《面向对象程序设计》期中考试

**一、单选题（每题3分，共30分）**

01、 以下类定义不能通过编译的是 [**C**]

A) class A {

public:

void func(int a,int b) {};

};

B) class B {

public:

void func(int a,int b=1){}

};

C) class C {

public:

void func(int a=3,int b){}

};

D) class D {

public:

void func(int a=3,int b=1){}

};

02、假定A是一个类的名字，下面四个语句总共会引发类A构造函数的调用多少次 [**B**]

① A \*p = new A;

② A p2[10];

③ A p3;

④ A \*p4[10];

A) 11 B) 12

C) 21 D) 22

03、请问下面的代码的输出是[**C**]

#include<iostream>

using namespace std;

class A {

public:

void work() { cout << "A"; }

};

class B : public A {

public:

void work() { cout << "B"; }

};

int main(){

B \*c = new B();

A \*d = new B();

c->work();

d->work();

}

A) AA B) AB

C) BA D) BB

04、下列说法正确的是 [**C**]

A) 非常量左值引用与常量左值引用既可以绑定左值，也可以绑定右值

B) int y=2,z=3;int &&w = y \* z;不能够正常运行

C) int y=3,z=3;const int &w = y \* z;能够正常运行

D) 在函数中的临时左值引用可以先定义，再赋值

05、以下代码能够通过编译的是 [**A**]

A) class A {

public:

A\* a;

};

int main() {A a; return 0;}

B) class B {

public:

B& b;

};

int main() {B b; return 0;}

C) class C {

public:

C c;

};

int main() {C c; return 0;}

D) class E {};

class D {

public:

E& d;

};

int main() {D d; return 0;}

06、以下说法不正确的是 [**B**]

A) sum为自定义类的一个变量，可以通过 sum.operator()(5, 6);去调用()运算符重载函数

B) 通过重载[]运算符: int operator[] (const char\* name);使得我们可以像map一样使用 beijing["mon"] = -3;

C) ClassName& operator++();为前缀自增运算符的重载声明

D) 运算符()必须作为成员函数重载

07、以下说法不正确的是 [**A**]

A) public继承中派生类的成员函数可以访问基类中的私有成员

B) private继承中派生类对象不允许向上转换

C) 显式调用基类构造函数只能在派生类构造函数的初始化成员列表中进行

D) 如果基类的某个构造函数被声明为私有，则不能在派生类中调用该私有构造函数

08、以下说法不正确的是 [**D**]

A) 类的静态成员（数据、函数）既可以通过对象来访问，也可以通过类名来访问

B) 非静态成员函数可以调用静态成员函数

C) 静态数据成员可以在.h文件中声明，在.cpp文件中初始化

D) 常量静态数据成员必须在类外进行初始化

09、以下说法不正确的是 [**D**]

A) 被声明为友元的函数或类，具有对出具友元声明的类的一切成员有访问权限。

B) A函数是B类的友元函数，B类是C类的友元类，A函数对C类没有特殊访问权限

C) 编译器认为某个函数不值得内联，就会忽略内联修饰符

D) 编译期间建立虚函数表VTABLE和虚函数指针VPTR

10、下面这段代码的运行结果为(\n表示换行) [**C**]

#include <iostream>

using namespace std;

class Base{

public:

virtual void foo(int){cout<<"Base::foo(int)"<<endl;}

};

class Derived : public Base {

public:

void foo(float) {cout<<"Derived::foo(float)"<<endl;}

};

int main() {

Derived d;

Base\* p = &d;

d.foo(3.0);

p->foo(3.0);

}

1. Base::foo(int)\nBase::foo(int)\n
2. Base::foo(int)\nDerived::foo(float)\n
3. Derived::foo(float)\nBase::foo(int)\n
4. Derived::foo(float)\nDerived::foo(float)\n

**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 [**AC**]

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) 实现编译时多态性的机制称为动态多态性

12、可以填在类B的print函数中的是[**AB**]

#include <iostream>

using namespace std;

class A {

public:

int a=1;

protected:

int b=2;

private:

int c=3;

};

class B: public A {

public:

A obj\_a;

void print() {

// print something

}

};

B obj\_b;

int main() {

obj\_b.print();

return 0;

}

A) cout << a << endl; B) cout << b << endl;

C) cout << c << endl; D) cout << obj\_a.b << endl;

13、下列关于模板的用法正确的是 [**ABC**]

A) template <typename T>

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

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

int main() {fun(1,2);}

B) template <typename T1, typename T2>

class Test{

public:

T1 funA(T1 a);

};

template <typename T1, typename T2>

T1 Test<T1, T2>::funA(T1 a){return a;}

int main() {Test<int,int> a; a.funA(3);}

C) class Test{

public:

template<typename T>

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

};

int main(){Test a; a.fun(3);}

D) **Test.h**

template<typename T>

class Test {

public:

void print(T x);

};

**Test.cpp**

#include “Test.h”

template<typename T>

void Test<T>::print(T x){cout << x << endl;}

**main.cpp**

#include “Test.h”

int main(){

Test<int> object;

object.print(4);

}

14、下列两个是什么构造函数（不需要按照顺序） [**CD**]

class Test {

public:

Test(Test src) {

cout << "Guess what’s this" << endl;

}

Test(Test&& src) {

cout<<"Guess again"<<endl;

}

};

A) 默认构造函数 B) 拷贝构造函数

C) 移动构造函数 D) 普通构造函数

15、下列关于STL的说法正确的是[**AD**]

A) vector在大小发生改变时，可能致使所有迭代器失效

B) 为了计算效率，list的访问主要依赖下标

C) 通过下标访问map<int,int>时，如果元素不存在，程序会抛出异常

D) 关联容器(如map、set)可以使用迭代器访问

**三、填空题（每题3分，共18分）**

16、抽象类是包含 **纯虚函数** 的类

17、基类中的public成员，在private继承的派生类中是 **private** (public/private/protected)成员

18、函数模板中结合decltype与 **auto** ，用于推导函数的返回值类型

19、类的静态数据成员 **能** (能/不能)被类的非静态成员函数修改

20、使用delete运算符释放指针变量ptr指向的内存空间后，ptr的值 **不变** (变/不变)

21、对模板进行特殊化处理时， **只有类模板**  (只有类模板/只有函数模板/类模板和函数模板)能够部分特化

**四、简答题（共4分）**

22、下面的程序是否会出现编译错误？如果会，说明含标号的语句中哪句会导致编译错误，以及出错的理由；如果不会，写出该程序的运行结果。

#include <iostream>

using namespace std;

void f(int &x) {

cout << "left " << x << endl;

}

void f(int &&x) {

cout << "right " << x << endl;

int &y = x; //(2)

const int &z = y + 2017; //(3)

f(z); //(4)

}

int main() {

f(2018); //(1)

return 0;

}

**会，标号为（4）的语句会导致编译错误，因为不存在f(const int&)的函数与其匹配**

**五、阅读代码题（共28分）**

**23. 给出以下面代码运行结果（8分）**

#include <iostream>

using namespace std;

class A {

int data = 2018;

public:

A() { cout << "A::A("<<data<<")\n" ;}

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

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

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

};

class B: public A {

int data = 2019;

public:

B() { cout << "B::B("<<data<<")\n" ;}

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

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

};

class C: public B {

public:

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

};

int main() {

B b;

A a = b;

B\* b2 = new C();

a.f(17.315);

b.f(17.315);

a.f(20);

b.f(20);

b2->f(20);

a.print();

return 0;

}

**答案：**

**A::A(2018)**

**B::B(2019)**

**A::A(2018)**

**B::B(2019)**

**A::f(17.315)**

**B::f(17)**

**A::f(20)**

**B::f(20)**

**C::f(20)**

**A data = 2018**

一二全对得**1**分，三四全对得**1**分，后面每个**1**分

**24、给出以下面代码运行结果：（6分）**

#include <iostream>

using namespace std;

class Test {

public:

int a,b;

Test() {a=1;b=2;}

Test(int c,int d=4) {a=c;b=d;}

void output() const {cout<<b;}

void output() {cout <<++a;}

};

int main() {

Test dz1;

const Test dz2(5);

dz1.output();

dz2.output();

cout << dz1.a << dz1.b << dz2.a << dz2.b;

return 0;

}

**答案：242254**

**25、test.cpp内容如下，运行命令： （7分）**

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

#include <iostream>

#include <utility>

using namespace std;

class Test {

public:

Test() {

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

}

Test(const Test& src) {

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

}

};

Test func(Test& obj1, Test obj2, Test&& obj3) {

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

obj1 = obj2;

obj2 = obj3;

return Test();

}

int main() {

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

Test t1; Test t2; Test t3;

t2 = func(t1, t2, move(t3));

return 0;

}

**答案：**

补全输出结果，并指明每一个输出是针对哪个具体量而进行的构造或拷贝构造

输出结果 针对量

main()...

**Test()** t1

**Test()** t2

**Test()**  t3

**Test(const Test&)** **obj2**

func()...

Test()

**Test(const Test&)**  **func(t1, t2, move(t3))返回的中间值\_**

**意思对即可给分**

**26、给定类如下，写出代码的准确输出（7分）**

#include <iostream>

using namespace std;

class MyString{

string s;

public:

MyString(int i):s(to\_string(i)){

cout<<"MyString init int "<<s<<endl;

}

MyString(string i):s(i){

cout<<"MyString init string "<<s<<endl;

}

operator int() const{

int i = std::stoi(s); // transform string to int

cout<<"transform to int "<<i<<endl;

return i;

}

};

int main() {

MyString a = 2;

MyString b = string("233");

MyString c = a + b;

return 0;

}

**答案：**

**MyString init int 2**

**MyString init string 233**

**transform to int 2**

**transform to int 233**

**MyString init int 235**

**或者**

**MyString init int 2**

**MyString init string 233**

**transform to int 233**

**transform to int 2**

**MyString init int 235**

**Note：a+b中a、b并没有严格的先后调用顺序**