1. **（10分）构造函数和析构函数的主要作用是什么？它们各有什么特性？**

构造函数是特殊的成员函数，对类实例化的时候自动调用相应的构造函数，进行对象的初始化，为对象分配空间。析构函数也是特殊的成员函数，执行的操作与构造函数相反，在对象销毁时释放对象的内存空间等。

* 构造函数特性：构造函数名与类名相同，不能声明返回类型；构造函数可以重载；函数体可以写在类里面或外面；公有构造函数不能像其他成员函数一样被调用，私有构造函数编译器不能调用，无法进行实例化；类中没有构造函数时，编译器会自动生成一个无参数的默认构造函数
* 析构函数特性：析构函数名与构造函数名一样，在前面要加~；无返回值，不能重载；对象销毁时，自动调用析构函数

1. **（10分）简要说明在不同的继承方式下，基类成员在派生类中的访问属性。**

无论派生类是哪种继承方式，基类中的私有成员都不能在派生类中直接访问。

当派生类保护继承基类时，基类中的私有成员不能直接访问，基类中的公有成员在派生类变成保护成员，派生类及派生类子类可以访问，派生类外部不能访问，保护成员仍为保护成员。

当派生类私有继承基类时，基类中的私有成员不能直接访问，基类中的公有成员和保护成员在派生类变为私有成员，在派生类中可以访问，派生类外不能访问。

1. **（10分）简单说明友元运算符重载函数和成员运算符重载函数的区别。**

+ - \* / =等双目运算符，进行友元运算符重载，参数列表中有两个参数，进行成员运算符重载，参数列表中有一个参数，++ --等单目运算符，进行友元运算符重载，参数列表中有一个参数，进行成员运算符重载，参数列表中没有参数。

双目运算符一般可以重载为友元或成员运算符。

1. **（10分）简要说明虚函数的作用，函数重载、覆盖和隐藏的区别。**

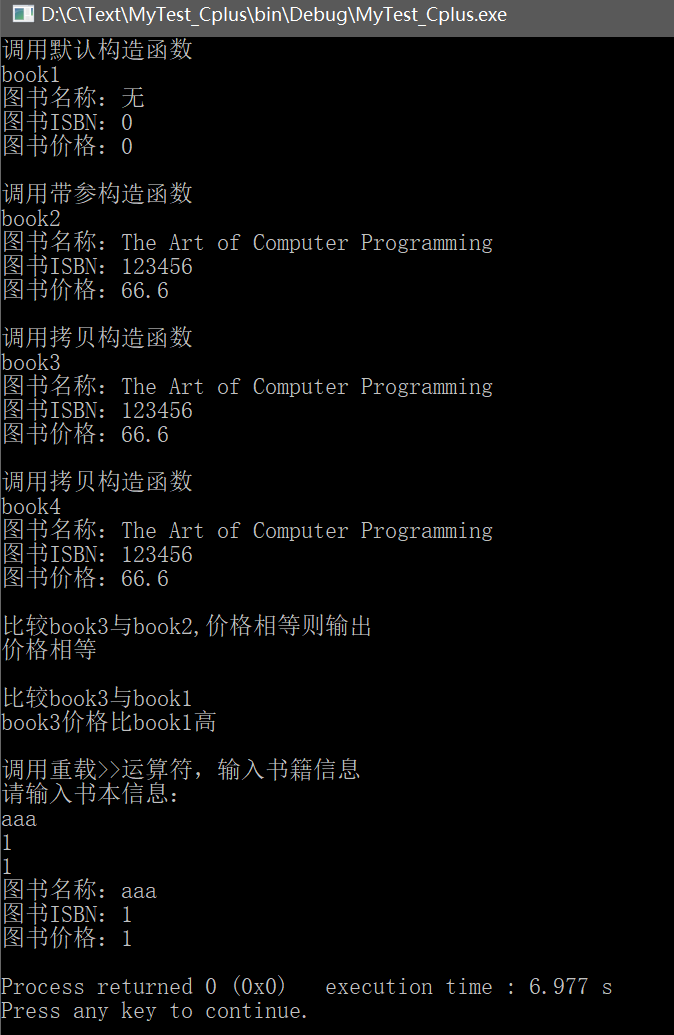
虚函数在基类中定义，前面加上virtual关键字，在派生类中重新定义该函数。虚函数实现了动态多态，即可以根据派生类对象的不同来调用基类或派生类中同名的函数。

函数重载是在同个类中，函数名一样，参数或参数类型必须不同，函数返回值类型也可以不同。

覆盖是在派生类与基类中，函数名与参数一样，基类函数是虚函数。

隐藏是派生类的函数屏蔽了基类中的同名函数，派生类中函数与基类同名，参数不同，则基类函数被隐藏，若参数相同，但基类函数没有virtual，则基类函数也被隐藏

1. **（40分）实现图书类Book，包含私有字段书名name\_(char\*类型)、编号isbn\_(int类型)、价格price\_(float类型)。实现构造函数、析构函数、拷贝构造函数、赋值运算符函数。重载‘>’,‘<’、‘==’运算符（按价格比较）以及输入‘>>’和输出‘<<’运算符。**



#include<iostream>

#include<cstring>

using namespace std;

class Book{

public:

Book()

{

name\_ = new char[strlen("无") + 1];

strcpy(name\_,"无");

isbn\_ = 0;

price\_ = 0;

}

Book(char\* name,int isbn,float price)

{

name\_ = new char[strlen(name) + 1];

strcpy(name\_,name);

isbn\_ = isbn;

price\_ = price;

}

~Book()

{

delete[] name\_;

}

Book(const Book& book)

{

name\_ = new char[strlen(book.name\_)+1];

strcpy(name\_,book.name\_);

isbn\_ = book.isbn\_;

price\_ = book.price\_;

}

Book& operator= (const Book& book)

{

delete[] name\_;

name\_ = new char[strlen(book.name\_)+1];

strcpy(name\_,book.name\_);

isbn\_ = book.isbn\_;

price\_ = book.price\_;

return \*this;

}

friend bool operator> (const Book& book1,const Book& book2)

{

if(book1.price\_>book2.price\_)

return true;

else return false;

}

friend bool operator< (const Book& book1,const Book& book2)

{

if(book1.price\_<book2.price\_)

return true;

else return false;

}

friend bool operator== (const Book& book1,const Book& book2)

{

if(book1.price\_==book2.price\_)

return true;

else return false;

}

friend std::istream& operator>>(std::istream& in,Book& book)

{

in>>book.name\_;

in>>book.isbn\_;

in>>book.price\_;

Book(book.name\_,book.isbn\_,book.price\_);

return in;

}

friend std::ostream& operator<<(std::ostream& os, const Book& book)

{

os<<"图书名称："<<book.name\_<<"\n"<<"图书ISBN："<<book.isbn\_<<

"\n"<<"图书价格："<<book.price\_<<endl;

return os;

}

private:

char\* name\_;

int isbn\_;

float price\_;

};

int main()

{

cout<<"调用默认构造函数"<<endl;

Book book1;

cout<<"book1\n"<<book1;

cout<<"\n调用带参构造函数"<<endl;

Book book2("The Art of Computer Programming",123456,66.6);

cout<<"book2\n"<<book2;

cout<<"\n调用拷贝构造函数"<<endl;

Book book3(book2);

cout<<"book3\n"<<book3;

cout<<"\n调用拷贝构造函数"<<endl;

Book book4 = book3;

cout<<"book4\n"<<book4;

cout<<"\n比较book3与book2,价格相等则输出"<<endl;

if(book3==book4)cout<<"价格相等"<<endl;

cout<<"\n比较book3与book1"<<endl;

if(book3>book1)cout<<"book3价格比book1高"<<endl;

else cout<<"book1价格比book3高"<<endl;

cout<<"\n调用重载>>运算符，输入书籍信息"<<endl;

cout<<"请输入书本信息："<<endl;

Book book5;

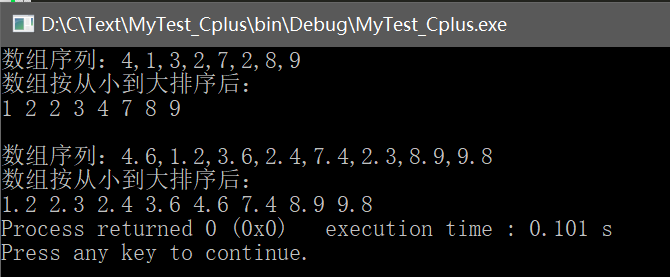
cin>>book5;

cout<<book5;

return 0;

}

1. **（20分）设计函数模板对一维数组排序(选择任意一种排序算法)。**



#include<iostream>

using namespace std;

template <class T>

void sortArray(T \*arr,int n)

{

for(int i=0;i<n-1;i++)

{

for(int j=0;j<n-i-1;j++)

{

if(arr[j]>arr[j+1])

{

T temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

}

int main()

{

cout<<"数组序列：4,1,3,2,7,2,8,9"<<endl;

int arr[] = {4,1,3,2,7,2,8,9};

sortArray(arr,8);

cout<<"数组按从小到大排序后："<<endl;

for(int i=0;i<8;i++)

{

cout<<arr[i]<<" ";

}

cout<<"\n\n数组序列：4.6,1.2,3.6,2.4,7.4,2.3,8.9,9.8"<<endl;

double arr2[] = {4.6,1.2,3.6,2.4,7.4,2.3,8.9,9.8};

sortArray(arr2,8);

cout<<"数组按从小到大排序后："<<endl;

for(int i=0;i<8;i++)

{

cout<<arr2[i]<<" ";

}

return 0;

}