МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

Лабораторная работа №6

По дисциплине «Программирование»

Выполнил студент группы №M3106

Казаков Никита Андреевич

Проверил

Повышев Владислав Вячеславович

Лабораторная работа №5. *Шаблоны, исключения.*
Написать шаблонную функцию или класс согласно варианту. Описать класс-исключение или иерархию классов-исключений. Генерировать исключения в соответствующих исключительных
ситуациях. Если у вас есть другие предложения по исключительным ситуациям – используйте их.

main.cpp

```
#include <iostream>
#include "console_handler.h"
int main() {
	bool p_end = true;
	while (p_end) {
		p_end = commands();
	}
	std::cout << "==========Shutting down==========\n";
	return 0;
}
```

custom_class.cpp

```
#include "custom_class.hpp"
Point::Point(int x, int y) : x(x), y(y) {};
bool operator> (Point left, int right) {
  return left.x > right;
}
bool operator< (Point left, int right) {
	return left.x < right;
}
bool operator== (Point left, int right) {
	return left.x == right;
}
bool operator> (Point left, Point right) {
	return left.x > right.x;
}
bool operator< (Point left, Point right) {
	return left.x < right.x;
}
bool operator== (Point left, Point right) {
	return left.x == right.x;
}
bool operator% (Point left, int right) {
	return left.x % right;
}
```

custom_class.hpp

```
#pragma once
class Point {
public:
	Point(int, int);
	friend bool operator> (Point, int);
	friend bool operator< (Point, int);
	friend bool operator== (Point, int);
	friend bool operator> (Point, Point);
	friend bool operator< (Point, Point);
	friend bool operator== (Point, Point);
```

```
	friend bool operator% (Point, int);
private:
	int x, y;
};
```

console_handler.cpp

```
#include <iostream>
#include <vector>
#include "console_handler.h"
#include "algorithms.hpp"
#include "custom_class.hpp"
void predicate_chooser () {
	std::cout << "==========CHOOSE PREDICATE==========" << '\n';
	std::cout << "Better for all_of:" << '\n';
	std::cout << "All == 3 - 1" << '\n';
	std::cout << "Each one > 10 - 2" << '\n';
	std::cout << "====================================" << '\n';
	std::cout << "Better for is_sorted:" << '\n';
	std::cout << "Ascending - 3" << '\n';
	std::cout << "Descending - 4" << '\n';
	std::cout << "====================================" << '\n';
	std::cout << "Better for is_predicate:" << '\n';
	std::cout << "Value == 1 - 5" << '\n';
	std::cout << "Value % 2 == 0 - 6" << '\n';
	std::cout << "====================================" << '\n';
	std::cout << "Cancel - 0" << '\n';
	std::cout << "~> ";
}
void all_of_item () {
	std::cout << "GETTING STARTED WITH THE all_of..." << '\n';
	predicate_chooser();
	int command;
	std::cin >> command;
	if (command == cancel_f) return;
	long long arr_size;
	std::cout << "====Choose custom // vector=====" << '\n';
	std::cout << "Custom class (Array of points) - 1" << '\n';
	std::cout << "Simple vector - 2" << '\n';
	std::cout << "~> ";
	short int type_chooser;
	std::cin >> type_chooser;
	if (type_chooser == 1) {
		std::cout << "Choose array size"<< '\n';
		std::cout << "~> ";
		std::cin >> arr_size;
		std::cout << "Fill vector of points (capacity = " << arr_size << ')' << '\n';
		std::cout << "~> ";
		std::vector<Point> temp_vector;
		for (int i = 0; i < arr_size; i++) {
			int x, y;
			std::cin >> x >> y;
			Point temp(x, y);
			temp_vector.push_back(temp);
		}
		switch (command) {
			case all_3_f: {
				std::cout << all_of(temp_vector.begin(), temp_vector.end(), [](Point i){
return i == 3; }) << '\n' << '\n';
```

```
                    return;
                } case all_10_f: {
                    std::cout << all_of(temp_vector.begin(), temp_vector.end(), [](Point i){
return i > 10; }) << '\n' << '\n';
                } default: {
                    std::cout << "Not implemented yet..." << '\n';
                    return;
                }
            }
        } else if (type_chooser == 2) {
            std::cout << "Choose vector size"<< '\n';
            std::cout << "~> ";
            std::cin >> arr_size;
            std::cout << "Fill vector (capacity = " << arr_size << ')' << '\n';
            std::cout << "~> ";
            std::vector<int> temp_vector;
            for (int i = 0; i < arr_size; i++) {
                int temp_value;
                std::cin >> temp_value;
                temp_vector.push_back(temp_value);
            }
            switch (command) {
                case all_3_f: {
                    std::cout << all_of(temp_vector.begin(), temp_vector.end(), [](int i){ return i
== 3; }) << '\n' << '\n';
                    return;
                } case all_10_f: {
                    std::cout << all_of(temp_vector.begin(), temp_vector.end(), [](int i){ return i
> 10; }) << '\n' << '\n';
                    return;
                } default: {
                    std::cout << "Not implemented yet..." << '\n';
                    return;
                }
            }
        } else {
            std::cout << "========UNKNOWN COMMAND========" << '\n';
        }
}
void is_sorted_item () {
        std::cout << "GETTING STARTED WITH THE is_sorted..." << '\n';
        predicate_chooser();
        int command;
        std::cin >> command;
        if (command == cancel_f) return;
        long long arr_size;
        std::cout << "====Choose custom // vector=====" << '\n';
        std::cout << "Custom class (Array of points) - 1" << '\n';
        std::cout << "Simple vector - 2" << '\n';
        std::cout << "~> ";
        short int type_chooser;
        std::cin >> type_chooser;
        if (type_chooser == 1) {
            std::cout << "Choose array size"<< '\n';
            std::cout << "~> ";
            std::cin >> arr_size;
            std::cout << "Fill vector of points (capacity = " << arr_size << ')' << '\n';
            std::cout << "~> ";
            std::vector<Point> temp_vector;
```

```
            for (int i = 0; i < arr_size; i++) {
                int x, y;
                std::cin >> x >> y;
                Point temp(x, y);
                temp_vector.push_back(temp);
            }
            switch (command) {
                case is_1_f: {
                    std::cout << is_sorted(temp_vector.begin(), temp_vector.end(), [](Point a,
Point b){ return a < b; }) << '\n' << '\n';
                    return;
                } case is_2_f: {
                    std::cout << is_sorted(temp_vector.begin(), temp_vector.end(), [](Point a,
Point b){ return a > b; }) << '\n' << '\n';
                    return;
                } default: {
                    std::cout << "Not implemented yet..." << '\n';
                    return;
                }
            }
        } else if (type_chooser == 2) {
            std::cout << "Choose vector size"<< '\n';
            std::cout << "~> ";
            std::cin >> arr_size;
            std::cout << "Fill vector (capacity = " << arr_size << ')' << '\n';
            std::cout << "~> ";
            std::vector<int> temp_vector;
            for (int i = 0; i < arr_size; i++) {
                int temp_value;
                std::cin >> temp_value;
                temp_vector.push_back(temp_value);
            }
            switch (command) {
                case is_1_f: {
                    std::cout << is_sorted(temp_vector.begin(), temp_vector.end(), [](int a, int
b){ return a < b; }) << '\n' << '\n';
                    return;
                } case is_2_f: {
                    std::cout << is_sorted(temp_vector.begin(), temp_vector.end(), [](int a, int
b){ return a > b; }) << '\n' << '\n';
                    return;
                } default: {
                    std::cout << "Not implemented yet..." << '\n';
                    return;
                }
            }
        } else {
            std::cout << "========UNKNOWN COMMAND========" << '\n';
        }
}
void is_palindrome_item() {
        std::cout << "GETTING STARTED WITH THE is_palindrome..." << '\n';
        predicate_chooser();
        int command;
        std::cin >> command;
        if (command == cancel_f) return;
        long long arr_size;
        std::cout << "====Choose Custom class (Point) // Integer=====" << '\n';
        std::cout << "Custom class (Point) - 1" << '\n';
```

```
	std::cout << "Simple vector - 2" << '\n';
	std::cout << "~> ";
	short int type_chooser;
	std::cin >> type_chooser;
	if (type_chooser == 1) {
		std::cout << "Choose amount of elements"<< '\n';
		std::cout << "~> ";
		std::cin >> arr_size;
		std::cout << "~> ";
		std::vector<Point> temp_vector;
		for (int i = 0; i < arr_size; i++) {
			int x, y;
			std::cin >> x >> y;
			Point temp(x, y);
			temp_vector.push_back(temp);
		}
		switch (command) {
			case predicate_1_f: {
				std::cout << is_palindrome(temp_vector.begin(), temp_vector.end(), [](Point
a, Point b){ return (a == 1) == (b == 1); }) << '\n' << '\n';
				return;
			} case predicate_2_f: {
				std::cout << is_palindrome(temp_vector.begin(), temp_vector.end(), [](Point
a, Point b){ return a % 2 == b % 2; }) << '\n' << '\n';
				return;
			} default: {
				std::cout << "Not implemented yet..." << '\n';
				return;
			}
		}
	} else if (type_chooser == 2) {
		std::cout << "Choose amount of elements" << '\n';
		std::cout << "~> ";
		std::cin >> arr_size;
		std::cout << "~> ";
		std::vector<int> temp_vector;
		for (int i = 0; i < arr_size; i++) {
			int temp_value;
			std::cin >> temp_value;
			temp_vector.push_back(temp_value);
		}
		switch (command) {
			case predicate_1_f: {
				std::cout << is_palindrome(temp_vector.begin(), temp_vector.end(), [](int a,
int b){ return (a == 1) == (b == 1); }) << '\n' << '\n';
				return;
			} case predicate_2_f: {
				std::cout << is_palindrome(temp_vector.begin(), temp_vector.end(), [](int a,
int b){ return a % 2 == b % 2; }) << '\n' << '\n';
				return;
			} default: {
				std::cout << "Not implemented yet..." << '\n';
				return;
			}
		}
	} else {
		std::cout << "========UNKNOWN COMMAND========" << '\n';
	}
}
```

```
bool commands () {
	std::cout << "==========CHOOSE ITEM==========" << '\n';
	std::cout << "all_of - 1" << '\n';
	std::cout << "is_sorted - 2" << '\n';
	std::cout << "is_palindrome - 3" << '\n' << '\n';
	std::cout << "Exit - 0" << '\n';
	std::cout << "================================" << '\n';
	std::cout << "~> ";
	int command;
	std::cin >> command;
	switch (command) {
		case exit_f: {
			return false;
		} case all_of_f: {
			all_of_item();
			return true;
		} case is_sorted_f: {
			is_sorted_item();
			return true;
		} case is_palindrome_f: {
			is_palindrome_item();
			return true;
		} default: {
			std::cout << "=========UNKNOWN COMMAND=========" << '\n';
			return true;
		}
	}
}
```

console_handler.h

```
#pragma once
enum comm_1 {
	exit_f,
	all_of_f,
	is_sorted_f,
	is_palindrome_f,
};
enum comm_2 {
	cancel_f,
	all_3_f,
	all_10_f,
	is_1_f,
	is_2_f,
	predicate_1_f,
	predicate_2_f,
};
bool commands();
void predicate_chooser();
```

algorithms.tpp

```
#pragma once
template<class beginning, class ending, class predicate>
bool all_of(beginning first, ending last, predicate temp_predicate) {
	while (first != last) {
		if (!temp_predicate(*first)) return false;
		first++;
	}
```

```
	return true;
}
template<class beginning, class ending, class predicate>
bool is_sorted(beginning first, ending last, predicate temp_predicate) {
	if (first != last) {
		beginning prev = first;
		while (++first != last) {
			if (!temp_predicate(*prev, *first)) return false;
			prev = first;
		}
	}
	return true;
}
template<class beginning, class ending, class predicate>
bool is_palindrome(beginning first, ending last, predicate temp_predicate) {
	if (first != last) {
		last--;
		while (first < last) {
			if (!temp_predicate(*first, *last)) return false;
			first++;
			last--;
		}
	}
	return true;
}
```

algorithms.hpp

```
#pragma once
template<class beginning, class ending, class predicate>
bool all_of(beginning, ending, predicate);
template<class beginning, class ending, class predicate>
bool is_sorted(beginning, ending, predicate);
template<class beginning, class ending, class predicate>
bool is_palindrome(beginning, ending, predicate);
#include "algorithms.tpp"
```