**计算机科学与工程学院 实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **实验课程名称** | | **C++程序设计实验** | | | **实验成绩** |  |
| **专业** | **计算机Ⅰ类** | | **班级** | **1707** | **指导教师签字** |  |
| **学号** | **20174631** | | **姓名** | **孔振华** | **实验报告批改时间** |  |
| **实验项目目录**   1. **实验一……类与对象** 2. **实验二……数组、指针与字符串** 3. **实验三……继承与派生** 4. **实验四……多态性** | | | | | | |
| **实验报告正文**  **实验一 类与对象**  **实验目的**  **1.掌握类的定义和使用**  **2.掌握类的定义和对象的声明**  **3.复习具有不同访问属性的成员的访问方式**  **4.观察构造函数和析构函数的执行过程**  **5.学习类的组合使用方法**  **6.使用VC++的debug调试功能观察程序流程，跟踪观察类的构造函数、析构函数、成员函数的执行顺序。**  **一．CPU**  **实验要求**  定义一个CPU类，包含等级（rank）、频率（frequency）、电压（voltage）等属性，有两个公有成员函数run、stop。其中，rank为枚举类型CPU\_Rank,定义为enum CPU\_Rank{P1=1,P2,P3,P4,P5,P6,P7}，frequency为单位是MHz的整型数，voltage为浮点型的电压值。观察构造函数和析构函数的调用顺序。  **实验内容**  首先定义枚举类型CPU\_Rank，例如enum CPU\_Rank{P1=1, P2, P3, P4, P5, P6, P7}，再定义CPU类，包含等级（rank）、频率（frequency）、电压（voltage）等私有数据成员，定义成员函数run、stop，用来输出提示信息，在构造函数和析构函数中也可以输出提示信息。在主程序中定义一个CPU的对象，调用其成员函数，观察类对象的构造与析构顺序，以及成员函数的调用。程序名：lab4\_1.cpp。  **源程序**  #include<iostream>  using namespace std;  enum CPU\_rank {P1 = 1, P2, P3, P4, P5, P6, P7};//枚举类型，从1至7  class CPU {  public:  CPU(CPU\_rank myrank, int myfre, double myvoltage);//构造函数  ~CPU();//析构函数  CPU(CPU &c);//复制构造函数  CPU\_rank getRank();//获取Rank  void Run();//Run!  void Stop();//Stop!  int getFre();//获取频率  double getVol();//获取voltage  private:  CPU\_rank rank;  int frequency;  double voltage;  };  CPU::CPU(CPU\_rank myrank, int myfre, double myvoltage) {  cout << "Constructor" << endl;  rank = myrank;  frequency = myfre;  voltage = myvoltage;  }  CPU::CPU(CPU &c) {  cout << "Copy Constructor" << endl;  rank = c.rank;  frequency = c.frequency;  voltage = c.voltage;  }  CPU::~CPU() {  cout << "Destructor" << endl;  cin.get();  }  CPU\_rank CPU::getRank() {  return rank;  }  void CPU::Run() {  cout << "CPU is Runing!" << endl;  };  void CPU::Stop() {  cout << "CPU stop!" << endl;  }  int CPU::getFre() {  return frequency;  }  double CPU::getVol() {  return voltage;  }  int main() {  int my\_rank;  int my\_fre;  double my\_vol;  cout << "Please input the CPU data: rank,frequency,voltage " << endl;//输入CPU数值  cin >> my\_rank >> my\_fre >> my\_vol;  if(my\_rank > 0 && my\_rank <= 7) {//判断输入是否合法  CPU CPU1((CPU\_rank)my\_rank, my\_fre, my\_vol);  CPU CPU2(CPU1);//复制  CPU2.Run();//RUN  CPU2.Stop();//STOP  } else {  cout << "Please inter the right data!" << endl;  }  return 0;  }  **运行结果**    **二．Computer**  **实验要求：**  定义一个简单的Computer类，有数据成员芯片（cpu）、内存（ram）、光驱（cdrom）等等，有两个公有成员函数run、stop。cpu为CPU类的一个对象，ram为RAM类的一个对象，cdrom为CDROM类的一个对象，定义并实现这个类。  **实验内容：**  首先定义CPU类（可使用第二题中的CPU类）、RAM类、CDROM类。再定义Computer类；定义私有数据成员cpu、ram、cdrom，定义公有成员函数run、stop，可在其中输出提示信息。在主程序中定义一个Computer的对象，调用其成员函数，观察类对象及其成员变量的构造与析构顺序，以及成员函数的调用。程序名：lab4\_2.cpp.  **源程序：**  #include<iostream>  using namespace std;  enum CPU\_rank {P1 = 1, P2, P3, P4, P5, P6, P7};//枚举类型，从1至7  class CPU {  public:  CPU(CPU\_rank myrank, int myfre, double myvoltage);//构造函数  ~CPU();//析构函数  CPU(CPU &c);//复制构造函数  CPU\_rank getRank();//获取Rank  void Run();//Run!  void Stop();//Stop!  int getFre();//获取频率  double getVol();//获取voltage  private:  CPU\_rank rank;  int frequency;  double voltage;  };  CPU::CPU(CPU\_rank myrank, int myfre, double myvoltage) {  cout << "Constructor" << endl;  rank = myrank;  frequency = myfre;  voltage = myvoltage;  }  CPU::CPU(CPU &c) {  cout << "Copy Constructor" << endl;  rank = c.rank;  frequency = c.frequency;  voltage = c.voltage;  }  CPU::~CPU() {  cout << "Destructor" << endl;  cin.get();  }  CPU\_rank CPU::getRank() {  return rank;  }  void CPU::Run() {  cout << "CPU is Runing!" << endl;  };  void CPU::Stop() {  cout << "CPU stop!" << endl;  }  int CPU::getFre() {  return frequency;  }  double CPU::getVol() {  return voltage;  }  class RAM {  public:  RAM(int mySize);  ~RAM();  RAM(RAM &r);  private:  int Size; //内存大小  };  RAM::RAM(int mySize) { //构造函数  cout << "RAM Constructor" << endl;  Size = mySize;  }  RAM::~RAM() { //析构函数  cout << "RAM Destructor" << endl;  }  RAM::RAM(RAM &r) { //复制构造函数  cout << "RAM Copy Constructor" << endl;  Size = r.Size;  }  //CDROM  class CDROM {  public:  CDROM(int my\_size); //构造函数  ~CDROM(); //析构函数  CDROM(CDROM & c); //复制构造函数  private:  int Size; //光驱大小  };  CDROM::CDROM(int my\_size) {  cout << "CDROM Constructor" << endl;  Size = my\_size;  }  CDROM::CDROM(CDROM &c ) {  cout << "CDROM Copy Constructor" << endl;  Size = c.Size;  }  CDROM::~CDROM() {  cout << "CDROM Destructor" << endl;  }  //Computer  class Computer {  public:  Computer(CPU\_rank cpu\_rank, int cpu\_fre, double cpu\_vol, int ram\_size, int cd\_size);  Computer(Computer & c);  ~Computer();  void Run(); //公有成员函数RUN  void Stop(); //公有成员函数STOP  private:  CPU cpu; //CPU  RAM ram; //RAM  CDROM cdrom; //CDROM  };  //构造函数，接受五个参数，前三个赋值CPU,后三个给RAM和CDROM  Computer::Computer(CPU\_rank cpu\_rank, int cpu\_fre, double cpu\_vol, int ram\_size, int cd\_size): cpu(cpu\_rank, cpu\_fre, cpu\_vol), ram(ram\_size), cdrom(cd\_size) {  cout << "Computer Constructor" << endl;  }  //复制构造  Computer::Computer(Computer &c): cpu(c.cpu), ram(c.ram), cdrom(c.cdrom) {  cout << "Computer Copy Constructor" << endl;  }  //析构  Computer::~Computer() {  cout << "Computer Destructor" << endl;  }  void Computer::Run() {  cout << "Computer Run!" << endl;  }  void Computer::Stop() {  cout << "Computer Stop!" << endl;  }  int main() {  cout<<"Please enter the argument"<<endl;  int m\_cpu\_rank, m\_cpu\_fre, m\_cpu\_vol, m\_ram\_size, m\_cd\_size;  cin >> m\_cpu\_rank >> m\_cpu\_fre >> m\_cpu\_vol >> m\_ram\_size >> m\_cd\_size;  Computer c1((CPU\_rank)m\_cpu\_rank, m\_cpu\_fre, m\_cpu\_vol, m\_ram\_size, m\_cd\_size);  Computer c2(c1);  c1.Run();  c1.Stop();  return 0;  }  **运行结果：**  **三．People**  **实验要求：**  设计一个用于人事管理的People(人员)类。考虑到通用性这里只抽象出所有类型人员都具有的属性：number（编号）、sex（性别）、birthday（出生日期）、id（身份证号）等等。其中“出生日期”定义为一个“日期”类内嵌子对象。用成员函数实现对人员信息的录入和显示。要求包括：构造函数和析构函数、拷贝构造函数、内联成员函数、组合。  **实验内容：**  首先定义data类来表示日期，然后再定义People，来实现内嵌。  **源程序：**  #include<iostream>  #include<string>  using namespace std;  //data类  class data {  public:  data(int myyear,int mymon,int myday);//构造函数  data(data &a);//复制构造  ~data();//析构  inline void showData();//展示日期  private:  int year;  int mon;  int day;  };  data::data(int myyear,int mymon,int myday) {  cout<<"data construct"<<endl;  year = myyear;  mon = mymon;  day = myday;  }  data::data(data &a) {  cout<<"data copy construct"<<endl;  year=a.year;  mon=a.mon;  day=a.day;  }  data::~data()  {  cout<<"data destructor"<<endl;  }  void data::showData() {  cout<<year<<" "<<mon<<" "<<day<<" ";  }  //定义People类  class People {  public:  People(string mynum,string mysex,string myid,data birth);//构造函数  People(People &p);//复制构造  ~People();//析构函数  inline void showData();//内联，展示函数  private:  string number;  string sex;  string id;  data birth;  };  People::People(string mynum,string mysex,string myid,data mybirth):number(mynum),sex(mysex),id(myid),birth(mybirth) {  cout<<"People construct"<<endl;  }  People::~People()  {  cout<<"People destructor"<<endl;  }  void People::showData() {  cout <<"number: "<<number<<endl  <<"sex: "<<sex<<endl  <<"id: "<<id<<endl;  birth.showData();  cout<<endl;  }  People::People(People &p):number(p.number),sex(p.sex),id(p.id),birth(p.birth) {  cout<<"People Copy construct"<<endl;  }  int main() {  string mynum;  string mysex;  string myid;  int myyear;  int mymon;  int myday;  getline(cin,mynum);  getline(cin,mysex);  getline(cin,myid);  cin>>myyear>>mymon>>myday;  data mydata(myyear,mymon,myday);  People a(mynum,mysex,myid,mydata);  a.showData();//展示信息  People b(a);//复制另一个对象  return 0;  }  **运行结果：**    **心得体会**  1.类主要包括数据成员和成员函数两部分，数据成员大多是封装隐藏的，成员函数就是对外连接的接口。  2.构造函数要与类名一样，负责类中数据成员的初始化；析构函数名字即类前加‘~’，负责销毁数据。  3.先调用构造函数或复制构造函数，然后依据调用顺序执行其他成员函数，最后执行析构函数。  4.先构造小类，再构造包含子类的大类。  **实验二 数组、指针与字符串**  **实验目的：**  1.学习使用数组  2.学习字符串数据的组织和处理  3.学习标准C++库的使用  4.掌握指针的使用方法  5.练习通过Debug观察指针的内容及其所指的对象的内容  6.联系通过动态内存分配实现动态数组，并体会指针在其中的作用  7.分别使用字符数组和标准C++库练习处理字符串的方法  **一．**  **实验要求：**  编写并测试3\*3矩阵转置函数，使用数组保存3\*3矩阵。  **实验内容：**  编写矩阵转置函数，输入参数为3\*3整形数组，使用循环语句实现矩阵元素的行列对调，注意在循环语句中究竟需要对哪些元素进行操作，编写main（）函数实现输入、输出。程序名：lab6\_1.cpp。  **源程序：**  #include <iostream>  using namespace std;  int n[3][3];  void dao(int a[][3]);  int main() {  for(int i = 0; i < 3; i++)  for(int j = 0; j < 3; j++) {  cin >> n[i][j];  }  dao(n);  cout<<"转置后"<<endl;  for(int i = 0; i < 3; i++) {  for(int j = 0; j < 3; j++) {  cout << n[i][j]<<" ";  }  cout<<endl;  }  return 0;  }  void dao(int a[][3]) {  for(int i = 0; i < 3; i++)  for(int j = i; j < 3; j++) {  int temp=a[i][j];  a[i][j]=a[j][i];  a[j][i]=temp;  }  }  **运行结果：**  **二．**  **实验要求：**  使用动态内存分配生成动态数组来重新完成上题，使用指针实现函数的功能。  **实验内容：**  改写矩阵转置函数，参数为整型指针，使用指针对数组元素进行操作，在main（）函数中使用new操作符分配内存生成动态数组。通过Debug观察指针的内容及其所指的对象中的内容。程序名：lab6\_2.cpp。  **源程序：**  #include <iostream>  using namespace std;  void dao(int \*\*a);  int main() {  int \*\*n = new int\*[3];  for(int i = 0; i < 3; i++)  n[i] = new int[3];  for(int i = 0; i < 3; i++)  for(int j = 0; j < 3; j++) {  cin >> n[i][j];  }  dao(n);  cout << "转置后" << endl;  for(int i = 0; i < 3; i++) {  for(int j = 0; j < 3; j++) {  cout << n[i][j] << " ";  }  cout << endl;  }  for(int i=0;i<3;++i)  delete [] n[i];  return 0;  }  void dao(int \*\*a) {  for(int i = 0; i < 3; i++)  for(int j = i; j < 3; j++) {  int temp = \*(\*(a+i)+j);  \*(\*(a+i)+j)=\*(\*(a+j)+i);  \*(\*(a+j)+i) = temp;  }  }  **运行结果：**  三．  **实验要求：**  编程实现两字符串的连接。要求使用字符数组保存字符串，不要使用系统函数。  **实验内容：**  编程实现两字符串的连接。定义字符数组保存字符串，在程序中提示用户输入两个字符串，实现两个字符串的连接，最后用cout语句显示输出。程序名：lab6\_3.cpp。用cin实现输入，注意，字符串的结束标志是ASCII码0，使用循环语句进行字符串见的字符拷贝。  **源程序：**  #include <iostream>  using namespace std;  int main() {  cout<<"请输入两个字符串"<<endl;  char a[1000];  char b[1000];  char ch;  int i = 0;  //读取第一个字符串  while(cin.get(ch) && ch != '\n') {  a[i] = ch;  i++;  }  a[i]='\0';  int len1=i;//获取长度  i=0;  //读取第二个字符串  while(cin.get(ch) && ch != '\n') {  b[i] = ch;  i++;  }  int len2=i;//获取长度  //逐个复制  for(int i=len1;i<len1+len2;i++)  {  a[i]=b[i-len1];  }  a[len1+len2]='\0';  cout<<a<<endl;  return 0;  }  **运行结果：**  **四．**  **实验要求：**  使用string类定义字符串对象，重新实现上一小题。  **实验内容：**  使用string类定义字符串对象，编程实现两字符串的连接。在string类中已重载了运算符“+=”实现字符串的连接，可以使用这个功能。程序名：lab6\_4.cpp  **源代码：**  #include <iostream>  #include<string>//包含头文件  using namespace std;  int main() {  cout<<"请输入两个字符串"<<endl  string a;  string b;  getline(cin,a);//读取一行输入  getline(cin,b);  a+=b;//重载“+=”  cout<<a<<endl;  return 0;  }  **运行结果：**    **五．**  **实验要求：**  定义一个Employee类，其中包括姓名、街道地址、城市和邮编等属性，以及change\_name()和display()等函数。Display()显示姓名、街道地址、城市和邮编等属性，change\_name()改变对象的姓名属性。实现并测试这个类。  **实验内容：**  在employee.h文件中定义Employee类。Employee类具有姓名、街道地址、城市和邮编等私有数据成员，在成员函数中，构造函数用来初始化所有数据成员;display()中使用cout显示姓名、街道地址、城市和邮编等属性，change\_name()改变类中表示姓名属性的数据成员。在主程序中定义这个类的对象并对起进行操作。程序名：lab6\_5.cpp。  **源程序：**  #include <iostream>  #include<string>  using namespace std;  class Employee {  public:  Employee(string my\_name="", string my\_address="", string my\_city="", int my\_post=0);  void Display();  void change\_name(string my\_name);  private:  string name;  string address;  string city;  int post;  };  Employee::Employee(string my\_name, string my\_address, string my\_city, int my\_post) {  cout<<"Employee constructor"<<endl;  name = my\_name;  address = my\_address;  city = my\_city;  post = my\_post;  }  void Employee::Display() {  cout << "name: " << name << endl  << "address: " << address << endl  << "city: " << city << endl  << "post: " << post << endl;  }  void Employee::change\_name(string my\_name){  name=my\_name;  }  int main() {  string my\_name;  string my\_address;  string my\_city;  int my\_post;  getline(cin, my\_name);  getline(cin, my\_address);  getline(cin, my\_city);  cin >> my\_post;  Employee a( my\_name, my\_address, my\_city, my\_post);  a.Display();  a.change\_name("孔振华");  a.Display();  return 0;  }  **运行结果：**    **六．**  **实验要求：**  定义包含5个元素的对象数组，每个元素都是Employee类型的对象。  **实验内容：**  使用上一小题中定义的Employee类定义对象数组emp[5]，使用循环语句把数据显示出来。程序名：lab6\_6.cpp。  **源程序：**  在五中增加：  //定义五个元素的对象  Employee stu[5];  **运行结果：**    **七．**  **实验内容：**  修改实验4中的选做实验中的people（人员）类。具有的属性如下：姓名char name[11]、编号char number[7]、性别char sex[3]、生日birthday、身份证号char id[16]。其中“出生日期”定义为一个“日期”类内嵌对象。用成员函数实现对人员信息的录入和显示。要求包括：构造函数和析构函数、拷贝构造函数、内联成员函数、聚集。在测试程序中定义people类的对象数组，录入数据并显示。  **实验要求：**  **源程序：**  #include<iostream>  #include<string>  #include<cstring>  using namespace std;  class data {  public:  data();  data(int myyear, int mymon, int myday);  data(data &a);  ~data();  inline void showData();  private:  int year;  int mon;  int day;  };  data::data() {  }  data::data(int myyear, int mymon, int myday) {  cout << "data construct" << endl;  year = myyear;  mon = mymon;  day = myday;  }  data::data(data &a) {  cout << "data copy construct" << endl;  year = a.year;  mon = a.mon;  day = a.day;  }  data::~data() {  cout << "data destructor" << endl;  }  void data::showData() {  cout << year << " " << mon << " " << day << " ";  }  class People {  public:  People();  People(char \*myname, char \*mynum, char \*mysex, char \*myid, data mybirth);  People(People &p);  Set(char \*myname, char \*mynum, char \*mysex, char \*myid, data mybirth);  ~People();  inline void showData();  private:  char name[11];  char number[7];  char sex[3];  char id[16];  data birth;  };  People::People() {  }  People::People( char \*myname, char \*mynum, char \*mysex, char \*myid, data mybirth) {  cout << "People construct" << endl;  strcpy(name, myname);  strcpy(number, mynum);  strcpy(sex, mysex);  strcpy(id, myid);  birth = mybirth;  }  People::Set(char \*myname, char \*mynum, char \*mysex, char \*myid, data mybirth) {  strcpy(name, myname);  strcpy(number, mynum);  strcpy(sex, mysex);  strcpy(id, myid);  birth = mybirth;  }  People::~People() {  cout << "People destructor" << endl;  }  void People::showData() {  cout << "name: " << name << endl  << "number: " << number << endl  << "sex: " << sex << endl  << "id: " << id << endl;  birth.showData();  cout << endl;  }  People::People(People &p) {  cout << "People Copy construct" << endl;  strcpy(name, p.name);  strcpy(number, p.number);  strcpy(sex, p.sex);  strcpy(id, p.id);  birth = p.birth;  }  int main() {  char myname[11];  char mynum[7];  char mysex[3];  char myid[16];  int myyear;  int mymon;  int myday;  People a[5];  for(int i = 0; i < 5; i++) {  cin >> myname >> mynum >> mysex >> myid;  cin >> myyear >> mymon >> myday;  data mydata(myyear, mymon, myday);  a[i].Set(myname, mynum, mysex, myid, mydata);  a[i].showData();  }  return 0;  }  **运行结果：**  **心得体会：**   1. **字符串可以用字符数组或者string类来存储。** 2. **String.h是C风格的头文件，包含了C中字符数组的函数。String是C++风格的。** 3. **可以直接用下标表示法， 也可以使用指针。**   **实验三 继承与派生**  **实验目的：**  1.学习定义和使用类的继承关系，定义派生类  2.熟悉不同继承方式下对基类成员的访问控制  3.学习利用虚基类解决二义性问题  **一．**  **实验要求&实验步骤：**  定义一个基类Animal，有私有整型成员变量age，构造其派生类dog，在其成员函数SetAge（int n）中直接给age赋值，看看会有什么问题，把age改为公有成员变量，还会有问题吗？编程尝试。  **源程序：**  #include<iostream>  using namespace std;  class Animal {  public:  int age;  };  class dog : public Animal {  public:  void SetAge(int myage);  };  void dog::SetAge(int myage) {  age = myage;  }  int main() {  dog a;  a.SetAge(15);  return 0;  }  **运行结果：**    **二．**  **实验要求：**  定义一个基类BaseClass,有整型变量Number，构造其派生类DerivedClass，观察构造函数和析构函数的执行情况。  **实验内容&实验步骤:**  编写程序定义一个基类BaseClass，构造其派生类DerivedClass，在构造函数和析构函数中用cout输出提示信息，观察构造函数和析构函数的执行情况。程序名：lab7\_2.cpp。  **源程序：**  #include<iostream>  using namespace std;  class BaseClass {  public:  BaseClass();  ~BaseClass();  private:  int Number;  };  BaseClass::BaseClass() {  cout << "BaseClass constructor" << endl;  }  BaseClass::~BaseClass() {  cout << "BaseClass destructor" << endl;  }  class DerivedClass : public BaseClass {  public:  DerivedClass();  ~DerivedClass();  };  DerivedClass::DerivedClass() {  cout << "DeriedClass constructor" << endl;  }  DerivedClass::~DerivedClass() {  cout << "DeriedClass destructor" << endl;  }  int main() {  DerivedClass a;  return 0;  }  **运行结果：**    **三．**  **实验要求：**  定义一个车（vehicle）基类，具有Maxspeed、Weight等成员，Run、Stop等成员函数，由此派生出自行车（bicycle）类、汽车（motorcar）类。自行车（bicycle）类有高度（Height）等属性，汽车（motorcar）类有座位数（SeatNum）等属性。从bicycle和motorcar派生出摩托车（motorbicycle）类，在继承过程中，注意把vehicle设置为虚基类。如果不把vehicle设置为虚基类，会有什么问题？编程尝试。  **实验内容&实验步骤：**  编写程序定义一个车（vehicle）基类，由此派生出自行车（bicycle）类、汽车（motorcar）类，注意把vehicle派生为虚基类。再从bicycle和motorcar派生出摩托车（motorcycle）类，在main()函数中测试这个类。程序名:lab7\_3.cpp。编译成功后，把vehicle设置为非虚基类，再编译一次，此时系统报错，无法编译成功。这是因为若不把vehicle设置为虚基类，会出现二义性错误，程序不能成功编译。  **源程序：**  #include<iostream>  using namespace std;  class vehicle {  private:  int Maxspeed;  int Weight;  public:  void Run();  void Stop();  };  class bicycle: virtual public vehicle {  private:  double Height;  };  class motorcar: virtual public vehicle {  private:  int SeatNum;  };  class motorbicycle: public motorcar, public bicycle {  };  int main(void) {  motorbicycle a;  cout << a.Weight;  return 0;  }  void vehicle::Run() {  cout << "vehicle Run!" << endl;  }  void vehicle::Stop() {  cout << "vehicle Stop!" << endl;  }  **运行结果：**  **思考题**  1.采用继承的方式，公有继承，保护继承或者私有继承。  2.先构造基类，再构造派生类。  3.在第一次继承的时候，在继承方式前加“virtual”，即可实现虚基类继承,解决二义性问题。  **心得体会：**   1. 学会了继承和多继承的方法，明白了私有成员不能被派生类直接访问。 2. 学会了虚基类继承的方式，以及二义性问题的解决方法。 3. 明白了基类和派生类构造和析构的顺序。   **实验四 多态性**  **实验目的：**  **1.掌握运算符重载的方法**  **2.学习使用虚函数实现动态多态性**  **实验要求：**  **定义Point类，有坐标\_x，\_y两个成员变量；对Point类重载“＋＋”(自增)、“――”(自减)运算符，实现对坐标值的改变。**    **实验内容：**  **编写程序定义Point类，在类中定义整型的私有成员变量\_x\_y，定义成员函数Point& operator++()；Point operator++(int)；以实现对Point类重载“++”(自增)运算符，定义成员函数Point＆ operator－－()；Point operator－－(int)；以实现对Point类重载“－－”(自减)运算符，实现对坐标值的改变。程序名：1ab8\_1．cpp。**  **源程序：**  **#include<iostream>**  **using namespace std;**  **class Point {**  **public:**  **Point(int myx,int myy);**  **void ShowPoint();**  **Point& operator ++();**  **Point& operator --();**  **Point operator ++(int);**  **Point operator --(int);**  **private:**  **int \_x;**  **int \_y;**  **};**  **Point::Point(int myx,int myy) {**  **\_x=myx;**  **\_y=myy;**  **}**  **void Point::ShowPoint() {**  **cout<<"("<<\_x<<","<<\_y<<")"<<endl;**  **}**  **Point& Point::operator ++() {**  **\_x++;**  **\_y++;**  **return \*this;**  **}**  **Point& Point::operator --() {**  **\_x--;**  **\_y--;**  **return \*this;**  **}**  **Point Point::operator ++(int) {**  **Point temp = \*this;**  **\_x++;**  **\_y++;**  **return temp;**  **}**  **Point Point::operator --(int) {**  **Point temp = \*this;**  **\_x--;**  **\_y--;**  **return temp;**  **}**  **int main() {**  **Point a(1,2);**  **a.ShowPoint();**  **++a;**  **a.ShowPoint();**  **a++;**  **a.ShowPoint();**  **a--;**  **a.ShowPoint();**  **--a;**  **a.ShowPoint();**  **return 0;**  **}**  **运行结果：**    **实验要求：**  **定义一个车(vehiele)基类，有Run、Stop等成员函数，由此派生出自行车(bicycle)类、汽车(motorcar)类，从bicycle和motorcar派生出摩托车(motorcycle)类，它们都有Run、Stop等成员函数。观察虚函数的作用。**  **实验内容：**  **编写程序定义一个车(vehicle)基类，有Run、Stop等成员函数，由此派生出自行车(bicycle)类、汽车(motorcar)类，从bicycle和motorcar派生出摩托车(motorcycle)类，它们都有Run、Stop等成员函数。在main()函数中定义vehicle、bicycle、motorcar、motorcycle的对象，调用其Run()、Stop()函数，观察其执行情况。再分别用vehicle类型的指针来调用这几个对象的成员函数，看看能否成功；把Run、Stop定义为虚函数，再试试看。程序名：lab8\_2．cpp。**  **源程序：**  **#include<iostream>**  **using namespace std;**  **class vehiele {**  **public:**  **virtual void Run();**  **virtual void Stop();**  **};**  **void vehiele::Run() {**  **cout<<"vehiele Run!"<<endl;**  **}**  **void vehiele::Stop() {**  **cout<<"vehiele Stop!"<<endl;**  **}**  **class bicycle : virtual public vehiele {**  **public:**  **virtual void Run();**  **virtual void Stop();**  **};**  **void bicycle::Run() {**  **cout<<"bicycle Run!"<<endl;**  **}**  **void bicycle::Stop() {**  **cout<<"bicycle Stop!"<<endl;**  **}**  **class motocar : virtual public vehiele {**  **public:**  **virtual void Run();**  **virtual void Stop();**  **};**  **void motocar::Run() {**  **cout<<"motocar Run!"<<endl;**  **}**  **void motocar::Stop() {**  **cout<<"motocar Stop!"<<endl;**  **}**  **class motorcycle : virtual public bicycle , virtual public motocar {**  **public:**  **virtual void Run();**  **virtual void Stop();**  **};**  **void motorcycle::Run() {**  **cout<<"motorcycle Run!"<<endl;**  **}**  **void motorcycle::Stop() {**  **cout<<"motorcycle Stop!"<<endl;**  **}**  **int main() {**  **vehiele a;**  **bicycle b;**  **motocar c;**  **motorcycle d;**  **vehiele \* p = &a;**  **p->Run();**  **p = &b;**  **p->Run();**  **p = &c;**  **p->Run();**  **p = &d;**  **p->Run();**  **return 0;**  **}**  **运行结果：**    **实验要求：**  (选做)对实验2中的People类重载“＝＝”运算符和“－”运算符，“＝＝”运算符判断两个people类对象的id属性是否相等；“－”运算符实现People类对象的赋值操作。  **实验内容：**  **源程序：**  **#include<iostream>**  **#include<string>**  **using namespace std;**  **class People {**  **public:**  **People(string myid);**  **bool operator ==(const People &b);**  **bool operator -(const People &b); //Is it "="?**  **private:**  **string id;**  **};**  **People::People(string myid){**  **id = myid;**  **}**  **bool People::operator ==(const People &b){**  **bool flag=false;**  **if(id == b.id)**  **flag = true;**  **return flag;**  **}**  **bool People::operator -(const People &b){**  **id=b.id;**  **}**  **int main(){**  **People a("20174631");**  **People b("20174698");**  **People c("20174631");**  **cout<< boolalpha<<(a==b)<<endl;**  **cout<< boolalpha<<(a==c)<<endl;**  **b-c;**  **cout<<boolalpha<<(a==b)<<endl;**  **return 0;**  **}**  **运行结果：**    **思考题：**   1. **如何将一个运算符重载为类的成员函数？**   **答：在类中，使用 operator函数，第二个（或更多）参数表示其他参与运算的成员。**   1. **如何将一个运算符重载为类的友元函数？**   **答：在类中，使用friend来声明友元，括号内的参数都是参与运算的全部成员。**   1. **如何实现运行时刻的多态？**   **答：在父类中，定义的函数，被子类继承之后，可以具有不同的数据类型或表现出不同的行为，这使得同一个属性或者行为在父类及其各个子类中具有不同的语义。**  **心得体会：**  **灵活地运用虚函数，虚继承，以及多态，提高了代码重用性，在大规模编程的时候更加方便。** | | | | | | |