# 第11次实验

# 多继承的实现方式

**1、实验学时：**2个学时。

**2、实验类型：**设计性实验。

**3、实验目的：**

（1）了解单继承和多继承的区别、掌握多继承和的方法；

（2）掌握多继承中派生类的形式和基类成员初始化的方法。

**4、实验内容：**

**实验题目1：类的多继承和初始化机制**

（1）程序功能简介

设计一个程序描述由沙发和床派生而成的沙发床的情形。

（2）程序设计说明

① 定义一个木头类Wood和一个金属类Metal，分别具有wamount和mamount等数据成员，用于存放一件家具使用的木头和金属材料的数量；分别具有UseWood()和UseMental()等成员函数，分别用于输出一件家具使用的木头和金属材料的数量，并且在两个类的构造函数中调用各自的成员函数；

② 从Wood和Metal两个类派生出一个家具类Furniture，除了构造函数外，该类不具有任何数据成员和成员函数，仅用于传递基类初始化所需的数据；

③ 从Furniture类分别派生出一个沙发类Sofa和一个床类Bed。沙发类Sofa包含重量（sofaweight）、容量（sofacapacity）等数据成员和CanSit()等成员函数，其中CanSit()用于输出一个沙发具有的重量（sofaweight）和可以坐下的人数（sofacapacity）；床类Bed包含重量（bedweight）、容量（bedcapacity）等数据成员和CanLie()等成员函数，其中CanLie()用于输出一个床具有的重量（bedweight）和可以躺下的人数（bedcapacity）；

④ 从Sofa和Bed两个类派生出一个沙发床类SofaBed，具有ItsFunction()等成员函数，其中ItsFunction()用于输出一个沙发床具有的重量（来自于Sofa和Bed两个类的重量之和）和可以容纳的人数（来自于Sofa和Bed两个类的容量之和）；

⑤ 为上述类分别添加合适的构造函数，实现上述类的各自数据成员的初始化，并考虑：上述类的普通成员函数的调用方式，比如是否可在各自构造函数中调用？

⑥ 如果需要可以添加适当的其它成员，编写完整的程序并测试。

（3）程序调试运行

运行程序查看结果，并进行源代码调试和优化。

**参考代码如下：**

#include <iostream>

using namespace std;

class Wood{

public:

int wamount;

Wood(int temp):wamount(temp){UseWood();}

void UseWood(){cout<<"The furniture uses "<<wamount<<" pieces woods."<<endl;}

};

class Mental{

public:

int mamount;

Mental(int temp):mamount(temp){UseMental();}

void UseMental(){cout<<"The furniture uses "<<mamount<<" pieces mentals."<<endl;}

};

class Furniture: public Wood, public Mental{

public:

Furniture(int t1,int t2):Wood(t1),Mental(t2){}

};

class Sofa: public Furniture{

protected:

int sofacapacity; double sofaweight;

public:

};

class Bed: public Furniture{

protected:

int bedcapacity; double bedweight;

public:

Bed(int t1,int t2,int t3,int t4):Furniture(t1,t2),bedcapacity(t3),bedweight(t4){CanLie();}

void CanLie()

{cout<<"The bed weights "<<bedweight<<" and can lie "<<bedcapacity<<" persons."<<endl;}

};

class SofaBed: public Sofa, public Bed{

public:

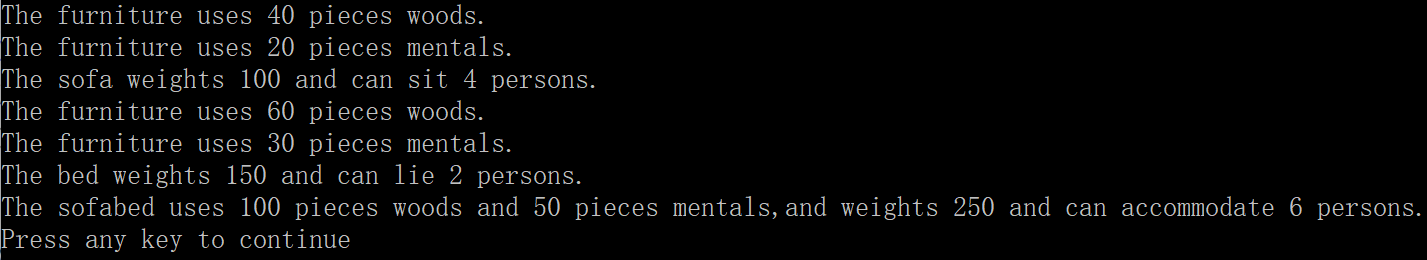
SofaBed(int t1,int t2,int t3,int t4,int t5,int t6,int t7,int t8):Sofa(t1,t2,t3,t4),Bed(t5,t6,t7,t8)

{ItsFunction();}

};

void main(){ SofaBed test(40,20,4,100,60,30,2,150); }

**运行结果如下：**



**实验题目2：设计一个日期时间类**

（1）程序功能简介

设计一个能够显示和修改当前日期和时间的类。

（2）程序设计说明

① 定义一个描述日期的类Date，构造函数完成年、月、日的初始化，包含一个修改日期的成员函数和一个显示日期的成员函数。该类可定义为：

class Date{

int Year,Month,Day;

public:

Date(int y=0,int m=0,int d=0): Year(y), Month(m), Day(d){}

void SetDate(int, int, int);

void ShowDate();

};

② 定义一个与Date类似的描述时间的类Time，然后用这两个类作为基类，公有派生出描述日期和时间的类DateTime，构造函数完成年、月、日、时、分、秒的初始化，同样包含一个修改日期和时间的成员函数和一个显示日期和时间的成员函数。DateTime类可定义为：

class DateTime: public Date, public Time{

public:

DateTime(int y=0,int mon=0,int d=0, int h=0,int min=0,int s=0){ShowDateTime();}

void SetDateTime();

void ShowDateTime();

};

注意：上述DateTime类定义体中的成员函数都需要调整；

③思考：将类DateTime改为由基类Date和Time保护派生或私有派生，程序能否正确编译和执行，为什么？

④ 根据类的需要添加适当的其它成员，编写完整的程序并测试。

（3）程序调试运行

运行程序查看结果，并进行源代码调试和优化。

**参考代码如下：**

#include <iostream.h>

#include <windows.h>

class Date{

int Year,Month,Day;

public:

Date(int y=0,int m=0,int d=0){Year=y; Month=m; Day=d;}

void SetDate(int y,int m,int d){Year=y;Month=m;Day=d;}

void ShowDate(){cout<<Year<<"-"<<Month<<"-"<<Day;}

};

class Time{

int Hour,Minute,Second;

public:

Time(int h=0,int m=0,int s=0){Hour=h; Minute=m; Second=s;}

void SetTime(int h,int m,int s){Hour=h;Minute=m;Second=s;}

void ShowTime(){cout<<Hour<<":"<<Minute<<":"<<Second;}

};

class DateTime:public Date, public Time{

public:

DateTime(int year=0,int month=0,int day=0,int hour=0,int minute=0,int second=0)

:Date(year,month,day),Time(hour,minute,second)

void SetDateTime(int year=0,int month=0,int day=0,int hour=0,int minute=0,int second=0)

void ShowDateTime()

};

void main(){

SYSTEMTIME sys;

GetLocalTime(&sys);

DateTime dt(sys.wYear,sys.wMonth,sys.wDay,sys.wHour,sys.wMinute,sys.wSecond);

char flag='y';

cout<<"是否需要修改日期和时间？(y/n)";

cin>>flag;

while(tolower(flag)=='y'){

cout<<"请输入新的日期和时间：";

cout<<"------------------------------------------"<<endl;

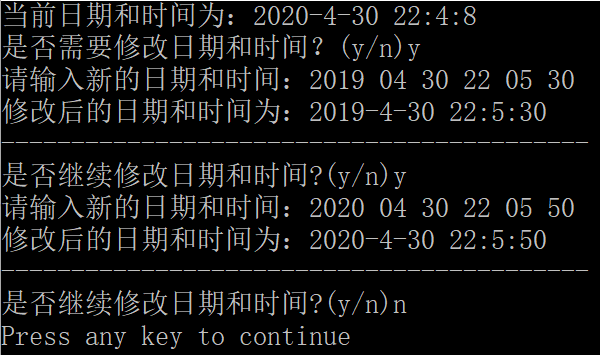
cout<<"是否继续修改日期和时间?(y/n)";

cin>>flag;

}

}

**运行结果如下：**



**5、实验结果：**

参考实验报告模板，撰写本次实验的实验报告，并提交实验报告和实验题目的源代码。