LAB 12

STL-Algorithm

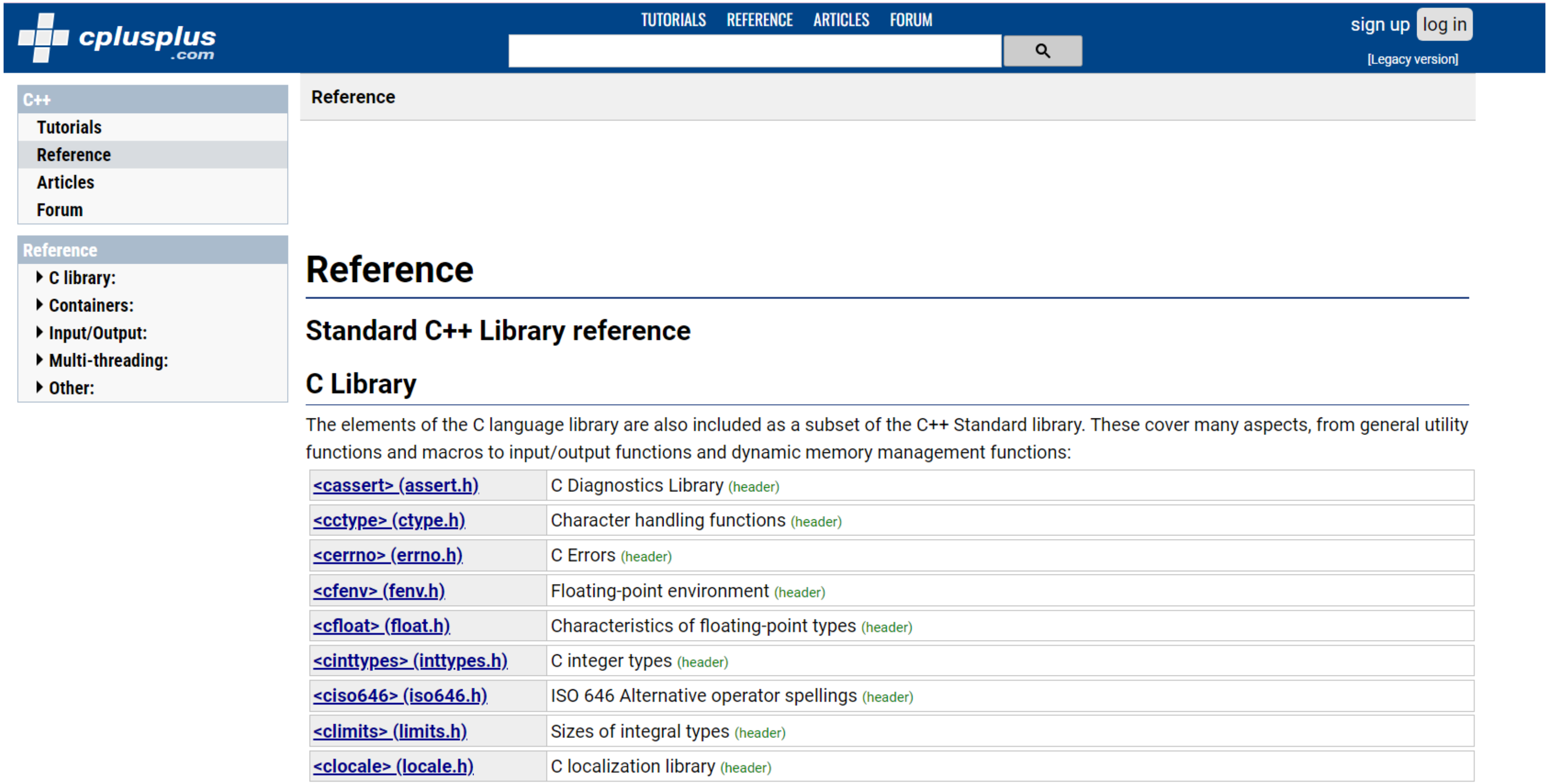
2024/5/27

STL: Standard Template Library

- **Containers**: hold objects, all of a specified type
- **Iterators**: a generic pointer to access objects in containers
- **Algorithms**: act on objects in containers
- Save you a lot of efforts to use those common data structures or algorithms

STL: Standard Template Library

- Reference - C++ Users



Reference

Reference

Standard C++ Library reference

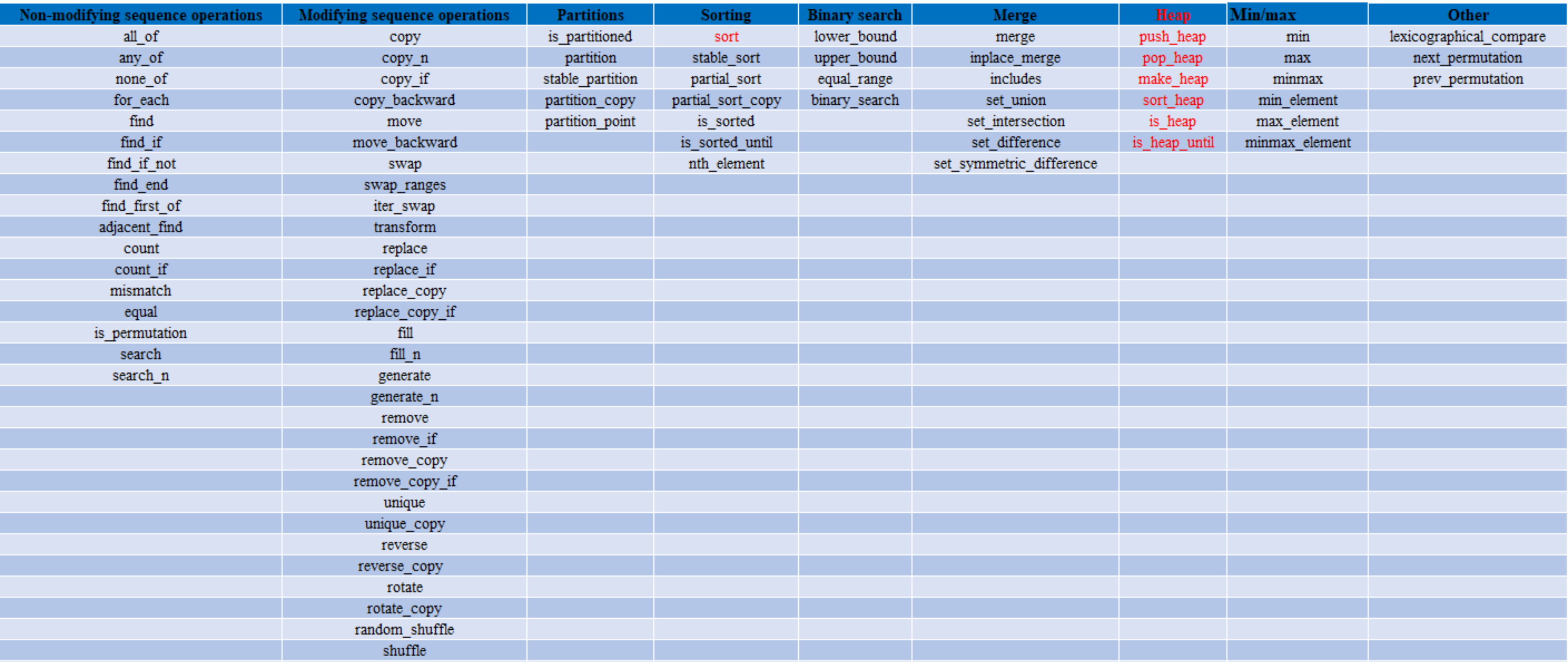
C Library

The elements of the C language library are also included as a subset of the C++ Standard library. These cover many aspects, from general utility functions and macros to input/output functions and dynamic memory management functions:

<cassert> (assert.h)	C Diagnostics Library (header)
<cctype> (ctype.h)	Character handling functions (header)
<cerrno> (errno.h)	C Errors (header)
<cfenv> (fenv.h)	Floating-point environment (header)
<cfloat> (float.h)	Characteristics of floating-point types (header)
<cinttypes> (inttypes.h)	C integer types (header)
<ciso646> (iso646.h)	ISO 646 Alternative operator spellings (header)
<climits> (limits.h)	Sizes of integral types (header)
<clocale> (locale.h)	C localization library (header)

Function in <algorithm>

https://cplusplus.com/reference/algorithm/



Non-modifying sequence operations	Modifying sequence operations	Partitions	Sorting	Binary search	Merge	Heap	Min/max	Other
all_of	copy	is_partitioned	sort	lower_bound	merge	push_heap	min	lexicographical_compare
any_of	copy_n	partition	stable_sort	upper_bound	inplace_merge	pop_heap	max	next_permutation
none_of	copy_if	stable_partition	partial_sort	equal_range	includes	make_heap	minmax	prev_permutation
for_each	copy_backward	partition_copy	partial_sort_copy	binary_search	set_union	sort_heap	min_element	
find	move	partition_point	is_sorted		set_intersection	is_heap	max_element	
find_if	move_backward		is_sorted_until		set_difference	is_heap_until	minmax_element	
find_if_not	swap		nth_element		set_symmetric_difference			
find_end	swap_ranges							
find_first_of	iter_swap							
adjacent_find	transform							
count	replace							
count_if	replace_if							
mismatch	replace_copy							
equal	replace_copy_if							
is_permutation	fill							
search	fill_n							
search_n	generate							
	generate_n							
	remove							
	remove_if							
	remove_copy							
	remove_copy_if							
	unique							
	unique_copy							
	reverse							
	reverse_copy							
	rotate							
	rotate_copy							
	random_shuffle							
	shuffle							

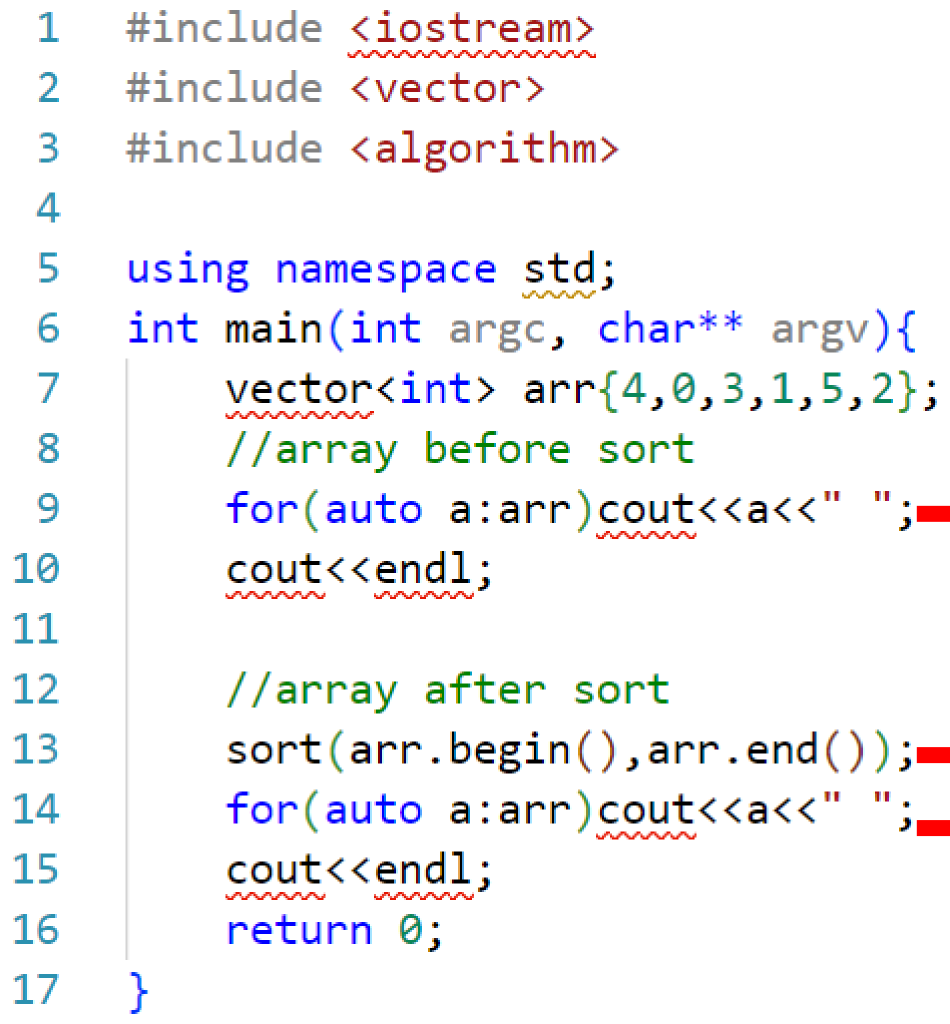
SORT

- #include <algroithm>

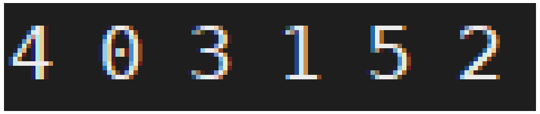
```
sort(begin,end);
sort(begin,end,order_comparison);
```

- begin: Initial position of the sequence to be sorted
- end: final position of the sequence to be sorted
- order_comparison: Binary function that accepts two elements in the range as arguments

SORT: sort (begin, end);

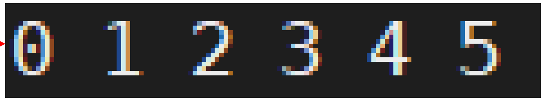


```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;
int main(int argc, char** argv){
    vector<int> arr{4,0,3,1,5,2};
    //array before sort
    for(auto a:arr)cout<<a<<" ";
    cout<<endl;

    //array after sort
    sort(arr.begin(),arr.end());
    for(auto a:arr)cout<<a<<" ";
    cout<<endl;
    return 0;
}
```



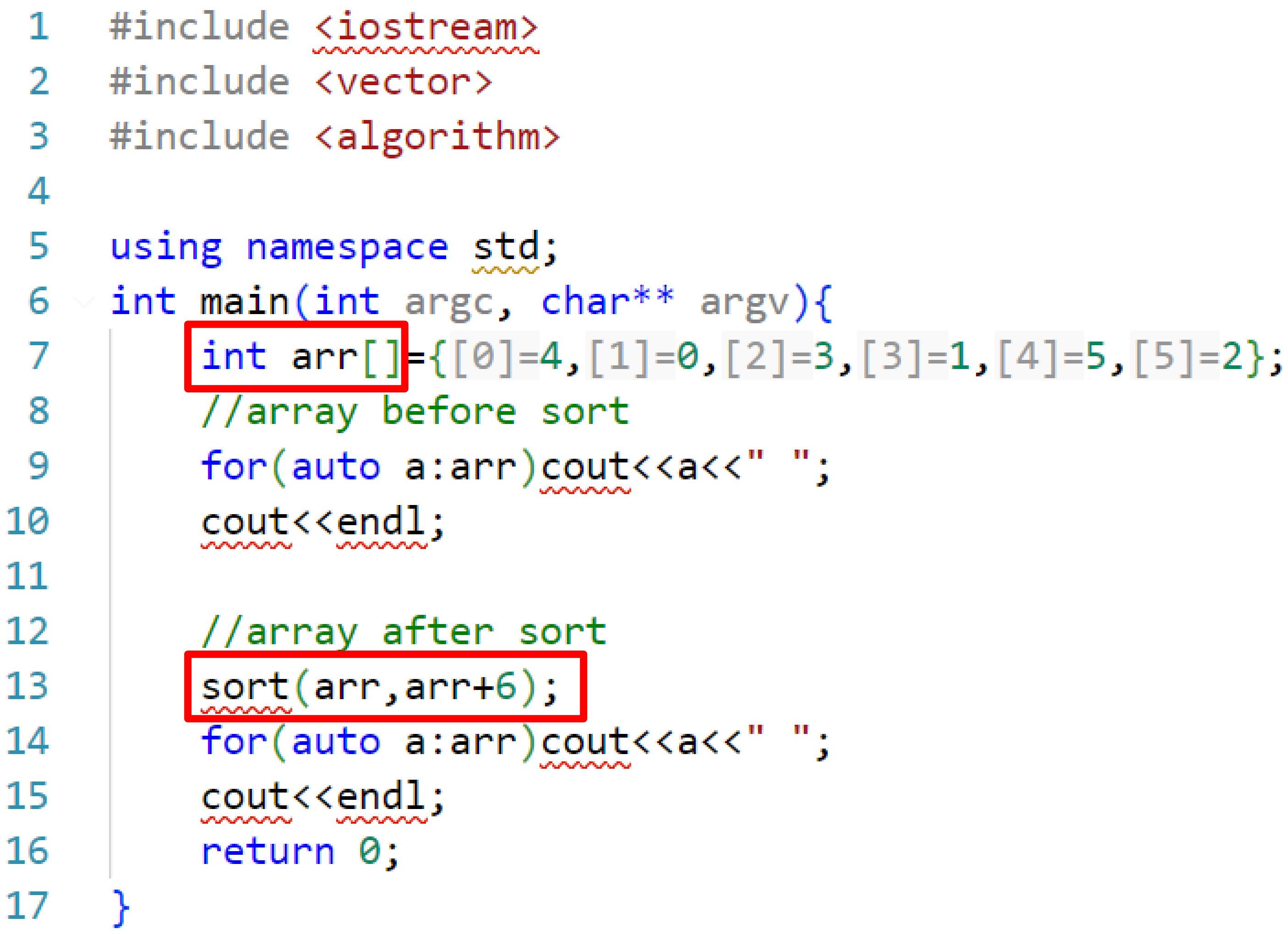
```
4 0 3 1 5 2
```

Sort the elements into ascending order



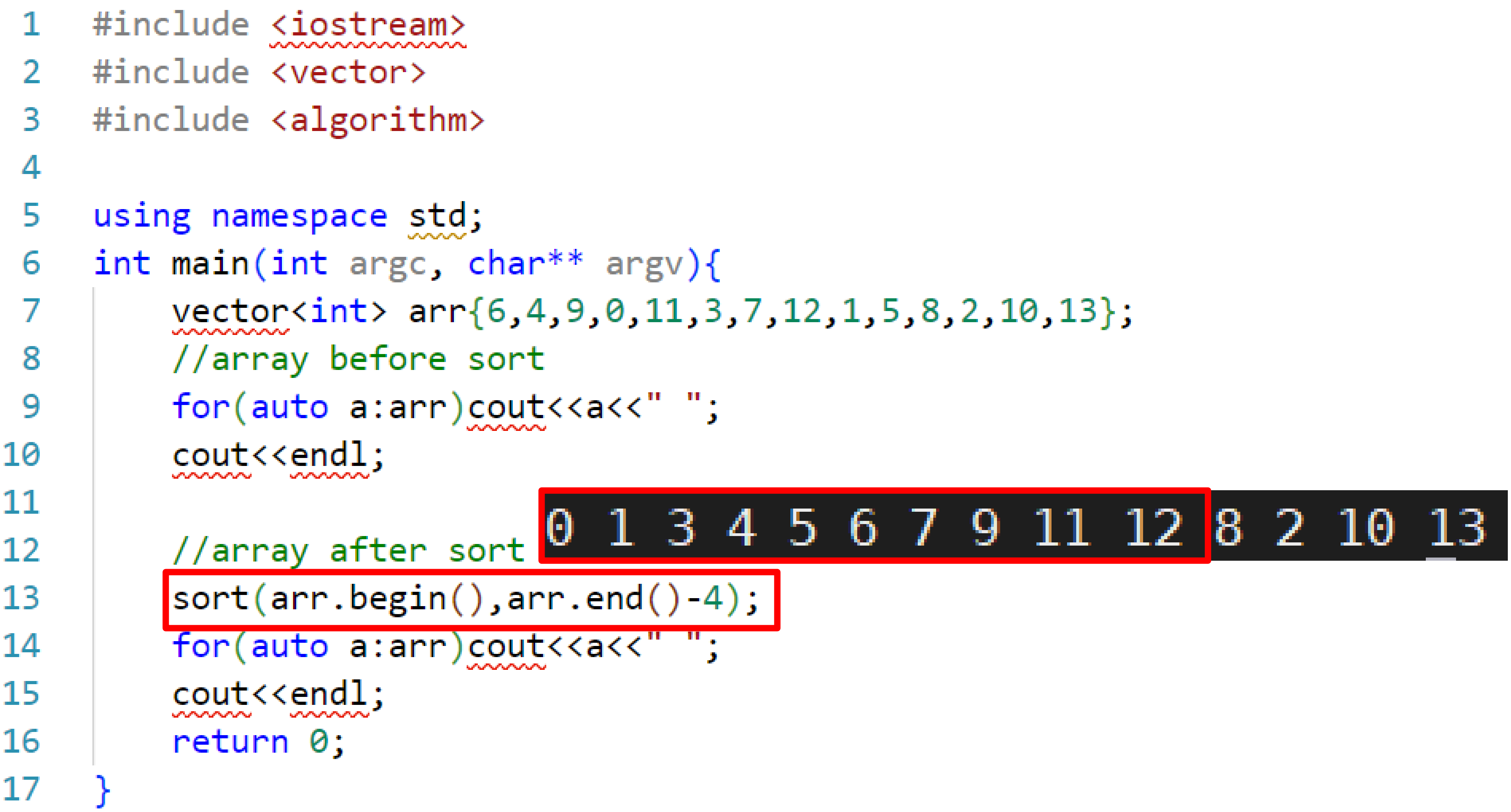
```
0 1 2 3 4 5
```

SORT: sort (begin, end);



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

using namespace std;
int main(int argc, char** argv){
    int arr[]={[0]=4,[1]=0,[2]=3,[3]=1,[4]=5,[5]=2};
    //array before sort
    for(auto a:arr)cout<<a<<" ";
    cout<<endl;

    //array after sort
    sort(arr,arr+6);
    for(auto a:arr)cout<<a<<" ";
    cout<<endl;
    return 0;
}
```

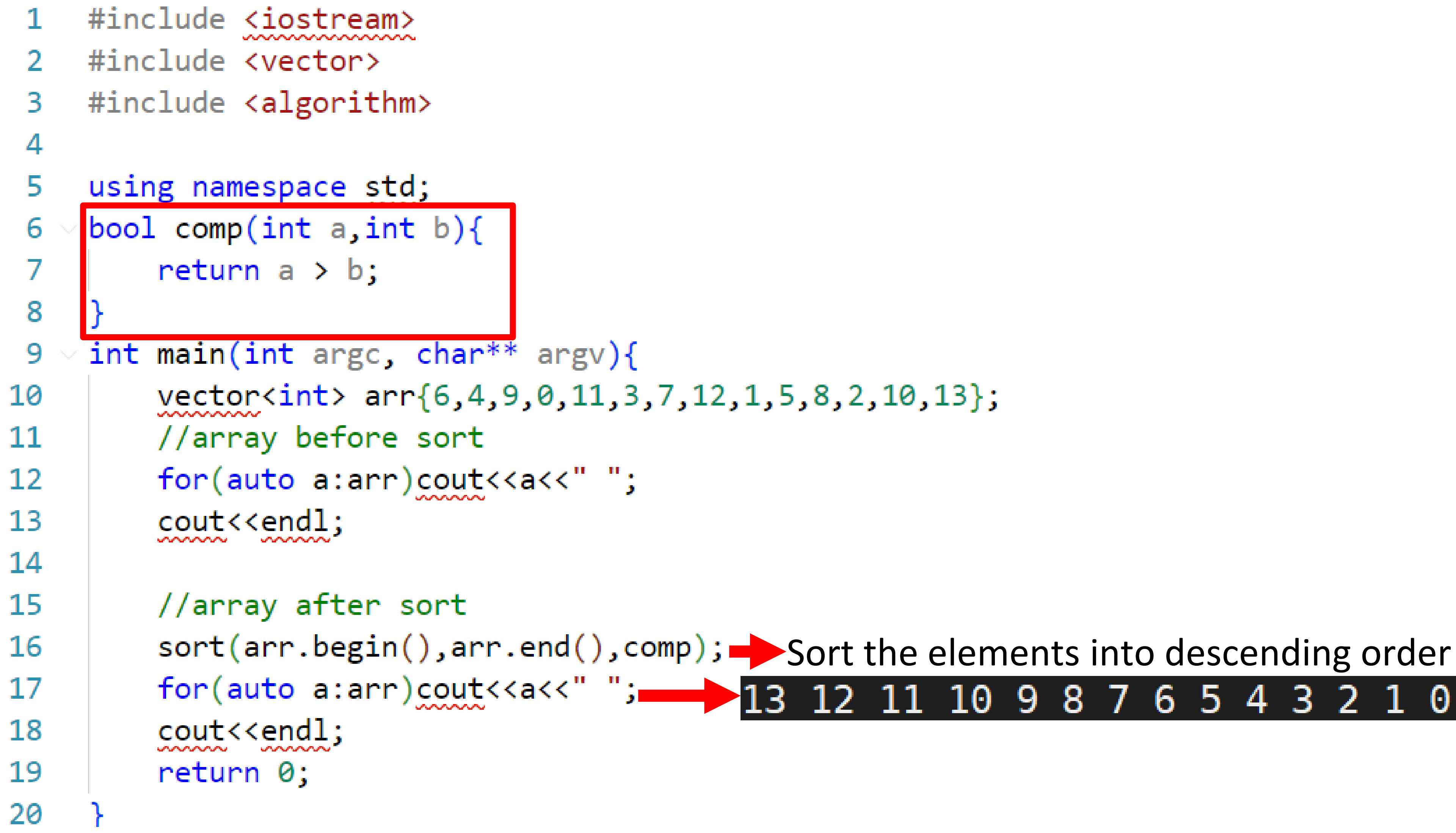
SORT: sort (begin, end);



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

using namespace std;
int main(int argc, char** argv){
    vector<int> arr{6,4,9,0,11,3,7,12,1,5,8,2,10,13};
    //array before sort
    for(auto a:arr)cout<<a<<" ";
    cout<<endl;

    //array after sort
    sort(arr.begin(),arr.end()-4);
    for(auto a:arr)cout<<a<<" ";
    cout<<endl;
    return 0;
}
```

```
0 1 3 4 5 6 7 9 11 12 8 2 10 13
```

SORT:sort (begin, end, order_comparison);

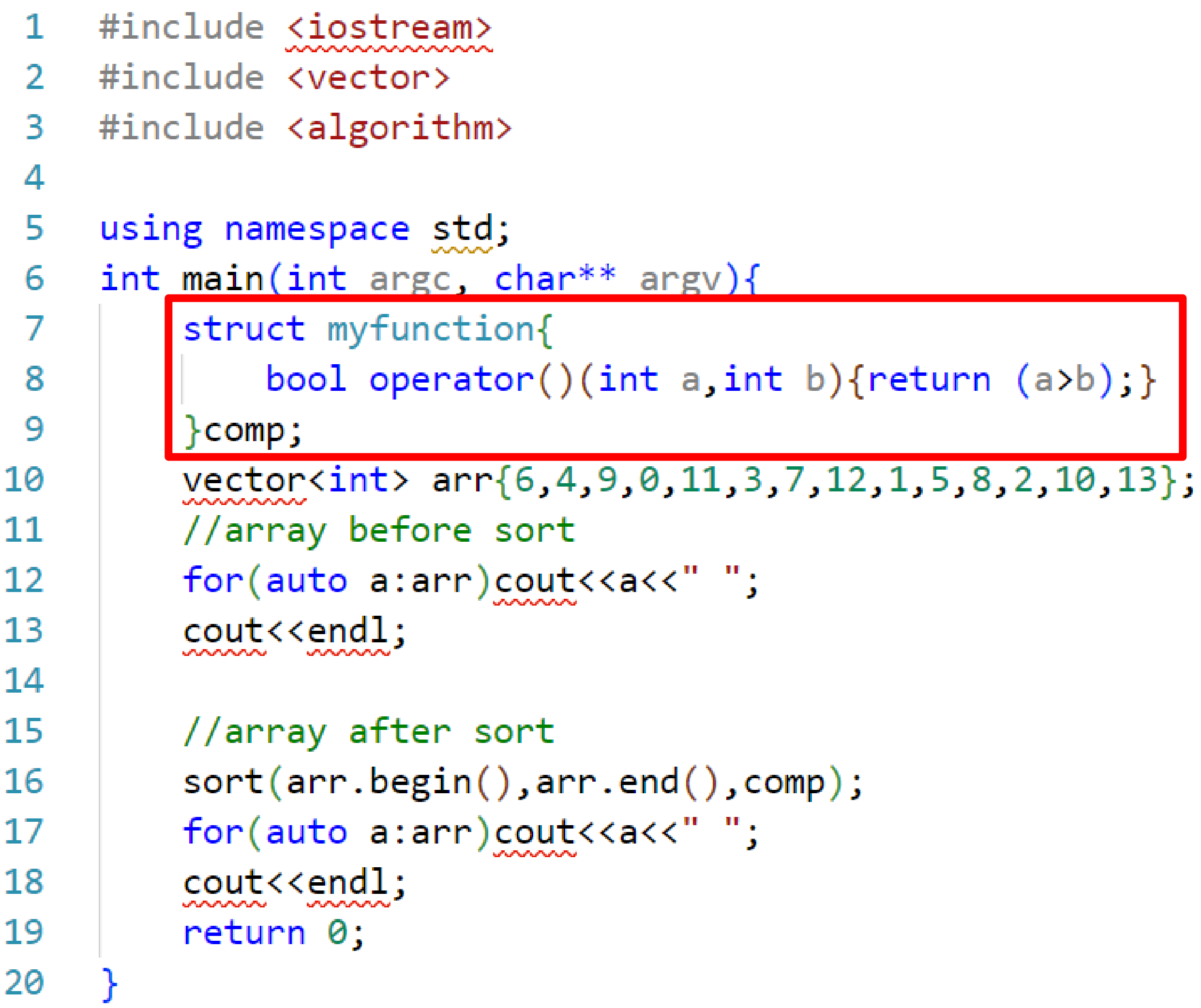


```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;
bool comp(int a,int b){
    return a > b;
}
int main(int argc, char** argv){
    vector<int> arr{6,4,9,0,11,3,7,12,1,5,8,2,10,13};
    //array before sort
    for(auto a:arr)cout<<a<<" ";
    cout<<endl;

    //array after sort
    sort(arr.begin(),arr.end(),comp);
    for(auto a:arr)cout<<a<<" ";
    cout<<endl;
    return 0;
}
```

Sort the elements into descending order

```
13 12 11 10 9 8 7 6 5 4 3 2 1 0
```

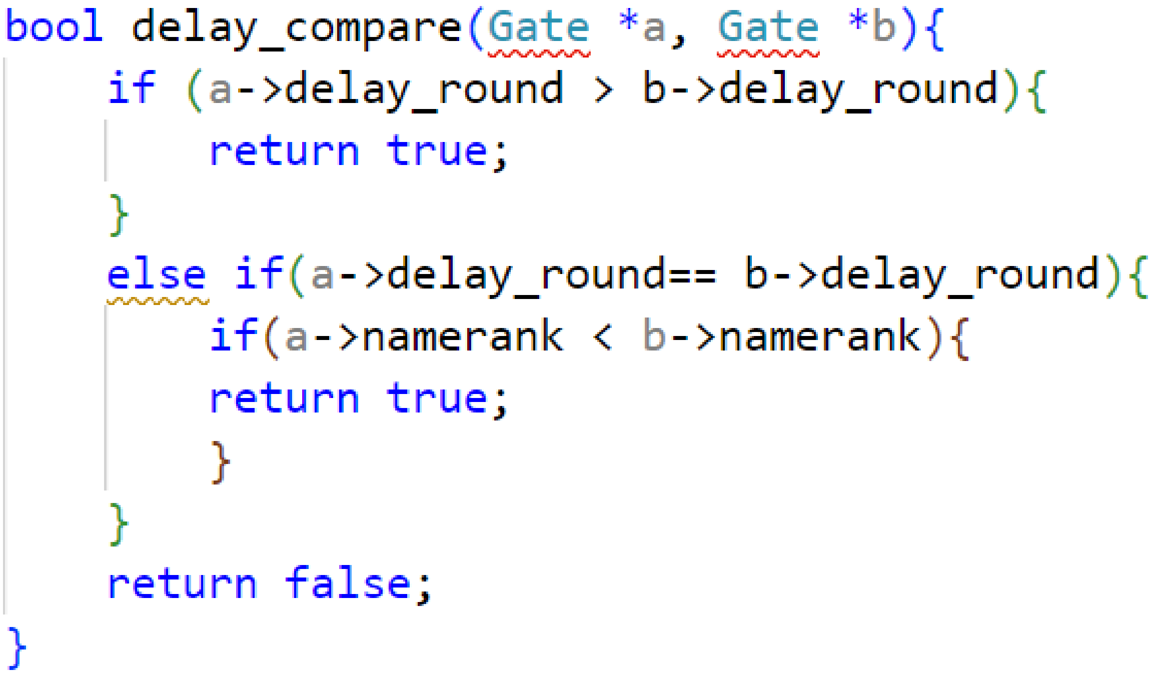
SORT:sort (begin, end, order_comparison);



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

using namespace std;
int main(int argc, char** argv){
    struct myfunction{
        bool operator()(int a,int b){return (a>b);}
    }comp;
    vector<int> arr{6,4,9,0,11,3,7,12,1,5,8,2,10,13};
    //array before sort
    for(auto a:arr)cout<<a<<" ";
    cout<<endl;

    //array after sort
    sort(arr.begin(),arr.end(),comp);
    for(auto a:arr)cout<<a<<" ";
    cout<<endl;
    return 0;
}
```

SORT:sort (begin, end, order_comparison);

- Sort() can also sort class, map……
- You can define your sort function by yourself.



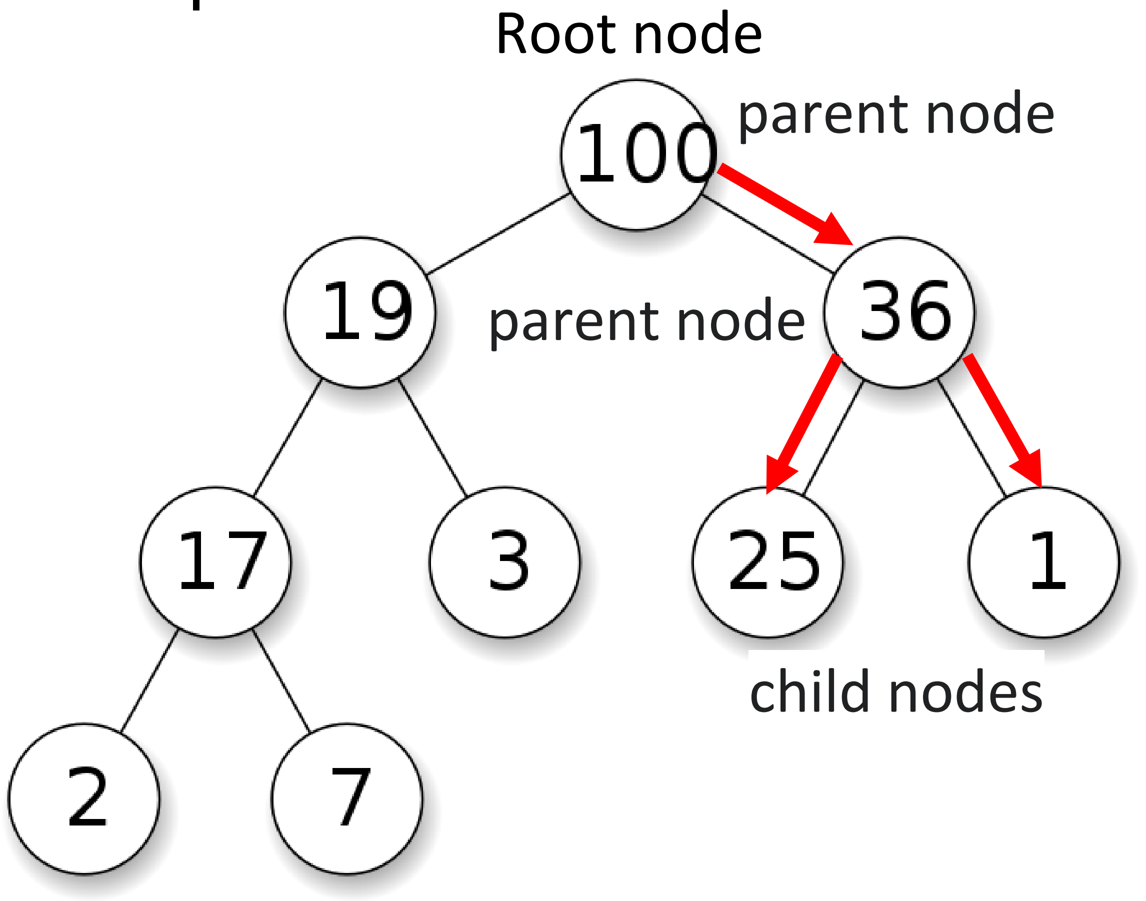
```
bool delay_compare(Gate *a, Gate *b){
    if (a->delay_round > b->delay_round){
        return true;
    }
    else if(a->delay_round== b->delay_round){
        if(a->namerank < b->namerank){
        return true;
        }
    }
    return false;
}
```

Heap

- A tree-based data structure
- Heaps are usually implemented with an array
  - Each element in the array represents a node of the heap
  - The parent / child relationship is defined implicitly by the elements' indices in the array

Heap

Tree representation



Root node

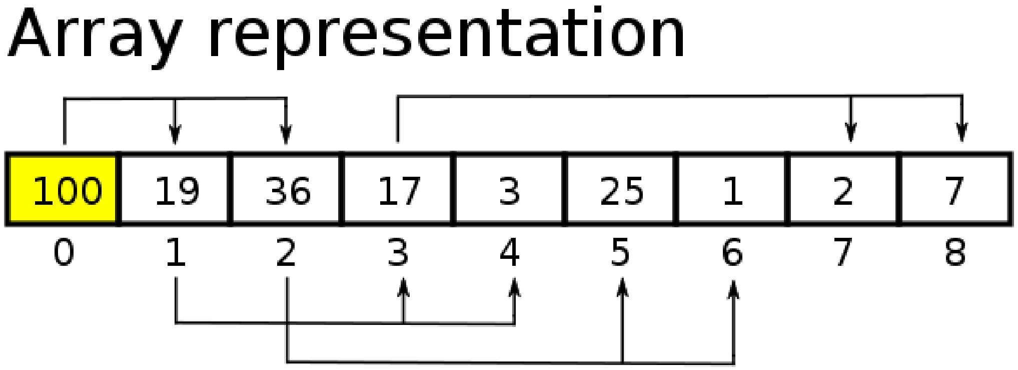
parent node

parent node

child nodes

Heap

- A node at index i
  - Children are at indices $2i+1$ and $2i+2$
  - Parent is at index $[(i-1)/2]$

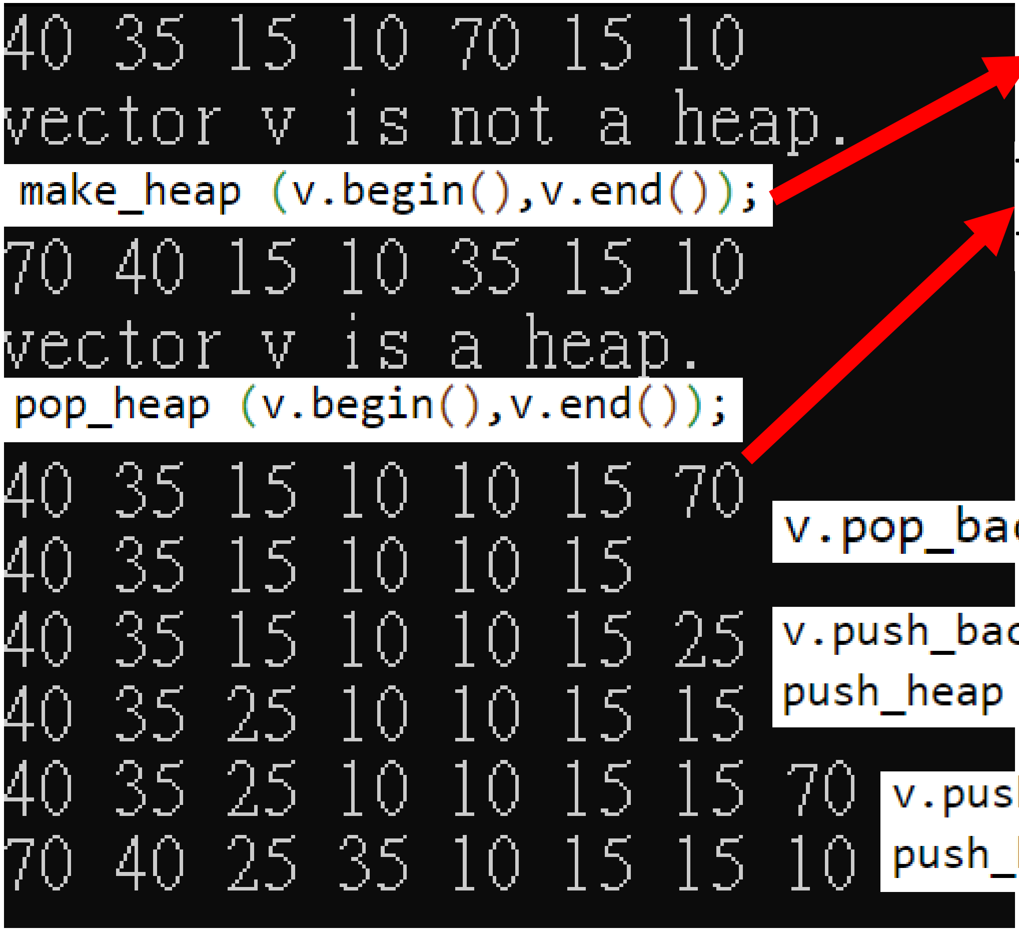


Array representation

100	19	36	17	3	25	1	2	7
0	1	2	3	4	5	6	7	8

Heap

```
int myints[] = {[0]=40,[1]=35,[2]=15,[3]=10,[4]=70,[5]=15,[6]=10};
vector<int> v(myints,myints+7);
```



```
40 35 15 10 70 15 10
vector v is not a heap.
```

```
make_heap (v.begin(),v.end());
```

Rearranges vector v to form a heap

```
70 40 15 10 35 15 10
vector v is a heap.
```

```
pop_heap (v.begin(),v.end());
```

The element with highest value move to (last-1)

```
40 35 15 10 10 15 70
```

```
v.pop_back();
```

Remove last element in vector

```
40 35 15 10 10 15
```

```
v.push_back(25);
push_heap (v.begin(),v.end());
```

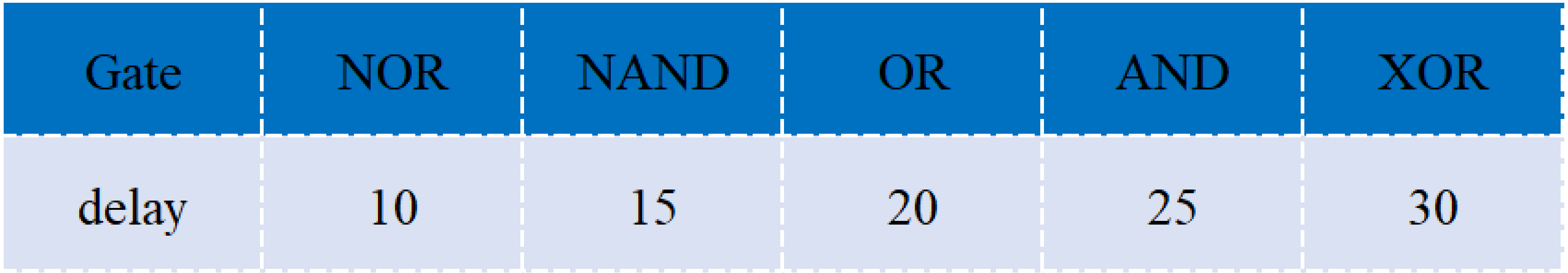
```
40 35 15 10 10 15 25
40 35 25 10 10 15 15
```

```
v.push_back(70);
push_heap (v.begin(),v.end());
```

```
40 35 25 10 10 15 15 70
70 40 25 35 10 15 15 10
```

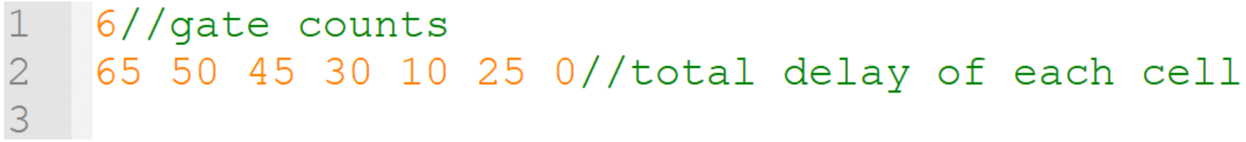
Lab Exercise

- The total delay of each cell is formed into a heap.
- This exercise requires calculating the delay of each cell to determine which type of logic gate the component is, and name all connecting wire.
- Each gate will only have 2input and 1input
- Gate delay=total delay – (delay from child nodes which have larger total delay)

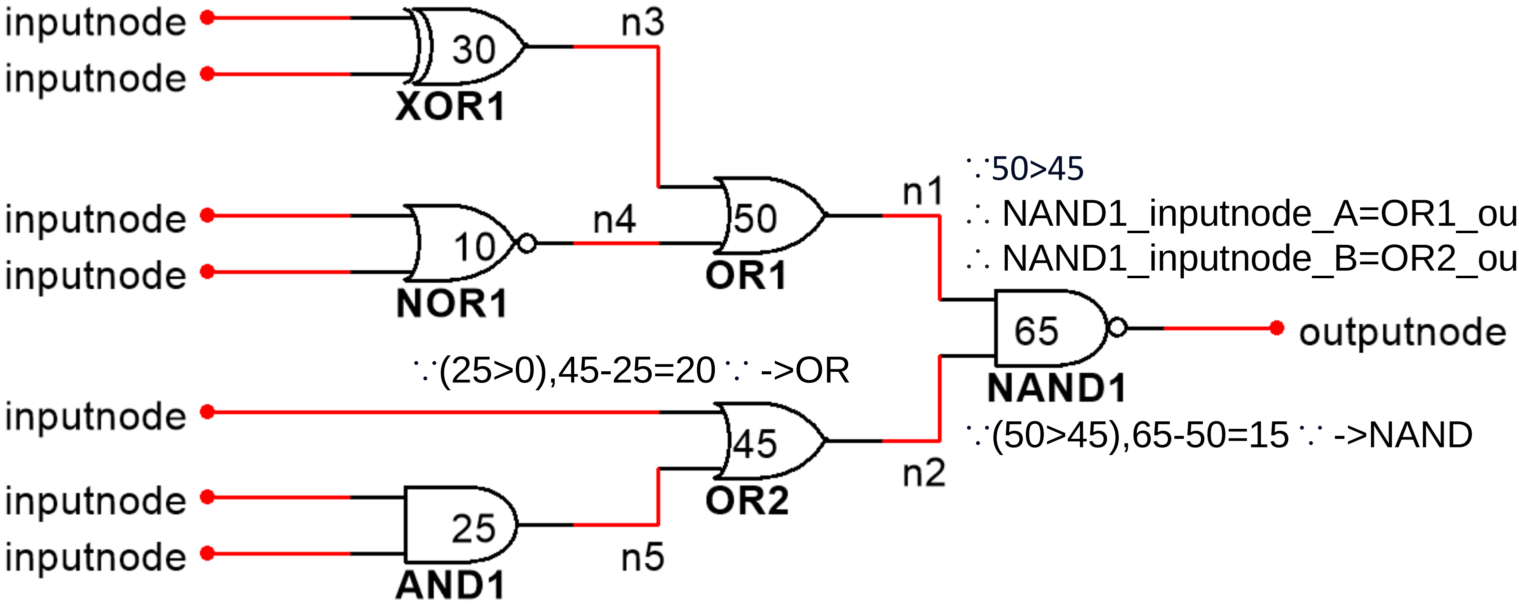


Gate	NOR	NAND	OR	AND	XOR
delay	10	15	20	25	30

Lab Exercise



```
6//gate counts
65 50 45 30 10 25 0//total delay of each cell

```



∵50>45
∴ NAND1_inputnode_A=OR1_output(n1)
∴ NAND1_inputnode_B=OR2_output(n2)

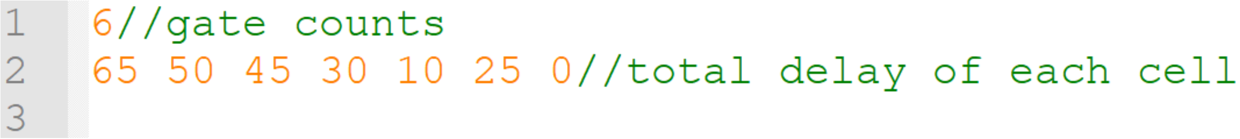
∵(25>0),45-25=20 ∵ ->OR

∵(50>45),65-50=15 ∵ ->NAND

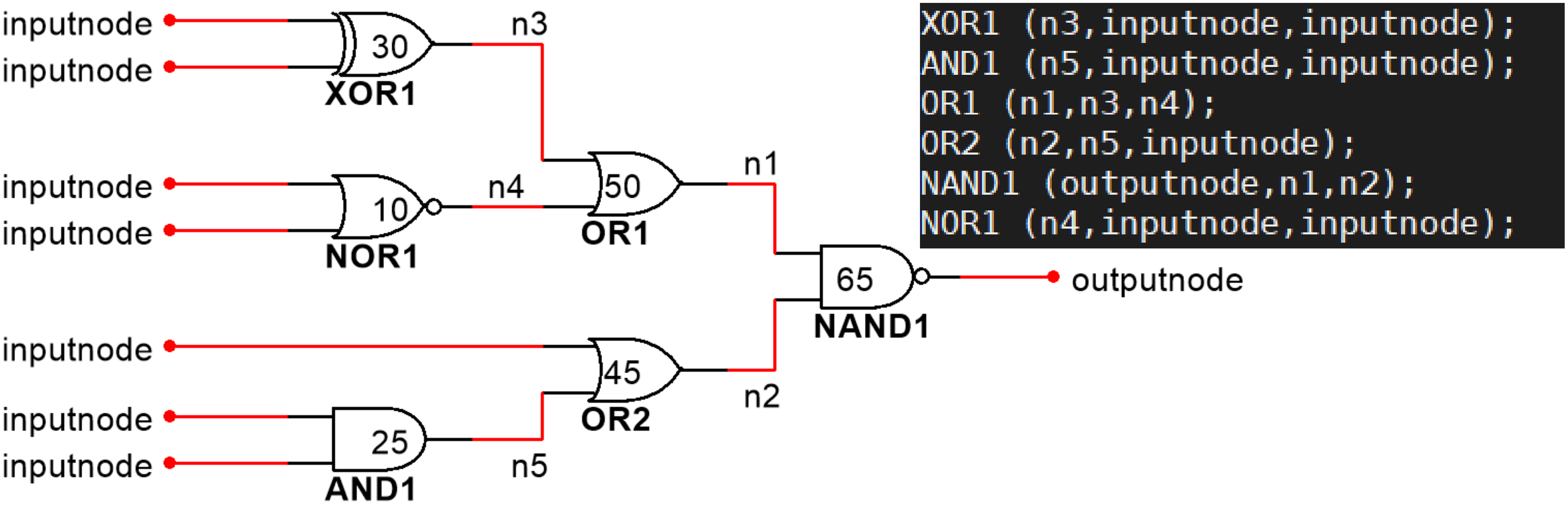
Gate	NOR	NAND	OR	AND	XOR
delay	10	15	20	25	30

Lab Exercise

- Output
  - Sort according to the delay of the logic gate into descending order
  - [gate_type][Gate_ID] ([outputnode],[inputnode_A],[inputnode_B]);



```
6//gate counts
65 50 45 30 10 25 0//total delay of each cell

```



```
XOR1 (n3,inputnode,inputnode);
AND1 (n5,inputnode,inputnode);
OR1 (n1,n3,n4);
OR2 (n2,n5,inputnode);
NAND1 (outputnode,n1,n2);
NOR1 (n4,inputnode,inputnode);
```

Compile & OJ Command

- Compile
  - g++ main.cpp -o Lab12
- OJ
  - /home/share/demo_OOP112_2 Lab 12

Submission

- You should exactly follow the output format
- Upload all your cpp to new E3
- Naming rule : studentID_LAB12.cpp
- Deadline is on E3
- Make sure your code can run on server