//主要修改：在基类新增了两个虚函数，新增了一个用于实现派生类成员函数的基类

//对main函数进行了精简，可以成功运行

// 类声明文件名：sclass3\_9\_companyStaff.h

// 编译预处理语句：如果未定义\_\_SCLASS3\_9\_COMPANYSTAFF\_H\_\_ 则定义它，

// 直到遇到#endif结束。

#ifndef \_\_SCLASS3\_9\_COMPANYSTAFF\_H\_\_

#define \_\_SCLASS3\_9\_COMPANYSTAFF\_H\_\_

#include<iostream>   //包含头文件，使用iostream库用于输入输出

#include<cstring>     //包含头文件，使用字符串处理函数

using namespace std; //使用std名字空间

const int LEN = 50;  //定义一个常量，表示字符数组长度

// CEducation（教育背景）类的声明

class CEducation

{

public:

    CEducation(){}; // 构造函数

    CEducation( char cSchool[], char cDegree )// 重载构造函数，为数据成员赋初值

    {

        strcpy( m\_cSchool, cSchool );

        m\_cDegree = cDegree;

    }

    void GetEdu( void );      // 输入教育背景的有关数据

    void PutEdu( void );     // 输出教育背景的有关信息

private:

    char m\_cSchool[LEN];     // 毕业学校

    char m\_cDegree;          // 最高学历：专科H、本科B、硕研M、博研D

};

// 以下是基类、聚合类CCompanyStaff的声明

class CCompanyStaff                                  // 员工类（根基类）

{

public:

    CCompanyStaff( void ){};                          // 构造函数

    ~CCompanyStaff( void ){};                         // 空析构函数

    void CalculateSal( void ){};                      // 计算实发工资

    void OutPut( void );                              // 输出员工基本信息

    void InPut( void );                               // 输如员工基本信息

    virtual void SetMData() {};

    virtual void SetSData() {};

protected:

    CEducation Edu;                                  // 内嵌对象（教育背景）

    int m\_iStaffNum;                                 // 工作编号

    char m\_cName[LEN];                               // 姓名

    float m\_fRateOfAttend;                           // 出勤率

    float m\_fBasicSal ;                              // 基本工资

    float m\_fPrize ;                                 // 奖金

    static int s\_iCount;                             // 静态累加器

};

//定义派生类：CAdminStaff（行政人员）

class CAdminStaff : public CCompanyStaff

{

public:

    CAdminStaff (){};                                // 派生类构造函数

    ~ CAdminStaff (){};                              // 派生类构造函数

    // 计算行政人员的实际发放工资：基本工资＋奖金×当月出勤率

    void CalculateSal( void )

    {

        cout << endl <<" 当月实发工资："

        << m\_fBasicSal + m\_fPrize \* m\_fRateOfAttend <<"元"<< endl;

    }

};

//定义派生类：CManager（经理）

class CManager : virtual public CCompanyStaff // 将CCompanyStaff指定为虚基类

{

public:

    CManager(){};                                     // 构造函数

    void SetMData( void )                             // 输入经理数据

    {

        cout << endl <<"  公司总销售额：";

        cin >> m\_fTAmount;

        cout << endl <<"  提成比例：";

        cin >> m\_fDeductTRate;

    }

    // 计算经理的实际发放工资：基本工资＋总销售额×提成比例

    void CalculateSal( void )

    {

        cout << endl <<"  当月实际发放工资：";

        cout << m\_fBasicSal + m\_fTAmount \* m\_fDeductTRate <<"元"<< endl;

    }

protected:

    float m\_fDeductTRate;                                // 经理提成比例

    float m\_fTAmount;                                    // 总销售额

};

//定义派生类：CSaleman（销售人员）

class CSaleman : virtual public CCompanyStaff // 将CCompanyStaff指定为虚基类

{

public:

    CSaleman (){};                                        // 构造函数

    void SetSData( void )                                // 设置销售员数据

    {

        cout << endl <<"  个人销售额：";

        cin >> m\_fPersonAmount;

        cout << endl <<"  提成比例：";

        cin >> m\_fDeductRate;

    }

     //计算销售员的实际发放工资：基本工资＋个人销售额×提成比例

    void CalculateSal( void )

    {

        cout << endl <<"  当月实际发放工资：";

        cout << m\_fBasicSal + m\_fPersonAmount \* m\_fDeductRate <<"元"<< endl;

    }

protected:

    float m\_fDeductRate;                                 // 销售员提成比例

    float m\_fPersonAmount;                               // 个人销售额

};

//定义派生类：CSaleManager（销售经理）

class CSaleManager : public CSaleman, public CManager    // 派生销售经理类

{

public:

    CSaleManager(){};                                     // 构造函数

    /\*

       计算销售经理实发工资:

       基本工资＋个人销售额×销售员提成比例＋总销售额×经理提成比例

    \*/

    void CalculateSal( void )

    {

        cout << endl <<"  当月实际发放工资：";

        cout << m\_fBasicSal + m\_fPersonAmount \* m\_fDeductRate

                + m\_fTAmount \* m\_fDeductTRate;

        cout <<"元"<< endl;

    }

};

//定义用于改变指针指向、输出的Context类

class Context

{

public:

    Context(CCompanyStaff\* pCCompanyStaff) : m\_pCCompanyStaff(pCCompanyStaff) {}

    void RequestCManager()

    {

        if (m\_pCCompanyStaff)

        {

            //核心步骤,根据指针指向的类型进行选择调用子类的函数。

            //虚函数使得其可以实现基类指针调用基类指针所指向的子类象的函数。动态绑定。

            m\_pCCompanyStaff->InPut();         // 调用基类的接口

            m\_pCCompanyStaff->SetMData();      // 调用经理类的接口

            m\_pCCompanyStaff->OutPut();        // 调用基类的接口

            m\_pCCompanyStaff->CalculateSal();  // 调用经理类的接口

        }

    }

    void RequestCSaleman()

    {

        if (m\_pCCompanyStaff)

        {

            m\_pCCompanyStaff->InPut();

            m\_pCCompanyStaff->SetSData();

            m\_pCCompanyStaff->OutPut();

            m\_pCCompanyStaff->CalculateSal();  // 调用销售人员类的接口

        }

    }

    void RequestCSaleManager()

    {

        if (m\_pCCompanyStaff)

        {

            m\_pCCompanyStaff->InPut();

            m\_pCCompanyStaff->SetMData();

            m\_pCCompanyStaff->SetSData();

            m\_pCCompanyStaff->OutPut();

            m\_pCCompanyStaff->CalculateSal();   // 调用销售经理类的接口

        }

    }

    void RequestCAdminStaff()

    {

        if (m\_pCCompanyStaff)

        {

            m\_pCCompanyStaff->InPut();

            m\_pCCompanyStaff->OutPut();

            m\_pCCompanyStaff->CalculateSal();   // 调用行政人员类的接口

        }

    }

    void ChangeCCompanyStaff(CCompanyStaff\* pCCompanyStaff)

    {

        m\_pCCompanyStaff = pCCompanyStaff;//改变指针指向的类的类型

    }

private:

    CCompanyStaff\* m\_pCCompanyStaff;

};

#endif   //结束编译预处理

// 类实现文件：sclass3\_9\_companyStaff.cpp

#include"sclass3\_9\_companyStaff.h" // 包含类声明文件

int CCompanyStaff::s\_iCount = 1000; // 初始化静态数据成员s\_iCount

// 实现类CEducation的成员函数GetEdu，输入信息

void CEducation::GetEdu( void )

{

    cout << endl <<"  毕业学校：";

    cin >> m\_cSchool;

    cout << endl <<"  最高学历：（专科H、本科B、硕研M、博研D）";

    cin >> m\_cDegree;

    // 用toupper函数将用户输入的学历规范为大写字符

    m\_cDegree = toupper( m\_cDegree );

}

// 实现类CEducation的成员函数PutEdu，输出信息

void CEducation::PutEdu( void )

{

    cout << endl <<"  毕业学校："<< m\_cSchool << endl;

    cout << endl <<"  最高学历：";

    switch ( m\_cDegree )

    {

        case 'H': cout <<"专科"<< endl;

                  break;

        case 'B': cout <<"本科"<< endl;

                  break;

        case 'M': cout <<"硕研"<< endl;

                  break;

        case 'D': cout <<"博研"<< endl;

                  break;

        default:  cout <<"空"<< endl;  // 输入的学历在选项之外则为"空"

                  break;

    }

}

// 实现类CCompanyStaff的成员函数InPut，输入员工各项数据

void CCompanyStaff:: InPut( void )

{

    m\_iStaffNum = ++s\_iCount;       // 静态累加器自加1后赋给工作编号

    cout << endl << "请输入编号为"<< m\_iStaffNum <<"号员工的姓名：";

    cin >> m\_cName; // 输入员工姓名

    Edu.GetEdu(); // 通过内嵌对象调用类CEducation的接口GetEdu，输入教育背景相关数据

    cout << endl << "  基本工资：";

    cin >> m\_fBasicSal;     // 输入基本工资

    cout << endl << "  奖金：";

    cin >> m\_fPrize;     // 输入奖金

    cout << endl <<"  出勤率：";

    cin >> m\_fRateOfAttend;  // 输入出勤率

    // 规范用户输入的出勤率的值

    if ( m\_fRateOfAttend > 1.0 )

    {

        m\_fRateOfAttend = 1.0;

    }

    else

        if ( m\_fRateOfAttend < 0 )

        {

            m\_fRateOfAttend = 0;

        }

}

// 实现类CCompanyStaff的成员函数OutPut，输出员工信息

void CCompanyStaff:: OutPut( void )

{

    cout << endl << "显示员工" << m\_cName <<"的基本数据：";

    cout << endl << "  工作编号：" << m\_iStaffNum;

    Edu.PutEdu();      // 通过内嵌对象调用类CEducation的接口PutEdu

    cout << endl << "  基本工资：" << m\_fBasicSal;

    cout << endl << "  奖金：" << m\_fPrize;

    cout << endl << "  出勤率：" <<  m\_fRateOfAttend \* 100 << "%";

}

//主文件：smain3\_9.cpp

#include"sclass3\_9\_companyStaff.h"   // 包含类声明文件

//测试程序

int main()

{

    char flag ='Y';            // 设置判断是否继续录入的标志并初始化为'Y'

    while ( toupper( flag ) == 'Y')

    {

        cout <<"请选择录入类别（1.经理2.销售员3.销售经理4.行政人员）";

        int n;

        cin >> n;

        CCompanyStaff\* pA = new CManager();

        CCompanyStaff\* pB = new CSaleman();

        CCompanyStaff\* pC = new CSaleManager();

        CCompanyStaff\* pD = new CAdminStaff();

        Context\* pContext = new Context(pA);//随便指一个

        if (n==1){ pContext->ChangeCCompanyStaff(pA);pContext->RequestCManager();}

        else if(n==2){ pContext->ChangeCCompanyStaff(pB);pContext->RequestCSaleman();}

        else if(n==3){ pContext->ChangeCCompanyStaff(pC);pContext->RequestCSaleManager();}

        else if(n==4){ pContext->ChangeCCompanyStaff(pD);pContext->RequestCAdminStaff();}

        else cout << "选择有误!"<< endl;

        delete pContext;

        delete pA;

        delete pB;

        delete pC;

        delete pD;

        cout << endl <<"是否继续录入信息？（Y/N）";

        cin >> flag;

    }

    system("pause");

    return 0;

}