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: «Информатика»

Тема: Классы и объекты. Инкапсуляция

Семестр 2

Выполнил работу

Студент группы РИС-22-1Б

Аксёнов Д.О.

Проверил

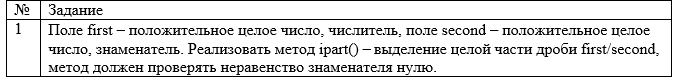
Доцент кафедры ИТАС

Полякова О.А.

Г. Пермь-2023

**Постановка задачи (общая и конкретного варианта).**

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.



**Fraction.h**

#pragma once

#include <iostream>

#include "fraction.h"

using namespace std;

struct fraction

{

double first;

int second;

void Init(int, int);

void Read(); //метод для инициализации полей void Read();

//метод для чтения значений полей

void Show();

int Power();

int ipart();

//метод для вывода значений полей double Power();

//вычисление степени

};

**Fraction.cpp**

#include "fraction.h"

//реализация метода для инициализации полей структуры

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

{

first = F; second = S;

}

//реализация метода для чтения значений полей структуры

void fraction::Read()

{

cout << "first:"; cin >> first; cout << "second:"; 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

int fraction::Power()

{

return pow(first, second);

}

int fraction::ipart()

{

return(first/second);

**Main.cpp**

#include "fraction.h"

fraction make\_fraction(int 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

}

void main()

{

setlocale(LC\_ALL, "");

fraction t;

t.Read();

if ((t.first > 0) && (t.second > 0))

{

cout <<"Целая часть F/S "<< t.ipart() << "\n";

}

else

{

cout << "Введите положительное целое число!" << endl;

}

//определение переменных А и В

fraction A;

fraction B;

A.Init(12, 3);//инициализация переменной

B.Read();//ввод полей переменных через консоль

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

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

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

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

cout << "\nСпособ через указатель\n";

//указатели

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

X->Read();//инициализация

X->Show();//вывод значений полей

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

//массивы

cout << "\nСпособ массивов без указателей\n";

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++)

{

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

cout << mas[i].Power() << endl;

}

//динамические масивы

cout << "\nСпособ массивов с указателями\n";

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++)

{

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

}

//вызов функции make\_fraction()

cout << "Способ через функцию" << endl;

int y; int z;

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

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

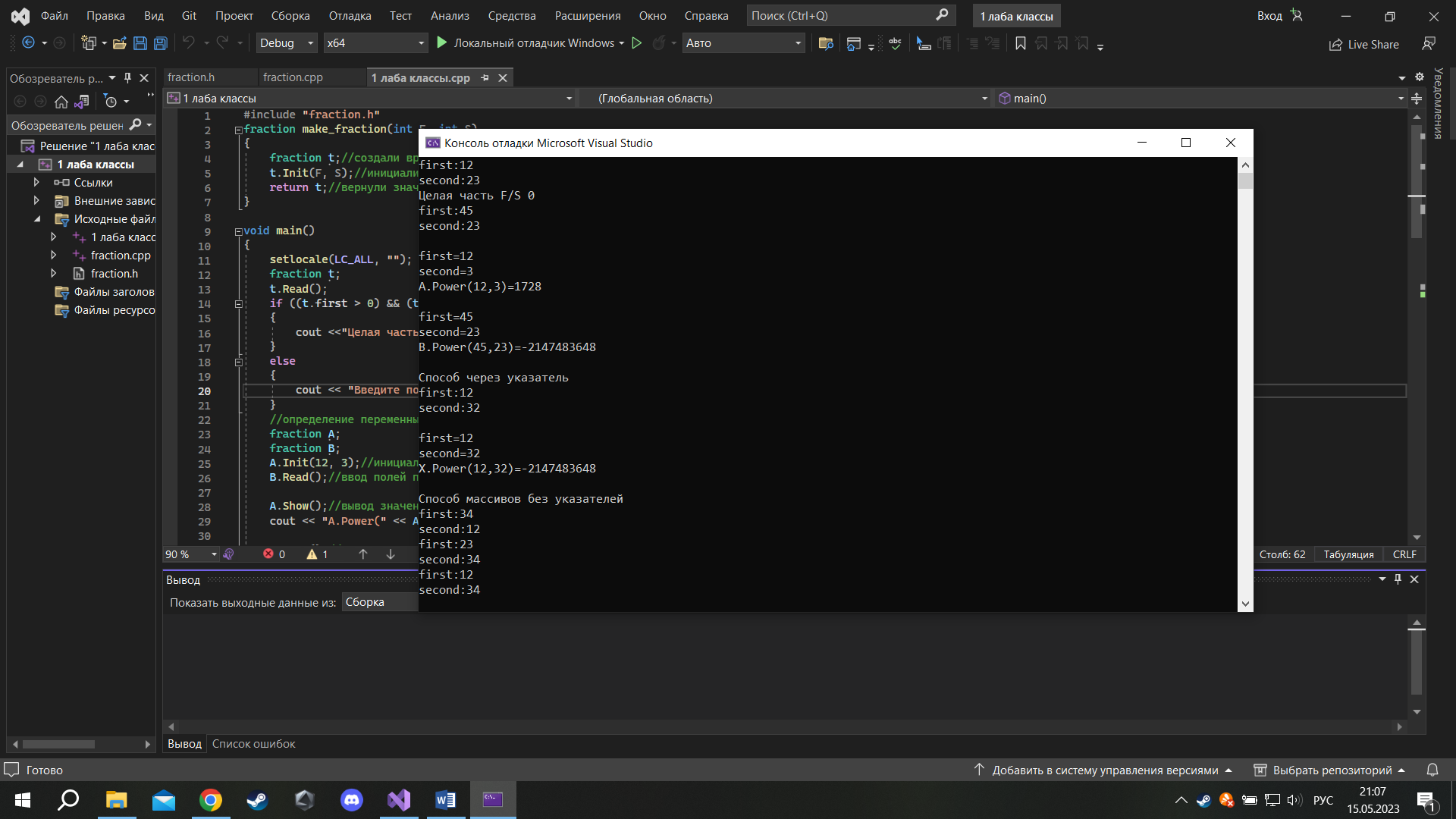
//переменная F формируется с помощью функции make\_fraction()

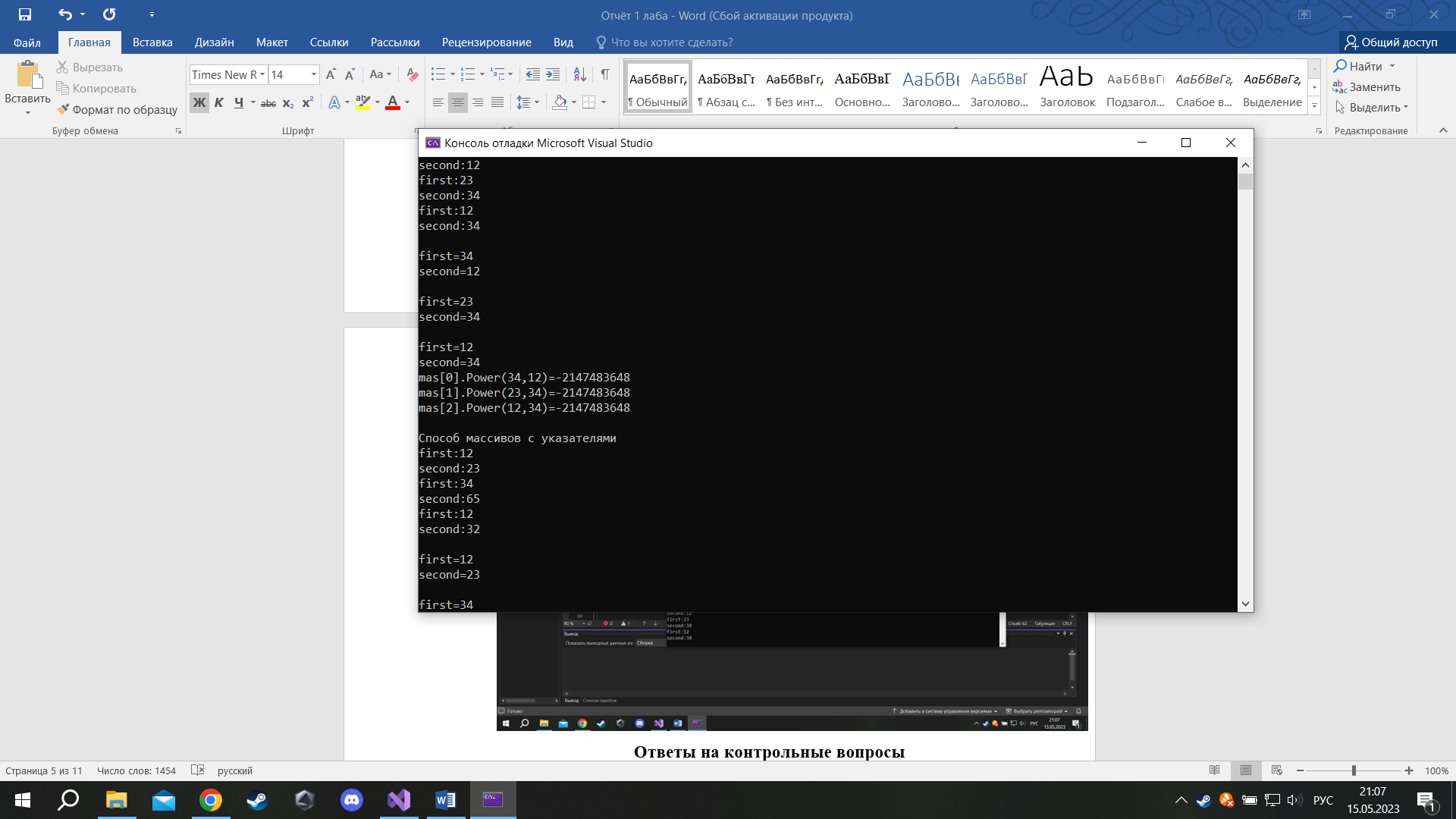
fraction F = make\_fraction(y, z);

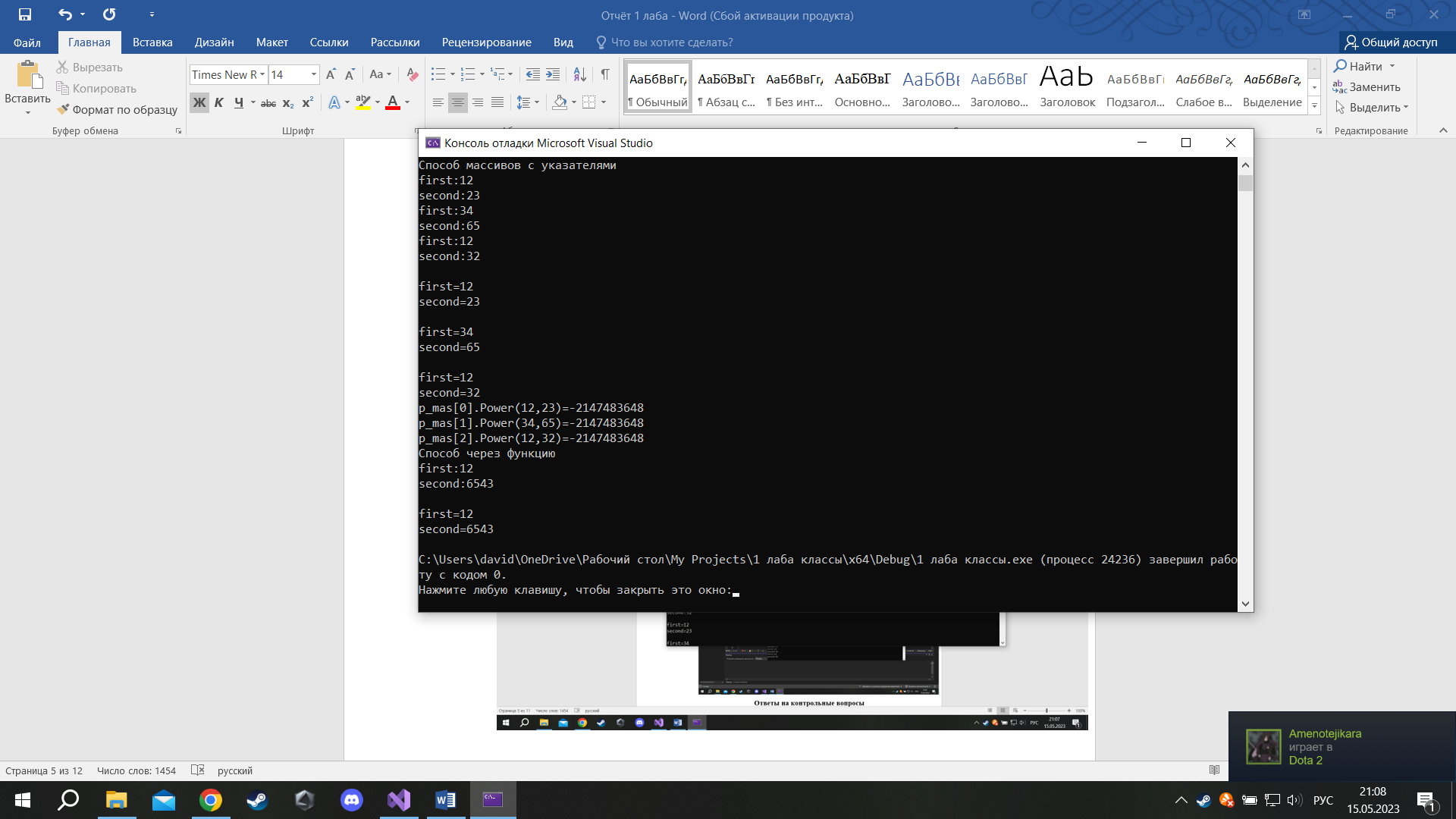
F.Show();

}

**Вывод результатов в консоль.**







**Ответы на контрольные вопросы**

1. Что такое класс?

Класс - абстрактный тип данных, который создает пользователь, состоящий из данных и функций. Позволяет представить часть кода в виде одного объекта. В объектно-ориентированном программировании программа представляется в виде множества объектов, что позволяет регулировать отношения между ними.

Пример класса Point:

class Point {

int x, y;

public:

int GetX() { return x; }

};

2. Что такое объект (экземпляр) класса?

Объект (экземпляр) - отдельный переменная класса, которая характеризуется данными его полей.

Пример объекта:

Point A;

3. Как называются поля класса?

Атрибуты, в примере из вопроса (1) это поля x, y

4. Как называются функции класса?

Методы, в примере из вопроса (1) это функция GetX()

5. Для чего используются спецификаторы доступа?

Спецификаторы используются для регулирования области видимости и времени жизни переменной. Действие любого спецификатора распространяется до следующего спецификатора или до конца класса.

6. Для чего используется спецификатор public?

Спецификатор public задаёт последующим переменным глобальную область видимости (доступны в коде вне класса) и время жизни до конца выполнения программы.

7. Для чего используется спецификатор private?

Спецификатор private задаёт последующим переменным локальную область видимости (доступны только в классе) и время жизни - во время компиляции кода описания класса.

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

Для типа данных class по умолчанию действует спецификатор private.

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

Для типа данных struct по умолчанию действует спецификатор public.

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

Для описания класса должен использоваться спецификатор public для того, чтобы его поля/методы были доступны для использования в других частях кода, и класс не был изолированной структурой данных.

11. Каким образом можно изменить значение атрибутов экземпляра класса?

Если атрибуты имеют public доступ, то изменить поля можно, обратившись по имени, по указателю или по ссылке.

class Point {

public:

int x, y;

};

int main() {

Point a; //доступ по имени

a.x = 20;

Point\* b = new Point; //доступ по указателю

b->x = 19;

Point& c = a; //доступ по ссылке

c.x = 18;

return 0;

}

Если атрибуты имеют private доступ, то изменить поля можно, обратившись к public методам-модификаторам.

class Point{

int x;

public:

void SetX(int n) { x= n; }

};

int main() {

Point a;

//a.x = 5; невозможно из-за private доступа поля

a.SetX(5);

return 0;

}

12. Каким образом можно получить значения атрибутов экземпляра класса?

Если атрибуты имеют public доступ, то получить доступ к полям можно, обратившись по имени, по указателю или по ссылке.

class Point {

public:

int x;

};

int main() {

Point a;

cout << a.x; //доступ по имени

Point\* b = new Point;

cout << b->x; //доступ по указателю

Point& c = a;

cout << c.x; //доступ по ссылке

return 0;

}

Если атрибуты имеют private доступ, то получить доступ к полям можно, обратившись к public методам-модификаторам.

class Point{

int x;

public:

int GetX() { return x; }

};

int main() {

Student a;

//cout << a.x; невозможно из-за private доступа поля

cout << a.GetX();

return 0;

}

13. Класс определен 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?

Так как у типа данных struct по умолчанию public доступ, то к полям можно обратиться непосредственно. Так как экземпляр класса задан через указатель, то к нему можно обратиться s->name, что равносильно записи (\*s).name.

14. Класс определен следующим образом

struct Student {

string name;

int group;

};

Объект класса определен следующим образом

Student s;

Как можно обратиться к полю name объекта s?

Так как у типа данных struct по умолчанию public доступ, то к полям можно обратиться непосредственно. Так как экземпляр класса задан по имени, то к нему можно обратиться 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?

Так как у типа данных class по умолчанию private доступ, то к полям нельзя обратиться непосредственно, и требуются специальные методы с public доступом, которые называются геттеры или селекторы.

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?

Так как у типа данных class по умолчанию private доступ, то к полям нельзя обратиться непосредственно, и требуются специальные методы с public доступом, которые называются геттеры или селекторы. Допустим, в public секции определен некоторый метод

void GetName() { return name; }

Тогда к полю можно обратиться следующим образом

s.GetName();

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?

Так как у данного класса указан public доступ, то к полям можно обратиться непосредственно. Так как экземпляр объявлен через указатель, то обратиться к нему можно s->name или (\*s).name.