Лабараторная работа 6

Клишевич Вадим, M3105

main.cpp

#include <bits/stdc++.h>  
#include "functions.h"  
  
using namespace std;  
  
bool foo(int x) {  
 return x > 0;  
}  
  
bool bar(double x) {  
 return x < 0;  
}  
  
bool lessInt(int x, int y) {  
 return x < y;  
}  
  
bool lessDouble(double x, double y) {  
 return x > y;  
}  
  
int main() {  
 vector<int> A = {-2, 1, -3};  
 set<double> B = {-2.5, -2.3, 3};  
  
 cout << "allOf\t\t\t" << allOf(A, foo) << " : " << allOf(B, bar) << "\n";  
  
 cout << "anyOf\t\t\t" << anyOf(A, foo) << " : " << anyOf(B, bar) << "\n";  
  
 cout << "noneOf\t\t\t" << noneOf(A, foo) << " : " << noneOf(B, bar) << "\n";  
  
 cout << "oneOf\t\t\t" << oneOf(A, foo) << " : " << oneOf(B, bar) << "\n";  
  
 cout << "isSorted\t\t" << isSorted(A.begin(), A.end(), lessInt) << " : "  
 << isSorted(B.rbegin(), B.rend(), lessDouble) << "\n";  
  
 cout << "isPartitioned\t" << isPartitioned(A, foo) << " : " << isPartitioned(B, bar) << "\n";  
  
 cout << "isPalindrome\t" << isPalindrome(A, foo) << " : " << isPalindrome(B, bar) << "\n";  
  
 cout << "findNot\t\t\t" << findNot(A.begin(), A.end(), 1) << " : "  
 << findNot(B.begin(), B.end(), 1.5) << "\n";  
  
 cout << "findBackward\t" << findBackward(A.begin(), A.end(), 2) << " : "  
 << findBackward(B.begin(), B.end(), -2.5) << "\n";  
  
 cout << "findBackward\t" << findBackward(A.begin(), A.end(), 5) << " : "  
 << findBackward(B.begin(), B.end(), 5.0) << "\n";  
}

functions.h

//  
// Created by vadim on 7.02.20.  
//  
#include <bits/stdc++.h>  
  
using namespace std;  
  
#pragma once  
  
template <typename container, class object>  
bool allOf(const container& c, bool (&f)(object));  
  
template <typename container, class object>  
bool anyOf(const container& c, bool (&f)(object));  
  
template <typename container, class object>  
bool noneOf(const container& c, bool (&f)(object));  
  
template <typename container, class object>  
bool oneOf(const container& c, bool (&f)(object));  
  
template <class InputIterator, class object>  
bool isSorted(const InputIterator& begin, const InputIterator& end, bool (&f)(object, object));  
  
template <typename container, class object>  
bool isPartitioned(const container& c, bool (&f)(object));  
  
template <class InputIterator, class object>  
object findNot(const InputIterator& begin, const InputIterator& end, object value);  
  
template <class InputIterator, class object>  
object findBackward(const InputIterator& begin, const InputIterator& end, object value);  
  
template <typename container, class object>  
bool isPalindrome(const container& c, bool (&f)(object));  
  
#include "functions.tpp"

functions.tpp

//  
// Created by vadim on 7.02.20.  
//  
template <typename container, class object>  
bool allOf(const container& c, bool (&f)(object)) {  
 for (auto& x : c) {  
 if (!f(x)) {  
 return false;  
 }  
 }  
 return true;  
}  
  
template <typename container, class object>  
bool anyOf(const container& c, bool (&f)(object)) {  
 for (auto& x : c) {  
 if (f(x)) {  
 return true;  
 }  
 }  
 return false;  
}  
  
template <typename container, class object>  
bool noneOf(const container& c, bool (&f)(object)) {  
 for (auto& x : c) {  
 if (f(x)) {  
 return false;  
 }  
 }  
 return true;  
}  
  
template <typename container, class object>  
bool oneOf(const container& c, bool (&f)(object)) {  
 bool was = false;  
 for (auto& x : c) {  
 if (f(x) && !was) {  
 was = true;  
 } else if (f(x)) {  
 return false;  
 }  
 }  
 return was;  
}  
  
template <class InputIterator, class object>  
bool isSorted(const InputIterator& begin, const InputIterator& end, bool (&f)(object, object)) {  
 if (begin == end) {  
 return true;  
 }  
  
 for (auto it1 = begin, it2 = next(begin); it2 != end; it1 = next(it1), it2 = next(it2)) {  
 if (!f(\*it1, \*it2)) {  
 return false;  
 }  
 }  
 return true;  
}  
  
template <typename container, class object>  
bool isPartitioned(const container& c, bool (&f)(object)) {  
 int was = -1;  
 for (auto& x : c) {  
 if (f(x) && was == -1) {  
 was = 1;  
 } else if (!f(x) && was == -1) {  
 was = 0;  
 } else if (f(x) && was == 0) {  
 return false;  
 } else if (!f(x) && was == 1) {  
 return false;  
 }  
 }  
 return true;  
}  
  
template <class InputIterator, class object>  
object findNot(const InputIterator& begin, const InputIterator& end, object value) {  
 for (auto it = begin; it != end; it = next(it)) {  
 if (\*it != value) {  
 return \*it;  
 }  
 }  
 return object();  
}  
  
template <class InputIterator, class object>  
object findBackward(const InputIterator& begin, const InputIterator& end, object value) {  
 for (auto it = prev(end); it != end; it = prev(it)) {  
 if (\*it == value) {  
 return \*it;  
 }  
 if (it == begin) {  
 break;  
 }  
 }  
 return object();  
}  
  
template <typename container, class object>  
bool isPalindrome(const container& c, bool (&f)(object)) {  
 for (auto it1 = c.begin(), it2 = prev(c.end()); it1 != c.end(); it1 = next(it1), it2 = prev(it2)) {  
 if (f(\*it1) ^ f(\*it2)) {  
 return false;  
 }  
 }  
 return true;  
}