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

Шаблонные функции

Выполнил студент группы № M3111
Гаврилов Алексей Евгеньевич
Подпись:

Проверил:
Повышев Владислав Вячеславович

Санкт-Петербург
2022

Текст задания

Требуется реализовать следующие обобщенные алгоритмы.

1. **all_of** - возвращает true, если все элементы диапазона удовлетворяют некоторому предикату. Иначе false
2. **any_of** - возвращает true, если хотя бы один из элементов диапазона удовлетворяет некоторому предикату. Иначе false
3. **none_of** - возвращает true, если все элементы диапазона не удовлетворяют некоторому предикату. Иначе false
4. **one_of** - возвращает true, если ровно один элемент диапазона удовлетворяет
некоторому предикату. Иначе false
5. **is_sorted** - возвращает true, если все элементы диапазона находятся в отсортированном порядке относительно некоторого критерия
6. **is_partitioned** - возвращает true, если в диапазоне есть элемент, делящий все элементы на удовлетворяющие и не удовлетворяющие некоторому предикату. Иначе false.
7. **find_not** - находит первый элемент, не равный заданному
8. **find_backward** - находит первый элемент, равный заданному, с конца
9. **is_palindrome** - возвращает true, если заданная последовательность является палиндромом относительно некоторого условия. Иначе false.

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.

При сдаче работы требуется продемонстрировать работу алгоритмов как на стандартных, так и на пользовательских типах данных, например CPoint, CRational, далее работает ваша индивидуальная (не “коллективная”) фантазия.

Решение с комментариями

```cpp
//fun.cpp
// Created by Volirvag on 23.04.2022.
//
#include "fun.h"
#include <iostream>
void Print(bool temp){
    if (temp)
        std::cout << "True" << std::endl;
    else
        std::cout << "False" << std::endl;
}
```

```cpp
//fun.h
// Created by Volirvag on 23.04.2022.
//

#ifndef LAB_6_1_FUN_H
#define LAB_6_1_FUN_H
using namespace std;
template<class T>
struct CPoint{
public:
    T x = 0;
    T y = 0;
    CPoint(){
        x = 0;
        y = 0;
    };

    CPoint(T &a, T &b){
        x = a;
        y = b;
    }
    ~CPoint(){
        //delete x;
        //delete y;
    };
    bool operator !=(CPoint<T> temp)
    {
        return ((x != temp.x) || (y != temp.y));
    }
    bool operator ==(CPoint<T> temp)
    {
        return ((x == temp.x) && (y == temp.y));
    }
    CPoint<T> &operator = (CPoint<T> temp){
        x = temp.x;
        y = temp.y;
        return *this;
    }

};

template<class T, class Predicate>
bool all_off(T &obj, Predicate &predicate){
    typename T::const_iterator it = obj.begin();

    while (it != obj.end()){
        if (predicate == *it)
            return false;
        it++;
    }
```

```
    return true;
}

template<class Iter, class Predicate>
bool is_partitioned(Iter begin, Iter end, Predicate predicate){
    auto it = begin;
    Predicate temp;
    int count = 0;
    while (it != end){
        if (predicate == *it){
            temp = *it;
            count++;
        }
        it++;
    }
    if (count != 1)
        return false;

    while (it != end){
        if (temp == *it)
            count--;
        if ((temp == *it) && (count < 1)){
            return false;
        }
        it++;
    }
    return true;
}

template<class T, class Predicate>
bool is_partitioned(T &obj, Predicate &predicate){
    typename T::const_iterator it = obj.begin();
    Predicate temp;
    int count = 0;
    while (it != obj.end()){
        if (predicate == *it){
            temp = *it;
            count++;
        }
        it++;
    }
    if (count != 1)
        return false;

    while (it != obj.end()){
        if (temp == *it)
            count--;
        if ((temp == *it) && (count < 1)){
            return false;
        }
        it++;
    }
    return true;
}

template<class T, class Predicate>
int find_backward(T &obj, Predicate &predicate){
    int count = 0;
    for (typename T::reverse_iterator it = obj.rbegin(); it !=obj.rend();
it++){
        if (predicate == *it){
            return count;
        }
        else
```

```
            count++;
    }
    return -1;
}

void Print(bool temp);
#endif //LAB_6_1_FUN_H
```

//main

```
#include <iostream>
#include <iterator>
#include <vector>
#include "fun.h"
using namespace std;


int main() {
    cout << "Hello, World!" << endl;
    cout << endl;
    cout << "------Process of working for all_off with type_int: -----" <<
endl;
    vector<int> vector_int_1(5,1);
    int pred_int = 1;
    Print(all_off<vector<int>, int>(vector_int_1, pred_int));

    cout << "Input predicate for int: " << endl;
    cin >> pred_int;
    Print(all_off<vector<int>, int>(vector_int_1, pred_int));

    cout << endl;
    cout << "------Process of working for all_off with type_char: -----" <<
endl;
    vector<char> vector_char_a(5, 'a');
    char pred_char = 'a';
    Print(all_off<vector<char>, char>(vector_char_a, pred_char));
    cout << "Input predicate for char: " << endl;
    cin >> pred_char;
    Print(all_off<vector<char>, char>(vector_char_a, pred_char));

    cout << endl;
    cout << "------Process of working for all_off with
type_vector<CPoint<int>>: -----" << endl;
    int x = 1;
    int y = 1;
    CPoint<int> base_point_int_1(x,y);
    vector<CPoint<int>> array_point_base;
    for (int i = 0; i < 5; i++)
        array_point_base.push_back(base_point_int_1);
    Print(all_off<vector<CPoint<int>>, CPoint<int>>(array_point_base,
base_point_int_1));
    cout << "Input int x and int y for temp CPoint: " << endl;
    cin >> x >> y;
    CPoint<int> base_point_int_rand(x, y);
    Print(all_off<vector<CPoint<int>>, CPoint<int>>(array_point_base,
base_point_int_rand));

    cout << endl;
    cout << "------Process of working for all_off with
type_vector<CPoint<char>>: -----" << endl;
    char x_char = 'a';
    char y_char = 'b';
    CPoint<char> base_point_char_a(x_char, y_char);
    vector<CPoint<char>> array_point_base_char;
```

```
    for (int i = 0; i < 5; i++)
        array_point_base_char.push_back(base_point_char_a);
    Print(all_off<vector<CPoint<char>>, CPoint<char>>(array_point_base_char,
base_point_char_a));
    cout << "Input char x and char y for temp CPoint: " << endl;
    cin >> x_char >> y_char;
    CPoint<char> base_point_char_rand(x_char, y_char);
    Print(all_off<vector<CPoint<char>>, CPoint<char>>(array_point_base_char,
base_point_char_rand));

    cout << endl;
    cout << "------Process of working for is_partitioned with type_int: -----
" << endl;
    int test_value = 2;
    vector<int> test_int_for_true;
    test_int_for_true.push_back(2);
    for (int i = 1; i < 5; i++)
        test_int_for_true.push_back(1);
    Print(is_partitioned<vector<int>, int>(test_int_for_true, test_value));
    test_int_for_true.clear();
    for (int i = 0; i < 5; i++)
        test_int_for_true.push_back(3);
    Print(is_partitioned<vector<int>, int>(test_int_for_true, test_value));

    cout << endl;
    cout << "------Process of working for is_partitioned with
type_vector<CPoint<int>>: -----" << endl;
    Print(is_partitioned<vector<CPoint<int>>, CPoint<int>>(array_point_base,
base_point_int_1));
    array_point_base.clear();
    int temp_1 = 1;
    array_point_base.push_back(CPoint<int>(temp_1,temp_1));
    temp_1 = 2;
    for (int i = 1; i < 5; i++)
        array_point_base.push_back(CPoint<int>(temp_1, temp_1));
    Print(is_partitioned<vector<CPoint<int>>, CPoint<int>>(array_point_base,
base_point_int_1));
    cout << endl;
    cout << "------Process of working for find_backward with
type_vector<int>: -----" << endl;
    vector<int> test1_int = {1, 2, 3, 4, 5};
    test_value = 3;
    cout << "Posses in set from the end: " <<find_backward<vector<int>,
int>(test1_int, test_value) << endl;
    cout << "Input int test_value for vector<int> test1_int = {1, 2, 3, 4,
5}: " << endl;
    cin >> test_value;
    cout << "Posses in set from the end: " << find_backward<vector<int>,
int>(test1_int, test_value) << endl;
    cout << endl;
    cout << "------Process of working for find_backward with
type_vector<char>: -----" << endl;
    vector<char> test1_char = {'a', 'b', 'c', 'd', 'e'};
    char test_value_char = 'b';
    cout << "Posses in set from the end: " <<find_backward<vector<char>,
char>(test1_char, test_value_char) << endl;
    cout << "Input char test_value_char for vector<char> test1_char = {'a',
'b', 'c', 'd', 'e'}: " << endl;
    cin >> test_value_char;
    cout << "Posses in set from the end: " << find_backward<vector<char>,
char>(test1_char, test_value_char) << endl;

    cout << endl;
    cout << "------Process of working for find_backward with
```

```
type_vector<CPoint<int>>: -----" << endl;
    vector<CPoint<int>> last_test;
    for (int i = 0; i < 5; i++)
        last_test.push_back(CPoint<int>(i,i));
    int key_x;
    int key_y;
    key_x = key_y = 2;
    CPoint<int> test_point_true(key_x, key_y);
    key_x = 1;
    key_y = 2;
    CPoint<int> test_point_false(key_x, key_y);
    cout << "Posses in set from the end: "
<<find_backward<vector<CPoint<int>>, CPoint<int>>(last_test, test_point_true)
<< endl;
    cout << "Posses in set from the end: "
<<find_backward<vector<CPoint<int>>, CPoint<int>>(last_test,
test_point_false) << endl;

    return 0;
}
```