In c: gcc -std=c89

In c++: g++ -std=c++98

E.g., g++ -std=c++98 -pedantic -Wall -o hello hello.cpp

Ending of code related to c++:

Implementation : .cpp, .cc, .cxx, .C

Header, hpp, hh, .hxx

Int main(void) in c 🡪 int main () in c++

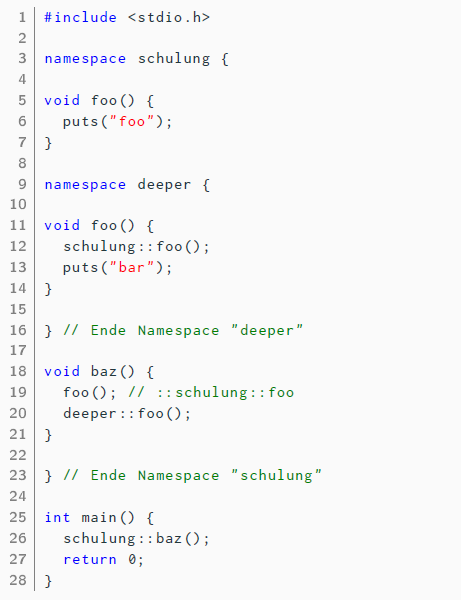
Namespaces:

Naming collision (ODR- one definition rule)

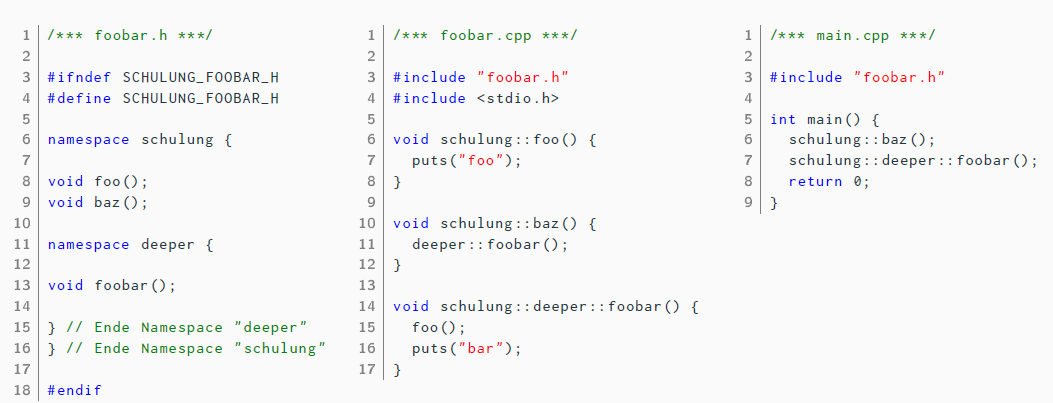
In c, we didn’t have namespace. Therefore, it was safe to type a long and safe name as struct EFSVorentwicklungsPunkt2D than a struct Punkt. But it c++ we have namespace.

How to open a namespace:

See these examples:



If you divide it into header and implementation file, it will look like this:



When you define namespaces, think of these 2 things:

1. Project name
2. Project structure.

Header in c++ standard library:

C++ takes the headers of the c standard library in the following pattern:

<xyz.h> 🡪 <cxyz>

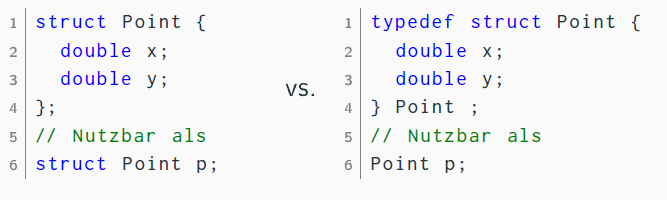
C libraries in c: #include <stdio.h> 🡪 puts, printf ( in c we didn’t had namespace)

C libraries in c++: #include <cstdio> 🡪 std::puts, std::printf

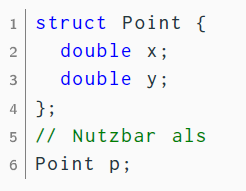
Note: the header of the style <xyz.h> is still supported in c++ for compatibility reasons. But don’t use.

Struct, union and enum in c++:

Remember in c, we had to use typedef for struct/union/enum XYZ with typedef, to use XYZ alone as the name of the datatype.



In c++ you don’t have to do that:



------------------------

Extra:

From the slide: ( I didn’t get it)

Durch Vorangestelltes :: kann ein

Name absolut angegeben und auf den

globalen Namespace verwiesen werden.

Das ist relativ selten nötig. See the example:

