



<https://godbolt.org> by Matt  
Godbolt

Code analysieren und Diskussionen gewinnen

# Was ist das?

The screenshot displays the Compiler Explorer interface in a Google Chrome browser. The address bar shows the URL <https://godbolt.org>. The page features a 'COMPILER EXPLORER' logo and navigation buttons like 'Add...' and 'More...'. The main area is divided into two panes. The left pane, titled 'C++ source #1', contains the following C++ code:

```
1 void maxArray(double* x, double* y) {  
2     for (int i = 0; i < 65536; i++) {  
3         if (y[i] > x[i])  
4             x[i] = y[i];  
5     }  
6 }  
7
```

The right pane, titled 'x86-64 clang 7.0.0 (Editor #1, Compiler #1) C++', shows the generated assembly code for the same program. The assembly includes instructions like `xor eax, eax`, `movsd xmm0, qword ptr [rsi + 8*rax]`, `ucomisd xmm0, qword ptr [rdi + 8*rax]`, `jbe .LBB0_3`, `movsd qword ptr [rdi + 8*rax], xmm0`, `add rax, 1`, `cmp rax, 65536`, `jne .LBB0_1`, and `ret`. The assembly is color-coded to match the C++ source. At the bottom, the output section shows 'Output (0/0)' for 'x86-64 clang 7.0.0' with a compilation time of '- 403ms (13191B)'.

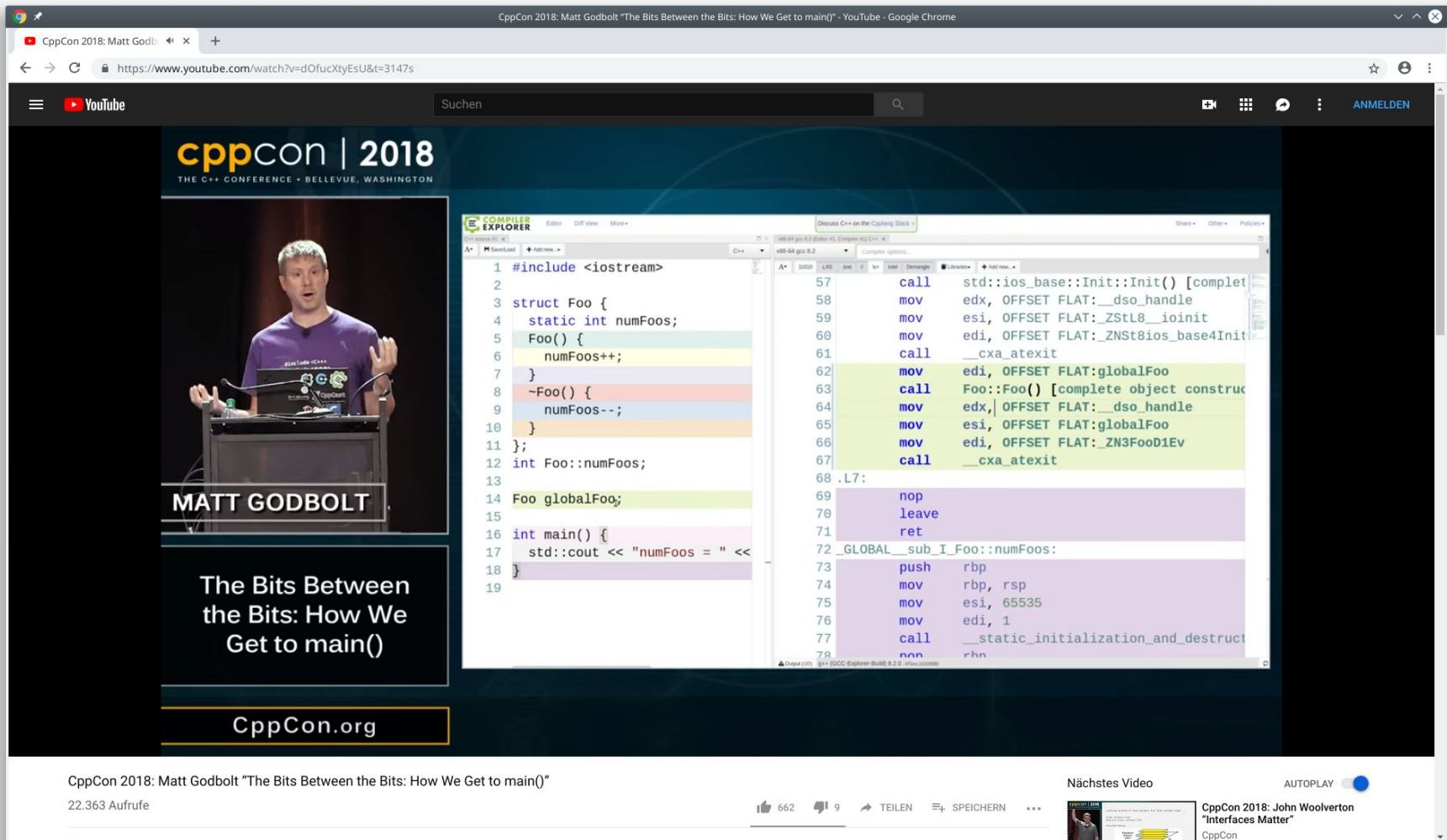
<https://godbolt.org/z/-MpVCV>



## Was ist das?

- Online Tool von Matt Godbolt
- Compiler Optimierungen aufzeigen
- Code schreiben, Assembler vergleichen
- Compiler und neue Features testen
- Code teilen und Wissen erweitern

# “The Bits Between the Bits: How We Get to main()”



CppCon 2018: Matt Godbolt "The Bits Between the Bits: How We Get to main()"

22.363 Aufrufe

662 9 TEILEN SPEICHERN

Nächstes Video

CppCon 2018: John Woolverton "Interfaces Matter"

```
1 #include <iostream>
2
3 struct Foo {
4     static int numFos;
5     Foo() {
6         numFos++;
7     }
8     ~Foo() {
9         numFos--;
10    };
11 };
12 int Foo::numFos;
13
14 Foo globalFoo;
15
16 int main() {
17     std::cout << "numFos = " <<
18 }
19
```

```
57 call std::ios_base::Init::Init() [complete]
58 mov     edx, OFFSET FLAT:__dso_handle
59 mov     esi, OFFSET FLAT:_ZStL8__ioinit
60 mov     edi, OFFSET FLAT:_ZNSt8ios_base4Init
61 call    __cxa_atexit
62 mov     edi, OFFSET FLAT:globalFoo
63 call    Foo::Foo() [complete object construct
64 mov     edx, OFFSET FLAT:__dso_handle
65 mov     esi, OFFSET FLAT:globalFoo
66 mov     edi, OFFSET FLAT:_ZN3FooD1Ev
67 call    __cxa_atexit
68 .L7:
69 nop
70 leave
71 ret
72 _GLOBAL__sub_I_Foo::numFos:
73 push    rbp
74 mov     rbp, rsp
75 mov     esi, 65535
76 mov     edi, 1
77 call    __static_initialization_and_destruct
78 nop
79 rbp
```

<https://www.youtube.com/watch?v=dOfucXtyEsU>



## Wofür brauche ich das?

- Compiler verstehen und vergleichen
- Neue Compiler Features demonstrieren
  - z.B. auf der CppCon genutzt
- Sprachen vergleichen (Rust <--> C++)
- Argumente gewinnen
  - “Memset ist schneller”, “STL ist langsam”



## Welche Sprachen gibt es auf [godbolt.org](https://godbolt.org)?

- C
- C++
- Rust
- D
- Go
- Haskell
- Swift
- Pascal
- Fortran
- Assembler
- ZIG
- ...



## C++ Compiler

- GCC 4.1, 4.4, 4.5, ..., 6, ..., 7.1, ..., 8, trunk
  - ARM GCC, MIPS GCC, MSP GCC
- Clang 3.0, 3.1, ..., 4, 5, 6, 7, trunk, Cppx (-Wlifetime)
- ICC 13, 16, 17, 18, 19
- MSVC 19 (unter Windows und Wine)
- Compiler für PowerPC
- ...

# C++ Bibliotheken

## Include libs

Boost	-	⬆	Brigand	-	⬆	Kvasir::mpl	-	⬆
cmcstl2	-	⬆	ctbignum	-	⬆	GSL	-	⬆
expected-lite	-	⬆	nlohmann::json	-	⬆	xtl	-	⬆
xsimd	-	⬆	xtensor	-	⬆	Abseil	-	⬆
Blaze	-	⬆	CTRE	-	⬆	Eigen	-	⬆
Google Benchmark	-	⬆	range-v3	-	⬆	dlib	-	⬆
libGuarded	-	⬆	cppcoro	-	⬆	{fmt}	-	⬆
HFSM	-	⬆	GLM	-	⬆	LLVM	-	⬆
Catch2	-	⬆	Doctest	-	⬆	EASTL	-	⬆
VCL	-	⬆	outcome	-	⬆	CNL	-	⬆

Libraries ▾ + Add new... ▾ ⚙ Add tool... ▾

```
22 }  
23 }  
24
```



# Features | Mehrere Editoren

The screenshot displays the Compiler Explorer web application in Google Chrome. The interface is split into two main panels, each containing a source code editor and an assembly output window.

**Left Panel (C++ source #1):**

- Source Code:** A C++ program that includes `<vector>` and defines a static function `sum` that iterates over a vector and returns the sum. The `main` function calls `sum({1,2,3})`.
- Compiler:** x86-64 clang 5.0.0 (Editor #1, Compiler #1) C++.
- Options:** -O2 -std=c++14.
- Assembly Output:** Shows the assembly for the `main` function, including instructions for pushing `rax`, moving `edi` to 12, calling `operator new` and `operator delete`, and returning.
- Output:** Output (0/0) x86-64 clang 5.0.0 - 804ms (229655B).

**Right Panel (C++ source #2):**

- Source Code:** A C++ program that includes `<vector>` and `<numeric>`, and uses `std::accumulate` to calculate the sum of a vector. The `main` function calls `sum({1,2,3})`.
- Compiler:** x86-64 clang 5.0.0 (Editor #2, Compiler #2) C++.
- Options:** -O2 -std=c++14.
- Assembly Output:** Shows the assembly for the `main` function, which is identical to the left panel's output.
- Output:** Output (0/0) x86-64 clang 5.0.0 - 911ms (249324B).

<https://godbolt.org/z/lqZ4LJ>

# Features | Diff View

The screenshot displays the Compiler Explorer interface in Google Chrome, showing a C++ source file and its assembly output for two compilers: x86-64 clang (trunk) and x86-64 gcc (trunk).

**Source File (C++ source #1):**

```
1 int testFunction(int* input, int length) {  
2     int sum = 0;  
3     for (int i = 0; i < length; ++i) {  
4         sum += input[i];  
5     }  
6     return sum;  
7 }  
8
```

**Assembly Output (x86-64 clang (trunk) vs x86-64 gcc (trunk)):**

Left: x86-64 clang (trunk) -O1	Right: x86-64 gcc (trunk) -O1
1 - testFunction(int*, in	1 + testFunction(int*, in
2 test esi, esi	2 test esi, esi
3 - jle .LBB0_1	3 + jle .L4
4 - mov ecx, esi	4 + mov rdx, rdi
5 - xor edx, edx	5 + lea eax, [rsi-1]
6 - xor eax, eax	6 + lea rcx, [rdi+4+rax
7 - .LBB0_4: # =>This Inn	7 + mov eax, 0
8 - add eax, dword ptr	8 + .L3:
9 - add rdx, 1	9 + add eax, DWORD PTR
10 - cmp rcx, rdx	10 + add rdx, 4
11 - jne .LBB0_4	11 + cmp rdx, rcx
	12 + jne .L3
12 ret	13 ret
13 - .LBB0_1:	14 + .L4:
14 - xor eax, eax	15 + mov eax, 0
15 ret	16 ret

**Compiler Explorer Interface Details:**

- Top Bar:** Compiler Explorer logo, Add... More... buttons, and Share Other Policies dropdowns.
- Source File:** C++ source #1, Save/Load, Add new... buttons, and C++ dropdown.
- Assembly Output:** x86-64 clang (trunk) (Editor #1, Compiler #1) C++ and x86-64 gcc (trunk) (Editor #1, Compiler #2) C++.
- Compiler Selection:** x86-64 clang (trunk) and x86-64 gcc (trunk) with -O1 optimization level.
- Assembly Output:** 11010, .LX0, lib.f, .text, //, and assembly code for testFunction(int\*, int):.
- Output:** Output (0/0) x86-64 clang (trunk) - 231ms (12047B) and Output (0/0) x86-64 gcc (trunk) - 230ms (4615B).

# Features | Conformance View

Compiler Explorer - Google Chrome

https://godbolt.org

COMPILER EXPLORER

Add... More

Share Other Policies

C++ source #1 x

Save/Load Add new... C++

```
1 template<typename... T>
2 auto sum(T... s){
3     return (... + s);
4 }
5
6 int main(int argc, char** /*argv*/) {
7     return sum(argc);
8 }
```

x86-64 clang (trunk) (Editor #1, Compiler #1) C++ x

x86-64 clang (trunk) Compiler options...

11010 .LX0: lib.f: .text // \s+ Intel Demangle

Libraries Add new... Add tool...

```
1 main: # @main
2 push rbp
3 mov rbp, rsp
4 sub rsp, 16
5 mov dword ptr [rbp - 4], 0
6 mov dword ptr [rbp - 8], edi
7 mov qword ptr [rbp - 16], rsi
8 mov edi, dword ptr [rbp - 8]
9 call auto sum<int>(int)
10 add rsp, 16
11 pop rbp
```

Output (0/4) x86-64 clang (trunk) - cached (12716B)

Conformance viewer (Editor #1) 5/10 x

Add compiler Libraries

x86-64 gcc (trunk)	✓	-std=c++17	↶ ↷
x86-64 gcc 8.2	✓	-std=c++17	↶ ↷
x86-64 gcc 7.1	✓	-std=c++17	↶ ↷
x86-64 gcc 6.1	✓	-std=c++17	↶ ↷
x86-64 gcc 5.5	✗	-std=c++1z	⌵ ↶ ↷

# Features | Konsole

The screenshot displays the Compiler Explorer interface in a Google Chrome browser window. The address bar shows the URL `https://godbolt.org`. The interface is divided into several sections:

- Top Bar:** Includes the Compiler Explorer logo, a search bar, and buttons for "Share", "Other", and "Policies".
- Source Code Editor (Left):** Displays C++ source code for "C++ source #1". The code defines a struct `A` with members `a`, `b`, and `c`, and a function `some_function` that takes an integer `num` and returns `num * a.b`. The function body contains a struct initialization `A a{.a = num, .b = num, .c = num};`.
- Compiler Selection (Top Right):** Shows "x86-64 gcc 8.2" as the selected compiler. The compiler options are set to `-std=c++17 -Wall -Wextra -pedantic`.
- Assembly Output (Right):** Displays the assembly code generated by the compiler for the `some_function` function. The assembly includes instructions for pushing the base pointer, moving the stack pointer, and performing arithmetic operations.
- Output Panel (Bottom):** Shows the compiler's output, including warnings. The warnings indicate that C++ designated initializers are only available with `-std=c++2a` or `-std=gnu++2a` when using `-Wpedantic`. The output also shows the compiler's return code as 0.

```
<source>: In function 'int some_function(int)':
<source>:6:9: warning: C++ designated initializers only available with -std=c++2a or -std=gnu++2a [-Wpedantic]
    A a{.a = num, .b = num, .c = num};
        ^
<source>:6:19: warning: C++ designated initializers only available with -std=c++2a or -std=gnu++2a [-Wpedantic]
    A a{.a = num, .b = num, .c = num};
                ^
<source>:6:29: warning: C++ designated initializers only available with -std=c++2a or -std=gnu++2a [-Wpedantic]
    A a{.a = num, .b = num, .c = num};
                            ^

Compiler returned: 0
```



The screenshot displays the Compiler Explorer web application. The top navigation bar includes the Compiler Explorer logo, a search bar, and buttons for 'Share', 'Other', and 'Policies'. The main interface is divided into three panes:

- Source Code Pane (Left):** Shows the C++ source code for a function `maxArray`. The code is as follows:
 

```

1 void maxArray(double* x, double* y) {
2     for (int i = 0; i < 65536; i++) {
3         if (y[i] > x[i])
4             x[i] = y[i];
5     }
6 }
7
8 
```
- Assembly Pane (Bottom Left):** Displays the assembly code generated by the compiler (x86-64 gcc 8.2). The assembly is as follows:
 

```

1 maxArray(double*, double*):
2     push    rbp
3     mov     rbp, rsp
4     mov     QWORD PTR [rbp-24], rdi
5     mov     QWORD PTR [rbp-32], rsi
6     mov     DWORD PTR [rbp-4], 0
7 .L5:
8     cmp     DWORD PTR [rbp-4], 65535
9     jg      .L7
10    mov     eax, DWORD PTR [rbp-4]

```
- Graph View Pane (Right):** Shows a control flow graph (CFG) for the assembly code. The graph consists of several basic blocks connected by edges:
  - Block 1 (Top):** Contains the function prologue: `push rbp, mov rbp, rsp, mov QWORD PTR [rbp-24], rdi, mov QWORD PTR [rbp-32], rsi, mov DWORD PTR [rbp-4], 0`.
  - Block 2 (Middle):** Labeled `.L5:`, containing `cmp DWORD PTR [rbp-4], 65535, jg .L7`. It has a green edge to Block 3 and a red edge to Block 4.
  - Block 3 (Bottom Left):** Labeled `.L7:`, containing `nop, pop rbp, ret`. It has a green edge back to Block 2.
  - Block 4 (Middle Right):** Labeled `.L5:@9`, containing instructions for incrementing the loop counter: `mov eax, DWORD PTR [rbp-4], cdqe, lea rdx, [0+rax*8], mov rax, QWORD PTR [rbp-32], mov rax, rdx, movsd xmm0, QWORD PTR [rax], movsd xmm1, QWORD PTR [rbp-4], cdqe, lea rdx, [0+rax*8], mov rax, QWORD PTR [rbp-24], add rax, rdx, movsd xmm0, QWORD PTR [rax], comisd xmm0, xmm1, jbe .L3`. It has a red edge back to Block 2 and a green edge to Block 5.
  - Block 5 (Bottom Middle):** Labeled `.L5:@23`, containing instructions for incrementing the loop counter: `mov cdqe, eax, DWORD PTR [rbp-4], lea rdx, [0+rax*8], mov rax, QWORD PTR [rbp-32], add rdx, rax, movsd xmm0, QWORD PTR [rbp-4], cdqe, lea rcx, [0+rax*8], mov rax, QWORD PTR [rbp-24], add rax, rcx, movsd xmm0, QWORD PTR [rdx], movsd QWORD PTR [rax], xmm0`. It has a green edge back to Block 2.
  - Block 6 (Bottom Left):** Labeled `.L3:`, containing `add jmp DWORD PTR [rbp-4], 1, .L5`. It has a green edge back to Block 2.

The bottom status bar shows the output of the compiler: `Output (0/0) x86-64 gcc 8.2 - 823ms (45848)`.

<https://qodbolt.org/z/02h5V7>

# Features | Tools: Clang AST

The screenshot displays the Compiler Explorer web application in Google Chrome. The interface is divided into several panes:

- Source Code Pane:** Contains C++ code for a function `maxArray` that iterates over an array and compares elements. The code is highlighted with syntax coloring.
- Compiler Options Pane:** Shows the selected compiler as `x86-64 clang 7.0.0` and a field for compiler options.
- Assembly Pane:** Displays the assembly output for the compiled code, showing instructions like `push rbp`, `mov rbp, rsp`, and `cmp dword ptr [rbp - 20],`.
- AST Pane:** Shows the Clang Abstract Syntax Tree (AST) for the source code. It is a tree structure representing the code's semantics, starting with `TranslationUnitDecl` and containing `FunctionDecl` for `maxArray`. The AST includes details about parameters, compound statements, for loops, and conditional statements.

The AST pane shows the following structure:

```
1 TranslationUnitDecl
2  \-FunctionDecl <line:2:1, line:7:1> line:2:6 maxArray 'void (double *, double *, int)'
3    \-ParmVarDecl <col:15, col:23> col:23 used x 'double *'
4      \-ParmVarDecl <col:26, col:34> col:34 used y 'double *'
5        \-CompoundStmt <col:37, line:7:1>
6          \-ForStmt <line:3:5, line:6:5>
7            \-DeclStmt <line:3:10, col:19>
8              \-VarDecl <col:10, col:18> col:14 used i 'int' cinit
9                \-IntegerLiteral <col:18> 'int' 0
10              \-<<<NULL>>>
11            \-BinaryOperator <col:21, col:25> 'bool' '<'
12              \-ImplicitCastExpr <col:21> 'int' <LValueToRValue>
13                \-DeclRefExpr <col:21> 'int' lvalue Var 0x558a8a4c5ef8 'i' 'int'
14                \-IntegerLiteral <col:25> 'int' 65536
15              \-UnaryOperator <col:32, col:33> 'int' postfix '++'
16                \-DeclRefExpr <col:32> 'int' lvalue Var 0x558a8a4c5ef8 'i' 'int'
17            \-CompoundStmt <col:37, line:6:5>
18              \-IfStmt <line:4:9, line:5:23>
19                \-<<<NULL>>>
20                \-<<<NULL>>>
21                \-BinaryOperator <line:4:13, col:23> 'bool' '>'
22                  \-ImplicitCastExpr <col:13, col:16> 'double' <LValueToRValue>
23                    \-ArraySubscriptExpr <col:13, col:16> 'double' lvalue
24                      \-ImplicitCastExpr <col:13> 'double *' <LValueToRValue>
25                        \-DeclRefExpr <col:13> 'double *' lvalue ParmVar 0x558a8a4c5ef8 'y' 'double *'
26                        \-ImplicitCastExpr <col:15> 'int' <LValueToRValue>
27                          \-DeclRefExpr <col:15> 'int' lvalue Var 0x558a8a4c5ef8 'x' 'int'
```

# Features | Opt View

The screenshot displays the Compiler Explorer web application in Google Chrome. The interface is divided into several sections:

- Header:** Includes the "Compiler Explorer" logo, navigation buttons ("Add...", "More"), and utility buttons ("Share", "Other", "Policies").
- Source Editor (C++ source #1):** Contains the following C++ code:

```
1 void maxArray(double* x, double* y) {  
2     auto fct = [&](int i) {  
3         if (y[i] > x[i]) x[i] = y[i];  
4     };  
5     for (int i = 0; i < 65535; i++)  
6         fct(i);  
7 }  
8 }
```
- Compiler Selection:** Shows "x86-64 clang 7.0.0" as the selected compiler, with a green checkmark and a "Compile" button.
- Options:** Includes checkboxes for "11010", ".LX0:", "lib.f:", ".text", "//", and "\s+", along with "Libraries" and "Add new..." buttons.
- Assembly Output (x86-64 clang 7.0.0 Opt Viewer):** Displays the optimized assembly code for the provided C++ source. A tooltip is visible over the assembly, stating: **Missed** - maxArray(double, double)::\$\_0::operator() not inlined into maxArray because it should never be inlined (cost=never).
- Footer:** Shows the output status as "Output (0/0)" and the compilation time as "x86-64 clang 7.0.0 - 254ms (23800B)".



## Wie geht's weiter?

- mit [godbolt.org](https://godbolt.org) spielen und austauschen
- lokale Instanz installieren
- andere Sprachen testen