Пермский Национальный Исследовательский  
Политехнический Университет

**Лабораторная работа № 5**

Информатика

за 2 семестр

Вариант № 1

Выполнил:

Студент группы РИС 20-1-бз

КургановН.В.

20-ЭТФ-631

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

2021

**Наследование. Виртуальные функции. Полиморфизм.**

1. **Цель задания**.
   1. Создание консольного приложения, состоящего из нескольких файлов в системепрограммированияVisualStudio.
   2. Создание иерархии классов с использованием простого наследования иабстрактногокласса.
   3. Изучениеполиморфизмаивиртуальныхметодов.
2. **Задание.**

Базовыйкласс:

ПАРА\_ЧИСЕЛ(PAIR)

Первое\_число (first) – int

Второе\_число(second) – int

Определить методы изменения полей и сравнения пар (пара p1 больше пары р2, если(p1.first>p2.first)||(p1.first==p2.first&&p1.second>p2.second).

Создать производный класс ДРОБЬ (FRACTION), с полями Целая\_часть\_числа и Дробная\_часть\_числа. Определить полный набор методов сравнения.

1. **Файл main.cpp**

#include<iostream>

#include"Pair.h"

#include"Fraction.h"

#include"Vector.h"

#include"object.h"

usingnamespacestd;

voidmain()

{

Vectorv(5);//вектор из 5 элементов

Pair a;//объект класса Car

cin>> a;

Fraction b;// объект класса Lorry

cin>> b;

Object\* p = &a;//ставим указатель на объект класса Car

v.Add(p);//добавляем объект в вектор

p = &b;//ставим указатель на объект класса Lorry

v.Add(p); //добавляем объект в вектор

cout<< v;//вывод вектора

}

1. **Файл Vector.cpp**

#include"Vector.h"

//конструктор без параметров

Vector::Vector(void)

{

beg = 0;

size = 0;

cur = 0;

}

//деструктор

Vector::~Vector(void)

{

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

beg = 0;

}

//конструктор с параметрами

Vector::Vector(intn)

{

beg = newObject \* [n];

cur = 0;

size = n;

}

//добавление объекта, на который указывает указатель p в вектор

voidVector::Add(Object\* p)

{

if (cur<size)

{

beg[cur] = p;

cur++;

}

}

//операция вывода

ostream&operator<<(ostream&out, constVector&v)

{

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

Object\*\* p = v.beg;//указатель на указатель типа Object

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

{

(\*p)->Show();//вызов метода Show() (позднее связывание)

p++;//передвигаем указатель на следующий объект

}

returnout;

}

1. **Файл object.cpp**

#include"object.h"

Object::Object(void)

{

}

Object::~Object(void)

{

}

1. **Файл object.h**

#pragmaonce

classObject

{

public:

Object(void);

public:

~Object(void);

virtualvoidShow() = 0;//чисто виртуальная функция

};

1. **Файл Vector.h**

#include"object.h"

#include<string>

#include<iostream>

usingnamespacestd;

classVector

{

public:

Vector(void);//конструктор без параметров

Vector(int);//конструктор копирования

public:

~Vector(void);//деструктор

voidAdd(Object\*);//добавление элемента в вектор

friendostream&operator<<(ostream&out, constVector&);//операциявывода

private:

Object\*\* beg;//указатель на первый элемент вектора

intsize;//размер

intcur;//текущая позиция

};

1. **Файл Fraction.cpp**

#include"Fraction.h"

#include<math.h>;

#include<tchar.h>

#include<iostream>

#include<iomanip>

#include<sstream>

intfractional\_part\_as\_int(doublenumber) {

intnumber\_of\_decimal\_places;

std::stringstreamss;

ss<<std::setprecision(9) <<number;

std::stringstrNum = ss.str();

size\_t pos = strNum.find('.');

if (pos != strNum.npos)

number\_of\_decimal\_places = strNum.size() - 1 - pos;

else

number\_of\_decimal\_places = 0;

doubledummy;

double frac = modf(number, &dummy);

returnround(frac \* pow(10, number\_of\_decimal\_places));

}

//не параметризованный конструктор

Fraction::Fraction(void) :Pair()

{

intPart = 0;

fractPart = 0;

}

//деструктор

Fraction::~Fraction()

{

}

//параметризованный конструктор

Fraction::Fraction(intf, ints, intip, intfp) :Pair(f, s)

{

intPart = ip;

fractPart = fp;

}Fraction::Fraction(intf, ints, doublenum) : Pair(f, s)

{

intPart = (int)num;

fractPart = fractional\_part\_as\_int(num);

}

//клонирование

Fraction::Fraction(constFraction&Fraction)

{

first = Fraction.first;

second = Fraction.second;

intPart = Fraction.intPart;

fractPart = Fraction.fractPart;

}

//модификатор first

voidFraction::Set\_intPart(inti)

{

intPart = i;

}

//модификатор second

voidFraction::Set\_fractPart(intf)

{

fractPart = f;

}

//оператор присваивания

Fraction&Fraction::operator=(constFraction&Fraction)

{

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

first = Fraction.first;

second = Fraction.second;

intPart = Fraction.intPart;

fractPart = Fraction.fractPart;

return \*this;

}

//операторысравнения

booloperator==(constFraction&f1, constFraction&f2)

{

return ((f1.first == f2.first &&f1.second == f2.second) && (f1.intPart == f2.intPart &&f1.fractPart == f2.fractPart));

}

booloperator!=(constFraction&f1, constFraction&f2)

{

return !(f1==f2);

}

booloperator>(constFraction&f1, constFraction&f2)

{

return ((f1.first>f2.first) || (f1.first == f2.first &&f1.second >f2.second)) && ((f1.intPart >f2.intPart) || (f1.intPart == f2.intPart &&f1.fractPart >f2.fractPart));

}

booloperator>=(constFraction&f1, constFraction&f2)

{

return (f1>f2) || (f1==f2);

}

booloperator<(constFraction&f1, constFraction&f2)

{

return ((f1.first<f2.first) || (f1.first == f2.first &&f1.second <f2.second)) && ((f1.intPart <f2.intPart) || (f1.intPart == f2.intPart &&f1.fractPart <f2.fractPart));

}

booloperator<=(constFraction&f1, constFraction&f2)

{

return (f1<f2) || (f1==f2);

}

//операторввода

istream&operator>> (istream&in, Fraction&Fraction)

{

cout<<"\nFirst: "; in>>Fraction.first;

cout<<"\nSecond: "; in>>Fraction.second;

cout<<"\nInt part: "; in>>Fraction.intPart;

cout<<"\nFraction part: "; in>>Fraction.fractPart;

returnin;

}

//операторвывода

ostream&operator<< (ostream&out, constFraction&Fraction)

{

out<<"\nFirst: "<<Fraction.first;

out<<"\nSecond: "<<Fraction.second;

out<<"\nFraction: "<<Fraction.intPart<<"."<<Fraction.fractPart;

out<<"\n";

returnout;

}

voidFraction::Show()

{

cout<<"\nFirst: "<< first;

cout<<"\nSecond: "<< second;

cout<<"\nFraction: "<<intPart<<"."<<fractPart;

cout<<"\n";

}

1. **Файл Pair.cpp**

#include"Pair.h"

//не параметризованный конструктор

Pair::Pair(void)

{

first = 0;

second = 0;

}

//деструктор

Pair::~Pair()

{

}

//параметризованный конструктор

Pair::Pair(intf, ints)

{

first = f;

second = s;

}

//клонирование

Pair::Pair(constPair&pair)

{

first = pair.first;

second = pair.second;

}

//модификатор first

voidPair::Set\_first(intf)

{

first = f;

}

//модификатор second

voidPair::Set\_second(ints)

{

second = s;

}

//операторприсваивания

Pair&Pair::operator=(constPair&pair)

{

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

first = pair.first;

second = pair.second;

return \*this;

}

voidPair::Show()

{

cout<<"\nFirst: "<< first;

cout<<"\nSecond: "<< second;

cout<<"\n";

}

//операторысравнения

booloperator==(constPair&p1, constPair&p2)

{

return (p1.first == p2.first &&p1.second == p2.second);

}

booloperator!=(constPair&p1, constPair&p2)

{

return !(p1==p2);

}

booloperator>(constPair&p1, constPair&p2)

{

return (p1.first>p2.first) || (p1.first == p2.first &&p1.second >p2.second);

}

booloperator>=(constPair&p1, constPair&p2)

{

return (p1>p2) || (p1==p2);

}

booloperator<(constPair&p1, constPair&p2)

{

return (p1.first<p2.first) || (p1.first == p2.first &&p1.second <p2.second);

}

booloperator<=(constPair&p1, constPair&p2)

{

return (p1<p2) || (p1==p2);

}

//оператор ввода

istream&operator>> (istream&in, Pair&pair)

{

cout<<"\nFirst: "; in>>pair.first;

cout<<"\nSecond: "; in>>pair.second;

returnin;

}

//операторвывода

ostream&operator<< (ostream&out, constPair&pair)

{

out<<"\nFirst: "<<pair.first;

out<<"\nSecond: "<<pair.second;

out<<"\n";

returnout;

}

1. **Файл Pair.h**

#pragmaonce

#include<iostream>

#include"object.h"

usingnamespacestd;

classPair :

publicObject

{

protected:

//атрибуты

intfirst;

intsecond;

public:

//деконструктор

virtual ~Pair();

//конструкторы

Pair(void);

Pair(int, int);

Pair(constPair&);

//селекторы

intGet\_first() { return first; }

intGet\_second() { return second; }

//модификаторы

voidSet\_first(int);

voidSet\_second(int);

//перегрузка присваивания

Pair&operator=(constPair&);

//перегрузка операторов сравнения

friendbooloperator> (constPair&p1, constPair&p2);

friendbooloperator<= (constPair&p1, constPair&p2);

friendbooloperator< (constPair&p1, constPair&p2);

friendbooloperator>= (constPair&p1, constPair&p2);

friendbooloperator== (constPair&p1, constPair&p2);

friendbooloperator!= (constPair&p1, constPair&p2);

//перегрузка ввода вывода

friendistream&operator>> (istream&in, Pair&p);

friendostream&operator<< (ostream&out, constPair&p);

virtualvoidShow();

};

1. **Файл Fraction.h**

#pragmaonce

#include"Pair.h";

classFraction :

publicPair

{

protected:

//атрибуты

intintPart;

intfractPart;

public:

//деконструктор

virtual ~Fraction();

//конструкторы

Fraction(void);

Fraction(int, int, int, int);

Fraction(int, int, double);

Fraction(constFraction&);

//селекторы

intGet\_intPart() { returnintPart; }

intGet\_fractPart() { returnfractPart; }

//модификаторы

voidSet\_intPart(int);

voidSet\_fractPart(int);

voidSet\_All(double);

//перегрузка присваивания

Fraction&operator=(constFraction&);

//перегрузка операторов сравнения

friendbooloperator> (constFraction&p1, constFraction&p2);

friendbooloperator<= (constFraction&p1, constFraction&p2);

friendbooloperator< (constFraction&p1, constFraction&p2);

friendbooloperator>= (constFraction&p1, constFraction&p2);

friendbooloperator== (constFraction&p1, constFraction&p2);

friendbooloperator!= (constFraction&p1, constFraction&p2);

//перегрузкавводавывода

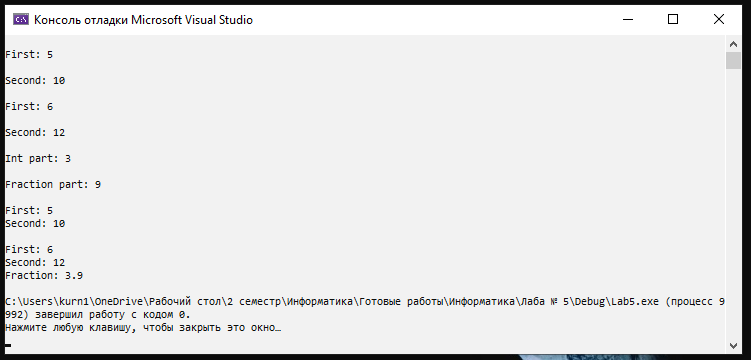
friendistream&operator>> (istream&in, Fraction&p);

friendostream&operator<< (ostream&out, constFraction&p);

virtualvoidShow();

};

1. **Результат работы программы**

****

1. **Контрольные вопросы.** 
   1. Какой метод называется чисто виртуальным? Чем он отличается от виртуальногометода?

Чисто виртуальный метод это метод содержащий признак = 0 вместо тела. Отличается от виртуального тем что при наследовании необходимо переопределять этот метод.

* 1. Какой класс называетсяабстрактным?

Класс называется абстрактным если содержит хотя бы 1 чисто виртуальный метод

* 1. Длячегопредназначеныабстрактныеклассы?

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

* 1. Чтотакоеполиморфныефункции?

Это функцию, параметром которой является указатель на абстрактный класс.

* 1. Чемполиморфизмотличаетсяотпринципаподстановки?

Возможностью работать с объектом любого типа в пределах одной иерархии

* 1. Привестипримерыиерархийсиспользованиемабстрактныхклассов.

Абстрактный класс 4-х угольник

класс выпуклый:4-х угольник

класс трапеция: выпуклый

класс параллелограмм: трапеция

Абстрактный класс животные

класс собака: животные

класс кошка: животные

класс птица: животные

* 1. Привестипримерыполиморфныхфункций.

voidVector::Add(Object \*p)

{

if (cur<size)

{

beg[cur] = p;

cur++;

}

}

* 1. В каких случаях используется механизм позднего связывания?

Когда мы не можем заранее определить от какого класса из иерархии будет вызвана функция