《C++面向对象程序设计》模拟试题二

本试卷卷面分数为 80 分。

一、单项选择题（本大题共 10 小题，每小题 2 分，共 20 分）

1．说明内联函数的关键字是（　　）。
A. inline　　B. virtual　　C. define　　D. static

2．假定 CAb 为一个类，则执行 CAb oX；语句时将自动调用该类的（ ）
A. 有参构造函数　　B. 无参构造函数
C. 拷贝构造函数　　D. 赋值重载函数

3．cin 是某个类的标准对象的引用，该类是（　　）。
A. ostream　　B. istream　　C. stdout　　D. stdin

4．下面的哪个保留字不能作为函数的返回类型？（　　）
A. void　　B. int　　C. new　　D. long

5．不能参与重载的运算符是（　　）。
A. 类　　B. 函数　　C. 函数模板　　D. 运算符

6．由于数据隐藏的需要，静态数据成员通常被说明为（　　）。
A. 私有的　　B. 公有的　　C. 保护的　　D. 不可访问的

7．编译时多态性使用什么获得？（　　）
A. 重载函数　　B. 继承　　C. 虚函数　　D. B 和 C

8．拷贝构造函数的参数通常是（　　）。
A. 无特殊要求　　B. 指向对象的指针
C. 自己类对象的常引用　　D. 对象

9．C++有几种联编？（　　）
A. 1 种　　B. 2 种　　C. 3 种　　D. 4 种

10．基类和派生类可以分别称为（　　）。
A. “大类”和“小类”　　B. “父类”和“子类”
C. “小类”和“大类”　　D. “子类”和“父类”

二、判断正误题（本大题共 5 小题，每小题 2 分，共 10 分）判断正误，在题后的括号内，正确的划上“√”错误的划上“×”。

1．不可以定义抽象类的对象。（　　）
2．内联函数的定义必须出现在第一次调用内联函数之前。（　　）
3．模板函数与函数模板的意义完全相同。（　　）
4．只有常成员函数才可以操作常对象。（　　）
5．引用可以不初始化。（　　）

三、填空题（本大题共 5 小题，每小题 2 分，共 10 分）

1．设函数 max 是由函数模板实现的，并且 max(3.5, 5)和 max(3, 5)都是正确的函数调用，则此函数模板具有（　　）个类型参数。
2．在 C++中，函数重载与虚函数帮助实现了类的（　　）性。
3．由 static 修饰的数据成员为该类的所有对象（　　）。
4．重载函数在参数类型或参数个数上不同，但（　　）相同。

5．使用 new 建立的动态对象在不用时必须用（　　）释放所占用的空间。

四、程序分析题（本大题共 4 小题，每小题 5 分，共 20 分）给出下面各程序的输出结果。

1．阅读下面程序，写出输出结果。

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

class CPosition
{
public:
    CPosition(int iPositionX = 0, int iPositionY = 0):m_iPositionX(iPositionX)
    {
        m_iPositionY = iPositionY;
    }

    int GetPositionX() const
    {
        return m_iPositionX;
    }

    int GetPositionY() const
    {
        return m_iPositionY;
    }

    void SetPositionX(int iPositionX)
    {
        m_iPositionX = iPositionX;
    }

    void SetPositionY(int iPositionY)
    {
        m_iPositionY = iPositionY;
    }


private:
    int m_iPositionX;           // X 坐标
    int m_iPositionY;           // X 坐标
};


int main(void)
{
    CPosition oPostion1;
```

```
    const CPosition oPostion2(6, 8);

    cout << oPostion1.GetPositionX() << endl;
    oPostion1.SetPositionX(16);
    cout << oPostion1.GetPositionX() << endl;
    oPostion1.SetPositionY(18);
    cout << oPostion1.GetPositionY() << endl;

    cout << oPostion2.GetPositionX() << endl;
    cout << oPostion2.GetPositionY() << endl;

    return 0;
}
```

上面程序的输出结果为：

2．阅读下面程序，写出输出结果。

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

template <class Type>
class CTest
{
public:
    CTest(Type m_tArray[], int iSize):m_pArray(m_tArray)
    {
        m_iSize = iSize;
    }

    void Print() const
    {
        for (int i = 0; i < m_iSize; i++)
        {
            cout << m_pArray[i] << "    ";
        }
    }

private:
    Type *m_pArray;
    int m_iSize;
};
```

```
int main(void)
{
	int a[] = {1, 0, 8};
	double b[] = {1.6, 1.8};

	CTest<int> oTest1(a, 3);
	oTest1.Print();

	CTest<double> oTest2(b, sizeof(b) / sizeof(double));
	oTest2.Print();

	cout << endl;

	return 0;
}
```

上面程序的输出结果为：

3．阅读下面程序，写出输出结果。

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

class CGoods
{
public:
	CGoods(int iWeight)
	{
		m_iWeight = iWeight;
		m_iTotalWeight = m_iTotalWeight + iWeight;
	}

	CGoods(const CGoods &oGood)
	{
		m_iWeight = oGood.m_iWeight;
		m_iTotalWeight = m_iTotalWeight + m_iWeight;
	}

	~CGoods()
	{
		m_iTotalWeight = m_iTotalWeight - m_iWeight;
```

```
    }

    void Print() const;

    static int GetTotalWeight()
    {
        return m_iTotalWeight;
    }

private:
    int m_iWeight;
    static int m_iTotalWeight;
};

int CGoods::m_iTotalWeight = 8;                // 初始化静态数据成员

void CGoods::Print() const
{
    cout << this->m_iWeight << "    " << this->m_iTotalWeight << "    ";
}

int main(void)
{
    CGoods oGood1(6);
    oGood1.Print();

    CGoods oGood2(oGood1);
    oGood2.Print();

    cout << CGoods::GetTotalWeight();
    cout << endl;

    return 0;
}
```

上面程序的输出结果为：

4．阅读下面程序，写出输出结果。

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

template <class Type>
```

```
class CTest
{
public:
	CTest(Type tA = 0, Type tB = 0, Type tC = 0):m_tC(tC)
	{
		m_tA = tA;
		m_tB = tB;
	}

	void Print()
	{
		cout << m_tA << endl;
		cout << m_tB << endl;
	}

	void Print() const
	{
		cout << m_tC << endl;
	}


private:
	Type m_tA, m_tB;
	const Type m_tC;
};


int main(void)
{
	CTest<float> oTest1;
	oTest1.Print();

	CTest<int> oTest2(1, 9, 6);
	oTest2.Print();

	const CTest<double> oTest3(0, 6, 1.8);
	oTest3.Print();

	cout << endl;

	return 0;
}
```

上面程序的输出结果为：

五、编程题（本大题共 2 个小题，共 20 分）

1．编写一个函数模板，用于求数组中各元素之和，并编写测试程序进行测试。（10 分）
函数模板声明如下：
template <class Type>
Type Sum(Type tArray[], int iSize)

2．定义一个复数类 Complex，二个数据成员为 double 型 r, i 为 private 属性。定义代二个参数的构造函数和一个 Show( ) 函数用以输出 r, i 的值，另外作为成员函数重载的运算符”+”的功能是将此类二个对象的数据成员 r 和 i 对应相加。这些成员函数的属性均为 public. 请用 C++编写此程序，并编写测试程序进行测试。（10 分）

参考答案

一、单项选择题（本大题共 10 小题，每小题 2 分，共 20 分）

1．A	2．B	3．B	4．C	5．A
6．A	7．A	8．C	9．B	10．B

二、判断正误题（本大题共 5 小题，每小题 2 分，共 10 分）判断正误，在题后的括号内，正确的划上“√”错误的划上“×”。

1．参考答案：√
2．参考答案：√
3．参考答案：×
4．参考答案：√
5．参考答案：×

三、填空题（本大题共 5 小题，每小题 2 分，共 10 分）不写解答过程，将正确的答案写在每小题的空格内。错填或不填均无分。

1．参考答案：2
2．参考答案：多态
3．参考答案：共享
4．参考答案：函数名
5．参考答案：delete

四、程序分析题（本大题共 4 小题，每小题 5 分，共 20 分）给出下面各程序的输出结果。

1．参考答案：
0
16
18
6
8

2．参考答案：1　0　8　1.6　1.8

3．参考答案：6　14　6　20　20

4．参考答案：
0
0
1
9
1.8

五、编程题（本大题共 2 个小题，共 20 分）

1．参考程序：

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

template <class Type>
Type Sum(Type tArray[], int iSize)
{
	Type tSum = 0;
	for (int i = 0; i < iSize; i++)
	{
		tSum = tSum + tArray[i];
	}
	return tSum;
}


int main(void)
{
	int a[] = {1, 2, 3};
	double b[] = {1.5, 2.8, 8.9, 8};

	cout << Sum(a, 3) << endl;
	cout << Sum(b, 4) << endl;

	return 0;
}
```

2．参考程序：

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

class Complex
{
private:
	double r, i;

public:
	Complex(double a, double b): r(a), i(b) {}
	void Show() { cout << r << "    " << i << endl; }
	Complex operator +(Complex    obj)
	{ return    Complex(r + obj.r, i + obj.i); }
};

int main()
{
```

```
    Complex c1(3.5, 4.5), c2(2.5, 5.5), c3(0.0, 0.0);
    c3 = c1 + c2;
    c3.Show();
    return 0;
}
```