Sorting Values

Sorting in C

```c
#include <cstdio>
#include <cstdlib>

int compare_ints(const void *lhs, const void *rhs) {
  return (int*)lhs - (int*)rhs;
}

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  qsort(values, 10, sizeof(int), &compare_ints);
  return values[0];
}

int main() {
  printf("%f", get_lowest());
}
```

https://godbolt.org/z/ZmGFdB

Sorting in C

```c
#include <cstdio>
#include <cstdlib>

int compare_ints(const void *lhs, const void *rhs) {
  return (int*)lhs - (int*)rhs;
}

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1}; ///
  qsort(values, 10, sizeof(int), &compare_ints);
  return values[0];
}

int main() {
  printf("%f", get_lowest());
}
```

https://godbolt.org/z/KKceqJ

Sorting in C

```
#include <cstdio>
#include <cstdlib>

int compare_ints(const void *lhs, const void *rhs) {
  return (int*)lhs - (int*)rhs;
}

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  qsort(values, 10, sizeof(int), &compare_ints); ///
  return values[0];
}

int main() {
  printf("%f", get_lowest());
}
```

https://godbolt.org/z/PxhkdB

Sorting in C

```
#include <cstdio>
#include <cstdlib>

int compare_ints(const void *lhs, const void *rhs) {
  return (int*)lhs - (int*)rhs; ///
}

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  qsort(values, 10, sizeof(int), &compare_ints);
  return values[0];
}

int main() {
  printf("%f", get_lowest());
}
```

https://godbolt.org/z/H5xYSe

Sorting in C

```
#include <cstdio>
#include <cstdlib>

int compare_ints(const void *lhs, const void *rhs) {
  return (int*)lhs - (int*)rhs;
}

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  qsort(values, 10, sizeof(int), &compare_ints);
  return values[0]; /// return lowest
}

int main() {
  printf("%f", get_lowest());
}
```

https://godbolt.org/z/C65bd9

Sorting in C

```
#include <cstdio>
#include <cstdlib>

int compare_ints(const void *lhs, const void *rhs) {
  return (int*)lhs - (int*)rhs;
}

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  qsort(values, 10, sizeof(int), &compare_ints);
  return values[0];
}

int main() {
  printf("%f", get_lowest()); /// What's printed?
}
```

https://godbolt.org/z/W86Yew

Sorting in C

```
#include <cstdio>
#include <cstdlib>

int compare_ints(const void *lhs, const void *rhs) {
  return (int*)lhs - (int*)rhs;
}

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  qsort(values, 10, sizeof(int), &compare_ints);
  return values[0];
}

int main() {
  printf("%f", get_lowest()); /// 0.000000 Why?
}
```

https://godbolt.org/z/x7HteD

Sorting in C

```
#include <cstdio>
#include <cstdlib>

int compare_ints(const void *lhs, const void *rhs) {
  return (int*)lhs - (int*)rhs;
}

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  qsort(values, 10, sizeof(int), &compare_ints);
  return values[0];
}

int main() {
  printf("%i", get_lowest()); /// What is printed?
}
```

https://godbolt.org/z/5wqWSC

Sorting in C

```
#include <cstdio>
#include <cstdlib>

int compare_ints(const void *lhs, const void *rhs) {
  return (int*)lhs - (int*)rhs;
}

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  qsort(values, 10, sizeof(int), &compare_ints);
  return values[0];
}

int main() {
  printf("%i", get_lowest()); /// 1341. Why?
}
```

https://godbolt.org/z/D9eUg-

Sorting in C

```
#include <cstdio>
#include <cstdlib>

int compare_ints(const void *lhs, const void *rhs) {
  return (int*)lhs - (int*)rhs; /// comparing pointers, not values
}

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  qsort(values, 10, sizeof(int), &compare_ints);
  return values[0];
}

int main() {
  printf("%i", get_lowest()); /// 1341. Why?
}
```

https://godbolt.org/z/rqKdrR

Sorting in C

```
#include <cstdio>
#include <cstdlib>

int compare_ints(const void *lhs, const void *rhs) {
  return *(int*)lhs - *(int*)rhs; ///
}

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  qsort(values, 10, sizeof(int), &compare_ints);
  return values[0];
}

int main() {
  printf("%i", get_lowest()); /// What is printed?
}
```

https://godbolt.org/z/aVt6SW

Sorting in C

```
#include <cstdio>
#include <cstdlib>

int compare_ints(const void *lhs, const void *rhs) {
  return *(int*)lhs - *(int*)rhs;
}

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  qsort(values, 10, sizeof(int), &compare_ints);
  return values[0];
}

int main() {
  printf("%i", get_lowest()); /// 11. Why?
}
```

https://godbolt.org/z/k72s2J

Sorting in C

```cpp
#include <cstdio>
#include <cstdlib>

int compare_ints(const void *lhs, const void *rhs) {
  return *(int*)lhs - *(int*)rhs;
}

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  qsort(values, 10, sizeof(int), &compare_ints); /// wrong length
  return values[0];
}

int main() {
  printf("%i", get_lowest()); /// 11. Why?
}
```

https://godbolt.org/z/n4eSSy

Sorting in C

```
#include <cstdio>
#include <cstdlib>

int compare_ints(const void *lhs, const void *rhs) {
  return *(int*)lhs - *(int*)rhs;
}

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  qsort(values, 11, sizeof(int), &compare_ints); ///
  return values[0];
}

int main() {
  printf("%i", get_lowest()); /// What is printed?
}
```

https://godbolt.org/z/wa5p99

Sorting in C

```
#include <cstdio>
#include <cstdlib>

int compare_ints(const void *lhs, const void *rhs) {
  return *(int*)lhs - *(int*)rhs;
}

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  qsort(values, 11, sizeof(int), &compare_ints);
  return values[0];
}

int main() {
  printf("%i", get_lowest()); /// 1!
}
```

https://godbolt.org/z/gEaNVC

Sorting in C++

Sorting in C++

```cpp
#include <cstdio>
#include <cstdlib> ///

int compare_ints(const void *lhs, const void *rhs) {
  return *(int*)lhs - *(int*)rhs;
}

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  qsort(values, 11, sizeof(int), &compare_ints); ///
  return values[0];
}

int main() {
  printf("%i", get_lowest());
}
```

https://godbolt.org/z/B9t4h6

Sorting in C++

```cpp
#include <cstdio>
#include <algorithm> ///

int compare_ints(const void *lhs, const void *rhs) {
  return *(int*)lhs - *(int*)rhs;
}

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  std::sort(std::begin(values), std::end(values)); ///
  return values[0];
}

int main() {
  printf("%i", get_lowest());
}
```

https://godbolt.org/z/pAVU27

Sorting in C++

```cpp
#include <cstdio>
#include <algorithm>

int compare_ints(const void *lhs, const void *rhs) { ///
  return *(int*)lhs - *(int*)rhs;                    ///
}                                                    ///

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  std::sort(std::begin(values), std::end(values));
  return values[0];
}

int main() {
  printf("%i", get_lowest());
}
```

https://godbolt.org/z/-ax7Yn

Sorting in C++

```cpp
#include <cstdio>
#include <algorithm> ///

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  std::sort(std::begin(values), std::end(values));
  return values[0];
}

int main() {
  printf("%i", get_lowest()); ///
}
```

https://godbolt.org/z/ynbyNP

Sorting in C++

```cpp
#include <cstdio>
#include <algorithm> ///

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  std::sort(std::begin(values), std::end(values));
  return values[0];
}

int main() {
  std::cout << get_lowest() << '\n'; ///
}
```

https://godbolt.org/z/pxGZNA

Sorting in C++

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

int get_lowest() {
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  std::sort(std::begin(values), std::end(values));
  return values[0];
}

int main() {
  std::cout << get_lowest() << '\n';
}
```

https://godbolt.org/z/xe5e9U

But If We Only Need The Lowest Element?

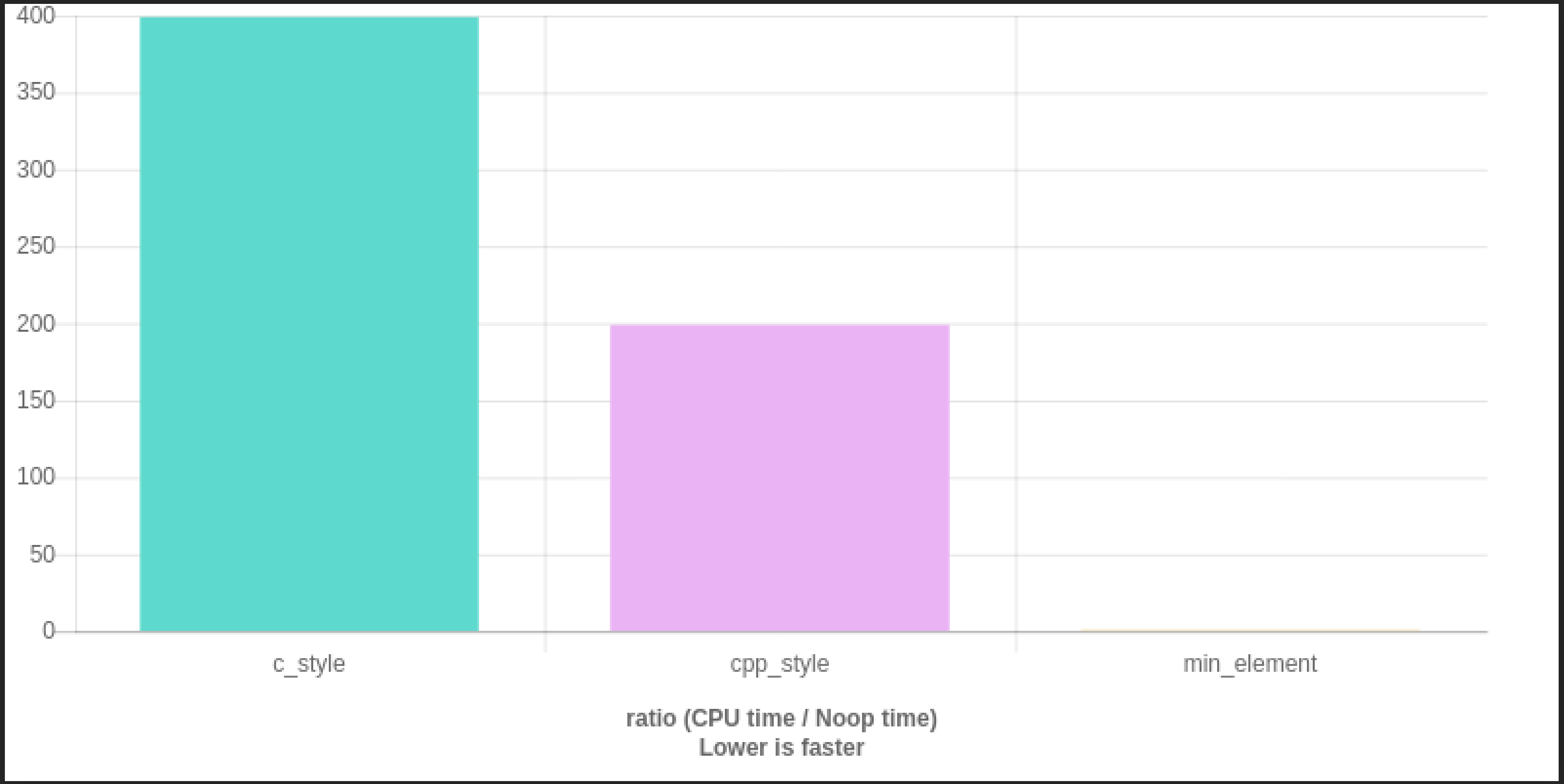
Sorting in C++

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

int get_lowest() {
  const int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  return *std::min_element(std::begin(values), std::end(values));
}

int main() {
  std::cout << get_lowest() << '\n';
}
```

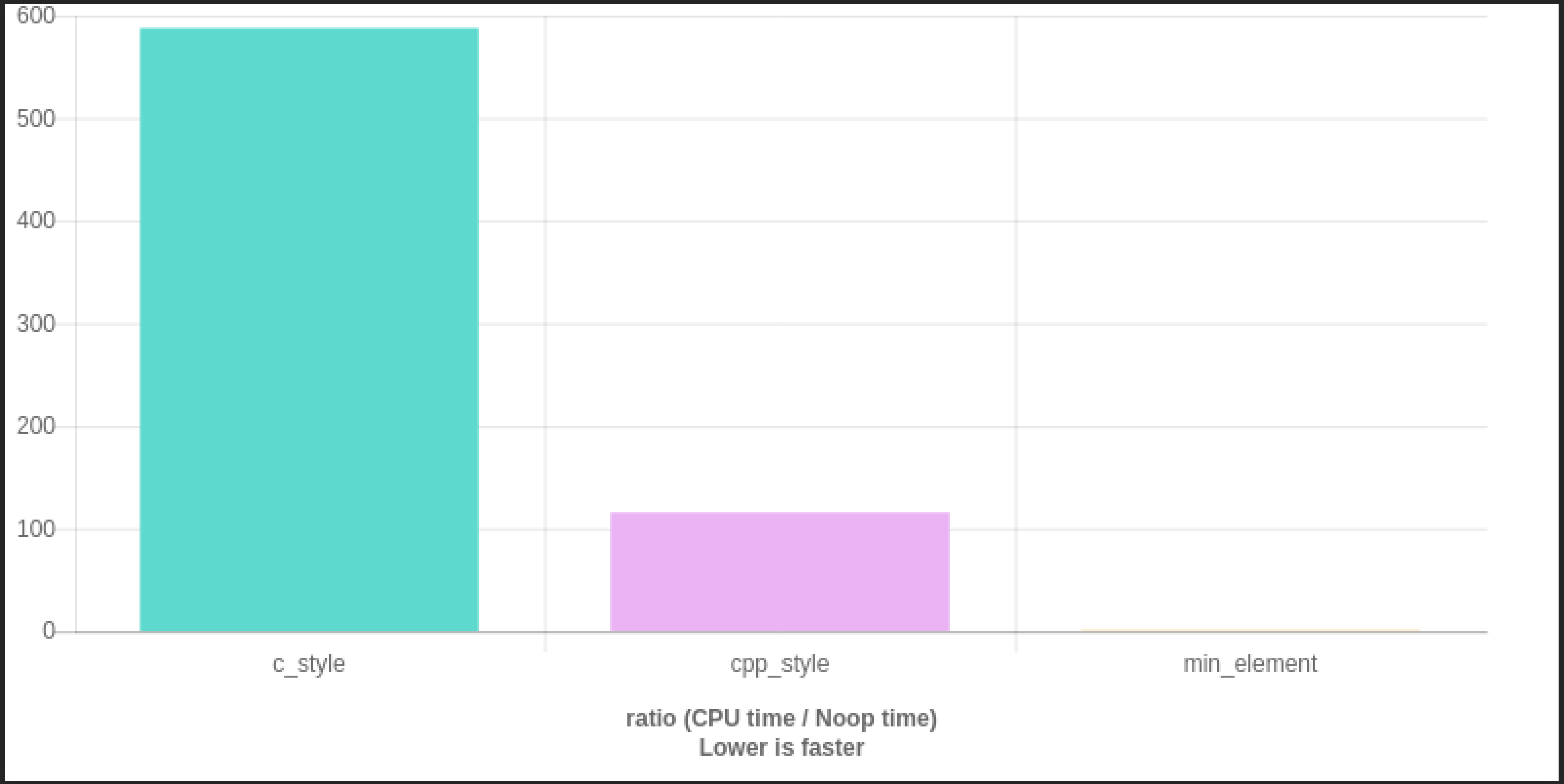
https://godbolt.org/z/PwK7SK

Comparison of Options

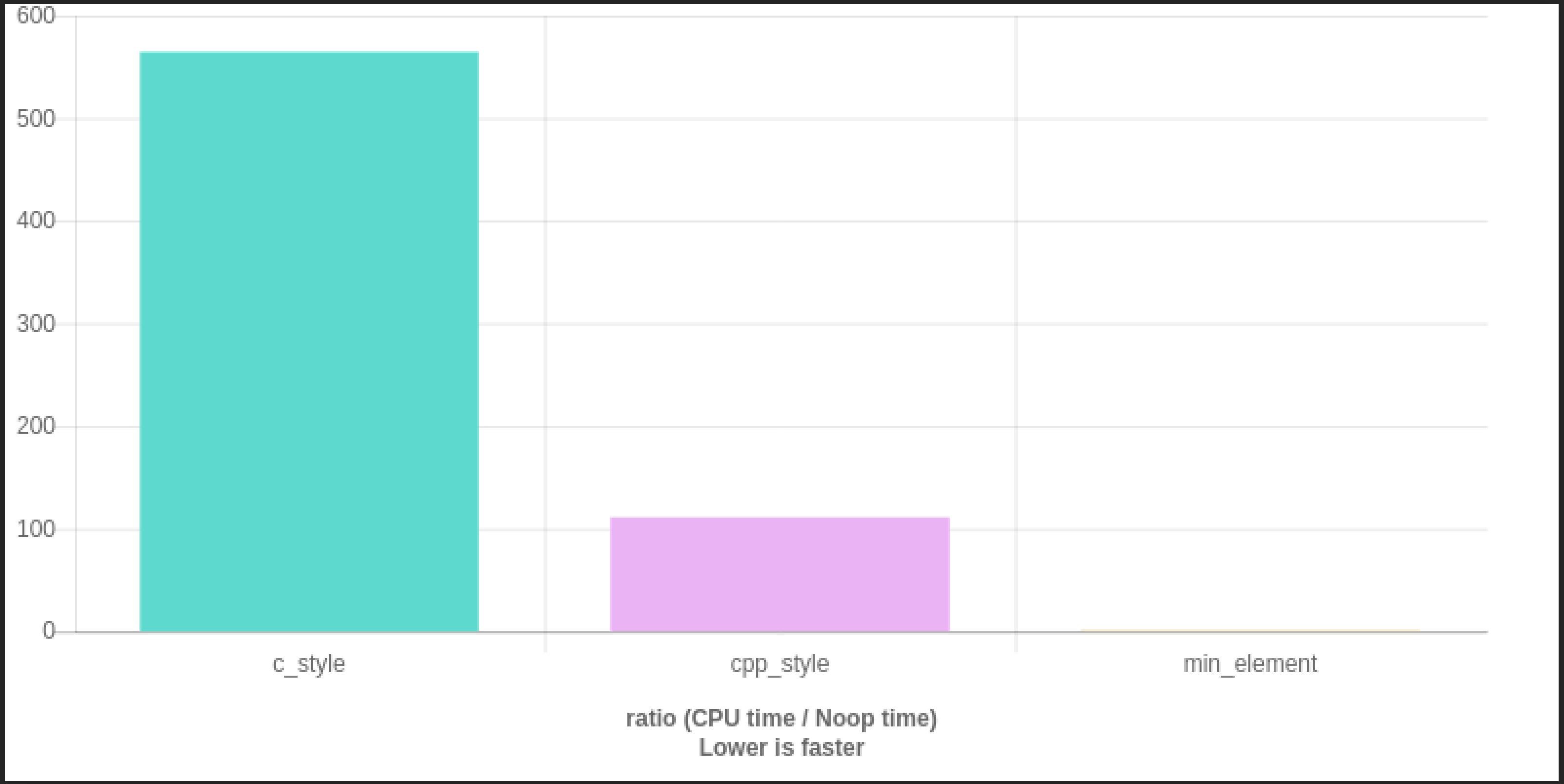
GCC 8.1


400
350
300
250
200
150
100
50
0
c_style
cpp_style
min_element
ratio (CPU time / Noop time)
Lower is faster

clang 6.0 libc++



clang 6.0 libstdc++



C Version

```
#include <algorithm>
int compare_ints(const void *lhs, const void *rhs)
{
  return *(int*)lhs - *(int*)rhs;
}

int sort_c_style()
{
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  qsort(values, 11, sizeof(int), &compare_ints);
  return values[0];
}                                             https://godbolt.org/z/jUAHru
```

- Operation is completely opaque to compiler / runtime
- Prone to size/length mismatches
- Each comparison requires a pointer indirection
- Technically correct call is `qsort(values, sizeof(values)/sizeof(int), sizeof(int), &compare_ints);`

C++ Version

```cpp
#include <algorithm>
int sort_cpp_style()
{
  int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  std::sort(std::begin(values), std::end(values));
  return values[0];
}
```

https://godbolt.org/z/udsEpW

- Data size / data length mismatches impossible
- Compiler has complete visibility into types and data used

`min_element` Version

```cpp
#include <algorithm>
int min_element_cpp_style()
{
  const int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  return *std::min_element(std::begin(values), std::end(values));
}
```
https://godbolt.org/z/gw9_Dp

- Uses part of the C++ standard library's algorithms
- Utilizes `const`
- Effectively free

How free is `min_element`?

```cpp
#include <algorithm>
#include <iterator>

int min_element_cpp_style()
{
  const int values[] = {1341,12341,362,841,79,11,434,29,152,178,1};
  return *std::min_element(std::begin(values), std::end(values));
}

int main()
{
  const int val = min_element_cpp_style();
  return val;
}
```

https://godbolt.org/z/ap8hh3

How free is `min_element`?

- Stronger typing gives the compiler more information
- C++ has a stronger type system than C
- Therefor, strongly typed C++ is more optimizable than C