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

Бреднев М.П.

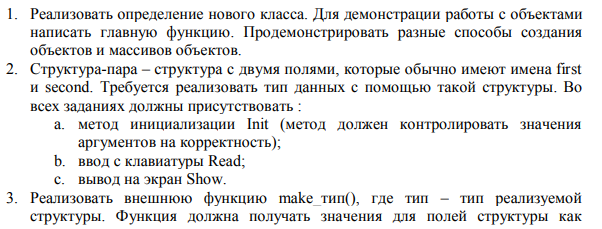
Проверил

Доцент кафедры ИТАС

Полякова О.А.

Г. Пермь-2023

**Постановка задачи**

**Вариант 7**



**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

**Код программы**

header.h

#pragma once

struct fraction

{

int first;

int second;

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

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

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

double Function(double X); // приведение времени в минуты

};

Source.cpp

#include <iostream>

#include "Header.h"

using namespace std;

// метод инициализации полей структуры

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

{

first = F;

second = S;

}

// метод для чтения значений полей

void fraction::Read()

{

cout << endl << "Первое >> "; cin >> first;

cout << "Второе >> "; cin >> second;

}

// метод для вывода значений полей

void fraction::Show()

{

cout << endl << "Первое = " << first;

cout << endl << "Второе = " << second;

cout << endl;

}

// метод для вычисления суммы

double fraction::Function(double X)

{

return (X \* first + second);

}

main.cpp

#include <iostream>

#include "Header.h"

#include <Windows.h>

using namespace std;

fraction make\_fraction(int F, int S)

{

fraction t;

t.Init(F, S);

return t;

}

void main()

{

setlocale(0, "");

// Определение переменных А и В

cout << endl << "===”\*°•.”\*°• ГлАвНое •°\*”.•°\*”===" << endl;

fraction A;

fraction B;

double X;

cout << "Введите Х : "; cin >> X;

A.Init(3, 40);

B.Read();

A.Show();

B.Show();

cout << endl;

// Вывод суммы Ax+B

cout << "Ответ A : " << A.Function(X) << endl;

cout << "Ответ B : " << B.Function(X) <<endl;

// Указатели

cout << endl << "===”\*°•.”\*°• УкАзАтЕлИ •°\*”.•°\*”===" << endl;

fraction\* C = new fraction;

C->Init(0, 0);

C->Show();

C->Read();

C->Function(X);

cout << "Ответ C : " << C->Function(X) << endl;;

// Массивы

cout << endl << "===”\*°•.”\*°• МаСсИвЫ •°\*”.•°\*”===" << endl;

fraction M[2];

for (int i = 0; i < 2; i++)

{

M[i].Read();

}

for (int i = 0; i < 2; i++)

{

cout << "Результат номер : " << i + 1 << " = " << M[i].Function(X) << endl;

}

// Динамические массивы

cout << endl << "===”\*°•.”\*°• ДиНаМиЧеСкИе МаСсИвЫ •°\*”.•°\*”===" << endl;

int n;

cout << "N : "; cin >> n;

fraction\* DM = new fraction[n];

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

{

DM[i].Read();

}

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

{

cout << "Ответ номер : " << i + 1 << " = " << DM[i].Function(X) << endl;

}

delete DM;

// Вызов функции make\_fraction

cout << endl << "===”\*°•.”\*°• MaKe\_FrAcTiOn •°\*”.•°\*”===" << endl;

int x, y;

cout << "Первое >> "; cin >> x;

cout << "Второе >> "; cin >> y;

// Формирование переменной(Т) с помощью функции make\_fraction

fraction T = make\_fraction(x, y);

cout <<"Ответ : " << T.Function(X);

}

**Результаты работы программы**

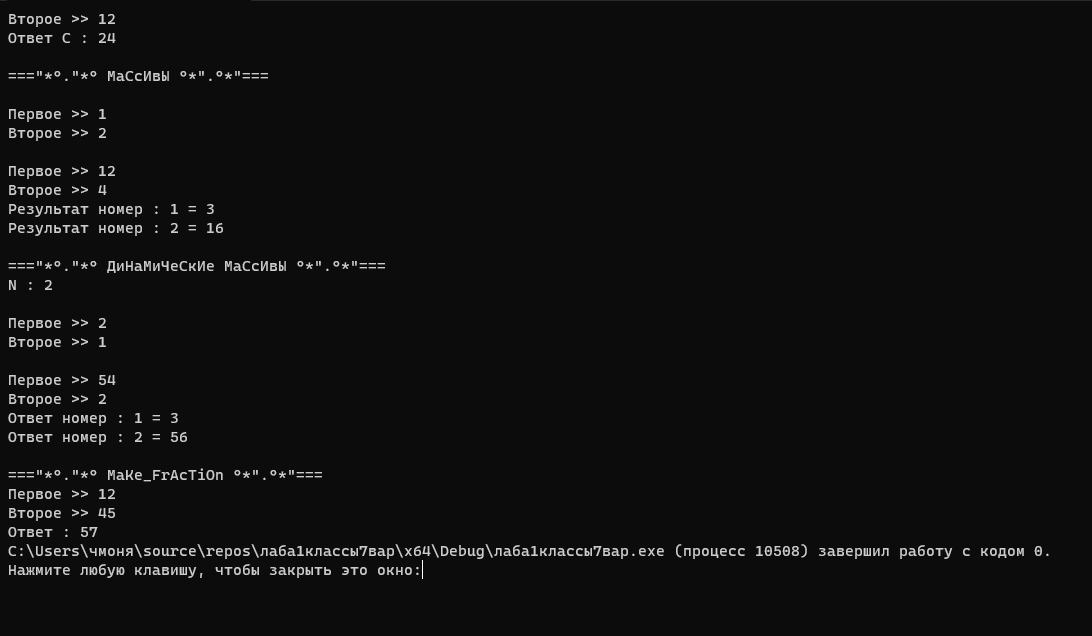
****

Рис.2 – результат работы программы