《面向对象程序设计》期中考试

一、单选题（每题3分，共30分）

01、 以下类定义不正确的是 [**A**]

A) class A { B) class B {

Public: public:

void Func(int x){…}; void Func( int x){…};

int Func(int x){…}; void Func( double x){…};

}; };

C) class C { D) class D {

public: public:

void Func(int x,int y){…}; int Func( int x){…};

void Func( int x){…}; char\* Func( char x){…};

}; };

02、假定A是一个类的名字，下面哪个语句不会引发类A构造函数的调用 [**D**]

A) A \* p = new A; B) A a;

C) A a[10]; D) A \* a[10];

03、若类B是从类A派生而来，而语句 p.Do(); 在被多次执行时，有的时候调用的是类A的Do成员函数，有的时候调用的是类B的Do成员函数，那么 p 的类型可以是 [**A**]

A) A & B) A \*

C) B D) A

04、关于左值引用，下列用法或叙述不正确的是 [**A**]

A) 引用可以先定义，再赋值；

B) 引用可以作为函数参数；

C) 函数可以返回引用；

D) 引用是另一个有名字变量的别名。

05、下列不能作为类的成员的是 [**B**]

A) 自身类对象的指针 B) 自身类对象

C) 自身类对象的引用 D) 另一个类的对象

06、以下说法正确的是 [**A**]

A) 同一个运算符可以被重载多次；

B) C++已有的运算符都可以重载;

C) 自增运算符重载时，重载函数不允许有输入参数；

D) 运算符函数的返回类型不能声明为基本数据类型。

07、以下说法正确的是 [**D**]

A) 派生类会继承基类的数据成员、函数成员和重载赋值运算符；

B) 基类中没有指定访问说明符时，编译器将默认该说明符是public；

C) 派生类不会继承基类的构造函数，因此不能用于创建派生类对象的基类部分；

D) 派生类的构造函数可以调用特定的基类构造函数，间接访问基类的私有成员。

08、以下说法不正确的是 [**B**]

A) 静态成员函数可以调用静态成员函数；

B) 静态成员函数可以调用非静态成员函数；

C) 非静态成员函数可以调用静态成员函数；

D) 非静态成员函数可以调用非静态成员函数。

09、以下说法正确的是 [**B**]

A）在const成员函数中不可以使用this指针；

B）在static成员函数中不可以使用this 指针；

C）抽象类的成员函数都是纯虚函数；

D）抽象类的派生类必须实现抽象类中的纯虚函数。

10、下面这段代码的运行结果为 [**A**]

#include<iostream>

using namespace std;

class A {

public:

void f() {cout<<"A::f"<<endl;}

};

class B:public A {

public:

void f() {cout<<"B::f"<<endl;}

};

int main()

{

A \*p = new B();

p->f();

delete p;

return 0;

}

A）A::f B）B::f

C）A::f\nB::f（\n表示换行） D）不产生输出

二、不定项选择题（每题4分，共20分）

11、有关多态性的说法正确的是 [**ACD**]

A) C++语言的多态性分为编译时的多态性和运行时的多态性；

B) 运行时的多态性可通过模板和虚函数实现；

C) 编译时的多态性可通过函数重载实现；

D) 实现运行时多态性的机制称为动态多态性。

12、假定A为一个类，a为该类公有的数据成员，x为指向该类对象的一个指针，则访问x中数据成员a的格式为 [**BC**]

A) x.a B) x->a

C) (\*x).a D) (\*x) -> a

13、下列关于模板的用法正确的是 [**CD**]

A) template <typename T>

T add(T x, T y) {return x+y;}

int fun(int a, double b) { T c = add(a,b); }

B) template <typename T1, typename T2>

class Test{

public:

T1 funA(T1 a);

T2 funB(T2 b);

};

template <typename T1>

T1 Test<T1>::funA(T1 a){return a;}

C) class A{

public:

template<typename T>

void fun(T item){cout<<item<<endl;}

};

D) template <typename T1, typename T2>

class Test {

public:

T1 fun (T1 a, T2 b);

};

template <typename T1, typename T2>

T1 Test<T1, T2>::fun (T1 a, T2 b){return a+b;}

14、以下哪些成员函数在用户没有为类实现时，编译器会自动为类生成他们的缺省实现 [**ABCD**]

A) 默认构造函数 B) 拷贝构造函数

C) 移动构造函数 D) 析构函数

15、下列关于类型转换说法正确的是 [**AC**]

A) dynamic\_cast<Dst\_Type>(Src\_var)中，若目标类与原类之间没有继承关系，则转换失败，返回空指针；

B) static\_cast<Dst\_Type>(Src\_var)中，派生类对象不能转换成基类对象，但派生类指针可以转换成基类指针；

C) 可以通过定义源类对象做参数的构造函数来实现自动类型转换，并且此构造函数只能接受一个非默认值的参数；

D) 禁止自动类型转换时，可以使用explicit关键字修饰类的任意成员函数。被explicit修饰的类不能发生相应的隐式和显式类型转换。

三、填空题（每题2分，共10分）

16、**抽象**类不许定义对象。

17、在继承中，默认的缺省继承方式为**private** (public/private)继承。

18、使用**decltype**可以对变量或表达式结果的类型进行推导。

19、只有类的**静态**成员可以通过类来直接访问。

20、指针变量所指内存空间可以通过**New**运算符在程序运行时申请。

四、阅读代码题（每题8分，共40分）

21、请补全下列代码空白处（共4处）

#include <iostream>

using namespace std;

class A {

public:

int data;

**A(int i)** { cout << "A::A("<<i<<")\n" ;}

void f(int i) { cout << "A::f(" << i << ")\n"; }

void f(double d) { cout << "A::f(" << d << ")\n"; }

};

class B: **public** A {

public:

int data{2017};

using A::A;

using A::f;

void f(int i) { cout << "B::f(" << i << ")\n"; }

void print() { cout << "data = " << data << endl; }

};

int main() {

B b(6);

b.print();

**b.f(20);**

b.f(17.315);

return 0;

}

运行结果：

A::A(6)

**data = 2017**

B::f(20)

A::f(17.315)

22、给出以下面代码运行结果：

#include <iostream>

using namespace std;

class Num {

int a;

public:

Num(int b=1) {a=b;}

void print() {cout <<++a<<"";}

void print()const {cout<<a<<endl;}

};

int main() {

Num x;

const Num y(3);

x.print();

y.print();

return 0;

}

**答案：23**

23、test.cpp内容如下，运行命令：

g++ -o test test.cpp --std=c++11 -fno-elide-constructors

请写出运行结果。

//test.cpp

#include <iostream>

using namespace std;

class Test {

public:

Test() {

cout << "Test()" << endl;

}

Test(const Test& src) {

cout << "Test(const Test&)" << endl;

}

~Test() {

cout << "~Test()" << endl;

}

};

Test func(Test obj) {

cout << "func()..." << endl;

return Test();

}

int main() {

cout << "main()..." << endl;

Test t;

t = func(t);

return 0;

}

**答案：**

**main()...  
Test()  
Test(const Test&)  
func()...  
Test()  
Test(const Test&)  
~Test()  
~Test()  
~Test()  
~Test()**

24、给定类如下，请写出代码的准确输出：

class SmallInt{

public:

SmallInt (int i=0): val(i){

cout<<"SmallInt\_Init"<<endl;

}

operator int() const {

cout<<"Int\_Transform"<<endl;

return val;

}

void print() {

cout << val << endl;

}

private:

size\_t val;

};

int main()

{

SmallInt si;

si = 4.10;

si = si + 3;

si.print();

return 0;

}

**答案：**

**SmallInt\_Init**

**SmallInt\_Init**

**Int\_Transform**

**SmallInt\_Init**

**7**

25、下列程序能否编译通过？如果修改编译错误代码，输出是什么？

#include <iostream>

#include <string>

using namespace std;

template <typename T>class Sum {

public: T a, b;

Sum(T op1, T op2) : a(op1), b(op2) {}

T calc() {return a + b;}

};

template <typename T> T calc(T a, T b) {

return a + b;

}

int main() {

cout << Sum(0.4,0.5).calc() << endl;

cout << Sum<string>("ggwp!","ezgame!!").calc() << endl;

int res = calc(0.1, 0.2);

cout << res << endl;

cout << calc<float>(1.1, 2.0) << endl;

return 0;

}

**答案：修改为：**

**int main() {**

**cout << Sum<float>(0.4,0.5).calc() << endl;**

**//输出0.9**

**cout << Sum<string>(“ggwp!”,“ezgame!!”).calc() << endl;**

**//输出ggwp!ezgame!!**

**int res = calc(0.1, 0.2);**

**cout << res << endl;**

**//输出0，有一个类型转换！**

**cout << calc<float>(1.1, 2.0) << endl;**

**//输出3.1**

**return 0;**

**}**