Corso di Programmazione in C

Lezione 1 — Introduzione e Ambiente di Sviluppo

```
/**
 * @author Rocco Mazzeo
 * @email rocco.mazzeo@gmail.com
 * @linkedin https://www.linkedin.com/in/roccomazzeo
 */
```

Obiettivi della Lezione

- Comprendere il contesto e le potenzialità del linguaggio C
- Imparare a installare e configurare l'ambiente di sviluppo
- Esplorare la struttura di un programma C
- Scrivere, compilare, eseguire e correggere semplici programmi
- Sviluppare autonomia nella risoluzione di problemi di base

> **Nota:** Questo materiale è pensato per studenti principianti. Alcuni argomenti saranno approfonditi nelle lezioni successive. Per domande o chiarimenti, non esitare a chiedere durante la lezione o tramite i canali dedicati.

Cos’è il Linguaggio C

- Linguaggio di programmazione imperativo, strutturato, compilato
- Sviluppato da Dennis Ritchie nel 1972 presso i Bell Labs
- Semplice, vicino all’hardware ma molto potente
- Pilastro nello sviluppo di:
  - Sistemi operativi (UNIX, Linux, Windows)
  - Driver, firmware, microcontrollori
  - Software ad alte prestazioni e videogiochi

Principali Linguaggi di Programmazione

Linguaggio	Anno di nascita	Creatore	Tipologia
C	1972	Dennis Ritchie	Compilato
Python	1991	Guido van Rossum	Interpretato
Java	1995	James Gosling	Compilato/VM
C++	1985	Bjarne Stroustrup	Compilato

Perché Imparare il C?

- Base teorica e pratica per comprendere altri linguaggi (C++, Java, Python)
- Aiuta a sviluppare pensiero logico e attenzione all’efficienza
- Linguaggio fondamentale in ambito scientifico, ingegneristico e industriale
- Spesso richiesto per l’accesso a corsi universitari e carriera tecnica

Storia Rapida del C

Anno	Evento
1972	Nasce come evoluzione del linguaggio B
1978	Pubblicazione del libro "The C Programming Language" (Kernighan & Ritchie)
1989	ANSI C (standardizzazione)
1999	C99, con nuove funzionalità
2011	C11, ulteriori evoluzioni

Dove si Usa il C Oggi?

- Sistemi operativi (kernel, utility)
- Firmware e microcontrollori (Arduino, ESP32)
- Applicazioni embedded e real-time
- Giochi e motori grafici
- Compiler e interpreti di altri linguaggi

Ambiente di Sviluppo: Cosa Serve?

- **Compilatore**: GCC, clang, MSVC
- **IDE completo**: Code::Blocks, Dev-C++, Visual Studio, CLion
- **Editor di testo**: VSCode, Sublime, Vim, Notepad++
- **Terminale**: per compilazione e test manuale

Cos’è un Compilatore?

- Programma che traduce il codice sorgente (umano) in linguaggio macchina (eseguibile dal computer)
- Controlla errori di sintassi prima dell’esecuzione
- Output: file eseguibile (.exe su Windows, senza estensione su Linux/Mac)

IDE vs Editor di Testo

- **IDE** (*Integrated Development Environment*): ambiente integrato di sviluppo con editor, compilatore, debugger (es. Code::Blocks)
    - Vantaggi: autocompletamento, gestione progetti, debug visuale
- **Editor**: solo scrittura, richiede compilazione manuale (es. Notepad++)
    - Più leggero, consigliato per chi ha già esperienza

Esercizio Guidato: Installazione di Code::Blocks o GCC

1. Scarica Code::Blocks da https://www.codeblocks.org/downloads/
2. Installa il programma seguendo la procedura guidata
3. Crea una nuova cartella “ProgettiC” sul desktop
4. Avvia Code::Blocks, crea un nuovo progetto “HelloWorld”

Come è Fatto un Programma C?

Struttura Minima

```c
#include <stdio.h> // Libreria standard input/output

int main() {
    // Corpo del programma
    return 0;
}
```

- `#include <stdio.h>` : importa funzioni essenziali (es. printf)
- `int main() { ... }` : punto di ingresso; tutto il codice parte da qui
- `return 0;` : restituisce al sistema operativo che il programma è andato a buon fine

Le Librerie in C

- Raccolte di funzioni pronte all'uso
- `#include <nome_libreria.h>`
  - `<stdio.h>`: standard input/output (printf, scanf)
  - `<stdlib.h>`: funzioni di utilità (malloc, rand)
  - `<math.h>`: funzioni matematiche (sqrt, pow)
  - `<string.h>`: funzioni per la manipolazione di stringhe (strlen, strcpy)

> **Nota:** La sintassi per includere le librerie (`#include <nome_libreria.h>`) deve essere esatta: anche un piccolo errore (ad esempio, dimenticare le parentesi angolari `< >` o scrivere male il nome della libreria) impedirà la compilazione del programma.

Approfondimento: Funzione main

- Può ricevere parametri (avanzato)
- Deve sempre restituire un intero (`int`)
- Da qui parte sempre l’esecuzione

```
/**
 * Esegue l'operazione principale della funzione.
 *
 * @returns Restituisce un valore che indica l'esito dell'operazione o il risultato calcolato,
 *          utile per determinare il comportamento successivo o per utilizzare il valore ottenuto.
 *
 * Nota: Il valore di ritorno è fondamentale per comprendere l'esito della funzione e può essere utilizzato
 *       per prendere decisioni logiche nel flusso del programma.
 */
```

Commenti nel Codice

```
// Questo è un commento su una riga

/*
   Questo è un commento
   su più righe
*/
```

- I commenti non vengono eseguiti, servono per spiegare il codice
- Fondamentali per la manutenzione e la collaborazione

Commenti iniziali nel Codice

```
/*
    programma: nomefile.c
    autore: tuo nome
    data: data odierna
    descrizione: breve descrizione del programma
*/
```

Esempio Pratico: Il classico “Hello, World!”

```c
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

- `printf` : funzione per scrivere su schermo
- `\n` : carattere speciale, va a capo

Esempio Maggiore: Input da Tastiera e Uso di Variabili

```c
#include <stdio.h>

int main() {
  char nome[50];
  int eta;
  printf("Come ti chiami? ");
  scanf("%s", nome);
  printf("Quanti anni hai? ");
  scanf("%d", &eta);
  printf("Ciao, %s! Hai %d anni.\n", nome, eta);
  return 0;
}
```

- `scanf`: funzione per leggere input da tastiera
- `%s` e `%d`: specificatori di formato per stringhe e interi
- `&eta`: indirizzo della variabile dove memorizzare il valore inserito

Esempio Maggiore: Input da Tastiera e Uso di Variabili

> **Nota:**
> Quando si usa `%s` con `scanf`, non si deve anteporre la `&` al nome della variabile stringa. Questo perché il nome dell'array (ad esempio `nome`) è già un puntatore al primo carattere della stringa. Scrivere `scanf("%s", nome);` è corretto, mentre `scanf("%s", &nome);` è sbagliato e può causare errori. Invece, per i tipi semplici come `int`, bisogna usare la `&` (ad esempio `scanf("%d", &eta);`), perché serve l'indirizzo della variabile.

Esempio Maggiore: Somma di Due Numeri

```c
#include <stdio.h>

int main() {
  int a, b, somma;
  printf("Inserisci il primo numero: ");
  scanf("%d", &a);
  printf("Inserisci il secondo numero: ");
  scanf("%d", &b);
  somma = a + b;
  printf("La somma è: %d\n", somma);
  return 0;
}
```

- Mostra come acquisire più valori e calcolare un risultato

Buone Pratiche per Principianti

- Scrivi sempre commenti chiari e utili
- Indenta il codice per renderlo leggibile
- Dai nomi significativi alle variabili
- Compila spesso, non aspettare la fine
- Non aver paura di sbagliare: ogni errore è un’occasione per imparare

Errori Tipici dei Principianti

- Dimenticare il punto e virgola `;`
- Usare variabili non inizializzate
- Dimenticare di includere le librerie necessarie
- Confondere `=` (assegnamento) con `==` (confronto)
- Scordare l'operatore `&` in `scanf` per variabili non array

Curiosità sul Linguaggio C

- Il nome “C” deriva dall’evoluzione del linguaggio “B”.
- Il C fu usato per scrivere i primi UNIX.
- GCC, il compilatore C più famoso, è scritto in C.
- Il C permette l’uso diretto dei puntatori.
- Le stringhe in C sono array di caratteri terminati da `\0`.
- Molti linguaggi (C++, Java, PHP) derivano dal C.
- Il C è uno standard nei benchmark di prestazioni.
- Non include funzioni grafiche: è minimale e portabile.

Mini-Esercizio 1

- Modifica il programma per stampare il tuo nome e cognome

Compilazione ed Esecuzione

Da IDE (es. Code::Blocks)

- Premi “Build & Run”
- L’IDE compila e lancia il programma automaticamente

Da Terminale (avanzato)

1. Salva il file come `miofile.c`
2. Apri il terminale nella cartella del file
3. Compila:
   `gcc miofile.c –o miofile`
4. Esegui:
   `./miofile` (Linux/Mac) oppure `miofile.exe` (Windows)

Approfondimento: Fasi di Compilazione

1. **Preprocessing**: risolve direttive `#include`, `#define`
2. **Compilazione**: trasforma il C in linguaggio assembly
3. **Assemblaggio**: crea il file oggetto
4. **Linking**: unisce i file oggetto con le librerie in un eseguibile

Errori Comuni in Compilazione

- Dimenticare il punto e virgola `;`
- Mancanza di parentesi `{ }`
- Uso errato delle maiuscole/minuscole (C è case sensitive)
- File salvato con estensione sbagliata

Mini-Esercizio 2

- Prova a togliere un punto e virgola o una parentesi, compila e osserva l'errore restituito

Debugging: Come Risolvere gli Errori

- Leggi il messaggio di errore: indica la riga e il tipo di errore
- Controlla sintassi e struttura
- Usa i commenti per isolare parti di codice sospette

Focus: La Funzione printf

```
printf("Ciao, %s! Hai %d anni.\n", "Luca", 25);
```

- `%s` : stampa una stringa
- `%d` : stampa un intero
- Puoi passare più argomenti separati da virgola
- Puoi usare più specificatori di formato (`%s` , `%d` , `%.1f`) e passare i valori corrispondenti nell'ordine giusto.
- `%.1f` stampa il numero con una cifra decimale.
- L'ordine dei parametri deve corrispondere a quello dei specificatori nella stringa di formato.

Mini-Esercizio 3

- Scrivi un programma che stampi la frase:
  “Benvenuto nel corso di C!”

Variabili e Tipi di Dato (Anticipazione)

- Una variabile è uno spazio di memoria con un nome
- In C, ogni variabile deve avere un tipo (int, float, char, ...)
- Esempio:

```c
int eta = 23;
char iniziale = 'M';
```

- Approfondiremo nella prossima lezione

Esercizio Finale Guidato

1. Scrivi un programma che stampi:
    - Il tuo nome e cognome
    - La tua età
    - Il nome della tua città
2. Compila ed esegui il programma
3. Cambia i dati e ricompila

Domande Frequenti

- **C è difficile?**
  È rigoroso, ma con la pratica diventa naturale!
- **Posso usare caratteri accentati?**
  In generale sì, ma attenzione alla codifica dei file.
- **Se sbaglio, rischio di rompere il PC?**
  No, gli errori si limitano al programma. Il sistema resta sicuro.

Riepilogo della Lezione

- Cos’è e dove si usa il linguaggio C
- Come si prepara l’ambiente di lavoro
- Struttura base di un programma C
- Scrittura, compilazione, esecuzione di un semplice programma
- Come leggere e risolvere gli errori più comuni

Compiti a Casa

- Modifica il programma per stampare un messaggio a tua scelta
- Prova a scrivere un programma che stampi una poesia in più righe
- Porta domande o dubbi alla prossima lezione