Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Информационных технологий и программирования

# Программирование на C++

Работа: Лабораторная работа №6

**Вариант 5**

Выполнил: Гаджиев Саид M3115

Санкт-Петербург

2023

***Лабораторная работа №6. Вариант 5: алгоритмы №2, №5, №9.***

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 параметризованными.

***Решение:***

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

namespace myFuncs {
    template<typename T>
    bool greater_than_3(T &value) {
        return value > 3;
    }

    template<typename T>
    bool less_than_0(T &value) {
        return value < 0;
    }

    template<typename T>
    bool less(T &a, T &b) {
        return a < b;
    }

    template<typename T>
    bool greater(T &a, T &b) {
        return a > b;
    }


    template<typename T>
    bool equal(T &a, T &b) {
        return a == b;
    }
```

```cpp
    template<typename T, typename P>
    bool any_of(const T &begin, const T &end, const P &p) {
        for (T i = begin; i != end; i++) {
            if (p(*i)) {
                return true;
            }
        }
        return false;
    }

    template<typename T, typename P>
    bool is_sorted(const T &begin, const T &end, const P &p) {
        if (begin == end) {
            return true;
        }
        for (T i = begin; i != end - 1; i++) {
            if (p(*(i + 1), *i)) {
                return false;
            }
        }
        return true;
    }

    template<typename T, typename P>
    bool is_palindrome(const T &begin, const T &end, const P &p) {
        T P_Begin = begin;
        T P_End = end - 1;
        while (P_Begin < P_End) {
            if (!p(*P_Begin++, *P_End--)) {
                return false;
            }
        }
        return true;
    }
}

int main() {
    std::cout << "Examples \"any_of\": " << std::endl;
    std::vector<int> vec = {1, 2, 3, 4, 5};
    bool any_gt_3 = myFuncs::any_of(vec.begin(), vec.end(),
myFuncs::greater_than_3<int>);
    bool any_lt_0 = myFuncs::any_of(vec.begin(), vec.end(),
myFuncs::less_than_0<int>);
    std::cout << std::boolalpha << any_gt_3 << " " << any_lt_0 << std::endl;

    std::cout << "Examples \"is_sorted\": " << std::endl;
    std::vector<int> sorted_vec = {1, 2, 3, 4, 5};
    bool is_sorted_ascending = myFuncs::is_sorted(sorted_vec.begin(),
sorted_vec.end(), myFuncs::less<int>);
    bool is_sorted_descending = myFuncs::is_sorted(sorted_vec.begin(),
sorted_vec.end(), myFuncs::greater<int>);
    std::cout << std::boolalpha << is_sorted_ascending << " " <<
is_sorted_descending << std::endl;

    std::cout << "Examples \"is_palindrome\": " << std::endl;
    std::vector<char> palindrome = {'a', 'b', 'c', 'd', 'c', 'b', 'a'};
    bool is_palindrome_str = myFuncs::is_palindrome(palindrome.begin(),
palindrome.end(), myFuncs::equal<char>);
    std::cout << std::boolalpha << is_palindrome_str << std::endl;
    return 0;
}
```

Код содержит реализацию некоторых функций в пространстве имён myFuncs, а также примеры их использования в функции main().

В пространстве имён myFuncs определены следующие шаблонные функции:

1. *greater_than_3*(T &value): принимает значение типа T по ссылке и возвращает true, если оно больше 3, иначе false;

2. less_than_0(T &value): принимает значение типа T по ссылке и возвращает true, если оно меньше 0, иначе false;

3. *less*(T &a, T &b): принимает два значения типа T по ссылке и возвращает true, если первое значение меньше второго, иначе false;

4. *greater*(T &a, T &b): принимает два значения типа T по ссылке и возвращает true, если первое значение больше второго, иначе false;

5. *equal*(T &a, T &b): принимает два значения типа T по ссылке и возвращает true, если они равны, иначе false;

6. **any_of**(const T &begin, const T &end, const P &p): принимает итераторы начала и конца диапазона, определяемого типом T, а также функцию p, которая принимает значение типа T и возвращает true или false. Возвращает true, если хотя бы один элемент диапазона удовлетворяет предикату p, иначе false;

7. **is_sorted**(const T &begin, const T &end, const P &p): принимает итераторы начала и конца диапазона, определяемого типом T, а также функцию p, которая принимает два значения типа T и возвращает true или false. Возвращает true, если элементы диапазона упорядочены в соответствии с предикатом p, иначе false;

8. **is_palindrome**(const T &begin, const T &end, const P &p): принимает итераторы начала и конца диапазона, определяемого типом T, а также функцию p, которая принимает два значения типа T и возвращает true или false. Возвращает true, если диапазон является палиндромом в соответствии с предикатом p, иначе false.