Министерство образования и науки РФ

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра Информационные технологии и автоматизированные системы

Лабораторная работа № 1

Тема: «Классы и объекты. Инкапсуляция»

Выполнил: студент группы РИС-22-2б

Мизёв В.А. ф

Проверил: доцент кафедры ИТАС

Полякова О.А.

г. Пермь – 2024

*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.

*Задание вариант 1:*

Поле first – положительное целое число, числитель, поле second – 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.

*Анализ задачи:*

Основываясь на задании необходимо реализовать один класс “Fraction”, в котором будет использоваться в качестве поля класса структура с двумя полями (first, second), данная структура будет иметь следующие название “Vector” с полями x, y соответственно, а также реализовать методы с инициализацией, чтением и выводом данных и метод из задания “ipart”, который выделяет целую часть дроби.

*Код программы:*

Прикреплён в приложении 1.

*UML-диаграмма класса:*

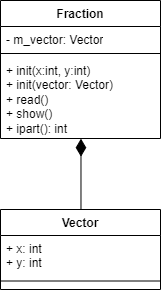


Рисунок 1 - UML-диаграмма класса Fraction

*Скриншот работы программы:*

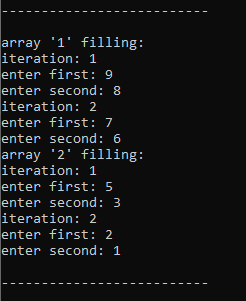


Рисунок 2 - Заполнение массивов

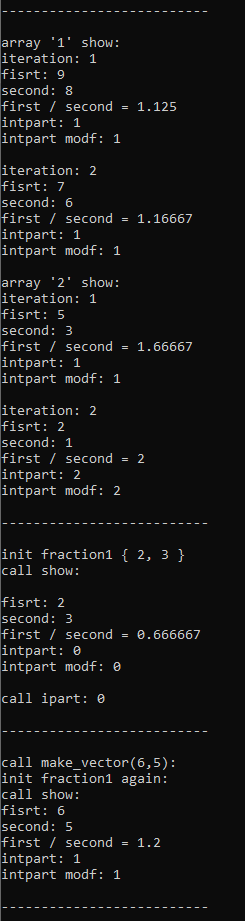


Рисунок 3 - Результат работы

*Контрольные вопросы:*

1. Что такое класс?

Класс – это тип или структура данных, определённых пользователем.

1. Что такое объект (экземпляр) класса?

Конкретный объект(экземпляр) класса (к примеру класс как шаблон описывающий фрукт, объекты банан, киви)

1. Как называются поля класса?

Данные класса.

1. Как называются функции класса?

Методы класса.

1. Для чего используется спецификаторы доступа?

Для установки доступности методов и полей классов, облегчение инкапсуляции компонентов.

1. Для чего используется спецификатор public?

Для общего доступа к членам класса.

К примеру, поля класса, объявленные с данными спецификатором доступа, могут использоваться через объект класса (или методы)

class A { int x; } A a; a.x = 5;

1. Для чего используется спецификатор private?

Для инкапсулирования данных. Когда необходимо скрыть некую реализацию внутри класса.

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

private

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

public

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

protected

1. Каким образом можно изменить значения атрибутов экземпляра класса?

Через прямое обращение A a; a.x = 5 или же непосредственно через метод класса a.setX(5);

1. Каким образом можно получить значения атрибутов экземпляра класса?

Через прямое обращение a.x или же через реализованный метод класса a.getX();

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?

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?

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?

В данном случаи никак, если не реализованы соответствующие методы в public.

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?

В данном случаи никак, если не реализованы соответствующие методы в public.

1. Класс описан следующим образом

class Student {

public:

char\* name;

int group;

public:

……………

};

Объект класса определен следующим образом

Student\* s=new Student;

Как можно обратиться к полю name объекта s?

s->name, при этом заранее проверить на нулевой указатель.

**ПРИЛОЖЕНИЕ 1**

lab1.cpp

//-------------------------------------

#include <iostream>

#include "lab1\_fraction.hpp"

//-------------------------------------

#define IDENT\_PRINT printf("\n--------------------------\n\n")

//-------------------------------------

Fraction::Vector make\_vector(int x, int y) {

return { x, y };

}

//-------------------------------------

int main() {

//create objects

Fraction fraction1;

Fraction\* fraction2 = new Fraction;

delete fraction2;

const int size = 2;

Fraction fractions[size];

Fraction\* ptrFractions = new Fraction[size];

Fraction\* ptr[] = { fractions, ptrFractions };

IDENT\_PRINT;

//filling array

for(size\_t i = 0; i < size\*2; ++i) {

if(i % size == 0)

std::cout << "array '" << (int)(i/(size)) + 1 << "' filling:\n";

std::cout << "iteration: " << (i%size) + 1 << std::endl;

ptr[(int)(i/(size))][i%size].read();

}

IDENT\_PRINT;

//show array

for(size\_t i = 0; i < size\*2; ++i) {

if(i % size == 0)

std::cout << "array '" << (int)(i/(size)) + 1 << "' show:\n";

std::cout << "iteration: " << (i%size) + 1 << std::endl;

ptr[(int)(i/(size))][i%size].show();

if(i != size\*2 - 1)

std::cout << std::endl;

}

IDENT\_PRINT;

//call init, read, show for first instance

std::cout << "init fraction1 { 2, 3 }\n";

fraction1.init(2, 3);

std::cout << "call show:\n\n";

fraction1.show();

std::cout << std::endl;

std::cout << "call ipart: " << fraction1.ipart() << std::endl;

IDENT\_PRINT;

std::cout << "call make\_vector(6,5):\n";

std::cout << "init fraction1 again:\n";

fraction1.init(make\_vector(6, 5));

std::cout << "call show:\n";

fraction1.show();

IDENT\_PRINT;

delete []ptrFractions;

return 0;

}

//-------------------------------------

lab1\_fraction.hpp

//-------------------------------------

#ifndef LAB1\_FRACTION\_H\_INCLUDED

#define LAB1\_FRACTION\_H\_INCLUDED

//-------------------------------------

#include <iostream>

//-------------------------------------

class Fraction {

public:

struct Vector {

int x,

y;

};

public:

Fraction();

~Fraction();

void init(int x, int y);

void init(Vector vector);

void read();

void show();

int ipart();

private:

Vector m\_vector;

};

//-------------------------------------

#endif // LAB1\_FRACTION\_H\_INCLUDED

//-------------------------------------

lab1\_fraction.cpp

//-------------------------------------

#include "lab1\_fraction.hpp"

#include <cassert>

#include <cmath>

//-------------------------------------

Fraction::Fraction() {

m\_vector = { 0, 0 };

}

//-------------------------------------

Fraction::~Fraction() {

//

}

//-------------------------------------

void Fraction::init(int x, int y) {

this->init({x, y});

}

//-------------------------------------

void Fraction::init(Vector vector) {

m\_vector = vector;

assert("The denominator cannot be zero!" && m\_vector.y != 0);

}

//-------------------------------------

void Fraction::read() {

std::cout << "enter first: ";

std::cin >> m\_vector.x;

std::cout << "enter second: ";

std::cin >> m\_vector.y;

assert("The denominator cannot be zero!" && m\_vector.y != 0);

}

//-------------------------------------

void Fraction::show() {

std::cout << "fisrt: " << m\_vector.x

<< "\nsecond: " << m\_vector.y << std::endl;

assert("The denominator cannot be zero!" && m\_vector.y != 0);

double intpart = 0;

modf((float)(m\_vector.x / m\_vector.y), &intpart);

std::cout << "first / second = " << (float)m\_vector.x / m\_vector.y

<< "\nintpart: " << ipart()

<< "\nintpart modf: " << intpart << std::endl;

}

//-------------------------------------

int Fraction::ipart() {

assert("The denominator cannot be zero!" && m\_vector.y != 0);

return m\_vector.x / m\_vector.y;

}

//-------------------------------------