Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

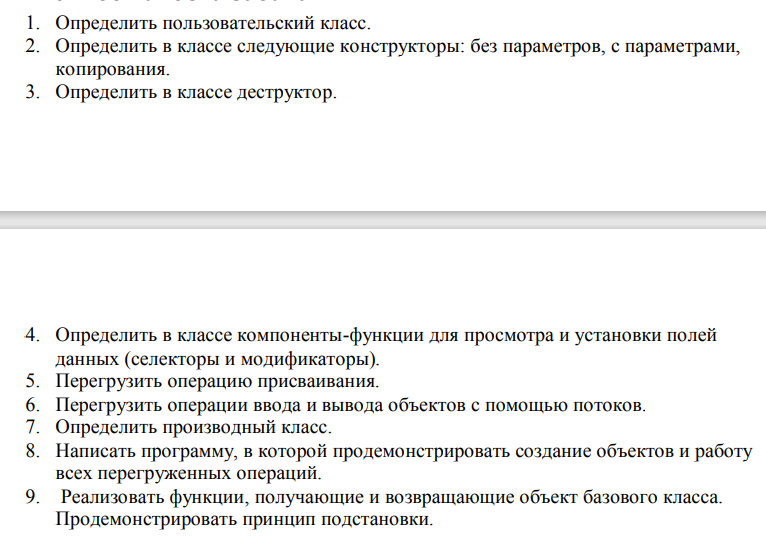
**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: Лабораторная работа по классам №4

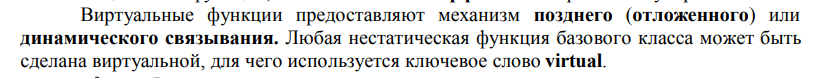
Выполнил работу  
Студент группы РИС-22-1Б  
Ишемцева Мария Андреевна  
Проверил  
Доцент кафедры ИТАС  
Полякова Ольга Андреевна

Постановка задачи:

 Анализ задачи:

Пользовательский класс – Person, производный класс Student. Использовано 3 стандартных конструктора, а также инициализация объектов при помощи сеттеров. Реализована перегрузка оператора=. Реализован метод вывода на экран данных(show()). Инициируем вывод производного класса функцией show().

Ответы на вопросы:

1. Для передачи полей потомкам, чтоб лишний раз не объявлять их в потомственном классе.
2. Полный доступ к методам и полям в public секции.
3. Private не наследуются, доступ возможен только в классе-предке.
4. Protected наследуется как private.
5. Class BaseClass: NewClass.
6. Конструкторы не наследуются.
7. Деструкторы не наследуются.
8. Сначала объект самого базового класса и дальше по старшинству.
9. Аналогично обычному деструктору.
10. 
11. Конструкторы нет, деструкторы да.
12. Да, виртуальность наследуется.
13. Класс потомок имеет доступ ко всем возможностям класса предка.
14. Закрытое наследование – наследование только реализации.
15. Везде, где можно использован объект базового класса, можно использовать вместо него и объект производного.
16. Name и Stage, private age не наследуется, post наследуется в private секции.
17. Student() {age=none; name=’none’;}; Employee() {post=’none’;}; Teacher() {stage=none;};
18. Student(int valueAge, string ValueName) {this->age= valueAge; this->name= ValueName;}; Employee(string ValuePost) {this->post= ValuePost;}; Teacher(int ValueStage) {this->stage= ValueStage;};
19. Student(const Student&other) {this->age=other.age; this->name= other.name;}; Employee(const Employee&other) {this->post= other.post;}; Teacher(const Teacher&other) {this->stage= other.stage;};
20. Student & operator=( const Student&other)

{

this->age=other.age;

this->name= other.name;

return \*this;

}

Employee & operator=( const Employee &other)

{

this->post= other.post; return \*this;

}

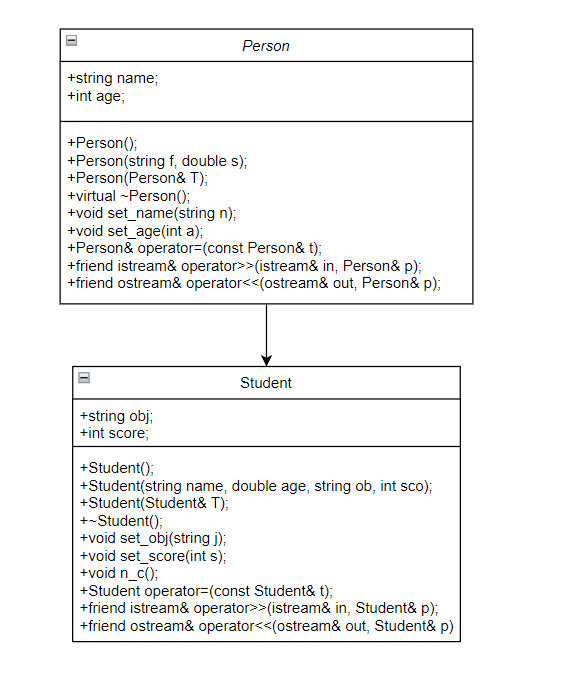
Teacher & operator=( const Teacher &other)

{

this->stage= other.stage; return \*this;

}

Блок-схема:



Код программы:

#include <iostream>

#include <windows.h>

//#include "Person.h"

#include "Student.h"

using namespace std;

void f1(Person& p) {

p.set\_name("Иван");

}

Person f2() {

Student s("Роман", 18, "Информатика", 4);

return s;

}

int main() {

setlocale(0, "RUS");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Person a;

cin >> a;

cout << "а - \n";

cout << a;

Person b("Катя", 18);

cout << "b -\n";

cout << b;

a = b;

cout << "Теперь а это b\n";

cout << "Вывод а\n";

cout << a;

Student c;

cin >> c;

cout << "Данные студента до изменения имени:\n";

cout << c;

c.n\_c();

f1(c);

cout << "Данные студента после изменения имени\n" << c;

a = f2();

cout << a;

return 0;

}

#include <iostream>

#include <windows.h>

//#include "Person.h"

#include "Student.h"

using namespace std;

void f1(Person& p) {

p.set\_name("Иван");

}

Person f2() {

Student s("Роман", 18, "Информатика", 4);

return s;

}

int main() {

setlocale(0, "RUS");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Person a;

cin >> a;

cout << "а - \n";

cout << a;

Person b("Катя", 18);

cout << "b -\n";

cout << b;

a = b;

cout << "Теперь а это b\n";

cout << "Вывод а\n";

cout << a;

Student c;

cin >> c;

cout << "Данные студента до изменения имени:\n";

cout << c;

c.n\_c();

f1(c);

cout << "Данные студента после изменения имени\n" << c;

a = f2();

cout << a;

return 0;

}

#include <iostream>

#include "Person.h"

using namespace std;

Person::Person() {

name = "";

age = 0;

cout << "Конструктор без параметров для " << this << endl;

}

Person::Person(string f, double s) {

name = f;

age = s;

cout << "Коснтруктор с параметрами для - " << this << endl;

}

Person::Person(Person& T) {

name = T.name;

age = T.age;

cout << "Конструктор копирования для - " << this << endl;

}

Person::~Person() {

cout << "Деструктор для - " << this << endl;

}

void Person::set\_name(string n) {

name = n;

}

void Person::set\_age(int a) {

age = a;

}

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

{

//проверка на самоприсваивание

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

name = t.name;

age = t.age;

return \*this;

}

istream& operator>>(istream& in, Person& p) {

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

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

return in;

}

ostream& operator<<(ostream& out, Person& p)

{

return (out << "Имя: " << p.name << endl << "Возраст: " << p.age << endl);

}

#include <iostream>

#include "Person.h"

using namespace std;

class Student :

public Person {

public:

string obj;

int score;

Student();

Student(string name, double age, string ob, int sco);

Student(Student& T);

~Student();

void set\_obj(string j);

void set\_score(int s);

void n\_c();

Student operator=(const Student& t);

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

friend ostream& operator<<(ostream& out, Student& p);

};

#include <iostream>

using namespace std;

class Person {

public:

string name;

int age;

Person();

Person(string f, double s);

Person(Person& T);

virtual ~Person();

void set\_name(string n);

void set\_age(int a);

Person& operator=(const Person& t);

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

friend ostream& operator<<(ostream& out, Person& p);

};

Скриншоты результатов:

