Department I - C Plus Plus

# Modern and Lucid C++ Advanced for Professional Programmers

## Week 0 – Introduction

Prof. Peter Sommerlad / Thomas Corbat
Rapperswil, 21.02.2019
FS2019

Cevelop
Your C++ deserves it

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

IFS
INSTITUTE FOR
SOFTWARE

# Modul: C++ Advanced (M_CPlA)

- **4 Credits (2h Vorlesung / 2h Übungen pro Woche)**
- **Testate**
  - 3 Abgaben (Zeitpunkt wird bei Bedarf noch angepasst)
  - Keine Note
  - Zulassungsbedingung für die Prüfung
- **Prüfung**
  - 120 Minuten
  - Open Book (Ausnahme alte Prüfungen oder Abschriften davon)

# Administratives

- **Neu Wiki auf Gitlab: https://gitlab.dev.ifs.hsr.ch**
  - Account kann mit HSR-Mailadresse erstellt werden
- **Austausch von nicht öffentlichen Inhalten über Skripte-Server:**
  - https://skripte.hsr.ch/Informatik/Fachbereich/C++_Advanced/CplA/
  - \\hsr.ch\root\alg\skripte\Informatik\Fachbereich\C++_Advanced\CplA
- **Arbeitsumgebung (Vorläufig)**
  - Aktueller Compiler (GCC 8.2.0+): Windows MinGW von https://nuwen.net/mingw.html
  - Cevelop Version 1.11.0 (www.cevelop.com)
- **An Auffahrt kein Unterricht: Donnerstag 30.05.2019 (Letzte Semesterwoche)**

# Why C++ Advanced?

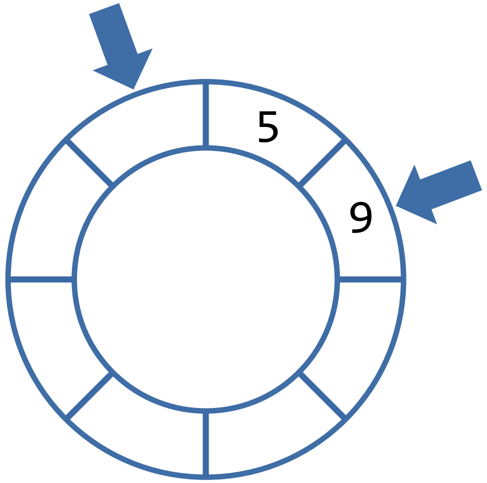
- **What is your motivation?**
- **What are your expectations?**

# Topics

- **Short refresh of C++ topics (Today)**
- **New Features in C++17**
- **Move semantics and type deduction**
- **Low level memory management**
- **Compile-time C++**
- **Multi-Threading / Memory Model**
- **Advanced Libraries/Development**
- **Whole Value Types**
- **Core Guidelines**
- **Build Systems**
- **C++ Standardization**

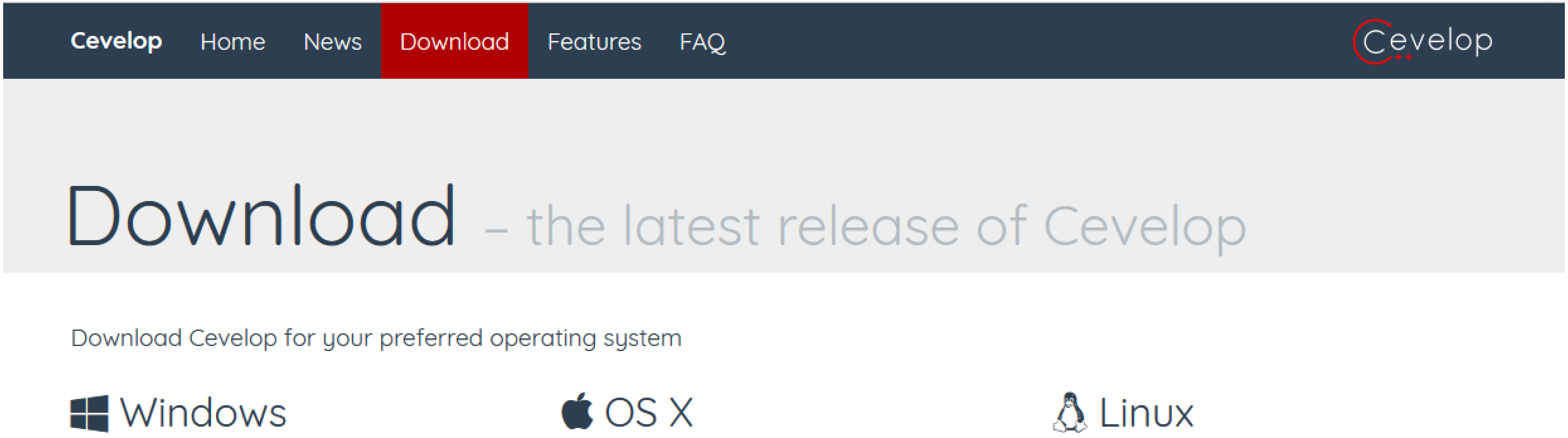
# Exercises

- **Typical tasks known from C++**
- **One specific exercise that will be part of several exercises**
  - Bounded Buffer
  - Start with a simple `std::array`-based version that will be extended during the semester
  - Parts of it will be Testat



# Cevelop 1.11.0 with Constificator and Stylechecker

- **Get Cevelop**
  - at https://www.cevelop.com/download
- **Install the additional plug-ins Constificator and Stylechecker**
  - Update site (already configured in Cevelop): https://www.cevelop.com/update/latest



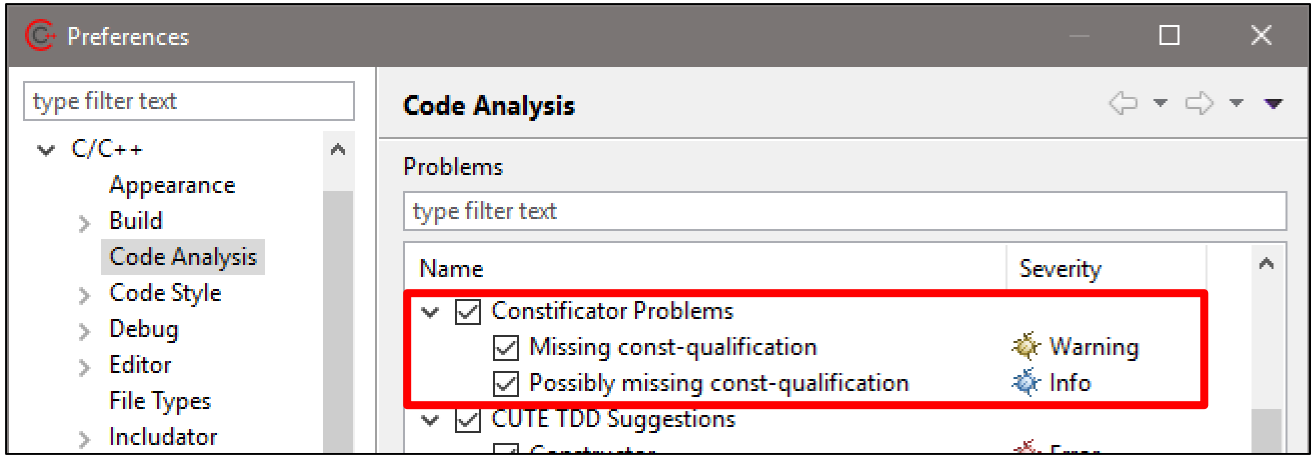
Download – the latest release of Cevelop

Download Cevelop for your preferred operating system

# Features of Cevelop (Constificator)

## Constificator

- If activated



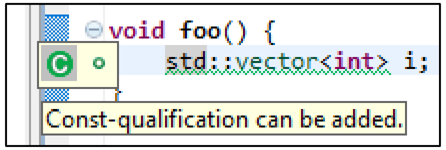
**Code Analysis**

Problems

type filter text

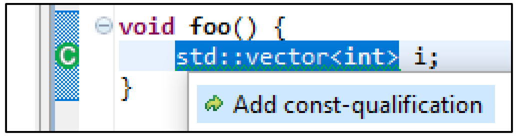
| Name | Severity |
|---|---|
| ☑ Constificator Problems | |
| ☑ Missing const-qualification | Warning |
| ☑ Possibly missing const-qualification | Info |
| ☑ CUTE TDD Suggestions | |
| ☐ Constructor | Error |

- Makes suggestions about missing const qualifiers

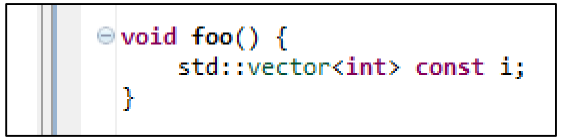


```
void foo() {
    std::vector<int> i;
```

Const-qualification can be added.

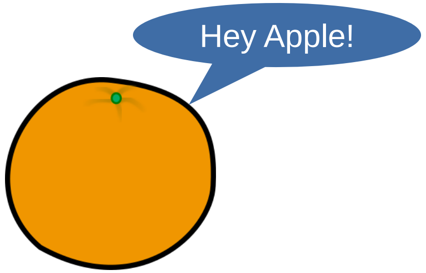


```
void foo() {
    std::vector<int> i;
}
```

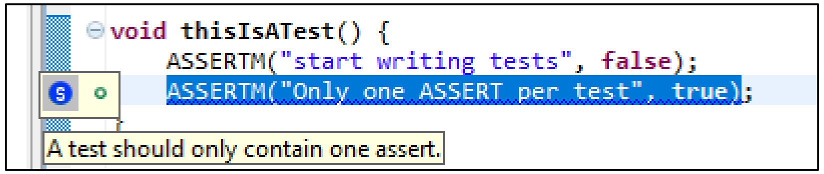


```
void foo() {
    std::vector<int> const i;
}
```

# What does Stylechecker do?



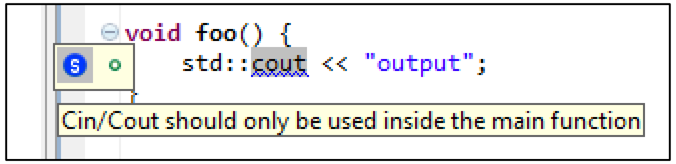
- **It annoys you in the first place!**
  - When you have multiple `ASSERT`s in your test cases



```
void thisIsATest() {
    ASSERTM("start writing tests", false);
    ASSERTM("Only one ASSERT per test", true);
```

A test should only contain one assert.

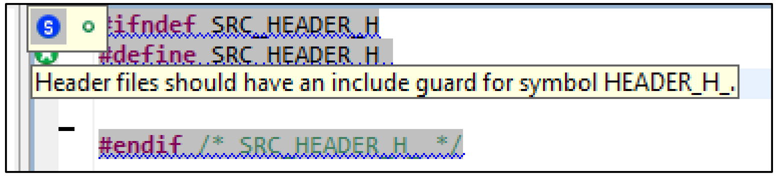
  - When you use `std::cin`/`std::cout` outside of the main function



```
void foo() {
    std::cout << "output";
```

Cin/Cout should only be used inside the main function

  - When there are issues with your include guard



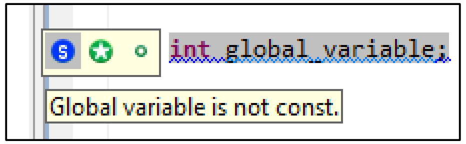
```
#ifndef SRC_HEADER_H
#define SRC_HEADER_H_

#endif /* SRC_HEADER_H_ */
```

Header files should have an include guard for symbol HEADER_H_.

# What does Stylechecker do?

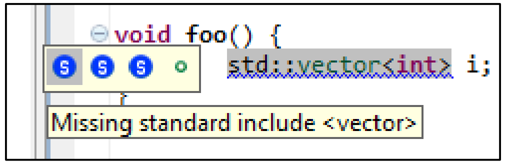
- **and reports...**
  - ... global non-const variables



```
int global_variable;
```

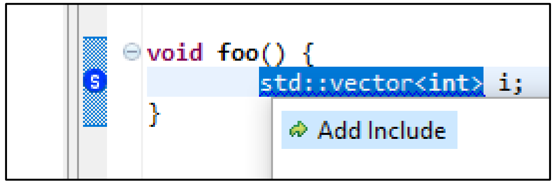
Global variable is not const.

  - ... missing includes of Standard Library headers and provides a resolution!



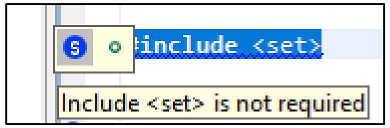
```
void foo() {
    std::vector<int> i;
```

Missing standard include <vector>



```
void foo() {
    std::vector<int> i;
}
```

  - ... unused includes of Standard Library headers

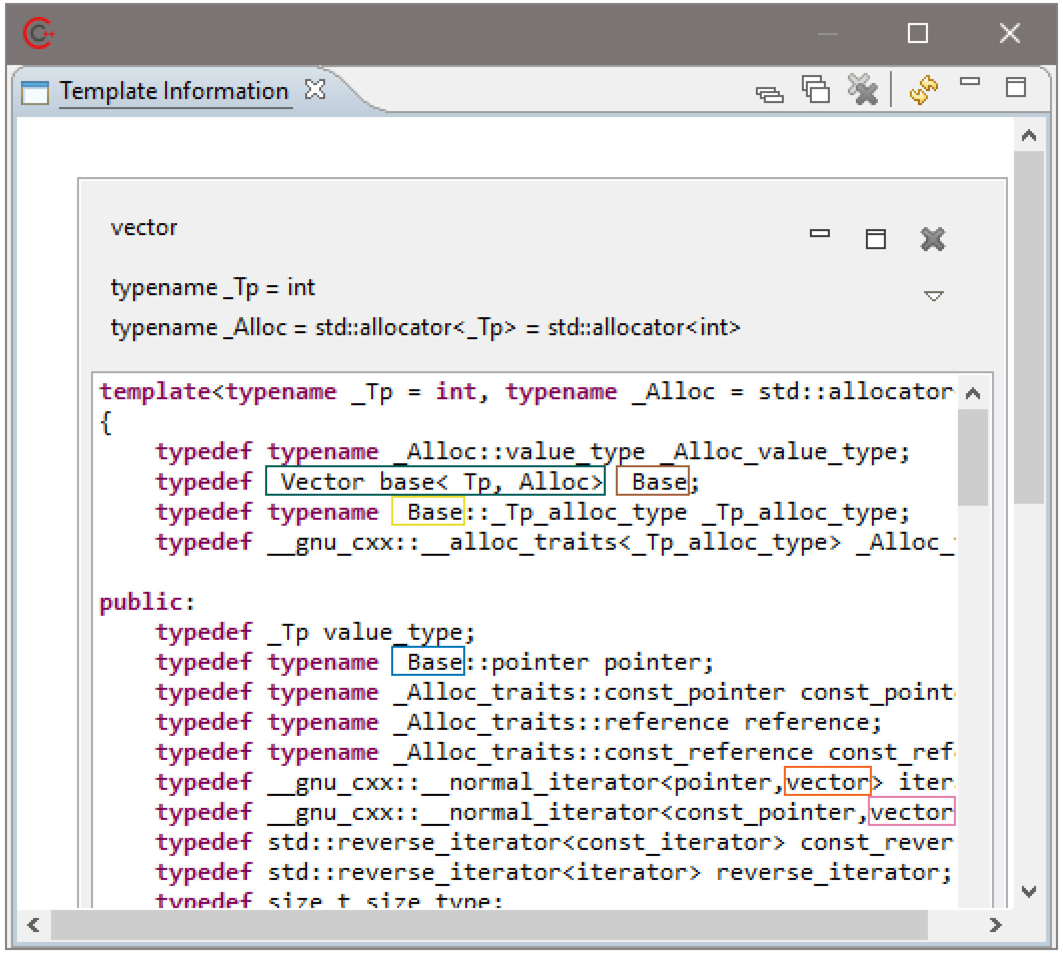


```
include <set>
```

Include <set> is not required

# (New) Features of Cevelop (Templator/Includator)

- **Template Instance Visualization**



vector

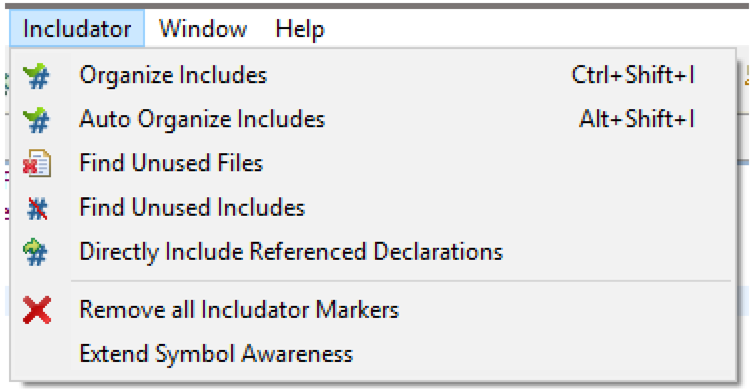
typename _Tp = int

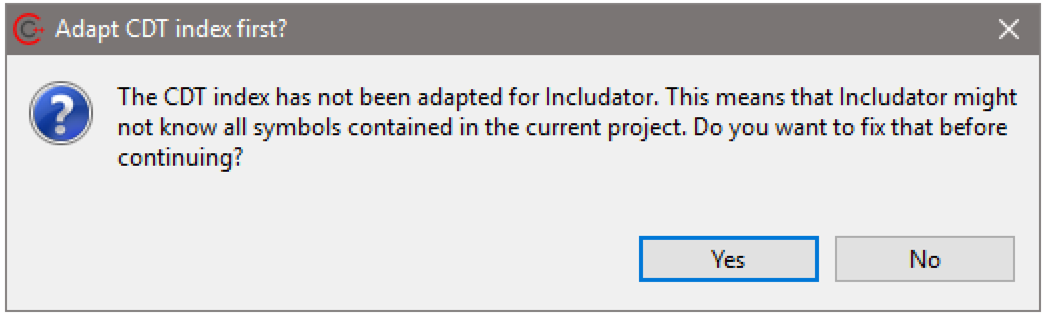
typename _Alloc = std::allocator<_Tp> = std::allocator<int>

```
template<typename _Tp = int, typename _Alloc = std::allocator
{
    typedef typename _Alloc::value_type _Alloc_value_type;
    typedef _Vector_base<_Tp, _Alloc> _Base;
    typedef typename _Base::_Tp_alloc_type _Tp_alloc_type;
    typedef __gnu_cxx::__alloc_traits<_Tp_alloc_type> _Alloc_

public:
    typedef _Tp value_type;
    typedef typename _Base::pointer pointer;
    typedef typename _Alloc_traits::const_pointer const_point
    typedef typename _Alloc_traits::reference reference;
    typedef typename _Alloc_traits::const_reference const_ref
    typedef __gnu_cxx::__normal_iterator<pointer,vector> iter
    typedef __gnu_cxx::__normal_iterator<const_pointer,vector
    typedef std::reverse_iterator<const_iterator> const_rever
    typedef std::reverse_iterator<iterator> reverse_iterator;
    typedef size_t size_type;
```

- **Include Organization**

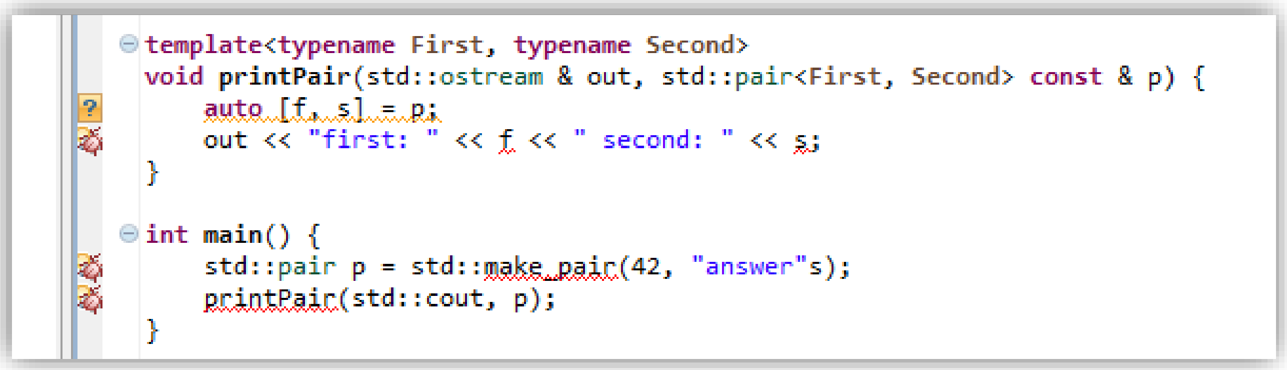

Includator
Window
Help
Organize Includes Ctrl+Shift+I
Auto Organize Includes Alt+Shift+I
Find Unused Files
Find Unused Includes
Directly Include Referenced Declarations
Remove all Includator Markers
Extend Symbol Awareness



The CDT index has not been adapted for Includator. This means that Includator might not know all symbols contained in the current project. Do you want to fix that before continuing?

# Resilience Required

- **You might encounter several bugs in Cevelop and its plug-ins (more than in CPI)**
  - We are using modern and sophisticated features of C++17, which are not completely supported yet



```
template<typename First, typename Second>
void printPair(std::ostream & out, std::pair<First, Second> const & p) {
    auto [f, s] = p;
    out << "first: " << f << " second: " << s;
}

int main() {
    std::pair p = std::make_pair(42, "answer"s);
    printPair(std::cout, p);
}
```

- **When in doubt ask your compiler**
- **When in more doubt ask several compilers**
  - http://melpon.org/wandbox
  - https://gcc.godbolt.org/