

C++ LEVEL UP

**ELEVE SEUS CONHECIMENTOS E CONQUISTE
O MUNDO DA PROGRAMAÇÃO**

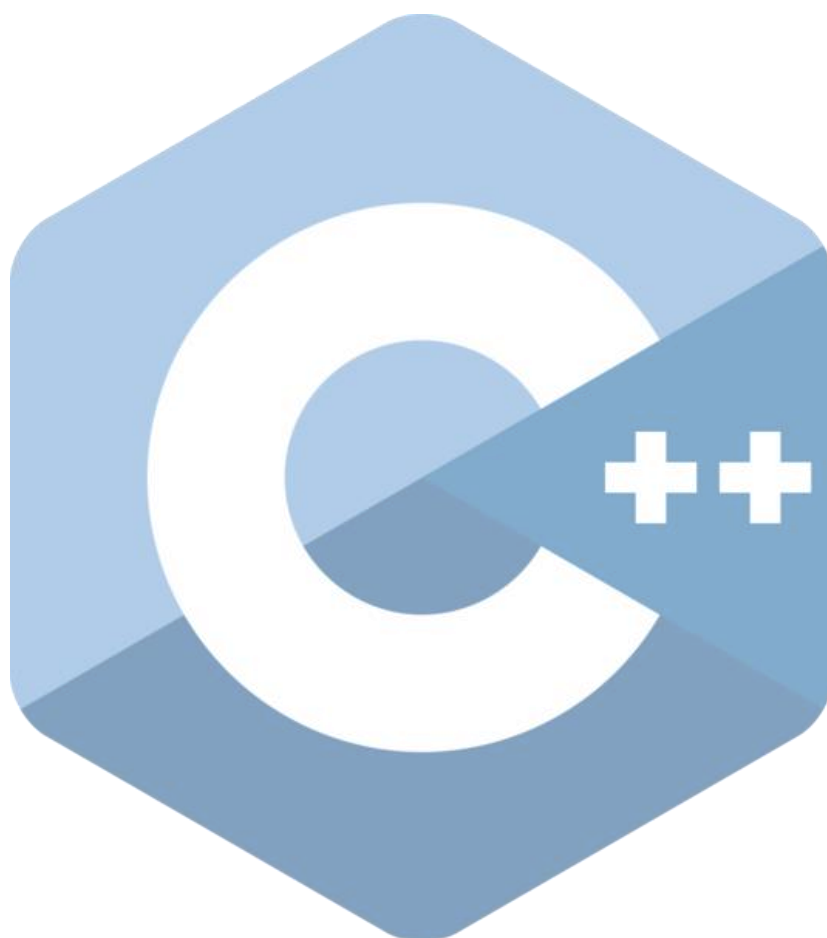


VICTOR FORTUNATO

Fundamentos básicos da Linguagem C++

Introdução ao C++

C++ é uma linguagem de programação poderosa e versátil, amplamente utilizada em desenvolvimento de software, jogos, sistemas embarcados e mais. Ela oferece uma combinação de programação de baixo nível e alto nível, permitindo um controle preciso sobre o hardware, ao mesmo tempo em que suporta abstrações complexas.



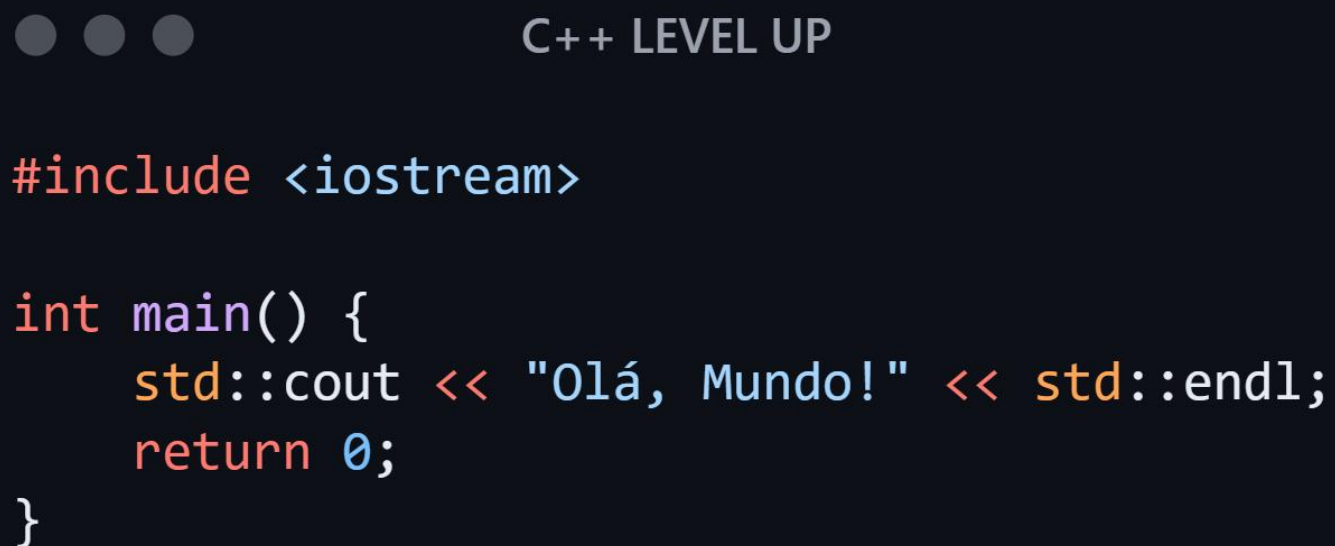
01

ESTRUTURA BÁSICA

Um programa C++ típico começa com a inclusão de bibliotecas e a definição da função `main()`, que é o ponto de entrada do programa.

Estrutura Básica de um Programa em C++

Neste exemplo abaixo, `#include <iostream>` é uma diretiva de pré-processador que inclui a biblioteca de entrada e saída padrão. A função `main()` é o ponto de entrada do programa, e `std::cout` é usado para imprimir texto na tela.



```
#include <iostream>

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

02

VARIAVEIS E TIPOS DE DADOS

Em C++, variáveis devem ser declaradas antes de serem usadas, especificando seu tipo de dado.

Declarando Variáveis e Tipos de Dados

Variáveis são usadas para armazenar dados. C++ oferece vários tipos de dados, como inteiros (int), caracteres (char), pontos flutuantes (float, double), entre outros. Aqui, declaramos uma variável inteira 'idade', uma variável de ponto flutuante 'altura', uma variável e uma variável de caractere 'inicial'.

```

C++ LEVEL UP

#include <iostream>

int main() {
    int idade = 25;
    float altura = 1.75;
    char inicial = 'A';

    std::cout << "Idade: " << idade << std::endl;
    std::cout << "Altura: " << altura << std::endl;
    std::cout << "Inicial: " << inicial << std::endl;

    return 0;
}
```

03

ESTRUTURAS DE CONTROLE

C++ suporta estruturas de controle como if, else, for, while, entre outras.

Estrutura Condicional

As estruturas condicionais permitem que o programa tome decisões. Neste exemplo o 'if' verifica se a variável 'idade' é maior ou igual a 18.

```
C++ LEVEL UP

#include <iostream>

int main() {
    int idade = 20;

    if (idade >= 18) {
        std::cout << "Você é maior de idade." << std::endl;
    } else {
        std::cout << "Você é menor de idade." << std::endl;
    }

    return 0;
}
```


Estrutura de Repetição

As estruturas de repetição, também chamada de laços de repetição, permitem executar um bloco de código várias vezes. Neste exemplo o laço 'for' executa o bloco de código repetidamente até que a condição seja falsa.

```
C++ LEVEL UP

#include <iostream>

int main() {
    for (int i = 0; i < 5; i++) {
        std::cout << "Número: " << i << std::endl;
    }

    return 0;
}
```

04

FUNÇÕES

Em C++, variáveis devem ser declaradas antes de serem usadas, especificando seu tipo de dado.

Blocos de Código Reutilizáveis

Funções são blocos de código reutilizáveis que realizam uma tarefa específica. Neste exemplo a função 'saudacao()' é chamada dentro do 'main()' para imprimir uma mensagem.

```
C++ LEVEL UP

#include <iostream>

void saudacao() {
    std::cout << "Bem-vindo ao C++!" << std::endl;
}

int main() {
    saudacao();
    return 0;
}
```

05

VETORES E MATRIZES

Em C++, variáveis devem ser declaradas antes de serem usadas, especificando seu tipo de dado.

Trabalhando com Matrizes

Uma matriz é uma coleção de variáveis de mesmo tipo, acessíveis com um único nome e armazenados contiguamente na memória. Neste exemplo uma matriz bidimensional 'matriz' é inicializada e seus valores são impressos.

C++ LEVEL UP

```
#include <iostream>

int main() {
    int matriz[2][2] = {{1, 2}, {3, 4}};

    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            std::cout << "Matriz[" << i << "][" << j << "]: " << matriz[i][j] << std::endl;
        }
    }

    return 0;
}
```

Declarando e Usando Vetores

Vetores são sequências de variáveis do mesmo tipo referenciadas por um nome único. Neste exemplo um vetor 'numeros' é declarado e inicializado com cinco valores.

```
● ● ● C++ LEVEL UP

#include <iostream>

int main() {
    int numeros[5] = {1, 2, 3, 4, 5};

    for (int i = 0; i < 5; i++) {
        std::cout << "Número: " << numeros[i] << std::endl;
    }

    return 0;
}
```

06

ORIENTAÇÃO A OBJETOS

Em C++, variáveis devem ser declaradas antes de serem usadas, especificando seu tipo de dado.

Classes e Objetos

C++ suporta programação orientada a objetos (OOP). Objetos são instâncias de classes, que são estruturas que combinam dados e métodos. Neste exemplo uma classe 'Pessoa' é definida com atributos 'nome' e 'idade' e um método 'apresentar()'.

```
C++ LEVEL UP

#include <iostream>

class Pessoa {
public:
    std::string nome;
    int idade;

    void apresentar() {
        std::cout << "Nome: " << nome << ", Idade: " << idade << std::endl;
    }
};

int main() {
    Pessoa pessoa1;
    pessoa1.nome = "João";
    pessoa1.idade = 25;

    pessoa1.apresentar();

    return 0;
}
```

07

HERANÇA

Em C++, variáveis devem ser declaradas antes de serem usadas, especificando seu tipo de dado.

Reutilizando Código

Herança é um conceito da programação orientada a objetos que permite criar uma nova classe baseada em uma classe existente. Neste exemplo a classe 'Cachorro' herda de 'Animal' e sobrescreve o método 'som()'.

```
C++ LEVEL UP

#include <iostream>

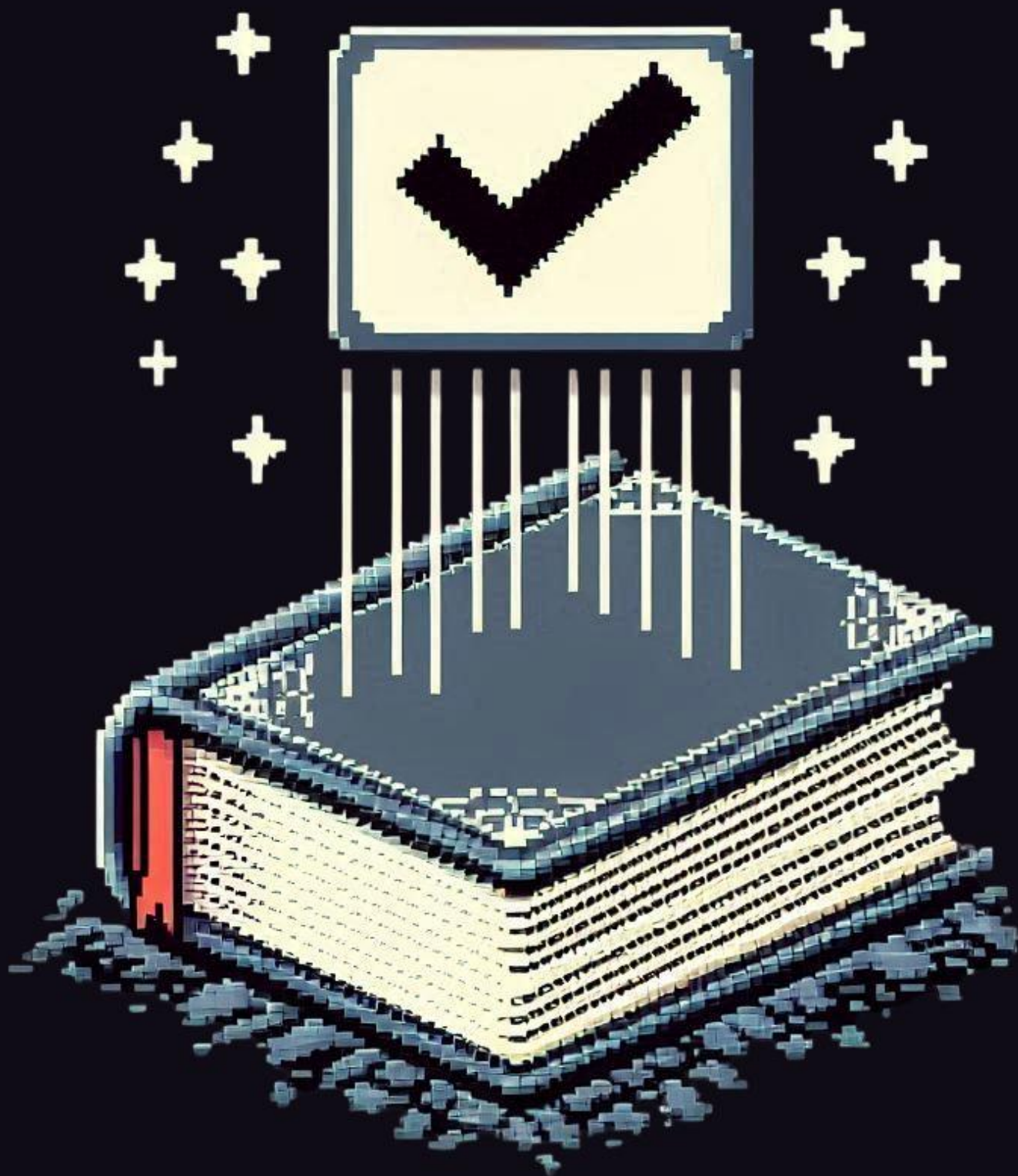
class Animal {
public:
    void som() {
        std::cout << "Som do animal" << std::endl;
    }
};

class Cachorro : public Animal {
public:
    void som() {
        std::cout << "Latido" << std::endl;
    }
};

int main() {
    Cachorro meuCachorro;
    meuCachorro.som();

    return 0;
}
```

AGRADECIMENTOS



OBRIGADOR PELA LEITURA DESTE EBOOK

Esse E-book foi gerado por IA, e diagramado e polido por humano.

O passo a passo se encontra no meu Github

