1、 名词解释

函数重载：

函数缺省参数：

常成员函数：

内联函数 inline：

引用：

封装：

继承：

静态成员：

构造函数：

拷贝构造函数：

构造函数初始化表：

析构函数：

多态：

虚函数：

纯虚函数：

抽象类：

运算符重载：

2、 简答题：

1) 简述 struct 与 class 的区别。

2) 简述虚函数与一般成员函数的区别

3) 简述静态成员函数与一般成员函数的区别。

4) 简述 private、protected、public 的作用

5) 简述用 C++编写程序的过程

6) 简述虚析构函数的作用

7) 面向对象程序设计方法与面向过程的方法相比，有什么优点

8) C++语言有什么样的特点

3、 改错：

1) 请指出一下程序中的 3 处错误

```
class  MyClass
{
public:
    MyClass(int ini) { member = ini; }
    void SetMember(int m) { member = m;}
    int  GetMember() const { return member; }
private:
    int member;
};
```

```
int main()
{
    MyClass obj1;
    MyClass obj2(3);
    obj1.member = 5;
    MyClass.SetMember(10);
    return 0;
}
```

2) 请找出以下程序中的五处错误，指出错误的行数，并改正

```
#include <iostream.h>
using namespace std;
class YearMonth {
public:
    YearMonth(int year, int month)
    : year_(year), month_(month)
    { }

    virtual void print() {
        cout << year_ << ":" << month_;
    }
private:
    int year_, month_;
};

class Date : public YearMonth{

    Date(int year, int month, int day)
    : year_ (year), month_ (month), day_(day)
    { }

    void print() {
        cout << year_ <<":"<< month_ <<":"<< day_;
    }
protected:
    int day_;
}
```

```
    void main()
    {
        Date date(1979, 9, 13);
        date.print();
    }
```

4、 阅读以下程序，并给出程序运行的输出

```
#include <iostream>
using namespace std;
class complex {
public:
    complex(double r, double i = 0) : real_(r), imag_(i){
    }

    double imag() const{
        return imag_;
    }

    double real() const{
        return real_;
    }

    friend ostream & operator<<(ostream &os, const complex & c) {
        if(c.imag_ == 0)
            os << c.real_;
        else
            os << "("<< c.real_ << "+" << c.imag_ << "i)";
    }
private:
    double real_;
    double imag_;
};

complex operator+(const complex &a, const complex &b) {
    return complex(a.real() + b.real(), a.imag() + b.imag());
}
```

```
int main() {
    complex a(1.5);
    complex b(2.5, 4.5);
    cout << "a = " << a << endl;
    cout << "b = " << b << endl;
    cout << a + b << endl;
    return 0;
}
```

5、 阅读下列程序，回答后面提出的问题。

```
#include <iostream>
using namespace std;
class Vehicle
{
private:
      int color;
public:
     void SetColor(int c) { color = c; }
     virtual void Move() {cout <<  “Vehicle moving !\n” ;   }
};

class Car : public Vehicle{
public:
     void Move() {cout <<  “Car moving !\n” ;}
};

class Bus : public Vehicle
{
public:
     void Move() {cout <<  “Bus moving !\n” ;}
};

class SportsCar : public Car
{
```

```
};

class Coupe : public Car
{
public:
     void Move() {cout <<  “Coupe moving !\n” ;}
};

int main()
{
    Vehicle *vec;
    vec = new Car;
    vec->Move();
    delete vec;

    vec = new Bus;
    vec->Move();
    delete vec;

    vec = new SportsCar;
    vec->Move();
    delete vec;

    vec = new Coupe;
    vec->Move();
    delete vec;
    return 0;
}
```

（1）　写出程序的运行结果。

（2）如果将 Vehicle 的成员函数 Move 的定义改为：

```
virtual void Move()  { cout <<  “Vehicle moving !\n” ;}
```

写出修改后的程序的运行结果。

6、 请按如下要求补充程序

声明一个 Shape 抽象类，在此基础上派生出 Rectangle 类（矩形）和 Circle 类（圆），二者都有 GetArea()函数计算对象的面积，请按下面的要求给出 Shape 类、Rectangle 类、Circle 类的实现。

（ 1 ）Rectangle 类有 Width 宽度，Height 高度属性；

（ 2 ）Circle 类有 Radius 半径属性；

（ 3 ）Rectangle 类和 Circle 类的构造函数用以初始化上述参数；

（ 4 ）每个类都有 GetArea 函数，用以计算此形状的面积；

（ 5 ）主函数（如下）已经写好，对 Rectangle、Circle 类进行测试，使它们以统一的操作界面输出面积。

```
#include <iostream>
using  namespace  std;
class Shape;
class Rectangle;
class Circle;
int main()
{
    Shape *shapes[5];
    shape[0] =  new Rectangle(1.0, 2.0);
    shape[1] =  new Circle(5.0);
    shape[2] =  new Circle(1.0);
    shape[3] =  new Rectange(3.0, 1.5);
    shape[4] =  new Circle(0.0);
    for( int i = 0; i < 5; i++)
        cout<<"Area of shape "<<i <<" is "<<shape[i]->GetArea()<<endl;
}

//以下请写出 Shape、Rectangle、Circle 三个类的实现。
```