**中国矿业大学计算机学院实验报告**

|  |
| --- |
| 课程名称  高级语言程序设计           实验名称 继承与派生  班级 计算机科学与技术2019-3班 姓名 王杰永 学号 03190886  仪器组号\_\_\_\_\_\_\_  实验日期 2020.12.10  实验报告要求：1.实验目的   2.实验内容（题目描述，源代码，运行截图，调试情况）    3.实验体会 |
| 一、实验目的  \*\*\*\*  二、实验内容  1、第一题  1.1题目描述  题目描述  对本章示范题的用于管理商店商品的实现程序进行完善:完成Wardrobe立柜类的具体定义与使用，并添加“帽子仓库类”以及“立柜仓库类”的定义及使用，以使程序能够对商店的这三种商品（衬衣、帽子、立柜）进行简单的管理与应用。  要对商品实现的操作有:商品的进库（增加某类商品及其库存量），商品的出库（减少某类商品及其库存量），以及某类商品总价格的计算。  输入  60 Tianjin Cotton 80 Beijing Wool 40 Suzhou Cotton M 30 Wuxi Wool S 160 Guangzhou Pine Yellow 200 Suzhou Oak Brown  输出  (注意：红色部分是输入数据)  5 \* shirt data in: price/place/material =>60 Tianjin Cotton  3 \* shirt data in: price/place/material =>80 Beijing Wool  60 Tianjin Cotton  60 Tianjin Cotton  60 Tianjin Cotton  60 Tianjin Cotton  60 Tianjin Cotton  80 Beijing Wool  80 Beijing Wool  80 Beijing Wool  shiSto.TotalPrice()=540  60 Tianjin Cotton  60 Tianjin Cotton  60 Tianjin Cotton  60 Tianjin Cotton  shiSto.TotalPrice()=240  5 \* Cap data in: price/place/material/style =>40 Suzhou Cotton M  3 \* Cap data in: price/place/material/style =>30 Wuxi Wool S  40 Suzhou Cotton M  40 Suzhou Cotton M  40 Suzhou Cotton M  40 Suzhou Cotton M  40 Suzhou Cotton M  30 Wuxi Wool S  30 Wuxi Wool S  30 Wuxi Wool S  capSto.TotalPrice()=290  40 Suzhou Cotton M  40 Suzhou Cotton M  40 Suzhou Cotton M  40 Suzhou Cotton M  capSto.TotalPrice()=160  5 \* Wardrobe data in: price/place/material/color =>160 Guangzhou Pine Yellow  3 \* Wardrobe data in: price/place/material/color =>200 Suzhou Oak Brown  160 Guangzhou Pine Yellow  160 Guangzhou Pine Yellow  160 Guangzhou Pine Yellow  160 Guangzhou Pine Yellow  160 Guangzhou Pine Yellow  200 Suzhou Oak Brown  200 Suzhou Oak Brown  200 Suzhou Oak Brown  WarSto.TotalPrice()=1400  160 Guangzhou Pine Yellow  160 Guangzhou Pine Yellow  160 Guangzhou Pine Yellow  160 Guangzhou Pine Yellow  WarSto.TotalPrice()=640  1.2 源代码  #include<iostream>  #include<cstdio>  #include<cstring>  #include<algorithm>  #include<stdio.h>  using namespace std;  class Base  {  private:  int price; char place[10], material[10];  public:  Base(int pri, char\* pla, char\* mat)  {  price = pri;  strcpy(place, pla);  strcpy(material, mat);  }  int GetPrice() { return price; }  void Display()  {  cout << price << " " << place << " " << material;  }  };  class shirt :public Base  {  public:  shirt(int pri, char\* pla, char\* mat) :Base(pri, pla, mat) {}  void shirtDisplay() { Base::Display(); }  int GetPrice() { return Base::GetPrice(); }  };  class cap :public Base  {  private:  char style[10];  public:  cap(int pri, char\* pla, char\* mat, char\* sty) :Base(pri, pla, mat)  {  strcpy(style, sty);  }  int GetPrice() { return Base::GetPrice(); }  void Display()  {  Base::Display();  cout << " " << style;  }  };  class Wardrobe :public Base  {  private:  char color[10];  public:  Wardrobe(int pri, char\* pla, char\* mat, char\* col) :Base(pri, pla, mat)  {  strcpy(color, col);  }  int GetPrice() { return Base::GetPrice(); }  void Display()  {  Base::Display();  cout << " " << color;  }  };  int main()  {  int pri; char pla[10], mat[10], sty[10], col[10];  cout << "5 \* shirt data in: price/place/material =>";  cin >> pri >> pla >> mat; cout << endl;  shirt s1(pri, pla, mat);  cout << "3 \* shirt data in: price/place/material =>";  cin >> pri >> pla >> mat; cout << endl;  shirt s2(pri, pla, mat);  for (int i = 1; i <= 5; i++) { s1.Display(); cout << endl; }  for (int i = 1; i <= 3; i++) { s2.Display(); cout << endl; }  cout << "shiSto.TotalPrice()=" << (s1.GetPrice() \* 5 + s2.GetPrice() \* 3) << endl;  for (int i = 1; i <= 4; i++) { s1.Display(); cout << endl; }  cout << "shiSto.TotalPrice()=" << (s1.GetPrice() \* 4) << endl;  cout << "5 \* Cap data in: price/place/material/style =>";  cin >> pri >> pla >> mat >> sty; cout << endl;  cap c1(pri, pla, mat, sty);  cout << "3 \* Cap data in: price/place/material/style =>";  cin >> pri >> pla >> mat >> sty; cout << endl;  cap c2(pri, pla, mat, sty);  for (int i = 1; i <= 5; i++) { c1.Display(); cout << endl; }  for (int i = 1; i <= 3; i++) { c2.Display(); cout << endl; }  cout << "capSto.TotalPrice()=" << (c1.GetPrice() \* 5 + c2.GetPrice() \* 3) << endl;  for (int i = 1; i <= 4; i++) { c1.Display(); cout << endl; }  cout << "capSto.TotalPrice()=" << (c1.GetPrice() \* 4) << endl;  cout << "5 \* Wardrobe data in: price/place/material/color =>";  cin >> pri >> pla >> mat >> col; cout << endl;  Wardrobe w1(pri, pla, mat, col);  cout << "3 \* Wardrobe data in: price/place/material/color =>";  cin >> pri >> pla >> mat >> col; cout << endl;  Wardrobe w2(pri, pla, mat, col);  for (int i = 1; i <= 5; i++) { w1.Display(); cout << endl; }  for (int i = 1; i <= 3; i++) { w2.Display(); cout << endl; }  cout << "WarSto.TotalPrice()=" << (w1.GetPrice() \* 5 + w2.GetPrice() \* 3) << endl;  for (int i = 1; i <= 4; i++) { w1.Display(); cout << endl; }  cout << "WarSto.TotalPrice()=" << (w1.GetPrice() \* 4) << endl;  return 0;  }  1.3 运行截图    1.4 调试情况  本题的题目要求有些长，在写代码的时候经常出现敲错别字的情况。  2、第二题  2.1题目描述  题目描述  利用继承性与派生类来管理学生教师档案:由Person（人员）类出发（作为基类），派生出Student（学生）及Teacher（教师）类；而后又由Student（学生）类出发（作为基类），派生出GraduateStudent（研究生）类。可假定这几个类各自具有的数据成员为:  Person（人员）类: 姓名、性别、年龄；  Student（学生）类: 姓名、性别、年龄、学号、系别；  Teacher（教师）类: 姓名、性别、年龄、职称、担任课程；  GraduateStudent（研究生）类: 姓名、性别、年龄、学号、系别、导师。 为简化起见，每个类可只设立构造函数以及显示类对象数据的成员函数Print。而后编制简单的主函数，说明上述有关的类对象，并对其类成员函数进行简单使用（调用）  样例输出 [Copy](javascript:CopyToClipboard($('#sampleoutput').text()))  == per1.Display() => name,age,sex  sun 42 M  == stu1.Display() => name,age,sex,Reg\_Number,department  guo 22 F 1001 comp  == teach1.Display() => name,age,sex,course,post  fang 38 M english professor  == gStu.Display() => name,age,sex,Reg\_Number,department,advisor  wu 25 M 1021 comp wei  2.2 源代码  #include<iostream>  #include<string>  using namespace std;  class Person {  protected:  string name;  char gender;  int age;  public:  Person(string name, char gender, int age) {  this->name = name;  this->gender = gender;  this->age = age;  }  void Print() {  cout << name << " " << age << " " << gender << endl;  }  };  class Student :public Person {  protected:  string stuNum;  string department;  public:  Student(string name, char gender, int age, string stuNum, string department)  :Person(name, gender, age) {  this->stuNum = stuNum;  this->department = department;  }  void Print() {  cout << name << " " << age << " " << gender << " " << stuNum << " " << department << endl;  }  };  class Teacher :public Person {  protected:  string title;  string course;  public:  Teacher(string name, char gender, int age, string title, string course)  :Person(name, gender, age) {  this->title = title;  this->course = course;  }  void Print() {  cout << name << " " << age << " " << gender << " " << course << " " << title << endl;  }  };  class GraduateStudent :public Student {  protected:  string advisor;  public:  GraduateStudent(string name, char gender, int age, string stuNum, string department, string advisor)  :Student(name, gender, age, stuNum, department) {  this->advisor = advisor;  }  void Print() {  cout << name << " " << age << " " << gender << " " << stuNum << " " << department << " " << advisor << endl;  }  };  int main() {  string name = "sun";  char gender = 'M';  int age = 42;  Person per1(name, gender, age);  cout << "== per1.Display() => name,age,sex" << endl;  per1.Print();  name = "guo";  age = 22;  gender = 'F';  string stuNum = "1001";  string department = "comp";  Student stu1(name, gender, age, stuNum, department);  cout << "== stu1.Display() => name,age,sex,Reg\_Number,department" << endl;  stu1.Print();  name = "fang";  age = 38;  gender = 'M';  string course = "english";  string title = "professor";  Teacher teach1(name, gender, age, title, course);  cout << "== teach1.Display() => name,age,sex,course,post" << endl;  teach1.Print();  name = "wu";  age = 25;  gender = 'M';  stuNum = "1021";  department = "comp";  string advisor = "wei";  GraduateStudent gStu(name, gender, age, stuNum, department, advisor);  cout << "== gStu.Display() => name,age,sex,Reg\_Number,department,advisor" << endl;  gStu.Print();  return 0;  }  2.3 运行截图    2.4 调试情况  本题在最开始两次提交的时候没有注意，在前两行之间多打印了一个换行，导致错误。  3、第三题  3.1题目描述 题目描述 自定义一个日期时间类DateTimeType，它含有类DateType与类TimeType的类对象作为其数据成员，并具有所列的其他几个成员函数。而后编制主函数，说明DateTimeType的类对象，并对其成员函数以及二对象成员所属类的公有成员函数进行使用。  class DateTimeType {  //自定义的日期时间类 DateTimeType  DateType date; //类 DateType 的类对象 date 作为其数据成员  TimeType time; //类 TimeType 的类对象 time 作为其另一个数据成员  public:  DateTimeType(int y0=1, int m0=1, int d0=1, int hr0=0, int mi0=0, int se0=0);  //构造函数，设定 DateTimeType 类对象的日期时间，并为各参数设置了默认值  DateType& GetDate(){ return date; } //返回本类的私有数据对象 data  TimeType& GetTime(){ return time; } //返回本类的私有数据对象 time  void IncrementSecond(int s);  //增加若干秒，注意“进位”问题  void PrintDateTime(); //屏幕输出日期时间对象的有关数据  };  注意，每一个DateTimeType类对象中总包含有一个DateType类对象（对象成员）以及一个TimeType类对象（对象成员），编制与实现本程序时，也必须包含DateType与TimeType自定义类（类型）。  之所以设置了公有的类成员函数GetDate与GetTime，是为类外如主函数处使用该类的私有数据成员date与time提供方便（否则的话，类外无法直接访问该类的私有数据成员）。另外，两成员函数返回的都为引用，为的是可将返回对象当作一个独立变量来使用（如可以继续作左值等）。例如，假设编制了如下形式的主函数:  void main() {  DateTimeType dttm1(1999,12,31,23,59,59), dttm2;  (dttm1.GetDate()).PrintDate(); //调用对象成员所属类的公有成员函数  cout<<endl;  dttm1.PrintDateTime(); //调用本派生类的成员函数 PrintDateTime  dttm2.PrintDateTime();  dttm1.IncrementSecond(30) ; //调用本派生类成员函数  dttm1.PrintDateTime();  } 样例输出 [Copy](javascript:CopyToClipboard($('#sampleoutput').text())) 1999-12-31  1999-12-31 23:59:59  1-1-1 0:0:0  2000-1-1 0:0:29  3.2 源代码  #include<iostream>  #include<string>  using namespace std;  int month[12] = { 31,28,31,30,31,30,31,31,30,31,30,31 };  class DateType {  int y, m, d;  public:  void setDate(int y, int m, int d) {  this->y = y;  this->m = m;  this->d = d;  }  void IncrementDay();  void PrintDate();  };  void DateType::IncrementDay() {  if (y % 4 == 0 && y % 100 != 0 || y % 400 == 0)  month[1] = 29;  if (d == month[m - 1]) {  d = 1;  if (m == 12) {  m = 1;  y++;  }  else  m++;  }  else  d++;  month[1] = 28;  }  void DateType::PrintDate() {  cout << y << "-" << m << "-" << d;  }  class TimeType {  public:  int h;  int m;  int s;  public:  void setTime(int h, int m, int s) {  this->h = h;  this->m = m;  this->s = s;  }  void PrintTime() {  cout << h << ":" << m << ":" << s;  }  };  class DateTimeType {  DateType date;  TimeType time;  public:  DateTimeType(int y0 = 1, int m0 = 1, int d0 = 1, int hr0 = 0, int mi0 = 0, int se0 = 0) {  date.setDate(y0, m0, d0);  time.setTime(hr0, mi0, se0);  }  DateType& GetDate() { return date; }  TimeType& GetTime() { return time; }  void IncrementSecond(int s)  {  long total = time.h \* 3600 + time.m \* 60 + time.s;  int xx = (total + s) / 86400;  for (int i = 0; i < xx; i++)  date.IncrementDay();  long newtotal = (total + s) % 86400;  int newh = newtotal / 3600;  newtotal -= (newh \* 3600);  int newm = newtotal / 60;  newtotal -= (newm \* 60);  int news = newtotal;  time.h = newh;  time.m = newm;  time.s = news;  }  void PrintDateTime()  {  date.PrintDate(); cout << " ";  time.PrintTime(); cout << endl;  }  };  int main() {  DateTimeType dttm1(1999, 12, 31, 23, 59, 59), dttm2;  (dttm1.GetDate()).PrintDate();  cout << endl;  dttm1.PrintDateTime();  dttm2.PrintDateTime();  dttm1.IncrementSecond(30);  dttm1.PrintDateTime();  return 0;  }  3.3 运行截图    3.4 调试情况  本题直接将上周的实验课写好的DateType类粘贴了过来，也就是说，已经写好了日期自增一天的函数。于是，在本题DateTimeType类的自增函数中，只需要考虑到时间的进位即可，通过时间计算发现如果需要日期进位，直接调用日期进位的函数。  三、实验体会  本周的实验再次体会到了面向对象的编程思想，将一个个要实现的方法封装到类中，函数的实现与主函数分离开，条理清楚。 |