**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«Национальный исследовательский университет ИТМО»**

Факультет информационных технологий и программирования

Лабораторная работа №6

Вариант 21.

*Обобщенные алгоритмы.*

**Выполнил студент группы № M3111**
Соловьев Михаил Александрович.

Санкт-Петербург
2023

В данном задании необходимо реализовать следующие обобщенные алгоритмы:
1. none_of - возвращает true, если все элементы диапазона не удовлетворяют некоторому предикату. Иначе false
2. is_sorted - возвращает true, если все элементы диапазона находятся в отсортированном порядке относительно некоторого критерия
3. find_not - находит первый элемент, не равный заданному
Каждый алгоритм должен быть выполнен в виде шаблонной функции, позволяющей взаимодействовать со стандартными контейнерами STL с помощью итераторов. Предикаты, условия, операторы сравнения должны быть параметризованы.

Код:

functions.h

```
#ifndef COURSE_C___FUNCTIONS_H
#define COURSE_C___FUNCTIONS_H

#include <iostream>
#include <algorithm>

struct CPoint {
    int x;
    int y;
};

bool operator==(const CPoint& left, const CPoint& right) {
    return left.x == right.x && left.y == right.y;
}

template <typename InputIterator, typename Predicate>
bool none_of_custom(InputIterator first, InputIterator last, Predicate predicate) {
    while (first != last) {
        if (predicate(*first)) {
            return false;
        }
        ++first;
    }
    return true;
}

template <typename ForwardIterator, typename Compare>
```

```
bool is_sorted_custom(ForwardIterator first, ForwardIterator last, Compare comp)
{
    if (first == last)
        return true;

    ForwardIterator next = first;
    while (++next != last) {
        if (comp(*next, *first)) {
            return false;
        }
        ++first;
    }

    return true;
}

template <typename InputIterator, typename T>
InputIterator find_not_custom(InputIterator first, InputIterator last, const T&
value) {
    while (first != last) {
        if (*first != value) {
            return first;
        }
        ++first;
    }
    return last;
}

bool is_negative(int num) {
    return num < 0;
}

bool is_origin(const CPoint& point) {
    return point.x == 0 && point.y == 0;
}

bool lessThanByX(const CPoint& left, const CPoint &right) {
    return left.x < right.x;
}

#endif //COURSE_C___FUNCTIONS_H
```

main.cpp

```
#include <iostream>
#include <vector>
#include "functions.h"

int main() {
    std::vector <int> numbers1 = {1, 2, 3, 4, 5};
    std::vector <int> numbers2 = {-1, 2, 3, 4, 5};
    std::vector <int> numbers3 = {6, 2, 3, 4, 5};

    CPoint targetPoint = {1, 0};

    std::vector <CPoint> custom_vector1 = {{1, 0}, {2, 0}, {3, 0}};
    std::vector <CPoint> custom_vector2 = {{0, 0}, {2, 0}, {3, 0}};
    std::vector <CPoint> custom_vector3 = {{6, 0}, {2, 0}, {3, 0}};

    auto it1 = find_not_custom(numbers1.begin(), numbers1.end(), 1);
    auto it2 = find_not_custom(custom_vector1.begin(), custom_vector1.end(),
targetPoint);

    std::cout << std::boolalpha;
    std::cout << none_of_custom(numbers1.begin(), numbers1.end(), is_negative)
<< "\n";
    std::cout << none_of_custom(numbers2.begin(), numbers1.end(), is_negative)
<< "\n";

    std::cout << is_sorted_custom(numbers1.begin(), numbers1.end(),
std::less<int>()) << "\n";
    std::cout << is_sorted_custom(numbers3.begin(), numbers1.end(),
std::less<int>()) << "\n";

    std::cout << *it1 << '\n';

    std::cout << "-------------" << std::endl;
    std::cout << "CUSTOM VALUE:\n";
    std::cout << "-------------" << std::endl;

    std::cout << none_of_custom(custom_vector1.begin(), custom_vector1.end(),
is_origin) << "\n";
    std::cout << none_of_custom(custom_vector2.begin(), custom_vector2.end(),
is_origin) << "\n";
```

```
    std::cout << is_sorted_custom(custom_vector1.begin(), custom_vector1.end(),
lessThanByX) << "\n";
    std::cout << is_sorted_custom(custom_vector3.begin(), custom_vector3.end(),
lessThanByX) << std::endl;
    std::cout << "{" << it2->x << ", " << it2->y << "}\n";

    return 0;
}
```

Вывод:

Я научился работать с шаблонными функциями и обобщенными алгоритмами.