Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе № 15.1**

Дисциплина: «Информатика»

Тема: ООП. Классы и объекты. Инкапсуляция

Вариант 12

Выполнила:

Студентка группы РИС-22-1б

Черкасова А.А.

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

**Пермь 2023**

**Содержание**

**Введение**

**Цель:**

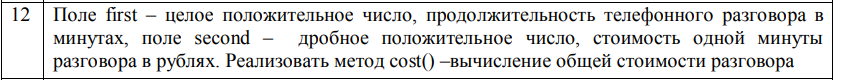
* Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
* Использование классов и объектов в ОО программе.

Для достижения поставленной цели, необходимо решить следующие **задачи**:

* Провести анализ задачи
* Реализовать задачу на языке С++
* Составить блок-схему

**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:
   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.

****

# **Анализ задачи**

1. Для решения задачи необходимо:
   1. Реализовать определение класса Tariff.
   2. Реализовать определение методов класса Tariff: Init, Read, Show, Cost.
   3. Разработать функцию make\_Tariff для выделения памяти под новый объект и его инициализации.
   4. Реализовать применение этих функций в главной функции.
2. В ходе работы были использованы типы данных:
   1. Для метода Init класса Tariff используются следующие аргументы:
   2. Тип unsigned int: продолжительность разговора в минутах.
   3. Тип float: стоимость минуты разговора в рублях.

Сам метод имеет тип void, поскольку при работе метода не нужно возвращать значение.

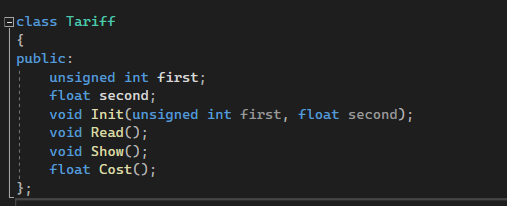
* 1. Для методов Read, Show, Cost класса Tariff не используются аргументы:

Сам метод имеет тип void, поскольку при работе метода не нужно возвращать значение.

* 1. Для функции make\_Tariff используются следующие аргументы:
  2. Тип unsigned int: продолжительность разговора в минутах.
  3. Тип float: стоимость минуты разговора в рублях.

Сама функция имеет тип void, поскольку при работе функции не нужно возвращать значение.

1. Для решения задачи данные были представлены в следующем виде:
   1. Для работы с данными используются поля и методы класса.



1. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Ввод данных реализован с помощью оператора cin, используемого при реализации метода Read и в главной функции.
   2. Вывод данных реализован с помощью оператора cout, используемого в главной функции и при реализации метода Show.
2. Поставленные задачи решены следующими действиями:
   1. Определение класса Tariff было реализовано в заголовочном файле Tariff.h, определение методов класса – в файле Tariff.cpp. Основной блок программы и функция make\_Tariff описаны в главном файле main.cpp.
   2. При работе с объектами класса Tariff обращение к методам аналогичен обращению к полям структуры.

# **UML – диаграмма**



# **Приложение А**

**Листинг программы**

*A) Заголовочный файл Tariff.h*

#pragma once

class Tariff

{

public:

unsigned int first; // Продолжительность разговора в минутах

float second; // Стоимость минуты в рублях

void Init(unsigned int first, float second); // Метод инициализации полей объекта

void Read(); // Метод заполнения полей объекта

void Show(); // Метод вывода полей объекта

float Cost(); // Стоимость разговора

};

*B) Файл с описанием методов класса Tariff.cpp*

#include "Tariff.h"

#include <iostream>

using namespace std;

void Tariff::Init(unsigned int min, float rub) // Метод инициализации полей объекта - "конструктор"

{

first = min;

second = rub;

}

void Tariff::Read() // Get

{

cout << "Продолжительность разговора (в минутах): ";

cin >> first;

while (first < 0)

{

cout << "Неверно введена продолжительность разговора! Попробуйте ещё раз: ";

cin >> first;

}

cout << "Стоимость минуты разговора: ";

cin >> second;

while (second < 0)

{

cout << "Неверно введена стоимость разговора! Попррбуйте ещё раз: ";

cin >> second;

}

}

void Tariff::Show() // Метод вывода полей объекта

{

cout << "Продолжительность разговора: " << first << " мин" << endl;

cout << "Стоимость одной минуты разговора: " << second << " р" << endl;

}

float Tariff::Cost() // Стоимость разговора

{

return first \* second;

}

*C) Файл с главной программой main.cpp*

#include <iostream>

#include "Tariff.h"

using namespace std;

Tariff make\_Tariff(unsigned int min, float rub) // Функция инициализации объекта

{

Tariff conv; // Выделение памяти под объект

conv.Init(min, rub); // Вызов метода инициализации полей объекта

return conv;

}

int main()

{

system("chcp 1251>NULL");

Tariff One;

Tariff Two;

One.Init(18, 1.75); // Вызов метода инициализации полей объекта 1

cout << "Инициализация объекта Two" << endl;

Two.Read(); // Вызов метода заполнения полей объекта 2

cout << "\t\tOne" << endl;

One.Show(); // Вызов метода вывода полей объекта 1

cout << "Стоимость разговора объекта One: " << One.Cost() << " р" << endl; // Вызов метода нахождения зарплаты за месяц объекта 1

cout << "\t\tTwo" << endl;

Two.Show(); // Вызов метода вывода полей объекта 2

cout << "Стоимость разговора объекта Two: " << Two.Cost() << " р" << endl; // Вызов метода нахождения зарплаты за месяц объекта 2

Tariff\* ptr\_Three = new Tariff; // Выделение памяти под объект 3 через указатель

cout << "\t\tThree" << endl;

ptr\_Three->Init(5, 2.3); // Вызов метода инициализации полей объекта 3

ptr\_Three->Show(); // Вызов метода вывода полей объекта 3

cout << "Стоимость разговора объекта Three: " << ptr\_Three->Cost() << " р" << endl << endl;

Tariff mas[3]; // Выделение памяти под множество объектов

cout << "mas\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Заполнение массива" << endl;

for (int i = 0; i < 3; i++)

{

cout << "Объект #" << i + 1 << endl;

mas[i].Read(); // Вызов метода заполнения полей каждого объекта из множества

}

cout << "\nВывод массива" << endl;

for (int i = 0; i < 3; i++)

{

cout << "Объект #" << i + 1 << endl;

mas[i].Show(); // Вызов метода вывода полей каждого объекта из множества

}

for (int i = 0; i < 3; i++)

{ // Вызов метода нахождения зарплаты за месяц объекта каждого объекта из множества

cout << "Стоимость разговора объекта # " << i + 1 << ": " << mas[i].Cost() << " р" << endl;

}

Tariff\* arr = new Tariff[3];

cout << "\narr\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl << endl;; // Выделение памяти под указатель на первый объект множества

cout << "Заполнение массива" << endl;

for (int i = 0; i < 3; i++)

{

cout << "Объект #" << i + 1 << endl;

arr[i].Read(); // Вызов метода заполнения полей каждого объекта из множества

}

cout << "\nВывод массива" << endl;

for (int i = 0; i < 3; i++)

{

cout << "Объект #" << i + 1 << endl;

arr[i].Show(); // Вызов метода вывода полей каждого объекта из множества

}

for (int i = 0; i < 3; i++)

{ // Вызов метода нахождения зарплаты за месяц объекта каждого объекта из множества

cout << "Стоимость разговора объекта #" << i + 1 << ": " << arr[i].Cost() << " р" << endl;

}

unsigned int minutes;

float rubles;

cout << "\nИнициализация объекта Four" << endl;

cout << "Введите продолжительность разговора в минутах: ";

cin >> minutes;

while (minutes < 0)

{

cout << "Неверно введена продолжительность разговора! Попробуйте ещё раз: ";

cin >> minutes;

}

cout << "Введите стоимость минуты разговора: ";

cin >> rubles;

Tariff Four = make\_Tariff(minutes, rubles); // Вызов Функция создания объекта D4 и заполнение его полей

cout << "\t\tFour" << endl;

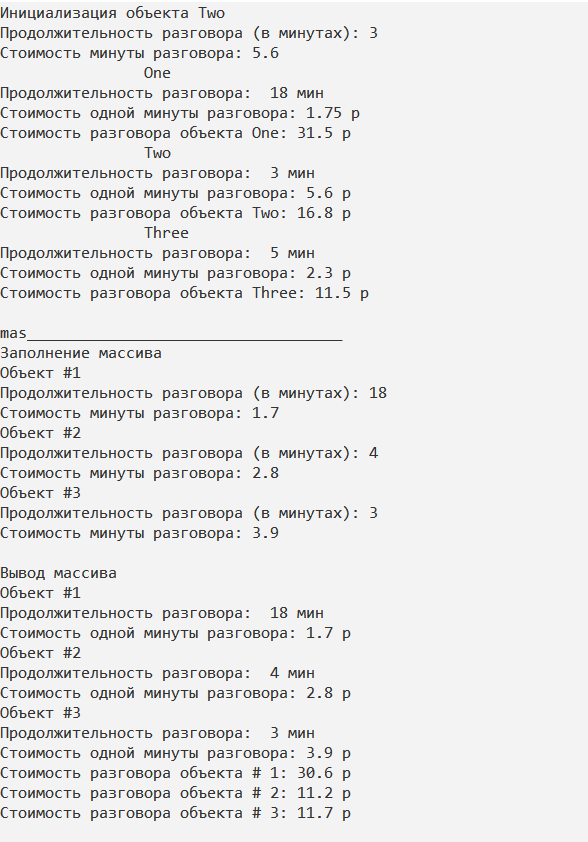
Four.Show(); // Вызов метода вывода полей объекта 4

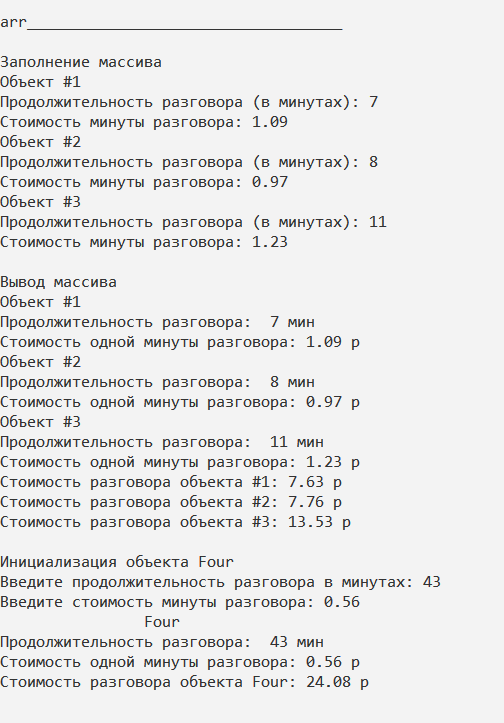
cout << "Стоимость разговора объекта Four: " << Four.Cost() << " р" << endl << endl;

}

# **Приложение Б**

**Результаты выполнения программы**





**Ответы на вопросы**

1. Что такое класс?

Класс – абстрактный тип данных, определяемый пользователем, представляет собой модель реального объекта в виде дынных и функций для работы с ними.

1. Что такое объект (экземпляр) класса?

Любая переменная ранее созданного класса называется объектом или экземпляром класса.

1. Как называются поля класса?

Поля в классе называются атрибутами.

1. Как называются функции класса?

Функции в классе называются методами.

1. Для чего используются спецификаторы доступа?

Спецификаторы доступа используются для управления видимости элементов класса (атрибутов и методов).

1. Для чего используется спецификатор public?

Спецификатор public используется для возможности взаимодействия с элементами с данным спецификатором и видимости этих элементов вне класса (обычно это методы).

1. Для чего используется спецификатор private?

Спецификатор private используется для видимости элементов только внутри класса (обычно это атрибуты класса).

1. Если описание класса начинается со спецификатора class, то какой спецификатор доступа будет использован по умолчанию? (private)

Для класса, начинающегося со спецификатора class, по умолчанию используется спецификатор доступа private.

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

Для класса, начинающегося со спецификатора struct, по умолчанию используется спецификатор доступа public.

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

При описании интерфейса используется спецификатор public, поскольку данный спецификатор обеспечивает возможность использования описанных в классе методов вне данного класса.

1. Каким образом можно изменить значения атрибутов экземпляра класса? (set)

Для изменения атрибутов экземпляра класса обычно используют определенными в классе специальными методами, называемыми модификаторами.

1. Каким образом можно получить значения атрибутов экземпляра класса?(get)

Для получения атрибутов экземпляра класса обычно используют определенными в классе специальными методами, называемыми селекторами.

1. 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?

Обращение к полю name объекта s аналогично обращению к полям указателя структуры и выглядит следующим образом:

s->name;

1. 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?

Обращение к полю name объекта s аналогично обращению к полям переменной структуры и выглядит следующим образом:

s.name;

1. 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?

Обращение к полю name объекта s в данном случае невозможно, т. к. все элементы класса имеют спецификатор доступа private.

1. 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?

Обращение к полю name объекта s возможно через селекторы и модификаторы класса.

1. 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?

Обращение к полю name объекта s возможно через селекторы, модификаторы и аналогично обращению к полям указателя структуры.