CIÊNCIA DA COMPUTAÇÃO
PROGRAMAÇÃO ORIENTADA A OBJETOS
PROF. JUDSON SANTIAGO

LABORATÓRIO 13

ORGANIZAÇÃO DE CÓDIGO

EXERCÍCIOS DE REVISÃO

VOCÊ DEVE ACOMPANHAR PARA REVISAR OS CONCEITOS IMPORTANTES

1. Incluir um arquivo de código fonte (.cpp) em outro é um erro comum para iniciantes na linguagem C++. Veja o que acontece se tentarmos compilar o projeto abaixo em que `Jogo.cpp` está sendo incluído em `Principal.cpp`.

Jogo.h:

```
#include <string>
using std::string;

class Jogo
{
private:
    string nome_;
    float preco_;
    int horas_;
public:
    Jogo(string nome, float preco, int horas);
    void Exibir();
};
```

Jogo.cpp:

```
#include <iostream>
#include "Jogo.h"

Jogo::Jogo(string nome, float preco, int horas)
    : nome_(nome), preco_(preco), horas_(horas) { }

void Jogo::Exibir()
{
    std::cout << nome_ << " R$" << preco_ << " " << horas_ << "h\n";
}
```

Principal.cpp:

```
#include "Jogo.cpp"

int main()
{
    Jogo gow { "God of War", 100.0f, 40 };
    gow.Exibir();
}
```

2. Como a declaração de uma classe normalmente fica em um arquivo de inclusão (.h) e a definição da classe fica em outro arquivo, um arquivo de código fonte (.cpp), é comum o programador introduzir inconsistências entre a declaração e a definição ao longo do tempo, a medida que atualiza e expande o código. Esse é um erro comum que não gera erros de compilação, mas gera problemas na ligação dos arquivos objeto.

   O código abaixo possui um exemplo de inconsistência. Inclua-o em um projeto e tente compilá-lo. Verifique a mensagem de erro. Em seguida corrija o programa para que ele compile sem erros.

   Mensagem.h:

```
#include <string>
using std::string;

class Msg
{
private:
    string texto_;
public:
    Msg();

    void Exibir();
    void Exibir(int n);
};
```

   Mensagem.cpp:

```
#include <iostream>
#include "Mensagem.h"
using std::cout;

Msg::Msg() : texto_("Vazia")
{
}

void Msg::Exibir()
{
    cout << texto_ << "\n";
}
```

   Programa.cpp:

```
#include "Mensagem.h"

int main()
{
    Msg msg;
    msg.Exibir();
    msg.Exibir(2);
}
```

   Dica: as mensagens do ligador normalmente são mais difíceis de entender, por isso é importante estar familiarizado com os erros mais comuns para não perder muito tempo na correção desse tipo de erro.

EXERCÍCIOS DE FIXAÇÃO

VOCÊ DEVE FAZER OS EXERCÍCIOS PARA FIXAR O CONTEÚDO

1. Veja o que acontece na compilação quando a definição de uma função é feita no .h e este arquivo é incluído em dois (ou mais) arquivos .cpp. O exemplo abaixo explora essa situação.

Exibe.h:

```
#include <iostream>
using std::cout;

void Exibir(int num)
{
   cout << num << "\n";
}
```

Exibe.cpp:

```
#include "Exibe.h"

// o conteúdo do arquivo não importa
```

Outro.cpp:

```
#include "Exibe.h"

int main()
{
    Exibir(42);
}
```

Depois modifique o exemplo, deixando apenas o protótipo da função no arquivo .h e colocando a definição da função no arquivo Exibe.cpp. Verifique se nesta nova organização ainda existem erros.

Dica: observe que a biblioteca `iostream`, bem como a instrução `using` não precisam ficar no .h pois a biblioteca não é necessária no protótipo da função, ela só é necessária na definição da função, local em que será usado o `cout`.

2. O programa abaixo contém as classes Palavra, sendo uma palavra composta por 20 caracteres, e Texto, representado por 10 palavras. Ou seja, a classe Texto precisa conhecer a classe Palavra. Por essa razão foi feita a inclusão de "Palavra.h" em "Texto.h".

Palavra.h:

```
class Palavra
{
private:
    char palavra_[20];

public:
    Palavra() {};  // implementação não importa
    ~Palavra() {};
};
```

Texto.h:

```
#include "Palavra.h"

class Texto
{
private:
    Palavra texto_[10];

public:
    Texto() {};  // implementação não importa
    ~Texto() {};
};
```

Se o programa principal incluir ambos os arquivos .h teremos uma declaração duplicada da classe Palavra, pois Texto.h já inclui Palavra.h.

Livro.cpp:

```
#include "Palavra.h"
#include "Texto.h"

int main()
{
    Palavra palavra;
    Texto texto;
}
```

Abaixo é possível ver que as mensagens de erro retornadas pelo Visual Studio não indicam diretamente que o erro é devido a uma declaração duplicada.

```
C2011: 'Palavra': 'class' type redefinition
C2079: 'Texto::texto' uses undefined class 'Palavra'
C2079: 'palavra' uses undefined class 'Palavra'
```

Teste o programa e corrija-o usando:

a) *pragma once*
b) *header guards*

3. É comum a declaração de uma classe incluir outros arquivos .h, como acontece com a classe Peixe abaixo que necessita da classe `string`. Porém inclusões devem ser feitas preferencialmente em arquivos .cpp. Muitas vezes os programadores acabam fazendo inclusões desnecessárias em arquivos .h, como é o caso da inclusão da biblioteca `iostream` no arquivo Peixe.h:

`Peixe.h`:

```
#include <iostream>
#include <string>
using std::cout;
using std::string;

class Peixe
{
private:
    string nome_;
    float peso_;
    unsigned tam_;

public:
    Peixe();
    ~Peixe();

    void Exibir();
};
```

Observe que o método Exibir da classe Peixe provavelmente vai precisar da biblioteca iostream e do cout, porém eles não são necessários na declaração da classe, apenas na implementação do método. Complete o programa, removendo as inclusões desnecessárias do arquivo Peixe.h, e adicionando o código e as inclusões apropriadas ao arquivo Peixe.cpp.

`Peixe.cpp`:

```
#include "Peixe.h"

Peixe::Peixe()
{
}

Peixe::~Peixe()
{
}

void Peixe::Exibir()
{
}
```

`Pescaria.cpp`:

```
#include "Peixe.h"

int main()
{
    Peixe peixe;
    peixe.Exibir();
}
```