实验报告

P374T5

（1）

B1.i和b1.j正确，因为i和j分别是基类A中公有成员和受保护成员继承到派生类B中依然是公有和受保护状态。

b1.k错误，因为k是私有成员，虽然被继承到派生类中但是不能访问。

（2）

可以调用，派生类b继承的是基类a中的所有函数，只要是非私有的就可以访问。

（3）

i和j可以，k为基类私有，不可访问。

（4）

可以引用基类A的成员i，j，派生类B的成员m，派生类C的成员p，不能访问基类A中的私有成员k，派生类B的私有成员n。

（5）

可以调用，c为public继承，可以访问受保护和公有成员。

（6）

只要是非私有都可以访问，所以能调用。

P374T7

实验目的：

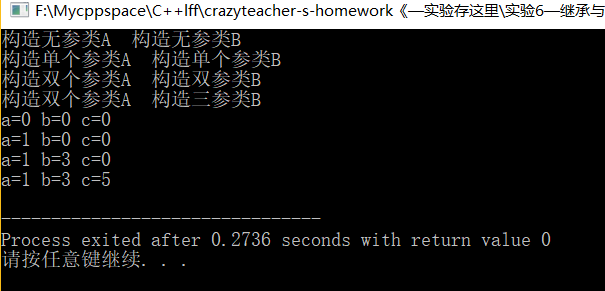
实现基础的派生类

了解派生类中对基类的构造的调用。

实验代码：

|  |
| --- |
| #include<iostream>  using namespace std;  class A  {  public:  A(){a=0;b=0;cout<<"构造无参类A ";}  A(int i){a=i;b=0;cout<<"构造单个参类A ";}  A(int i,int j){a=i;b=j;cout<<"构造双个参类A ";}  void display(){cout<<"a="<<a<<" b="<<b;}  private:  int a;  int b;  };  class B : public A  {  public:  B(){c=0;cout<<"构造无参类B\n";}  B(int i):A(i){c=0;cout<<"构造单个参类B\n";}  B(int i,int j):A(i,j){c=0;cout<<"构造双参类B\n";}  B(int i,int j,int k):A(i,j){c=k;cout<<"构造三参类B\n";}  void display1()  {  display();  cout<<" c="<<c<<endl;  }  private:  int c;  };  int main()  {  B b1;  B b2(1);  B b3(1,3);  B b4(1,3,5);  b1.display1();  b2.display1();  b3.display1();  b4.display1();  return 0;  } |

运行结果:



结果分析：

由结果可以看出main函数中定义的b1,b2,b3,b4四个对象分别调用了基类A中的无参，单个参数，两个参数，两个参数的构造函数，定义对象时先调用基类构造函数，再调用派生类构造函数。

P374T9

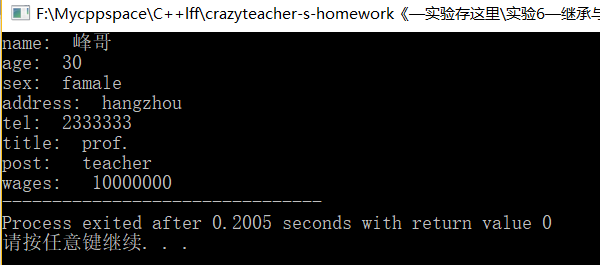
实验目的：

学会使用多重继承。

实验代码

|  |
| --- |
| #include<iostream>  #include<string>  using namespace std;  class teacher  {  public:  teacher(string nam,int a,string se,  string addr,string t,string tit);  void display();  protected:  int age;  string name;  string sex;  string address;  string tel;  string title;  };  teacher::teacher(string nam,int a,string se,  string addr,string t,string tit):  name(nam),age(a),sex(se),address(addr),  tel(t),title(tit){}  void teacher::display()  {  cout<<"name"<<name<<endl;  cout<<"age"<<age<<endl;  cout<<"sex"<<sex<<endl;  cout<<"address"<<address<<endl;  cout<<"tel"<<tel<<endl;  cout<<"title"<<title<<endl;  }  class cadre  {  public:  cadre(int a,string nam,string pos,string se,string addr,string t);  void display();  protected:  int age;  string post;  string name;  string sex;  string address;  string tel;  };  cadre::cadre(int a,string nam,string pos,string se,string addr,string t):  age(a),name(nam),post(pos),sex(se),address(addr),tel(t){}  void cadre::display()  {  cout<<"name"<<name<<endl;  cout<<"age"<<age<<endl;  cout<<"post"<<post<<endl;  cout<<"sex"<<sex<<endl;  cout<<"address"<<address<<endl;  cout<<"tel"<<tel<<endl;  }  class teacher\_cadre:public teacher,public cadre  {  public:  teacher\_cadre(int age,string name,string post,string sex,  string address,string tel,string title,int wages);  void show();  private:  int wages;  };  teacher\_cadre::teacher\_cadre(int age,string name,string post,  string sex,string address,string tel,string title,int wage):  teacher(name,age,sex,address,tel,title),  cadre(age,name,post,sex,address,tel),wages(wage){}  void teacher\_cadre::show()  {  teacher::display();  cout<<"post"<<cadre::post<<endl;  cout<<"wages"<<wages;  }  int main()  {  teacher\_cadre c(30,"峰哥","teacher","famale","hangzhou","2333333","prof.",10000000);  c.show();  return 0;  } |

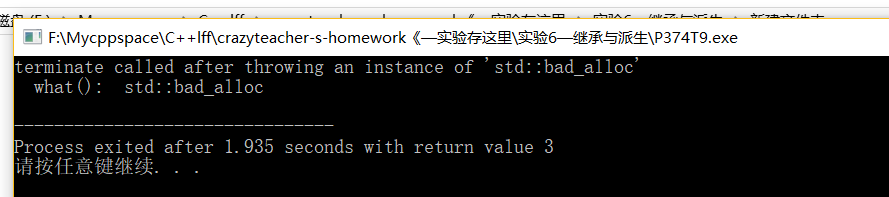
运行结果：

结果分析：

对两个多个基类的继承要注意不要搞混同名函数

按序给数据成员赋值输出。

出现过的错误



在写派生类的时候少了一个数据成员，初始化后输出的时候内存爆了。

写代码一定要细致！

P374T10

实验目的：

实现类的继承与组合

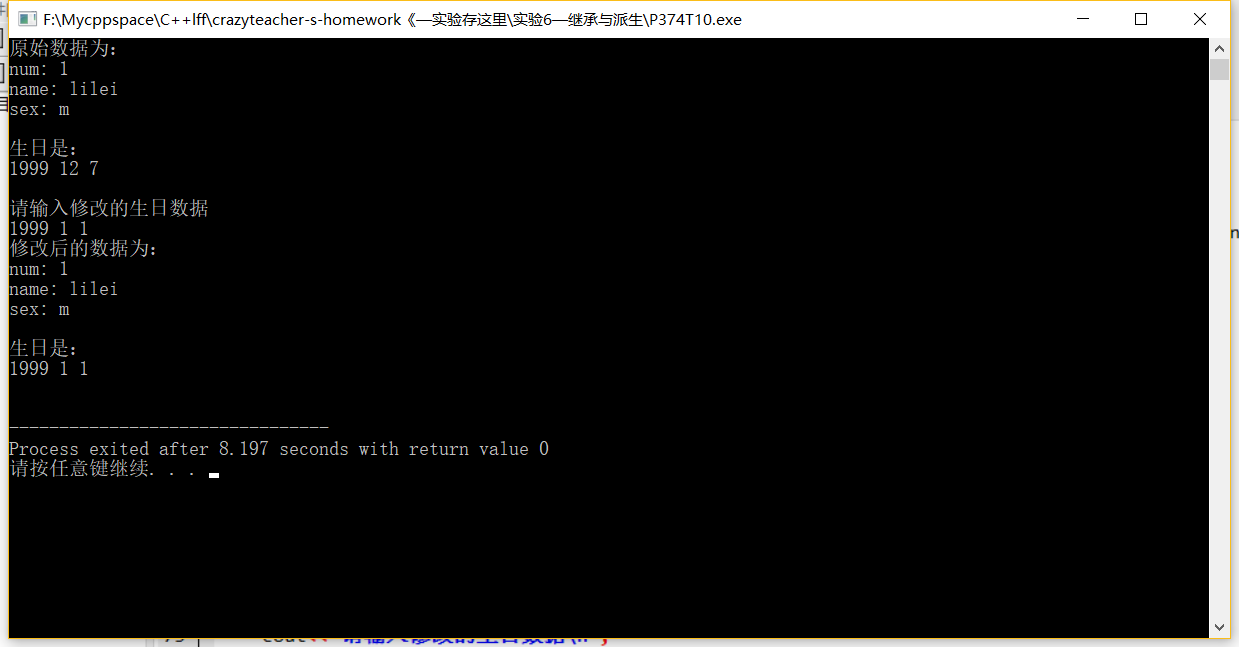
同名函数的使用

数据可修改

实验代码：

|  |
| --- |
| #include<iostream>  #include<cstring>  #include<string>  using namespace std;  class teacher  {  public:  teacher(int num,string name,char sex);  void display();  private:  int num;  string name;  char sex;  };  teacher::teacher(int num,string name,char sex):  num(num),name(name),sex(sex){}  void teacher::display()  {  cout<<"num: "<<num<<endl  <<"name: "<<name<<endl  <<"sex: "<<sex<<endl;  }  class birthdate  {  public:  birthdate(int year,int month,int day);  void display();  void reset(int y,int m,int d);  private:  int year;  int month;  int day;  };  birthdate::birthdate(int year,int month,int day):  year(year),month(month),day(day){}  void birthdate::display()  {  cout<<"生日是：\n";  cout<<year<<' '<<month<<' '<<day;  }  void birthdate::reset(int y,int m,int d)  {  year = y;  month = m;  day = d;  }  class professor:public teacher  {  public:  professor(int num,string name,char sex,int year,int month,int day);  void display();  void resetbirthdate(int y,int m,int d);  private:  birthdate birthday;  };  professor::professor(int num,string name,char sex,int year,int month,int day):teacher(num,name,sex),birthday(year,month,day){}  void professor::display()  {  teacher::display();  cout<<endl;  birthday.display();  cout<<endl<<endl;  }  void professor::resetbirthdate(int y,int m,int d)  {  birthday.reset(y,m,d);  }  int main()  {  cout<<"原始数据为：\n";  professor prof1(1,"lilei",'m',1999,12,7);  prof1.display();  cout<<"请输入修改的生日数据\n";  int y,m,d;  cin>>y>>m>>d;  cout<<"修改后的数据为：\n";  prof1.resetbirthdate(y,m,d);  prof1.display();  } |

运行结果



结果分析：

实现了对生日数据的修改，并同步到profession类中，并输出，具体操作是在birthdate类里用函数调用私有数据，再在profession类里用私有数据成员birthdate类对象birthday调用该函数实现修改。

5、自行车类

实验要求：

1. Vehicle(含private成员变量:限速+车重)，Bicycle(含private成员变量:限速+车重+车高)，Motor\_veh(含private成员变量:限速+车重+限载人数)；
2. 四个类中都包含同名成员函数show()，可输出各自的类对象信息；
3. 成员函数需类内声明，类外定义；
4. 参考题目给出的main()及运行结果设计实现类。

实验代码

头文件部分

|  |
| --- |
| #ifndef VEHICLEHEAD\_H\_H  #define VEHICLEHEAD\_H\_H  #include<iostream>  #include<cstring>  #include<string>  using namespace std;  class vehicle  {  public:  vehicle(float l\_s,float w);  void show();  private:  float limit\_speed;  float weight;  };  class bicycle:virtual public vehicle  {  public:  bicycle(float l\_s,float w,float h);  void show();  void show\_h();  private:  float high;  };  class motor\_veh:virtual public vehicle  {  public:  void show();  void show\_s();  motor\_veh(float l\_s,float w,int seat);  private:  int seat;  };  class motorcycle:public motor\_veh,public bicycle  {  public:  motorcycle(float l\_s,float w,float high,int seat);  void show();  };  #endif |

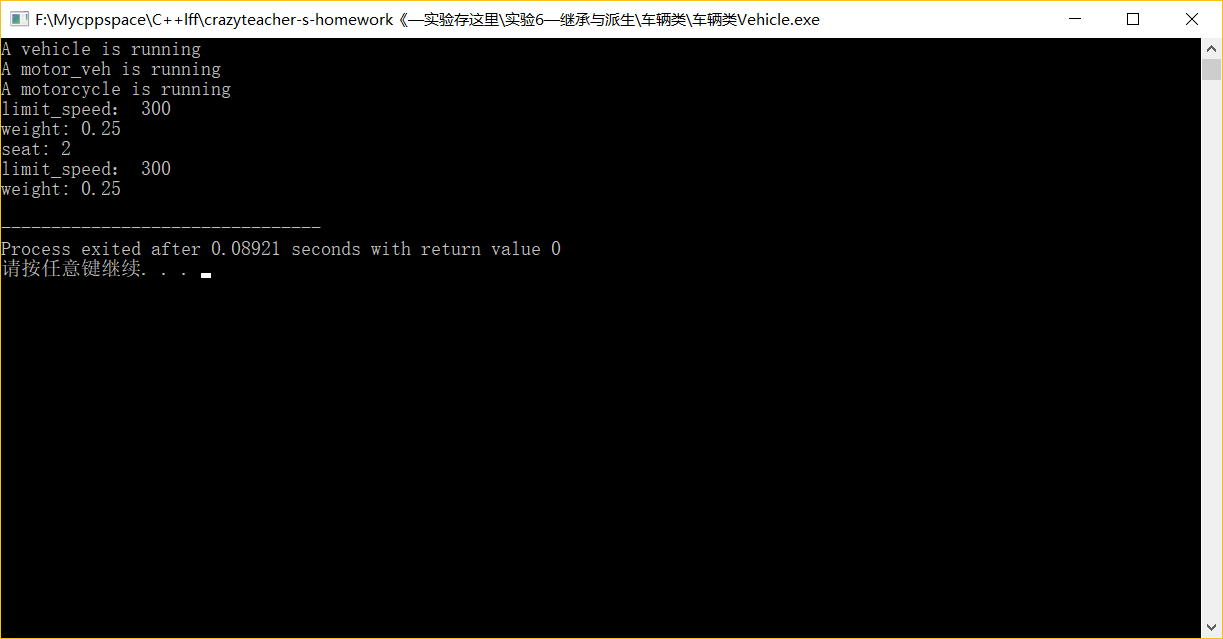
函数定义

|  |
| --- |
| #include<iostream>  #include "vehiclehead.h"  using namespace std;  vehicle::vehicle(float l\_s,float w):limit\_speed(l\_s),weight(w){cout<<"A vehicle is running\n";}  void vehicle::show()  {  cout<<"limit\_speed： "<<limit\_speed<<endl  <<"weight: "<<weight<<endl;  }  bicycle::bicycle(float l\_s,float w,float h):vehicle(l\_s,w),high(h){}  void bicycle::show()  {  vehicle::show();  cout<<"high: "<< high<<endl;  }  void bicycle::show\_h()  {  cout<<"high: "<<high<<endl;  }  motor\_veh::motor\_veh(float l\_s,float w,int seat):vehicle(l\_s,w),seat(seat){cout<<"A motor\_veh is running\n";}  void motor\_veh::show()  {  vehicle::show();  cout<<"seat: "<<seat<<endl;  }  void motor\_veh::show\_s()  {  cout<<"seat: "<<seat<<endl;  }  motorcycle::motorcycle(float l\_s,float w,float h,int seat):vehicle(l\_s,w),bicycle(l\_s,w,h),motor\_veh(l\_s,w,seat){cout<<"A motorcycle is running\n";}  void motorcycle::show()  {  vehicle::show();  motor\_veh::show\_s();  } |

Main函数

|  |
| --- |
| #include "vehiclehead.h"  using namespace std;  int main()  {  motorcycle m(300,0.25,1.1,2); //限速300，车重0.25吨，车高1.1米，限载人数2  m.show();  m.vehicle::show();  return 0;  } |

运行结果：



结果分析：

因为测试句子放在了构造函数里所以和样例有所出入。但可以体现出构造函数的顺序。第四行到第六行的结果是main函数里m.show()函数的输出结果。后两行是m.vehicle::show()的运行结果。

6、工资管理

实验要求：

1. 四个类中都包含同名成员函数pay()和display()，可计算各类员工工资并输出各类员工的相关信息；
2. 成员函数需类内声明，类外定义；
3. 创建员工对象时，可通过终端录入员工个人信息，包括所有员工的员工编号id、姓名name以及月工资salary；
4. 根据公司制度，计算各类员工的工资，并可在输出某员工工资的同时，显示该员工姓名、工号、工种、工作时长、销售额等等可能与该员工工资计算方法相关的全面信息。公司薪金政策如下：工资行政管理人员拿固定月工资5000元/月；外聘技术人员按100元/小时的时薪计算月工资；销售人员按当月销售额的7%提成作为当月工资；销售经理以基础工资8000，加上本人销售额的7%及其主管部门当月销售总额的千分之二作为当月工资。

实验代码：

头文件部分

|  |
| --- |
| #ifndef SALARYHEAD\_H\_H  #define SALARYHEAD\_H\_H  #include<iostream>  using namespace std;  class employee{  public:  employee():id(""),name(""),salary(0){}z  employee(string id,string name):id(id),name(name),salary(0){}  virtual void pay()=0 ;  virtual void display();  virtual void setdate()=0 ;  protected:  string id;  string name;  double salary;  };  class manager:virtual public employee //行政管理人员  {  public:  manager():employee(){}  manager(string id,string name):employee(id,name){}  virtual void setdate();  virtual void display();  virtual void pay();  };  class technician:virtual public employee  {  public:  technician():employee(),time(0){}  technician(string id,string name,int time):employee(id,name),time(time){}  virtual void display();  virtual void pay();  virtual void setdate();  protected:  int time;  };  class salesman:virtual public employee  {  public:  salesman():employee(),sales(0){}  salesman(string id,string name,double sales):employee(id,name),sales(sales){}  virtual void display();  virtual void pay();  virtual void setdate();  protected:  double sales;  };  class salesmanager:virtual public employee  {  public:  salesmanager():employee(),sales(0),sumsales(0){}  salesmanager(string id,string name,double sales,double sumsales):employee(id,name),sales(sales),sumsales(sumsales){}  virtual void display();  virtual void pay();  virtual void setdate();  protected:  double sales;  double sumsales;  };  #endif |

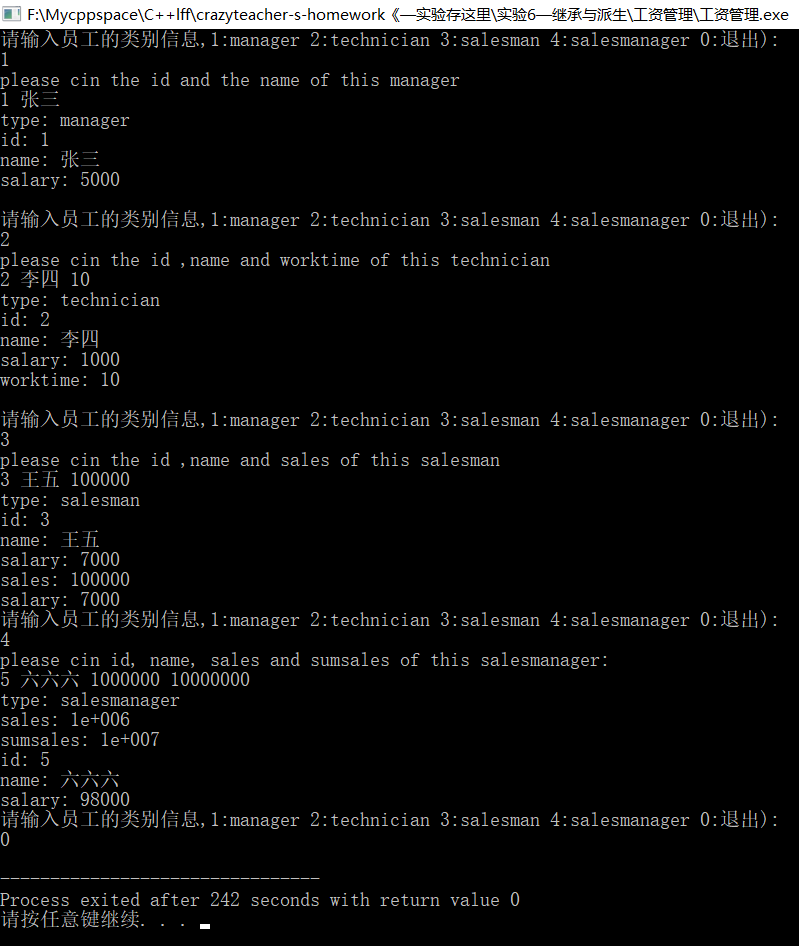
函数定义

|  |
| --- |
| #include<iostream>  #include"salaryhead.h"  using namespace std;  void employee::display()  {  cout<<"id: "<<id<<endl  <<"name: "<<name<<endl  <<"salary: "<<salary<<endl;  }  void manager::pay()  {  salary = 5000;  }  void manager::setdate()  {  cout<<"please cin the id and the name of this manager\n";  cin>>id>>name;  }  void manager::display()  {  cout<<"type: manager\n";  employee::display();  cout<<endl;  }  void technician::setdate()  {  cout<<"please cin the id ,name and worktime of this technician\n";  cin>>id>>name>>time;  }  void technician::pay()  {  salary = time\*100;  }  void technician::display()  {  cout<<"type: technician\n";  employee::display();  cout<<"worktime: "<<time<<endl<<endl;  }  void salesman::setdate()  {  cout<<"please cin the id ,name and sales of this salesman\n";  cin>>id>>name>>sales;  }  void salesman::pay()  {  salary = 0.07\*sales;  }  void salesman::display()  {  cout<<"type: salesman\n";  employee::display();  cout<<"sales: "<<sales<<endl  <<"salary: "<<salary<<endl;  }  void salesmanager::pay()  {  salary = 0.07\*sales+0.002\*sumsales+8000;  }  void salesmanager::setdate()  {  cout<<"please cin id, name, sales and sumsales of this salesmanager: \n";  cin>>id>>name>>sales>>sumsales;  }  void salesmanager::display()  {  cout<<"type: salesmanager\n"  <<"sales: "<<sales<<endl  <<"sumsales: "<<sumsales<<endl;  employee::display();  } |

Main函数部分

|  |
| --- |
| #include <iostream>  #include "salaryhead.h"  using namespace std;  int main()  {  while(1)  {  int t;  cout << "请输入员工的类别信息,1:manager 2:technician 3:salesman 4:salesmanager 0:退出):"<< endl;  cin>>t;  if(t == 0) break;  else if(t == 1)  {  manager Manager;  Manager.setdate();  Manager.pay();  Manager.display();  }  else if(t == 2)  {  technician tech;  tech.setdate();  tech.pay();  tech.display();  }  else if(t == 3)  {  salesman sale;  sale.setdate();  sale.pay();  sale.display();  }  else if(t == 4)  {  salesmanager mana;  mana.setdate();  mana.pay();  mana.display();  }  }  } |

运行结果：



结果分析：

做了一个很丑的菜单，在主函数里本来想利用多态来写，但是 不知道遇到了什么问题暂时还没解决，所以就给每个类定义了对象来实现的。对每个类型的职业，输出相关的数据，利用pay（）函数算出工资。

总结：

总的来说这次实验的难度不大，在实验里有遇到以前从来没遇到过的错误，算是长点脑子，以后少犯错。