Министерство высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

по лабораторной работе №1

Тема: «Классы и объекты. Инкапсуляция.»

Выполнил

Студент группы РИС-22-2б

Радостев И.А.

Проверил доц. Кафедры ИТАС

Полякова Ольга Андреевна

Пермь 2023

# Постановка задачи

1. Реализовать определение нового класса. Для демонстрации работы с объектами написать главную функцию. Продемонстрировать разные способы создания объектов и массивов объектов.
2. Структура-пара – структура с двумя полями, которые обычно имеют имена first и second. Требуется реализовать тип данных с помощью такой структуры. Во всех заданиях должны присутствовать :
   1. метод инициализации Init (метод должен контролировать значения аргументов на корректность);
   2. ввод с клавиатуры Read;
   3. вывод на экран Show.
3. Реализовать внешнюю функцию make\_тип(), где тип – тип реализуемой структуры. Функция должна получать значения для полей структуры как параметры функции и возвращать структуру как результат. При передаче ошибочных параметров следует выводить сообщение и заканчивать работу.

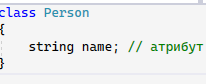
ВАРИАНТ 6:

Линейное уравнение y=Ax+B. поле first-дробное число, коэффициент A, поле second- дробное число, коэффициент B. реализовать метод root()- вычисление корня линейного уравнения, мотод должен проверять неравенство коэффициента A нулю..

# Контрольные вопросы

* 1. Что такое класс?

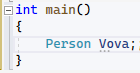
*Класс* - абстрактный тип данных, определяемым пользователем. Представляет собой модель реального объекта в виде данных и функций для работы с ними.



* 1. Что такое объект (экземпляр) класса?

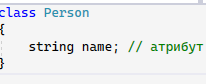
*Экземпляр класса –* это описание конкретного объекта класса.

Класс – Person, объект – Vova.



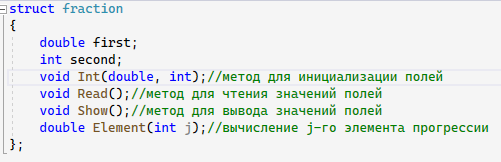
* 1. Как называются поля класса?

*Атрибуты.*



* 1. Как называются функции класса?

*Методы.*



* 1. Для чего используются спецификаторы доступа?

Управление видимостью элементов класса (чтение/запись).

* 1. Для чего используется спецификатор public?

Для общедоступных элементов класса

* 1. Для чего используется спецификатор private?

Для закрытых элементов класса

* 1. Если описание класса начинается со спецификатора class, то какой спецификатор доступа будет использоваться по умолчанию?

Private

* 1. Если описание класса начинается со спецификатора struct, то какой спецификатор доступа будет использоваться по умолчанию?

Public

* 1. Какой спецификатор доступа должен использоваться при описании интерфейса класса? Почему?

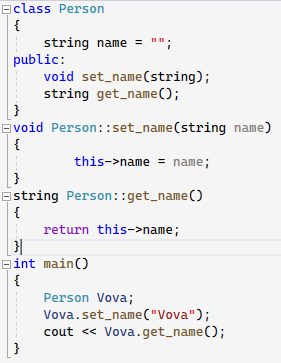
Public. Так как нужно иметь доступ к методу, чтобы его описать в дальнейшем.

* 1. Каким образом можно изменить значения атрибутов экземпляра класса?

С помощью метода – модификатора (сеттер).

* 1. Каким образом можно получить значения атрибутов экземпляра класса?

С помощью метода – селектор (геттер).





struct Student

{

string name;

int group;

………

};

Объект класса определен следующим образом

Student \*s=new Student;

Как можно обратиться к полю name объекта s?

s->name;



struct Student

{

string name;

int group;

……..

};

Объект класса определен следующим образом

Student s;

Как можно обратиться к полю name объекта s?

s.name;



class Student

{

string name;

int group;

…….

};

Объект класса определен следующим образом Student \*s=new Student;

Как можно обратиться к полю name объекта s?

С помощью геттера



class Student

{

string name;

int group;

public:

Объект класса определен следующим образом Student s;

Как можно обратиться к полю name объекта s?

С помощью геттера



class Student

{

public:

char\* name;

int group;

………

};

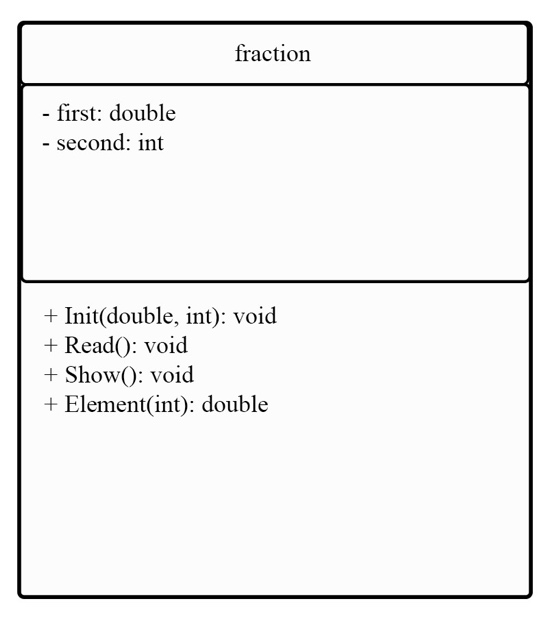
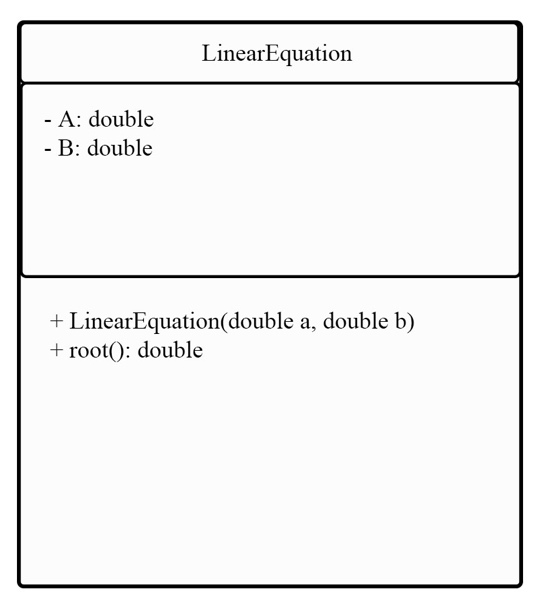
Объект класса определен следующим образом

Student \*s=new Student;

Как можно обратиться к полю name объекта s?

s->name.

UML-диаграмма



Код программы:

Fraction.h:

#pragma once

#include <iostream>

using namespace std;

struct fraction

{

double first;

int second;

void Init(double, int);//метод для инициализации полей

void Read();//метод для чтения значений полей

void Show();//метод для вывода значений полей

double Element(int j );//вычисление j-го элемента прогрессии.

};

Fraction.cpp:

#include "C:\Users\skory\source\repos\Laba 1\fraction.h"

//реализация метода для инициализации полей структуры

void fraction::Init(double F, int S)

{

first = F;

second = S;

}

//реализация метода для чтения значений полей структуры

void fraction::Read()

{

cout << "\nfirst?"; cin >> first;

cout << "\nsecond?"; cin >> second;

}

//реализация метода для вывода значений полей структуры

void fraction::Show()

{

cout << "\nfirst=" << first;

cout << "\nsecond=" << second;

cout << "\n";

}

//метод для возведения в степень

double fraction::Element(int j)

{

return first \* pow(second, j);

}LAB\_OOP\_1.cpp:

#include <iostream>

using namespace std;

class LinearEquation { // объявление класса LinearEquation

private:

double A; // поле first - коэффициент A

double B; // поле second - коэффициент B

public:

LinearEquation(double a, double b) { // конструктор класса с двумя аргументами типа double

A = a;

B = b;

}

double root() { // метод root() вычисляет корень линейного уравнения и возвращает его значение

if (A == 0) { // проверка неравенства коэффициента A нулю

cout << "Ошибка: коэффициент A равен нулю" << endl; // вывод сообщения об ошибке

return -1; // возвращаем -1, так как корня нет

}

else {

return -B / A; // вычисляем корень линейного уравнения и возвращаем его значение

}

}

};

int main() {

double a, b;

cout << "Введите коэффициенты линейного уравнения (A и B): ";

cin >> a >> b;

LinearEquation equation(a, b); // создание объекта класса LinearEquation с заданными коэффициентами

double result = equation.root(); // вызов метода root() для вычисления корня линейного уравнения

if (result != -1) { // если метод root() вернул значение отличное от -1, то выводим результат

cout << "Корень линейного уравнения: " << result << endl;

}

return 0;

}