МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Лабораторная работа №5**

по дисциплине: Объектно-ориентированное программирование Тема: Классы, виды отношений. Наследование.

Выполнил: студент группы ПВ-223

Дмитриев А.А.

Проверил:

Черников С.В.

Белгород 2024 г.

**Цель работы:** Получение теоретических знаний в области разработки классов, получение практических навыков реализаций классов и отношений между ними.

**Задание:** Программа решения заданных произвольных уравнений.

1. Создать абстрактный класс Function с методом вычисления значения

функции y=f(x) в заданной точке.

2. Создать производные классы:

Line (y=ax+b),

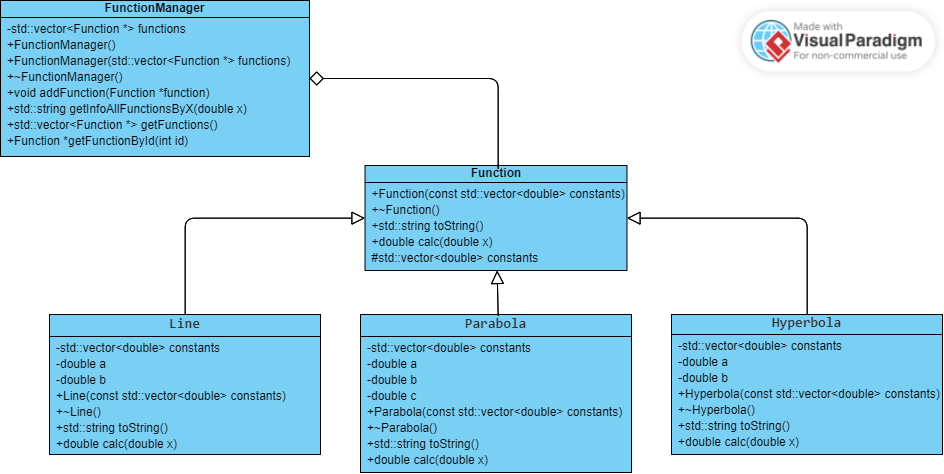
Parabola (y=ax 2 +bx +c),

Hyperbola (y=a/x+b)

со своими методами вычисления значения в заданной точке.

3. Создать массив n функций и вывести полную информацию о значении

данных функций в точке x.



class Function

{

private:

public:

    Function(const std::vector<double> constants);

    ~Function();

    virtual std::string toString();

    virtual double calc(double x) = 0;

protected:

    std::vector<double> constants;

};

class Hyperbola : public Function

{

private:

    std::vector<double> constants;

    double a;

    double b;

public:

    Hyperbola(std::vector<double> constants);

    ~Hyperbola();

    double calc(double x) override;

    std::string toString() override;

};

class Line : public Function

{

private:

    std::vector<double> constants;

public:

    Line(std::vector<double> constants);

    ~Line();

    double calc(double x) override;

    std::string toString() override;

protected:

    double a;

    double b;

};

class Parabola : public Function

{

private:

    std::vector<double> constants;

public:

    Parabola(std::vector<double> constants);

    ~Parabola();

    double calc(double x) override;

    std::string toString() override;

protected:

    double a;

    double b;

    double c;

};

class FunctionManager

{

private:

    std::vector<Function \*> functions;

public:

    FunctionManager();

    FunctionManager(std::vector<Function \*> functions);

    ~FunctionManager();

    Function \*getFunctionById(int id);

    std::vector<Function \*> getFunctions();

    std::string getInfoAllFunctionsByX(double x);

    void addFunction(Function \*function);

};

Function::Function(std::vector<double> constants) {

    this->constants = constants;

}

Function::~Function() {

    delete this;

}

std::string Function::toString() {

    std::string res;

    for (double constant : constants)

        res += std::to\_string(constant) + " ";

    return res;

}

Hyperbola::Hyperbola(std::vector<double> constants) : Function::Function(constants)

{

    if (constants.size() != 2)

        throw std::invalid\_argument("num of constants not equal 2");

    a = constants[0];

    b = constants[1];

}

Hyperbola::~Hyperbola() { delete this; }

double Hyperbola::calc(double x)

{

    return a / x + b;

}

std::string Hyperbola::toString() {

    return std::to\_string(a) + " / x + " + std::to\_string(b);

}

Line::Line(std::vector<double> constants) : Function::Function(constants)

{

    if (constants.size() != 2)

        throw std::invalid\_argument("num of constants not equal 2");

    a = constants[0];

    b = constants[1];

}

Line::~Line() {

    delete this;

}

double Line::calc(double x)

{

    return a \* x + b;

}

std::string Line::toString() {

    return std::to\_string(a) + " \* x + " + std::to\_string(b);

}

Parabola::Parabola(std::vector<double> constants) : Function::Function(constants)

{

    if (constants.size() != 3)

        throw std::invalid\_argument("num of constants not equal 3");

    a = constants[0];

    b = constants[1];

    c = constants[2];

}

Parabola::~Parabola()

{

    delete this;

}

double Parabola::calc(double x)

{

    return a \* x \* x + b \* x + c;

}

std::string Parabola::toString() {

    return std::to\_string(a) + " \* x2 + " + std::to\_string(b) + " x + " + std::to\_string(c);

}

FunctionManager::FunctionManager()

{

}

FunctionManager::FunctionManager(std::vector<Function \*> functions)

{

    this->functions = functions;

}

FunctionManager::~FunctionManager()

{

    delete this;

}

void FunctionManager::addFunction(Function \*function)

{

    functions.push\_back(function);

}

std::vector<Function \*> FunctionManager::getFunctions()

{

    return functions;

}

Function \*FunctionManager::getFunctionById(int id)

{

    return functions[id];

}

std::string FunctionManager::getInfoAllFunctionsByX(double x)

{

    std::string res;

    for (Function \*function : functions)

        res += "f(" + std::to\_string(x) + ") = " + function->toString() + " = " + std::to\_string(function->calc(x)) + "\n";

    res += "\b";

    return res;

}

Пример работы:

int main()

{

    Line l1({2, 2});

    Line l2({2, 2});

    Parabola p1({3, 2, 1});

    Line l3({2, 2});

    Hyperbola h1({2, 2});

    FunctionManager fm({&l1, &l2, &l3});

    fm.addFunction(&p1);

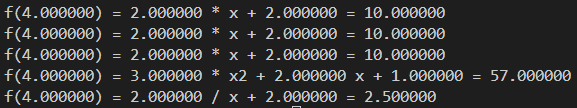
    fm.addFunction(&h1);

    std::cout << fm.getInfoAllFunctionsByX(4);

    return 0;

}

Выходные данные:



**Вывод:** В ходе лабораторной работы получили теоретические знания в области разработки классов, получили практические навыки реализации классов и отношений между ними.