座号

2018 学年度冬季学期试卷(A 卷)

课程名: 面向对象程序设计 课程号:08305121 学分: 5

得分

一、判断题（每小题 2 分，共 20 分）

1. 编译系统为类的非静态成员函数添加一个隐含的指针形参 **this**，该指针的指向不能被改变，即表达式 **this**++是错误的。 ( )
2. 有指针变量定义及初始化 **int** *p=**new int**[10];执行 **delete** [] p;操作将结束指针变量 p 的生命期。 ( )
3. 程序中的类模板并不参加编译。由类模板生成的模板类(被称为第一次实例化)参加编译。模板类创建的对象被称为第二次实例化，该对象可承载数据。 ( )
4. 声明类 A 为类 B 的友类，则类 A 的所有成员函数皆为类 B 的友元函数。 ( )
5. 类的静态成员函数必须通过该类的对象才能被调用（即给对象发消息访问静态成员函数）。 ( )
6. 重载运算符函数时，运算符函数的形参可以带默认值。 ( )
7. 函数在调用时，将创建值传递的形参，并用实参初始化该形参，函数返回时销毁该形参。 ( )
8. 引用返回的函数所返回的是一个变量（或对象）。该变量（或对象）的生命期应该长于函数的本次调用的执行期。 ( )
9. 派生类的成员函数可以访问基类的所有成员，包括访问属性为 **private** 的。 ( )
10. 含有纯虚函数的类被称为抽象类。可以创建抽象类的对象、声明抽象类的引用、以及定义目标类型为抽象类的指针变量。 ( )

得分

二、填空题（每空 2 分，共 20 分）

请根据运行结果，完成程序。

```
#include <iostream>
#include <cmath>
using namespace ①_____;
class Complex
{
public:
   Complex(double real②___, double imag=0):re(③_____),im(④_____)
   {
   }
   friend ostream & operator<<(ostream ⑤_____, const Complex &c)
   {
      out << '(' << c.re << ", " << c.im << ')';
      return ⑥____;
   }
   friend istream & operator>>(istream ⑦_____, Complex &c)
   {
      char str[200];
      in.getline(str, 200, '(');
      in >> c.re;
      in.getline(str, 200, ',');
      in >> c.im;
      in.getline(str, 200, ')');
      return in;
   }
   ⑧_____ Real()
   {
      return re;
   }
   double & Imag()
   {
      return im;
   }
   double abs() const                    // 复数的模长
   {
      return sqrt(re*re+im*im);
   }
   ⑨_____Complex operator*(const Complex &c1, const Complex &c2)
   {
      Complex temp;
      temp.re = c1.re*c2.re - c1.im*c2.im;
      temp.im = c1.re*c2.im + c1.im*c2.re;
      return temp;
   }
```

```
   ⑩ _____operator*=(const Complex &c)
   {
      return *this = *this * c;
   }
private:
   double re, im;
};

int main()
{
   Complex c, c1(3), c2(3, 4);
   cout << c << '\n' << c1 << '\n' << c2 << endl;
   cout << c2.abs() << endl;
   cout << c2*c2 << endl;
   c.Real() = 3;
   c.Imag() = 4;
   c *= c;
   cout << c << endl;
   return 0;
}
```

运行结果

```
(0, 0)
(3, 0)
(3, 4)
5
(-7, 24)
(-7, 24)
```

得分 ____

三、阅读程序写出运行结果（每行 1 分，共 30 分）

3.1（10 分）

```
#include <iostream>
using namespace std;

template <typename T> class Point
{
public:
   Point(const T &x=0, const T &y=0);
   Point(const Point<T> &p);
   ~Point();
   T & operator[](int index);
   template <typename TYPE> friend
      ostream & operator<<(ostream &out, const Point<TYPE> &p);
protected:
   T _x, _y;
};

template <typename T>
Point<T>::Point(const T &x, const T &y) : _x(x), _y(y)
{
   cout << "构造对象 " << *this << endl;
}
template <typename T>
Point<T>::Point(const Point<T> &p) : _x(p._x), _y(p._y)
{
   cout << "拷贝构造对象 " << *this << endl;
}
template <typename T>
Point<T>::~Point()
{
   cout << "析构对象 " << *this << endl;
}
template <typename T>
T & Point<T>::operator[](int index)
{
   if(index==0) return _x;
   else return _y;
}
template <typename TYPE>
ostream & operator<<(ostream &out, const Point<TYPE> &p)
{
   out << "(" << p._x << ", " << p._y << ")";
   return out;
}

int main()
{
   Point<int> a(-1, 2), b(a);
   Point<unsigned int> u(1, 2);
   Point<char> c('a', 'b');
   Point<double> d(3.5, 5.5);

   a[0] = 10;
   b[0] = 20;
   c[1] = 'B';
   d[1] = 0;
   u[0] = 2;
   return 0;
}
```

运行结果(3.1)

3.2（10 分）

```
#include <iostream>
using namespace std;

class RMB
{
public:
```

```
RMB(unsigned int x=0)
{
   yuan = x / 100;
   jiao = x / 10 % 10;
   fen  = x % 10;
}
operator unsigned int() const
{     // 类型转换函数，可将 RMB 类型的对象转换成 unsigned int 类型数据
   return 100*yuan+10*jiao+fen;
}
RMB & operator++()
{
   return *this = *this + 1;
}
RMB operator++(int)
{
   RMB temp(*this);
   ++(*this);
   return temp;
}
friend ostream & operator<<(ostream & out, const RMB & r)
{
   out <<"¥"<< r.yuan <<"元"<< r.jiao <<"角"<< r.fen <<"分";
   return out;
}
protected:
   unsigned int yuan, jiao, fen;
};

int main()
{
   RMB rmb(12345);
   cout << rmb << endl;
   rmb = 100;
   cout << rmb << endl;
   cout << ++(++rmb) << endl;
   cout << rmb++ << endl;
   cout << rmb << endl;
   rmb = 100;
   cout << 2*rmb << endl;
   rmb = 2*rmb;
   cout << rmb << endl;
   rmb = 100;
   cout << rmb/2 << endl;
   cout << RMB(rmb/2) << endl;
   cout << rmb << endl;
   rmb = 11;
   rmb = rmb*rmb;
   cout << rmb << endl;
   cout << ((rmb > 80)? "" : "不") << "大于 80 分" << endl;
   return 0;
}
```

运行结果(3.2) 部分输出结果已给出

200

¥2 元 0 角 0 分

3.3（10 分）

```
#include <iostream>
#include <cstring>
using namespace std;
class Animal
{
public:
   Animal(double weight=0, double age=0) : w(weight), a(age)
   {
      cout << "Constructing an Animal." << endl;
   }
   virtual  ~Animal()
   {
      cout << "Destructing an Animal." << endl;
   }
   void Print()
   {
      cout << "[ANIMAL] ";                    // 此处无换行
   }
   virtual void Show()
   {
      cout << "An Animal (" << w << "Kg, "
           << a << " years old). " << endl;
   }
protected:
   double w, a;                               // 体重、年龄
};

class Cat : public Animal
{
public:
   Cat(char *pName="NoName", double weight=0, double age=0)
                                          : Animal(weight,age)
   {
      strncpy(name, pName, sizeof(name));
      name[sizeof(name)-1] = '\0';
      cout<<"Constructing a Cat, " <<name<< " created."<<endl;
   }
```

```
    ~Cat()
    {
        cout<<"Destructing a Cat, " <<name<< " deleted."<<endl;
    }
    void Print()
    {
        cout << "[CAT] ";                          // 此处无换行
    }
    void Show()
    {
        cout  << name << ", a cat (" << w << "Kg, "
              << a << " years old). Meow..." << endl;
    }
protected:
    char name[20];
};

int main()
{
    Animal *pa;
    Cat *pc;
    pa = new Cat("Tom", 1, 2);
    pc = new Cat("Frisky", 3, 4);
    pa->Print(); pa->Show();
    pc->Print(); pc->Show();
    delete pa;
    delete pc;
    return 0;
}
```

运行结果(3.3)

得分 ____

四、（30 分）完成类的设计。设计 n 维向量类（**class** Vector;）要求如下。

① （2 分）向量的维数根据创建对象的需要而定，向量分量的数据类型为 **double**；

② （16 分，每个函数 4 分）默认的构造函数构造 0 维向量且转换构造函数构造指定整数维的 0 向量（即向量的各个分量皆为 0）、拷贝构造函数实现深拷贝、赋值运算符函数实现深赋值运算、析构函数释放向量占用的堆资源；

③ （12 分，每个函数 3 分）设计并实现 4 个成员函数 max、min、range、average 分别计算向量各分量的最大值、最小值、极差（最大值-最小值）和算术平均值（0 维向量的最大值、最小值、极差、算术平均值皆规定为 0）；

④ 该类具有较好的容错性能，如：各成员函数皆能妥善处理 0 维向量的情形。