

```
/**
    Curso de C++ Moderno
    Lista de exercícios Módulo 02: Compilação em detalhes

    @author Arthur Nunes de Paiva Santos Queiroz
    @version 03/2017
*/
```

*) Respostas dos exercícios do slide (p. 27)

1) O compilador utilizado gera 11846 linhas de código para o programa "hello".

Compilador:

gcc (Ubuntu 4.8.4-2ubuntu1~14.04.3) 4.8.4

2) Modo 1: Definir função "violateODR" como "inline" em "a.h"

```
#ifndef A_H__
#define A_H__

inline void violateODR(){

}

#endif
```

Modo 2: Definir função "violateODR" em um arquivo separado "a.cpp"

```
#include "a.h"

void violateODR(){

}
```

3) Uma vez que C++ suporta function overloading, o nome de uma função não pode ser utilizado como um ID único durante a linkagem, por isso ele modifica o nome da função adicionando a ele informação dos seus parâmetros (mangling).

A expressão "extern 'C'" faz com que o compilador não realize o mangling de forma que código em C possa "linkar" uma função C++.

Um outro caso de uso é um programa C++ precisar utilizar uma biblioteca C, o que pode ser feito declarando as a interface C como "extern 'C'".

Fonte: <http://stackoverflow.com/questions/1041866/in-c-source-what-is-the-effect-of-extern-c>

4) Se uma diretiva "using" for utilizada em um arquivo header, todos os arquivos que incluírem o header irão inadvertidamente importar o namespace, tornando-se susceptíveis a colisão de nomes.

Fonte: <http://stackoverflow.com/questions/2232496/is-it-wrong-to-use-c-using-keyword-in-a-header-file>

5) Namespaces anônimos permitem declarar identificadores únicos para uma translation unit. Isso permite utilizar identificadores

comuns sem correr risco de ocorrer colisões durante a linkagem. O uso de namespaces apresenta algumas diferenças em relação a utilização da keyword "static". Por exemplo, identificadores em namespaces anonymous possuem linkagem externa, enquanto identificadores declarados "static" possuem linkagem interna.

Não faz sentido utilizar namespaces anônimos em arquivos header porque os identificadores definidos no header serão interpretados como nomes únicos para a translation unit que o incluiu, de forma que definições presentes em outras translation units não serão encontradas durante a linkagem.

Fonte: <http://www.comeaucomputing.com/techtalk/#nostatic>

6) A macro interpreta qualquer elemento separado por vírgulas como um parâmetro, de forma que um template com mais de um tipo é interpretado como dois parâmetros pela macro (uma vez que os tipos do template são separados por vírgulas), gerando um erro de compilação.

Ex:

```
$ g++ exe_06.cpp -o exe_06.out
exe_06.cpp:21:37: error: macro "SETTER" passed 3 arguments, but
takes just 2
    SETTER(property, Tuple<int, double>);
                        ^
exe_06.cpp:21:2: error: 'SETTER' does not name a type
    SETTER(property, Tuple<int, double>);
    ^
```

7) p1.cpp: função inline

```
inline int ADD2(int A, int B) {
    return A+B;
}

void g(){
    int x = ADD2(1,10);
}

int main(){
    void g();
    return 0;
}
```

p2.cpp: enum

```
namespace Numbers{
    enum Number {
        One = 1,
        Two = 2,
    };
}
typedef Numbers::Number Number;

void k(int v){
    switch(v){
        case Numbers::One: ;
```

```

        case Numbers::Two: ;
    }
}

int main(){
    k(Numbers::One);
    return 0;
}

```

p3.cpp: template

```

#include <string>

template<typename T>
struct MyClass {
    MyClass(T){}
    T data_;
};

using namespace std;
typedef MyClass<int> MyClass_int;
typedef MyClass<string> MyClass_string;

void h(){
    MyClass_int m(10);
    MyClass_string n("foo");
}

int main(){
    h();
    return 0;
}

```

p4.cpp: valor const

```

const int SIZE = 100;

template<int size>
class MyArray{};

void f(){
    MyArray<SIZE> a;
}

int main(){
    f();
    return 0;
}

```

=====

*) Referências

<http://www.manpages.info/linux/gcc.1.html>

<http://stackoverflow.com/questions/5370539/what-is-the-meaning-of-lines->

starting-with-a-hash-sign-and-number-like-1-a-c

[1] <http://stackoverflow.com/questions/1041866/in-c-source-what-is-the-effect-of-extern-c>

<https://isocpp.org/wiki/faq/mixing-c-and-cpp>

<http://stackoverflow.com/questions/2232496/is-it-wrong-to-use-c-using-keyword-in-a-header-file>

<http://stackoverflow.com/questions/2448242/struct-with-template-variables-in-c>

<http://wiki.ros.org/CppStyleGuide>

<http://stackoverflow.com/questions/3081815/c-errors-while-compiling>

<http://stackoverflow.com/questions/19016253/g-undefined-reference-std>