C++期末考试模拟卷

考试时间：2 小时

一．选择题(每小题 2 分，共 30 分)

(1)关于 C++与 C 语言的关系的描述中错误的是(D)

A. C 语言是 C++的一个子集　　B. C++的编译器可以编译 C 语言代码

C. C++对 C 语言进行了一些改进　　D. C++和 C 语言都是面向对象的

(2)C++中，函数原型不能标识(C)

A. 函数的参数类型与数量　　B. 函数的默认参数

C. 函数的处理流程　　D. 函数的返回值类型

(3)使用派生类的主要原因是(A)

A. 提高代码的可重用性　　B. 提高程序的运行效率

C. 加强类的封装性　　D. 实现数据的隐藏

分析：派生类直接从基类处获得非私有成员，无需重新写一遍，因此提高代码重用性

(4)类模板在使用时实际是先将类模板实例化成一个具体的(A)

A. 类　　B. 对象　　C. 函数　　D. 模板类

(5)该函数适合使用哪项技术进行效率优化(C)

```
int func(int a, int b, int c) { return 2 * (a + b) / c; }
```

A. 默认参数　　B. 函数重载　　C. 函数内联　　D. 虚函数重写

分析：该函数语句较少，因此使用函数内联展开可以避免运行时的调用开销，提高运行时效率

(6)成员函数可声明为静态的，条件是它的实现中不访问(C)

A. 非 const 成员　　B. 非 virtual 成员　　C. 非 static 成员　　D. 非 public 成员

分析：非 static 成员依赖于对象存在，而 static 成员属于类，和对象无关，因此 static 成员函数不可访问非 static 成员

(7)程序在调用重载函数时，往往根据一些条件确定哪个重载函数被调用，这个过程的名称以及不能作为依据的一项分别是(A)

A. 静态联编；返回值　　B. 实例化；参数类型

C. 静态联编；名称　　D. 实例化；参数数量

分析：将函数调用与实际执行的函数匹配起来的行为称为联编。仅通过返回值不能构成函数重载。

(8)下面哪个函数和 int func(int a, int b)构成重载函数时一定不会产生二义性(D)

A. string func(int a, int b, int c = 10);　　B. int func(int a, int b = 2);

C. string func(int& a, int &b);　　D. int func(string a, string b);

分析：首先重载函数和返回值无关，仅要求名称和参数列表不同且不产生二义性。A 选项由于存在默认参数，当调用时仅传两个 int 数据时产生二义性；B 选项直接不满足重载要求；C 选项传两个 int 变量时产生二义性

(9)若 A、B 是两个类，A←B 且 A 和 B 各含有一个整型公有成员变量 n，则(B)

A. B 类中无法访问 A 类的 n

B. B 类中可以通过 A::n 访问 A 类的 n

C. A 类中可以通过 B::n 访问 B 类的 n

D. B 类中必须通过 B::n 访问 B 类的 n

分析：父子类含有同名成员时，默认指子类的该成员，需要通过父类限定符显式访问父类的该成员

(10)一个类可包含析构函数的个数是(B)

A. 0

B. 1

C. 无数个，但要求名称相同且参数列表不同

D. 无数个，且名称和参数列表相同，编译器根据情况决定调用哪个

(10)C++中进行文件操作时需要包含头文件(D)

A. <iostream>　　B. <stdlib>　　C. <file>　　D. <fstream>

(11)对于语句 std::cout << x << std::endl; 描述错误的是(A)

A. “<<”称作流提取运算符 B. “endl”的作用是回车换行

C. “x”是一个变量 D. “cout”是一个对象

(12)在 C++中，若 A←C 且 B←C，则下列说法正确的是(D)

A. C++不允许这种情况

B. C 对 A 采用的继承方式必须和对 B 采用的一样

C. 若同时还有 D←A 且 D←B，则 C++不允许这种情况，否则允许

D. 若同时还有 D←A 且 D←B，则可以采用虚继承来避免二义性

分析：C++允许多继承，可以分别指定继承方式。若多个父类还有共同父类称之为菱形继承，有可能导致父类成员多副本问题，可以通过虚继承技术解决。

(13)当需要打开 D 盘上的 xxk.dat 文件用于输入时，定义文件流对象的语句为(B)

A. fstream fin(“D:\\xxk.dat”, 1)

B. ifstream fin(“D:\\xxk.dat”, ios::in)

C. ofstream fin(“D:\\xxk.dat”)

D. ifstream fin(“D:\\xxk.dat”, ios::app)

(14)若要对类 AB 定义加号操作符重载成员函数，实现两个 AB 类对象的加法，并返回相加结果，则该运算符函数的声明语句应为(B)

A. AB operator+(AB& a, AB& b) B. AB operator+(AB& a)

C. operator+(AB a) D. AB& operator+( )

分析：重载+号时，左侧操作数是当前对象，右侧操作数是 operator+函数的参数

(15)给定如下两个类 BASE 和 DERIVED，则 DERIVED 的构造函数合理的是(A)

```
class BASE {
private: int na;
public: BASE(int n) { na = n; }
};
class DERIVED : protected BASE {
private: int nb;
public: // 略...
};
```

A. DERIVED(int n1, int n2) : BASE(n1), nb(n2) { ... }

B. DERIVED(int n1, int n2) : n1(na), n2(nb) { ... }

C. DERIVED(int n1, int n2) : nb(n2) { BASE(n1); }

D. DERIVED(int n1, int n2) : BASE(n1) { nb(n2); }

分析：基类 BASE 没有默认构造函数，必须显式调用有参构造函数来构造，这个调用必须发生在派生类 DERIVED 的构造函数调用之后，执行之前，即参数列表中。

二．填空题(每空 1 分，共 10 分)

(1)定义类的动态对象数组需要使用 new[] 运算符，此操作只能自动调用该类的 无参(默认) 构造函数对其中的对象进行构造。

(2)一个类的拷贝构造函数应接收的参数为 (该类)引用 。

(3)若需要指明函数 int func(string s); 为类 AB 的友元函数，则应在类 AB 的定义中加入一条语句： friend int func(string s); 。

(4)在 C++中，将不同功能库的类与函数分模块进行实现并提供 头文件 供其他模块包含使用；为了避免来自不同库的同名符号产生二义性，应使用 名字空间 加以区分。

(5)异常处理目的是应对程序在 运行 (填编译或运行)时有可能遇到的意外错误情况，避免程序直接中断，给程序一个机会在 catch 语句块中对意外情况进行处理。

(6)一个类由于层级太宽泛而无法确定其某些行为的具体执行方式，需要留待更精确的子类来描述，则应将这些行为声明为 纯虚函数 ，这样的类称为 抽象类 。

三．判断题(每题 1 分，共 10 分)

(1)在基类中被声明为虚函数的成员函数必须在每个派生类中继续声明为虚函数，才能具有多态的特征(×)

(2)函数重载技术的加入使得 C++和 C 语言有了本质区别(×)

(3)引用就是某个变量的别名，对引用的操作，实质上就是对被引用的变量的操作(√)

(4)通过抽象分析得到的类的属性和行为可以用 UML 类图进行描述(√)

(5)友元函数是类的成员函数，所以可以存取或修改该类中的私有成员(×)

(6)就算被 inline 关键字声明的函数也不一定会被内联(√)

(7)同样存储整型数据 65535，通过文本文件存储比通过二进制文件存储更节省空间(×)

(8)块 catch(...){ } 应置于多个 catch 块的最前面(×)

(9)调用函数 template <typename T> T func(T a); 时可以直接写 func(n); 让编译器自行推断 T 所指代的类型(√)

(10)为了控制读写流程，类应为私有数据成员提供读写函数，对于数据成员 xxx，其读写函数的名称必须叫做 get_xxx 和 set_xxx (×)

四．简答题(共 15 分)

(1)请简述 C++源代码如何在计算机上运行(3 分)

答：C++源代码本质是文本文件(1 分)，由编译器将其转换为可执行的机器代码后在计算机上运行(提到编译器和机器代码 1 分)。编译过程一共有四步：预处理、编译、汇编和链接(提到编译过程 1 分)

(2)请简述 overload 和 override 的异同(4 分)

答：overload 指函数重载，override 指虚函数重写(1 分)。其相同点是通过某种机制实现函数调用和被调函数的自动匹配以实现代码复用(1 分)。其不同点是 overload 是静态联编，在编译期根据函数参数类型决定调用哪个函数(1 分)，而 override 是动态联编，在运行时根据指针实际指向的对象决定调用哪个重写的虚函数(1 分)。

(3)若程序员没有定义拷贝构造函数，则能否通过另一个同类对象构造新对象？这可能会产生什么问题？如何解决？(4 分)

答：若无显式定义，则编译器会自动为类生成一个拷贝构造函数(1 分)，但其只进行浅拷贝(1 分)，即简单复制成员数据。若类中含有有指针或文件等外部资源，则有可能导致对外部资源的访问冲突(1 分)，如需避免这种情况，需要显式定义一个拷贝构造函数，按实际需求进行深拷贝(1 分)。

(4)请简述为何 C++除兼容 C 语言 malloc/free 之外仍要引入 new/delete (4 分)

答：new/delete 是运算符(1 分)而 malloc/free 是函数，它们皆用于堆内存的动态分配与回收(1 分)。由于 C++新增了面向对象编程，而创建和销毁对象需要进行构造和析构，但 malloc/free 只进行内存申请与释放(1 分)，因此 C++中新增 new/delete 运算符用于动态对象的创建与销毁，后者除了管理内存空间的申请与释放，还会调用动态对象的构造函数与析构函数(1 分)。

五．代码阅读题(每题 3 分，共 12 分)

(1)下列代码段的输出结果是：3b

```
template <typename KK> KK func(KK a, KK b, KK c) {
        return a > b ? (a > c ? a : c) : (b > c ? b : c);
}
cout << func(1, 2, 3) << func('a', 'b', 'C');
```

分析：func(1, 2, 3)将类型参数 KK 推断为 int，执行三元表达式求得最大值 3, 而 func('a', 'b', 'C')将 KK 推断为 char，执行三元表达式求得 ASCII 值的最大者 ‘b’

(2)下列代码段的输出结果是：5

```
int m = 1, n = 3;
n = ++m * n--;
cout << n;
```

分析：m 进行前自增，用自增之后的值 2 进行表达式运算，n 进行后自减，用原值 3 进行表达式计算，得到 2 * 3 = 6，将 6 赋值给 n 完成表达式，但此时 n 还没完成自减，其自减后值为 5

(3)下列代码段的输出结果是：abcabcabc

```
class A {
public:
        string s;
        A(string s) : s(s) { }
        A operator* (int n) {
                string t = “” ;
                for (int i = 0; i < n; i++) { t += s; }
                return A(t);
        }
};
A a1("abc");
A a2 = a1 * 3;
cout << a2.s;
```

分析：类 A 重载了 * 号，其参数(即 * 的右操作数)是一个 int 数据 n，其效果是将成员字符串 s 横向复制 n 次得到 t 并用 t 构造新的 A 类对象作为结果返回

(4)若存在下列函数定义，则调用 fun2(10); 的输出结果是： 10 continued in fun2

```
void fun1(int a) {
        try {   if(a % 3 != 0)  throw(a);   }
        catch (string e) { cout << e << endl; }
        cout << "continued in fun1";
}
void fun2(int a) {
        try { fun1(a); }
        catch (int e) { cout << e << endl; }
        cout << "continued in fun2";
}
```

分析：函数内抛出异常时，如果该函数内部无 catch 块可处理该异常，则中止函数运行并在其调用行继续抛此异常，直到有一个 catch 块可处理则进入，处理完直接从此 try-catch 块后继续程序逻辑。

六．程序补全题(共 8 分)

(1)(2 分)请合理地补全类 A 的构造函数以及数据成员 n 的 setter 函数

```
class A {
private:
	int n;
public:
	A(int n) {
		this->n = n ;
	}
	int get_n() const {
		return n;
	}
	void set_n(const int n) {
		this->n = n ;
	}
};
```

(2)(3 分)请补全下列代码，避免可能的资源泄露

```
class A {
public:
        A() { }
        virtual ~A() { }
};
class B : protected A {
        int* ptr;
        ofstream fout;
public:
        B() {
                ptr = new int[100];
                fout.open("myFile.txt", ios::out);
        }
        ~B() {
                delete ptr;
                fout.close()  ;
        }
        // B 的其他成员函数 ...
};
A* p = (A*) new B();
// 使用 p ...
delete p  ;
```

分析：子类 B 含有外部资源句柄(内存指针和文件流对象)，因此需要在析构函数内释放这些资源。若父类 A 的析构函数不声明为虚函数，则通过 A 指针 p 指向 B 的动态对象后要销毁这个对象时，delete p 将不会调用 B 类对象的析构函数，造成资源泄露。

(3)(3 分)请补全下列代码，使函数 func 可以让各种动物都进行自我介绍

```
class Animal {
public: virtual void say() = 0;
};
class Cat : public Animal{
public: void say() { cout << "I'm a cat" << endl; }
};
class Dog : public Animal {
public: virtual void say() { cout << "I'm a dog" << endl; }
};
class Corgi : public Dog {
public: void say() { cout << "I'm a corgi dog" << endl; }
};
void func( Animal* p ) {
        p->say();
}
int main() {
        Corgi corgi1, corgi2;
        Cat cat1, cat2, cat3;
        Dog dog1, dog2;
        func( &cat1 ); // 让 cat1 进行自我介绍
        func( &corgi2 ); // 让 corgi2 进行自我介绍
        // 让其他动物进行自我介绍...
}
```

分析：基类 Animal 是一个抽象的概念，其 say 方法无法确定具体行为，因此声明为纯虚函数。猫 Cat 和狗 Dog 都是具体的动物类因此继承自 Animal，柯基类 Corgi 是更具体的狗因此继承自 Dog。要实现 func 对不同的动物都支持让其调用 say 方法，即是将不同层级的子类对象统一视作父类 Animal 看待，因此 func 的形参应当为指向 Animal 的指针，而实参应当为对各实际对象取地址得到的指针。

七．程序设计题(共 15 分)

(1)(7 分)请实现一个类 A，要求通过一个静态成员变量记录其实时对象数量，并编写合理的测试代码来动态创建和销毁 A 类对象以及检查对象计数。

```
#include <iostream>
using namespace std;
class A {
private:
	static int count;
public:
	A() { count++; }
	static int get_count() { return count; }
	~A() { count--; }
};
int A::count = 0;
int main() {
	A *p1 = new A();
	cout << A::get_count() << endl;
	A* p2 = new A();
	cout << A::get_count() << endl;
	delete p1;
	cout << A::get_count() << endl;
	delete p2;
	cout << A::get_count() << endl;
	return 0;
}
```

得分点：

- 类内部声明 static int count 并在类外部将其初始化为 0 (2 分)
- 构造/析构函数中将 count 进行自增/自减(2 分)
- 通过 A:: 类名限定符访问静态成员(2 分) (将 count 声明为 public 直接访问也可以，但仍要 A::)
- 通过合理的代码段动态增减对象个数，展示计数结果(1 分)

(2)(8 分)请实现图形类 Shape，其包含属性颜色 color 以及求解图形周长和面积的方法，并实现其两个子类圆形 Circle 和矩形 Rectangle，其中圆形有属性半径 radius，矩形有属性宽 width 和高 height。请注意根据事物特征按需使用虚函数等技术。请分别在 header.h 和 implement.cpp 中编写上述类的声明和实现，并在 main.cpp 中提供合理的测试代码。

header.h:

```
#include <string>
using namespace std;
class Shape {
private:
	string color;
public:
	Shape(string color);
	virtual float get_C() = 0;
	virtual float get_S() = 0;
};
class Circle : public Shape {
private:
	float radius;
public:
	Circle(string color, float radius);
	float get_C();
	float get_S();
};
class Rectangle : public Shape {
private:
	float width;
	float height;
public:
	Rectangle(string color, float width, float height);
	float get_C();
	float get_S();
};
```

implement.cpp:

```
#include "header.h"
Shape::Shape(string color) : color(color) {}
Circle::Circle(string color, float radius) : Shape(color) {
	this->radius = radius < 0.f ? 0.f : radius;
}
float Circle::get_C() {
	return 2 * 3.14f * radius;
}
float Circle::get_S() {
	return 3.14f * radius * radius;
}
Rectangle::Rectangle(string color, float width, float height) : Shape(color) {
	this->width = width < 0.f ? 0.f : width;
	this->height = height < 0.f ? 0.f : height;
}
float Rectangle::get_C() {
	return 2 * (width + height);
}
float Rectangle::get_S() {
	return width * height;
}
```

main.cpp:

```
#include <iostream>
#include "header.h"
using namespace std;
int main() {
	Circle c("red", 3.5f);
	Rectangle r("blue", 3.f, 4.f);
	cout << c.get_C() << endl;
	cout << c.get_S() << endl;
	return 0;
}
```

得分点：

- 在 header.h 内编写类声明，在 implement.cpp 中通过类名限定符编写成员函数的定义 (2 分)
- 按照需求定义了各个类的数据成员(2 分)
- 将 Shape 的 get_C 和 get_S 声明为纯虚函数并在 Circle 和 Rectangle 中通过重写提供具体计算方法(2 分)
- 按照圆和矩形的实际情况进行合理的构造参数检查以及合理地维护了构造函数参数列表等(1 分)
- main.cpp 中包含了 header.h 头文件并提供了合理的测试代码(1 分)