CIÊNCIA DA COMPUTAÇÃO
PROGRAMAÇÃO ORIENTADA A OBJETOS
PROF. JUDSON SANTIAGO

LABORATÓRIO 6

DESTRUTORES

EXERCÍCIOS DE REVISÃO

VOCÊ DEVE RESPONDER PARA REVISAR OS CONCEITOS IMPORTANTES

1. Em que momento os destrutores são chamados?

2. É sempre necessário criar um destrutor? O que acontece se não criarmos?

3. Para que servem os destrutores?

EXERCÍCIOS DE FIXAÇÃO

VOCÊ DEVE FAZER OS EXERCÍCIOS PARA FIXAR O CONTEÚDO

1. Alguns compiladores implementam a notação de chamada explícita do construtor pela criação de objetos temporários seguidos por uma atribuição, mesmo na inicialização de novos objetos.

```
Jogo sackboy = Jogo("Sackboy", 150.0f);   // cria objeto temporário?

Jogo horizon;
horizon = Jogo("Horizon", 199.0f);        // objeto temporário
```

   Teste se o seu compilador faz isso adicionando mensagens no construtor e destrutor da classe Jogo. Faça o teste com os cenários de inicialização e atribuição de objetos mostrados acima.

   Dica: prefira inicialização sobre atribuição para ter a melhor eficiência

2. Crie um programa para descobrir qual a ordem de construção e destruição de variáveis locais. Para isso crie objetos de uma classe que exiba mensagens no seu construtor e destrutor.

```
int main()
{
    Jogo sackboy { "Sackboy", 150.0f };
    Jogo spiderman { "Spiderman", 200.0f };
    Jogo horizon { "Horizon", 199.0f };
}
```

   Dica: objetos alocados na pilha são destruídos em ordem inversa da sua criação.

3. Os atributos de um objeto constante não podem ser modificados. Construa um programa para testar o que acontece se tentarmos chamar métodos que modificam e que não modificam os atributos de um objeto constante.

```
const Jogo ratchet { "Ratchet & Clank", 150.0f };
ratchet.atualizar(125.0f);  // modifica
ratchet.exibir();           // não modifica
```

   Atualize a declaração do método exibir, como mostrado abaixo, e teste novamente.

```
void exibir() const;
```

   Dica: sempre use const em métodos que não alteram os objetos. O uso de const é uma promessa feita ao compilador de que aquele método não irá alterar os atributos do objeto.

EXERCÍCIOS DE APRENDIZAGEM

VOCÊ DEVE ESCREVER PROGRAMAS PARA REALMENTE APRENDER

1. Transforme o registro Atleta e as funções abaixo em uma classe contendo:

   - Um construtor padrão
   - Um construtor com parâmetros para inicializar acertos e tentativas
   - Um destrutor, se necessário
   - Um método inline para implementar a função calcular
   - Utilize const na declaração dos métodos que não alteram atributos

```
struct Atleta
{
    int acertos;
    int tentativas;
    float percentual;
};

void calcular(Atleta& atl)
{
    if (atl.tentativas != 0)
        atl.percentual = 100.0f * atl.acertos / atl.tentativas;
    else
        atl.percentual = 0;
}

void exibir(const Atleta& atl)
{
    cout << "Acertos: " << atl.acertos << "  ";
    cout << " Tentativas: " << atl.tentativas << "  ";
    cout << " Percentual: " << atl.percentual << "\n";
}

Atleta& acumular(Atleta& soma, const Atleta& atl)
{
    soma.tentativas += atl.tentativas;
    soma.acertos += atl.acertos;
    calcular(soma);
    return soma;
}
```

2. Teste a classe Atleta criando um programa que realize operações com os 3 atletas abaixo e com um atleta especial chamado time, que deve acumular o total de acertos e tentativas de todos os atletas.

```
Atleta rick { 13, 14 };
Atleta john { 10, 16 };
Atleta mark { 7, 9 };
Atleta time;
```