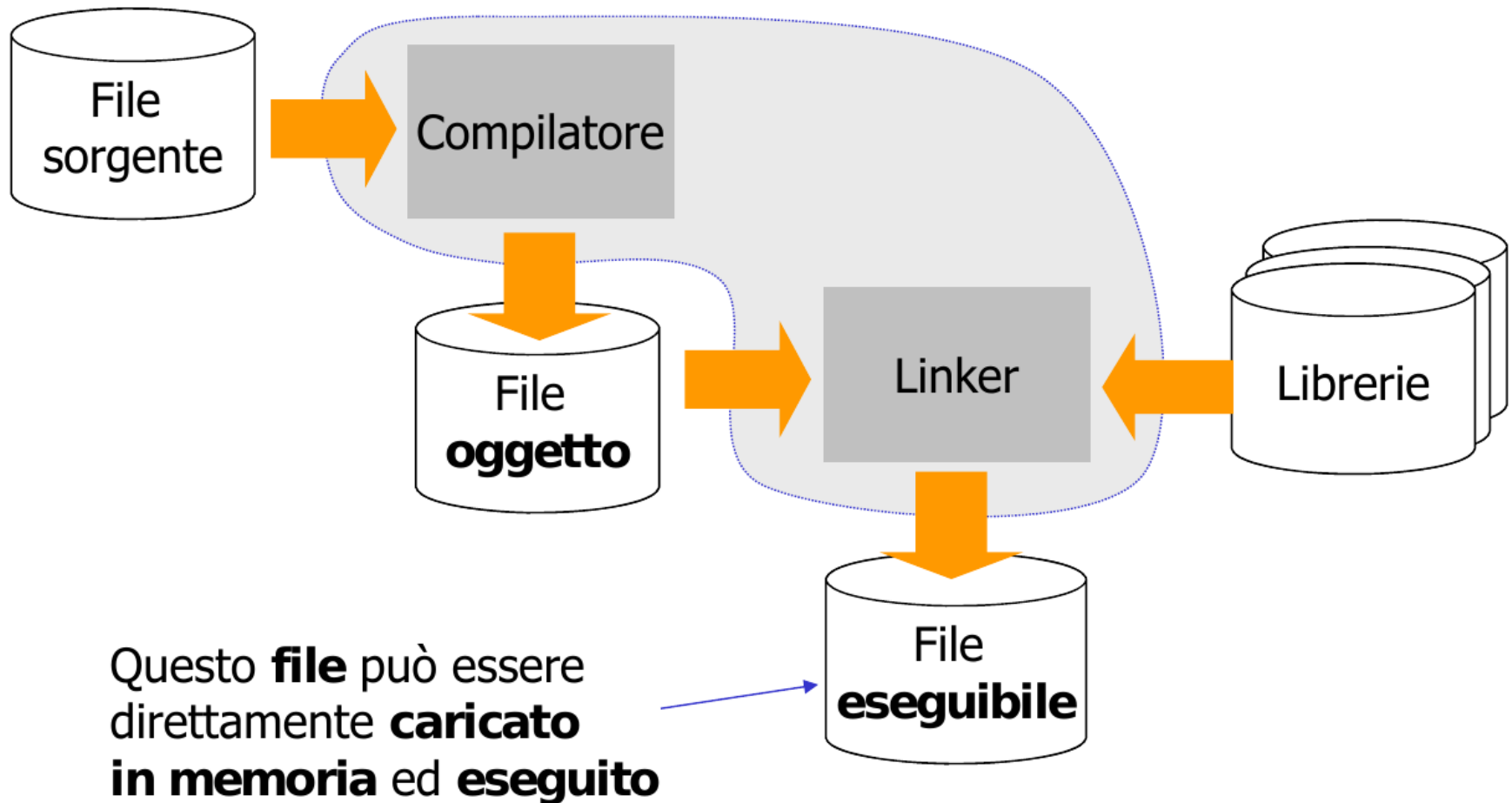
---





# Ciclo di Sviluppo del Software

---



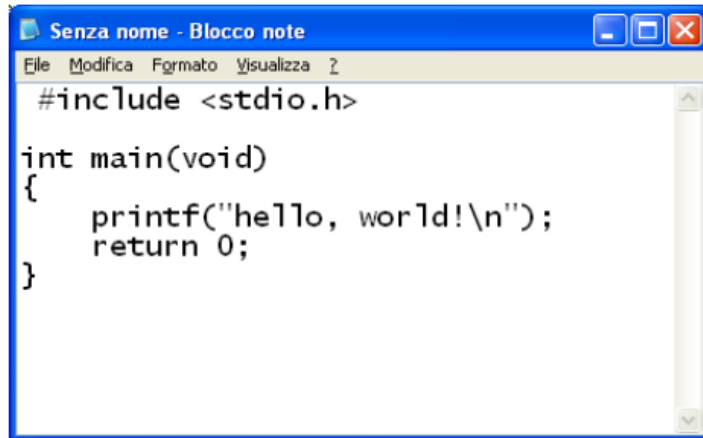
# Linguaggi di Programmazione

---

- Linguaggi per la codifica di algoritmi:
  - scrittura sotto forma di programmi che possano essere compresi da un elaboratore
- Dal linguaggio della macchina ai linguaggi di alto livello:
  - sforzo di traduzione da linguaggio naturale a linguaggio macchina sempre più affidato alla macchina stessa

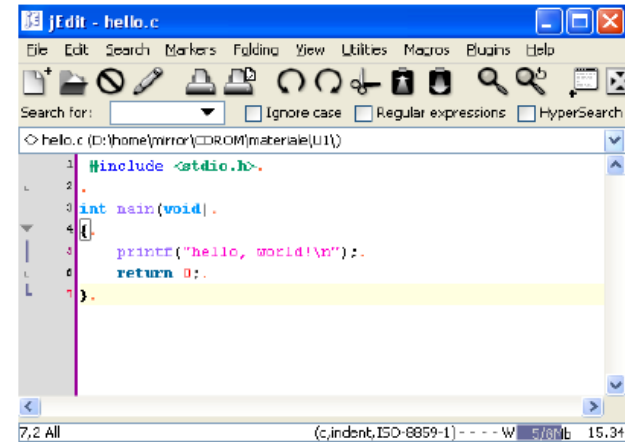
# Scrivere un programma

- **Strumento : Editor di testo**
  - Notepad
  - Notepad++
  - Geany



```
#include <stdio.h>

int main(void)
{
    printf("hello, world!\\n");
    return 0;
}
```



```
1 #include <stdio.h>.\n2 .\n3 int main(void).\n4 {\n5     printf("hello, world!\\n");.\n6     return 0:.\n7 }
```

- **Output : file di testo**
  - mioprogramma.c

# Il programma C

---

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    printf("hallo world!!");
```

```
}
```

# Compilare il programma

---

- **Strumento:** Compilatore (un software)
  - Intel compiler
  - GNU compiler
  - Mingw compiler

**Output:** un file oggetto

Il risultato è la traduzione del sorgente in linguaggio macchina (myoprogramma.o)

*Differenza tra linguaggio assembler e linguaggio macchina ???*

*Ottenere un linguaggio assembler?*

# Linking

---

- Input: file(s) oggetto
- Strumento: Compilatore
- Output: File eseguibile
  - ***Mioprogramma.exe***

# Test

---

- Input: file(s) Dati di Test
- Strumento: Debugger
- Output: ***Risultato Corretto?***