Die bekanntesten (Online-)Compiler im Vergleich

Rainer Grimm

Training, Coaching und Technologieberatung

www.ModernesCpp.de

Compiler

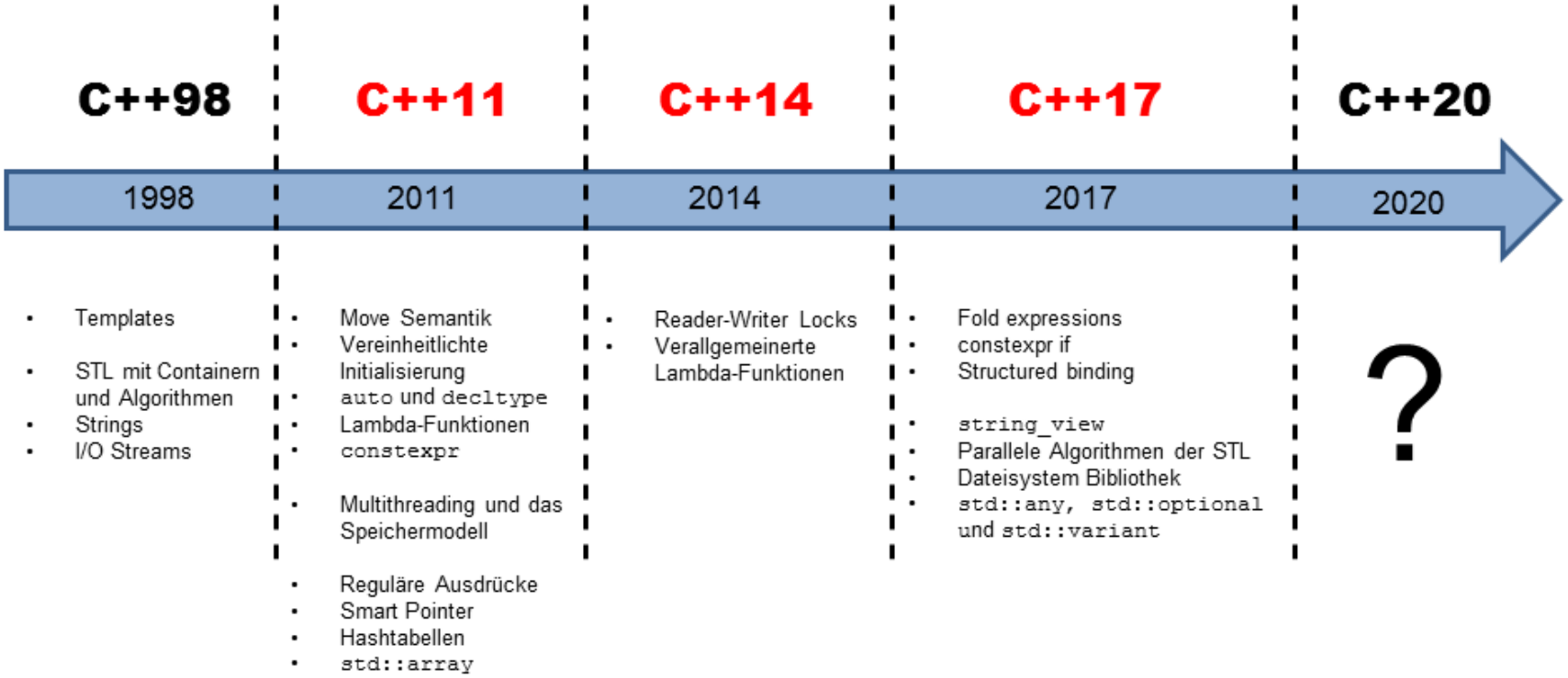
- C++11, C++14 und C++17
- Compiler Flags
- Offline to Online
- Unter die Haube geschaut

πάντα ῥεῖ

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

C++14 feature	Paper(s)	Version	GCC	Clang	MSVC	EDG eccp	Intel C++	IBM XLC++	Sun/Oracle C++	Embarcadero C++ Builder	Cray	Portland Group (PGI)
Tweaked wording for contextual conversions	N3323	c++14-lang	4.9	3.4	18.0*	4.9	16.0	13.1.2*	5.15			
Binary literals	N3472	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	N3638	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)	N3648	c++14-lang	4.5/4.9	3.4	19.0*	4.10	15.0		5.15			
Generic (polymorphic) lambda expressions	N3649	c++14-lang	4.9	3.4	19.0*	4.10	16.0	13.1.2*	5.15			
Variable templates	N3651	c++14-lang	5	3.4	19.0*	4.11	17.0	13.1.2*	5.15			
Extended constexpr	N3652	c++14-lang	5	3.4	19.10*	4.11	17.0	13.1.2*	5.15			
Member initializers and aggregates (NSDMI)	N3653	c++14-lang	5	3.3	19.10*	4.9	16.0		5.14			
Clarifying memory allocation (avoiding/fusing allocations)	N3664	c++14-lang	N/A	3.4	N/A	N/A			N/A			
[[deprecated]] attribute	N3760	c++14-lang	4.9	3.4	19.0*	4.9	15.0* 16.0	13.1.2*	5.14			
Sized deallocation	N3778	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 eccp	Intel C++	IBM XLC++	Sun/Oracle C++	Embarcadero C++ Builder	Cray	Portland Group (PGI)	HP aCC	Digital Mars C++
alignas	N2341	c++11	4.8	3.0	19.0*	4.8	15.0	13.1.2*	5.13	Yes	8.6	2015		
alignof	N2341	c++11	4.5	2.9	19.0*	4.8	15.0	13.1.2*	5.13	Yes	8.4	2015		
Atomic operations	N2427	c++11	4.4	3.1	17.0*	Yes	13.0	13.1.2*	5.14	Yes	8.4	2015		
auto	N1984(v1.0)	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	N1653	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
constexpr	N2235	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	
decltype	v1.0: N2343 v1.1: N3276	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	N2346	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	N1986	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	N2437	c++11	4.5	3.0	18.0*	4.4	13.0	12.1	5.13	Yes	8.4	2015	A.06.27	
Extended friend declarations	N1791	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	
extern template	N1987	c++11	3.3	Yes	12.0*	3.9	9.0	11.1	5.13	Yes	8.4	2015	A.06.25	
Forward enum declarations	N2764	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 eccp	Intel C++	IBM XLC++	Sun/Oracle C++	Embarcadero C++ Builder	Cray	Portland Group (PGI)
Allow lambda-capture [=, this]	P0409R2	c++2a-lang	8	6								
__VA_OPT__	P0306R4	c++2a-lang	8 (partial)*	6								
Designated initializers	P0329R4	c++2a-lang	4.7 (partial)* 8	3.0 (partial)*								
template-parameter-list for generic lambdas	P0428R2	c++2a-lang	8									
Default member initializers for bit-fields	P0683R1	c++2a-lang	8	6								
Initializer list constructors in class template argument deduction	P0702R1	c++2a-lang	8	6								
const&-qualified pointers to members	P0704R1	c++2a-lang	8	6								
Concepts	P0734R0	c++2a-lang	6 (TS only)									
Lambdas in unevaluated contexts	P0315R4	c++2a-lang	9									
Three-way comparison operator	P0515R3	c++2a-lang		8 (partial)*								

C++17 feature	Paper(s)	Version	GCC	Clang	MSVC	EDG eccp	Intel C++	IBM XLC++	Sun/Oracle C++	Embarcadero C++ Builder	Cray	Portland Group (PGI)
New auto rules for direct-list-initialization	N3922	c++17-lang	5	3.8	19.0*	4.10.1	17.0					17.7
static_assert with no message	N3928	c++17-lang	6	2.5	19.10*	4.12	18.0					17.7
typename in a template template parameter	N4051	c++17-lang	5	3.5	19.0*	4.10.1	17.0					17.7
Removing trigraphs	N4086	c++17-lang	5	3.5	16.0*	5.0						
Nested namespace definition	N4230	c++17-lang	6	3.6	19.0*	4.12	17.0					17.7
Attributes for namespaces and enumerators	N4266	c++17-lang	4.9 (namespaces) / 6 (enumerators)	3.6	19.0*	4.11	17.0					17.7
u8 character literals	N4267	c++17-lang	6	3.6	19.0*	4.11	17.0					17.7
Allow constant evaluation for all non-type template arguments	N4268	c++17-lang	6	3.6	19.12*	5.0						
Fold Expressions	N4295	c++17-lang	6	3.6	19.12*	4.14	19.0					18.1
Remove Deprecated Use of the register Keyword	P0001R1	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 >= 4.8
  - Clang >= 3.2
  - MSVC (Windows Portierung unter GitHub)

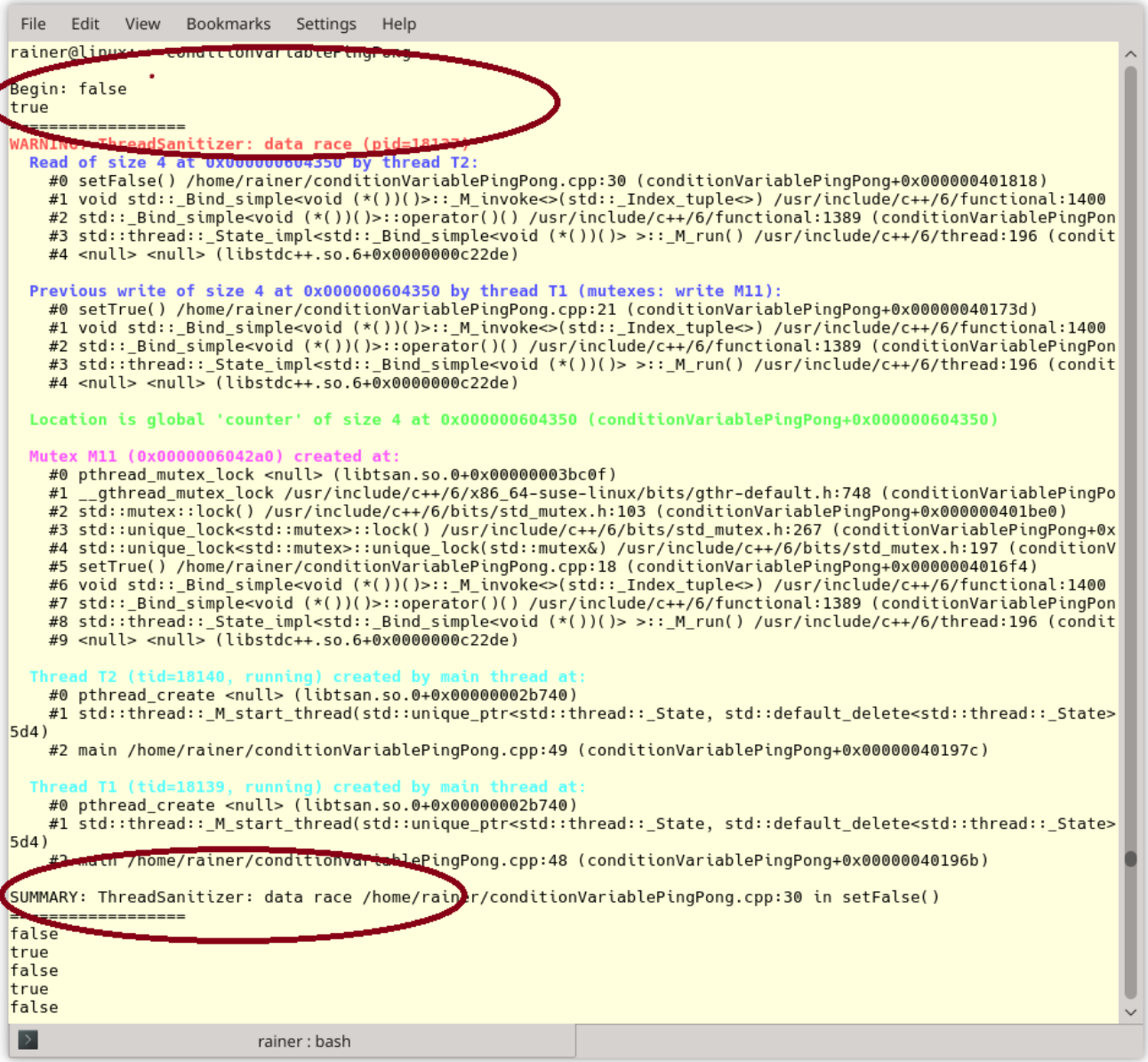
- Dynamische Codeanalysis
  - AddressSanitizer (Adressen): `-fsanitize=address -g`
  - LeakSanitizer (Speicherlecks): `-fsanitize=address -g`
  - ThreadSanitizer (Data Races und Deadlocks): `-fsanitize=thread -g`
  - MemorySanitizer (Nicht initialisierter Speicher): `-fsanitize=memory -g`

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



Compiler

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

Compiler Flags

Offline to Online

Unter die Haube geschaut

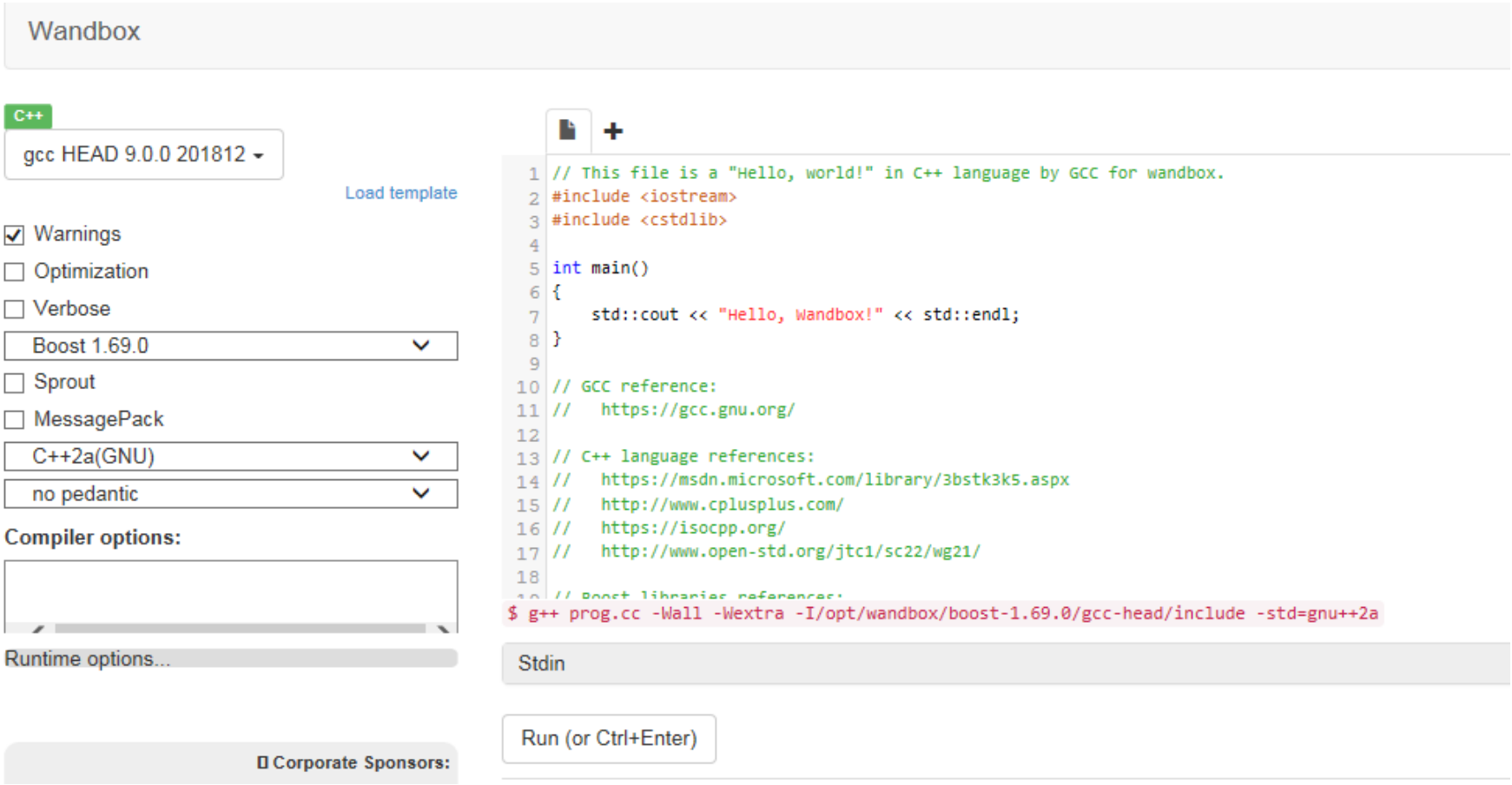
Arne Mertz Übersicht

C++ Online Compiler

Name	Number Compilers	C++ Version	Boost Version	Execution	Distinguishing Features	Other Languages
Codiva.io	1	C++17	1.65	✔	Clang, user input, multiple files, continuous compilation every few keystrokes, sharing and embeding in blogs	✔
paiza.IO	1	C++14		✔	multiple files, collaborative live editing, full screen editor, Internet connection, GitHub (gist) integration	✔
Wandbox	35	C++17	1.64	✔	multiple files	✔
Compiler Explorer (Godbolt)	60+	C++17	1.64		compile to assembly as you type, on multiple compilers	✔
Coliru	2	C++17	1.63 (header only)	✔	GCC & Clang, freely editable shell command line	
Quick-Bench	1	C++17		only benchmarks	benchmarks functions against each other	
Cppinsights	1	C++17			compile to a more verbose code that tells what the compiler does under the hood	
Rextester	3	C++14	1.58 (header only)	✔	GCC, Clang, MSVC, collaborative live editing features	✔
Ideone	1	C++14	1.62 (header only)	✔	GCC	✔
C++ Shell	1	C++11-14	1.55 (header only)	✔	GCC, interactive Stdin	
repl.it	1	C++17		✔	GCC, interactive Stdin	✔
Tutorialspoint CodingGround	1	C++11		✔	multiple files like proper IDE, GCC but sluggish web app	
Geeksforgeeks	1	C++14	1.58	✔	GCC, full screen editor	✔
Codepad	1	C++03	1.34	✔	GCC	✔
TIO - Try It Online	1	C++14		✔	Easy sharing, split source in header, source and footer	✔
LoopPerfect C++ Fiddle					interactive C++ interpreter/terminal, but currently broken	

Wandbox

Wandbox - Der Mächtigste



Coliru

Coliru - Bestandteil von 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.

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

Associative containers

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

`set`	collection of unique keys, sorted by keys (class template)
`map`	collection of key-value pairs, sorted by keys, keys are unique (class template)
`multiset`	collection of keys, sorted by keys (class template)
`multimap`	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).

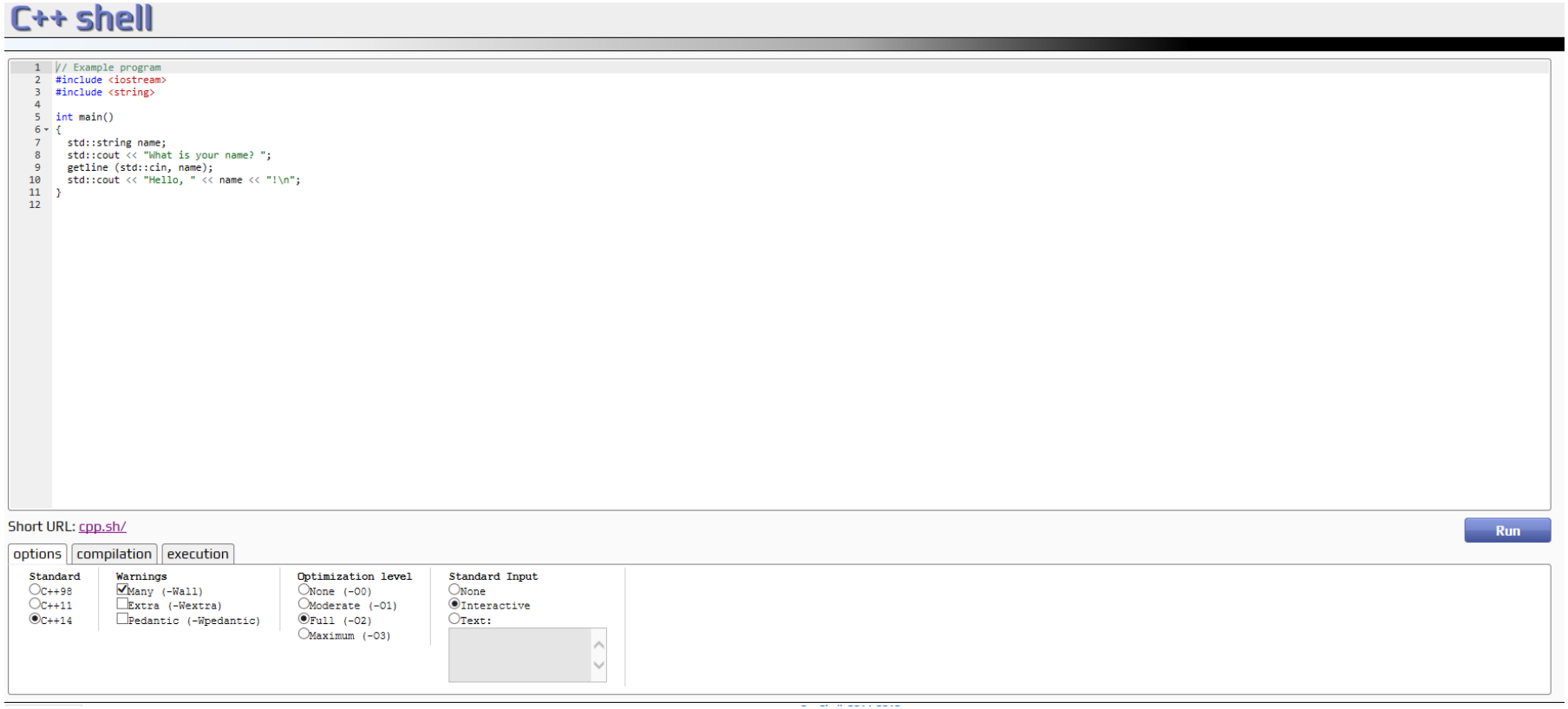
`unordered_set` (C++11)	collection of unique keys, hashed by keys (class template)
`unordered_map` (C++11)	collection of key-value pairs, hashed by keys, keys are unique (class template)
`unordered_multiset` (C++11)	collection of keys, hashed by keys (class template)
`unordered_multimap` (C++11)	collection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

cppreference.com

- 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

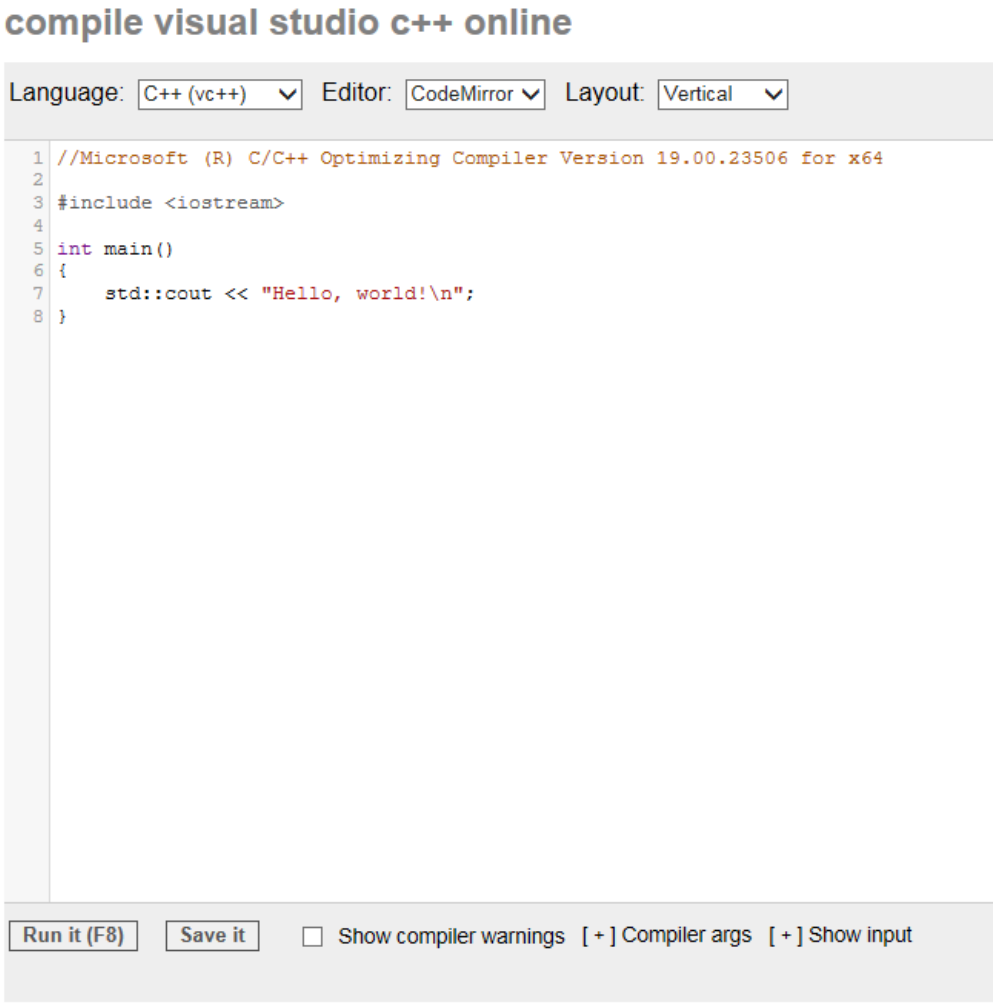
C++ Shell

C++ Shell - Angenehmes Interface



Rextester

Rextester – Windows Compiler



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
```

H□

```
`��`���0@@
@0
@0k��x���0@@f@9@
@0@p���汾5@"@v@o@)"0@P@0@p@@Y@h<@@E@@f@@o@@~:@@|
@@@@@@@@@@@V @@5@S@@K@nDx86_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(_ZNSsD2Ev+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                               /home/rainer/undefinedBehaviour
00601000-00602000 r--p 00001000 08:13 6907                               /home/rainer/undefinedBehaviour
00602000-00603000 rw-p 00002000 08:13 6907                               /home/rainer/undefinedBehaviour
00f0e000-00f40000 rw-p 00000000 00:00 0                                  [heap]
7f815c000000-7f815c021000 rw-p 00000000 00:00 0
7f815c021000-7f8160000000 ---p 00000000 00:00 0
7f8163f4f000-7f81640ea000 r-xp 00000000 00:27 1415842                    /lib64/libc-2.22.so
7f81640ea000-7f81642ea000 ---p 0019b000 00:27 1415842                    /lib64/libc-2.22.so
7f81642ea000-7f81642ee000 r--p 0019b000 00:27 1415842                    /lib64/libc-2.22.so
7f81642ee000-7f81642f0000 rw-p 0019f000 00:27 1415842                    /lib64/libc-2.22.so
7f81642f0000-7f81642f4000 rw-p 00000000 00:00 0
7f81642f7000-7f816430e000 r-xp 00000000 00:27 1359149                    /lib64/libgcc_s.so.1
7f816430e000-7f816450d000 ---p 00017000 00:27 1359149                    /lib64/libgcc_s.so.1
7f816450d000-7f816450e000 r--p 00016000 00:27 1359149                    /lib64/libgcc_s.so.1
7f816450e000-7f816450f000 rw-p 00017000 00:27 1359149                    /lib64/libgcc_s.so.1
7f816450f000-7f816460a000 r-xp 00000000 00:27 1415850                    /lib64/libm-2.22.so
7f816460a000-7f816480a000 ---p 000fb000 00:27 1415850                    /lib64/libm-2.22.so
7f816480a000-7f816480b000 r--p 000fb000 00:27 1415850                    /lib64/libm-2.22.so
7f816480b000-7f816480c000 rw-p 000fc000 00:27 1415850                    /lib64/libm-2.22.so
7f816480f000-7f816498a000 r-xp 00000000 00:27 1359277                    /usr/lib64/libstdc++.so.6.0.25
7f816498a000-7f8164b8a000 ---p 0017b000 00:27 1359277                    /usr/lib64/libstdc++.so.6.0.25
7f8164b8a000-7f8164b94000 r--p 0017b000 00:27 1359277                    /usr/lib64/libstdc++.so.6.0.25
7f8164b94000-7f8164b96000 rw-p 00185000 00:27 1359277                    /usr/lib64/libstdc++.so.6.0.25
7f8164b96000-7f8164b99000 rw-p 00000000 00:00 0
7f8164b9f000-7f8164bc0000 r-xp 00000000 00:27 1415834                    /lib64/ld-2.22.so
7f8164d8c000-7f8164d8f000 rw-p 00000000 00:00 0
7f8164dbe000-7f8164dc0000 rw-p 00000000 00:00 0
7f8164dc0000-7f8164dc1000 r--p 00021000 00:27 1415834                    /lib64/ld-2.22.so
7f8164dc1000-7f8164dc2000 rw-p 00022000 00:27 1415834                    /lib64/ld-2.22.so
7f8164dc2000-7f8164dc4000 rw-p 00000000 00:00 0
7f8164dc4000-7f8164dc5000 rw-p 00000000 00:00 0
7ffc56ef7000-7ffc56f18000 rw-p 00000000 00:00 0                          [stack]
7ffc56fb7000-7ffc56fba000 r--p 00000000 00:00 0                          [vvar]
7ffc56fba000-7ffc56fbc000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
Aborted (core dumped)
rainer@linux:~>
```

rainer : bash

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.

```cpp
#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:

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

int main(){

    std::cout.operator<<(std::endl);

    std::vector<int> vec{ std::initializer_list<int>{1, 2, 3, 4, 5, 6, 7, 8, 9} };

     class __lambda_10_56
     {
       public: inline /*constexpr */ int operator()(int i) const
       {
         return i * i;
       }

     };

     std::transform(vec.begin(), vec.end(), vec.begin(), __lambda_10_56{});

    {
      std::vector<int, std::allocator<int> > & __range = vec;
      __gnu_cxx::__normal_iterator<int *, std::vector<int, std::allocator<int> > > __begin = __range.begin();
      __gnu_cxx::__normal_iterator<int *, std::vector<int, std::allocator<int> > > __end = __range.end();

      for( ; __gnu_cxx::operator!=(__begin, __end); __begin.operator++() )
      {
        int v = __begin.operator*();
        std::operator<<(std::cout.operator<<(v), " ");
      }
    }

    std::operator<<(std::cout, "\n\n");

}
```

C++ Insight von Andreas Fertig

```cpp
#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)

}
```

```cpp
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
```

```cpp
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

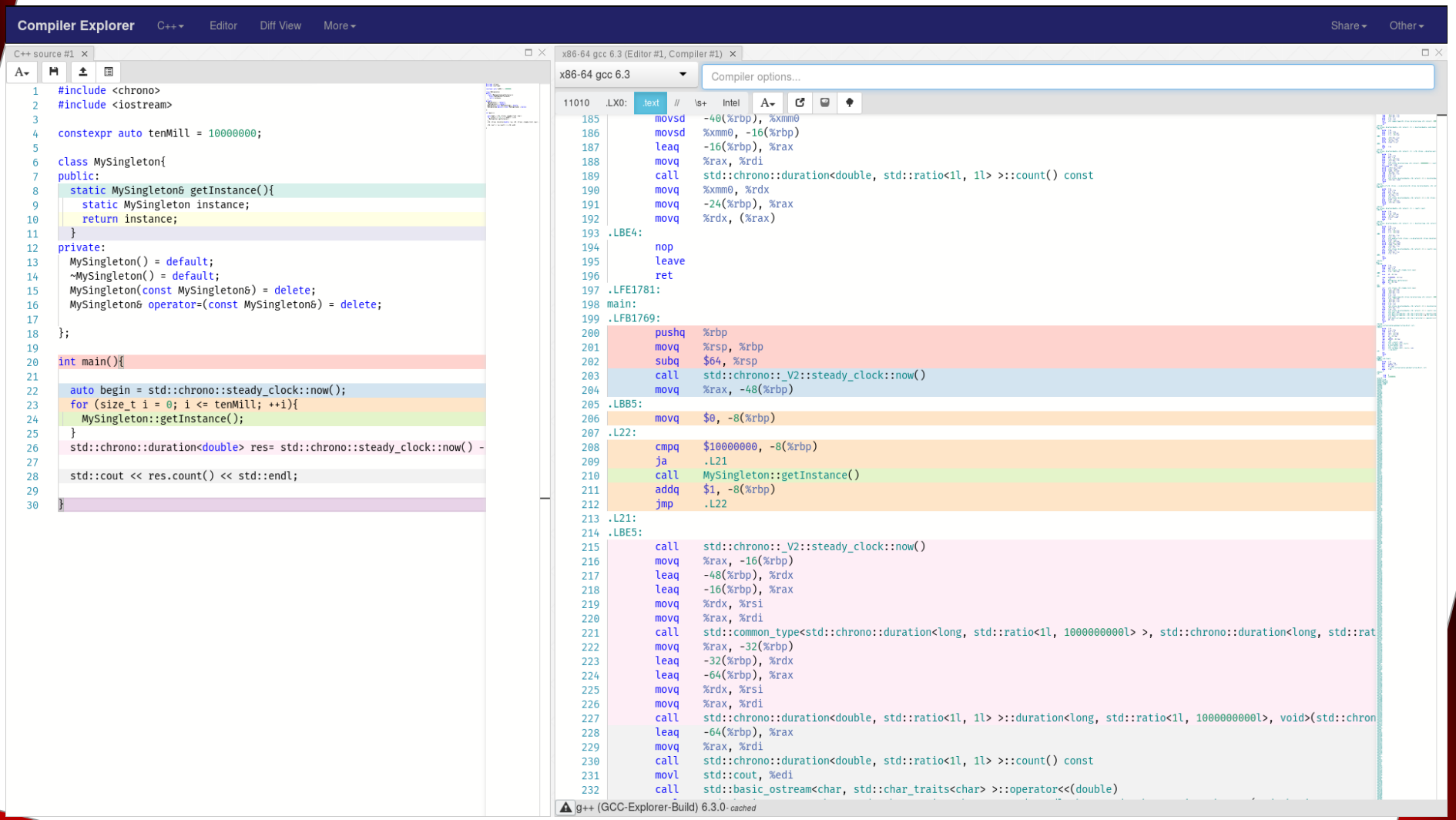
Compiler Explorer von Matt Godbolt

Compiler Explorer erzeugt die Assembler Instruktionen.

- Unterstützt mehr als 60 Compiler
- Stellt Sourcecode den Assemblerinstruktionen gegenüber
- Erlaubt gleichzeitige mehrere Computerarchitekturen darzustellen

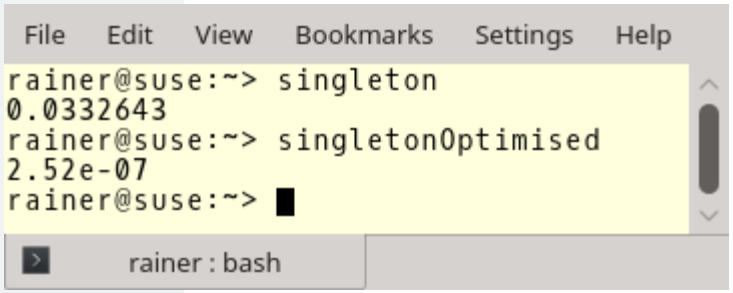
- Vorteile:
  - Erlaubt tiefe Einblicke in den Kompilierungs- und Optimierungsprozess
  - Zeigt, ob ein Ausdruck bereits beim Kompilieren berechnet oder eine Funktion `inline` aufgerufen wird
  - Stellt die Assemblerinstruktionen abhängig vom Optimierungslevel dar

Compiler Explorer von Matt Godbolt



Compiler Explorer von Matt Godbolt

```cpp
#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;

}
```



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    $10000000, -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

```cpp
#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;

}
```

```asm
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_tra
```

Compiler Explorer von Matt Godbolt

- Beispiele
  - gcd
  - Singleton

Blogs

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

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

Rainer Grimm

Training, Coaching und Technologieberatung

www.ModernesCpp.de