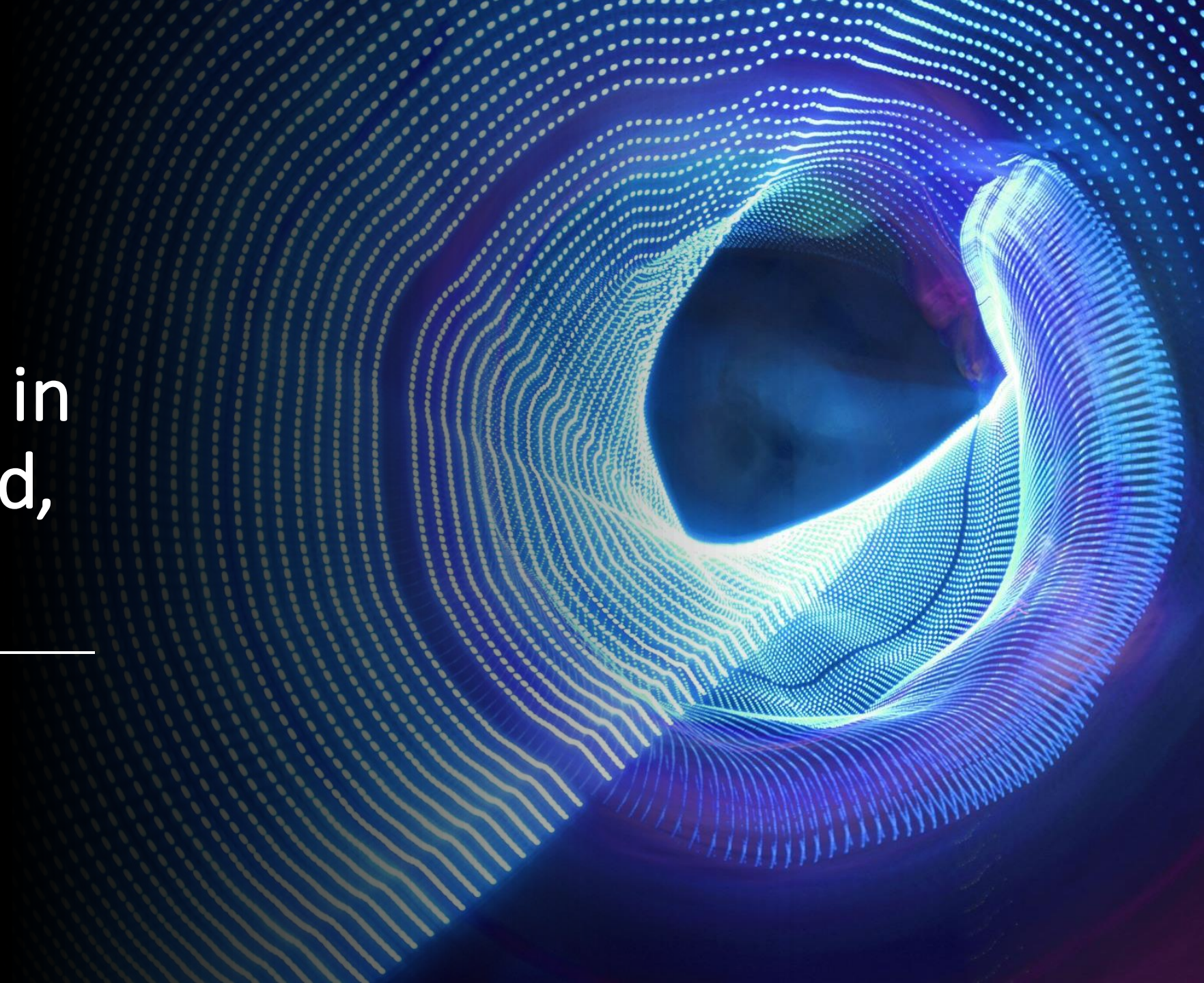


What has my  
compiler done in  
the background,  
let's see ?

---



# Challenges of building/playing with compiler

- Time
- Powerful resources(hardware, memory) needed
- Energy consuming
- Lots of flags to tune to build it
- Different compilers ? Rebuild again?
- Different hardware ? Rebuild again ?
- Close source compilers no way to build them ?
- Sharing codes ?
- Older version access ?

```
14/06/2023 12:16.19 /home/mobaxterm cmake -G Ninja -B ../build/ -S . -DCMAKE_INSTALL_PREFIX:PATH=<PATH TO>/iree-install
-DCMAKE_BUILD_TYPE=Debug -DCMAKE_C_COMPILER=clang -DCMAKE_CXX_COMPILER=clang++ -DCMAKE_C_COMPILER_LAUNCHER=ccache -DCMAKE_CXX_COMPILER_LAUNCHER=ccache
-DIREE_BUILD_SAMPLES=ON -DIREE_BUILD_TESTS=ON -DIREE_BUILD_DOCS=ON -DIREE_INPUT_MHLO:BOOL=OFF -DIREE_INPUT_TORCH:BOOL=OFF -DIREE_INPUT_TOSA:BOOL=OFF
-DIREE_TARGET_BACKEND_CUDA:BOOL=ON -DIREE_TARGET_BACKEND_LLVM_CPU:BOOL=ON -DIREE_TARGET_BACKEND_VMVX:BOOL=ON -DIREE_TARGET_BACKEND_METAL_SPIRV:BOOL=OFF
-DIREE_TARGET_BACKEND_ROCM:BOOL=OFF -DIREE_TARGET_BACKEND_DEFAULTS:BOOL=OFF -DIREE_TARGET_BACKEND_VULKAN_SPIRV:BOOL=OFF -DIREE_TARGET_BACKEND_WEBGPU:BOOL=OFF
-DIREE_DEV_MODE:BOOL=ON -DIREE_COMPILER_BUILD_SHARED_LIBS=ON \ -DIREE_DEFAULT_CPU_LLVM_TARGETS:STRING="X86" -DIREE_BUILD_PYTHON_BINDINGS=ON
-DIREE_ENABLE_LLD=ON -DIREE_ENABLE_ASSERTIONS=ON
```



- Best ways to share/save small code snippets.



Taking a picture of  
your code



# Demo examples

- <https://godbolt.org/z/oKKizT88o>
- Python vs c code and same output.
- <https://godbolt.org/z/PYd83T7Y4>
- Boost vs libcpp(error showing we are using library)
- <https://godbolt.org/z/rvjYqYbeh>
- Performance boost vs libcpp



# Seeing the errors by hovering over them.

The screenshot displays the Compiler Explorer web application. The top navigation bar includes the 'COMPILER EXPLORER' logo, a search bar with 'C++ on Sea' and the text 'The international C++ conference in the UK, by the sea', and links for 'Sponsors', 'intel', 'CONAN', 'SolidSands', 'Share', 'Policies', and 'Other'. Below the navigation bar, the main interface is divided into three sections. The left section, titled 'C++ source #1', contains the following C++ code:

```
3 // Using libc++ library
4 #include <vector>
5 #include <iota>
6 #include <numeric>
7
8 int sumWithLibCxx(int n) {
9     std::vector<int> numbers(n);
10    std::iota(numbers.begin(), numbers.end(), 1);
11    return std::accumulate(numbers.begin(), numbers.end(), 0);
12 }
13
14 int main() {
15     int number = 1000000;
16
17     // Measure execution time using libc++
18     auto start = std::chrono::high_resolution_clock::now();
19     int sumLibCxx = sumWithLibCxx(number);
20 }
```

The right section, titled 'Executor x86-64 clang 16.0.0 (C++, Editor #1)', shows the compiler output. It indicates that the program could not be executed and provides the following error messages:

```
Compiler returned: 1
Compiler stderr
<source>:9:10: error: no member named 'vector' in namespace 'std'
    std::vector<int> numbers(n);
    ~~~~~^
<source>:9:20: error: expected '(' for function-style cast or type construction
    std::vector<int> numbers(n);
    ~~~~~^
<source>:9:22: error: use of undeclared identifier 'numbers'
    std::vector<int> numbers(n);
    ~~~~~^
<source>:10:15: error: use of undeclared identifier 'numbers'
    std::iota(numbers.begin(), numbers.end(), 1);
    ~~~~~^
```

The bottom section, titled 'C++ source #3', is currently empty. The status bar at the bottom indicates the executor is 'x86-64 gcc 13.1 (C++, Editor #3)'.

C++ source #1

Save/Load Add new... Vim CppInsights Quick-bench C++

```
14
15 int main() {
16     int number = 1000000;
17
18     // Measure execution time using libc++
19     auto start = std::chrono::high_resolution_clock::now();
20     int sumLibCxx = sumWithLibCxx(number);
21     auto end = std::chrono::high_resolution_clock::now();
22     auto durationLibCxx = std::chrono::duration_cast<std::chrono::microseconds>(end - start);
23
24     // Output the results
25     std::cout << "Sum using libc++: " << sumLibCxx << std::endl;
26     std::cout << "Execution time using libc++: " << durationLibCxx << " microseconds" << std::endl;
27
28     return 0;
29 }
30
```

Executor x86-64 clang 16.0.0 (C++, Editor #1)

Wrap lines Libraries Overrides Compilation Arguments Stdin

x86-64 clang 16.0.0

Compiler options...

Program returned: 0

Program stdout

Sum using libc++: 1784293664

Execution time using libc++: 23632 microseconds

x86-64 clang 16.0.0 - cached

C++ source #3

Save/Load Add new... Vim CppInsights Quick-bench C++

```
23 auto start = std::chrono::high_resolution_clock::now();
24 auto end = std::chrono::high_resolution_clock::now();
25 auto durationLibCxx = std::chrono::duration_cast<std::chrono::microseconds>(end - start);
26
27 // Measure execution time using Boost
28 start = std::chrono::high_resolution_clock::now();
29 int sumBoost = sumWithBoost(number);
30 end = std::chrono::high_resolution_clock::now();
31 auto durationBoost = std::chrono::duration_cast<std::chrono::microseconds>(end - start);
32
33 // Output the results
34 std::cout << "Sum using Boost: " << sumBoost << std::endl;
35 std::cout << "Execution time using Boost: " << durationBoost << " microseconds" << std::endl;
36
37 return 0;
38 }
39
```

Executor x86-64 clang 16.0.0 (C++, Editor #3)

Wrap lines Libraries (1) Overrides Compilation Arguments Stdin

x86-64 clang 16.0.0

Compiler options...

Program returned: 0

Program stdout


Sum using Boost: 1784293664

Execution time using Boost: 59406 microseconds

x86-64 clang 16.0.0 - cached

Libraries changed



C++ source #1 

A ▾

Save/Load

+ Add new... ▾

Vim

CppInsights

Quick-bench

C++ ▾

14

15 `int main() {`

16 `int number = 1000000;`

17

18 `// Measure execution time using libc++`

19 `auto start = std::chrono::high_resolution_clock::now();`

20 `int sumLibCxx = sumWithLibCxx(number);`

21 `auto end = std::chrono::high_resolution_clock::now();`

22 `auto durationLibCxx = std::chrono::duration_cast<std::chrono::microseconds>(end - start);`

23

24 `// Output the results`

25 `std::cout << "Sum using libc++: " << sumLibCxx << std::endl;`


26 `std::cout << "Execution time using libc++: " << durationLibCxx << " microseconds" << std::endl;`

27

28 `return 0;`


29 `}`


30

Executor x86-64 gcc (trunk) (C++, Editor #1) 

A ▾

☐ Wrap lines

 Libraries



 Overrides

Compilation

> Arguments

Stdin

Com


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


Program returned: 0


Program stdout


Sum using libc++: 1784293664

Execution time using libc++: 39946 microseconds

 x86-64 gcc (trunk)

 - cached



C++ source #3 

A ▾

Save/Load

+ Add new... ▾

Vim

CppInsights

Quick-bench

C++ ▾

23 `auto start = std::chrono::high_resolution_clock::now();`

24 `auto end = std::chrono::high_resolution_clock::now();`

25 `auto durationLibCxx = std::chrono::duration_cast<std::chrono::microseconds>(end - start);`

26

27 `// Measure execution time using Boost`

28 `start = std::chrono::high_resolution_clock::now();`

29 `int sumBoost = sumWithBoost(number);`

30 `end = std::chrono::high_resolution_clock::now();`

31 `auto durationBoost = std::chrono::duration_cast<std::chrono::microseconds>(end - start);`

32

33 `// Output the results`

34 `std::cout << "Sum using Boost: " << sumBoost << std::endl;`


35 `std::cout << "Execution time using Boost: " << durationBoost << " microseconds" << std::endl;`

36

37 `return 0;`


38 `}`


39

Executor x86-64 clang 16.0.0 (C++, Editor #3) 

A ▾

☐ Wrap lines

 Libraries (1)



 Overrides

Compilation

> Arguments

Stdin

Com


x86-64 clang 16.0.0   Compiler options...


Program returned: 0


Program stdout

Sum using Boost: 1784293664

Execution time using Boost: 59406 microseconds

 x86-64 clang 16.0.0

 - cached



Compilers changed  
<https://godbolt.org/z/aWj9djYWd>



# Different compilers different timings!

<https://godbolt.org/z/f9G7M1TGv>

The screenshot displays the Godbolt Compiler Explorer interface with four compiler configurations. Each configuration shows the source code, the compiler selected, and the execution results. The source code is a C++ program that calculates the sum of a vector of integers and measures the execution time using `std::chrono`.

Compiler	Program returned	Sum using libc++	Execution time using libc++
x86-64 gcc (trunk)	0	1784293664	35431 microseconds
x86-64 clang (amd-stg-open)	0	1784293664	47481 microseconds
x86-64 gcc 10.3	0	1784293664	26651 microseconds
x86-64 clang 16.0.0	0	1784293664	26019 microseconds

The source code for all configurations is as follows:




```
#include <iostream>
#include <chrono>
#include <vector>
// Using libc++ library
#include <numeric>

int sumWithLibCxx(int n) {
    std::vector<int> numbers(n);
    std::iota(numbers.begin(), numbers.end(), 1);
    return std::accumulate(numbers.begin(), numbers.end(), 0);
}

int main() {
    int number = 1000000;
    // Measure execution time using libc++
    auto start = std::chrono::high_resolution_clock::now();
    int sum = sumWithLibCxx(number);
    auto end = std::chrono::high_resolution_clock::now();
    std::cout << sum << "\n";
    std::cout << "Execution time: " << std::chrono::duration_cast<std::chrono::microseconds>(end - start).count() << " microseconds\n";
}
```

COMPILER EXPLORER

Do you have any suggestions, requests or bug reports?  
Feel free to [contact us](#) at anytime

Sponsors    Share

++ source #1

Save/Load Add new... Vim CppInsights Quick-bench C++

```
1 #include <iostream>
2 #include <chrono>
3 #include <vector>
4 // Using libc++ library
5 #include <numeric>
6
7
8 int sumWithLibCxx(int n) {
9     std::vector<int> numbers(n);
10    std::iota(numbers.begin(), numbers.end(), 1);
11    return std::accumulate(numbers.begin(), numbers.end(), 0);
12 }
13
14
15 int main() {
16     int number = 1000000;
17 }
```

Executor x86-64 icx 2022.2.1 (C++, Editor #1)

A Wrap lines Libraries Overrides Compilation Arguments Stdin Compil

x86-64 icx 2022.2.1

Program returned: 0

Program stdout

Sum using libc++: 1784293664

Execution time using libc++: 16977 microseconds

x86-64 icx 2022.2.1 - cached

++ source #2

Save/Load Add new... Vim CppInsights Quick-bench C++

```
1 #include <iostream>
2 #include <chrono>
3 #include <vector>
4 // Using libc++ library
5 #include <numeric>
6
7
8 int sumWithLibCxx(int n) {
9     std::vector<int> numbers(n);
10    std::iota(numbers.begin(), numbers.end(), 1);
11    return std::accumulate(numbers.begin(), numbers.end(), 0);
12 }
13
14
15 int main() {
16     int number = 1000000;
17
18     // Measure execution time using libc++
19     auto start = std::chrono::high_resolution_clock::now();
20     sumWithLibCxx(number);
21     auto end = std::chrono::high_resolution_clock::now();
22     auto duration = std::chrono::duration_cast<std::chrono::microseconds>(end - start);
23 }
```

Executor x86-64 clang 16.0.0 (C++, Editor #2)

A Wrap lines Libraries Overrides Compilation Arguments Stdin Compil

x86-64 clang 16.0.0

Program returned: 0

Program stdout

Sum using libc++: 1784293664

Execution time using libc++: 26019 microseconds

x86-64 clang 16.0.0 - cached

Different architectures different timings!  
<https://godbolt.org/z/hnb6TEhxG>

The image shows the Compiler Explorer interface with two tabs: Python and C++.

**Python Tab:**

```
1 import time
2
3 def sum_with_libcxx(n):
4     numbers = list(range(1, n+1))
5     #print (numbers)
6     return sum(numbers)
7
8 def main():
9     number = 10000
10
11     # Measure execution time using libc++
12     start = time.perf_counter()
13     sum_libcxx = sum_with_libcxx(number)
14     end = time.perf_counter()
15     duration_libcxx = (end - start) * 1000000 # Convert to microseconds
16
17     # Output the results
```

Execution results for Python 3.10:

```
Program returned: 0
Program stdout
Sum using libc++: 50005000
Execution time using libc++: 500.3940050099045 microseconds
```

**C++ Tab:**

```
11     std::iota(numbers.begin(), numbers.end(), 1);
12
13     // std::cout << "Vector contents: ";
14     // for (const auto& number : numbers) {
15     //     std::cout << number << " ";
16     // }
17     // return std::accumulate(numbers.begin(), numbers.end(), 0);
18     return std::accumulate(numbers.begin(), numbers.end(), 0);
19 }
20
21 int main() {
22     int number = 10000;
23
24     // Measure execution time using libc++
25     auto start = std::chrono::high_resolution_clock::now();
26     int sumLibCxx = sumWithLibCxx(number);
27     auto end = std::chrono::high_resolution_clock::now();
28     auto durationLibCxx = std::chrono::duration_cast<std::chrono::microseconds>(end - start);
```

Execution results for x86-64 clang 16.0.0:

```
Program returned: 0
Program stdout
Sum using libc++: 50005000
Execution time using libc++: 294 microseconds
```

Different programming languages  
<https://godbolt.org/z/TM68Paqcd>

# Assembly code

<https://godbolt.org/z/nT7vGfqjG>

The screenshot displays the Godbolt Compiler Explorer interface. The top section shows the browser address bar with the URL <https://godbolt.org/z/nT7vGfqjG>. Below the browser, the Compiler Explorer header includes the Compiler Explorer logo, a search bar, and various navigation links. The main area is divided into three panes:

- Source Code Pane (Left):** Displays C++ source code for a program that measures execution time using `libc++`. The code includes a function `sumWithLibCxx` and a `main` function that sets a number to 10000, measures the time to sum the numbers, and outputs the results. A yellow arrow points from the line `int number = 10000;` in the source code to the corresponding assembly instruction.
- Assembly Pane (Right):** Shows the assembly code generated by the compiler. The assembly includes instructions for pushing the stack frame, moving the stack pointer, and calling the `sumWithLibCxx` function. The instruction `mov DWORD PTR [rbp-4], 10000` is highlighted, corresponding to the `int number = 10000;` line in the source code.
- Output/Execution Pane (Bottom):** Shows the program's output and execution details. The output indicates that the program returned 0 and printed the sum and execution time. The execution time is 326 microseconds.

The bottom status bar shows the compiler used: `x86-64 clang 16.0.0` and the target architecture: `x86-64 gcc 13.1`.



# No filters applied, 18045 lines of assembly!










<https://godbolt.org/z/T3Y175bqd>



<https://godbolt.org/z/PfYne86aW>




This is what a compiler does lately when you just say it to compile something, there is a lot going on in the background.









C++ source #2   Save/Load  Add new... ▾  Vim  CppInsights  Quick-bench

C++ ▾







```
1 #include <iostream>
2 #include <chrono>
3 #include <vector>
4 #include <numeric>
5 int sumWithLibCxx(int n) {std::vector<int> numbers(n);std::iota(numbers.begin(), numbers.end(), 1);
6 }
```

- ⚙ PVS-Studio
- ⚙ clang-format
- ⚙ clang-query (trunk)
- ⚙ clang-tidy (trunk)
- ⚙ include-what-you-use (0.12)
- ⚙ ldd
- ⚙ llvm-cov (trunk)
- ⚙ llvm-mca (trunk)
- ⚙ OSACA (0.5.0)
- ⚙ pahole (trunk)
- ⚙ readelf (trunk)
- ⚙ nm (trunk)

-  Sonar
- ⚙ strings
- ⚙ x86-to-6502

x86-64 gcc 13.1 (Editor #2)  x86-64 gcc 13.1   Output...  Filter...  Libraries  Overrides  Add new...  Add tool...

```
1 sumWithLibCxx(int):
2     push    rbp
3     mov     rbp, rsp
4     push    rbx
```

clang-format x86-64 gcc 13.1 (Editor #2, Compiler #1)    Wrap lines  Arguments  Stdin

```
#include <chrono>
#include <iostream>
#include <numeric>
#include <vector>
int sumWithLibCxx(int n) {
    std::vector<int> numbers(n);
    std::iota(numbers.begin(), numbers.end(), 1);
    return std::accumulate(numbers.begin(), numbers.end(), 0);
}
int main() {
    int number = 10000;
    auto start = std::chrono::high_resolution_clock::now();
    int sumLibCxx = sumWithLibCxx(number);
    auto end = std::chrono::high_resolution_clock::now();
    auto durationLibCxx =
        std::chrono::duration_cast<std::chrono::microseconds>(end - start)
        .count();
    std::cout << "Sum using libc++: " << sumLibCxx << std::endl;
    std::cout << "Execution time using libc++: " << durationLibCxx
        << " microseconds" << std::endl;
    return 0;
}
```

Tooling for free.

<https://godbolt.org/z/PdGG1xqMT>

Too obfuscated code.

```

#include **/<time.h>
#include <ncurses.h>
# include <stdlib.h>

/**
y;y<H&& /*...Semi-Automatic.* /y< p/W+2;\
y++)for(x=p% W,x=!!/*..MineSweeper...*/x;x<W&& x<p%W+2;x++)
#define _(x,y)COLOR_#x,COLOR_#y /* click / (R)estart / (Q)uit */
#define Y(n)attrset(COLOR_PAIR(n)),mvprintw(/* IOCCC2019 or IOCCC2020 */
typedef int I;I*M,W,H,S,C,E,X,T,c,p,q,i,j,k;char G[]=" x",U[256];I F(I p){ I
r=0,x,y=p/W,q;0()q=y*W+x,r+=M[q]^=p-q?(M[q]&16)<<8:0;return r;}I K(I p
,I f,I g){ I x=(g+ f/256)%16-(f+g/256)%16,y=p/W,c=0,n=g/4096
,m=x==n?0:x==g /16%16-f/16%16-n?256:-1; if(m+1)0()if
((4368&M[n=y*W +x])==4112){ M[c=1,n]=(M[n]&~16)|m; }
return c;}void D() {I p,k,o=0,n=C,m=0,q=0;if(LINES-1<H
(),Y(4)LINE/2,COLS/2-16,"Make the ter\
minal bigger!");else{for (p=0;p<S;o+=k==3,Y(k)p/W+1,p%W*2,G),p++)G[1]="
"!..12345678"[k=E?256&M[p ]?n--,2:E-2||M[p]%2<1?M[p]&16?q=p,m++,3:4+F(p)%16:
1:3];k=T+time(0);T=o||T>=0||E-1?T:k;k=T?0?k:T;Y(7)0,0,"%03d%*s%03d",n>999?999:n,W*
2-6,"",k>999?999:k);Y(9)0,W-1,E>1?"X-("E-1||o?"-":"8-")");M[q]|=256*(n==m&&n); }
refresh();}short B[]={_(RED,BLACK),_(WHITE,BLUE),_(GREEN,RED),_(MAGENTA,YELLOW),_(
CYAN,RED)};I main(I A,char**V){MEVENT e;FILE*f;srand(time(0));initscr();for(start\
_color();X<12;X++){init_pair(X+1,B[X&&X<10?X-1:2],B[X?X<3?2:1:0]);}noecho();cbreak
();timeout(9);curs_set(0);keypad(stdscr,TRUE);for(mousemask(BUTTON1_CLICKED|BUTTO\
N1_RELEASED,0);){S=A<2?f=0,W=COLS/2,H=LINE-1,C=W*H/5,0:fscanf(f=fopen(V[A-1],"r"
),"%d %d %d",&W,&H,&C)>3; ;S+=W*H;M=realloc(M,S*sizeof(I)*2);for(i=0
;i<S;i++)!f?M[i]=i,i&&(k=M[j=rand()i],M[j]=M[i],M[i]=k):fscanf(f,
"%d",M+i);if(f)fclose(f);T=E=X=0;for(clear();D(),c=getch(),c-'r'
&&(c-KEY_RESIZE||E);){ if(c=='q'){ return(endwin(),0); }if(c==
KEY_MOUSE&&getmouse(&e)==OK&&e.x/2<W&&e.y<=H){if(!e.y&&(W-2<e.x&&
e.x<W+2)){break;}p=e.x/2+e.y*W-W;if(p>=0){if(!E){for(i=0;i<S;i++)M[S+M
[i]]=i,M[i]=16+(M[i]<C);C-=M[p]&1;M[p]=16;E=1;T=-time(0);}if(E<2)M[p]&=(M[p]
&257)==1?T+=time(0),E=2,273:257;}}for(p=0;p<S&&E==1;M[p++]&=273){}for(i=
(X+S-1)%S;E==1&&i!=X;X=(X+1)%S){if(!(M[p=M[X+S]]&272)){if(K(p,c=F(p)
,0)){goto N;}} for(k=p/W-2,k=k<0?0:k;k<p/W+3&&k <H;k++)for(j=
p%W-2,j =j<0?0;j;j<W&&j<p%W+3;if (!(M[q=
k*W +j++]&272)){ if(K(p,
(q))){ goto N; }F(q)
; }F(p); }N:; } } }
/*(c)Yusukse Endoh*/

```

And it works for obfuscated codes as well!

IOCCC competition

- [https://godbolt.org/z/eodKr\\_h3vc](https://godbolt.org/z/eodKr_h3vc)
- <https://www.ioccc.org/2020/endoh2/prog.c>
- CV23 ? -> what's this
- How important is indented
- And readable code
- Importance of tools for programming

## Works

```
#include<stdio.h>
#include<stdlib.h>
#define M malloc(sizeof(
#define R fread(x,sizeof(float),Z*w,f)
#define E free(

char*G=
" "
" ""
"@(*AP( "
"
"
"
"
"
"
"0 #@ . \"C^*/XH\"Q2_U(D&(5*E0C 4B54DV X \"!9A@ 89H$ !\"
?R @#%)14,A#9)24$B 1\"
\" )|0,$Y145$R Y145$^&!81T! #9)24DV#Y)24DP ;&P -C4 $$$B@ |04%|04 @4(D$ #!(1\"
\"4 @#U574$^~0D)! &Z1D9'_$*!@8%^Z!@8'_(&1D9'_(\\\"0D)#_&Z1D8%^/\\0$!#_ '\\!_X$/X! 0\"
\"$&(-$*!# _ $! 0'_/] , $$_/\\&&&$ 'Z!@8%^&\\\"0D)#_ 'V\\\"A8%^&>8D)#_ $Z1D9$B(\\\" _X\\\" /X\"
\"! 0'^/ , PSP/P#' /\\/,<H$\\\"C', @'R#, &AD8F'(&!_P ,,,, _X&!! @0\\\" 0 \"
\"$! 0$! @0( \\5%14\\\" X1$0G_!$1$1$.//\\2$1$. T5%14. \\\"0D'\\\" 'Y\")\"
\"144X \\0$ C_ !7Q 7A$! $2# 3_ $!_H \\0'Q ? \\0\"
\"$! ? X1$1$.#A$1\\\"1_'](1$0X @0$ @? (5%14)\"
\" $!>X0!\\\"\\\" 0$>!@& 088!X!#@>$>\"
\"!$*! H1#X)|04X|$9%1,1\"
\"(&!;A 0 _P \"
\" ! 0;H&!"
\" @$\\\"""
\"! /"
\"\\0": int s,p,e,c,t,r,a,l; /* leakage*/
```

```

int
F(int o
,int n, int t
){char*p=(o*(136-o)
>=1260)*7*(o-9)+G; for(
n=! (o=-1); n=*p+++64*n-*G
,(o+=3)>>>2<5-t; ); return 255
&n>>2*o%8; } float L[0x400];int
T(float a,float b,float l,float*e)
{return L[0x3ff]||!(e[1]=b*a+1*e)||
!T(1*a-b**e,b,l,e+1); } float f(int x)
{int w=s<<!!T(1,6.135885e-3,-1.882472e-5
+1,L); x=(x%w+w)%w<<9; return(L[x/s]*s-x%
s)+L[x/s+1]*(x%8)/s; } int C(float*h,float*
i,int r,int P){ int x,y,c; for(y=0; h<i; ++y,h
+=r){ c=P<1; for(*h=x=0; x<8; c+=(1&P)>>>8&P+(*
h)=f(e*(1+a*(2*x+1))*y)*f(s/2-e*a*y+e*a*y*y*(t/p
)/(s/p))))),++x); float z=f(y*t)*f(y*t); *h=z*(1+3*
z+2*z*z)/6**h/c; } return h-i-r+1; } void d(int w,
int n,FILE*f){int Z=s/t; int S[5]; float**P=M float*
)*c;for(r=0;r<c;++r){ P[r]=M float)*Z; } C(P[r],P[r]
+Z,l,r); } float*x=M float)*Z*w); for( ; ){ for(r=0;
++r){ int o=1; for(R,a=256; o&&-a; )for(o=t=0; t<Z;
t!)=x[w*t+n],++t); S[r]=a; } if(R-Z*w)break; for(R=a= -
a<11?22:1)<127; )for(r=-1; ++r<5&&F(a,l,r))=S[r]; );if(a
a); } for( ; --c; E P[c]); E P; fclose(f);}int main(int
55;c=256; t=75; r=q>1?v[1][0]*c+v[1][1]:0; a=7;1=16;if(r
(v[2]), atoi(v[3]), q<5?stdin:fopen(v[4], "r")); return 0; } char*H=malloc(c),*
"Usage\n" %s ["-h] [-d # cid] file1 file2 ... (or stdin)+(q-1&r==11624
w=2>q?1:q -1; FILE**f=M FILE*)w); f[0]=stdin; if(q>1)for(r=0; r<w; f[r]=fopen(v[r
+1],"r"),++r); int l=s/t
*w; float *b=M float)*1 ),t=b+1; int i=M int)*w); for( ; ){ int x=h&1>ungetc(* H++,*f); for(r =w; r--;)
{ i[r]=getc (f[r]); if(i[r]<0)i[r]=++x; if(x==w)break; while(++r<6) while(C(b+++t,w,f*(i++,r< 5,r))))); i=
w; fwrite(b=w,sizeof(float),1,stdout); } fflush(stdout); } while(fclose(f[--w]),w); E f); E b); E i); E h);}

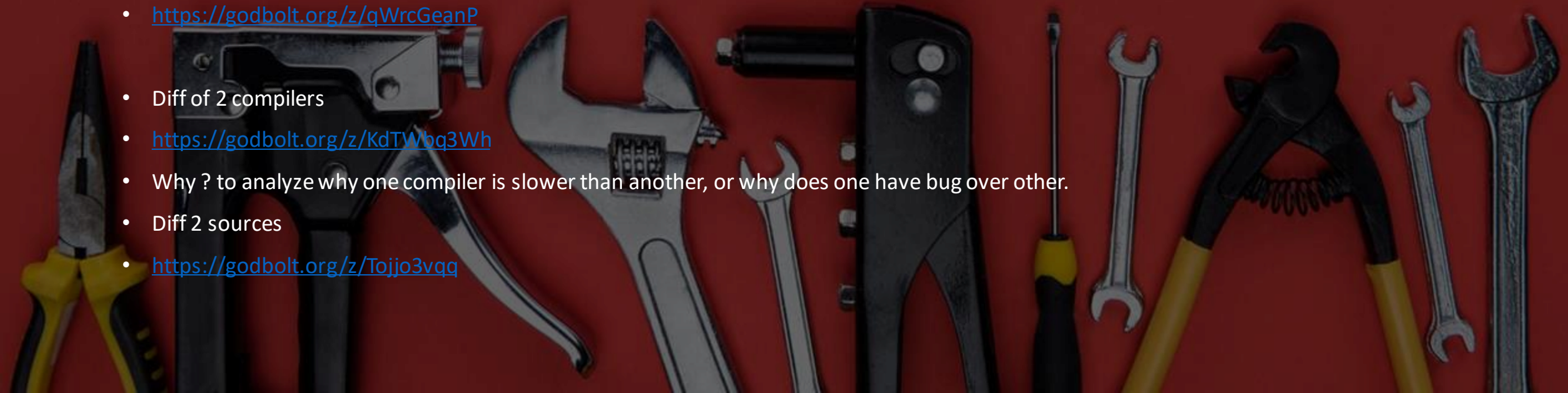
```



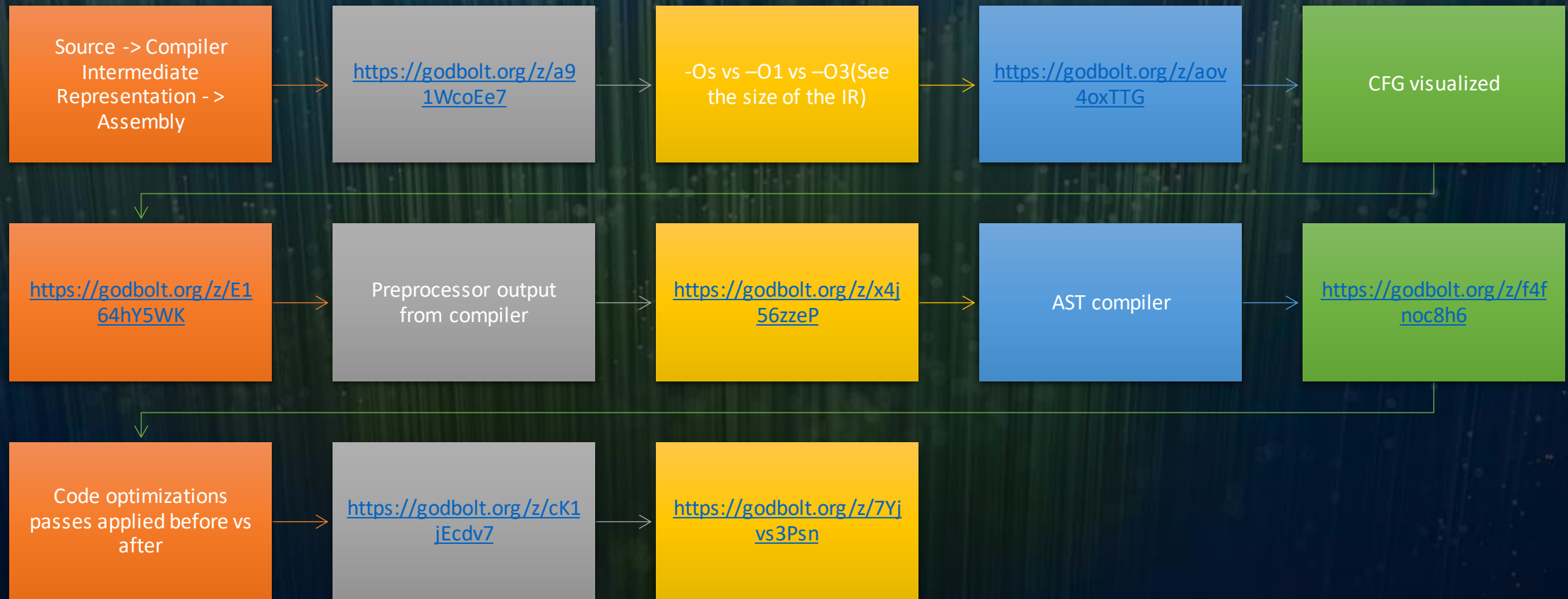
# More tooling

<https://godbolt.org/z/eozxe7q6q> (nm tool)

- 
- Color change of CE – if you feel the color of the day works best for you.
  - <https://godbolt.org/z/qWrcGeanP>
  - Diff of 2 compilers
  - <https://godbolt.org/z/KdTWbq3Wh>
  - Why ? to analyze why one compiler is slower than another, or why does one have bug over other.
  - Diff 2 sources
  - <https://godbolt.org/z/Tojjo3vqq>

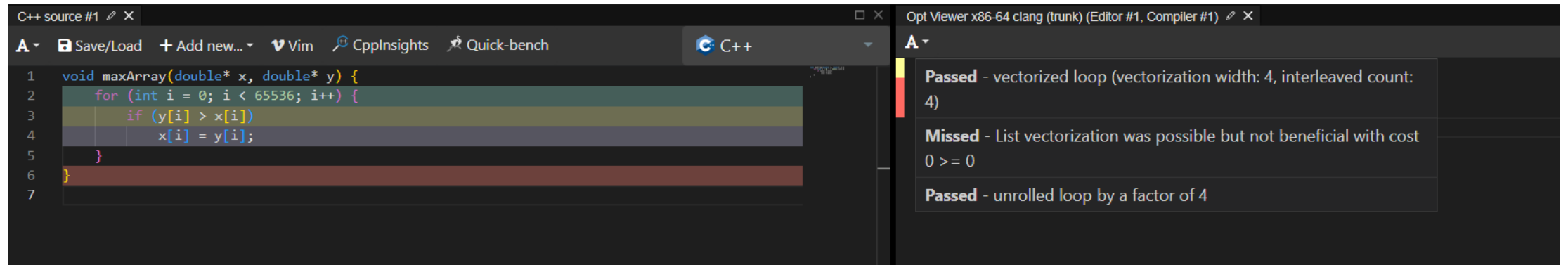


# The journey from source to assembly.



# Optimization viewer

<https://godbolt.org/z/c91T1zecb>



The screenshot displays the Godbolt optimization viewer interface. On the left, the C++ source code is shown in a dark-themed editor. The code defines a function `maxArray` that takes two double arrays, `x` and `y`, and iterates over them, comparing elements and updating `x[i]` if `y[i]` is greater. The loop runs from `i = 0` to `i < 65536`. On the right, the optimization results for the x86-64 clang (trunk) compiler are displayed. The results are organized into a table with three rows, each showing a status (Passed or Missed) and a description of the optimization.

Status	Description
Passed	vectorized loop (vectorization width: 4, interleaved count: 4)
Missed	List vectorization was possible but not beneficial with cost $0 \geq 0$
Passed	unrolled loop by a factor of 4

Understanding assembly  
was not so easy for me  
when I started.

x86-64 gcc 13.

A ▾ ⚙ Output.

1 square(i

2

3

4

5

6

7

8

9

Copies the second operand (source operand) to the first operand (destination operand). The source operand can be an immediate value, general-purpose register, segment register, or memory location; the destination register can be a general-purpose register, segment register, or memory location. Both operands must be the same size, which can be a byte, a word, a doubleword, or a quadword.

More information available in the context menu.

mov DWORD PTR [rbp-4], 10

mov eax, DWORD PTR [rbp-20]

imul eax, eax

pop rbp

ret



# About me - Matt Godbolt

Hi there, I'm Matt Godbolt — pleased to meet you. Please, come in, sit down, make yourself at home.



Author behind this helpful tool.

Let me tell you a bit about myself. I was born on August 16<sup>th</sup> 1976 to Richard and Christine Godbolt. My first computer came at age 8, a [48k Sinclair Spectrum](#), you know the ones with rubber keys. I was captivated, and remember fondly one Christmas getting my poor long-suffering mother to read out long program listings while I typed them in, hours of work and then a fairly pitiful result like a Union Jack flag

- 30+ languages supported (<https://godbolt.org/api/languages>)

## COMPILER STATS

- 400+ compilers
- 250+ GB

Ada	Analysis	Assembly	C	C++	Clean	Cppx
CUDA	D	Fortran	Go	Haskell	ispc	LLVM IR
OCaml	Pascal	Rust	Swift	Zig		

# What language do people use the most on CE ?

- <https://ce.grafana.net/public-dashboards/326d9aa2606b4efea25f4458a4c3f065?orgId=0&refresh=1m>

Compilations in the Last Week (top 20)

c++ 2,752,795	c 538,897	rust 129,379	python 41,401	java 15,486
assembly 15,083	zig 14,567	fortran 12,785	cuda 11,720	d 10,036
swift 8,996	llvm 8,527	csharp 6,381	go 5,391	haskell 4,817
circle 4,223	carbon 2,517	javascript 2,491	cppx 2,397	cpp2_cppfront 2,331