

```
#include <iostream>
```

```
int main(){
```

```
    std::cout << "Hello World!" << std::endl;
```

```
    std::vector<int> myVec(10);
```

```
    std::iota(myVec.begin(), myVec.end(), 1);
```

```
    std::cout << "myVec: " << std::endl;
```

```
    for ( auto i: myVec ) std::cout << i << " ";
```

```
    std::cout << "\n\n";
```

```
    std::vector<int> myVec2(20);
```

```
    std::iota(myVec2.begin(), myVec2.end(), 1);
```

```
    std::cout << "myVec2: " << std::endl;
```

```
    for ( auto i: myVec2 ) std::cout << i << " ";
```

```
    std::cout << "\n\n";
```

```
    return 0;
```

# Die bekanntesten (Online-)Compiler im Vergleich

Rainer Grimm

Training, Coaching und  
Technologieberatung

[www.ModernesCpp.de](http://www.ModernesCpp.de)

# Compiler

C++11, C++14 und C++17

Compiler Flags

Offline to Online

Unter die Haube geschaut



πάντα ρεῖ



# Compiler

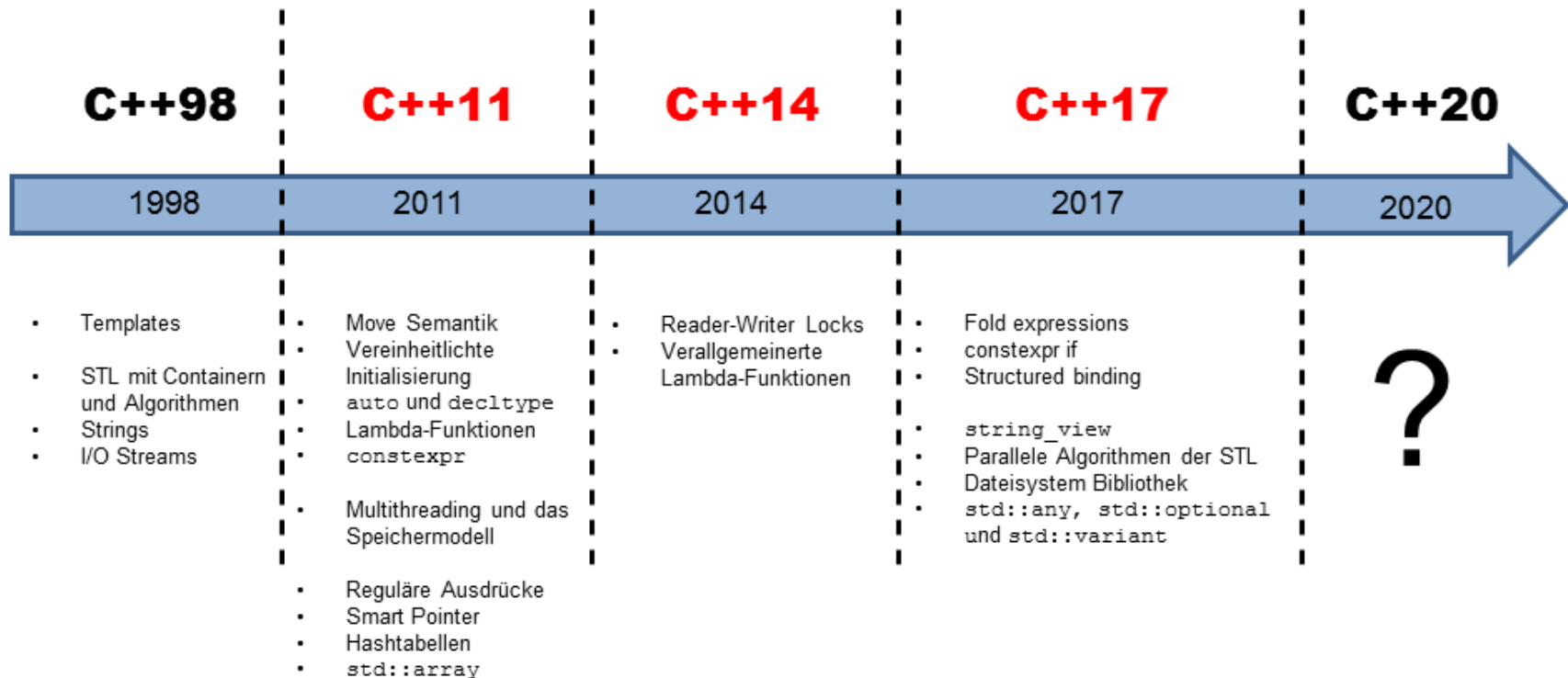
C++11, C++14 und C++17

Compiler Flags

Offline to Online

Unter die Haube geschaut

# Der aktuelle C++ Standard



# Standardunterstützung der großen Drei

C++ Compiler	C++11	C++14	C++17
GCC	4.8.1	5.0	7
Clang	3.3	3.4	5
MSVC	19.0	19.1	19.15

Als einzige Plattform unterstützt MSVC teilweise zum jetzigen Zeitpunkt die parallele STL.

# Die großen Drei – Die Details

Unterstützung der aktuellen C++ Standards: [cppreference.com](http://cppreference.com)

C++14 feature	Paper(s)	Version	GCC	Clang	MSVC	EDG ecpp	Intel C++	IBM XL C++	Sun/Oracle C++	Embarcadero C++ Builder	Cray	Portland Group (PGI)
Tweaked wording for contextual conversions	<a href="#">N3323</a>	c++14-lang	4.9	3.4	18.0*	4.9	16.0	13.1.2*	5.15			
Binary literals	<a href="#">N3472</a>	c++14-lang	4.3/4.9	2.9	19.0*	4.10	11.0	13.1.2*	5.14			
decltype(auto), Return type deduction for normal functions	<a href="#">N3638</a>	c++14-lang	4.8/4.9	3.3/3.4	19.0*	4.9	15.0	13.1.2*	5.15			
Initialized/Generalized lambda captures (init-capture)	<a href="#">N3648</a>	c++14-lang	4.5/4.9	3.4	19.0*	4.10	15.0		5.15			
Generic (polymorphic) lambda expressions	<a href="#">N3649</a>	c++14-lang	4.9	3.4	19.0*	4.10	16.0	13.1.2*	5.15			
Variable templates	<a href="#">N3651</a>	c++14-lang	5	3.4	19.0*	4.11	17.0	13.1.2*	5.15			
Extended constexpr	<a href="#">N3652</a>	c++14-lang	5	3.4	19.10*	4.11	17.0	13.1.2*	5.15			
Member initializers and aggregates (NSDMI)	<a href="#">N3653</a>	c++14-lang	5	3.3	19.10*	4.9	16.0		5.14			
Clarifying memory allocation (avoiding/fusing allocations)	<a href="#">N3664</a>	c++14-lang	N/A	3.4	N/A	N/A			N/A			
<code>[[deprecated]]</code> attribute	<a href="#">N3760</a>	c++14-lang	4.9	3.4	19.0*	4.9	15.0* 16.0	13.1.2*	5.14			
Sized deallocation	<a href="#">N3778</a>	c++14-lang	5	3.4	19.0*	4.10.1	17.0		5.14			

C++11 feature	Paper(s)	Version	GCC	Clang	MSVC	EDG ecpp	Intel C++	IBM XL C++	Sun/Oracle C++	Embarcadero C++ Builder	Cray	Portland Group (PGI)	HP aCC	Digital Mars C++
<code>alignas</code>	<a href="#">N2341</a>	c++11	4.8	3.0	19.0*	4.8	15.0	13.1.2*	5.13	Yes	8.6	2015		
<code>alignof</code>	<a href="#">N2341</a>	c++11	4.5	2.9	19.0*	4.8	15.0	13.1.2*	5.13	Yes	8.4	2015		
Atomic operations	<a href="#">N2427</a>	c++11	4.4	3.1	17.0*	Yes	13.0	13.1.2*	5.14	Yes	8.4	2015		
<code>auto</code>	<a href="#">N1984(v1.0)</a>	c++11	4.4(v1.0)	Yes	16.0*	4.1(v0.9)	11.0(v0.9) 12.0(v1.0)	11.1(v1.0)	5.13	Yes	8.4	2015	A.06.25	
C99 preprocessor	<a href="#">N1653</a>	c++11	4.3	Yes	19.0* (partial - buggy variadic macros)	4.1	11.1	10.1	5.9	Yes	8.4	2015	A.06.25	Yes
<code>constexpr</code>	<a href="#">N2235</a>	c++11	4.6	3.1	19.0* (partial)	4.6	13.0* 14.0	12.1* 13.1	5.13	Yes	8.4	2015	A.06.28	
<code>decltype</code>	v1.0: <a href="#">N2343</a> v1.1: <a href="#">N3276</a>	c++11	4.3(v1.0) 4.8.1(v1.1)	2.9	16.0*	4.1(v1.0)	11.0(v1.0) 12.0(v1.1)	11.1(v1.0)	5.13	Yes	8.4	2015	A.06.25	8.52(v1.0)
Defaulted and deleted functions	<a href="#">N2346</a>	c++11	4.4	3.0	18.0*	4.1	12.0	13.1	5.13	Yes	8.4	2015	A.06.25	
Delegating constructors	<a href="#">N1986</a>	c++11	4.7	3.0	18.0*	4.7	14.0	11.1	5.13	Yes	8.4	2015	A.06.28	
Explicit conversion operators	<a href="#">N2437</a>	c++11	4.5	3.0	18.0*	4.4	13.0	12.1	5.13	Yes	8.4	2015	A.06.27	
Extended <code>friend</code> declarations	<a href="#">N1791</a>	c++11	4.7	2.9	16.0* (partial)	4.1	11.1* 12.0	11.1	5.13	Yes	8.4	2015	A.06.25	
<code>extern template</code>	<a href="#">N1987</a>	c++11	3.3	Yes	12.0*	3.9	9.0	11.1	5.13	Yes	8.4	2015	A.06.25	
Forward <code>enum</code> declarations	<a href="#">N2764</a>	c++11	4.6	3.1	17.0*	4.5	11.1* 14.0	12.1	5.13	Yes	8.4	2015		



# Die großen Drei – Die Details

C++2a feature	Paper(s)	Version	GCC	Clang	MSVC	EDG ecpp	Intel C++	IBM XL C++	Sun/Oracle C++	Embarcadero C++ Builder	Cray	Portland Group (PGI)
Allow lambda-capture [=, this]	<a href="#">P0409R2</a>	c++2a-lang	8	6								
__VA_OPT__	<a href="#">P0306R4</a>	c++2a-lang	8 (partial)*	6								
Designated initializers	<a href="#">P0329R4</a>	c++2a-lang	4.7 (partial)* 8	3.0 (partial)*								
template-parameter-list for generic lambdas	<a href="#">P0428R2</a>	c++2a-lang	8									
Default member initializers for bit-fields	<a href="#">P0683R1</a>	c++2a-lang	8	6								
Initializer list constructors in class template argument deduction	<a href="#">P0702R1</a>	c++2a-lang	8	6								
const&-qualified pointers to members	<a href="#">P0704R1</a>	c++2a-lang	8	6								
Concepts	<a href="#">P0734R0</a>	c++2a-lang	6 (TS only)									
Lambdas in unevaluated contexts	<a href="#">P0315R4</a>	c++2a-lang	9									
Three-way comparison operator	<a href="#">P0515R3</a>	c++2a-lang		8 (partial)*								

C++17 feature	Paper(s)	Version	GCC	Clang	MSVC	EDG ecpp	Intel C++	IBM XL C++	Sun/Oracle C++	Embarcadero C++ Builder	Cray	Portland Group (PGI)
New auto rules for direct-list-initialization	<a href="#">N3922</a>	c++17-lang	5	3.8	19.0*	4.10.1	17.0					17.7
static_assert with no message	<a href="#">N3928</a>	c++17-lang	6	2.5	19.10*	4.12	18.0					17.7
typename in a template template parameter	<a href="#">N4051</a>	c++17-lang	5	3.5	19.0*	4.10.1	17.0					17.7
Removing trigraphs	<a href="#">N4086</a>	c++17-lang	5	3.5	16.0*	5.0						
Nested namespace definition	<a href="#">N4230</a>	c++17-lang	6	3.6	19.0*	4.12	17.0					17.7
Attributes for namespaces and enumerators	<a href="#">N4266</a>	c++17-lang	4.9 (namespaces) / 6 (enumerators)	3.6	19.0*	4.11	17.0					17.7
u8 character literals	<a href="#">N4267</a>	c++17-lang	6	3.6	19.0*	4.11	17.0					17.7
Allow constant evaluation for all non-type template arguments	<a href="#">N4268</a>	c++17-lang	6	3.6	19.12*	5.0						
Fold Expressions	<a href="#">N4295</a>	c++17-lang	6	3.6	19.12*	4.14	19.0					18.1
Remove Deprecated Use of the register Keyword	<a href="#">P0001R1</a>	c++17-lang	7	3.8	19.11*	4.13	18.0					17.7



# Compiler

C++11, C++14 und C++17

Compiler Flags

Offline to Online

Unter die Haube geschaut

# Standards und Optimierungen

- Standards:
  - GCC und Clang: `-std=c++11`, `-std=c++14` oder `-std=c++17`
  - MSVC: keine Spezifikation notwendig
    - Parallele STL: `/std=c++latest`
- Optimierung:
  - GCC und Clang: `-O3` steht für maximale Optimierung
  - MSVC: `/Ox` steht für maximale Optimierung

# Sanitizer

- Sanitizer
  - GCC  $\geq 4.8$
  - Clang  $\geq 3.2$
  - MSVC (Windows Portierung unter [GitHub](#))
- Dynamische Codeanalysis
  - [AddressSanitizer](#) (Adressen): `-fsanitize=address -g`
  - [LeakSanitizer](#) (Speicherlecks): `-fsanitize=address -g`
  - [ThreadSanitizer](#) (Data Races und Deadlocks): `-fsanitize=thread -g`
  - [MemorySanitizer](#) (Nicht initialisierter Speicher): `-fsanitize=memory -g`

# ThreadSanitizer

```
bool dataReady= false;

std::mutex mutex_;
std::condition_variable condVar1;
std::condition_variable condVar2;

int counter=0;
int COUNTLIMIT=50;

void setTrue(){

    while(counter <= COUNTLIMIT){

        std::unique_lock<std::mutex> lck(mutex_);
        condVar1.wait(lck,[]{return dataReady == false;});
        dataReady= true;
        ++counter;
        std::cout << dataReady << std::endl;
        condVar2.notify_one();

    }
}
```

```
void setFalse(){

    while(counter < COUNTLIMIT){

        std::unique_lock<std::mutex> lck(mutex_);
        condVar2.wait(lck,[]{return dataReady == true;});
        dataReady= false;
        std::cout << dataReady << std::endl;
        condVar1.notify_one();

    }
}

int main(){

    std::cout << std::boolalpha << std::endl;

    std::cout << "Begin: " << dataReady << std::endl;

    std::thread t1(setTrue);
    std::thread t2(setFalse);

    t1.join();
    t2.join();

    dataReady= false;
    std::cout << "End: " << dataReady << std::endl;

    std::cout << std::endl;

}
```

# ThreadSanitizer

```
File Edit View Bookmarks Settings Help
rainer@linux:~/conditionVariablePingPong$ ./conditionVariablePingPong
Begin: false
true
=====
WARNING: ThreadSanitizer: data race (pid=18133)
Read of size 4 at 0x000000004350 by thread T2:
#0 setFalse() /home/rainer/conditionVariablePingPong.cpp:30 (conditionVariablePingPong+0x000000401818)
#1 void std::_Bind_simple<void (*)()>::_M_invoke<std::_Index_tuple<> /usr/include/c++/6/functional:1400
#2 std::_Bind_simple<void (*)()>::operator()() /usr/include/c++/6/functional:1389 (conditionVariablePingPong+0x000000401818)
#3 std::thread::_State_impl<std::_Bind_simple<void (*)()> >::_M_run() /usr/include/c++/6/thread:196 (conditionVariablePingPong+0x00000000c22de)
#4 <null> <null> (libstdc++.so.6+0x00000000c22de)

Previous write of size 4 at 0x000000004350 by thread T1 (mutexes: write M11):
#0 setTrue() /home/rainer/conditionVariablePingPong.cpp:21 (conditionVariablePingPong+0x00000040173d)
#1 void std::_Bind_simple<void (*)()>::_M_invoke<std::_Index_tuple<> /usr/include/c++/6/functional:1400
#2 std::_Bind_simple<void (*)()>::operator()() /usr/include/c++/6/functional:1389 (conditionVariablePingPong+0x000000401818)
#3 std::thread::_State_impl<std::_Bind_simple<void (*)()> >::_M_run() /usr/include/c++/6/thread:196 (conditionVariablePingPong+0x00000000c22de)
#4 <null> <null> (libstdc++.so.6+0x00000000c22de)

Location is global 'counter' of size 4 at 0x000000004350 (conditionVariablePingPong+0x000000004350)

Mutex M11 (0x0000000042a0) created at:
#0 pthread_mutex_lock <null> (libtsan.so.0+0x000000003bc0f)
#1 __gthread_mutex_lock /usr/include/c++/6/x86_64-suse-linux/bits/gthr-default.h:748 (conditionVariablePingPong+0x00000000401be0)
#2 std::mutex::lock() /usr/include/c++/6/bits/std_mutex.h:103 (conditionVariablePingPong+0x00000000401be0)
#3 std::unique_lock<std::mutex>::lock() /usr/include/c++/6/bits/std_mutex.h:267 (conditionVariablePingPong+0x00000000401be0)
#4 std::unique_lock<std::mutex>::unique_lock(std::mutex&) /usr/include/c++/6/bits/std_mutex.h:197 (conditionVariablePingPong+0x000000004016f4)
#5 setTrue() /home/rainer/conditionVariablePingPong.cpp:18 (conditionVariablePingPong+0x000000004016f4)
#6 void std::_Bind_simple<void (*)()>::_M_invoke<std::_Index_tuple<> /usr/include/c++/6/functional:1400
#7 std::_Bind_simple<void (*)()>::operator()() /usr/include/c++/6/functional:1389 (conditionVariablePingPong+0x00000000401818)
#8 std::thread::_State_impl<std::_Bind_simple<void (*)()> >::_M_run() /usr/include/c++/6/thread:196 (conditionVariablePingPong+0x00000000c22de)
#9 <null> <null> (libstdc++.so.6+0x00000000c22de)

Thread T2 (tid=18140, running) created by main thread at:
#0 pthread_create <null> (libtsan.so.0+0x000000002b740)
#1 std::thread::_M_start_thread(std::unique_ptr<std::thread::_State, std::default_delete<std::thread::_State> >*) /usr/include/c++/6/thread:196 (conditionVariablePingPong+0x0000000040197c)
#2 main /home/rainer/conditionVariablePingPong.cpp:49 (conditionVariablePingPong+0x0000000040197c)

Thread T1 (tid=18139, running) created by main thread at:
#0 pthread_create <null> (libtsan.so.0+0x000000002b740)
#1 std::thread::_M_start_thread(std::unique_ptr<std::thread::_State, std::default_delete<std::thread::_State> >*) /usr/include/c++/6/thread:196 (conditionVariablePingPong+0x0000000040196b)
#2 main /home/rainer/conditionVariablePingPong.cpp:48 (conditionVariablePingPong+0x0000000040196b)

SUMMARY: ThreadSanitizer: data race /home/rainer/conditionVariablePingPong.cpp:30 in setFalse()
=====
false
true
false
true
false
```



# Compiler

C++11, C++14 und C++17

Compiler Flags

Offline to Online

Unter die Haube geschaut

# Arne Mertz Übersicht

## C++ Online Compiler

Name	Number Compilers	C++ Version	Boost Version	Execution	Distinguishing Features	Other Languages
<a href="#">Codiva.io</a>	1	C++17	1.65	✓	Clang, user input, multiple files, continuous compilation every few keystrokes, sharing and embedding in blogs	✓
<a href="#">paiza.IO</a>	1	C++14		✓	multiple files, collaborative live editing, full screen editor, Internet connection, GitHub (gist) integration	✓
<a href="#">Wandbox</a>	35	C++17	1.64	✓	multiple files	✓
<a href="#">Compiler Explorer (Godbolt)</a>	60+	C++17	1.64		compile to assembly as you type, on multiple compilers	✓
<a href="#">Coliru</a>	2	C++17	1.63 (header only)	✓	GCC & Clang, freely editable shell command line	
<a href="#">Quick-Bench</a>	1	C++17		only benchmarks	benchmarks functions against each other	
<a href="#">Cppinsights</a>	1	C++17			compile to a more verbose code that tells what the compiler does under the hood	

<a href="#">Rextester</a>	3	C++14	1.58 (header only)	✓	GCC, Clang, MSVC, collaborative live editing features	✓
<a href="#">Ideone</a>	1	C++14	1.62 (header only)	✓	GCC	✓
<a href="#">C++ Shell</a>	1	C++11-14	1.55 (header only)	✓	GCC, interactive Stdin	
<a href="#">repl.it</a>	1	C++17		✓	GCC, interactive Stdin	✓
<a href="#">Tutorialspoint CodingGround</a>	1	C++11		✓	multiple files like proper IDE, GCC but sluggish web app	
<a href="#">Geeksforgeeks</a>	1	C++14	1.58	✓	GCC, full screen editor	✓
<a href="#">Codepad</a>	1	C++03	1.34	✓	GCC	✓
<a href="#">TIO - Try It Online</a>	1	C++14		✓	Easy sharing, split source in header, source and footer	✓
<a href="#">LoopPerfect C++ Fiddle</a>					interactive C++ interpreter/terminal, but currently broken	

# Wandbox

## Wandbox - Der Mächtigste

Wandbox

C++  
gcc HEAD 9.0.0 201812 ▾

[Load template](#)

☒ Warnings  
☐ Optimization  
☐ Verbose  
Boost 1.69.0 ▾  
☐ Sprout  
☐ MessagePack  
C++2a(GNU) ▾  
no pedantic ▾  
Compiler options:  
  
Runtime options...

Corporate Sponsors:

```
1 // This file is a "Hello, world!" in C++ language by GCC for wandbox.
2 #include <iostream>
3 #include <cstdlib>
4
5 int main()
6 {
7     std::cout << "Hello, Wandbox!" << std::endl;
8 }
9
10 // GCC reference:
11 // https://gcc.gnu.org/
12
13 // C++ language references:
14 // https://msdn.microsoft.com/library/3bstk3k5.aspx
15 // http://www.cplusplus.com/
16 // https://isocpp.org/
17 // http://www.open-std.org/jtc1/sc22/wg21/
18
19 // Boost libraries references:
20 $ g++ prog.cc -Wall -Wextra -I/opt/wandbox/boost-1.69.0/gcc-head/include -std=gnu++2a
```

Stdin

Run (or Ctrl+Enter)

# Coliru

## Coliru - Bestandteil von [cppreference.com](http://cppreference.com)

### Containers library

---

The Containers library is a generic collection of class templates and algorithms that allow programmers to easily implement common data structures like queues, lists and stacks. There are three classes of containers -- sequence containers, associative containers, and unordered associative containers -- each of which is designed to support a different set of operations.

The container manages the storage space that is allocated for its elements and provides member functions to access them, either directly or through iterators (objects with properties similar to pointers).

Most containers have at least several member functions in common, and share functionalities. Which container is the best for the particular application depends not only on the offered functionality, but also on its efficiency for different workloads.

#### Sequence containers

Sequence containers implement data structures which can be accessed sequentially.

<b>array</b> (C++11)	static contiguous array (class template)
<b>vector</b>	dynamic contiguous array (class template)
<b>deque</b>	double-ended queue (class template)
<b>forward_list</b> (C++11)	singly-linked list (class template)
<b>list</b>	doubly-linked list (class template)

#### Associative containers

Associative containers implement sorted data structures that can be quickly searched ( $O(\log n)$  complexity).

<b>set</b>	collection of unique keys, sorted by keys (class template)
<b>map</b>	collection of key-value pairs, sorted by keys, keys are unique (class template)
<b>multiset</b>	collection of keys, sorted by keys (class template)
<b>multimap</b>	collection of key-value pairs, sorted by keys (class template)

#### Unordered associative containers

Unordered associative containers implement unsorted (hashed) data structures that can be quickly searched ( $O(1)$  amortized,  $O(n)$  worst-case complexity).

<b>unordered_set</b> (C++11)	collection of unique keys, hashed by keys (class template)
<b>unordered_map</b> (C++11)	collection of key-value pairs, hashed by keys, keys are unique (class template)
<b>unordered_multiset</b> (C++11)	collection of keys, hashed by keys (class template)
<b>unordered_multimap</b> (C++11)	collection of key-value pairs, hashed by keys (class template)

# cppreference.com

- [cppreference.com](http://cppreference.com)
  - Wiki-Seite
  - Sehr exakte Dokumentation der verschiedenen C++ Standards
    - C++98 - C++20
  - Hinweise zur Implementierung der Feature
  - Viele lauffähige Codebeispiele
    - Englische Version ist notwendig
  - Nach der Ausführung eines Programms
    - Eigener Code kann verwendet werden
    - Compiler (GCC und Clang) mit C++ Standard kann ausgewählt werden



# C++ Shell

## C++ Shell - Angenehmes Interface

### C++ shell

```
1 // Example program
2 #include <iostream>
3 #include <string>
4
5 int main()
6 {
7     std::string name;
8     std::cout << "What is your name? ";
9     getline (std::cin, name);
10    std::cout << "Hello, " << name << "!\n";
11 }
12
```

Short URL: [cpp.sh/](http://cpp.sh/)

Run

options | compilation | execution

Standard

☐ C++98  
☐ C++11  
☒ C++14

Warnings

☒ Many (-Wall)  
☐ Extra (-Wextra)  
☐ Pedantic (-Wpedantic)

Optimization level

☐ None (-O0)  
☐ Moderate (-O1)  
☒ Full (-O2)  
☐ Maximum (-O3)

Standard Input

☐ None  
☒ Interactive  
☐ Text:

# Rextester

## Rextester – Windows Compiler

compile visual studio c++ online

Language: C++ (vc++) Editor: CodeMirror Layout: Vertical

```
1 //Microsoft (R) C/C++ Optimizing Compiler Version 19.00.23506 for x64
2
3 #include <iostream>
4
5 int main()
6 {
7     std::cout << "Hello, world!\n";
8 }
```

Run it (F8)

Save it

☐ Show compiler warnings

[ + ] Compiler args

[ + ] Show input

# CppMem und Metashell

- Zwei Exoten
  - [CppMem](#)
    - Interaktives C/C++ Speichermodell
  - [Metashell](#)
    - Interaktive Template Metaprogrammierung Shell

# Vorteile von Online-Compilern

- Mehrwert
  - Neue Feature des C++-Standard können evaluiert werden
  - Schwer verständliche Fehlermeldungen lassen sich oft mit einem anderen Compiler (Clang) verständlicher darstellen
  - Code lässt sich einfach verifizieren, wenn verschiedene Compiler verwendet werden
  - Undefiniertes Verhalten kann einfacher entdeckt werden

# Undefiniertes Verhalten

## Rekursive Aufrufe von Konstruktoren

```
struct C {  
    C(char) : C(42.0) {}    // ill-formed due to recursion  
    C(double) : C('a') { } // ill-formed due to recursion  
};  
  
int main(){  
    C('a');  
    C(3.5);  
}
```



# Undefiniertes Verhalten

- MSVC



```
Compiled with /EHsc /nologo /W4
main.cpp

Compilation successful!

Total compilation time: 125ms

Maximum execution time exceeded!
```

- GCC



```
Start
Segmentation fault
Finish
```

- Clang



```
Start
prog.cc:3:15: error: constructor for 'C' creates a delegation cycle [-Wdelegating-ctor-cycles]
    C(double) : C('a') { } // ill-formed due to recursion
              ^
prog.cc:2:3: note: it delegates to
    C(char) : C(42.0) { } // ill-formed due to recursion
    ^
prog.cc:3:3: note: which delegates to
    C(double) : C('a') { } // ill-formed due to recursion
    ^
1 error generated.
1
Finish
```

# Undefiniertes Verhalten

Verwendung eines Strings, der seine Gültigkeit verloren hat.

```
#include <functional>
#include <iostream>
#include <string>

std::function<std::string()> makeLambda() {
    const std::string val = "on stack created";
    return [&val]{return val;};
}

int main(){

    auto bad = makeLambda();
    std::cout << bad();
}
```

# Undefiniertes Verhalten

on stack created

created

```
Start
0
Finish
```

HQ

```
`00`000@0
@0
@0k00x000@0f090
00@p000泔50"0v0o0)"0@P@0@p00Y@h00000000000000000000<00E00f00o00~0000000000000000:00|000000000000
000000000000V 0050S00K0nDx86_64
```

```
Start
Segmentation fault
Finish
```

# Undefiniertes Verhalten

```
rainer@linux:~> undefinedBehaviour
Segmentation fault (core dumped)
rainer@linux:~> undefinedBehaviour
Segmentation fault (core dumped)
rainer@linux:~> undefinedBehaviour
*** Error in `undefinedBehaviour': free(): invalid pointer 0x00007ffc56f14f98 ***
===== Backtrace: =====
/lib64/libc.so.6(+0x740ef)[0x7f8163fc30ef]
/lib64/libc.so.6(+0x79646)[0x7f8163fc8646]
/lib64/libc.so.6(+0x7a393)[0x7f8163fc9393]
/usr/lib64/libstdc++.so.6(_ZN5Sd2Ev+0x3e)[0x7f81648e7b6e]
undefinedBehaviour[0x400bd5]
/lib64/libc.so.6(__libc_start_main+0xf5)[0x7f8163f6f725]
undefinedBehaviour[0x4009f9]
===== Memory map: =====
00400000-00402000 r-xp 00000000 08:13 6907
00601000-00602000 r--p 00001000 08:13 6907
00802000-00803000 rw-p 00002000 08:13 6907
00f0e000-00f40000 rw-p 00000000 00:00 0
7f815c000000-7f815c021000 rw-p 00000000 00:00 0
7f815c021000-7f8160000000 ---p 00000000 00:00 0
7f8163f4f000-7f81640ea000 r-xp 00000000 00:27 1415842
7f81640ea000-7f81642ea000 ---p 0019b000 00:27 1415842
7f81642ea000-7f81642ee000 r--p 0019b000 00:27 1415842
7f81642ee000-7f81642f0000 rw-p 0019f000 00:27 1415842
7f81642f0000-7f81642f4000 rw-p 00000000 00:00 0
7f81642f7000-7f816430e000 r-xp 00000000 00:27 1359149
7f816430e000-7f816450d000 ---p 00017000 00:27 1359149
7f816450d000-7f816450e000 r--p 00016000 00:27 1359149
7f816450e000-7f816450f000 rw-p 00017000 00:27 1359149
7f816450f000-7f816460a000 r-xp 00000000 00:27 1415850
7f816460a000-7f816480a000 ---p 000fb000 00:27 1415850
7f816480a000-7f816480b000 r--p 000fb000 00:27 1415850
7f816480b000-7f816480c000 rw-p 000fc000 00:27 1415850
7f816480f000-7f816498a000 r-xp 00000000 00:27 1359277
7f816498a000-7f8164b8a000 ---p 0017b000 00:27 1359277
7f8164b8a000-7f8164b94000 r--p 0017b000 00:27 1359277
7f8164b94000-7f8164b96000 rw-p 00185000 00:27 1359277
7f8164b96000-7f8164b99000 rw-p 00000000 00:00 0
7f8164b9f000-7f8164bc0000 r-xp 00000000 00:27 1415834
7f8164d8c000-7f8164d8f000 rw-p 00000000 00:00 0
7f8164dbe000-7f8164dc0000 rw-p 00000000 00:00 0
7f8164dc0000-7f8164dc1000 r--p 00021000 00:27 1415834
7f8164dc1000-7f8164dc2000 rw-p 00022000 00:27 1415834
7f8164dc2000-7f8164dc4000 rw-p 00000000 00:00 0
7f8164dc4000-7f8164dc5000 rw-p 00000000 00:00 0
7ffc56ef7000-7ffc56f18000 rw-p 00000000 00:00 0
7ffc56fb7000-7ffc56fba000 r--p 00000000 00:00 0
7ffc56fba000-7ffc56fbc000 r-xp 00000000 00:00 0
fffffffff600000-fffffffff601000 r-xp 00000000 00:00 0
Aborted (core dumped)
rainer@linux:~> █
```

```
/home/rainer/undefinedBehaviour
/home/rainer/undefinedBehaviour
/home/rainer/undefinedBehaviour
[heap]
```

```
/lib64/libc-2.22.so
/lib64/libc-2.22.so
/lib64/libc-2.22.so
/lib64/libc-2.22.so
```

```
/lib64/libgcc_s.so.1
/lib64/libgcc_s.so.1
/lib64/libgcc_s.so.1
/lib64/libgcc_s.so.1
/lib64/libm-2.22.so
/lib64/libm-2.22.so
/lib64/libm-2.22.so
/lib64/libm-2.22.so
/usr/lib64/libstdc++.so.6.0.25
/usr/lib64/libstdc++.so.6.0.25
/usr/lib64/libstdc++.so.6.0.25
/usr/lib64/libstdc++.so.6.0.25
```

```
/lib64/ld-2.22.so
```

```
/lib64/ld-2.22.so
/lib64/ld-2.22.so
```

```
[stack]
[vvar]
[vdso]
[vsyscall]
```

# Compiler

C++11, C++14 und C++17

Compiler Flags

Offline to Online

Unter die Haube geschaut



# C++ Insight von Andreas Fertig

C++ Insight bringt die Compiler Magie ans Licht.

```
#include <algorithm>
#include <iostream>
#include <vector>

int main(){

    std::cout << std::endl;

    std::vector<int> vec{1, 2, 3, 4, 5, 6, 7, 8, 9};
    std::transform(vec.begin(), vec.end(), vec.begin(),
        [](auto i){ return i * i; });

    for (auto v: vec){
        std::cout << v << " ";
    }

    std::cout << "\n\n";
}
```

Insight:

```
1 #include <algorithm>
2 #include <iostream>
3 #include <vector>
4
5 int main(){
6
7     std::cout.operator<<(std::endl);
8
9     std::vector<int> vec{ std::initializer_list<int>{1, 2, 3, 4, 5, 6, 7, 8, 9} };
10
11     class __lambda_10_56
12     {
13     public: inline /*constexpr */ int operator()(int i) const
14     {
15         return i * i;
16     }
17
18     };
19
20     std::transform(vec.begin(), vec.end(), vec.begin(), __lambda_10_56{});
21
22     {
23         std::vector<int, std::allocator<int> > & __range = vec;
24         __gnu_cxx::__normal_iterator<int *, std::vector<int, std::allocator<int> > > __begin = __range.begin();
25         __gnu_cxx::__normal_iterator<int *, std::vector<int, std::allocator<int> > > __end = __range.end();
26
27         for( ; __gnu_cxx::operator!==( __begin, __end); __begin.operator++() )
28         {
29             int v = __begin.operator*();
30             std::operator<<(std::cout.operator<<(v), " ");
31         }
32     }
33
34     std::operator<<(std::cout, "\n\n");
35
36 }
```

# C++ Insight von Andreas Fertig

```
#include <utility>

struct MyType{
    MyType(int, double, bool){};
};

template <typename T, typename ... Args>
T createT(Args&& ... args){
    return T(std::forward<Args>(args) ... );
}

int main(){

    int lvalue{2020};

    int uniqZero = createT<int>();           // (1)
    auto uniqEleven = createT<int>(2011);    // (2)
    auto uniqTwenty = createT<int>(lvalue);  // (3)
    auto uniqType = createT<MyType>(lvalue, 3.14, true); // (4)
}
```

```
21 /* First instantiated from: insights.cpp:18 */
22 #ifdef INSIGHTS_USE_TEMPLATE
23 template<>
24 int createT<int, >()
25 {
26     return int();
27 }
28 #endif
29
30
31 /* First instantiated from: insights.cpp:19 */
32 #ifdef INSIGHTS_USE_TEMPLATE
33 template<>
34 int createT<int, int>(int && __args0)
35 {
36     return int(std::forward<int>(__args0));
37 }
38 #endif
```

```
41 /* First instantiated from: insights.cpp:20 */
42 #ifdef INSIGHTS_USE_TEMPLATE
43 template<>
44 int createT<int, int &>(int & args)
45 {
46     return int(std::forward<int &>(args));
47 }
48 #endif
49
50
51 /* First instantiated from: insights.cpp:21 */
52 #ifdef INSIGHTS_USE_TEMPLATE
53 template<>
54 MyType createT<MyType, int &, double, bool>(int & args, double && __args1, bool && __args2)
55 {
56     return MyType(std::forward<int &>(args), std::forward<double>(__args1), std::forward<bool>(__args2));
57 }
58 #endif
```

# C++ Insight von Andreas Fertig

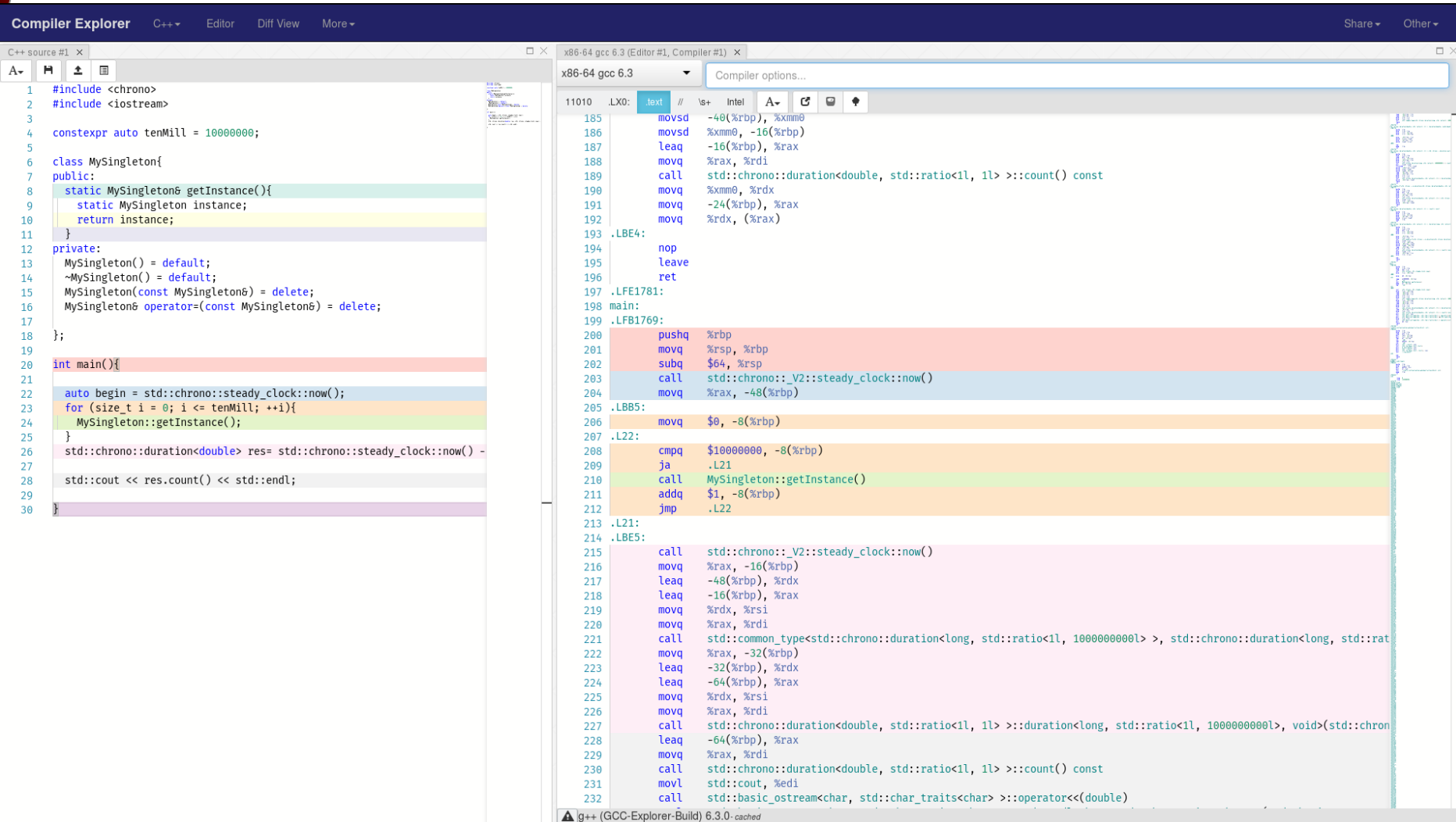
- Beispiele
  - [auto](#)
  - [Automatisch erzeugte Methoden](#)
  - [Range-basierte for-Schleife](#)
  - [Lambda-Funktionen](#)
  - [Perfect-Forwarding mit Variadic Templates](#)
  - [Operatoren Überladung](#)

# Compiler Explorer von Matt Godbolt

Compiler Explorer erzeugt die Assembler Instruktionen.

- Unterstützt mehr als 60 Compiler
  - Stellt Sourcecode den Assemblerinstruktionen gegenüber
  - Erlaubt gleichzeitige mehrere Computerarchitekturen darzustellen
- 
- Vorteile:
    - Erlaubt tiefe Einblicke in den Kompilierungs- und Optimierungsprozess
    - Zeigt, ob ein Ausdruck bereits beim Kompilieren berechnet oder eine Funktion `inline` aufgerufen wird
    - Stellt die Assemblerinstruktionen abhängig vom Optimierungslevel dar

# Compiler Explorer von Matt Godbolt



The screenshot displays the Compiler Explorer interface. The left pane shows the C++ source code for a singleton class and a main function. The right pane shows the assembly output generated by x86-64 gcc 6.3. The assembly is color-coded to match the source code, with labels like .LBE4, .LFB1769, .LBB5, .L22, .L21, .LBE5, and .L23. The status bar at the bottom indicates the compiler is g++ (GCC-Explorer-Build) 6.3.0 - cached.

```
1 #include <chrono>
2 #include <iostream>
3
4 constexpr auto tenMill = 10000000;
5
6 class MySingleton{
7 public:
8     static MySingleton& getInstance(){
9         static MySingleton instance;
10        return instance;
11    }
12 private:
13     MySingleton() = default;
14     ~MySingleton() = default;
15     MySingleton(const MySingleton&) = delete;
16     MySingleton& operator=(const MySingleton&) = delete;
17 };
18
19 int main(){
20
21     auto begin = std::chrono::steady_clock::now();
22     for (size_t i = 0; i <= tenMill; ++i){
23         MySingleton::getInstance();
24     }
25     std::chrono::duration<double> res= std::chrono::steady_clock::now() -
26
27     std::cout << res.count() << std::endl;
28
29
30 }
```

```
11010 .LX0: .text //
185     movsd    -40(%rbp), %xmm0
186     movsd    %xmm0, -16(%rbp)
187     leaq     -16(%rbp), %rax
188     movq     %rax, %rdi
189     call     std::chrono::duration<double, std::ratio<1l, 1l> >::count() const
190     movq     %xmm0, %rdx
191     movq     -24(%rbp), %rax
192     movq     %rdx, (%rax)
193 .LBE4:
194     nop
195     leave
196     ret
197 .LFE1781:
198 main:
199 .LFB1769:
200     pushq    %rbp
201     movq     %rsp, %rbp
202     subq     $64, %rsp
203     call     std::chrono::_V2::steady_clock::now()
204     movq     %rax, -48(%rbp)
205 .LBB5:
206     movq     $0, -8(%rbp)
207 .L22:
208     cmpq     $10000000, -8(%rbp)
209     ja       .L21
210     call     MySingleton::getInstance()
211     addq     $1, -8(%rbp)
212     jmp      .L22
213 .L21:
214 .LBE5:
215     call     std::chrono::_V2::steady_clock::now()
216     movq     %rax, -16(%rbp)
217     leaq     -48(%rbp), %rdx
218     leaq     -16(%rbp), %rax
219     movq     %rdx, %rsi
220     movq     %rax, %rdi
221     call     std::common_type<std::chrono::duration<long, std::ratio<1l, 100000000l> >, std::chrono::duration<long, std::ratio<1l, 100000000l> >::count() const
222     movq     %rax, -32(%rbp)
223     leaq     -32(%rbp), %rdx
224     leaq     -64(%rbp), %rax
225     movq     %rdx, %rsi
226     movq     %rax, %rdi
227     call     std::chrono::duration<double, std::ratio<1l, 1l> >::duration<long, std::ratio<1l, 100000000l>, void>(std::chrono::duration<double, std::ratio<1l, 1l> >::count() const
228     leaq     -64(%rbp), %rax
229     movq     %rax, %rdi
230     call     std::chrono::duration<double, std::ratio<1l, 1l> >::count() const
231     movl     %edi, %edi
232     call     std::basic_ostream<char, std::char_traits<char> >::operator<<(double)
```

g++ (GCC-Explorer-Build) 6.3.0 - cached

# Compiler Explorer von Matt Godbolt

```
#include <chrono>
#include <iostream>

constexpr auto tenMill = 10000000;

class MySingleton{
public:
    static MySingleton& getInstance(){
        static MySingleton instance;
        return instance;
    }
private:
    MySingleton() = default;
    ~MySingleton() = default;
    MySingleton(const MySingleton&) = delete;
    MySingleton& operator=(const MySingleton&) = delete;
};

int main(){

    auto begin = std::chrono::steady_clock::now();
    for (size_t i = 0; i <= tenMill; ++i){
        MySingleton::getInstance();
    }
    std::chrono::duration<double> res= std::chrono::steady_clock::now() - begin;

    std::cout << res.count() << std::endl;
}
```



```
File Edit View Bookmarks Settings Help
rainer@suse:~> singleton
0.0332643
rainer@suse:~> singletonOptimised
2.52e-07
rainer@suse:~> █
> rainer: bash
```

# Compiler Explorer von Matt Godbolt

## Nicht optimierte Variante

```
19
20 int main(){
21
22     auto begin = std::chrono::steady_clock::now();
23     for (size_t i = 0; i <= tenMill; ++i){
24         MySingleton::getInstance();
25     }
26     std::chrono::duration<double> res= std::chrono::steady_clock::now() -
27
28     std::cout << res.count() << std::endl;
29
30 }
```

```
207 .L22:
208     cmpq    $100000000, -8(%rbp)
209     ja      .L21
210     call    MySingleton::getInstance()
211     addq    $1, -8(%rbp)
212     jmp     .L22
213 .L21:
```

## Optimierte Variante

```
20 int main(){
21
22     auto begin = std::chrono::steady_clock::now();
23     for (size_t i = 0; i <= tenMill; ++i){
24         MySingleton::getInstance();
25     }
26     std::chrono::duration<double> res= std::chrono::steady_clock::now() -
27
28     std::cout << res.count() << std::endl;
29
30 }
```

```
2 main:
3 .LFB1793:
4     pushq   %rbx
5     call    std::chrono::_V2::steady_clock::now()
6 .LVL0:
7     movq    %rax, %rbx
8 .LVL1:
9     call    std::chrono::_V2::steady_clock::now()
```

# Compiler Explorer von Matt Godbolt

```
#include <iostream>
```

```
constexpr int gcd(int a, int b){  
    while (b != 0){  
        auto t = b;  
        b = a % b;  
        a = t;  
    }  
    return a;  
}
```

```
int main(){  
  
    auto res = gcd(121, 11);  
    constexpr auto res1 = gcd(100, 10);  
  
    std::cout << "res: " << res << std::endl;  
    std::cout << "res1: " << res1 << std::endl;  
}
```



```
main:
```

```
    push rbp  
    mov rbp, rsp  
    sub rsp, 16  
    mov esi, 11  
    mov edi, 121  
    call gcd(int, int)  
    mov DWORD PTR [rbp-4], eax  
    mov DWORD PTR [rbp-8], 10  
    mov esi, OFFSET FLAT:.LC0  
    mov edi, OFFSET FLAT:_ZSt4cout  
    call std::basic_ostream<char, std::char_traits<char>>::operator<<<(std::basic_ostream<char, std::char_traits<char>>*, const char*)@plt
```



# Compiler Explorer von Matt Godbolt

- Beispiele
  - [gcd](#)
  - [Singleton](#)

# Blogs

[www.grimm-jaud.de](http://www.grimm-jaud.de) [De]

[www.ModernesCpp.com](http://www.ModernesCpp.com) [En]

Rainer Grimm

Training, Coaching und  
Technologieberatung

[www.ModernesCpp.de](http://www.ModernesCpp.de)