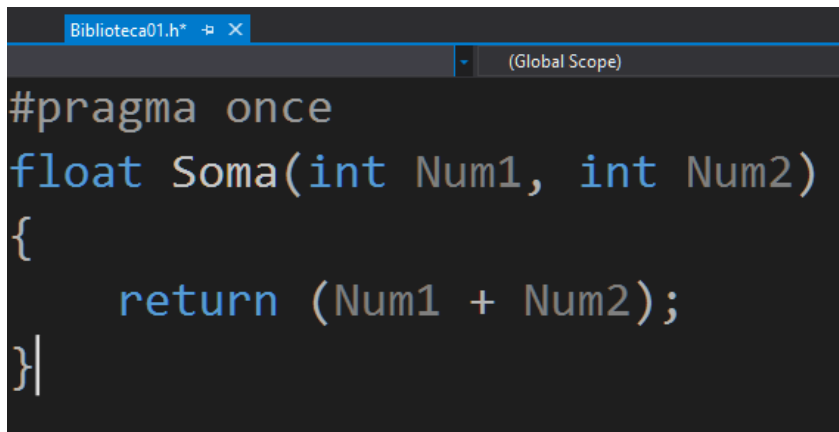


ENTENDO O CONCEITO DE NAMESPACES

Em relação a Namespaces confira:

Vamos imaginar que você tenha um biblioteca de nome **Biblioteca01.h** que tenha a seguinte função

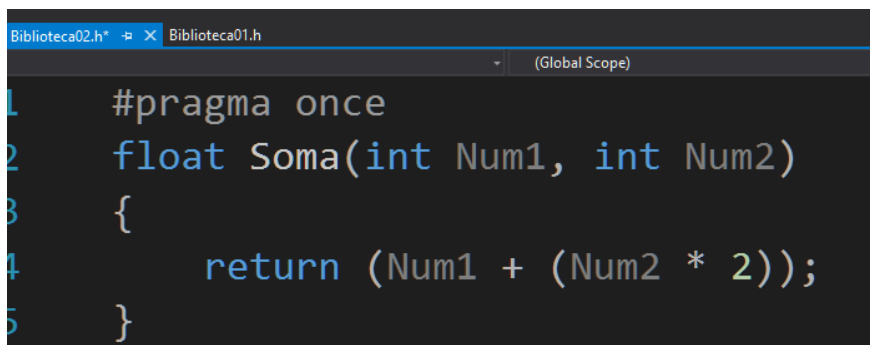
```
#pragma once
float Soma(int Num1, int Num2)
{
    return (Num1 + Num2);
}
```



```
#pragma once
float Soma(int Num1, int Num2)
{
    return (Num1 + Num2);
}
```

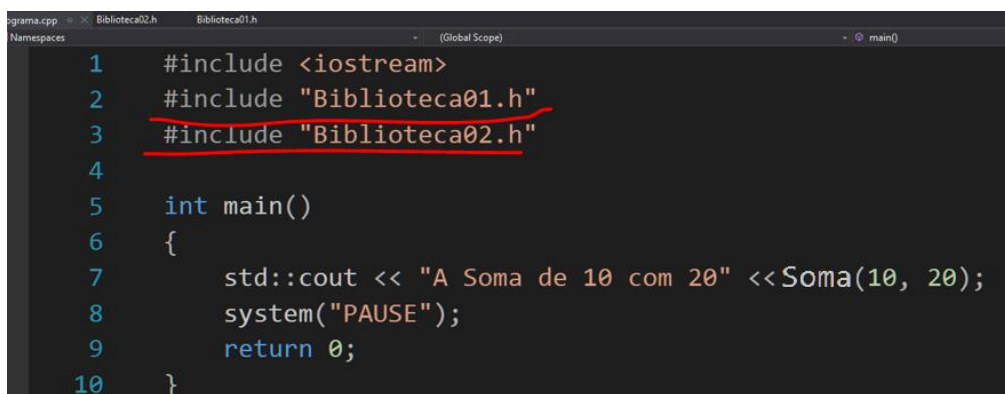
Vamos imaginar que você crie uma outra biblioteca de nome **Biblioteca02.h** que tenha a mesma função só que com retorno diferente

```
#pragma once
float Soma(int Num1, int Num2)
{
    return (Num1 + Num2);
}
```



```
1 #pragma once
2 float Soma(int Num1, int Num2)
3 {
4     return (Num1 + (Num2 * 2));
5 }
```

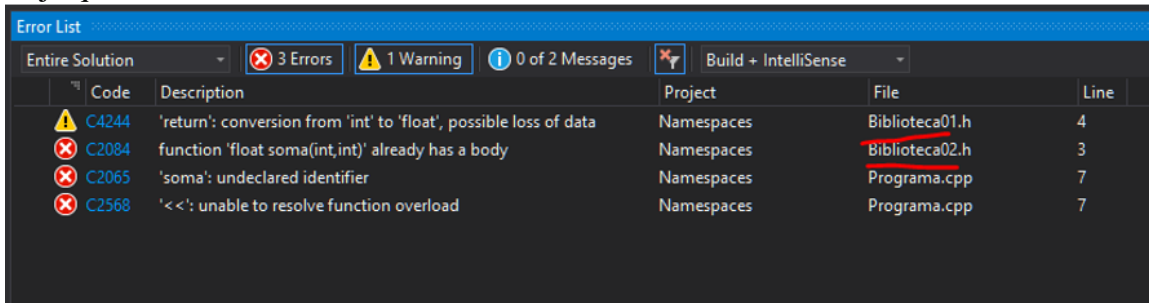
Agora você crie o seguinte programa abaixo de nome Programa.cpp e que usa(inclui) as duas bibliotecas



```
1 #include <iostream>
2 #include "Biblioteca01.h"
3 #include "Biblioteca02.h"
4
5 int main()
6 {
7     std::cout << "A Soma de 10 com 20" << Soma(10, 20);
8     system("PAUSE");
9     return 0;
10 }
```

Ao tentar executar o código ocorrerá erro pois a função de nome Soma está presente tanto da Biblioteca01.h quanto na Biblioteca02.h e não há como o compilador decidir qual será usada.. Isso pode ocorrer no decorrer do desenvolvimento e é interessante termos mecanismos para evitar estes conflitos. Poderia pensar em mudar o nome da variável Soma em alguma das bibliotecas, mas não seria uma saída elegante e eficiente.

Veja que ocorre erro



Veja que acima o visual studio indica que já existe uma função com mesma assinatura e que não foi possível resolver esta sobrecarga (overload) de funções.

Para resolver estes problemas de conflitos de nomes, etc foram criados os espaços de nomes ou Namespaces. O Objetivo destas Namespaces é criar uma região que seja identificada de forma que você possa referenciar ela correr o risco de conflitos de nomes como no exemplo acima

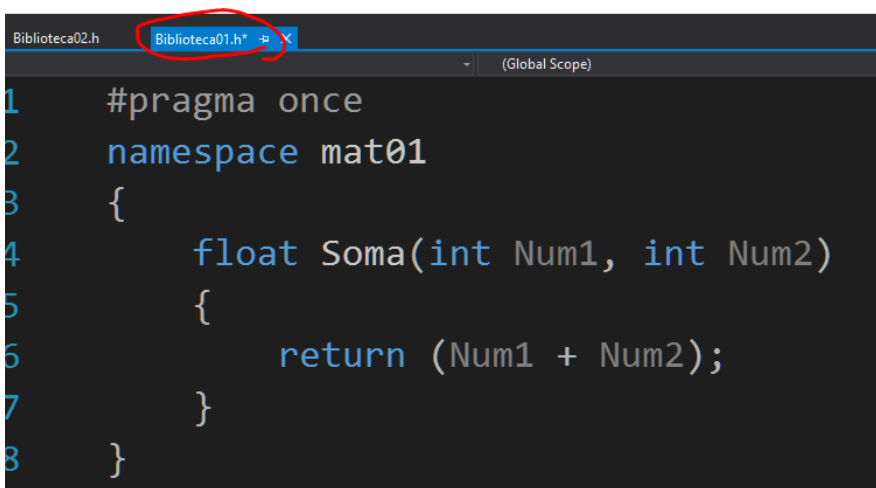
Como assim?

Vamos resolver o problema acima e ficará mais nítida esta utilização

Troque o Código da Biblioteca01.h por este

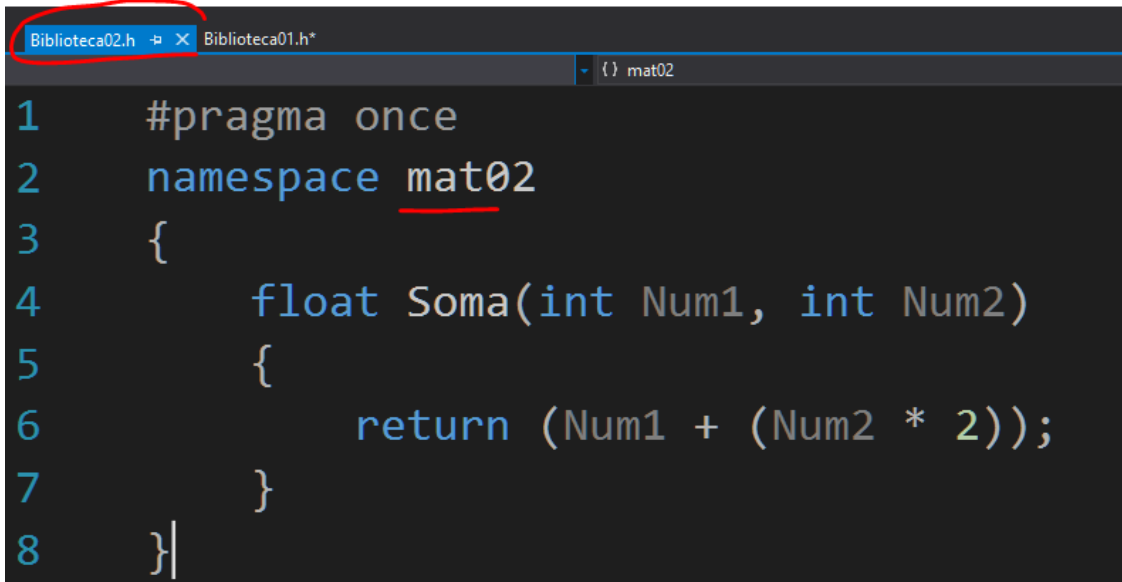
```
#pragma once
namespace mat01
{
    float Soma(int Num1, int Num2)
    {
        return (Num1 + Num2);
    }
}
```

Obs: Eu estou implementando no próprio arquivo.h para simplificar, mas no geral temos o Biblioteca01.h apenas com as assinaturas das funções e um Biblioteca01.cpp de mesmo nome com as implementações. Você vai aprender a fazer bibliotecas em breve no curso e vai entender esta observação.



Agora troque Troque o Código da Biblioteca02.h por este código que declara uma Namespace mat02

```
#pragma once
namespace mat02
{
    float Soma(int Num1, int Num2)
    {
        return (Num1 + (Num2 * 2));
    }
}
```



```
1  #pragma once
2  namespace mat02
3  {
4      float Soma(int Num1, int Num2)
5      {
6          return (Num1 + (Num2 * 2));
7      }
8  }
```

Feito isso em seu código Programa.cpp você pode chamar as funções Soma de cada Namespace e não haverá conflito

Basta usar NomeDaNameSpace::Soma neste caso ficará assim:

```
#include <iostream>
#include "Biblioteca01.h"
#include "Biblioteca02.h"

int main()
{
    std::cout << "A Soma de 10 com 20: " << mat01::Soma(10, 20) << "\n";
    std::cout << "A Soma de 10 com 20*2: " << mat02::Soma(10, 20) << "\n";
    system("PAUSE");
    return 0;
}
```


Veja que agora mat01::Soma(10, 20) está chamando a função Soma do namespace mat01 e que está na Biblioteca01.h

Da mesma forma mat02::Soma(10, 20) está chamando a função Soma do namespace mat02 e que está na Biblioteca02.h

Com isso o programa Executa sem problemas e você consegue verificar a importância das namespaces

```
#include <iostream>
#include "Biblioteca01.h"
#include "Biblioteca02.h"

int main()
{
    std::cout << "A Soma de 10 com 20: " << mat01::Soma(10, 20) << "\n";
    std::cout << "A Soma de 10 com 20*2: " << mat02::Soma(10, 20) <<
        "\n";
    system("PAUSE");
    return 0;
}
```



portanto **std::cout** indica que existe uma namespace dentro da biblioteca iostream e que dentro do namespace std existe a função cout. Se tiver outra biblioteca com a função cout basta usar o namespace desta biblioteca e não haverá conflito com a função cout da biblioteca std

Claro que existem várias outras coisas que podem ser feitas com namespaces como namespaces aninhados etc, mas isso é assunto para uma outra aula...

Abraço