Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное 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.»

Семестр 2

*Работу выполнил: Учащийся группы ИВТ - 22-2б: Цыбуцынин Фёдор Александрович*

*Работу проверил: доцент кафедры ИТАС:*

*Полякова Ольга Андреевна*

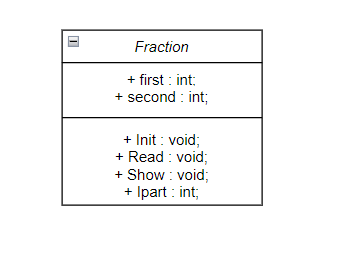
Г. Пермь – 2023

**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
3. метод инициализации Init (метод должен контролировать значения аргументов на корректность);
4. ввод с клавиатуры Read;
5. вывод на экран Show.
6. 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 №1**

Поле first – положительное целое число, числитель, поле second – положительное целое число, знаменатель. Реализовать метод ipart() – выделение целой части дроби first/second, метод должен проверять неравенство знаменателя нулю

**UML – диаграмма**

**Рис. 1 – UML-диаграмма класса fraction**

**Код программы**

Fraction.h:

struct fraction

{

int first;

int second;

void Init(int, int);

int ipart();

};

Fraction.cpp:

#include <iostream>

#include "fratction.h"

using namespace std;

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

{

first = F;

second = S;

}

void fraction::Read()

{

setlocale(0, "Rus");

cout << "Числитель: ";

cout << "Знаменатель: ";

cin >> second;

}

void fraction::Show()

{

cout << "Числитель: " << first << endl;

cout << "Знаменатель: " << second << endl;

}

int fraction::ipart()

{

return first / second;

}

Класс лаба 1:

#include <iostream>

#include "fratction.h"

using namespace std;

int main()

{

setlocale(0, "rus");

fraction A;

int first, second;

cout << "Введтите числитель: ";

cin >> first;

cout << "Введите знаменатель: ";

cin >> second;

if (second == 0)

{

cout << "Че за херня, чел";

}

else

{

A.Init(first, second);

cout << first << '/' << second << '=' << A.ipart();

}

return 0;

}

**Контрольные вопросы**

1. Что такое класс?

Класс - абстрактный тип данных, который создаёт 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 Fruit

{

string color;

string name;

public:

string Output()

{

return color;

}

};

1. Что такое объект (экземпляр) класса?

Объект - это переменная класса, которая характеризуется данными его полей.

Пример объекта:

Fruit Apple;

1. Как называются поля класса?

Атрибуты, такие как string color и string name из примера в вопросе 1.

1. Как называются функции класса?

Функции класса называются методами класса. В примере кода в вопросе 1 это string Output().

1. Для чего используются спецификаторы класса?

Спецификаторы класса определяют область видимости и время жизни данных, хранящихся в нём. Действие любого спецификатора распространяется до следующего спецификатора или до конца класса.

1. Для чего используется спецификатор public?

Данный спецификатор задаёт полям класса глобальную область видимости и время жизни до конца выполнения программы, то есть этот спецификатор делает поля класса доступными в коде вне класса.

1. Для чего используется спецификатор private?

Данный спецификатор в отличие от предыдущего задаёт полям класса локальную область видимости, то есть они видны только внутри самого класса, и время жизни во время компиляции кода класса.

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

Для данных такого типа по умолчанию будет устанавливаться спецификатор private.

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

Для данных такого типа по умолчанию будет устанавливаться спецификатор public.

1. 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.

1. Каким образом можно изменить значение атрибутов экземпляра класса?

- Если атрибуты имеют доступ public, то изменить поля можно по указателю, по ссылке и по имени:

void main()

{

pressure first;

first.value = 100; // По имени

pressure\* second = new pressure;

second->value = 20; // По указателю

pressure& third = first;

third.value = 1; // По ссылке

}

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

class grade

{

int value;

public:

void ForChanges(int n)

{

value = n;

}

};

void main()

{

grade Student;

int n = 5;

Student.value = n; // выдаст ошибку

Student.ForChanges(n);

}

1. Каким образом можно получить значения атрибутов экземпляра класса?

- Если атрибуты имеют доступ public, то изменить поля можно по указателю, по ссылке и по имени:

class pressure

{

public:

int value;

};

void main()

{

pressure first;

first.value = 5;

cout << first.value; // По имени

pressure\* second = new pressure;

cout << second->value; // По указателю

pressure& third = first;

cout << third.value; // По ссылке

}

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

class grade

{

int value = 5;

public:

int ForChanges()

{

return value;

}

};

void main()

{

grade Student;

cout << Student.value; // выдаст ошибку

cout << Student.ForChanges();

}

1. Класс определён следующим образом

struct Student {

string name;

int group;

};

Объект класса определён следующим образом

Student \*s = new Student;

Как можно обратиться к полю name объекта s?

Так как у типа данных struct по умолчанию public доступ, а экземпляр класса задан через указатель, то к нему можно обратиться через указатель s->name.

1. Класс определён следующим образом

struct Student {

string name;

int group;

};

Объект класса определён следующим образом

Student s;

Как можно обратиться к полю name объекта s?

Поскольку у типа данных struct по умолчанию стоит доступ public, а объект класса задан по имени, то к его полю name можно обратиться по имени s.name.

1. Класс описан следующим образом

class Student {

string name;

int group;

};

Объект класса определён следующим образом

Student \*s = new Student;

Как можно обратиться к полю name объекта s?

Поскольку у типа данных class по умолчанию стоит спецификатор private, а класс не содержит в себе специальных методов с доступом public, к полям этого класса обратиться не получится.

1. Класс описан следующим образом

class Student {

string name;

int group;

public:

…

};

Объект класса определён следующим образом

Student s;

Как можно обратиться к полю name объекта s?

Поскольку у типа данных class по умолчанию стоит спецификатор private, то к полям нельзя обратиться непосредственно, для требуются специальные методы с public доступом, которые называются геттеры или сеттеры. Если в части класса с доступом public объявлен метод void GetName() { return name; }, то к полю можно обратиться как s.GetName();

1. Класс описан следующим образом

class Student {

public:

char\* name;

int group;

};

Объект класса определён следующим образом

Student \*s = new Student;

Как можно обратиться к полю name объекта s?

Так как у типа данных struct по умолчанию public доступ, а экземпляр класса задан через указатель, то к нему можно обратиться через указатель s->name.