Министерство науки и высшего образования РФ

Федеральное бюджетное государственное образовательное учреждение высшего профессионального образования

Пермский национальный исследовательский политехнический университет (ПНИПУ)

Факультет: Электротехнический (ЭТФ)

Направление: 09.03.01 «Информатика и вычислительная техника (Автоматизированные системы обработки информации и управления)»

Кафедра: «Информационные технологии и автоматизированных систем» (ИТАС)

Информатика

Лабораторная работа №4

Простое наследование. Принцип подстановки

Студент: Балтаев Э. А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись)

Группа: АСУ-20-1бзу

Работу проверил:

доцент кафедры ИТАС

Полякова О. А.

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Подпись:\_\_\_\_\_\_\_\_\_\_\_

Пермь 2021г.

Постановка задачи

1. Определить пользовательский класс

2. Определить в классе следующие конструкторы: без параметров с параметрами и копирования

3. Определить в классе деструктор

4. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы)

5. Перегрузить операцию присваивания

6. Перегрузить операцию ввода и вывода объектов с помощью потоков

7. Определить производный класс

8. Написать программу, в которой продемонстрировать создание объектов и работу всех перегруженных операций.

9. Реализовать функции, получающие и возвращающие объект базового класса. Продемонстрировать принцип подстановки.

Задание варианта

Базовый класс:

ПАРА\_ЧИСЕЛ (PAIR)

Первое\_число (first) – int

Второе\_число (second) – int

Определить методы проверки на равенство и операцию перемножения полей. Реализовать операцию вычитания пар по формуле (a,b)-(c-d) = (a-b, c-d).

Создать производный класс ПРОСТАЯ\_ДРОБЬ (Rational), с полями Числитель и Знаменатель. Переопределить операцию вычитания и определить операции сложения и умножения простых дробей.

Описание базового класса

#pragma once

#include <iostream>

using namespace std;

class Pair

{

protected:

int first;

int second;

public:

Pair() { first = second = 0; }

Pair(int f, int s) { first = f; second = s; }

Pair(const Pair& p) { first = p.first; second = p.second; }

~Pair() {}

int get\_first() { return first; }

int get\_second() { return second; }

void set\_first(int f) { first = f; }

void set\_second(int s) { second = s; }

Pair& operator = (const Pair& p);

bool operator == (const Pair& p);

Pair operator \* (int val);

Pair operator - (const Pair& p);

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

friend ostream& operator<< (ostream& out, const Pair& p);

};

Определение дочернего класса

#pragma once

#include "pair.h"

class Rational :

public Pair

{

public:

Rational() : Pair() {}

Rational(int f, int s) : Pair() { set\_first(f); set\_second(s); }

Rational(const Rational& r) : Pair(r) {}

~Rational() {}

void set\_first(int f);

void set\_second(int s);

Rational& operator= (const Rational& r);

Rational operator- (const Rational& r);

Rational operator+ (const Rational& r);

Rational operator\* (const Rational& r);

friend istream& operator>> (istream& in, Rational& r);

friend ostream& operator<< (ostream& out, const Rational& r);

};

Определение компонентных функций базового класса

#include "pair.h"

Pair& Pair::operator=(const Pair& p)

{

if (&p == this)

return \*this;

first = p.first;

second = p.second;

return \*this;

}

bool Pair::operator==(const Pair& p)

{

if (first == p.first && second == p.second)

return true;

return false;

}

Pair Pair::operator\*(int val)

{

Pair temp(first \* val, second \* val);

return temp;

}

Pair Pair::operator-(const Pair& p)

{

Pair temp(first - second, p.first - p.second);

return temp;

}

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

{

cout << "first? "; in >> p.first;

cout << "second? "; in >> p.second;

return in;

}

ostream& operator<<(ostream& out, const Pair& p)

{

out << "(" << p.first << "," << p.second << ")";

return out;

}

Определение компонентных функций дочернего класса

#include "Rational.h"

void Rational::set\_first(int f)

{

first = f;

}

void Rational::set\_second(int s)

{

if (s == 0) {

cout << "Ошибка знаменателя, равен 0" << endl;

second = 1;

}

else

second = s;

}

Rational& Rational::operator=(const Rational& r)

{

if (&r == this)

return \*this;

first = r.first;

second = r.second;

return \*this;

}

Rational Rational::operator-(const Rational& r)

{

Rational temp;

// если знаменатели равны)

if (second == r.second) {

temp.first = first - r.first;

temp.second = second;

}

else {

temp.second = second \* r.second;

temp.first = first \* r.second - r.first \* second;

}

return temp;

}

Rational Rational::operator+(const Rational& r)

{

Rational temp;

// если знаменатели равны)

if (second == r.second) {

temp.first = first + r.first;

temp.second = second;

}

else {

temp.second = second \* r.second;

temp.first = first \* r.second + r.first \* second;

}

return temp;

}

Rational Rational::operator\*(const Rational& r)

{

Rational temp;

temp.first = first \* r.first;

temp.second = second \* r.second;

return temp;

}

istream& operator>>(istream& in, Rational& r)

{

int f, s;

cout << "num? "; in >> f; r.set\_first(f);

cout << "den? "; in >> s; r.set\_second(s);

return in;

}

ostream& operator<<(ostream& out, const Rational& r)

{

if (r.first > 0 && r.second || r.first < 0 && r.second < 0) {

out << abs(r.first) << "/" << abs(r.second);

}

else {

if (r.first < 0 && r.second > 0)

out << r.first << "/" << r.second;

else if (r.first > 0 && r.second < 0)

out << "-" << r.first << "/" << abs(r.second);

}

return out;

}

Определение главного файла

#include <iostream>

#include "Rational.h"

// принцип подстановки

void showBase(Pair& r) {

cout << "\nПолучение базового класса из дочернего\n";

cout << r << endl;

}

Pair getBase() {

cout << "\nВозврат базового класса из дочернего\n";

Rational r;

cin >> r;

return r;

}

int main()

{

system("chcp 1251");

cout << "\nБазовый класс\n";

Pair p1;

cin >> p1;

Pair p2(4, 10);

cout << "p1 - " << p1 << endl;

cout << "p2 - " << p2 << endl;

cout << "p1 == p2 - " << (p1 == p2) << endl;

p2 = p2 \* 3;

cout << "p2 \* 3 - " << p2 << endl;

cout << "p1 - p2 - " << (p1 - p2) << endl;

cout << "\nДочерний класс\n";

Rational r1;

cin >> r1;

Rational r2 = r1;

cout << "r1 - " << r1 << endl;

cout << "r2 = r1 - " << r2 << endl;

cout << "\nПринцип подстановки\n";

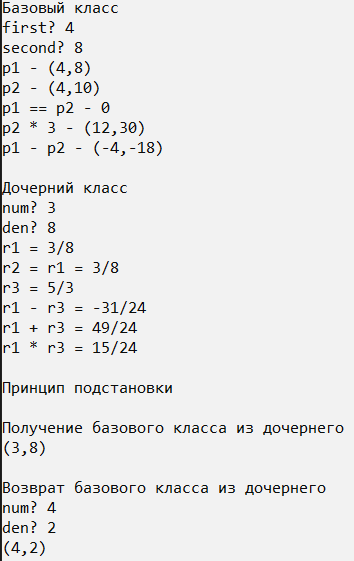
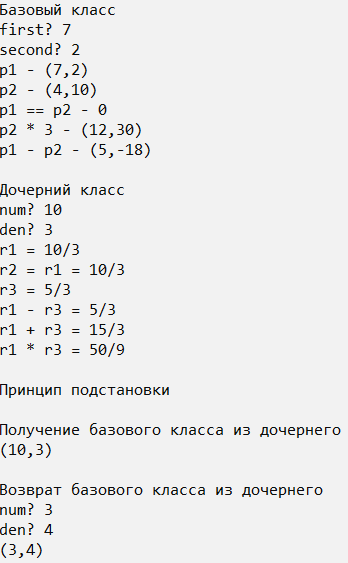
showBase(r1);

Pair p3 = getBase();

cout << p3 << endl;

}

Результаты программы

Ответы на контрольные вопросы

1) Чтобы добавить некоторую функциональность в дочерние классы, расширить базовый класс или переопределить некоторые методы

2) Доступно как в дочерних классах, так и вне файла

3) Доступно только в базовом классе

4) Доступно только в дочерних классах

5) class дочерний : классы\_родители { компоненты\_дочернего }

6) Нет

7) Нет

8) Сверху вниз

9) Снизу вверх

10) Виртуальная функция позволяет переопределить функцию базового класса со своей реализацией. Связывание же – будет использоваться метод того типа (класса) к которому указывает ссылка (указатель) в момент вызова.

11) Конструкторы нет, деструкторы да

12) Да

13) Открытое (class Child: public Parent): наследник является частью родителя (где используется тип базового класса, может использовать тип дочернего – принцип подстановки)

14) Закрытие (class Child: private Parent): наследование реализации – все методы базового класса нужно реализовывать заново. Принцип подстановки не работает.

15) На место объекта (ссылки) базового класса может быть подставлен объект (ссылка) дочернего класса

16) Доступные поля: name, post, stage

17)

Student() { … }

Employee() : Student() { … }

Teacher() : Employee() { … }

18)

Student(int a, string n) { age = a; name = n; }

Employee(int a, string n, post p) : Student(a, n) { post = p; }

Teacher(int a, string n, post p, stage s) : Employee(a, n, p) { stage = s; }

19)

Student(const Student& s) { age = s.age; name = s.name; }

Employee(const Employee& e) : Student(e) { post = e.post; }

Teacher(const Teacher& t) : Employee(t) { stage = t.stage; }

20)

Student& operator= (const Student& s)

{

if(&s == this)

return \*this;

age = s.age;

name = s.name;

return \*this;

}

Employee& operator= (const Employee & e)

{

if(&e == this)

return \*this;

setAge(e.getAge());

name = e.name;

post = e.post;

return \*this;

}

Teacher& operator= (const Teacher & t)

{

if(&t == this)

return \*this;

setAge(t.getAge());

name = t.name;

post = t.post;

stage = t.stage;

return \*this;

}