淮 阴 工 学 院 课 程 考 试 试 卷 参 考 答 案

专业：计算机类专业　课程名称:C++程序设计实验(下)　学分：1　试卷编号(A)
课程编号：1319623　考试方式：考查　考试时间：120　分钟
拟卷人(签字):于永源　拟卷日期：2017.06.08　审核人(签字)：于永源

注意：考试采用机试方式。首先，创建考生文件夹（考生文件夹命名为：学号+姓名+班级，如 1234567890 张三计算机 1161），然后，将 1 至 5 题的程序源代码文件分别以 Project01.cpp, Project02.cpp, Project03.cpp, Project04.cpp, Project05.cpp 命名，并放入考生文件夹中，最后使用极域将考生文件夹提交至教师机。确认提交无误后，方可离开考场。

得分统计表：

题 号	一	二	三	四	五	六	七	八	九	十	总 分
得 分											

得分 ____

一、程序设计题：（每题 20 分，共 5 题）

一、类与对象

计算机在表示简单几何图形时，通常采用顶点表示的形式，即：只表示出几何图形上的几个顶点，然后通过顶点之间的有序连接来刻画几何图形。在计算机中，可以通过下面的类来表示一个顶点：
顶点类 **Vertex** 的定义：
(1) 私有数据成员：横坐标 (double x)、纵坐标 (double y)；
(2) 构造函数：**Vertex**(double _x, double _y)，对顶点的横、纵坐标进行初始化；
(3) 成员函数：double getX() const，获取顶点的横坐标信息；
(4) 成员函数：double getY() const，获取顶点的纵坐标信息；
(5) 成员函数：void draw() const，显示顶点信息 (显示格式： <x, y>)。

计算机在表示线段时，只需描述它的两个顶点信息即可。下面的类可以用来表示一个线段：
线段类 **Segment** 的定义：
(1) 私有数据成员：线段的两个顶点 (**Vertex** start; **Vertex** end;)、线段的颜色 (char *color)；
(2) 构造函数：**Segment**(double x1, double y1, double x2, double y2, char *_color)，对线段的两个顶点以及线段的颜色进行初始化；
(3) 析构函数：**~Segment**()，释放存储线段颜色信息的堆空间；
(4) 成员函数：double length() const，计算线段的长度;
(5) 成员函数：void draw() const，显示线段信息 (显示示例：红色: <10, 20>---<30, 40>)。

设计一个程序，测试线段类的所有功能。

【参考答案】

```
#include <iostream>
#include <cstring>
#include <cmath>
using namespace std;
class Vertex
{
public:
  Vertex(double _x, double _y): x(_x), y(_y)    { }
  double getX() const    { return x; }
  double getY() const    { return y; }
  void draw() const    { cout<<"<"<<x<<","<<y<<">"; }
private:
  double x;
  double y;
};
class Segment
{
public:
  Segment(double x1, double y1, double x2, double y2, char *_color): start(x1, y1), end(x2, y2)
  {
      color = new char[strlen(_color)+1];
      strcpy(color, _color);
  }
  ~Segment()
  {
      if(color!=0)
          delete [] color;
  }
  double length() const
  {
      return sqrt((start.getX()-end.getX())*(start.getX()-end.getX())+
              (start.getY()-end.getY())*(start.getY()-end.getY()));
  }
  void draw() const
  {
      cout<<color<<": ";
      start.draw();
```

学号 姓名 班级 装 订 线

```
        cout<<"---";
        end.draw();
        cout<<endl;
  }
private:
  Vertex start;
  Vertex end;
  char *color;
};
int main()
{
  Segment seg(10, 20, 30, 40, "红色");
  seg.draw();
  cout<<"长度: "<<seg.length()<<endl;
  return 0;
}
```

二、继承与多态

随着信息技术的不断发展，用户需求的不断提高，计算机软件也在日益更新。新的软件不仅要提供新的功能，还要向下兼容旧软件的所有功能。Geometry 软件最初设计用于计算二维图形的几何信息。例如：Geometry 软件中的 Circle 模块用于计算圆形的几何信息。Circle 模块由如下类进行表示：

圆形类 **Circle** 的定义：

(1) 保护数据成员：圆形的半径 (double radius)；

(2) 构造函数：**Circle**(double _radius)，对圆形的半径进行初始化；

(3) 虚成员函数：**virtual** double area() const，计算圆形的面积。

随着三维图形的广泛应用，Geometry 软件将 Circle 模块分别升级成 Ball 模块和 Cylinder 模块，分别用于计算球体和圆柱体的几何信息。Ball 模块和 Cylinder 模块分别由如下类进行表示：

球体类 **Ball** 的定义：公有继承 **Circle** 类

(1) 构造函数：**Ball**(double _radius)，对球体的半径进行初始化；

(2) 成员函数：double area() const，派生类重新实现基类的虚函数，计算球体的表面积；

(3) 成员函数：double volume() const，计算球体的体积。

圆柱体类 **Cylinder** 的定义：公有继承 **Circle** 类

(1) 保护数据成员：圆柱体的高度 (double height)；

(2) 构造函数：**Cylinder**(double _radius, double _height)，对圆柱体的半径和高度进行初始化；

(3) 成员函数：double area() const，派生类重新实现基类的虚函数，计算圆柱体的表面积；

(4) 成员函数：double volume() const，计算圆柱体的体积。

通过以下程序来测试 Geometry 软件的兼容性和可扩展性：

```
int main()
{
      Ball ball(13.14);
      Cylinder cylinder(13.14, 20);
      Circle *base = &ball;
      cout<<“球体表面积：”<<base->area()<<endl;
      cout<<“球体体积：”<<ball.volume()<<endl;
      base = &cylinder;
      cout<<“圆柱体表面积：”<<base->area()<<endl;
      cout<<“圆柱体体积：”<<cylinder.volume()<<endl;
      return 0;
}
```

【参考答案】

```
#include <iostream>
using namespace std;
const double PI = 3.14;
class Circle
{
public:
  Circle(double _radius): radius(_radius)  {  }
  virtual double area() const
  {
        return PI*radius*radius;
  }
protected:
  double radius;
};
class Ball: public Circle
{
public:
  Ball(double _radius): Circle(_radius)
  { }
  double area() const
  {
        return 4*PI*radius*radius;
  }
  double volume() const
```

```
 {
        return 4.0/3*PI*radius*radius*radius;
 }
};
class Cylinder: public Circle
{
public:
 Cylinder(double _radius, double _height): Circle(_radius), height(_height)
 { }
 double area() const
 {
        return 2*PI*radius*radius+2*PI*radius*height;
 }
 double volume() const
 {
        return PI*radius*radius*height;
 }
protected:
 double height;
};
int main()
{
 Ball ball(13.14);
 Cylinder cylinder(13.14, 20);
 Circle *base = &ball;
 cout<<"球体表面积：  "<<base->area()<<endl;
 cout<<"球体体积：  "<<ball.volume()<<endl;
 base = &cylinder;
 cout<<"圆柱体表面积：  "<<base->area()<<endl;
 cout<<"圆柱体体积：  "<<cylinder.volume()<<endl;
 return 0;
}
```

三、运算符重载

在黑市交易中，为了保护买卖双方的交易隐私，所交易的物品通常并不是直接明码标价的。买卖双方是通过手势来进行要价和还价的。卖家通过手势来告知买家某个物品的标价，买家则可以通过手势来表明自己的心里价位，或买卖双方通过手势在某一价位的基础上进行加价或减价。例如：下面的手势分别表示数字 0-9。


1


2


3


4


5


6


7


8


9


0

下面的手势表示在某一价位上进行加价或减价，具体加价或减价的金额则通过手势另外进行表示。


加价


减价

在黑市交易中，所交易的物品可由下面的类进行表示：
物品类 **Product** 的定义：
(1) 私有数据成员：物品名 (char *name)、物品标价 (double price)；
(2) 构造函数：**Product**(char *_name, double _price)，对物品名和物品标价进行初始化；
(3) 析构函数：**~Product**()，释放存储物品名的堆空间；
(4) 成员函数：void print() const，打印物品信息 (打印示例：清明上河图: ￥5000000)；
(5) **加法运算符重载成员函数**：对物品进行加价操作，具体加价金额由形参 money 的值决定；
(6) **减法运算符重载成员函数**：对物品进行减价操作，具体减价金额由形参 money 的值决定。

设计一个程序，通过对 Product 类的测试来模拟黑市交易的过程。

【参考答案】

```
#include <iostream>
#include <cstring>
using namespace std;
class Product
{
public:
  Product(char *_name, double _price): price(_price)
  {
```

```
    name = new char[strlen(_name)+1];
    strcpy(name, _name);
}
~Product()
{
    if(name!=0)
        delete [] name;
}
void print() const
{
    cout<<name<<": ￥"<<price<<endl;
}
Product operator+(double money) const
{
    return Product(name, price+money);
}
Product operator-(double money) const
{
    return Product(name, price-money);
}
Product& operator=(const Product& prod)
{
    price = prod.price;
    name = new char[strlen(prod.name)+1];
    strcpy(name, prod.name);
    return *this;
}
private:
  char *name;
  double price;
};
int main()
{
  Product prod("清明上河图", 5000000);
  prod = prod + 5000;
  prod = prod - 200;
  prod.print();
  return 0;
}
```

四、模板

数学家欧阳洵一直致力于通用算法的研究。他指出，实际有效的算法不能受限于具体的数据类型以及算法所执行的物理环境，真正通用的算法要能做到“以不变应万变”。近期，数学家欧阳洵正在探索“集合极化性”问题。该问题描述如下：

令集合中的数据总数为 n (n>3)，集合中的最大值与最小值分别为 maxv 和 minv。并将最大值与最小值的平均值 pivot=(maxv+minv)/2 作为集合的支点。然后统计集合中的数据比 pivot 大的数量，并令统计结果为 m。若 2m>n+2，则该集合为“正向极化”；若 2m<n-2，则该集合为“负向极化”；若 n-2≤2m≤n+2，则该集合为“零极化”。

设计一个函数模板 void polarity(**T** arr[], int n)，帮助数学家欧阳洵来实现求解集合极化性问题的通用算法，并分别通过 int a[10] = {2, 3, 5, 17, 8, 20, 39, 28, 55, 66}和 double b[10] = {0.3, 5.1, 6.8, 7.7, 3.7, 5.6, 4.8, 2.9, 5.3, 4.2}两个集合的数据进行测试。

【参考答案】

```
#include <iostream>
using namespace std;
template <class T>
void polarity(T arr[], int n)
{
  T maxv = arr[0];
  T minv = arr[0];
  for(int i=1; i<n; ++i)
  {
      if(arr[i]>maxv)
          maxv = arr[i];
      if(arr[i]<minv)
          minv = arr[i];
  }
  T pivot = (maxv+minv)/2;
  int m = 0;
  for(int i=0; i<n; ++i)
      if(arr[i]>pivot)
          ++m;
  if(2*m>n+2)
      cout<<"正向极化"<<endl;
  else if(2*m<n-2)
      cout<<"负向极化"<<endl;
  else
```

```
        cout<<"零极化"<<endl;
}
int main()
{
  int a[10] = {2, 3, 5, 17, 8, 20, 39, 28, 55, 66};
  double b[10] = {0.3, 5.1, 6.8, 7.7, 3.7, 5.6, 4.8, 2.9, 5.3, 4.2};
  polarity(a, 10);
  polarity(b, 10);
  return 0;
}
```

五、文件

在互联网技术高度发达的今天，网络上的信息交互时时刻刻都在进行着。在这些海量信息中，有些是无关紧要的私人信息，有些则是高度涉密的敏感性信息。如何对重要信息进行有效地保密，是密码学研究领域所关注的焦点问题。日前，世界银行行长 Robert Chapman 需要向联合国秘书长汇报 2016 年每个月份的银行存款与贷款情况。由于这些高度涉密数据不能被外界知道，因此，Robert Chapman 行长对数据进行了简单的加密处理，并将加密后的数据存放在文件中，然后通过互联网将存有加密数据的文件发送给联合国秘书长。Robert Chapman 行长的加密过程如下：

(1) 将 2016 年 12 个月份的银行存款数据分别进行如下操作：deposit/10000+260；

(2) 将加密后的银行存款数据写入到 **D:\\deposit.txt 文本文件**中；

(3) 将 2016 年 12 个月份的银行贷款数据分别进行如下操作：loan/2000-125；

(4) 将加密后的银行贷款数据写入到 **D:\\loan.bin 二进制文件**中；

联合国秘书长在收到两份文件后，通过读取文件中的数据并对其进行解密便可以获得真实的银行存款与贷款数据。

设计一个程序，利用存款数据 double deposit[12] = {100000, 200000, 300000, 400000, 500000, 600000, 700000, 800000, 900000, 1000000, 1100000, 1200000}和贷款数据 double loan[12] = {200000, 400000, 600000, 800000, 1000000, 1200000, 1400000, 1600000, 1800000, 2000000, 2200000, 2400000}来验证 Robert Chapman 行长所采用的数据加密算法的安全性与正确性，并将解密后的数据输出显示到屏幕上。

【参考答案】

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
  double deposit[12] = {100000, 200000, 300000, 400000, 500000, 600000, 700000, 800000, 900000,
1000000, 1100000, 1200000};
  double loan[12] = {200000, 400000, 600000, 800000, 1000000, 1200000, 1400000, 1600000,
1800000, 2000000, 2200000, 2400000};
  for(int i=0; i<12; ++i)
  {
        deposit[i] = deposit[i]/10000+260;
        loan[i] = loan[i]/2000-125;
  }
  ofstream fout("D:\\deposit.txt");
  if(!fout.fail())
        for(int i=0; i<12; ++i)
              fout<<deposit[i]<<" ";
  fout.close();
  fout.open("D:\\loan.bin",ios::binary);
  if(!fout.fail())
        fout.write((char *)loan, sizeof(loan));
  fout.close();
  ifstream fin("D:\\deposit.txt");
  if(!fin.fail())
        for(int i=0; i<12; ++i)
        {
              fin>>deposit[i];
              deposit[i] = 10000*(deposit[i]-260);
              cout<<deposit[i]<<" ";
        }
    cout<<endl;
    fin.close();
    fin.open("D:\\loan.bin",ios::binary);
    if(!fin.fail())
          fin.read((char *)loan, sizeof(loan));
    fin.close();
    for(int i=0; i<12; ++i)
    {
          loan[i] = 2000*(loan[i]+125);
          cout<<loan[i]<<" ";
    }
    cout<<endl;
    return 0;
}
```