1. Cos'è C++?

C++ è un linguaggio di programmazione di uso generale che estende il linguaggio C con caratteristiche di programmazione orientata agli oggetti (OOP). È conosciuto per la sua efficienza e flessibilità, permettendo sia la programmazione a basso livello che l'uso di concetti avanzati come la programmazione orientata agli oggetti e i modelli.

2. Installazione di C++

Per programmare in C++, hai bisogno di un compilatore. Ecco alcuni compilatori comunemente utilizzati:

- **GCC** (GNU Compiler Collection): disponibile su sistemi Unix e Linux.
- **MinGW**: una versione di GCC per Windows.
- **Visual Studio**: IDE che include il compilatore C++ di Microsoft.

Una volta installato un compilatore, puoi usare un editor di testo o un IDE come **Visual Studio Code** o **CLion** per scrivere ed eseguire codice C++.

3. Primi Passi con C++

Primo Programma: "Hello, World!"

Ecco un semplice programma C++ che stampa "Hello, World!" sulla console:

cpp
Copia codice
```cpp
#include <iostream>   // Libreria per input e output

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

Spiegazione

- `#include <iostream>`: include la libreria per input/output.
- `int main()`: funzione principale dove inizia l'esecuzione del programma.
- `std::cout`: oggetto che stampa testo in console.
- `return 0;`: indica che il programma è terminato con successo.

4. Variabili e Tipi di Dato

Dichiarazione di Variabili

Le variabili devono essere dichiarate con un tipo prima di essere usate:

cpp
Copia codice
```cpp
int numeroIntero = 10;        // Variabile intera
double numeroDecimale = 3.14; // Variabile decimale
char carattere = 'A';         // Carattere singolo
bool logico = true;           // Booleano
```

Tipi di Dato Comuni

- **int**: numero intero.
- **float** e **double**: numeri decimali (doppia precisione con `double`).
- **char**: singolo carattere.
- **bool**: valori booleani (`true` o `false`).
- **string**: testo, disponibile includendo `<string>`.

Esempio:

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

int main() {
    std::string saluto = "Ciao, mondo!";
    std::cout << saluto << std::endl;
    return 0;
}
```

5. Operatori

C++ supporta vari operatori, tra cui:

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

Esempio:

cpp
Copia codice
```cpp
int x = 10;
int y = 5;
int somma = x + y;
bool confronto = (x == y); // false
```

6. Strutture di Controllo

Condizionali

Le strutture `if`, `else if` e `else` permettono di prendere decisioni nel codice.

cpp
Copia codice
```cpp
int età = 18;
if (età >= 18) {
    std::cout << "Sei maggiorenne" << std::endl;
} else {
    std::cout << "Sei minorenne" << std::endl;
}
```

Cicli

C++ offre diversi cicli, tra cui `for`, `while`, e `do-while`.

cpp
Copia codice
```cpp
// Ciclo for
for (int i = 0; i < 5; i++) {
    std::cout << i << std::endl;
}

// Ciclo while
int count = 0;
while (count < 5) {
    std::cout << count << std::endl;
    count++;
}
```

7. Funzioni

Le funzioni in C++ permettono di riutilizzare il codice. Sono dichiarate con un tipo di ritorno, il nome, i parametri e il corpo della funzione.

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

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

int main() {
    int risultato = somma(10, 5);
    std::cout << "Somma: " << risultato << std::endl;
    return 0;
}
```

8. Array e Vettori

Array

Gli array memorizzano liste di valori dello stesso tipo.

cpp
Copia codice
```cpp
int numeri[5] = {1, 2, 3, 4, 5};
std::cout << numeri[0]; // Stampa 1
```

Vettori

Per una gestione dinamica degli array, utilizziamo i **vettori** della libreria `<vector>`.

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

int main() {
    std::vector<int> numeri = {1, 2, 3};
    numeri.push_back(4);    // Aggiunge 4 alla fine del vettore
```

```
    std::cout << numeri[2]; // Stampa 3
    return 0;
}
```

9. Programmazione Orientata agli Oggetti (OOP)

C++ supporta OOP, permettendo di creare classi e oggetti.

Classi e Oggetti

Le classi sono modelli per creare oggetti. Contengono **proprietà** e **metodi**.

cpp
Copia codice

```
#include <iostream>
#include <string>

class Persona {
public:
    std::string nome;
    int età;

    void saluta() {
        std::cout << "Ciao, mi chiamo " << nome << " e ho " << età
<< " anni." << std::endl;
    }
};

int main() {
    Persona persona;
    persona.nome = "Mario";
    persona.età = 30;
    persona.saluta();
    return 0;
}
```

Costruttori

I costruttori inizializzano gli oggetti al momento della creazione.

cpp
Copia codice

```cpp
class Persona {
public:
    std::string nome;
    int età;

    Persona(std::string n, int e) {
        nome = n;
        età = e;
    }
};
```

Ereditarietà

L'ereditarietà permette di creare una nuova classe basata su una classe esistente.

cpp
Copia codice
```cpp
class Animale {
public:
    void mangia() {
        std::cout << "L'animale sta mangiando." << std::endl;
    }
};

class Cane : public Animale {
public:
    void abbaia() {
        std::cout << "Il cane abbaia." << std::endl;
    }
};
```

10. Puntatori e Memoria Dinamica

C++ gestisce la memoria dinamica usando puntatori e operatori come `new` e `delete`.

cpp
Copia codice
```cpp
int* puntatore = new int; // Alloca memoria per un intero
*puntatore = 10;
std::cout << *puntatore << std::endl; // Stampa il valore 10
delete puntatore; // Libera la memoria
```

11. Gestione degli Errori

C++ gestisce gli errori usando le eccezioni (`try`, `catch` e `throw`).

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

int divide(int a, int b) {
    if (b == 0) {
        throw std::invalid_argument("Divisione per zero");
    }
    return a / b;
}

int main() {
    try {
        std::cout << divide(10, 0) << std::endl;
    } catch (const std::invalid_argument& e) {
        std::cout << "Errore: " << e.what() << std::endl;
    }
    return 0;
}
```

12. Librerie Standard

C++ offre molte librerie standard che semplificano lo sviluppo:

- **<iostream>**: per input/output.
- **<string>**: per gestire stringhe.
- **<vector>**: per vettori.
- **<algorithm>**: per funzioni di ordinamento, ricerca e manipolazione.

Esempio di utilizzo di `<algorithm>`:

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

```cpp
int main() {
    std::vector<int> numeri = {5, 3, 9, 1, 4};
    std::sort(numeri.begin(), numeri.end());
    for (int n : numeri) {
        std::cout << n << " ";
    }
    return 0;
}
```

Questa guida copre le basi di C++: dalle dichiarazioni di variabili alle strutture di controllo, OOP, puntatori, e librerie