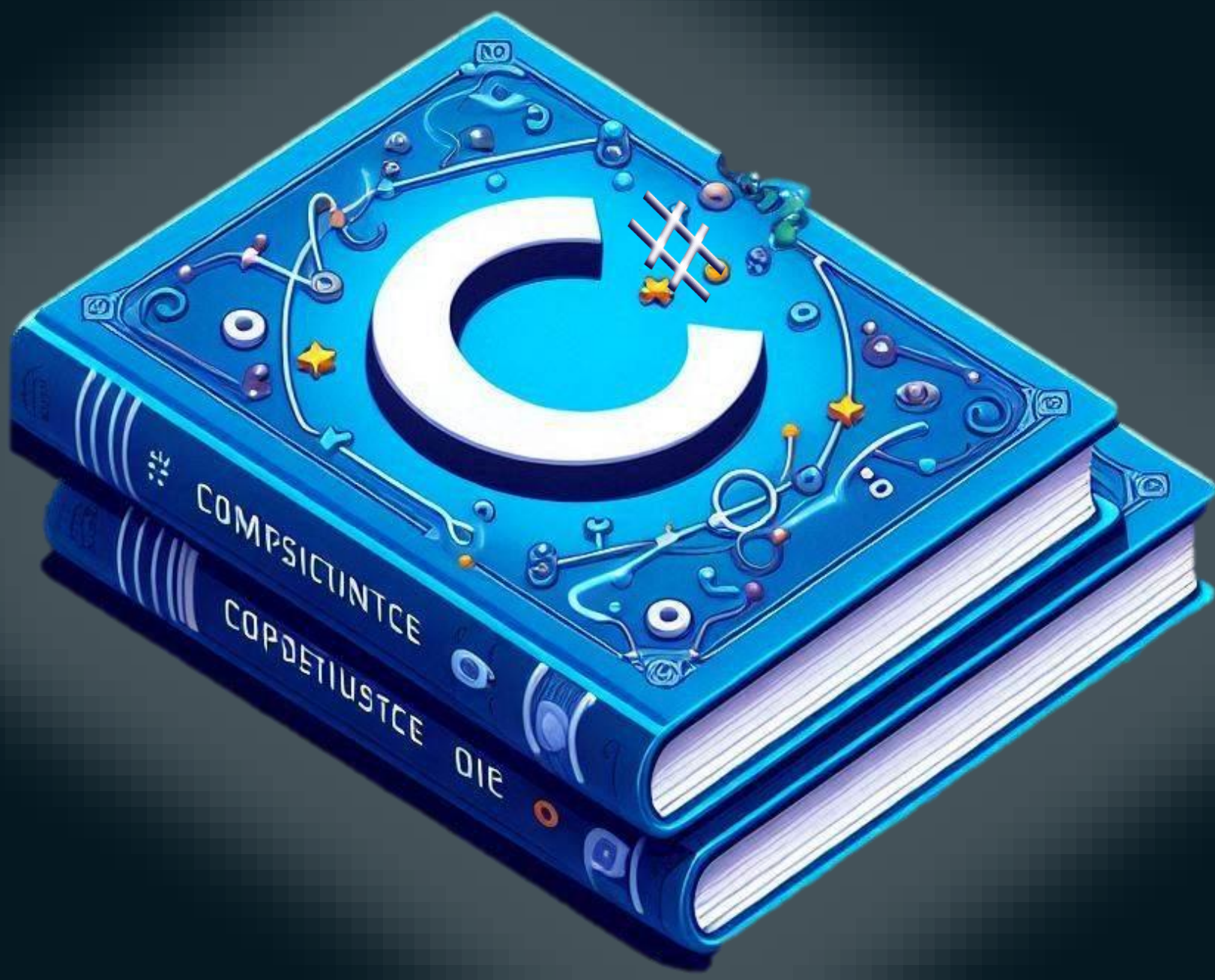
MAESTRIA EM C#



CRIANDO CÓDIGO DE ALTA PERFORMANCE

01

Fundamentos e Exemplos Práticos

Fundamentos e Exemplos Práticos

Estrutura Básica de um Programa C#

Para começar, vamos ver a estrutura básica de um programa em C#.

```
using System;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, World!");
        }
    }
}
```

Neste exemplo, utilizamos o namespace para organizar nosso código, definimos uma classe Program e um método Main, que é o ponto de entrada do aplicativo.

Fundamentos e Exemplos Práticos

Variáveis e Tipos de Dados

As variáveis são essenciais para armazenar dados temporários. O C# suporta vários tipos de dados.

```
int age = 30;
double salary = 50000.50;
string name = "John";
bool isEmployed = true;
```

Neste exemplo, declaramos variáveis de diferentes tipos: int (inteiro), double (número de ponto flutuante), string (cadeia de caracteres) e bool (booleano).

Fundamentos e Exemplos Práticos

Estruturas de Controle

Estruturas de controle são usadas para tomar decisões e repetir ações.

```
if (age > 18)
{
    Console.WriteLine("Adult");
}
else
{
    Console.WriteLine("Minor");
}
```

```
for (int i = 0; i < 5; i++)
{
    Console.WriteLine("Iteration: " + i);
}

int j = 0;
while (j < 5)
{
    Console.WriteLine("Iteration: " + j);
    j++;
}
```

Fundamentos e Exemplos Práticos

Métodos

Métodos são blocos de código que executam tarefas específicas e podem ser reutilizados.

```
static void Greet(string name)
{
    Console.WriteLine("Hello, " + name + "!");
}

static void Main(string[] args)
{
    Greet("Alice");
    Greet("Bob");
}
```

Aqui, o método Greet recebe um parâmetro name e exibe uma saudação personalizada.

Fundamentos e Exemplos Práticos

Classes e Objetos

O C# é uma linguagem orientada a objetos, o que significa que você trabalha com classes e objetos.

```
class Person
{
    public string Name { get; set; }
    public int Age { get; set; }

    public void Introduce()
    {
        Console.WriteLine("Hi, I'm " + Name + " and I'm " + Age + " years old.");
    }
}

static void Main(string[] args)
{
    Person person = new Person();
    person.Name = "Alice";
    person.Age = 25;
    person.Introduce();
}
```

Neste exemplo, definimos uma classe Person com propriedades Name e Age, e um método Introduce que exibe uma mensagem.

Fundamentos e Exemplos Práticos

Coleções

Coleções são usadas para armazenar múltiplos valores.

```
List<string> names = new List<string>();
names.Add("Alice");
names.Add("Bob");

foreach (string name in names)
{
    Console.WriteLine(name);
}
```

Usamos uma lista (List) para armazenar e iterar sobre uma coleção de nomes.

Fundamentos e Exemplos Práticos

Tratamento de Exceções

O tratamento de exceções é importante para lidar com erros de forma elegante.

```
try
{
    int result = 10 / 0;
}
catch (DivideByZeroException ex)
{
    Console.WriteLine("Error: " + ex.Message);
}
```

Neste exemplo, capturamos uma exceção de divisão por zero e exibimos uma mensagem de erro.

CONCLUSÃO

CONCLUSÃO

Este ebook abordou os principais pontos do C# de maneira simples e prática. Compreender esses fundamentos é essencial para desenvolver aplicações robustas e eficientes. Continue praticando e explorando mais recursos do C# para se tornar um desenvolvedor ainda mais competente.

AGRADECIMENTO

OBRIGADO POR LER ATÉ AQUI!

Gostaria de expressar minha sincera gratidão por dedicar seu tempo a ler este ebook sobre C#. Espero que tenha encontrado informações valiosas e exemplos práticos que ajudem a aprimorar suas habilidades em programação. Foi um prazer compartilhar este conhecimento com você.

Se desejar continuar esta jornada de aprendizado e compartilhar suas experiências, não hesite em se conectar comigo no LinkedIn ou conferir meus projetos no GitHub. Agradeço novamente por seu interesse e desejo muito sucesso em sua carreira de desenvolvedor.

Atenciosamente,

Luis Fernando