# Laboratorio di programmazione per sistemi ciberfisici

## 2. Introduzione alla programmazione

*Enrico Martini*

*October 9, 2025*

# La programmazione

***Obiettivo.*** Bisogna esprimere la soluzione ad un problema in un linguaggio che possa essere capito dalla macchina.

- Un **programma** è un insieme di istruzioni necessarie per risolvere un problema.
- Passi per la risoluzione di un problema:
  - **Analisi logica** della consegna
  - Traduzione della consegna in **algoritmo**
  - Traduzione dell'algoritmo in **istruzioni** (C, C++, Python,...)
  - **Test** del programma

# Linguaggi di programmazione

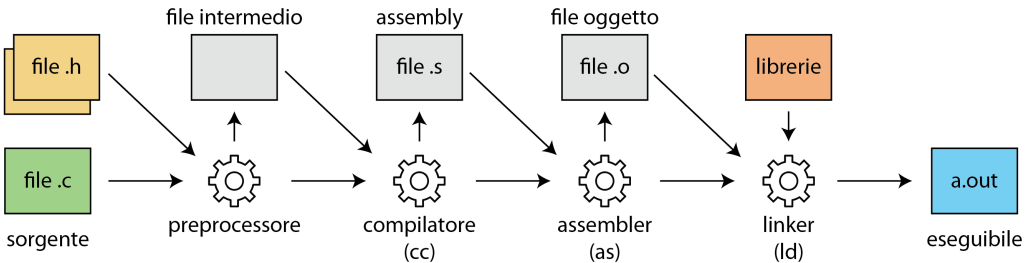
- Inizialmente bisognava programmare in **codice binario** (010001010)
- Più avanti svilupparono i linguaggi a **basso livello** (es: assembly)
- Successivamente nacquero i linguaggi di **alto livello** (es: C, Python)

```
1 hello db 'Hello, World!', 0xA
2 mov eax, 4
3 mov ebx, 1
4 mov ecx, hello
5 mov edx, 13
6 int 0x80
7 mov eax, 1
8 xor ebx, ebx
9 int 0x80
```

```
1 printf("Hello World!");
```

# Compilazione di un programma

Esistono particolari programmi chiamati **compilatori** che traducono le istruzioni di un linguaggio ad alto livello in istruzioni di basso livello.



# Primo Programma C

Il programma da compilare viene prima scritto in un file di testo (chiamato **codice sorgente**)

Esempio `hello_world.c`:

```c
#include <stdio.h>

int main(void) {
  printf("Hello world\n");       // Stampa "Hello world" sul terminale
  return 0;
};
```

# Building

Una volta salvato, bisogna trasformare il codice C in eseguibile, attraverso il processo di **building**:

- Se scopre degli **errori**, si blocca e lo segnala al programmatore
- Il programma, se privo di errori, viene **tradotto** prima in linguaggio assembly e poi in linguaggio binario
- In caso di dipendenze esterne, il programma binario viene **unito** agli altri programmi di cui necessita (e.g., printf, scanf, sqrt, . . . )

Comandi da eseguire sul terminale:

```
1 gcc <nome codice sorgente>.c    # Compila il codice sorgente e genera a.out
2 ./a.out                         # Esegue il programma a.out
```

# Analisi del primo programma

```c
#include <stdio.h>                  // Libreria per interagire con terminale

int main(void) {                    // Indica dove inizia il programma
  printf("Hello world\n");          // Funzione che stampa sul terminale
  return 0;                         // Significa che il programma ha funzionato
};
```

- le lettere maiuscole sono distinte da quelle minuscole
- i **commenti** servono a documentare un programma (`//`)
- Tutte le istruzioni devono terminare con un `;`
- Il carattere `\n` serve per andare a capo quando si stampa

# La funzione printf()

- Contenuta all'interno della libreria `stdio.h`
- Accetta una stringa (`"..."`) come argomento principale
- Può stampare anche dei dati, inserendo un segnaposto (es: `%i`)

Esempi di utilizzo:

```c
printf("Ciao Mondo\n");                      // "Ciao Mondo"
printf("L'albero ha %i mele.\n", 5);         // "L'albero ha 5 mele."
printf("Ho fatto %i esami su %i.\n", 2, 8); // "Ho fatto 2 esami su 8."
printf("%i + %i = %i\n", 12, 15, 12 + 15);  // "12 + 15 = 27"
```

# Le variabili

Una variabile identifica una **porzione di memoria** destinata a contenere dati che possono variare.

- **Dichiarazione.** Chiedo al programma di riservare uno slot in memoria
- **Assegnamento.** Salvo un valore all'interno di quello slot di memoria

```c
#include <stdio.h>

int main(void) {

  int A;      // Dichiarazione (variabile chiamata A di tipo intero)
  A = 5;      // Assegnamento (salvo nello slot chiamato A il valore 5)
  int B = 3; // Dichiarazione e assegnamento combinati
  return 0;
};
```

# Come chiamare le variabili

Ogni variabile:

- Deve iniziare per **lettera o underscore**
  (`var, _variabile, A , b, ...`)
- Deve contenere una sequenza di **caratteri alfa-numerici**
  (`var1, var2, ...`)
- E' **case-sensitive** (`var` $\neq$ `Var`)
- Non può avere parole **riservate** (no `int, main, return , ...`)
- Non può contenere **spazi** (no `variabile 1, A B, ...`)
- Deve avere un **nome rappresentativo** (no `pippo, pluto, ...`)

# I Tipi Fondamentali

| Keyword | Descrizione | Segnaposto |
|---|---|---|
| `int` | Numero intero | `%i` |
| `float` | Numero decimale a precisione singola | `%f` |
| `double` | Numero decimale a precisione doppia | `%lf` |
| `char` | Singolo carattere | `%c` |
| `_Bool` | Valore booleano (0,1) | `%i` |

```c
int    x = 10;            // numeri interi: 1, 2, -10, 1243
float  y = 0.2;           // numeri decimali: 0.1, 145.2 , -2353.27
double z = 0.21342342;    // numeri decimali molto piccoli: 0.00000000012
char   w = 'm';           // caratteri della tabella ASCII: 'a', 'Z', '9'
_Bool  v = 0;             // numero che corrisponde a VERO (1) o FALSO (0)
```

# Tabella ASCII

```
cook@pop-os:~$ ascii -d
   0 NUL     16 DLE     32        48 0     64 @     80 P     96 `    112 p
   1 SOH     17 DC1     33 !      49 1     65 A     81 Q     97 a    113 q
   2 STX     18 DC2     34 "      50 2     66 B     82 R     98 b    114 r
   3 ETX     19 DC3     35 #      51 3     67 C     83 S     99 c    115 s
   4 EOT     20 DC4     36 $      52 4     68 D     84 T    100 d    116 t
   5 ENQ     21 NAK     37 %      53 5     69 E     85 U    101 e    117 u
   6 ACK     22 SYN     38 &      54 6     70 F     86 V    102 f    118 v
   7 BEL     23 ETB     39 '      55 7     71 G     87 W    103 g    119 w
   8 BS      24 CAN     40 (      56 8     72 H     88 X    104 h    120 x
   9 HT      25 EM      41 )      57 9     73 I     89 Y    105 i    121 y
  10 LF      26 SUB     42 *      58 :     74 J     90 Z    106 j    122 z
  11 VT      27 ESC     43 +      59 ;     75 K     91 [    107 k    123 {
  12 FF      28 FS      44 ,      60 <     76 L     92 \    108 l    124 |
  13 CR      29 GS      45 -      61 =     77 M     93 ]    109 m    125 }
  14 SO      30 RS      46 .      62 >     78 N     94 ^    110 n    126 ~
  15 SI      31 US      47 /      63 ?     79 O     95 _    111 o    127 DEL
```

# Esercizio 1

Compilare ed eseguire il seguente programma:

```c
#include <stdio.h>

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

# Esercizio 2

Compilare ed eseguire il programma che calcola la somma di due numeri interi e stampa a video il risultato:

```c
#include <stdio.h>

int main(void) {
  int value1;
  int value2;
  int sum;
  value1 = 10;
  value2 = 23;
  sum = value1 + value2;
  printf("La somma di %i e %i e':%i\n", value1, value2, sum);
  return 0;
};
```

# Esercizio 3

Scrivere un programma che calcola la differenza tra `0.12357439` e `24.962816` e stampa a video il risultato.

# Esercizio 4

Scrivere un programma che stampi a video la tabellina del 43 (usare $*$ per la moltiplicazione).

```
43 x 1 = 43
43 x 2 = ...
43 x 3 = ...
43 x 4 = ...
43 x 5 = ...
43 x 6 = ...
43 x 7 = ...
43 x 8 = ...
43 x 9 = ...
43 x 10 = 430
```

# Esercizio 5

Scrivere un programma che dichiari 4 variabili A,B,C,D che dovranno contenere numeri decimali. Assegnare alle variabili, rispettivamente, i valori: 12.3, 15.5321, $-93.0$ e 12746.0372. Stampare a video il risultato della somma di A con B e della differenza tra D e C.

```
A + B = ...
D - C = ...
```

# Esercizio 6

Svolgere gli esercizi presenti su Moodle, alla sezione ”Esercizi”, dal titolo ”Introduzione”.