# 实验报告

### 实验一：Employee

代码：

#include<iostream>

#include<iomanip>

#include<string>

using namespace std;

class Date {

public:

/\* 默认构造函数，以fullyear的形式给出年月日，默认值为1990年1月1日，同时设置日期分隔符为“-” \*/

Date(int year = 1990, int month = 1, int day = 1) {

this->year = year;

this->day = day;

this->month = month;

}

/\* get、set方法 \*/

// 设置日期，如果有非法的月或日，将其置为1

void setDate(int year, int month, int day) {

this->year = year;

this->day = day;

this->month = month;

}

void setYear(int year) {

this->year = year;

}

int getYear() {

return this->year;

}

void setMonth(int month) {

this->month = month;

}

int getMonth() {

return this->month;

}

void setDay(int day) {

this->day = day;

}

int getDay() {

return this->day;

}

void setSeparator(char separator) {

this->separator = separator;

}

/\* 输出函数，请使用setfill(‘0’)和setw(2)，需要包含<iomanip>头文件 \*/

void printFullYear() {

cout << this->year << this->separator << setfill('0') << setw(2) << this->month << this->separator << setfill('0') << setw(2) << this->day << endl;

} // 以YYYY-MM-DD的形式打印，2011-01-08

void printStandardYear() {

cout << this->year % 100 << this->separator << setfill('0') << setw(2) << this->month << this->separator << setfill('0') << setw(2) << this->day << endl;

} // 以YY-MM-DD的形式打印，比如11-01-08

/\* 计算函数 \*/

// 计算当前日期与参数日期之间相差几个整年，仅考虑参数日期比当前日期晚的情况

int fullYearsTo(Date& date) {

int wholeYear = date.year - this->year;

if (this->month < date.month) {

//cout << wholeYear;

}

else

{

wholeYear--;

//cout << wholeYear;

}

return wholeYear;

cout << " 满" << wholeYear << "岁了" << endl;

};

/\* 计算当前日期与参数日期之间相差多少天(考虑闰年)，如果参数日期在当前日期之前，返回负数。 \*/

int daysTo(Date& date) {

int run = 0;

int days = 0;

if (date.year < this->year)

{

for (int i = date.year; i <= this->year; i++)

{

if (i % 400 == 0 || i % 4 == 0 && i % 100 != 0) {

run++;

}

}

days = wholeMonth[month] - date.day;

for (int i = date.month + 1; i < 13; i++)

{

days += wholeMonth[i];

}

//cout << days<< endl;

days += (this->year - date.year - 1) \* 365 + run;

for (int i = 1; i < this->month; i++)

{

days += wholeMonth[i];

}

days += this->day;

//cout << "-" << days << endl;

return -days;

}

else

{

for (int i = date.year; i >= this->year; i--)

{

if (i % 400 == 0 || i % 4 == 0 && i % 100 != 0) {

run++;

}

}

//int wholeMonth[13] = { 0,31,28,31,30,31,30,31,31,30,31,30,31 };

days = wholeMonth[this->month] - this->day;

for (int i = this->month + 1; i < 13; i++)

{

days += wholeMonth[i];

}

days += (date.year - this->year - 1) \* 365 + run;

for (int i = 1; i < date.month; i++)

{

days += wholeMonth[i];

}

days += date.day;

//cout << "-" << days << endl;

return days;

}

}

int getDayOfYear() {

int days = 0;

for (int i = 0; i < this->month; i++)

{

days += wholeMonth[i];

}

if (month>2)

{

if (isLeapyear(this->year))

{

days++;

}

}

return days;

} //计算当前日期是本年的第几天

int getLeftDaysYear() {

if (isLeapyear(this->year))

{

return 366 - this->getDayOfYear();

}

return 365 - this->getDayOfYear();

}//计算当前日期距本年结束还有几天，不包括当前日期这一天

bool isLeapyear(int year1) {

if (year1 % 400 == 0 || year1 % 4 == 0 && year1 % 100 != 0) {

return true;

}

return false;

}//断参数年是否是闰年。

private:

int year;

int month;

int day;

char separator = '-'; // 日期分隔符；

static int wholeMonth[13];

};

int Date::wholeMonth[13] = { 0,31,28,31,30,31,30,31,31,30,31,30,31 };

class Employee {

public:

//构造函数，使用“成员初始化器”初始化数据成员

Employee(string firstName, string lastName, Date& birthDate, Date& hireDate) {

this->firstName = firstName;

this->lastName = lastName;

this->birthDate = birthDate;

this->hireDate = hireDate;

}

//打印员工的信息。调用Date类的print函数，打印员工的生日和雇佣日期。

void print() {

cout << this->firstName << ' ' << this->lastName << "'s birthDate and HireDate: " << endl;

this->birthDate.printFullYear();

this->hireDate.printFullYear();

}

//计算员工在参数指定的日期时，满多少岁。请使用Date类的fullYearsTo函数

int getAge(Date& date) {

return this->birthDate.fullYearsTo(date);

}

//计算该员工在参数指定的日期时，工作满了多少年。

int getYearsWorked(Date& date) {

return this->hireDate.fullYearsTo(date);

}

//计算该员工在参数指定的日期时，工作了多少天。使用Date类的daysTo函数。

int getDaysWorked(Date& date) {

return this->hireDate.daysTo(date);

}

//~Employee(); //析构函数

private:

string firstName;

string lastName;

Date birthDate; //内嵌对象，出生日期

Date hireDate; //内嵌对象，雇用日期

};

void main() {

Date birth(1969, 8, 11);

Date hire(1998, 4, 1);

Date today(2010, 4, 30);

Employee manager("Bob", "Blue", birth, hire);

cout << endl;

manager.print();

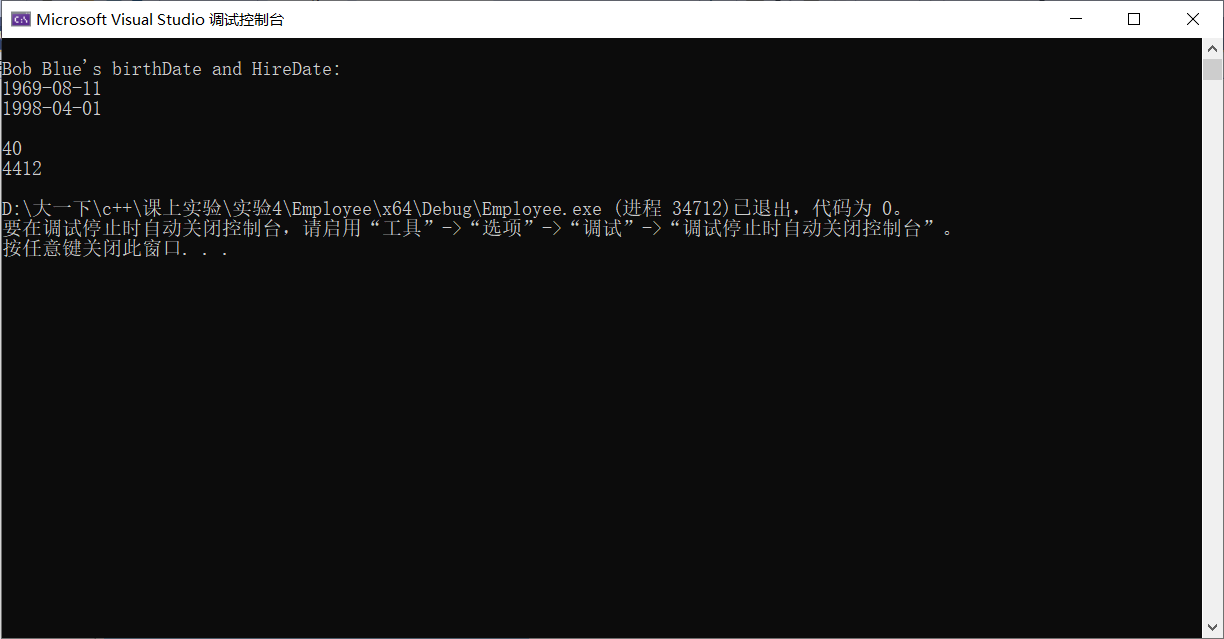
cout << endl;

cout << manager.getAge(today) << endl;

cout << manager.getDaysWorked(today) << endl;

}

截图：



### 实验二：CD

代码：

#include<iostream>

#include<string>

using namespace std;

class CD {

public:

CD(string name, string songs[]) {

this->singer = name;

for (int i = 0; i < 6; i++)

{

this->songs[i] = songs[i];

}

}

string getSinger() {

return this->singer;

} // 获得歌手的名称

string getSong(int index) {

return this->songs[index];

}// 获得某首歌的歌名

void listSongs() {

cout << "Singer: " << this->singer << endl;

for (int i = 0; i < 6; i++)

{

cout << i + 1 << ". " << this->songs[i] << endl;

}

} // 列出CD的内容

private:

string singer; // 歌手的名字。

string songs[6]; // 每张专辑6首歌的名字。

};

class CDPlayer {

public:

//CDPlayer();

/\*提供给用户一个菜单，通过这个菜单，用户可以选择：

1. 插入CD

2. 播放CD

3. 弹出CD

0. 关机 \*/

void showMenu() {

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*1. " << "插入CD\*" << endl;

cout << "\*2. " << "播放CD\*" << endl;

cout << "\*3. " << "弹出CD\*" << endl;

cout << "\*0. " << "关机\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

}

/\*插入CD. void insertCD(CD\* cd)，形参是指向CD对象的指针。如果CDPlayer中已经有CD，提示先取出CD；如果CDPlayer中没有CD，显示插入了哪位歌星的CD。\*/

void insertCD(CD\* cd) {

if (this->cdIn)

{

cout << "请先取出CD！" << endl;

}

else

{

this->cd = cd;

cout << "插入了" << this->cd->getSinger() << "的CD...... " << endl;

this->cdIn = true;

}

}

/\*弹出CD. CD \*ejectCD()，返回值是指向该CD对象的指针。如果CDPlayer中已经有CD，显示弹出了哪位歌星的CD，返回该CD的指针；如果CDPlayer中没有CD，提示CDPlayer中没有CD，返回NULL。\*/

void ejectCD() {

if (this->cdIn=false)

{

cout << "CDPlayer中没有CD!" << endl;

//return 0;

}

else

{

cout << "弹出了" << this->cd->getSinger() << "的CD...... " << endl;

this->cd = NULL;

this->cdIn = false;

//return 0;

}

}

/\*播放CD。如果CDPlayer中已经有CD，显示正在播放哪位歌星的CD，并打印CD中歌曲的清单；如果CDPlayer中没有CD，显示CDPlayer中没有CD，并提示用户插入CD。\*/

void play() {

if (this->cdIn==true)

{

cout << "正在播放" << this->cd->getSinger() << "的CD...... " << endl;

cd->listSongs();

}

else

{

cout << "请先插入CD！" << endl;

}

}

void exitCD() {

cout << "欢迎下次使用" << endl;

system("pause");

exit(0);

}

private:

/\* 插入CDPlayer中的CD，它是指向CD对象的指针。没有CD时，为null。使用指针，很好地模拟 了CD对象不是播放器的一部分，播放器只是读取放入其中的CD的内容。\*/

CD\* cd = NULL;

bool cdIn = false; // CDPlayer中是否已经插入CD

};

void main() {

string name;

string songs[6];

cout << "制造CD......" << endl;

// 输入歌手名字

cout << " Singer's Name: " << endl;

cin >> name; // 输入：周杰伦

// 输入该歌手的六首歌名（青花瓷、菊花台、双节棍等）

for (int i = 0; i < 6; i++) {

cout << " song" << (i + 1) << "#: ";

cin >> songs[i];

}

int chose = 0;

CD cd(name, songs); //制造CD

cd.listSongs(); //显示CD的内容

CDPlayer player; //制造CDplayer

while (true)

{

player.showMenu();

//生成播放机的按钮

cin >> chose;

switch (chose)

{

case 1:

player.insertCD(&cd);

system("pause");

system("cls");

break;

case 2:

player.play();

system("pause");

system("cls");

break;

case 3:

player.ejectCD();

system("pause");

system("cls");

break;

case 0:

player.exitCD();

system("cls");

break;

default:

system("cls");

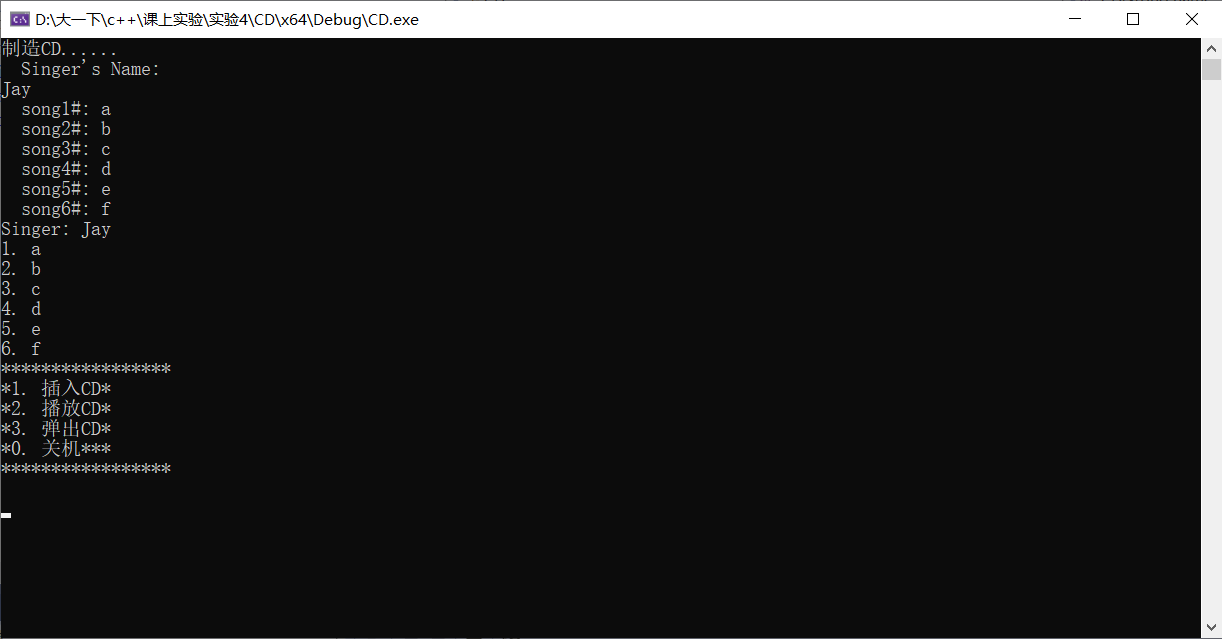
break;

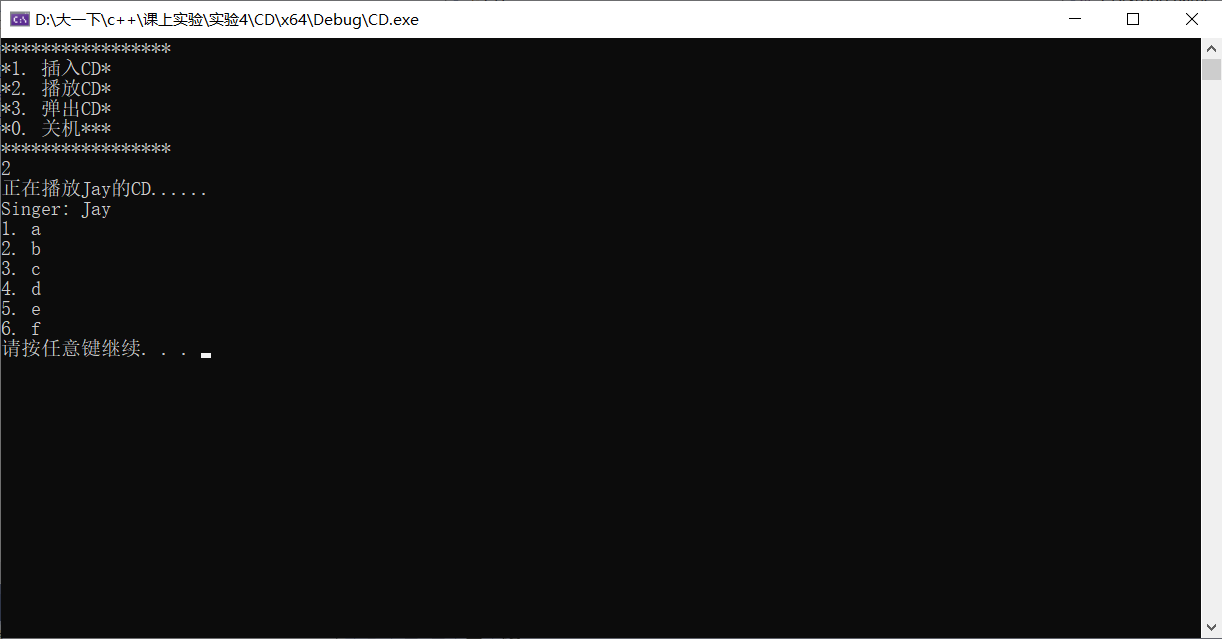
}

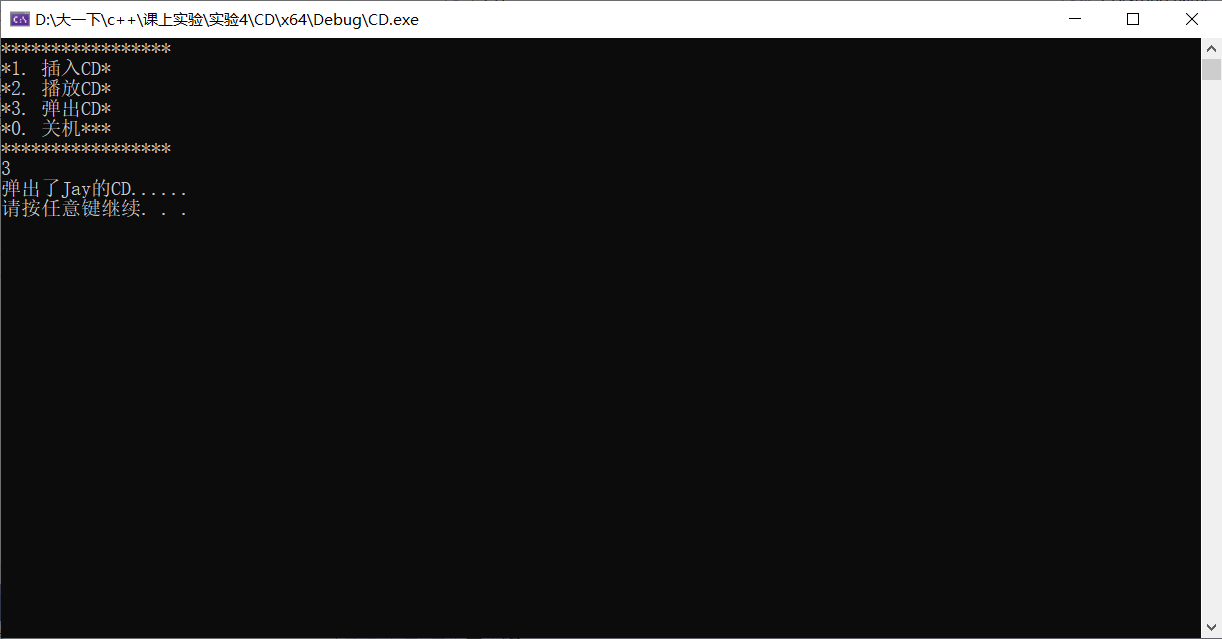
}

}

截图：







### 实验三：Cart

代码：

#include<iostream>

using namespace std;

#include<string>

class Cart;

class Commodity {

friend class Cart;// 商品类

public:

Commodity() {

}

Commodity(string name, int price, int num) {

this->name = name;

this->price = price;

this->num = num;

}

void printInfo() {

cout << " 商品的名称是： " << this->name << ' ' << "价格是： " << this->price << ' ' << "数量是：" << this->num << endl;

} // 输出该商品的信息：名称、标牌价格、购买数量

private:

string name;

int price;

int num;

};

class Cart { // 购物车类

public:

Cart() {

}

void addItem(Commodity& item) {

this->iterms[step] = item;

step++;

this->SumPrice += item.num \* item.price;

} // 添加一定数量的商品到购物车

void checkout() {

/\*cout << "您需要支付" << this->SumPrice << "元。" << endl;\*/

//return SumPrice;

cout << endl;

}// 对购物车中的商品进行结算

void printInvoice(){

cout << "您需要支付" << this->SumPrice << "元。" << endl;

for (int i = 0; i < this->step; i++)

{

cout << this->iterms[i].name << ' ' << this->iterms[i].price << ' ' << this->iterms[i].num << endl;

}

} // 将商品信息输出到显示器

private:

Commodity iterms[20];

int SumPrice = 0;

int step = 0;

};

int main() {

Commodity tShirt("Tshirt", 79, 2);// 创建服装对象，名称、价格、数量

Commodity suit("suit", 1099, 1); // 套装

Commodity hat("hat", 129, 3); // 帽子

Commodity tv("tv set", 4899, 1); // 创建家电对象，名称、价格、数量

Commodity ac("air condition", 5280, 1);// 空调

Cart myCart;

//将商品添加到购物车

myCart.addItem(tShirt);

myCart.addItem(suit);

myCart.addItem(hat);

myCart.addItem(tv);

myCart.addItem(ac);

myCart.checkout(); // 购物车商品结算，显示顾客需要支付的总金额

myCart.printInvoice(); // 将购物清单输出到显示器上

return 0;

}

截图：

