Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

**«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПЕРМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра “Информационные технологии и автоматизированные системы”

**ОТЧЕТ**

**по лабораторной работе №18.1**

Дисциплина: “Информатика”

Тема: Методы внешней сортировки: метод прямого слияния, сбалансированного слияния, метод многофазной сортировки

Вариант №12

Выполнил:

Студент группы РИС-20-2б

Морозова.Е.М

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Цель работы**

Использование классов и объектов в ОО программе

**Постановка задачи**

Реализовать определение нового класса. Для демонстрации работы с объектами написать главную функцию. Продемонстрировать разные способы создания объектов и массивов объектов.

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 :

метод инициализации Init (метод должен контролировать значения аргументов на корректность);

ввод с клавиатуры Read;

вывод на экран Show.

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 в рублях. Реализовать метод cost - вычисление общей стоимости разговора.

**Анализ задачи**

1. Определение класса:

struct fraction {

double first;

int second;

void Init(double, int); // метод для инициализации полей

void Read(); // метод для чтения значений полей

void Show(); // метод для вывода значений полей

double Summa(); // вычисление степени

};

1. Определение компонентных функций:

#include <iostream>

#include "fraction.h"

using namespace std;

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

while (F < 0 || S < 0) {

cin >> F;

cin >> S;

}

first = F;

second = S;

}

void fraction::Read() {

cout << "\nfirst? ";

cin >> first;

while (first < 0) {

cin >> first;

}

cout << "\nsecond? ";

cin >> second;

while (second < 0) {

cin >> first;

}

}

void fraction::Show() {

cout << "\nfirst = " << first;

cout << "\nsecond = " << second;

cout << endl;

}

double fraction::Summa() {

return first \* second;

}

1. Определение функции make:

fraction make\_fraction(double F, int S) {

fraction t;

t.Init(F, S);

return t;

}

1. Функция main():

**int main()**

{

fraction A;

fraction B;

A.Init(3.0, 2);

B.Read();

A.Show();

B.Show();

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

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

fraction\* X = new fraction;

X->Init(2.0, 5);

X->Show();

X->Summa();

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

fraction mas[3];

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

mas[i].Read();

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

mas[i].Show();

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

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

mas[i].Summa();

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

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

}

fraction\* p\_mas = new fraction[3];

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

p\_mas[i].Read();

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

p\_mas[i].Show();

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

p\_mas[i].Summa();

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

cout << p\_mas[i].Summa() << endl;

}

double y;

int z;

cout << "first? ";

cin >> y;

cout << "second? ";

cin >> z;

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

F.Show();

return 0;

**}**

**Ответы на вопросы:**

1. Класс – это конструкция, объединяющая данные и функции, которые в основном обрабатывают эти данные.
2. Объект (экземпляр класса) – это переменная класса.
3. Поля класса называются атрибутами класса.
4. Функции класса называются методами.
5. Спецификаторы доступа используются для указания возможности доступа к данным (это применяется с целью сокрытия данных, для того, чтобы уберечь данные от несанкционированного доступа или случайного изменения).
6. Спецификатор public используется для открытия доступа к полям и методам экземпляра данного класса.
7. Спецификатор private используется для сокрытия полей и методов экземпляра класса.
8. Для классов умолчанию используется спецификатор доступа private.
9. Для структур по умолчанию используется спецификатор доступа public.
10. Для описания интерфейса класса должен использоваться спецификатор public, чтобы пользователь имел возможность взаимодействовать с методами класса.
11. Изменить значения атрибутов класса можно с помощью методов класса, называемых модификаторами.
12. Получить значения атрибутов класса можно с помощью методов класса, называемых селекторами.
13. s->name;
14. s.name;
15. С помощью функции-модификатора, если она имеется.
16. С помощью функции-модификатора, если она имеется.

17.s->name;

**Скриншоты**

