**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Программирование на основе классов и шаблонов»

Отчет по лабораторной работе №7

«Виртуальные функции и классы»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-22Б |  | Преподаватель каф. ИУ5 |
| Поляков Л. С. |  | Бурмистрова М.В. |
| Подпись и дата: |  | Подпись и дата: |
|  |  |  |

Москва, 2024 г.

# Цель работы

Целью лабораторной работы изучение механизмов динамического связывания. Студенты изучают понятия абстрактного класса, виртуальных функций и виртуальных классов. Они осваивают на практике по индивидуальным заданиям использование этих прием при программировании в среде С++.

# Постановка задачи

1. Для определения иерархии классов связать отношением наследования классы, приведенные в приложении (для [заданного варианта](https://iu5edu.ru/wiki/cpp2/docs/labs/lab7/TaskVariants/)). Из перечисленных классов выбрать один, который будет стоять во главе иерархии. Это **абстрактный класс**.
2. Определить в классах **все необходимые конструкторы и деструкторы**. Не забыть в абстрактном классе определить **виртуальный деструктор**.
3. В конструкторах и деструкторах для абстрактного класса и производных классов предусмотреть **вывод отладочных сообщений** об их вызове в консоль. Например:

std::cout << "A()" << std::endl; *// при вызове конструктора*  
std::cout << "~A()" << std::endl; *// при вызове деструктора*

1. **Компонентные данные класса** специфицировать как protected.
2. В абстрактном классе объявить виртуальный метод show, который отвечает за **отображение данных класса**. В каждом производном классе реализовать данный метод с учетом специфики каждого класса.
3. Определение классов, их реализацию, демонстрационную программу поместить в отдельные файлы. Данные файлы должны быть упакованы в **отдельную статическую библиотеку**.

**Требования:**

* В работе запрещено использование контейнеров STL (vector, list и др.).

**Вариант 6**

журнал, книга, печатное издание, учебник

# Разработка алгоритма

## Структура проекта

.

├── CMakeLists.txt

├── PrintEdition

│   ├── Book.cpp

│   ├── Book.hpp

│   ├── CMakeLists.txt

│   ├── Journal.cpp

│   ├── Journal.hpp

│   ├── PrintEdition.cpp

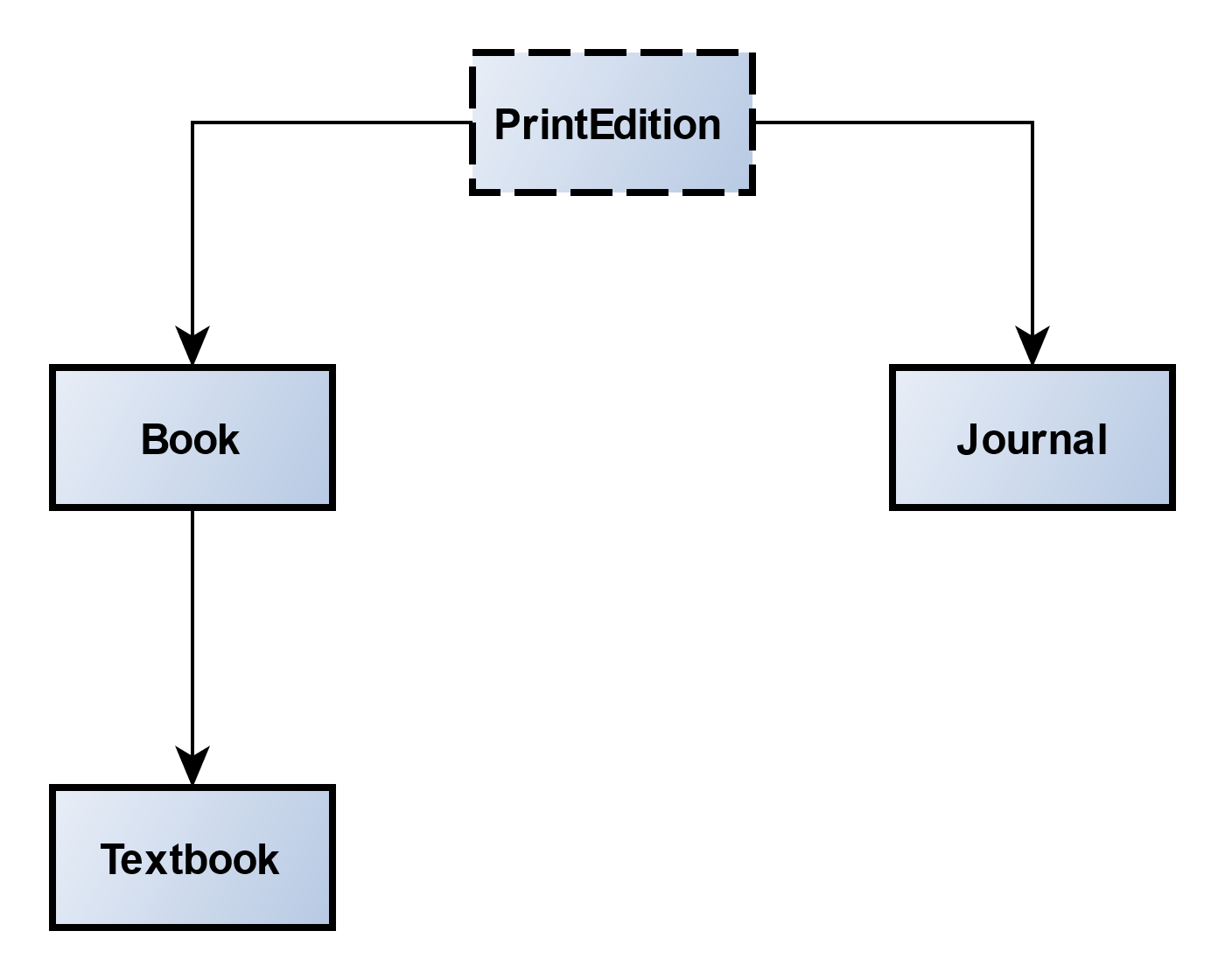
│   ├── PrintEdition.hpp

│   ├── Textbook.cpp

│   └── Textbook.hpp

└── main.cpp

**Иерархия классов**



## Текст программы

**PrintEdition.hpp**

#ifndef LAB\_7\_PRINTEDITION\_HPP  
#define LAB\_7\_PRINTEDITION\_HPP  
  
#include "../../Lab 5.2/MyContainers/MyVector.hpp"  
  
class PrintEdition {  
protected:  
 char\* title{};  
 static MyVector<PrintEdition \*> edits;  
 void setTitle(const char\*);  
public:  
 PrintEdition();  
 PrintEdition(const PrintEdition&);  
 explicit PrintEdition(const char\*);  
 void addEdition(PrintEdition\*);  
 virtual ~PrintEdition();  
 virtual void print() const = 0;  
 static void printArray();  
};  
  
  
#endif //LAB\_7\_PRINTEDITION\_HPP

**PrintEdition.cpp**

#include "PrintEdition.hpp"  
  
  
MyVector<PrintEdition\*> PrintEdition::edits;  
  
PrintEdition::PrintEdition() : title(const\_cast<char \*>("")){}  
  
PrintEdition::PrintEdition(const PrintEdition& edition) : title(edition.title) { edits.addElement(this); }  
  
PrintEdition::PrintEdition(const char\* \_title) {  
 setTitle(\_title);  
 addEdition(this);  
}  
  
PrintEdition::~PrintEdition() = default;  
  
void PrintEdition::setTitle(const char\* newTitle) {  
 auto size = std::strlen(newTitle) + 1;  
 this->title = new char[size];  
 std::strncpy(this->title, newTitle, size);  
}  
  
void PrintEdition::addEdition(PrintEdition \*edition) { // static  
 edits.addElement(edition);  
}  
  
void PrintEdition::print() const {  
 std::cout << "Печатное издание: " << this->title << std::endl;  
}  
  
void PrintEdition::printArray() {  
 for (int i = 0; i < edits.getSize(); i++) {  
 edits[i]->print();  
 }  
}

**Book.hpp**

#ifndef LAB\_7\_BOOK\_HPP  
#define LAB\_7\_BOOK\_HPP  
  
#include "PrintEdition.hpp"  
  
class Book : public PrintEdition {  
protected:  
 char\* author{};  
 void setAuthor(const char \*);  
public:  
 Book();  
 Book(const Book& copy);  
 explicit Book(const char\*, const char\*);  
 void print() const override;  
};  
  
  
#endif //LAB\_7\_BOOK\_HPP

**Book.cpp**

#include "Book.hpp"  
  
  
Book::Book() : PrintEdition::PrintEdition(), author(const\_cast<char \*>(""))  
{  
  
}  
  
Book::Book(const char\* \_title, const char\* \_author) : PrintEdition::PrintEdition(\_title)  
{  
 setAuthor(\_author);  
}  
  
Book::Book(const Book &copy) : PrintEdition::PrintEdition(copy), author(copy.author)  
{  
  
}  
  
void Book::setAuthor(const char\* newAuthor) {  
 auto size = std::strlen(newAuthor) + 1;  
 this->author = new char[size];  
 std::strncpy(this->author, newAuthor, size);  
}  
  
void Book::print() const {  
 std::cout << "Книга: " << this->title  
 << " Автор: " << this->author  
 << std::endl;  
}

**Textbook.hpp**

#ifndef LAB\_7\_TEXTBOOK\_HPP  
#define LAB\_7\_TEXTBOOK\_HPP  
  
  
#include "Book.hpp"  
  
  
class Textbook : public Book {  
protected:  
 int publicationYear{};  
public:  
 Textbook();  
 Textbook(const Textbook& copy);  
 explicit Textbook(const char\*, const char\*, int);  
 void print() const override;  
};  
  
  
#endif //LAB\_7\_TEXTBOOK\_HPP

**Textbook.cpp**

#include "Textbook.hpp"  
  
Textbook::Textbook() : Book::Book(), publicationYear(0)  
{  
  
}  
  
Textbook::Textbook(const Textbook &copy) : Book::Book(copy), publicationYear(copy.publicationYear)  
{  
  
}  
  
Textbook::Textbook(const char \*\_title, const char \*\_author, int \_publicationYear) : Book::Book(\_title, \_author),  
 publicationYear(\_publicationYear){}  
  
void Textbook::print() const {  
 std::cout << "Книга: " << this->title  
 << " Автор: " << this->author  
 << " Год издания: " << this->publicationYear  
 << std::endl;  
}

**Journal.hpp**

#ifndef LAB\_7\_JOURNAL\_HPP  
#define LAB\_7\_JOURNAL\_HPP  
  
  
#include "PrintEdition.hpp"  
  
class Journal : virtual public PrintEdition {  
protected:  
 double price{};  
public:  
 Journal() : PrintEdition::PrintEdition(), price(0) {};  
 Journal(const Journal& copy);  
 explicit Journal(const char\*, double);  
 void print() const override;  
};  
  
  
#endif //LAB\_7\_JOURNAL\_HPP

**Journal.cpp**

#include "Journal.hpp"  
  
Journal::Journal(const char\* \_title, double \_price) : PrintEdition::PrintEdition(\_title), price(\_price)  
{  
  
}  
  
Journal::Journal(const Journal &copy) : PrintEdition::PrintEdition(copy), price(copy.price)  
{  
  
}  
  
void Journal::print() const {  
 std::cout << "Журнал: " << this->title << " Цена: " << this->price << std::endl;  
}

**PrintEdition/CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.26)  
  
set(PROJECT\_NAME PrintEdition)  
project(${PROJECT\_NAME})  
  
set(CMAKE\_CXX\_STANDARD 20)  
  
set(${PROJECT\_NAME}\_HEADERS  
 PrintEdition.hpp  
 Book.hpp  
 Journal.hpp  
 Textbook.hpp  
)  
  
set(${PROJECT\_NAME}\_SOURCES  
 PrintEdition.cpp  
 Book.cpp  
 Journal.cpp  
 Textbook.cpp  
)  
  
set(${PROJECT\_NAME}\_SOURCE\_LIST  
 ${${PROJECT\_NAME}\_SOURCES}  
 ${${PROJECT\_NAME}\_HEADERS}  
)  
  
add\_library(${PROJECT\_NAME} STATIC ${${PROJECT\_NAME}\_SOURCE\_LIST})

**main.cpp**

#include <iostream>  
#include "PrintEdition/Textbook.hpp"  
#include "PrintEdition/Journal.hpp"  
  
int main() {  
 setlocale(LC\_ALL, "russian");  
  
 Journal journal("playboy", 1337);  
 Book book("Drei Kameraden", "Remarque");  
 Textbook textbook("Anti Demidovich", "Demidovich", 1954);  
  
 PrintEdition::printArray();  
 return 0;  
}

**CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.26)  
  
set(PROJECT\_NAME Lab\_7)  
project(${PROJECT\_NAME})  
  
set(CMAKE\_CXX\_STANDARD 20)  
  
set(${PROJECT\_NAME}\_HEADERS  
  
)  
  
set(${PROJECT\_NAME}\_SOURCES  
 main.cpp  
)  
  
add\_subdirectory(PrintEdition)  
  
set(${PROJECT\_NAME}\_SOURCE\_LIST  
 ${${PROJECT\_NAME}\_SOURCES}  
 ${${PROJECT\_NAME}\_HEADERS}  
)  
  
add\_executable(${PROJECT\_NAME} ${${PROJECT\_NAME}\_SOURCE\_LIST})  
  
target\_link\_libraries(${PROJECT\_NAME} PrintEdition)

# Анализ Результатов

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

# Вывод

Я научился

* Применять механизмы динамического связывания на практике
* Строить иерархию классов
* Работать со статическими библиотеками с использованием Cmake

Познакомился с такими понятиями как абстрактный класс, виртуальная функция и виртуальный класс, изучил базовые принципы работы с ними и ситуации в которых стоит их использовать.