

Cos'è il C++?

- Linguaggio di programmazione **general-purpose** (usato per qualsiasi scopo)
- **Estensione del C** con aggiunta di **OOP** (Programmazione Orientata agli Oggetti)
- **Compilato** → Il codice viene tradotto in linguaggio macchina prima dell'esecuzione

Dove si usa?

- ✓ Sviluppo di sistemi operativi
 - ✓ Creazione di giochi AAA (Unreal Engine, Unity)
 - ✓ Software ad alte prestazioni (browser, database)
 - ✓ Programmazione embedded (IoT, dispositivi smart)
-

2. STRUTTURA DI UN PROGRAMMA C++

Elementi fondamentali:

1. **Librerie** (`#include`) → Importano funzionalità aggiuntive
 - Esempio: `<iostream>` per input/output
2. **Funzione** `main()` → Punto di partenza dell'esecuzione
3. **Istruzioni** → Comandi eseguiti in sequenza
4. **Commenti** → Note per i programmatori (`//` o `/* ... */`)

Esempio Concettuale:

Copy

Download

1. Includi la libreria per input/output
 2. Inizia la funzione principale
 3. Stampa "Ciao Mondo"
 4. Termina il programma
-

3. VARIABILI E TIPI DI DATO

Concetto Base:

Le variabili sono **contenitori** per memorizzare dati.

Tipi Primitivi:

Tipo	Descrizione	Esempio
int	Numeri interi	42, -7
float	Numeri decimali (precisione singola)	3.14f
doubl e	Numeri decimali (doppia precisione)	2.71828
char	Singolo carattere	'A', '\$'
bool	Valore logico	true, false

Dichiarazione/Inizializzazione:

Copy

Download

tipo nomeVariabile = valore;

4. OPERATORI

Aritmetici:

- + (addizione), - (sottrazione), * (moltiplicazione), / (divisione), % (modulo)

Comparazione:

- == (uguaglianza), != (disuguaglianza), >, <, >=, <=

Logici:

- && (AND), || (OR), ! (NOT)

5. STRUTTURE DI CONTROLLO

Condizionali:

- if/else: Esegue blocchi in base a condizioni

Copy

Download

```
SE (condizione) {
```

```
    // codice se vero
```

```
} ALTRIMENTI {
```

```
    // codice se falso
```

```
}
```

- switch: Scelta tra multipli casi

Copy

Download

```
SWITCH (variabile) {
```

```
    CASO val1: // codice; BREAK;
```

```
    CASO val2: // codice; BREAK;
```

```
    DEFAULT: // codice se nessun caso matcha
```

```
}
```

Cicli:

- for: Ripetizione con contatore

Copy

Download

```
PER (inizio; condizione; incremento) {
```

```
    // codice ripetuto
```

```
}
```

- while: Ripetizione finché una condizione è vera

Copy

Download

```
MENTRE (condizione) {
```

```
    // codice
```

```
}
```

- do...while: Ciclo con condizione alla fine

Copy

Download

```
FAI {  
    // codice  
} MENTRE (condizione);
```

6. FUNZIONI

Definizione:

Blocchi di codice riutilizzabili che:

- Accettano **input** (parametri)
- Eseguono operazioni
- Restituiscono un **output** (tramite return)

Sintassi Concettuale:

Copy

Download

```
tipoRitorno nomeFunzione(parametro1, parametro2) {  
    // corpo  
    return valore;  
}
```

Esempio:

Copy

Download

FUNZIONE somma(num1, num2) → ritorna num1 + num2

7. OOP (PROGRAMMAZIONE A OGGETTI)

Classi e Oggetti:

- **Classe:** Modello/template che definisce attributi e metodi
- **Oggetto:** Istanza concreta di una classe

Pilastri dell'OOP:

1. **Incapsulamento:** Nasconde i dettagli interni (private/public)
2. **Ereditarietà:** Crea nuove classi basate su esistenti
3. **Polimorfismo:** Oggetti che rispondono diversamente allo stesso metodo

Esempio Concettuale:

Copy

Download

CLASSE Animale {

 ATTRIBUTI: nome, età

 METODI: mangia(), dormi()

}

OGGETTO cane = nuovo Animale("Fido", 3)

8. GESTIONE DELLA MEMORIA

Puntatori (*):

- Variabili che memorizzano **indirizzi di memoria**
- Esempio: `int* ptr = &variabile`

Allocazione Dinamica:

- `new`: Alloca memoria
 - `delete`: Libera memoria
-

9. ERRORI COMUNI

- ✗ Dimenticare ; alla fine delle istruzioni
 - ✗ Confondere = (assegnazione) con == (confronto)
 - ✗ Accesso a memoria non allocata (segmentation fault)
 - ✗ Loop infiniti per condizioni sbagliate
-

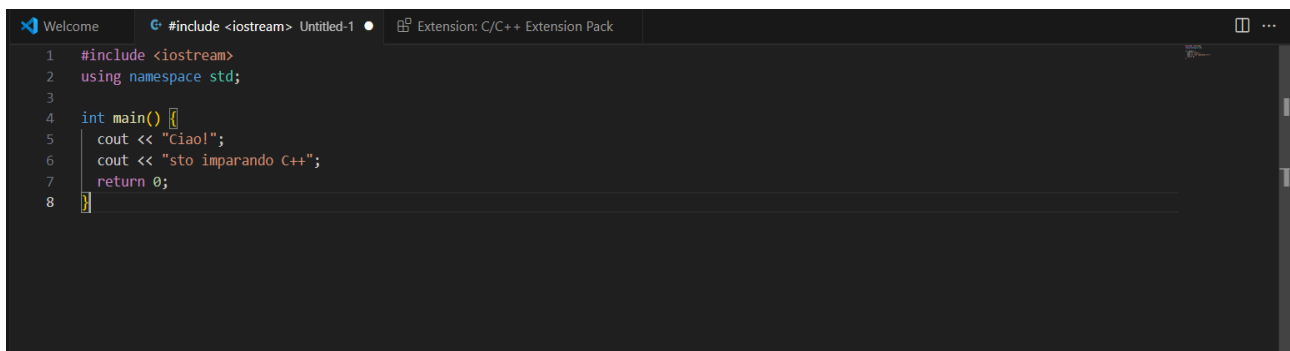
10. BEST PRACTICES

- ✓ Usa nomi significativi per variabili/funzioni
 - ✓ Commenta il codice complesso
 - ✓ Dividi il programma in funzioni/moduli
 - ✓ Testa spesso durante lo sviluppo
-

DOVE ANDARE ORA?

1. **Esercitati** con problemi semplici (calcolatrice, gestione liste)
2. **Esplora** le **Standard Template Library (STL)**: vector, string, map
3. **Approfondisci** concetti avanzati: templates, multithreading

Ecco un esempio:



```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout << "Ciao!";
6     cout << "sto imparando C++";
7     return 0;
8 }
```