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**

**Тема:** Лабораторная работа № 1 по ООП

Семестр: 2

Выполнил студент ИВТ-22-2б:

Солодов Александр Андреевич

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

Пермь 2023

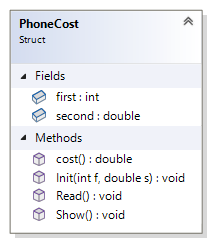
**Постановка задачи**

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:
   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.

Вариант 12.

Поле first – целое положительное число, продолжительность телефонного разговора в минутах, поле second – дробное положительное число, стоимость одной минуты разговора в рублях. Реализовать метод cost() –вычисление общей стоимости разговора

**UML-диаграмма**



**Программное решение**

PhoneCost.h

#pragma once

#include <iostream>

using namespace std;

struct PhoneCost {

int first; //продолжительность телефонного разговора в минутах

double second; //стоимость одной минуты разговора в рублях

void Init(int f, double s);

void Read();

void Show();

double cost();

};

PhoneCost.cpp

#include "PhoneCost.h"

//функция для заполнения данных

void PhoneCost::Init(int f, double s) {

if (f > 0 && s > 0) {

first = f;

second = s;

}

else {

cout << "Количество минут и стоимость должны быть больше нуля!" << endl;

}

}

//функция для считывания данных с клавиатуры

void PhoneCost::Read(){

int f = 0;

double s = 0.0;

cout << "\nВведите количество миинут: ";

cin >> f;

while (f <= 0) {

cout << "Укажите корректное количество минут, оно должно быть больше нуля: " ;

cin >> f;

}

first = f;

cout << "Введите стоимость одной минуты в рублях: ";

cin >> s;

while (s <= 0.0) {

cout << "Укажите корректную стоимость, она должна быть больше нуля: ";

cin >> s;

}

second = s;

}

//функция для отображения данных

void PhoneCost::Show(){

cout << "\nКоличество минут (first) = " << first << endl;

cout << "Стоимость минуты в рублях (second) = " << second << endl;

}

//функция для вычисления стоимости разговора

double PhoneCost::cost() {

return first \* second;

}

Main.cpp

/\* Вариант 12

Поле first – целое положительное число, продолжительность телефонного разговора в минутах,

поле second – дробное положительное число, стоимость одной минуты разговора в рублях.

Реализовать метод cost() – вычисление общей стоимости разговора

\*/

#include "PhoneCost.h"

PhoneCost makePhoneCost(int x, double y) {

PhoneCost temp;

temp.Init(x, y);

return temp;

}

int main() {

setlocale(LC\_ALL, "RU");

PhoneCost example = makePhoneCost(12, 2.5);

example.Show();

example.Read();

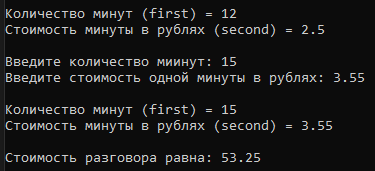
example.Show();

cout << "\nСтоимость разговора равна: " << example.cost() << endl;

return 0;

}

**Результат работы программы**



**Контрольные вопросы**

*1.Что такое класс?*

Класс – абстрактным тип данных, определяемый пользователем. Класс представляет собой модель реального объекта в виде данных и функций для работы с ними. Позволяет представить часть кода в виде одного объекта.

Пример:

class Pair {

private:

int left;

double right;

public:

int get\_left() { return left; }

double get\_right() { return right; }

};

*2. Что такое объект (экземпляр) класса?*

Объектом называется отдельный элемент класса, который характеризуется данными его полей. Второе название объекта класса - экземпляр класса. Пример:

int main() {

Pair h;

return 0;

}

*3. Как называются поля класса?*

Поля класса или атрибуты, составляют структуру класса.

Элементы int left; right second; из вопроса 1.

*4. Как называются функции класса?*

Методы класса – функции для работы с объектами класса.

Элементы из вопроса 1.

int get\_left() { return left; }

double get\_right() { return right; }

*5. Для чего используются спецификаторы доступа?*

Управление видимостью элементов класса. Спецификаторы бывают private, public из вопроса 1, 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 исполнения программы.

Пример:

public:

int get\_left() { return left; }

double get\_right() { return right; }

*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:

int left;

double right;

*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 {

int left;

double right;

int get\_left() { return left; }

double get\_right() { return right; }

};

*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 {

int left;

double right;

int get\_left() { return left; }

double get\_right() { return right; }

};

*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?*

Изменить значения атрибутов можно через сеттер или обратившись напрямую.

*12. Каким образом можно получить значения атрибутов экземпляра класса?*

Получить значения атрибутов можно через геттер или обратившись напрямую.

*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;

}