Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: «Классы и объекты. Инкапсуляция»

Выполнил работу

Студент группы РИС-23-3Б

Асташин Д.А.

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2024

**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 имеют имена first и second. Требуется реализовать тип данных с помощью такой структуры. Во всех заданиях должны присутствовать:

a. метод инициализации Init (метод должен контролировать значения

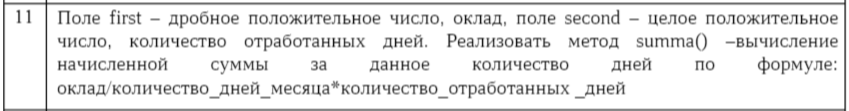
аргументов на корректность);

b. Ввод с клавиатуры Read;

с. 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.



**Алгоритм решения:**

1. Создаем структуру fraction.
2. Создаем методы инициализации, ввода, вывода и суммы.
3. В главной функции последовательно проверяем работоспособность всех функции.

**Код программы:**

Fraction.h

#pragma once

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 Summa(); // метод для вычисления начисленной суммы за данное количество дней

};

Fraction.cpp

#include <iostream>

#include "fraction.h"

using namespace std;

//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 << "Введите оклад: "; cin >> first;

cout << "Введите количество отработанных дней: "; cin >> second;

}

//реализация метода для вывода значений полей структуры

void fraction::Show()

{

cout << "Оклад: " << first << " ; ";

cout << "Количество отработанных дней: " << second;

cout << "\n";

}

//метод ля вычисления суммы за данное количество рабочих дней

double fraction::Summa()

{

//return (first \* second) / 30;

return first / 30 \* second;

}

Lab1\_main.cpp

#include <iostream>

#include "fraction.h"

using namespace std;

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() {

system("chcp 1251");

fraction A;

A.Init(1, 1);

A.Read();

A.Show();

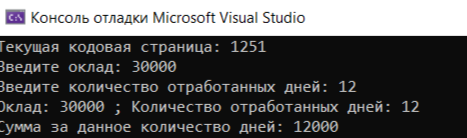
cout << "Сумма за данное количество дней: ";

cout << A.Summa();

return 0;

}

**Результат работы программы:**

****

**Ответы на контрольные вопросы:**

1. Что такое класс?

Класс – это абстрактный тип данных, определяемый пользователем. Представляет собой модель реального объекта в виде данных и функций для работы с ними.

1. Что такое объект (экземпляр) класса?

Объект (экземпляр) класса - это переменная типа класса. Их может быть бесконечное количество.

1. Как называются поля класса?

Поля - переменные внутри класса, также называются атрибутами класса.

1. Как называются функции класса?

Классы имеют свои функции, которые называются методами класса.

1. Для чего используются спецификаторы доступа?

Спецификаторы доступа используются для изменения видимость компонентов. Всего есть 3 спецификатора public, private, protected. В структуре (struct) все поля и методы по умолчанию имеют модификатор доступа public, а в классе (class) все поля и методы по умолчанию имеют модификатор доступа private.

1. Для чего используется спецификатор public?

При спецификаторе доступа public общедоступные компоненты доступны в любой части программы. Они могут быть использованы любой функцией как внутри класса, так и вне его. Доступ из вне осуществляется через имя объекта.

1. Для чего используется спецификатор private?

При спецификаторе доступа private собственные компоненты локализованы в классе и не доступны извне. Они могут использоваться функциями - членами данного класса и функциями - "друзьями" того класса, в котором они описаны.

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

По умолчанию будет исполльзоваться private.

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

По умолчанию будет использоваться public.

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

Спецификатор public - для доступа к методам интерфейса.

1. Каким образом можно изменить значения атрибутов экземпляра класса?

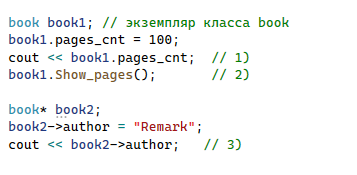


1. Каким образом можно получить значения атрибутов экземпляра класса?

1. Напрямую

2. Через public методы классса

3. Через указатель.



1. Класс описан следующим образом 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 = "name"

1. Класс описан следующим образом class Student

{ string name; int group; ……. }; Объект класса определен следующим образом Student \*s=new Student; Как можно обратиться к полю name объекта s?

s->name

1. Класс описан следующим образом class Student { string name; int group; public: ….. }; Объект класса определен следующим образом Student s; Как можно обратиться к полю name объекта s?

s.name

1. Класс описан следующим образом 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

**Ссылка на Git:**

https://github.com/astidii/PNRPU/tree/main/1%20semester/OOP%20labs/1%20lab