**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

**«Сибирский государственный университет науки и технологий   
имени академика М.Ф. Решетнева»**

Институт информатики и телекоммуникаций

Кафедра информатики и вычислительной техники

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ**

Языки программирования

|  |
| --- |
| Наследование |

Руководитель А.В. Проскурин

подпись, дата инициалы, фамилия

Обучающийся БПИ21-02 В. М. Самохвалов

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2023 г.

# ЦЕЛЬ РАБОТЫ

Получение практических навыков разработки и отладки программ, c использованием механизма наследования.

# порядок выполнения работы

1. Ознакомиться с общей постановкой задачи.
2. Ознакомится с вариантом задания – соответствует вашему номеру в списке группы (при нехватке заданий вариант задания вычисляется как номер\_в\_списке\_группы - количество\_заданий).
3. Разработать классы согласно варианту задания.
4. Написать и отладить программу на подготовленных наборах тестовых данных.
5. Подготовить отчет по лабораторной работе. Отчет должен включать в себя:

•  титульный лист;

•  цель лабораторной работы;

•  постановку задачи;

•  схему наследования классов (UML диаграмма классов);

•  текст программы с комментариями;

•  демонстрацию работы программы (Снимки экрана при выполнении действий программы с описанием).

•  краткие ответы на контрольные вопросы;

•  выводы по лабораторной работе.

1. Защитить лабораторную работу перед преподавателем.

# постановка задачи

Вариант No21.

Создать класс Line (Линия) с чистым виртуальным методом нахождения длины линии.

На его основе создать классы:

- ColoredLine (Цветная линия) с полем «цвет»;

- PolyLine (Полилиния) с массивом длин участков ломаной линии (кроме первого).

Для массива указателей на объекты этих классов предусмотреть возможность:

- вывода характеристик объектов;

- вывода объектов, длина которых находится в заданном диапазоне;

- вывода массива, отсортированного по длине линий.

Создать класс Picture, содержащий массив указателей на объекты этих классов.

Написать демонстрационную программу, в которой будут использоваться все методы

классов.

# ХОД РАБОТЫ

Main.cpp

#include <iostream>  
#include <vector>  
#include "Inh.cpp"  
*int* main() {  
 *//<editor-fold desc="Point">* Point p1(1, 2), p2(3, 4), p3(5, 6), p4(7, 8), p5(9, 10), p6(11, 12);  
 std::cout << "Points:" << std::endl;  
 std::cout << p1 << ", " << p2 << ", " << p3 << ", " << p4 << ", " << p5 << ", " << p6 << std::endl;  
 *//</editor-fold>  
 //<editor-fold desc="Line">* Line l1(p1, p2), l2(p3, p4), l3(p5, p6);  
 std::cout << "Lines:" << std::endl;  
 std::cout << l1 << " = " << l1.length() << ", " << l2 << " = " << l2.length() << ", " << l3 << " = " << l3.length() << std::endl;  
 *//</editor-fold>  
 //<editor-fold desc="ColoredLine">* ColoredLine cl1(p1, p2, "red"), cl2(p3, p4, "green"), cl3(p5, p6, "blue");  
 std::cout << "Colored lines:" << std::endl;  
 std::cout << cl1 << " = " << cl1.length() << ", " << cl2 << " = " << cl2.length() << ", " << cl3 << " = " << cl3.length() << std::endl;  
 *//</editor-fold>  
 //<editor-fold desc="PolyLine">* std::vector<Line<*int*>> lines;  
 Point<*int*> A(1, 2), B(2,8), C(8,48);  
 Line AB(A, B), BC(B, C);  
 lines.push\_back(AB);  
 lines.push\_back(BC);  
 PolyLine pl1(lines);  
 std::cout << "Polyline:" << std::endl;  
 std::cout << pl1 << " = " << pl1.length() << std::endl;  
 *//sort lines by length* std::cout << "Unsorted lines:" << std::endl;  
 *for* (*auto* line : lines) {  
 std::cout << line << " = " << line.length() << std::endl;  
 }  
 std::sort(lines.begin(), lines.end(), [](Line<*int*> a, Line<*int*> b) { *return* a.length() < b.length(); });  
 std::cout << "Sorted lines:" << std::endl;  
 *for* (*auto* line : lines) {  
 std::cout << line << " = " << line.length() << std::endl;  
 }  
 *//</editor-fold>  
 //<editor-fold desc="Filters">* std::cout << "Filters" << std::endl;  
 std::vector<Line<*int*>> lines2;  
 Point<*int*> A2(1, 2), B2(2,8), C2(8,48), D2(1, 2), E2(2,8), F2(8,48);  
 Line AB2(A2, B2), BC2(B2, C2), DE2(D2, E2), EF2(E2, F2);  
 lines2.push\_back(AB2);  
 lines2.push\_back(BC2);  
 lines2.push\_back(DE2);  
 lines2.push\_back(EF2);  
 PolyLine pl2(lines2);  
 std::cout << "Polyline:" << std::endl;  
 std::cout << pl2 << " = " << pl2.length() << std::endl;  
 std::vector<Line<*int*>> filteredLines = pl2.filterByLength(10, 100);  
 std::cout << "Filtered lines:" << std::endl;  
 *for* (*auto* line : filteredLines) {  
 std::cout << line << " = " << line.length() << std::endl;  
 }  
 *//</editor-fold>  
 return* 0;  
}

Inh.h

#ifndef **INH\_INH\_H**#define **INH\_INH\_H**#include <string>  
#include <utility>  
#include <iostream>  
#include <cmath>  
  
*template*<*typename* T>  
*class* Point {  
*private*:  
 T x;  
 T y;  
*public*:  
 Point(T x, T y);  
  
 Point() = *default*;  
  
 T getX();  
  
 *void* setX(T \_x);  
  
 T getY();  
  
 *void* setY(T \_y);  
};  
  
*template*<*typename* T>  
*class* Line : *public* Point<T> {  
*private*:  
 Point<T> p1;  
 Point<T> p2;  
*public*:  
 Line(Point<T> p1, Point<T> p2);  
  
 Line() = *default*;  
  
 Point<*int*> getP1();  
  
 *void* setP1(*const* Point<T> &\_p1);  
  
 *const* Point<T> &getP2();  
  
 *void* setP2(*const* Point<T> &\_p2);  
  
 *//с чистым виртуальным методом нахождения длины линии.  
 virtual* T length();  
};  
  
  
*template*<*typename* T>  
*class* ColoredLine : *public* Line<T> {  
*private*:  
 std::string color;  
*public*:  
 ColoredLine(Point<T> p1, Point<T> p2, std::string \_color);  
  
 ColoredLine() = *default*;  
  
 std::string getColor();  
  
 *void* setColor(*const* std::string &\_color);  
  
 T length() *override*;  
  
};  
  
  
*template*<*typename* T>  
*class* PolyLine : *public* Line<T> {  
 std::vector<Line<T>> lines;  
  
*public*:  
 *explicit* PolyLine(std::vector<Line<T>> &lines);  
  
 std::vector<Line<T>> &getLines();  
  
 *void* setLines(std::vector<Line<T>> &\_lines);  
  
 T length() *override*;  
  
 std::vector<Line<T>> filterByLength(T min, T max);  
};  
  
*template*<*typename* T>  
T PolyLine<T>::length() {  
 T sum = 0;  
 *for* (*auto* &line: lines) {  
 sum += line.length();  
 }  
 *return* sum;  
}  
  
#endif *//INH\_INH\_H*

Inh.cpp

#include <utility>  
#include <vector>  
#include "Inh.h"  
  
*//<editor-fold desc="Point">  
template*<*typename* T>  
Point<T>::Point(T x, T y) : x(x), y(y) {}  
  
*template*<*typename* T>  
T Point<T>::getX() {  
 *return* x;  
}  
  
*template*<*typename* T>  
T Point<T>::getY() {  
 *return* y;  
}  
  
*template*<*typename* T>  
*void* Point<T>::setY(T \_y) {  
 y = \_y;  
}  
  
*template*<*typename* T>  
*void* Point<T>::setX(T \_x) {  
 x = \_x;  
}  
  
*template*<*typename* T>  
std::ostream &*operator*<<(std::ostream &os, Point<T> &point) {  
 os << "(" << point.getX() << ", " << point.getY() << ")";  
 *return* os;  
}  
  
*//</editor-fold>  
//<editor-fold desc="Line">  
template*<*typename* T>  
Line<T>::Line(Point<T> p1, Point<T> p2) : p1(p1), p2(p2) {}  
  
*template*<*typename* T>  
Point<*int*> Line<T>::getP1() {  
 *return* p1;  
}  
  
*template*<*typename* T>  
*void* Line<T>::setP1(*const* Point<T> &\_p1) {  
 p1 = \_p1;  
}  
  
*template*<*typename* T>  
*const* Point<T> &Line<T>::getP2() {  
 *return* p2;  
}  
  
*template*<*typename* T>  
*void* Line<T>::setP2(*const* Point<T> &\_p2) {  
 p2 = \_p2;  
}  
  
*template*<*typename* T>  
T Line<T>::length() {  
 *return* sqrt(pow(p1.getX() - p2.getX(), 2) + pow(p1.getY() - p2.getY(), 2));  
}  
  
*template*<*typename* T>  
std::ostream &*operator*<<(std::ostream &os, Line<T> &line) {  
 Point p1 = line.getP1();  
 Point p2 = line.getP2();  
 os << "[" << p1 << ", " << p2 << "]";  
 *return* os;  
}  
  
*//</editor-fold>  
//<editor-fold desc="ColoredLine">  
template*<*typename* T>  
ColoredLine<T>::ColoredLine(Point<T> p1, Point<T> p2, std::string \_color) : Line<T>(p1, p2), color(std::move(\_color)) {}  
  
*template*<*typename* T>  
std::string ColoredLine<T>::getColor() {  
 *return* color;  
}  
  
*template*<*typename* T>  
*void* ColoredLine<T>::setColor(*const* std::string &\_color) {  
 color = \_color;  
}  
  
*template*<*typename* T>  
std::ostream &*operator*<<(std::ostream &os, ColoredLine<T> &coloredLine) {  
 Point p1 = coloredLine.getP1();  
 Point p2 = coloredLine.getP2();  
 os << "[" << p1 << ", " << p2 << ", " << coloredLine.getColor() << "]";  
 *return* os;  
}  
  
*//lenght  
template*<*typename* T>  
T ColoredLine<T>::length() {  
 *return* Line<T>::length();  
}  
*//</editor-fold>  
//<editor-fold desc="PolyLine">  
template*<*typename* T>  
PolyLine<T>::PolyLine(std::vector<Line<T>> &lines) {  
 *this*->lines = lines;  
}  
  
  
*template*<*typename* T>  
*void* PolyLine<T>::setLines(std::vector<Line<T>> &\_lines) {  
 lines = \_lines;  
}  
  
*template*<*typename* T>  
std::vector<Line<T>> &PolyLine<T>::getLines() {  
 *return* lines;  
}  
  
*template*<*typename* T>  
std::ostream &*operator*<<(std::ostream &os, PolyLine<T> &polyLine) {  
 std::vector<Line<T>> lines = polyLine.getLines();  
 std::cout << "{";  
 *for* (Line<T> line: lines) {  
 os << line << ", ";  
 }  
 os << "}";  
 *return* os;  
}  
*//filter by length between min and max  
template*<*typename* T>  
std::vector<Line<T>> PolyLine<T>::filterByLength(T min, T max) {  
 std::vector<Line<T>> filteredLines;  
 *for* (Line<T> line: lines) {  
 T length = line.length();  
 *if* (length >= min && length <= max) {  
 filteredLines.push\_back(line);  
 }  
 }  
 *return* filteredLines;  
}  
*//</editor-fold*

Вывод

/Users/vadimsam/CLionProjects/inh/cmake-build-debug/inh

Points:

(1, 2), (3, 4), (5, 6), (7, 8), (9, 10), (11, 12)

Lines:

[(1, 2), (3, 4)] = 2, [(5, 6), (7, 8)] = 2, [(9, 10), (11, 12)] = 2

Colored lines:

[(1, 2), (3, 4), red] = 2, [(5, 6), (7, 8), green] = 2, [(9, 10), (11, 12), blue] = 2

Polyline:

{[(1, 2), (2, 8)], [(2, 8), (8, 48)], } = 46

Unsorted lines:

[(1, 2), (2, 8)] = 6

[(2, 8), (8, 48)] = 40

Sorted lines:

[(1, 2), (2, 8)] = 6

[(2, 8), (8, 48)] = 40

Filters

Polyline:

{[(1, 2), (2, 8)], [(2, 8), (8, 48)], [(1, 2), (2, 8)], [(2, 8), (8, 48)], } = 92

Filtered lines:

[(2, 8), (8, 48)] = 40

[(2, 8), (8, 48)] = 40

Process finished with exit code 0

# ВЫВОДЫ

Получены практические навыки разработки и отладки программ, c использованием механизма наследования.