Министерство образования Республики Беларусь

УО «Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №2**

По дисциплине: “Языки программирования”

Тема: “Наследование и виртуальные функции”  
**Вариант №6**

**Выполнил**: ПО-7 Рекун И.В.

**Проверила:** Бойко Д.О.

Брест 2021

**Цель:**

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
5. объекты различных классов и помещаются в список, после чего список просматривается.
6. Сделать соответствующие методы не виртуальными и посмотреть, что будет.
7. Реализовать вариант, когда объект добавляется в список при создании, т.е. в конструкторе (смотри пункт 6 следующего раздела).

**Вариант задания:**

Перечень классов: книга, журнал, печатное издание, учебник.

**Код программы:**

**main.cpp**

#include "class.h"

std::list <Printed\_edition\*> Printed\_edition::obj\_list;

int main() {

Journal first("Accept the lab, pls", 1945, 100);

Textbook second("Programming Language", 2020, 30, "OAiP");

Book third("Passing the session", 2021, "Gleb Yakubets");

Textbook fourth("Math for down", 2016, 10, "Mathematic");

Printed\_edition::print();

}

**class.h**

#define LABA\_2\_YAP\_C\_\_\_CLASS\_H

#include <iostream>

#include <string>

#include <list>

class Printed\_edition {

protected:

static std::list <Printed\_edition\*> obj\_list;

std::string name;

int date\_foundation;

public:

Printed\_edition(std::string Name, int Date\_foundation);

void add();

virtual ~Printed\_edition();

int getDate\_foundation();

std::string getName();

void setName(std::string name);

void setDate\_foundation(int date\_foundation);

virtual void show() = 0;

static void print();

};

class Journal : public Printed\_edition {

protected:

int price;

public:

Journal(std::string Name, int Date\_foundation, int Price);

virtual ~Journal();

void show() override;

};

class Textbook :public Journal {

protected:

std::string lesson\_name;

public:

Textbook(std::string Name, int Date\_foundation, int Price, std::string Lesson\_name);

virtual ~Textbook();

void show() override;

};

class Book :public Printed\_edition {

protected:

std::string author;

public:

Book(std::string Name, int Date\_foundation, std::string author);

virtual ~Book();

void show() override;

};

#endif

**classes.cpp**

Printed\_edition::Printed\_edition(std::string Name, int Date\_foundation) :

name(Name), date\_foundation(Date\_foundation)

{

std::cout << "Printed\_edition construct\n";

add();

}

void Printed\_edition::add() {

obj\_list.push\_back(this);

}

Printed\_edition :: ~Printed\_edition() {

std::cout << "Printed\_edition desruct\n";

}

int Printed\_edition::getDate\_foundation() {

return date\_foundation;

}

std::string Printed\_edition::getName() {

return name;

}

void Printed\_edition::setName(std::string name) {

Printed\_edition::name = name;

}

void Printed\_edition::setDate\_foundation(int date\_foundation) {

Printed\_edition::date\_foundation = date\_foundation;

}

void Printed\_edition::print() {

if (!obj\_list.empty()) {

int i = 1;

for (Printed\_edition\* obj : obj\_list) {

std::cout << i << "-st object:\n";

obj->show();

i++;

}

}

}

Journal::Journal(std::string Name, int Date\_foundation, int Price) :

Printed\_edition(Name, Date\_foundation), price(Price) {

std::cout << "Journal construct\n";

}

Journal:: ~Journal() {

std::cout << "Journal destruct\n";

}

void Journal::show() {

std::cout << "Journal name - " << name << "\n";

std::cout << "Date of foundation - " << date\_foundation << " years\n";

std::cout << "Journal price - " << price << " euro\n";

}

Textbook::Textbook(std::string Name, int Date\_foundation, int Price, std::string Lesson\_name) :

Journal(Name, Date\_foundation, Price), lesson\_name(Lesson\_name) {

std::cout << "Textbook construct\n";

}

Textbook:: ~Textbook() {

std::cout << "Textbook destruct\n";

}

void Textbook::show() {

std::cout << "Name - " << name << "\n";

std::cout << "Date of writing - " << date\_foundation << " years\n";

std::cout << "Price - " << price << " euro\n";

std::cout << "Lesson name - " << lesson\_name << "\n";

}

Book::Book(std::string Name, int Date\_foundation, std::string Author):

Printed\_edition(Name, Date\_foundation), author(Author) {

std::cout << "Book construct\n";

}

Book:: ~Book() {

std::cout << "Book destruct\n";

}

void Book::show() {

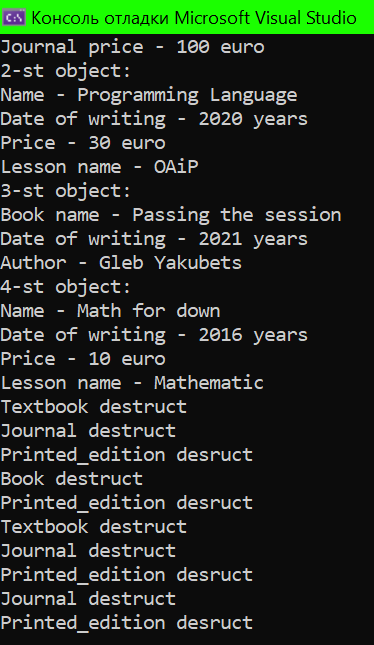
std::cout << "Book name - " << name << "\n";

std::cout << "Date of writing - " << date\_foundation << " years\n";

std::cout << "Author - " << author << "\n";

}

**Результат программы:**

****

**Вывод:**

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