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

Барыбин К.Ю.

Проверил доц. Кафедры ИТАС

Полякова Ольга Андреевна

Пермь 2023

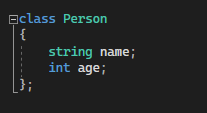
# Постановка задачи

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.

# Контрольные вопросы

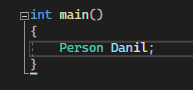
* 1. Что такое класс?

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



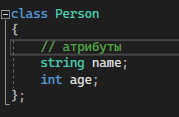
* 1. Что такое объект (экземпляр) класса?

*Экземпляр класса –* это описание конкретного объекта класса. Допустим: Класс – человек, объект – Данил.



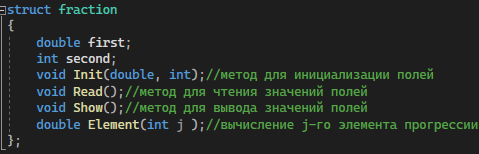
* 1. Как называются поля класса?

*Атрибуты.*



* 1. Как называются функции класса?

*Методы.*



* 1. Для чего используются спецификаторы доступа?

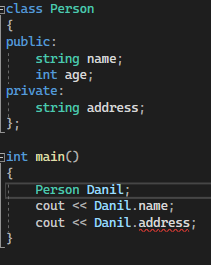
Управление видимостью элементов класса (чтение/запись).

* 1. Для чего используется спецификатор public?

Для общедоступных элементов класса

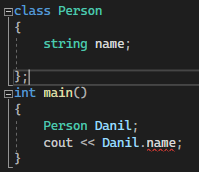
* 1. Для чего используется спецификатор private?

Для закрытых элементов класса.



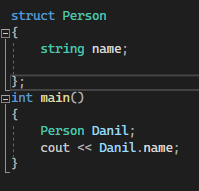
* 1. Если описание класса начинается со спецификатора class, то какой спецификатор доступа будет использоваться по умолчанию?

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?

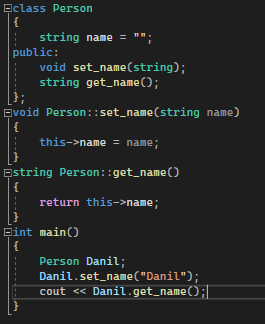
Public. Так как нужно иметь доступ к методу, чтобы его описать в дальнейшем.

* 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;

………

};

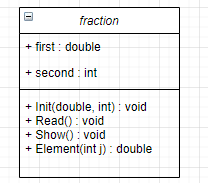
Объект класса определен следующим образом

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\Dimak\Desktop\С++\Lab\_OOP\_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 "C:\Users\Dimak\Desktop\С++\Lab\_OOP\_1\fraction.h"

fraction make\_fraction(double F, int S)

{

fraction t;//создали временную переменную

t.Init(F, S);//инициализировали поля переменной t с помощью параметров функции

return t;//вернули значение переменной t

}

int main()

{

//определение переменных А и В

fraction A;

fraction B;

A.Init(3.0, 2);//инициализация переменной А

B.Read();//ввод полей переменных В

A.Show();//вывод значений полей переменной А

B.Show();//вывод значений полей переменной В

//вывод значения степени, вычисленного с помощью функции Element(5)

cout << "A.Power(" << A.first << "," << A.second << ")=" << A.Element(5) << endl;

cout << "B.Power(" << B.first << "," << B.second << ")=" << B.Element(5) << endl;

//указатели

fraction\* X = new fraction;//выделение памяти под динамическую переменную

X->Init(2.0, 5);//инициализация

X->Show();//вывод значений полей

X->Element(5);//вычисление степени

cout << "X.Element(" << X->first << "," << X->second << ")=" << X->Element(5) << endl;

//массивы

delete X;

fraction mas[3];//определение массива

for (int i = 0; i < 3; i++)

mas[i].Read();//чтение значений полей

for (int i = 0; i < 3; i++)

mas[i].Show(); //вывод значений полей

for (int i = 0; i < 3; i++)

{

mas[i].Element(5); //вычисление степени

cout << "mas[" << i << "].Power(" << mas[i].first << "," << mas[i].second << ")=";

cout << mas[i].Element(5) << endl;

}

//динамические масивы

fraction\* p\_mas = new fraction[3];//выделение памяти

for (int i = 0; i < 3; i++)

p\_mas[i].Read();//чтение значений полей

for (int i = 0; i < 3; i++)

p\_mas[i].Show();//вывод значений полей

for (int i = 0; i < 3; i++)

{

p\_mas[i].Element(5);//вычисление степени

cout << "p\_mas[" << i << "].Power(" << p\_mas[i].first << "," << p\_mas[i].second;

cout << ")=" << p\_mas[i].Element(5) << endl;

}

delete[] p\_mas;

//вызов функции make\_fraction()

double y; int z;

cout << "first?"; cin >> y;

cout << "second?"; cin >> z;

//переменная F формируется с помощью функции make\_fraction()

fraction F = make\_fraction(y, z);

F.Show();

system("pause");

return 0;

}