深 圳 大 学 实 验 报 告

课程名称：面向对象程序设计

实验项目名称：实验七 继承与派生

学院：医学院

专业：生物医学工程

指导教师：李乔亮、邓云

报告人：陈焕鑫　学号：**2016222042**　班级：生工 **2** 班

实验时间：**2018.12.5**

实验报告提交时间：**2018.12.17**

教务部制

实验目的：

熟练掌握 C++中类的继承与派生功能的使用

实验内容：

1. 利用类的继承特性，对漫画书、英语教材、诗集进行抽象，设计合理的基类，要求书名根据用户输入自动分配空间，且名字可更改。

   要求：

   1） 各派生类在基类的基础上，至少新增一种属性；

   2） 采用 public 公有继承，使用标准格式完成类的设计；

   3） 给派生类与基类设计合理的构造，析构函数，并在函数中打印相关信息；

   4） 在主程序中定义多个对象（多本漫画书、英语教材、诗集），分析对象构造与析构的过程。

2. 在题目 1 类的基础上，自动计算所定义的书本的数量（要求不同类型的书籍分开计数）。

实验环境与程序代码：

实验环境：win10 系统下的 Visual Studio 2017

程序代码如下所示：

本程序总共包括 main.cpp, book.h, book.cpp, English.h, English.cpp, poem.h, poem.cpp, comic.h comic.cpp 九个文件。book.h, book.cpp 文件对中定义了书的基类，在该基类内部有变量：书名，价格以及书的数目（静态类型），因为书名和价格是对于书来说比较重要的属性，经常要被继承类所调用，所以用作保护权限，使得继承类可以方便地使用自己的书名和价格。书的数目在每次调用构造函数的时候就会递增，调用析构函数的时候就会递减。而且由于它是静态的变量，所以对于所有属于该类的对象来说，书的数目是公有的变量，类中还提供了更改书名的函数 changeBookName，调用该函数，需要用户输入新的书名，系统根据用户的输入重新自动分配新的存储空间。在 English 文件对中，CEnglish 类在 CBook 类的基础上添加了难度等级的变量 complexity，在 Poem 文件对中，CPoem 类在 CBook 类的基础上添加了作者名称的变量 author，在 comic 文件对中，CComic 类在 CBook 类的基础上添加了流行程度的变量 popularity。

```
//book.h
#ifndef _BOOK_H_
#define _BOOK_H_

#include <iostream>
using namespace std;

class CBook
{
private:
  static int book_num;
protected:
  char *book_name;
  float price;
public:
  CBook();
  CBook(const char *i_name, float i_price);
  CBook(const CBook &copy_c);
  ~CBook();
  void changeBookName(void);
  void printInfo(void);
  static void printAllNum(void);
};

#endif



//book.cpp
#include "book.h"
```

```cpp
int CBook::book_num = 0;


CBook::CBook()
{
  book_name = NULL;
  price = 0;
  book_num++;
  cout << "Base nil para construct! " << endl;
}

CBook::CBook(const char *i_name, float i_price)
{
  book_name = new char[strlen(i_name) + 1];
  if (book_name != NULL)
  {
    strcpy(book_name, i_name);
  }
  price = i_price;
  book_num++;
  cout << "Base para construct "  << book_name << endl;
}

CBook::CBook(const CBook &copy_c)
{
  book_name = new char[strlen(copy_c.book_name) + 1];
  if (book_name != NULL)
  {
    strcpy(book_name, copy_c.book_name);
  }
  price = copy_c.price;
  book_num++;
  cout << "Base copy construct " << book_name << endl;
}

CBook::~CBook()
{
  cout << "Deconstruct base:" << book_name << endl;
  if(book_name != NULL)
  {
    delete[]book_name;
  }
  book_num--;
  cout << "Base deconstruct! " << endl;
}

void CBook::changeBookName(void)
{
  char name_srting[20];
  cout << "书的原名为: " << book_name << endl;
  cout << "请输入新的书名: ";
  cin >> name_srting;
  delete[]book_name;
  book_name = new char[strlen(name_srting) + 1];
  if (book_name != NULL)
  {
    strcpy(book_name, name_srting);
  }
  cout << "新的数目情况为: " << endl;
  printInfo();
}

void CBook::printInfo(void)
{
  cout << "-------------------------------------" << endl;
  cout << "书名: " << book_name << endl;
  cout << "价格: " << price << endl;
  cout << "-------------------------------------" << endl;
}

void CBook::printAllNum(void)
{
  cout << "书的总数目为: " << book_num << endl;
}

//comic.h
#ifndef _COMIC_H_
#define _COMIC_H_

#include "book.h"

typedef enum
{
  UNSOUGHT,
  COMMON,
  HOT
}Popularity_grade;
```

```
class CComic :public CBook
{
private:
  static int comic_num;
  int popularity;
public:
  CComic();
  CComic(const char *i_name, float i_price, int i_pop);
  CComic(const CComic &copy_c);
  ~CComic();
  void printComicNum(void);
};

#endif

//comic.cpp
#include "comic.h"

int CComic::comic_num = 0;

CComic::CComic() :CBook()
{
  popularity = UNSOUGHT;
  comic_num++;
  cout << "Comic nil para construct!" << endl;
}

CComic::CComic(const char *i_name, float i_price, int i_pop) :CBook(i_name, i_price)
{
  popularity = i_pop;
  comic_num++;
  cout << "Comic para construct " << book_name << endl;
}

CComic::CComic(const CComic &copy_c) :CBook(copy_c)
{
  popularity = copy_c.popularity;
  comic_num++;
  cout << "Comic copy construct " << book_name << endl;
}

CComic::~CComic()
{
```

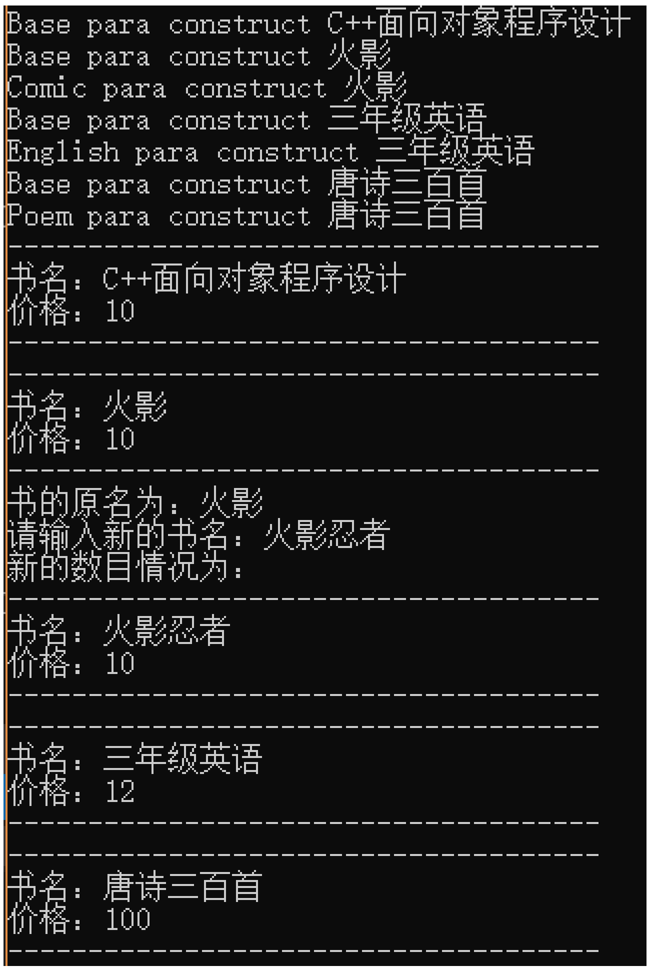
```cpp
  cout << "Comic deconstruct: " << book_name << endl;
  comic_num--;
}

void CComic::printComicNum(void)
{
  cout << "Comic book amount: " << comic_num << endl;
}

//English.h
#ifndef _ENGLISH_H_
#define _ENGLISH_H_

#include "book.h"

typedef enum
{
  EASY,
  MEDIUM,
  DIFFICULT
}Complexity_grade;

class CEnglish :public CBook
{
private:
  static int English_num;
  int complexity;
public:
  CEnglish();
  CEnglish(const char *i_name, float i_price, int level);
  CEnglish(const CEnglish &copy_c);
  ~CEnglish();
  void printEnglishNum(void);
};

#endif

//English.cpp
#include "English.h"

int CEnglish::English_num = 0;

CEnglish::CEnglish() :CBook()
{
  complexity = EASY;
  English_num++;
  cout << "English nil para construct!" << endl;
}

CEnglish::CEnglish(const CEnglish &copy_c) : CBook(copy_c)
{
  complexity = copy_c.complexity;
  English_num++;
  cout << "English copy construct " << book_name << endl;
}

CEnglish::CEnglish(const char *i_name, float i_price, int level) : CBook(i_name,
i_price)
{
  complexity = level;
  English_num++;
  cout << "English para construct " << book_name << endl;
}

CEnglish::~CEnglish()
{
  cout << "English deconstruct: " << book_name << endl;
  English_num--;
}

void CEnglish::printEnglishNum(void)
{
  cout << "English book amount: " << English_num << endl;
}

//poem.h
#ifndef _POEM_H_
#define _POEM_H_

#include "book.h"

class CPoem :public CBook
{
private:
  static int poem_num;
  char *author;
public:
  CPoem();
```

```cpp
  CPoem(const char *i_name, float i_price, const char *i_author);
  CPoem(const CPoem &copy_c);
  ~CPoem();
  void printPoemNum(void);
};

#endif

//poem.cpp
#include "poem.h"

int CPoem::poem_num = 0;

CPoem::CPoem() :CBook()
{
  author = NULL;
  poem_num++;
  cout << "Poem nil para construct!" << endl;
}

CPoem::CPoem(const char *i_name, float i_price, const char *i_author) :CBook(i_name,
i_price)
{
  author = new char[strlen(i_author) + 1];
  if (author != NULL)
  {
    strcpy(author, i_author);
  }
  poem_num++;
  cout << "Poem para construct " << book_name << endl;
}

CPoem::CPoem(const CPoem &copy_c)  :CBook(copy_c)
{
  author = new char[strlen(copy_c.author) + 1];
  if (author != NULL)
  {
    strcpy(author, copy_c.author);
  }
  poem_num++;
  cout << "Poem copy construct " << book_name << endl;
}

CPoem::~CPoem()
```

```
{
  cout << "Poem deconstruct: " << book_name << endl;
  poem_num--;
}

void CPoem::printPoemNum(void)
{
  cout << "Poem book amount: " << poem_num << endl;
}

//main.cpp
#include "book.h"
#include "comic.h"
#include "English.h"
#include "poem.h"

int main()
{
  CBook a("C++面向对象程序设计", 10.0);
  CComic b("火影", 10.0, HOT);
  CEnglish c("三年级英语", 12.0, EASY);
  CPoem d("唐诗三百首", 100.0, "唐代众多诗人");
  a.printInfo();
  b.printInfo();
  b.changeBookName();
  c.printInfo();
  d.printInfo();
  d.printPoemNum();
  c.printEnglishNum();
  b.printComicNum();
  CBook::printAllNum();

  return 0;
}
```

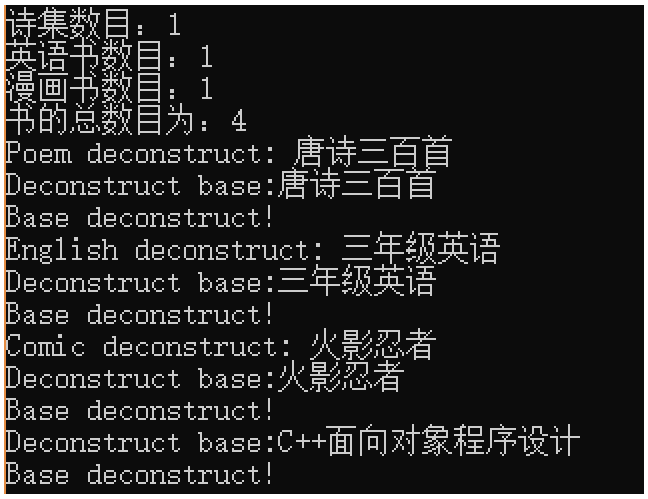
实验结果与分析：

程序运行结果如图 1-1 和图 1-2 所示：



```
Base para construct C++面向对象程序设计
Base para construct 火影
Comic para construct 火影
Base para construct 三年级英语
English para construct 三年级英语
Base para construct 唐诗三百首
Poem para construct 唐诗三百首
-------------------------------------
书名：C++面向对象程序设计
价格：10
-------------------------------------
-------------------------------------
书名：火影
价格：10
-------------------------------------
书的原名为：火影
请输入新的书名：火影忍者
新的数目情况为：
-------------------------------------
书名：火影忍者
价格：10
-------------------------------------
-------------------------------------
书名：三年级英语
价格：12
-------------------------------------
-------------------------------------
书名：唐诗三百首
价格：100
-------------------------------------
```

图 1-1 程序运行结果（1）



```
诗集数目：1
英语书数目：1
漫画书数目：1
书的总数目为：4
Poem deconstruct: 唐诗三百首
Deconstruct base:唐诗三百首
Base deconstruct!
English deconstruct: 三年级英语
Deconstruct base:三年级英语
Base deconstruct!
Comic deconstruct: 火影忍者
Deconstruct base:火影忍者
Base deconstruct!
Deconstruct base:C++面向对象程序设计
Base deconstruct!
```

图 1-2 程序运行结果（2）

由结果可以看出，在构造派生类对象的过程中，首先调用基类的构造函数，然后才是派生类自己的构造函数，而且是根据程序的顺序有上往下一个一个的构造。在析构派生类的时候：（1）后生成的对象首先析构；（2）先调用派生类的析构函数，然后才是调用基类的析构函数。

指导教师批阅意见： 成绩评定： 指导教师签字： 年　　月　　日
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。