Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

**Пермский национальный исследовательский политехнический университет**

Факультет Электротехнический Кафедра ИТАС

Специальность Промышленная Робототехника

**ОТЧЁТ**

**о лабораторной работе №5**

Наследование. Виртуальные функции. Полиморфизм.

|  |  |
| --- | --- |
|  | Выполнил:  Студент группы ПРТ-21-1Б  Торган Г.А.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Проверил:  Доцент кафедры ИТАС Полякова О.А.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Пермь 2022**

**Постановка задачи:**

Базовый класс:

ЧЕЛОВЕК (PERSON)

Имя (name) – string

Возраст (age) – int

Определить методы изменения полей.

Создать производный класс TEACHER, имеющий поля Предмет – string и Количество часов – int. Определить методы изменения полей, а также увеличения и уменьшения часов

**Текст программы:**

**Lab5.cpp**

#include "Object.h"

#include "P.h"

#include "L.h"

#include "Vector.h"

#include <string>

#include <iostream>

using namespace std;

int main()

{

Vector v(5);//вектор из 5 элементов

PERSON a;//объект класса C

cin >> a;

TEACHER b;// объект класса L

cin >> b;

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

v.Add(p);//добавляем объект в вектор

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

v.Add(p); //добавляем объект в вектор

cout << v;//вывод вектора

}

**L.cpp**

#include "L.h"

TEACHER::TEACHER(void) :PERSON()

{

time = 0;

}

TEACHER::~TEACHER(void)

{

}

TEACHER::TEACHER(string M, int C,int G) :PERSON(M, C)

{

time = G;

}

TEACHER::TEACHER(const TEACHER& L)

{

name = L.name;

age = L.age;

time = L.time;

}

void TEACHER::Set\_time(int G)

{

time = G;

}

TEACHER& TEACHER::operator=(const TEACHER& l)

{

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

name = l.name;

age = l.age;

return \*this;

}

istream& operator>>(istream& in, TEACHER& l)

{

cout << "\nname:"; in >> l.name;

cout << "\nage:"; in >> l.age;

cout << "\ntime:"; in >> l.time;

return in;

}

ostream& operator<<(ostream& out, const TEACHER& l)

{

out << "\nname : " << l.name;

out << "\nage : " << l.age;

out << "\ntime : " << l.time;

out << "\n";

return out;

}

void TEACHER::Show()

{

cout << "\nname : " << name;

cout << "\nage : " << age;

cout << "\ntime : " << time;

cout << "\n";

}

**L.h**

#pragma once

#include "P.h"

class TEACHER :

public PERSON

{

public:

TEACHER(void);

public:

~TEACHER(void);

void Show();//функция для просмотра атрибутов класса с помощью указателя

TEACHER(string, int, int);

TEACHER(const TEACHER&);

int Get\_time() { return time; }

void Set\_time(int);

TEACHER& operator=(const TEACHER&);

friend istream& operator>>(istream& in, TEACHER& l);

friend ostream& operator<<(ostream& out, const TEACHER& l);

protected:

int time;

};

**object.h**

#pragma once

class Object

{

public:

Object(void);

public:

~Object(void);

virtual void Show() = 0;//чисто виртуальная функция

};

**Vector.h**

#pragma once

#include "object.h"

#include <string>

#include <iostream>

using namespace std;

class Vector

{

public:

Vector(void);//конструктор без параметров

Vector(int);//конструктор копирования

public:

~Vector(void);//деструктор

void Add(Object\*);//добавление элемента в вектор

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

private:

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

int size;//размер

int cur;//текущая позиция

};

**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(int n)

{

beg = new Object \* [n];

cur = 0;

size = n;

}

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

void Vector::Add(Object\* p)

{

if (cur < size)

{

beg[cur] = p;

cur++;

}

}

//операция вывода

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

{

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

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

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

{

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

p++;//передвигаем указатель на следующий объект

}

return out;

}

**P.h**

#pragma once

#include "object.h"

#include <string>

#include <iostream>

using namespace std;

class PERSON :

public Object

{

public:

PERSON(void);

public:

virtual ~PERSON(void);

void Show();//функция для просмотра атрибутов класса с помощью указателя

PERSON(string, int);

PERSON(const PERSON&);

string Get\_name() { return name; }

int Get\_age() { return age; }

void Set\_name(string);

void Set\_age(int);

PERSON& operator=(const PERSON&);

friend istream& operator>>(istream& in, PERSON& c);

friend ostream& operator<<(ostream& out, const PERSON& c);

protected:

string name;

int age;

};

**Vector.cpp**

#include "P.h"

PERSON::PERSON(void)

{

name = "";

age = 0;

}

PERSON::~PERSON(void)

{

}

PERSON::PERSON(string M, int C)

{

name = M;

age = C;

}

PERSON::PERSON(const PERSON& PERSON)

{

name = PERSON.name;

age = PERSON.age;

}

void PERSON::Set\_age(int C)

{

age = C;

}

void PERSON::Set\_name(string M)

{

name = M;

}

PERSON& PERSON::operator=(const PERSON& c)

{

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

name = c.name;

age = c.age;

return \*this;

}

istream& operator>>(istream& in, PERSON& c)

{

cout << "\nname:"; in >> c.name;

cout << "\nage:"; in >> c.age;

return in;

}

ostream& operator<<(ostream& out, const PERSON& c)

{

out << "\nname : " << c.name;

out << "\nage : " << c.age;

out << "\n";

return out;

}

void PERSON::Show()

{

cout << "\nname : " << name;

cout << "\nage : " << age;

cout << "\n";

}

**Ответ для варианта №14**