Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ**

**о работе по информатике**

Семестр: 2

На тему: «ООП. Классы и объекты. Инкапсуляция»

Выполнил студент ИВТ-22-2б:

Шилов Валерий Михайлович

Проверил доцент 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:

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.

**Описание класса**

Поле first – положительное целое число, номинал купюры; номинал может принимать значения 1, 2, 5. 10, 50, 100, 500, 1000, 5000, поле second – положительное целое число, количество купюр данного достоинства. Реализовать метод summa() – вычисление денежной суммы.

**UML-диаграмма**

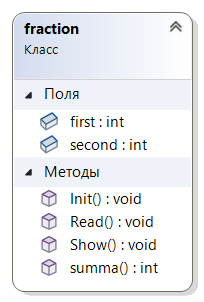


Рисунок 1 – UML-диаграмма

**Результат работы программы**

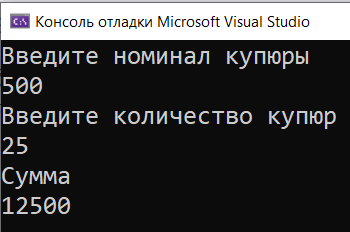


Рисунок 2 – результат работы программы

**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:

class fraction

{

public:

int first;

int second;

int summa();

void Read();

void Show();

void Init();

};

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 fisrt и second

4. Как называются функции класса?

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

5. Для чего используются спецификаторы доступа?

Спецификаторы доступа управляют видимостью элементов класса. Действие любого спецификатора распространяется до следующего спецификатора или до конца класса. За счёт спецификаторов доступа осуществляется инкапсуляция.

Спецификаторы бывают private, public, а также protected.

6. Для чего используется спецификатор public?

Открытый тип доступа. Поля и методы имеют глобальную область видимости и могут использоваться в любой части программы. Время жизни до конца испjлнения программы.

Пример:

public:

//К методам можно обращаться в любой части программы

int get\_first() { return first; }

double get\_second() { return second; }

7. Для чего используется спецификатор private?

Закрытый тип доступа. Локальная область видимости (доступ к полям и методам только внутри класса) и время жизни - во время компиляции кода описания класса. Доступ вне класса через методы типа public.

Пример:

private:

//Обращение к полям только с помощью public-методов

int first;

double second;

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

Если при описании класса используется слово class, то по умолчанию все методы и поля класса будут скрытыми – private.

Пример:

class Pair

{ //По умолчанию private

int first;

double second;

int get\_first() { return first; }

double get\_second() { return second; }

};

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

Если при описании класса используется слово struct, то все поля и методы по умолчанию будут общедоступными – public.

Пример:

struct Pair

{ //По умолчанию public

int first;

double second;

int get\_first() { return first; }

double get\_second() { return second; }

};

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

Интерфейс класса описывается после спецификатора public для того, чтобы его методы были доступны для использования в других частях кода, и класс не был изолированной структурой данных.

Пример:

class Pair

{

private:

int first;

double second;

public: //Интерфейс

int get\_first() { return first; }

double get\_second() { return second; }

};

11. Каким образом можно изменить значения атрибутов экземпляра класса?

Если атрибуты имеют доступ типа public, то изменить поля можно, обратившись напрямую по имени, с помощью указателя или по ссылке.

Пример:

class Pair

{

public:

int first;

double second;

};

int main()

{

Pair p;

p.first = 20; //Доступ по имени

Pair\* p1 = new Pair;

p1->second = 18.7; //Доступ с помощью указателя

Pair& p2 = p;

p2.first = 25; //Доступ по ссылке

return 0;

}

Изменить информацию о содержимом полей, описанных после спецификатора private можно только с помощью специальных public-методов которые называются модификаторами (сеттерами).

Пример:

class Pair

{

private:

int first;

double second;

public:

//Модификатора (сеттеры)

void set\_first(int f) { first = f; }

void set\_second(double s) { second = s; }

};

int main()

{

Pair p;

p.first = 20; //Доступ по имени невозможен

p.set\_first(20); //Возможен доступ с помощью модификатора

return 0;

}

12. Каким образом можно получить значения атрибутов экземпляра класса?

Если атрибуты имеют доступ типа public, то получить значение поля можно, обратившись напрямую по имени, с помощью указателя или по ссылке.

Пример:

class Pair

{

public:

int first;

double second;

};

int main()

{

Pair p;

cout << p.first;//Доступ по имени

Pair\* p1 = new Pair;

cout << p1->second; //Доступ с помощью указателя

Pair& p2 = p;

cout << p2.first; //Доступ по ссылке

return 0;

}

Получить информацию о содержимом полей, описанных после спецификатора private можно только с помощью специальных методов, которые называются селекторами (геттерами).

Пример:

class Pair

{

private:

int first;

double second;

public:

//Селекторы (геттеры)

int set\_first() { return first; }

double set\_second() { return second; }

};

int main()

{

Pair\* p1 = new Pair;

cout << p1->second; //Доступ с помощью указателя невозможен

cout << p1->get\_first();

return 0;

}

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?

Так как у типа данных struct по умолчанию public доступ, то к полям можно обратиться напрямую. Так как экземпляр класса задан через указатель, то к нему можно обратиться следующим образом

int main()

{

Student\* s = new Student;

s->name = "Ivan";

}

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?

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

int main()

{

Student s;

s.name = "Ivan";

}

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 напрямую к полям обратиться нельзя. Требуются специальные методы типа public, но они здесь не описаны и область public вообще отсутствует.

int main()

{

Student\* s = new Student;

s->name = "Ivan"; //Ошибка при обращении с помощью указателя

}

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 обратиться нельзя. Требуются специальные методы типа public (например, модификаторы), которые можно описать в области public:

class Student

{

string name;

int group;

public:

void set\_name(string s) { name = s; }

};

int main()

{

Student s;

s.set\_name("Ivan"); //Обращение к методы по имени

}

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 типа данного класса доступ к полям public, то к ним можно обратиться напрямую. Так как экземпляр класса задан через указатель, то к нему можно обратиться следующим образом:

int main()

{

Student\* s = new Student;

char arr1[] = "Ivan";

s->name = arr1;

}