**暨南大学本科实验报告专用纸**

课程名称 C++程序设计 成绩评定

实验项目名称 类编程实验 指导教师 王勇杰

实验项目编号 11 实验项目类型 编程型 实验地点 C105

学生姓名 李炜鹏 学号 2017052544

学院 智能科学与工程学院 系 计算机 专业 信息安全

实验时间2019年12月10日 午～12月13日 午 温度 ℃湿度

1. **实验目的**
2. 认识并理解C++的OOP编程思想
3. 理解并上机实践C++的OOP编程以及OOP的继承
4. 实验要求：阅读课本第15章的内容， 试调试运行实验十一中的源代码
5. 实验报告要求：
   1. 实验目的
   2. 实验原理
   3. 注释必要的代码
   4. 实验结果的截图
   5. 实验评估
6. **实验原理**

本实验涉及的知识点为面向对象编程的相关知识点，总结如下：

1. 类的继承：
   1. 单继承、多重继承：一个派生类只从一个基类派生，称单继承；一个派生类有两个对多个基类，称多重继承。
   2. 基类、派生类：派生类是基类的具体化，而基类是派生类的抽象。
2. 运算符函数的重载：本实验涉及的是流插入运算符和流提取运算符：

I. 用户自己定义的类型的数据，是不能直接用"<<"">>"进行输入输出的，若想用，必须重载

II. 不能定义为成员函数，因为若为成员函数，则第一个参数必须有该类的对象，而这里只能是istream或者ostream，所以应该定义为友元函数，第一个参数为istream &或者ostream &，第二个参数为类的引用

1. 虚析构函数：基类通常都应该定义一个虚析构函数，及时该函数不不执行任何实际操作也是如此
2. **主要仪器设备**

**仪器：**计算机

**实验环境：**Visual Studio 2017或Dev-C++

1. **源程序**

写出程序的源程序。并且注释代码中每条语句

**Query.h**

#ifndef QUOTE\_H

#define QUOTE\_H

#include <memory>

#include <iostream>

#include <string>

#include <cstddef>

//以原价售出的商品

//派生类将定义各种折扣策略

class Quote {

//定义两个友元函数：重载流插入运算符和流提取运算符

//用户自己定义的类型的数据，是不能直接用"<<"">>"进行输入输出的，若想用，必须重载

//不能定义为成员函数，因为若为成员函数，则第一个参数必须有该类的对象，而这里只能是istream或者ostream

friend std::istream& operator>>(std::istream&, Quote&);

friend std::ostream& operator<<(std::ostream&, const Quote&);

public:

Quote() : price(0.0) { }//默认构造函数

Quote(const std::string &book, double sales\_price) :bookNo(book), price(sales\_price) { }//含参数的构造函数

//如果删除了指向派生对象的基本指针，则需要虚析构函数

virtual ~Quote() { } //析构函数的动态绑定

std::string isbn() const { return bookNo; }//返回书的ISBN号

//返回指定数量的商品的总销售价格派生类将覆盖并应用不同的折扣算法

//派生类将覆盖并应用不同的折扣算法

virtual double net\_price(std::size\_t n) const

{

return n \* price;

}

//虚函数返回自身的动态分配副本

virtual Quote\* clone() const { return new Quote(\*this); }

private:

std::string bookNo; //书的ISBN号

protected:

double price; //原价

};

//抽象基类：用于保存折现率和数量

//派生类将使用这些数据实施定价策略

class Disc\_quote : public Quote {

public:

Disc\_quote() : quantity(0), discount(0.0) { }//默认构造函数

Disc\_quote(const std::string& book, double price, std::size\_t qty, double disc) : Quote(book, price), quantity(qty), discount(disc) { }//含参数的构造函数

double net\_price(std::size\_t) const = 0;//？

std::pair<size\_t, double> discount\_policy() const

{

return std::make\_pair(quantity, discount);

}

protected:

std::size\_t quantity; //数量

double discount; //折扣

};

//当售出指定数量的同一本书时，折扣就会生效

//折扣表示为降低正常价格所用的分数

class Bulk\_quote : public Disc\_quote { // Bulk\_quote继承自Disc\_quote

public:

Bulk\_quote() { }//默认构造函数

Bulk\_quote(const std::string& book, double p, std::size\_t qty, double disc) : Disc\_quote(book, p, qty, disc) { }//含参数的构造函数

//重载基类成员函数，以实施批量购买折扣政策

double net\_price(std::size\_t) const;

//复制以生成副本

Bulk\_quote\* clone() const { return new Bulk\_quote(\*this); }

};

//仅指定数量的副本享有折扣（折扣价的一部分）

//以标准价格出售的其他副本

class Lim\_quote : public Disc\_quote {

public:

Lim\_quote(const std::string& book = "", double sales\_price = 0.0, std::size\_t qty = 0, double disc\_rate = 0.0) :Disc\_quote(book, sales\_price, qty, disc\_rate) { }//含参数的构造函数

//重载基类成员函数，以实施有限的折扣政策

double net\_price(std::size\_t) const;

//复制以生成副本

Lim\_quote\* clone() const { return new Lim\_quote(\*this); }

};

double print\_total(std::ostream &, const Quote&, std::size\_t);

#endif

**Quote.cpp**

#include "Quote.h"

#include <algorithm>

using std::min;

#include <cstddef>

using std::size\_t;

#include <iostream>

using std::ostream; using std::endl;

using std::cout;

//计算并打印给定份数的价格，并应用折扣类

double print\_total(ostream &os, const Quote &item, size\_t n)

{

//调用Quote::net\_price或Bulk\_quote::net\_price，取决于绑定到item参数的对象的类型

double ret = item.net\_price(n);

os << "ISBN: " << item.isbn() << " # sold: " << n << " total due: " << ret << endl;

return ret;

}

//如果购买了指定数量的物品，使用折扣价

double Bulk\_quote::net\_price(size\_t cnt) const

{

if (cnt >= quantity)

return cnt \* (1 - discount) \* price;

else

return cnt \* price;

}

//打折的以折扣价卖，以正常折扣价定价的其他商品

double Lim\_quote::net\_price(size\_t cnt) const

{

size\_t discounted = min(cnt, quantity);//打折的数量

size\_t undiscounted = cnt - discounted;//不打折的数量

return discounted \* (1 - discount) \* price + undiscounted \* price;

}

**Basket.h**

#ifndef BASKET\_H

#define BASKET\_H

#include <iostream>

#include <string>

#include <set>

#include <map>

#include <utility>

#include <cstddef>

#include <stdexcept>

#include <memory>

#include "Quote.h"

//购物车

class Basket {

public:

// Basket uses synthesized copy-control members（Basket使用合成的复制控制成员？）

//因为在C ++ 11之前没有类内初始化程序

//我们需要定义一个默认的构造函数来初始化项目

Basket() : items(compare) { }

void add\_item(const std::shared\_ptr<Quote> &sale)

{

items.insert(sale);

}

void add\_item(const Quote& sale) //复制给定的对象

{

items.insert(std::shared\_ptr<Quote>(sale.clone()));

}

//打印每本书的总价

//以及购物车中所有书的总和

double total\_receipt(std::ostream&) const;

//为了debug，打印购物车中的内容

void display(std::ostream&) const;

private:

//比较shared\_ptrs

static bool compare(const std::shared\_ptr<Quote> &lhs, const std::shared\_ptr<Quote> &rhs)

{

return lhs->isbn() < rhs->isbn();

}

//由比较成员排序

typedef bool(\*Comp)(const std::shared\_ptr<Quote> &, const std::shared\_ptr<Quote> &);

std::multiset<std::shared\_ptr<Quote>, Comp> items;

};

#endif

**Basket.cpp**

#include "Quote.h"

#include "Basket.h"

#include <cstddef>

using std::size\_t;

#include <set>

using std::multiset;

#include <string>

using std::string;

#include <iostream>

using std::ostream; using std::endl;

using std::cout;

#include <memory>

using std::shared\_ptr;

//显示Basket里面的内容

void Basket::display(ostream &os) const

{

os << "Basket size: " << items.size() << endl;

// 在购物篮中打印每个不同的ISBN以及

// 订购的数量及其价格将是多少

// upper\_bound将迭代器返回到集合中的下一个元素

for (auto next\_item = items.begin(); next\_item != items.end(); next\_item = items.upper\_bound(\*next\_item))

{

// 购物篮中至少有一个与此键相关的元素

os << (\*next\_item)->isbn() << " occurs " //IBSN

<< items.count(\*next\_item) << " times" //数量

<< " for a price of "

<< (\*next\_item)->net\_price(items.count(\*next\_item))//价格

<< endl;

}

}

double Basket::total\_receipt(ostream &os) const

{

double sum = 0.0; //总价

// iter指向的是一批具有相同ISBN的元素中的第一个元素

// upper\_bound返回一个迭代器，该元素刚好在该批处理结束之后

for (auto iter = items.begin(); iter != items.end(); iter = items.upper\_bound(\*iter)) {

// 购物篮中至少有一个与此键相关的元素

// 打印这本书的订单项

sum += print\_total(os, \*\*iter, items.count(\*iter));

}

os << "Total Sale: " << sum << endl; //打印最终的总价

return sum;

}

**Basket\_main.cpp**

#include <memory>

using std::shared\_ptr;

#include "Basket.h"

#include <iostream>

using std::cout; using std::endl;

int main()

{

Basket sale;

sale.add\_item(shared\_ptr<Quote>(new Quote("123", 45)));

sale.add\_item(shared\_ptr<Quote>(new Quote("123", 45)));

sale.add\_item(shared\_ptr<Quote>(new Quote("123", 45)));

sale.add\_item(shared\_ptr<Quote>(new Bulk\_quote("345", 45, 3, .15)));

sale.add\_item(shared\_ptr<Quote>(new Bulk\_quote("345", 45, 3, .15)));

sale.add\_item(shared\_ptr<Quote>(new Bulk\_quote("345", 45, 3, .15)));

sale.add\_item(shared\_ptr<Quote>(new Bulk\_quote("345", 45, 3, .15)));

sale.add\_item(shared\_ptr<Quote>(new Bulk\_quote("345", 45, 3, .15)));

sale.add\_item(shared\_ptr<Quote>(new Bulk\_quote("345", 45, 3, .15)));

sale.add\_item(shared\_ptr<Quote>(new Bulk\_quote("345", 45, 3, .15)));

sale.add\_item(shared\_ptr<Quote>(new Bulk\_quote("345", 45, 3, .15)));

sale.add\_item(shared\_ptr<Quote>(new Bulk\_quote("678", 55, 5, .25)));

sale.add\_item(shared\_ptr<Quote>(new Bulk\_quote("678", 55, 5, .25)));

sale.add\_item(shared\_ptr<Quote>(new Bulk\_quote("678", 55, 5, .25)));

sale.add\_item(shared\_ptr<Quote>(new Bulk\_quote("678", 55, 5, .25)));

sale.add\_item(shared\_ptr<Quote>(new Bulk\_quote("678", 55, 5, .25)));

sale.add\_item(shared\_ptr<Quote>(new Bulk\_quote("678", 55, 5, .25)));

sale.add\_item(shared\_ptr<Quote>(new Bulk\_quote("678", 55, 5, .25)));

sale.add\_item(shared\_ptr<Quote>(new Bulk\_quote("678", 55, 5, .25)));

sale.add\_item(shared\_ptr<Quote>(new Lim\_quote("abc", 35, 2, .10)));

sale.add\_item(shared\_ptr<Quote>(new Lim\_quote("abc", 35, 2, .10)));

sale.add\_item(shared\_ptr<Quote>(new Lim\_quote("abc", 35, 2, .10)));

sale.add\_item(shared\_ptr<Quote>(new Lim\_quote("abc", 35, 2, .10)));

sale.add\_item(shared\_ptr<Quote>(new Lim\_quote("abc", 35, 2, .10)));

sale.add\_item(shared\_ptr<Quote>(new Lim\_quote("abc", 35, 2, .10)));

sale.add\_item(shared\_ptr<Quote>(new Quote("def", 35)));

sale.add\_item(shared\_ptr<Quote>(new Quote("def", 35)));

sale.total\_receipt(cout);

Basket bsk;

//参数是ISBN，价格，最小数量和折扣

bsk.add\_item(shared\_ptr<Quote>(new Bulk\_quote("0-201-82470-1", 50, 5, .19)));

bsk.add\_item(shared\_ptr<Quote>(new Bulk\_quote("0-201-82470-1", 50, 5, .19)));

bsk.add\_item(shared\_ptr<Quote>(new Bulk\_quote("0-201-82470-1", 50, 5, .19)));

bsk.add\_item(shared\_ptr<Quote>(new Bulk\_quote("0-201-82470-1", 50, 5, .19)));

bsk.add\_item(shared\_ptr<Quote>(new Bulk\_quote("0-201-82470-1", 50, 5, .19)));

bsk.add\_item(shared\_ptr<Quote>(new Lim\_quote("0-201-54848-8", 35, 2, .10)));

bsk.add\_item(shared\_ptr<Quote>(new Lim\_quote("0-201-54848-8", 35, 2, .10)));

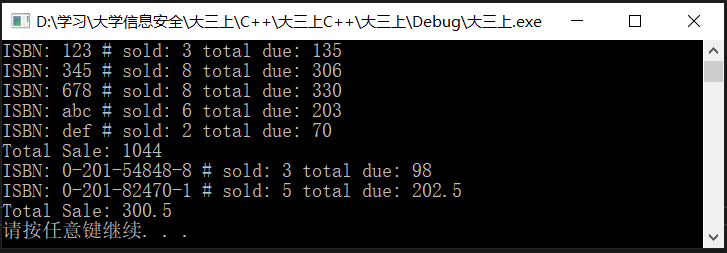
bsk.add\_item(shared\_ptr<Quote>(new Lim\_quote("0-201-54848-8", 35, 2, .10)));

bsk.total\_receipt(cout);

system("pause");

return 0;

}

****

在Query.h里面，首先定义了一个Quote基类（用于描述书的ISBN号、原价等属性和基本功能），然后定义了一个抽象基类Disc\_quote（此类给出了打折的功能但没有具体实现，此类是作为基类让子类继承的），在Disc\_quote类作为基类，定义了两个类Bulk\_quote类和Lim\_quote类，分别使用两种不同的折扣方法进行打折。在Basket.h头文件里面，定义了一个Basket类，实现了加入新物品，计算总价，输出购物车中的内容等功能。在main函数里面通过添加了不同名称、不同数量、不同IBNS号的书，打印其数量、总价等信息，程序运行结果符合预期。

1. **实验评估**

本实验主要考察了C++的面向对象编程，涉及了继承、重载、构造、析构等知识点，强调了面向对象编程的概念以及基本用法，通过本实验，我巩固了C++ 面向对象编程的基本用法，综合提升了C++的基本编程能力，实现了一个实现购物折扣的小算法，是一个比较有意义的实验。