How to check an array or vector is sorted?

[intro]

is\_sorted

[namespace]

std

[library]

<algorithm>

[syntax]

(1)template <class ForwardIterator>

bool is\_sorted (ForwardIterator first, ForwardIterator last);

(2)template <class ForwardIterator, class Compare>

bool is\_sorted (ForwardIterator first, ForwardIterator last, Compare comp);

[return]

Return true iff the array or vector is sorted.

If the comp is given, then the is\_sorted function will use this function as comparison to check the array or vector is sorted.

Otherwise, it will use default comparison (check the array is sorted by ascending order)

[P.S.]

The second function in syntax section support only g++ version (2) or above.

[equivalent]

For version (2), the is\_sorted is implemented as the following code.

template <class ForwardIterator>

bool is\_sorted (ForwardIterator first, ForwardIterator last)

{

if (first==last)

return true;

ForwardIterator next = first;

while (++next!=last)

{

if (comp(\*next,\*first)) //for version (2)

return false;

++first;

}

return true;

}

[code]

#include <iostream>

#include <algorithm>

#include <array>

using namespace std;

int main ()

{

std::array<int,4> foo {2,4,1,3};

do

{

// try a new permutation:

std::prev\_permutation(foo.begin(),foo.end());

// print range:

std::cout << "foo:";

for (int& x:foo)

{

cout <<" "<< x;

}

cout <<endl;

} while (!std::is\_sorted(foo.begin(),foo.end()));

cout << "the range is sorted!\n";

return 0;

}

[result]

foo: 2 3 4 1

foo: 2 3 1 4

foo: 2 1 4 3

foo: 2 1 3 4

foo: 1 4 3 2

foo: 1 4 2 3

foo: 1 3 4 2

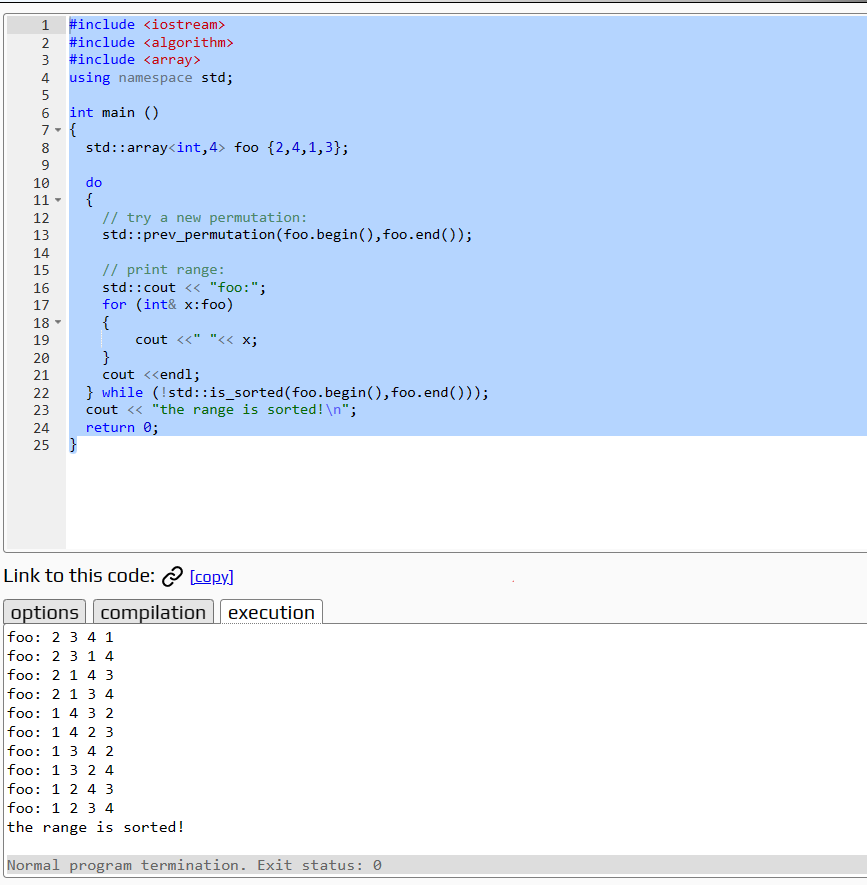
foo: 1 3 2 4

foo: 1 2 4 3

foo: 1 2 3 4

the range is sorted!

[screenshot]



[ref]

<https://cplusplus.com/reference/algorithm/is_sorted/>