Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

**по лабораторной работе № 18.5**

Дисциплина: Информатика

Тема: «Объектно-ориентированное программирование. Виртуальные функции. Полиморфизм»

Вариант 1

Выполнил работу:

студент группы РИС-20-1Б

Азмагулов Артём Вадимович

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

Пермь

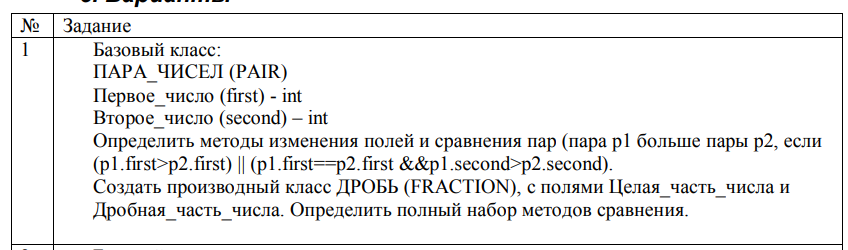
2021

**Цель работы**

* 1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
  2. Создание иерархии классов с использованием простого наследования и абстрактного класса.
  3. Изучение полиморфизма и виртуальных методов.

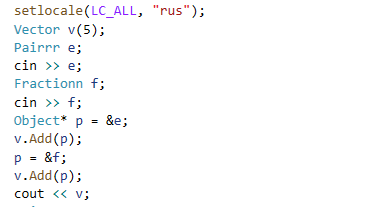
**Постановка задачи**

* + 1. Определить абстрактный класс.
    2. Определить иерархию классов, в основе которой будет находиться абстрактный класс (см. лабораторную работу №4).
    3. Определить класс Вектор, элементами которого будут указатели на объекты иерархии классов.
    4. Перегрузить для класса Вектор операцию вывода объектов с помощью потоков.
    5. В основной функции продемонстрировать перегруженные операции и полиморфизм Вектора

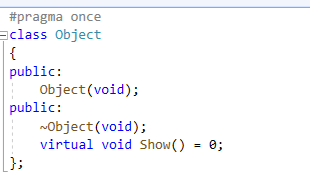


**Анализ задачи**

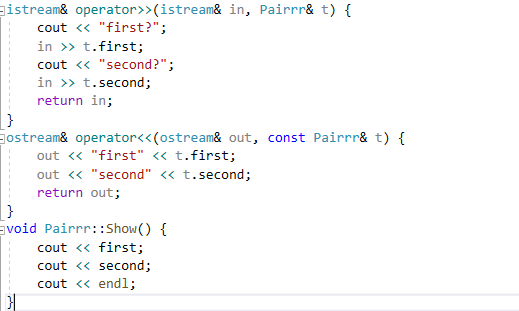
1. Необходимые действия
2. Добавить объекты классов Fraction, Pair в вектор.



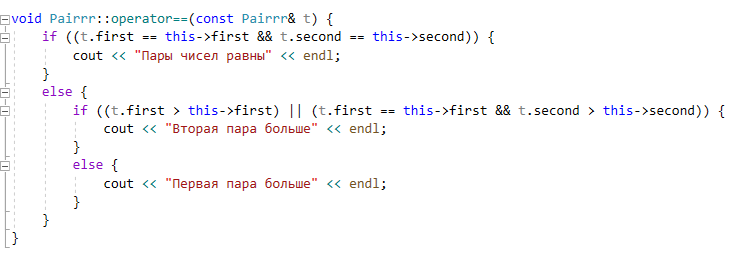
1. Реализовать абстрактный класс



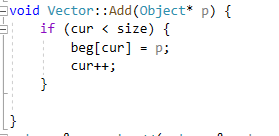
1. Реализовать ввод-вывод значений объектов класса Pairr с помощью потоков ввода-вывода



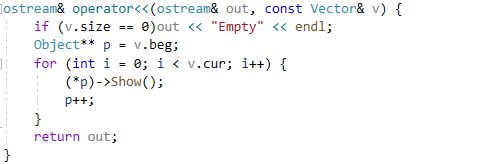
1. Реализовать операторы сравнения в классе TRIAD



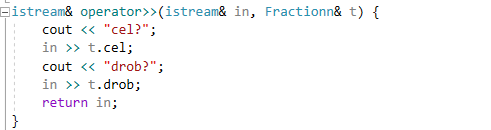
1. Реализовать добавление элементов в классе Vector



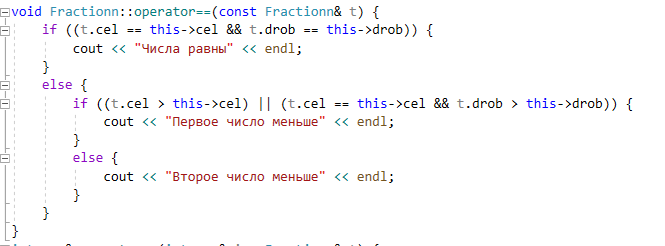
1. Реализовать вывод элементов в классе Vector



1. Реализовать операции ввода-вывода в классе Fraction с помощью потоков ввода-вывода



1. Реализовать операторы сравнения в классе TIME



1. Программный код

#pragma once

#include <iostream>

#include "Object.h"

#include "Pairrr.h"

#include "Fractionn.h"

#include <string>

#include "Vector.h"

using namespace std;

void main() {

setlocale(LC\_ALL, "rus");

Vector v(5);

Pairrr e;

cin >> e;

Fractionn f;

cin >> f;

Object\* p = &e;

v.Add(p);

p = &f;

v.Add(p);

cout << v;

Pairrr a;

Pairrr b;

cout << "Первая пара чисел" << endl;

cin >> a;

cout << "Вторая пара чисел" << endl;

cin >> b;

a == b;

Fractionn c;

Fractionn d;

cout << "Первое число" << endl;

cin >> c;

cout << "Второе число" << endl;

cin >> d;

c == d;

system("pause");

}

Object.h

#pragma once

class Object {

public:

Object() {};

~Object() {};

virtual void show() = 0;

};

Pairrr.h

#pragma once

#include <iostream>

#include "Object.h"

#include <string>

using namespace std;

class Pairrr :

public Object{

public:

Pairrr(void);

public:

virtual ~Pairrr(void);

void Show();

Pairrr(int f, int s) ;

Pairrr(const Pairrr& t1);

int get\_first() { return first; }

int get\_second() { return second; }

void set\_first(int f);

void set\_second(int s);

//bool get\_Pair(const Pair& t1);

void operator==(const Pairrr&);

Pairrr& operator=(const Pairrr&);

friend istream& operator>>(istream& in, Pairrr& t);

friend ostream& operator<<(ostream& out, const Pairrr& t);

protected:

int first, second;

};

Pairrr.cpp

#include "Pairrr.h"

Pairrr::Pairrr(void){

first = 0;

second = 0;

}

Pairrr::~Pairrr(void) {

}

Pairrr::Pairrr(int c, int p) {

first = c;

second = p;

}

Pairrr::Pairrr(const Pairrr& t) {

first = t.first;

second = t.second;

}

Pairrr& Pairrr::operator=(const Pairrr& c) {

if (&c == this)return \*this;

first = c.first;

second = c.second;

return \*this;

}

void Pairrr::set\_first(int f) {

first = f;

}

void Pairrr::set\_second(int s) {

second = s;

}

void Pairrr::operator==(const Pairrr& t) {

if ((t.first == this->first && t.second == this->second)) {

cout << "Пары чисел равны" << endl;

}

else {

if ((t.first > this->first) || (t.first == this->first && t.second > this->second)) {

cout << "Вторая пара больше" << endl;

}

else {

cout << "Первая пара больше" << endl;

}

}

}

istream& operator>>(istream& in, Pairrr& t) {

cout << "first?";

in >> t.first;

cout << "second?";

in >> t.second;

return in;

}

ostream& operator<<(ostream& out, const Pairrr& t) {

out << "first" << t.first;

out << "second" << t.second;

return out;

}

void Pairrr::Show() {

cout << first;

cout << second;

cout << endl;

}

Vector.h

#pragma once

#include"Object.h"

#include<string>

#include<iostream>

using namespace std;

class Vector

{public:

Vector(void);

Vector(int);

public:

~Vector(void);

void Add(Object\*);

friend ostream& operator<<(ostream& out, const Vector&);

private:

Object\*\* beg;

int size;

int cur;

};

Vector.cpp

#include "Vector.h"

Vector::Vector(void) {

beg = 0;

size = 0;

cur = 0;

}

Vector::~Vector(void) {

if (beg != 0) delete[]beg;

beg = 0;

}

Vector::Vector(int n) {

beg = new Object \* [n];

cur = 0;

size = n;

}

void Vector::Add(Object\* p) {

if (cur < size) {

beg[cur] = p;

cur++;

}

}

ostream& operator<<(ostream& out, const Vector& v) {

if (v.size == 0)out << "Empty" << endl;

Object\*\* p = v.beg;

for (int i = 0; i < v.cur; i++) {

(\*p)->Show();

p++;

}

return out;

}

Fraction.h

#pragma once

#include "Pairrr.h"

#include<iostream>

class Fractionn :public Pairrr {

public:

int cel, drob;

Fractionn() { cel = 0, drob = 0; };

Fractionn(int f, int s) { cel = f, drob = s; }

Fractionn(const Fractionn& t1) { first = t1.cel; second = t1.drob; }

~Fractionn() {};

void operator==(const Fractionn&);

friend istream& operator>>(istream& in, Fractionn& t);

friend ostream& operator<<(ostream& out, const Fractionn& t);

};

Fraction.cpp

#include "Fractionn.h"

void Fractionn::operator==(const Fractionn& t) {

if ((t.cel == this->cel && t.drob == this->drob)) {

cout << "Числа равны" << endl;

}

else {

if ((t.cel > this->cel) || (t.cel == this->cel && t.drob > this->drob)) {

cout << "Первое число меньше" << endl;

}

else {

cout << "Второе число меньше" << endl;

}

}

}

istream& operator>>(istream& in, Fractionn& t) {

cout << "cel?";

in >> t.cel;

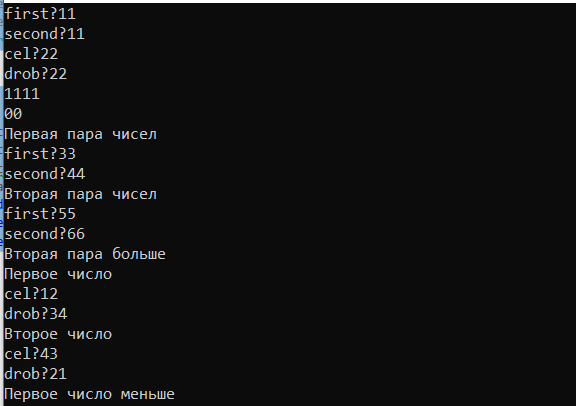
cout << "drob?";

in >> t.drob;

return in;

}

1. Демонстрация работы программы



1. Типы переменных

Object, Pairrr, Fraction, int, Vector

1. Контрольные вопросы
2. Какой метод называется чисто виртуальным? Чем он отличается от виртуального метода?

*-* Чисто виртуальный метод содержит признак = 0

1. Какой класс называется абстрактным?

- Класс, содержащий хотя бы один чисто виртуальный метод, называется абстрактным

1. Для чего предназначены абстрактные классы?

- Абстрактные классы предназначены для представления общих понятий, которые предполагается конкретизировать в производных классах

1. Что такое полиморфные функции?

- Это функции, работающие с объектом любого типа в пределах одной иерархии

1. Чем полиморфизм отличается от принципа подстановки?

- полиморфизм – это механизм, позволяющий работать с объектами разных типов одной иерархии. Принцип подстановки – это правило хорошего полиморфизма, которое говорит о том, что все методы класса – родителя должны быть применимы к объекту производного класса

1. Привести примеры иерархий с использованием абстрактных классов

- абстрактный класс <- человек <- сотрудник

1. Привести примеры полиморфных функций.

- При разработке компьютерной игры используется абстрактный класс. От него наследуются классы «истребитель главного героя» и «вражеские истребители». Функция перезарядки орудия работает как для вражеских истребителей, так и для истребителя главного героя

1. В каких случаях используется механизм позднего связывания?

- используется в случае вызова виртуального метода в производном классе. При этом виртуальный метод должен быть переписан в классе - наследнике