Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

Тема: «Лабораторная работа №1»

Выполнил

Студент группы РИС-22-2б

Прядеин И.А.

Проверил доцент кафедры

ИТАС

Полякова О. А.

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 8:**

Поле first - дробное число x, координата точки, поле second - дробное число y, координата точки. Реализовать метод distance() - вычисление расстояния от точки с координатами (first, second) до начала координат.

**Исходный код программы:**

**Файл “fraction.h”:**

struct fraction {

double first;

double second;

void Init(double, double); // инициализация полей

void Read(); // чтение значений полей

void Show(); // вывод значений полей

double distance(); // вычисление дистанции

};

**Файл “fraction.cpp”:**

#include <iostream>

#include <cmath>

#include "fraction.h"

using namespace std;

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

first = F;

second = S;

}

void fraction::Read() {

cout << "\nfirst: ";

cin >> first;

cout << "\nsecond: ";

cin >> second;

}

void fraction::Show() {

cout << "\nfirst: " << first;

cout << "\nsecond: " << second << "\n";

}

double fraction::distance() {

return sqrt(first \* first + second \* second);

}

**Файл “Lab\_1”:**

#include <iostream>

#include <cmath>

#include "fraction.cpp"

using namespace std;

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

fraction t;

t.Init(F, S);

return t;

}

int main() {

// Переменные A и B

fraction A;

fraction B;

A.Init(3.0, 2.3);

B.Read();

A.Show();

B.Show();

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

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

// Указатели

fraction \*X = new fraction;

X->Init(2.3, 5.4);

X->Show();

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

// Массивы

fraction array[3];

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

array[i].Read();

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

array[i].Show();

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

cout << "array[" << i << "].distance(" << array[i].first << "," << array[i].second << ") = " << array[i].distance() << endl;

// Динамические массивы

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

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

p\_array[i].Read();

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

p\_array[i].Show();

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

cout << "p\_array[" << i << "].distance(" << p\_array[i].first << "," << p\_array[i].second << ") = " << p\_array[i].distance() << endl;

// make\_fraction()

double x, y;

cout << "first: ";

cin >> x;

cout << "second: ";

cin >> y;

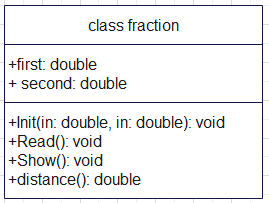
fraction F = make\_fraction(x, y);

F.Show();

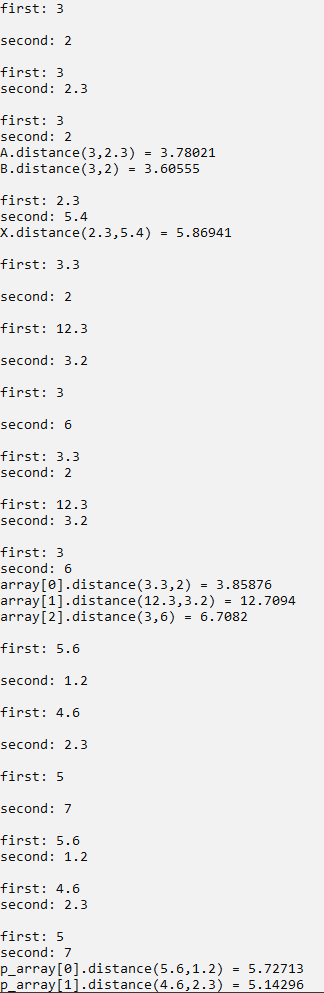
return 0;

}

**UML диаграмма:**



**Скриншот результата выполнения программы**



**Ответы на вопросы**

1. Что такое класс?

Класс – это абстрактный пользовательский тип данных, который позволяет представить модель реального или воображаемого объекта в виде данных и функций для работы с ними.

class my\_class {

};

1. Что такое объект (экземпляр) класса?

Объектом называется экземпляр класса – элемент класса

class my\_class {

};

int main() {

my\_class element;

};

1. Как называются поля класса?

Данные класса называются полями.

class my\_class {

int a; // поля

};

1. Как называются функции класса?

Функции класса - методы.

class my\_class {

public:

int a;

void show() {

cout << a << endl;

}

};

1. Для чего используются спецификаторы доступа?

Спецификаторы доступа управляют видимостью элементов класса.

1. Для чего используются спецификатор public?

public – методы и данные класса определяют его внешний интерфейс

Доступ к ним разрешён из любой части кода.

class my\_class {

public:

int a;

};

int main() {

my\_class element;

element.a = 4;

return 0;

}

1. Для чего используется спецификатор private?

private - методы и данные класса определяют его реализацию

- доступ разрешён только внутри этого класса.

- закрытые методы класса обычно используются публичными методами, решая внутренние задачи класса.

#include <iostream>

using namespace std;

class my\_class {

private:

int a;

public:

void set(int a) {

this->a = a;

}

void show() {

cout << "Переменная a = " << a << endl;

}

};

int main() {

my\_class element;

element.set(5);

element.show();

return 0;

}

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

По умолчанию спецификатор доступа для всех полей и методов - private.

class my\_class {

int a;

void set(int a) {

this->a = a;

}

void show() {

cout << "Переменная a = " << a << endl;

}

};

int main() {

my\_class element;

element.set(5); // ошибка доступа

element.show(); // ошибка доступа

return 0;

}

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

#include <iostream>

using namespace std;

struct my\_struct {

int a;

void set(int a) {

this->a = a;

}

void show() {

cout << "Переменная a = " << a << endl;

}

};

int main() {

my\_struct element;

element.set(5);

element.show(); // Переменная a = 5

return 0;

}

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

Для использования при описании интерфейса класса используется спецификатор public, так как доступ к методам класса разрешён из любой части программы.

1. Каким образом можно изменить значения атрибутов экземпляра класса?

Значения атрибутов экземпляра класса можно изменить с помощью модификаторов. Модификатор - метод, который изменяет какой-либо атрибут.

struct my\_struct {

int a;

void set(int a) {

this->a = a;

}

};

int main() {

my\_struct element;

element.set(5); // Изменяет атрибут а

return 0;

}

1. Каким образом можно получить значения атрибутов экземпляра класса?

Значения атрибутов экземпляра класса можно получить с помощью селекторов. Селектор - метод, который возвращает какой-либо атрибут.

struct my\_struct {

int a;

void set(int a) {

this->a = a;

}

int get() { // Селектор

return a;

}

void show() {

cout << "Переменная a = " << a << endl;

}

};

int main() {

my\_struct element;

element.set(5); // Изменяет атрибут а

int b = element.get(); // Получает атрибут а и присваивает его переменной b

cout << "Переменная b = " << b << endl;

return 0;

}

1. Класс описан 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 объект класса определён через указатель доступ к его полям осуществляется при помощи ->

s->name = “Ivan”;

1. Класс описан следующим образом

struct Student {

string name;

int group;

...

};

Объект класса определён следующим образом

Student s;

Как можно обратиться к полю name объекта s?

Объект класса определён напрямую, поэтому доступ к полю name осуществляется таким образом: s.name = “Ivan”;

1. Класс описан следующим образом

class Student {

string name;

int group;

...

};

Объект класса определён следующим образом

Student s;

Как можно обратиться к полю name объекта s?

Так как описание класса начинается со спецификатора class, поля структуры имеют спецификатор доступа private. Поэтому для доступа к полю name осуществляется при помощи модификатора, расположенного в классе:

class Student {

string name;

int group;

public:

void set\_name(string name) {

this->name = name;

}

};

int main() {

set\_name(“Ivan”);

return 0;

}

1. Класс определён следующим образом

class Student {

public:

string name;

int group;

...

};

Объект класса определён следующим образом

Student s;

Как можно обратиться к полю name объекта s?

Так как для поля name определён спецификатор доступа public, а объект задан напрямую, обратиться к полю name можно с помощью: s.name = “Ivan”;

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?

s->name = “Ivan”