CIÊNCIA DA COMPUTAÇÃO
PROGRAMAÇÃO ORIENTADA A OBJETOS
PROF. JUDSON SANTIAGO

LABORATÓRIO 14

NAMESPACES

EXERCÍCIOS DE REVISÃO

VOCÊ DEVE ACOMPANHAR PARA REVISAR OS CONCEITOS IMPORTANTES

1. O código abaixo representa um programa C++ típico. Ele é composto por vários arquivos de inclusão (.h) e de código fonte (.cpp), faz uso de namespaces e explora várias das situações estudadas:

   - Namespaces aninhados
   - Declarações using
   - Diretivas using
   - Apelidos

   Observe o uso de namespaces no programa.

   Window.h:

```
namespace Volt
{
    namespace Core
    {
        class Window
        {
        private:
            int color;
            int width;
            int height;

        public:
            Window();
            void Size(int width, int height);
            void Color(int color);
            void Show();
        };
    }
}
```

Input.h:

```
namespace Volt
{
    namespace Input
    {
        struct Mouse
        {
            int x;
            int y;
        };
    }
}
```

Graphics.h:

```
#include "Input.h"

namespace Volt
{
    namespace Graphics
    {
        using namespace Input;
        void Paint(Mouse pos);
    }
}
```

Game.cpp:

```
#include "Window.h"
#include "Input.h"
#include "Graphics.h"

namespace Engine = Volt::Core;
using namespace Volt::Input;
using Volt::Graphics::Paint;

int main()
{
    Engine::Window window;
    window.Size(1920, 1080);
    window.Color(240);
    window.Show();

    Mouse mouse { 10,10 };
    Paint(mouse);
}
```

Conclua o programa construindo Window.cpp e Graphics.cpp com a implementação da classe **Window** e da função **Paint** de forma a obter a seguinte saída:

```
Window (1920 x 1080) Color (240)
Mouse (10,10)
```

EXERCÍCIOS DE FIXAÇÃO

VOCÊ DEVE FAZER OS EXERCÍCIOS PARA FIXAR O CONTEÚDO

1. No programa abaixo substitua o uso de diretivas using por declarações using.

```
#include <iostream>
#include <string>
using namespace std;

namespace Cadastro
{
    struct Pessoal
    {
        string nome;
        int idade;
    };
}

int main()
{
    using namespace Cadastro;

    Pessoal novo;
    cout << "Nome: ";
    cin >> novo.nome;

    cout << "Idade: ";
    while (!(cin >> novo.idade))
    {
        cout << "Erro! Entre com uma idade.\n";
        cin.clear();
        while (cin.get() != '\n')
            continue;
    }

    cout << "Cadastrado: (" << novo.nome << ", " << novo.idade << ")\n";
}
```

2. Reescreva o programa da questão anterior eliminando completamente as declarações e diretivas using. Avalie os prós e contras de cada abordagem com relação a facilidade de codificação e clareza do código.

```
#include <iostream>
#include <string>
using namespace std;

namespace Cadastro
{
    struct Pessoal {
        string nome;
        int idade;
    };
}

int main()
{
    using namespace Cadastro;
```

3. No código abaixo é possível trocar os nomes qualificados das variáveis por nomes não qualificados? Ou seja, é possível remover todos os operadores de escopo (::) pelo uso de declarações using ou diretivas using? Tente remover o máximo possível e depois analise que solução parece mais apropriada.

```
namespace Canvas
{
    int layer;
    int color;
}

namespace Image
{
    int layer;
    int width;
    int height;
}

int main()
{
    Canvas::layer = 0;
    Canvas::color = 0;

    Image::layer  = 0;
    Image::width  = 0;
    Image::height = 0;
}
```

4. Execute o programa abaixo e tente entender o porquê do seu resultado.

```
#include <iostream>
using namespace std;

void Inverte();

namespace N1
{
    int x = 1;
}

namespace N2
{
    int x = 2;
}

int main()
{
    using namespace N1;
    cout << x << endl;
    {
        int x = 4;
        cout << x << ", " << N1::x << ", " << N2::x << endl;
    }
    using N2::x;
    cout << x << endl << endl;

    Inverte();
}
```

```
void Inverte()
{
    using namespace N2;
    cout << x << endl;
    {
        int x = 4;
        cout << x << ", " << N1::x << ", " << N2::x << endl;
    }
    using N1::x;
    cout << x << endl << endl;

}
```

5. O exemplo abaixo mostra a criação de um namespace **sem nome**. Criando um namespace sem nome, faz com que não seja possível usar declarações ou diretivas using com ele. Só é possível utilizar os nomes dentro desse namespace no próprio arquivo em que ele está declarado, ou seja, os nomes declarados nesse namespace possuem ligação interna, ao contrário dos nomes declarados em namespaces normais que possuem ligação externa. Isso significa que declarar variáveis em um namespace sem nome é equivalente a declarar estas variáveis com static.

SemNome.cpp:

```
void Mostrar();

namespace
{
    int frio = 10;
    int quente = 38;
}

static int gelo = 0;
int brasa = 45;

int main()
{  Mostrar(); }
```

Mostrar.cpp:

```
#include <iostream>
using namespace std;

extern int brasa;
extern int gelo;
extern int frio;
extern int quente;

void Mostrar()
{
    cout << "Brasa: " << brasa << endl;
    // cout << "Gelo: " << gelo << endl;
    // cout << "Frio: " << frio << endl;
    // cout << "Quente: " << quente << endl;
}
```

Faça o teste para confirmar o que foi dito através do programa acima.

EXERCÍCIOS DE APRENDIZAGEM

VOCÊ DEVE ESCREVER PROGRAMAS PARA REALMENTE APRENDER

1. Com base no namespace definido abaixo, complete o programa criando mais dois arquivos, Vendas.cpp com a implementação da classe Vendas e Principal.cpp com uma função main que teste a classe Vendas.

Vendas.h:

```
namespace Loja
{
    const int Trimestres = 4;

    class Vendas
    {
    private:
        double vendas[Trimestres];
        double media;
        double max;
        double min;

    public:
        Vendas(const double vet[], int n);

        void Ler();
        void Mostrar();
    };
}
```

O construtor deve receber um vetor de números de tamanho qualquer e copiar os 4 primeiros elementos para o vetor interno da classe, calculando em seguida a média, o maior e menor valor do vetor. Se o vetor recebido possuir menos que 4 elementos, completar o vetor interno com zeros.

`Ler:` o método deve preencher o vetor interno com dados obtidos diretamente do usuário. Peça para ele digitar os valores de venda dos trimestres, parando quando ele digitar 4 valores ou digitar um valor negativo, o que ocorrer primeiro. Se o vetor não for preenchido com 4 valores, complete o vetor com zeros.

`Mostrar:` o método deve exibir todos os elementos do vetor, a sua média, o maior e o menor valor.