






C++
Classes e Objetos




Introdução a Orientação a Objetos 

- Problemas da Programação Procedural
 - Descentralização (repetição) de código
 - Dificuldade de manutenção
 - Dificuldade em substituir desenvolvedores
 - Muitas pessoas responsáveis por mesmos códigos em diferentes partes do projeto
 - Pouco reaproveitamento de código



Introdução a Orientação a Objetos 

- Orientação a Objetos
 - Resolve os problemas apresentados da Programação Procedural
 - Concentra responsabilidades nos locais certos
 - Flexibiliza a aplicação
 - Encapsula a lógica de negócio
 - Melhora a comunicação entre desenvolvedores
 - Aumenta a reutilização de código
 - Aumenta o ciclo de vida dos projetos
 - Menor custo de desenvolvimento e criação



Orientação a Objetos



- Siglas
 - OO: Orientação a Objetos
 - LOO: Linguagem Orientada a Objetos
 - POO: Programação Orientada a Objetos
- Características
 - Códigos pertencem a classes
 - Classes possuem propriedades próprias
 - Classes possuem funcionalidades próprias
 - Classes interagem com outras classes

Como funciona (visão geral)



Modelo Procedural

nome
sobrenome
velocidade
endereço
anos

CADASTRO USUARIOS...

CADASTRO CARROS...

Cad. 1

CODIGO..

Cad. 2

CODIGO..

Cad. 3

CODIGO..

Modelo OO

usuario->nome
usuario->sobrenome
carro->velocidade
carro->endereço
usuario->anos

usuario->salvar()

carro->salvar()

Cad. 1

CODIGO..

Cad. 2

CODIGO..

Cad. 3

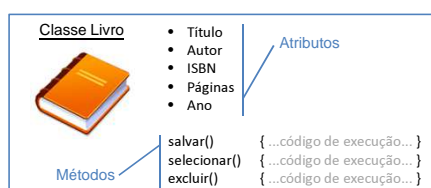
CODIGO..

Classe Usuario
CODIGO..

Classes



- Estrutura de dados
- Representa um tipo de dado
- Comportamento próprio



Atributos e Métodos



- **Atributos**
 - Características da classe
 - Tipos primitivos e/ou outras classes
 - Geralmente representados por **substantivos**
 - Exemplos: *nome, idade, endereço, cor, tamanho*
- **Métodos**
 - Ações que a classe pode realizar
 - Podem receber parâmetros e retornar valores
 - Geralmente representados por **verbos**
 - Exemplos: *salvar, ler, abrir, fechar, emprestar, devolver*

Criando classes



- Operador **class**

```
class NomeDaClasse {
    /* Atributos */
    /* Métodos */
}
```

Cabeçalhos e Implementações



- **Cabeçalhos (.h)**
 - Nome da classe
 - Atributos
 - Assinatura dos métodos
- **Implementações (.cpp)**
 - Código dos métodos

```
class Usuario {
public:    // Modificador de acesso

    char nome[60];
    int idade;

    void apresentar();
}
```

```
#include "Usuario.h"
#include <string>

/* Métodos */
void Usuario::apresentar() {
    cout << "Meu nome é " << this->nome;
}
```

Objetos

- Classe não é um objeto
- Classe é um modelo de objeto (template)
- Objetos são instâncias de classes

Criando objetos

- Operador **new**
- Cria uma instância a partir de uma classe

Construtores

- Método construtor
- Customiza o código de inicialização da classe
- Método de mesmo nome da classe

```
class Usuario {
    // Construtor padrão
    Usuario();
}
```

```
#include "Usuario.h"
#include <string>

Usuario::Usuario()
{
    // Possíveis inicializações ou outras ações
}
```

```
// Utilizando o construtor padrão
usuario = new Usuario();
strcpy(usuario->nome, "André");
usuario->idade = 30;
usuario->apresentar();
```

Sobrecarga de métodos



- Método de mesmo nome de outro
- Diferente assinatura (parâmetros)
- Construtores customizados e outros

```
class Usuario {
    // Construtor padrão
    Usuario();
    // Construtor customizado
    Usuario(char nome[], int idade);
}

#include "Usuario.h"
#include <string>

Usuario::Usuario() {}

Usuario::Usuario(char nome[], int idade)
{
    strcpy(this->nome, nome);
    this->idade = idade;
}
```

```
// Utilizando o construtor padrão
usuario = new Usuario();
strcpy(usuario->nome, "André");
usuario->idade = 30;
usuario->apresentar();

// Utilizando o construtor customizado
usuario = new Usuario("André", 30);
usuario->apresentar();
```

Destrutor



- Método destrutor
- Customiza o código de exclusão da classe
- Liberação de memória
- Método de mesmo nome da classe, com '~'
- Invocado pelo comando **delete**

```
class Usuario {
    // Destrutor
    ~Usuario();
}

#include "Usuario.h"

Usuario::~Usuario()
{
}
```

```
// Utilizando o construtor padrão
usuario = new Usuario();
strcpy(usuario->nome, "André");
usuario->idade = 30;

// Destrindo o objeto
delete usuario;

// Ou também: delete(usuario);
```

Operador *this*

- Diferencia um atributo do objeto de um atributo do método
- Fornece a referência do próprio objeto para outro método
- Funciona somente dentro do próprio objeto

```
void Usuario::meuMetodo(int var)
{
    cout << var << endl;
    cout << this->var << endl;
}
```

```
Usuario *usuario = new Usuario();

usuario->var = 10;
usuario->meuMetodo(20);

// Imprime 20 e depois 10
```

Aulas práticas e manuais on-line





Assista agora as aulas práticas.

[Clique aqui](#) para visualizar as aulas práticas disponíveis
