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: Лабораторная работа №1

# « Классы и объекты. Инкапсуляция »

# Семестр 2

Выполнил работу

Студент группы РИС-22-1Б

Зырянов Ростислав

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Г. 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 : a. метод инициализации Init (метод должен контролировать значения аргументов на корректность); b. ввод с клавиатуры Read; c. 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 15**

Элемент арифметической прогрессии aj вычисляется по формуле:   
aj=a0·r j. Поле first – дробное число, первый элемент прогрессии a0, поле second – положительное целое число, постоянное отношение r. Реализовать метод element(int j) –вычисление j-го элемента прогрессии.

**Ответы на контрольные вопросы**

1. **Что такое класс?** Класс – пользовательский тип данных. Шаблон для создания объектов, который определяет состояние (поля) и поведение (методы) объекта.
2. **Что такое объект (экземпляр) класса?** Объект (экземпляр) класса - это конкретный экземпляр класса, созданный на основе его определения. Он имеет состояние (значения полей) и поведение (реализацию методов), определенные в классе.
3. **Как называются поля класса?** Поля класса - это переменные, определенные внутри класса, которые хранят состояние объекта.
4. **Как называются функции класса?** Функции класса - это методы, определенные внутри класса, которые определяют поведение объекта.
5. **Для чего используются спецификаторы доступа?** Спецификаторы доступа используются для определения того, как поля и методы класса могут быть доступны извне класса. Они контролируют уровень доступа к данным и методам объекта.
6. **Для чего используется спецификатор public?** Спецификатор public используется для указания того, что поля и методы класса могут быть доступны извне класса, в том числе из других файлов и модулей.
7. **Для чего используется спецификатор private?** Спецификатор private используется для указания того, что поля и методы класса могут быть доступны только изнутри класса и не могут быть использованы извне класса.
8. **Если описание класса начинается со спецификатора class, то какой спецификатор доступа будет использоваться по умолчанию?** Если описание класса начинается со спецификатора class, то по умолчанию используется спецификатор private.
9. **Если описание класса начинается со спецификатора struct, то какой спецификатор доступа будет использоваться по умолчанию?** Если описание класса начинается со спецификатора struct, то по умолчанию используется спецификатор public.
10. **Какой спецификатор доступа должен использоваться при описании интерфейса класса? Почему?** При описании интерфейса класса должен использоваться спецификатор public, потому что интерфейс класса определяет публичные методы и поля, доступные для использования извне класса.
11. **Каким образом можно изменить значения атрибутов экземпляра класса?** Значения атрибутов экземпляра класса можно изменять непосредственно через оператор "." (точка), обращаясь к соответствующему полю объекта и присваивая ему новое значение.
12. **Каким образом можно получить значения атрибутов экземпляра класса?** Значения атрибутов экземпляра класса можно получить непосредственно через оператор "." (точка), обращаясь к соответствующему полю объекта.
13. **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?** Для доступа к полю name объекта s, необходимо использовать оператор "->" (стрелка): s->name.
14. **Класс описан следующим образом struct Student { string name; int group; …….. }; Объект класса определен следующим образом Student s; Как можно обратиться к полю name объекта s?** Для доступа к полю name объекта s, необходимо использовать оператор "." (точка): s.name.
15. **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?** Для доступа к полю name объекта s, необходимо использовать оператор "->" (стрелка): s->name.
16. **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?** Для доступа к полю name объекта s, необходимо использовать оператор "." (точка): s.name.
17. **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?** Для доступа к полю name объекта s необходимо использовать оператор ->. Например, чтобы изменить значение поля name, можно написать: s->name = "John";.

**Диаграмма класса**

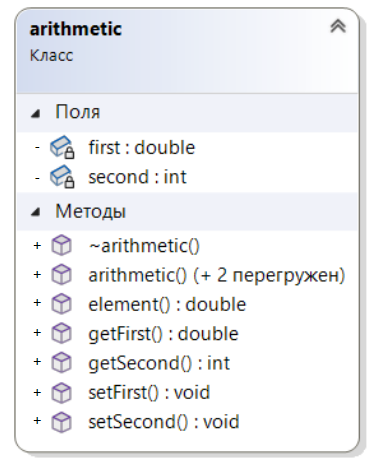


Рисунок 1 – класс arithmetic

**Описание класса**

#pragma once

class arithmetic

{

private:

double first;

int second;

public:

arithmetic();

arithmetic(double first, int second);

arithmetic(arithmetic const& object);

double element(int j);

void setFirst(double first);

void setSecond(int second);

double getFirst();

int getSecond();

~arithmetic();

};

**Определение методов**

#pragma once

#include "Header.h"

arithmetic::arithmetic()

{

first = 0;

second = 0;

}

double arithmetic::element(int j)

{

return first \* pow(second, j-1);

}

void arithmetic::setFirst(double first)

{

this->first = first;

}

void arithmetic::setSecond(int second)

{

this->second = second;

}

double arithmetic::getFirst()

{

return first;

}

int arithmetic::getSecond()

{

return second;

}

arithmetic::arithmetic(double first, int second)

{

this->first = first;

this->second = second;

}

arithmetic::arithmetic(const arithmetic& object)

{

this->first = object.first;

this->second = object.second;

}

arithmetic::~arithmetic()

{

first = 0;

second = 0;

}

**Результаты работы программы**

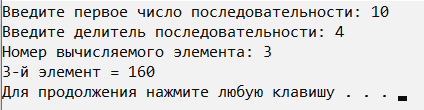


Рис.2 – результат работы программы