Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

По дисциплине: «ЯП»

Тема: «Наследование и виртуальные функции»

Выполнил:

Студент 2 курса

Группы ПО-7

Мозоль И.П.

Проверил:

Бойко Д.О.

Брест 2021

Вариант 1

Цель работы: Получить практические навыки создания иерархии классов и использования статических компонентов класса.

**Постановка задачи:**

Написать программу, в которой создается иерархия классов. Включить полиморфные объекты в связанный список, используя статические компоненты класса. Показать использование виртуальных функций.

Порядок выполнения работы:

1. Определить иерархию классов (в соответствии с вариантом).
2. Определить в классе статическую компоненту - указатель на начало связанного списка объектов и статическую функцию для просмотра списка.
3. Реализовать классы.
4. Написать демонстрационную программу, в которой создаются

объекты различных классов и помещаются в список, после чего список просматривается.

1. Сделать соответствующие методы не виртуальными и посмотреть, что будет.
2. Реализовать вариант, когда объект добавляется в список при создании, т.е. в конструкторе.

**Вариант задания:** Перечень классов: студент, преподаватель, персона, завкафедрой;

**“class.h”**

#ifndef **LABA\_2\_YAP\_C\_\_\_CLASS\_H**#define **LABA\_2\_YAP\_C\_\_\_CLASS\_H**#include **<iostream>**#include **<string>**#include **<list>  
class** Person{  
**protected**:  
 **static** std::list <Person\*> obj\_list;  
 std::string name;  
 **int** age;  
**public**:  
 Person(std::string Name, **int** Age);  
 **void** add();  
 **virtual** ~Person();  
 **int** getAge();  
 std::string getName();  
 **void** setName(std::string name);  
 **void** setAge(**int** age);  
 **virtual void** show()=0;  
 **static void** print();  
};  
**class** Teacher: **public** Person {  
**protected**:  
 **int** work\_exp;  
**public**:  
 Teacher(std::string Name, **int** Age, **int** Work\_exp);  
 **virtual** ~Teacher();  
 **void** show() **override**;  
};  
**class** Head\_dep:**public** Teacher{  
**protected**:  
 std::string dep\_name;  
**public**:  
 Head\_dep(std::string Name, **int** Age, **int** Work\_exp, std::string Dep\_name);  
 **virtual** ~Head\_dep();  
 **void** show() **override**;  
};  
**class** Student:**public** Person{  
**protected**:  
 **int** course;  
**public**:  
 Student(std::string Name, **int** Age, **int** Course);  
 **virtual** ~Student();  
 **void** show() **override**;  
};  
  
#endif *//LABA\_2\_YAP\_C\_\_\_CLASS\_H*

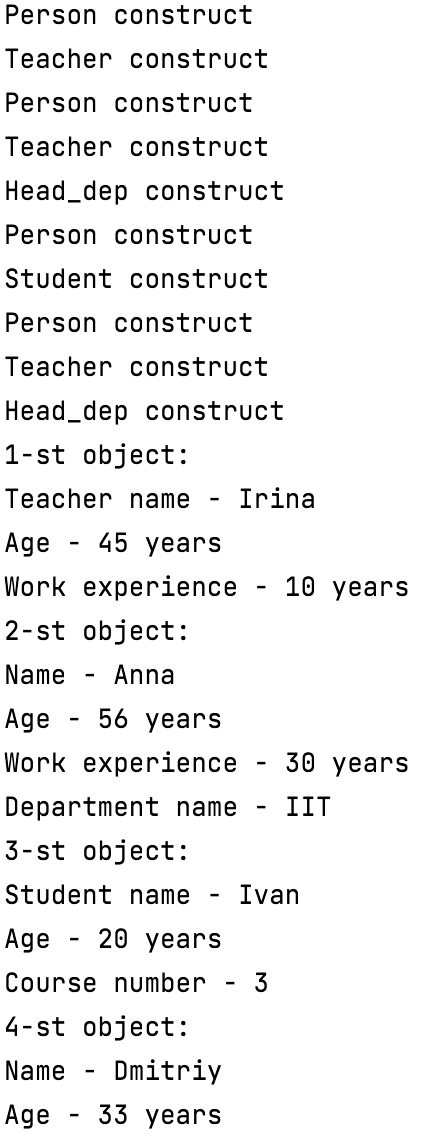
**“class.cpp”**

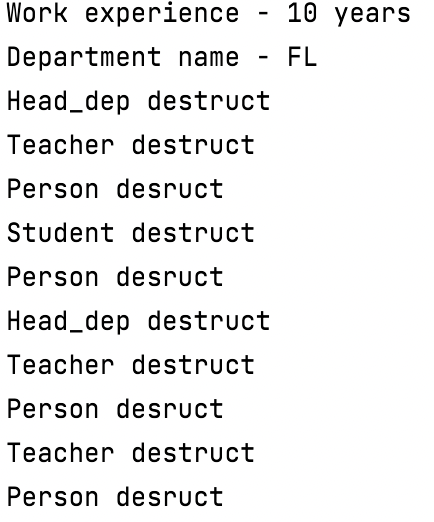
#include **"class.h"**Person :: Person(std::string Name, **int** Age):  
name(Name),age(Age)  
{  
 std::cout <<**"Person construct\n"**;  
 add();  
}  
**void** Person :: add(){  
 obj\_list.push\_back(**this**);  
}  
Person :: ~Person(){  
 std::cout << **"Person desruct\n"**;  
}  
**int** Person:: getAge() {  
 **return** age;  
}  
std::string Person:: getName(){  
 **return** name;  
}  
  
**void** Person:: setName(std::string name) {  
 Person::name = name;  
}  
  
**void** Person:: setAge(**int** age) {  
 Person::age = age;  
}  
  
**void** Person:: print(){  
 **if**(!obj\_list.empty()) {  
 **int** i = 1;  
 **for** (Person \*obj: obj\_list) {  
 std::cout << i << **"-st object:\n"**;  
 obj->show();  
 i++;  
 }  
 }  
}  
  
  
Teacher:: Teacher(std::string Name, **int** Age, **int** Work\_exp):  
Person(Name,Age), work\_exp(Work\_exp){  
 std::cout <<**"Teacher construct\n"**;  
}  
  
Teacher:: ~Teacher(){  
 std::cout <<**"Teacher destruct\n"**;  
}  
  
**void** Teacher:: show() {  
std::cout <<**"Teacher name - "**<< name <<**"\n"**;  
std::cout <<**"Age - "** << age <<**" years\n"**;  
std::cout <<**"Work experience - "**<<work\_exp <<**" years\n"**;  
}  
  
  
Head\_dep:: Head\_dep(std::string Name, **int** Age, **int** Work\_exp, std::string Dep\_name):  
Teacher(Name,Age,Work\_exp),dep\_name(Dep\_name){  
 std::cout <<**"Head\_dep construct\n"**;  
}  
Head\_dep:: ~Head\_dep(){  
 std::cout <<**"Head\_dep destruct\n"**;  
}  
**void** Head\_dep:: show() {  
std::cout <<**"Name - "**<< name <<**"\n"**;  
std::cout <<**"Age - "** << age <<**" years\n"**;  
std::cout <<**"Work experience - "**<<work\_exp <<**" years\n"**;  
std::cout <<**"Department name - "**<<dep\_name <<**"\n"**;  
}  
  
  
Student:: Student(std::string Name, **int** Age, **int** Course):  
Person(Name,Age),course(Course){  
 std::cout <<**"Student construct\n"**;  
}  
Student:: ~Student(){  
 std::cout <<**"Student destruct\n"**;  
}  
**void** Student:: show() {  
std::cout <<**"Student name - "**<< name <<**"\n"**;  
std::cout <<**"Age - "** << age <<**" years\n"**;  
std::cout <<**"Course number - "**<<course <<**"\n"**;  
}

**“main.cpp”**

#include **"class.h"**std::list <Person\*> Person:: obj\_list;  
  
**int** main() {  
 Teacher first(**"Irina"**,45,10);  
 Head\_dep second(**"Anna"**,56,30,**"IIT"**);  
 Student third(**"Ivan"**,20,3);  
 Head\_dep fourth(**"Dmitriy"**,33,10,**"FL"**);  
 Person::print();  
}

**Результат работы программы:**

****

****

**Вывод:** Я получил практические навыки создания иерархии классов и использования статических компонентов класса.