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 №4**

Простое наследование. Принцип подстановки.

|  |  |
| --- | --- |
|  | 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:**

**Lab4.cpp**

#include <iostream>

#include "Person.h"

#include "TEACHER.h"

using namespace std;

// глобальные функции

void f1(Person& c)

{

c.Set\_name("Mihail");

cout << c;

}

Person f2()

{

TEACHER l("Natalia", 32, "Math", 85);

return l;

}

int main()

{

//работа с классом Person

Person a;

cin >> a;

cout << a;

Person b("Maxim", 27);

cout << b;

a = b;

cout << a;

//работа с классом TEACHER

TEACHER c;

cin >> c;

cout << c;

//принцип подстановки

f1(c);//передаем объект класса TEACHER

a = f2();//создаем в функции объект класса TEACHER

cout << a;

return 0;

}

**Person.h**

#pragma once

#pragma once

#include <string>

#include <iostream>

using namespace std;

class Person

{

//конструктор без параметров

public:

Person(void);

public:

//деструктор

virtual ~Person(void);

//констрктор с параметрами

Person(string, int);

//конструктор копирования

Person(const Person&);

//селекторы

string Get\_name() { return name; }

int Get\_age() { return age; }

//модификаторы

void Set\_name(string);

void Set\_age(int);

//перегрузка операции присваивания

Person& operator=(const Person&);

//глобальные операторы-функции ввода-вывода

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

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

//атрибуты

protected:

string name;

int age;

};

**Person.cpp**

#include "Person.h"

//конструктор без параметров

Person::Person(void)

{

name = "";

age = 0;

}

//деструктор

Person::~Person(void)

{

}

//конструктор с параметрами

Person::Person(string M, int C)

{

name = M;

age = C;

}

//конструктор копирования

Person::Person(const Person& Person)

{

name = Person.name;

age = Person.age;

}

//модификаторы

void Person::Set\_age(int C)

{

age = C;

}

void Person::Set\_name(string M)

{

name = M;

}

//перегрузка операции присваивания

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

{

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

name = c.name;

age = c.age;

return \*this;

}

//глобальная функция для ввода

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

{

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

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

return in;

}

//глобальная функция для вывода

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

{

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

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

out << "\n";

return out;

}

**Teacher.h**

#pragma once

#include "Person.h"

//класс TEACHER наследуется от класса Person

class TEACHER :

public Person

{

public:

TEACHER(void);//конструктор без параметров

public:

~TEACHER(void);//деструктор

TEACHER(string, int,string, int);//конструктор с параметрами

TEACHER(const TEACHER&);//конструктор копирования

int Get\_hour() { return hour; }//модификатор

void Set\_hour(int);//селектор

string Get\_subject() { return subject; }//модификатор

void Set\_subject(string);//селектор

TEACHER& operator=(const TEACHER&);//операция присваивания

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

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

protected:

string subject;

int hour;//атрибут

};

**Teacher.cpp**

#include "TEACHER.h"

//using namespace TEACHER;

using namespace std;

//конструктор без параметров

TEACHER::TEACHER(void) :Person()

{

subject = "";

hour = 0;

}

//дестрктор

TEACHER::~TEACHER(void)

{

}

//конструктор с параметрами

TEACHER::TEACHER(string M, int C, string P ,int G) :Person(M, C)

{

subject = P;

hour = G;

}

//конструктор копирования

TEACHER::TEACHER(const TEACHER& L)

{

name = L.name;

age = L.age;

subject = L.subject;

hour = L.hour;

}

//модификатор

void TEACHER::Set\_hour(int G)

{

hour = G;

}

void TEACHER::Set\_subject(string P)

{

subject = P;

}

//оперция присваивания

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

{

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

name = l.name;

age = l.age;

subject = l.subject;

hour = l.hour;

return \*this;

}

//операция ввода

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

{

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

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

cout << "\nsubject:"; in >> l.subject;

cout << "\nhour:"; in >> l.hour;

return in;

}

//операция вывода

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

{

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

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

out << "\nsubject : " << l.subject;

out << "\nhour : " << l.hour;

out << "\n";

return out;

}

**Ответ для варианта №14**

