一、单选题，选择一个最优答案 (1x10 = 10 分)

1．下列关于成员函数特征的描述中，(　　) 是错误的。

A. 成员函数一定是内联函数　　B. 成员函数可以重载

C. 成员函数可以设置参数的默认值　　D. 成员函数可以是静态的

2．对于任何情况下，编译器都**不会添加的函数是**(　　)

A．无参数构造函数　　B．拷贝构造函数

C．赋值运算符函数　　D．输入输出运算符函数

3. (　　)**不是类的**成员函数。

A．构造函数　　B．析构函数

C．友元函数　　D．拷贝构造函数

4. 公有派生类的成员函数体内可以访问基类的 (　　)

A．公有成员和私有成员　　B．受保护成员和私有成员

C．公有成员和受保护成员　　D．公有成员、受保护成员和私有成员

5. 假定 Tom 为 CStudent 类的一个对象，则执行” CStudent Jack =Tom；”语句时将自动调用该类的(　　)

A．缺省构造函数　　B．转换构造函数

C．拷贝构造函数　　D．赋值运算符函数

6. 下列对派生类的描述中，(　　) 是错误的。

A. 一个派生类可以作为另一个派生类的基类

B. 派生类至少有一个基类

C. 派生类的成员除了它自己的成员外, 还包含了它的基类的成员

D. 派生类中继承的基类成员的访问权限到派生类中保持不变

7.　下列说法中，正确的是(　　)

A．多重继承的派生类可以有两个以上基类，可以增大代码的重用性，没什么弊端

B．没有虚基类时，基类构造函数的调用顺序取决于**基类被继承时的顺序**

C．基类构造函数的调用顺序由它们在**初始化列表中**的顺序决定

D．多重继承所产生二义性问题，无法消除，所以多个基类不能有相同的成员。

8．下列虚基类的声明中正确的是(　　)。

A. class virtual B: public　A　　B. virtual class B: public　A

C. class B: public　A　virtual　　D. class B: virtual　public　A

9．　下列关于函数重载与函数模板的说法，哪一个是不正确的(　　)

A．当函数的函数体不同但功能相同时，可以设计为重载函数

B．当函数体相同，仅仅操作的数据类型不同时，可设计为函数模板

C．重载函数和函数模板都是抽象的，都需实例化

D．函数模板是有待于实例化为模板函数的，重载函数可以直接使用

10．　动态绑定与静态绑定的说法不正确的是(　)

A．静态绑定在程序的编译阶段就已确定，动态绑定在程序运行期间确定

B．动态绑定是通过继承、虚函数及指针来实现的，它灵活性大

C．静态类型用于编译程序检查类型的合法性，动态类型用于是运行时动态绑定

D．动态绑定效率高，灵活性大，要尽可能的使用

二、填空题 (1x10 = 10 分)

1. 在 C++中，使用_____分配的内存, 使用 delete 进行释放.

2. 在类模板中, template 关键字后的尖括号内的类型参数都要冠以保留字_____或_____。

3. 假定 COneClass 为一个类，则执行“COneClass array[10];”语句时，系统自动调用该类的构造函数的次数为_____。

4. 调用对象的成员函数时，编译器会传递一个隐含的 this 指针, 该指针指向_____。

5．class CComplex 用成员函数重载**负号**运算符-的函数原型是_____

6.　类的_____、_____和没有无参数构造函数的子对象必须在初始化列表中进行初始化。

7．在类的对象被释放时, _____函数会被自动调用。

8．为了实现动态多态，派生类需要重新定义基类的_____

9．如果一个类中含有纯虚函数，则该类称为_____，不能直接创建实例，只能做基类。

10．C++中，缺省是_____，采用虚继承可以实现共享继承.

三、改错题，指出错误，并改正之(5x3 =15)

1．　改正类声明的错误

```
class CPoint{
        int mX(0);
        int    mY;
  public：
        CPoint (int x = 0,int y) {mX =x; mY = y;}
  }
```

2. #include <iostream>

using namespace std;

```
class CAnimal {
public:
	void set_age(int x)  {
		age= x;
	}
	int get_age()   {
		return age;
	}
private:
	int age;
};
class CFish: public CAnimal {
public:
	void set_weight(int w)   {
		wight = w;
		}
protected:
	int wight;
private:
	void set_age(int x) { age =x;}
};

int main(){
	 CFish obj;
	 obj.set_age(15);
	 obj.set_weight (1000);
	 cout << obj.get_age() << "\n";
	 return 0;
}
```

3. 类的组合

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

class CStudent{
public:
	int m_age;
	string m_name;


};
```

```
class CClass{
 public:
     CStudent m_Member[40];
     void CClass(){};
}
```

四、程序填空题(5x3=15)

1. 全局对象、局部对象的构造析构

```
#include <string>
#include <iostream>
using std::string;
using std::cout;
using std::endl;

class A
{
public:
    A(string s)
    {
        str.assign(s);
        cout << str << ":A 构造" << endl;
    }
    ~A()
    {
        cout << str << ":A 析构" << endl;
    }
private:
    string str;
};
```

_____

```
int main()
{
```

_____

```
    return 0;
}
```

在横线处填上合适的语句,不改变其他语句，使程序运行时输出：

全局：A 构造

main：A 构造

main：A 析构

全局：A 析构

2．整型类

```
// "integer.h"
class Integer {
public:
    Integer(){  m_value = 0; }

    Integer(int value) { m_value = value;  }

    _____( int value) {
        int tmpValue = m_value + value;
        return Integer(tmpValue);
    }
_____(int value) {    m_value = value;    }

private:
    int m_value;
};


//main.cpp
#inlude <iostream>
#include "integer.h"
using namespace std;

int main()
{
Integer integer = Integer(10);
Integer tmpInteger = 100; //重载=运算符
tmpInteger = integer + 1; //重载+运算符
return 0;
}
```

在横线处填上代码，完成此整型类。

3．动态多态的应用

```
#include <iostream.h>
class CPserson {
public:
_____ youtask (     ) =0;
};
class CTeacher :public CPerson
{
public:
```

```
void youtask(){_____}
};

class CStudent: :public CPerson
{
public:
void youtask (){_____}
};
int main(){
CPerson * pPerson = NULL;
CTeacher TeacherWang；
CStudent John；
pPerson = & TeacherWang;
pPerson-> youtask ();

pPerson = & John;
pPerson-> youtask ();
}
```

在横线处填上适当代码，使程序运行结果如下(不包括引号)：

“教学科研”

“上课学习”

五、理解题，写出下面程序的输出结果，并做简要解释(5x4=20 分)

1. 类

```
class Cents
{
private:
    int m_cents;
public:
    Cents(int cents = 0)
    {
        m_cents = cents;
    }

    // Overloaded int cast
    operator int() { return m_cents; }

    int getCents() { return m_cents; }
    void setCents(int cents) { m_cents = cents; }
};

class Dollars
{
private:
```

```
    int m_dollars;
public:
    Dollars(int dollars=0)
    {
        m_dollars = dollars;
    }

     // Allow us to convert Dollars into Cents
     operator Cents() { return Cents(m_dollars * 100); }
};

void printCents(Cents cents)
{
    std::cout << cents; // cents will be implicitly cast to an int here
}

int main()
{
    Dollars dollars(9);
    printCents(dollars); // dollars will be implicitly cast to a Cents here

    return 0;
}
```

输出结果：

2 继承多态

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

class CParent {
   public:
     void print_classname( ) { cout << " class CParent \n ";    }
     CParent (){cout<<"CParent ()\n";}
     virtual ~ CParent (){cout<<"~ CParent ()\n";}
};
class CSon: public CParent {
  public:
        CSon (){ cout<< "CSon ()"<<end;}
        void print_classname( ) { cout << "class CSon \n";    }
        virtual ~B(){cout<<"~ CSon ( )\n";}
};
void main( )
{
```

```
    CParent *pParent =new    CSon ( );
    pParent ->print_calssname( );
  delete pParent;
}
```
输出结果：

3．特殊成员

```
class MyClass {
  public:
      MyClass(){ count++; }
      ~MyClass(){ count--; }
      static int GetCount(){ return count; }
  private:
      static int count;
  };
  int MyClass::count = 0;

  int main(void) {
      MyClass obj1;
      cout << MyClass::GetCount() << endl;

      MyClass obj2(obj1);
        cout << MyClass::GetCount() << endl;

      MyClass *p = new MyClass();
      cout << MyClass::GetCount() << endl;

      delete p;
        cout << MyClass::GetCount() << endl;
  return 0;
}
```
上面程序的输出结果为:

4 异常

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

struct MyException : public exception {
   const char * what () const {
      return "C++ Exception";
```

```
    }
};

int main() {
    try {
        throw MyException();
    }catch(MyException& e) {
        std::cout << "MyException caught" << std::endl;
        std::cout << e.what() << std::endl;
    } catch(std::exception& e) {
        //Other errors
    }
}
```

输出结果：

六、简答题：（5*4=20 分）

1．何为引用型变量？ 使用引用有何好处？引用经常被用在什么场合？

2． 继承和组合有何共同点？举例说明。

3． 面向对象有哪三大特征？各自的含义和相互关系是什么？

4． 说说函数模板和类模板的异同。

七、程序设计题(10 分)

1. 写一个 CShape 类和它的两个派生类圆 CCircle 和矩形 CRectangle，使得下面程序可以分别输出圆和矩形的面积，并且可以输出所有属于 CShape 类的实例的个数。提示：注意静态成

员的初始化、virtual 的使用以及部分析构现象

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

int main(){
CShape * pShape;
CCircle * pC = new Circle(0.0, 0.0,5.0);// （0.0, 0.0）为圆心坐标,5.0 为半径
CRectangle  * pRect = new CRectangle(1.0,2.0,5.0,7.0); //(1.0,2.0)为左下角坐标,
（5.0,7.0）为右上角坐标

pShape = pC;
pShape->getArea();

pShape = &pRect;
pShape->getArea();

cout<<pShape->getInstanceCount()<<endl;//返回 2

delete pC;
delete pRect;

cout<<pShape->getInstanceCount()<<endl;//返回 0
return 0;
}
```