# #include <algorithm>

- **std::find**

**InputIterator find (InputIterator first, InputIterator last, const T& val);**

หา val ในช่วงข้อมูล [first,last) ถ้าเจอ ให้รีเทิร์น iterator ชี้ตำแหน่งของ val แต่ถ้าไม่เจอ ให้รีเทิร์น last

```
vector<int> v; // สมมุติว่าใส่ของเข้าไปก็แล้วกันนะ
vector<int>::iterator it;
    it = find (v.begin(), v.end(), 30);
    if (it == v.end()) cout << "30 not found " << endl;
    else cout << "30 is in v " << endl;
```

# แบบฝึกหัด

เขียนโปรแกรมอ่าน input (เอาเป็น int เท่านั้น) จากคีย์บอร์ด

size 1 2 3 .etc มาสร้างเวคเตอร์ หลังจากนั้นให้ user พิมพ์หาค่าที่ต้องการว่ามีหรือไม่ (พิมพ์ว่ามีหรือไม่มีเป็นการตอบออกมา)

เราใช้ std::find ในการหา

# map::find กับ std::find

- ถ้าใช้ std มันจะเป็นการพยายามหา pair ทั้ง pair
- แต่ถ้าใช้ของ map เราให้ key แล้วมันจะหา value

- ดังนั้นใช้งานคนละแบบ

- แต่จริงๆ ถ้าใช้งานกับ map ตัวเมธอดของ map เอง จะเร็วกว่า เพราะ search ใน structure ของ map ที่เก็บดาต้า (ซึ่งจริงๆ เก็บเป็นต้นไม้ ดังนั้นจะหาดาต้าได้เร็วกว่า std::find ซึ่งต้องไล่หาดาต้าแต่ละตัวด้วย iterator เท่านั้น

- **std::find_if**

InputIterator find_if (InputIterator first, InputIterator last, UnaryPredicate pred)

หา ในช่วงข้อมูล [first,last) ว่า pred เป็น true หรือเปล่า ถ้าเจอ ให้รีเทิร์น iterator ชี้ตำแหน่งของ val แต่ถ้าไม่เจอ ให้รีเทิร์น last

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
bool IsOdd (int i) { return ((i%2)==1); }

int main () {
    std::vector<int> myvector;
    myvector.push_back(10); myvector.push_back(25);
    myvector.push_back(40); myvector.push_back(55);

     std::vector<int>::iterator it = std::find_if (myvector.begin(),
    myvector.end(), IsOdd);

    std::cout << "The first odd value is " << *it << '\n'; return 0;
    }
```

- **std::sort**

**void sort (RandomAccessIterator first, RandomAccessIterator last);**

เรียงดาต้า จากช่วง [first,last) ให้เรียงจากน้อยไปมาก (การเรียงเทียบด้วยเครื่องหมาย < )

**void sort (RandomAccessIterator first, RandomAccessIterator last, Compare comp);**

เรียงดาต้า จากช่วง [first,last) ให้เรียงจากน้อยไปมาก (การเรียงเทียบด้วย comp object )

# Default sort

```
bool myfunction (int i,int j) { return (i<j); }

int main()
{
    std::array<int, 10> s = {5, 7, 4, 2, 8, 6, 1, 9, 0, 3};

    // sort using the default operator<
    std::sort(s.begin(), s.end());
    for (auto a : s) {
        std::cout << a << " ";
    }
    std::cout << '\n';
```

ostream std::cout

# Sort โดยใช้ ฟังก์ชัน greater ที่มีอยู่แล้วใน std

```
std::array<int, 10> s2 = {5, 7, 4, 2, 8, 6, 1, 9, 0, 3};
// sort using a standard library compare function object
std::sort(s2.begin(), s2.end(), std::greater<int>());
for (auto a : s2) {
    std::cout << a << " ";
}
std::cout << '\n';
```

# Sort โดยเขียน compare object เอง

```
 std::array<int, 10> s3 = {9, 6, 2, 1, 4, 5, 3};
// sort using a custom function object
struct {
    bool operator()(int a, int b)
    {
        return a < b;
    }
} customLess;

std::sort(s3.begin(), s3.end(), customLess);
for (auto a : s3) {
    std::cout << a << " ";
}
std::cout << '\n';
```

# Sort โดยใส่ lambda expression

```
std::array<int, 10> s4 = {9, 8, 2, 1, 7, 5, 3};
// sort using a lambda expression
std::sort(s4.begin(), s4.end(), [](int a, int b) {
    return b < a;
});
for (auto a : s4) {
    std::cout << a << " ";
}
std::cout << '\n';
```

# Sort ด้วยฟังก์ชั่นที่นิยามมาก่อนเอง

```
std::array<int, 10> s5 = {9, 8, 2, 1, 7, 5, 3};
//sort using function


std::sort (s5.begin(), s5.end(), myfunction);
for (auto a : s5) {
    std::cout << a << " ";
}
std::cout << '\n';
```

algorithm ยังมีฟังก์ชั่นอื่นๆอีกมาก

- Write a program that creates a string then puts the characters in alphabetical order (there's a sort algorithm).

- Write a program that creates a string then shuffles the characters (there's a random_shuffle algorithm)

- Write a program that creates a vector of strings and then puts the strings in alphabetical order

- Write a program that creates 2 strings then prints out the letters they have in common.

- Write a program that creates a string then removes the spaces in the string (use remove and erase).

- Write a program that creates a string then counts the number of times that a particular letter ('t' for example) appears in the string. (there's a count algorithm).

- Write a function that given 2 strings tells you if one is a reverse of the other (use reverse on one string and see if it matches the other)

- Write a program that creates a vector of floats and reverses them.