Guida alla Programmazione in C/C++

1. Cos'è C e Cos'è C++?

- **C**: È un linguaggio di programmazione di basso livello, popolare per la sua efficienza e flessibilità, usato per sistemi operativi, software di sistema, e applicazioni che richiedono alte prestazioni.
- **C++**: Estensione del C, che aggiunge caratteristiche della programmazione orientata agli oggetti, come classi, incapsulamento, ereditarietà e polimorfismo, rendendolo un linguaggio più versatile per applicazioni moderne.

2. Installazione e Ambiente di Sviluppo

Per programmare in C o C++ è necessario:

- **Compilatore**: GCC (GNU Compiler Collection) per C e C++, oppure MSVC su Windows. Su Linux e macOS, GCC è spesso preinstallato.
- **IDE/Editor di Testo**: Visual Studio Code, Code::Blocks, CLion, oppure anche solo un editor come Vim o Emacs.

3. Struttura di un Programma C

Ecco un esempio di base per un programma in C:

c
Copia codice
```c
#include <stdio.h>

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

4. Struttura di un Programma C++

Ecco invece un programma base in C++:

cpp
Copia codice
```cpp
#include <iostream>

int main() {
    std::cout << "Hello, World!" << std::endl;
    return 0;
```

```
}
```

In entrambi i casi:

- **Inclusione di librerie**: `#include` aggiunge le librerie necessarie. In C utilizziamo `<stdio.h>` per funzioni di input/output come `printf`, mentre in C++ usiamo `<iostream>` per `std::cout`.
- **Funzione `main()`**: Punto di ingresso del programma. Restituisce un intero e tipicamente torna `0` per indicare che il programma è terminato correttamente.

5. Tipi di Dati e Variabili

C e C++ hanno vari tipi di dati:

- **Interi**: `int`, `short`, `long`
- **Virgola mobile**: `float`, `double`
- **Caratteri**: `char`
- **Booleani** (`bool`, solo in C++): valori `true` o `false`

Esempio:

cpp
Copia codice
```
int numero = 10;
float decimale = 5.5;
char carattere = 'A';
bool verità = true; // Solo C++
```

6. Operatori

Gli operatori in C/C++ includono:

- **Aritmetici**: `+`, `-`, `*`, `/`, `%`
- **Assegnazione**: `=`, `+=`, `-=`, `*=`, `/=`
- **Confronto**: `==`, `!=`, `>`, `<`, `>=`, `<=`
- **Logici**: `&&`, `||`, `!`

Esempio:

cpp
Copia codice
```
int a = 10;
int b = 5;
int somma = a + b;
```

```
bool maggiore = a > b;
```

7. Strutture di Controllo

Le strutture di controllo dirigono il flusso del programma:

Condizionali: `if`, `else if`, `else`
cpp
Copia codice
```
if (a > b) {
    std::cout << "a è maggiore di b" << std::endl;
} else {
    std::cout << "a non è maggiore di b" << std::endl;
}
```

-

Ciclo `for`:
cpp
Copia codice
```
for (int i = 0; i < 10; i++) {
    std::cout << i << std::endl;
}
```

-

Ciclo `while` e `do-while`:
cpp
Copia codice
```
int i = 0;
while (i < 10) {
    std::cout << i << std::endl;
    i++;
}
```

-

8. Funzioni

Le funzioni organizzano il codice in blocchi riutilizzabili.

Esempio di funzione in C++:

cpp
Copia codice

```cpp
#include <iostream>

int somma(int a, int b) {
    return a + b;
}

int main() {
    std::cout << somma(5, 3) << std::endl;
    return 0;
}
```

9. Programmazione Orientata agli Oggetti (OOP) in C++

Solo in C++, la programmazione orientata agli oggetti permette di usare:

- **Classi e Oggetti**: Le classi definiscono proprietà e comportamenti di oggetti.
- **Ereditarietà**: Una classe può ereditare attributi e metodi da un'altra classe.
- **Incapsulamento**: Nasconde i dettagli interni di una classe.
- **Polimorfismo**: Consente di trattare oggetti di classi diverse in modo uniforme.

Esempio di una semplice classe in C++:

cpp
Copia codice

```cpp
#include <iostream>
using namespace std;

class Persona {
    private:
        string nome;
        int età;

    public:
        Persona(string n, int e) : nome(n), età(e) {}

        void presentati() {
            cout << "Ciao, sono " << nome << " e ho " << età << "
anni." << endl;
        }
};

int main() {
    Persona p("Mario", 25);
```

```
    p.presentati();
    return 0;
}
```

10. Gestione della Memoria e Puntatori (C e C++)

La gestione della memoria è fondamentale, specialmente in C:

Puntatori: I puntatori memorizzano gli indirizzi di memoria.
c
Copia codice
```
int a = 5;
int *p = &a; // `p` è un puntatore che contiene l'indirizzo di `a`
```

-
- **Allocazione Dinamica della Memoria**:
  - In C, usa `malloc` e `free` per allocare e deallocare memoria.
  - In C++, usa `new` e `delete`.

Esempio in C++:
cpp
Copia codice
```
int *p = new int;
*p = 10;
delete p;
```

-

11. Gestione delle Eccezioni (solo C++)

In C++, le eccezioni consentono di gestire errori runtime:

cpp
Copia codice
```
try {
    int a = 10 / 0; // Genera un'eccezione
} catch (const std::exception& e) {
    std::cout << "Errore: " << e.what() << std::endl;
}
```

12. Input/Output di Base

In C: Usa `printf` e `scanf`.
c

Copia codice
```
int num;
printf("Inserisci un numero: ");
scanf("%d", &num);
printf("Numero inserito: %d\n", num);
```

-

In C++: Usa `std::cout` e `std::cin`.
cpp
Copia codice
```
int num;
std::cout << "Inserisci un numero: ";
std::cin >> num;
std::cout << "Numero inserito: " << num << std::endl;
```

-

13. Array e Vettori

Array: Una struttura di dati a dimensione fissa.
cpp
Copia codice
```
int numeri[5] = {1, 2, 3, 4, 5};
```

-

Vettori in C++: `std::vector` permette di usare array dinamici.
cpp
Copia codice
```
#include <vector>
std::vector<int> numeri = {1, 2, 3, 4, 5};
numeri.push_back(6); // Aggiunge un elemento
```

-

14. Compilare ed Eseguire un Programma C/C++

- **Compilazione**: Salva il file con estensione `.c` o `.cpp` e compila:
  - In C: `gcc programma.c -o programma`
  - In C++: `g++ programma.cpp -o programma`
- **Esecuzione**: Esegui il file compilato con `./programma`.

15. Conclusione

Questa guida rappresenta un'introduzione alla programmazione in C/C++. Per diventare esperti, è importante approfondire concetti avanzati come gestione avanzata della memoria, algoritmi complessi, gestione dei file e altro ancora. Buona programmazione!