一、选择题

1. 下面说法正确的是A

A.流操作算子也同样适用于文件流

B. fstream 只能从⽂件读取数据，不能向⽂件写⼊数据。//可以

C. iostream类可以用于读写文件。//不可以 它是与控制台关联的

D. istream 是⽤于输出的流类。//输入

2. 下列四句话中正确的有\_3\_\_个。

1.函数模板中可以有不止一个类型参数。//对

2编译器必须通过函数模板调用语句中的实参类型，来实例化函数模板的中的类型参数//对

3 不可以定义和函数模板同名的函数。//错 可以重载 参数数量不同

4 函数模板中的类型参数可以用来表示函数模板的返回值类型。//对

3. 以下函数或类使用时一定不会出现二义性的有：B

A. void x(int y); void x(int p, int q=0);//会出现

B. class x{public: int p;}; class y{public:int p;}; class z: public x, public y{};//对 名字不同

C. template<class Foo> int print(Foo x); template<class Bar> void print(Bar y);//会出现二义性

D. template<class Foo> int print(Foo x); void print(int y);//会出现 比如输入一个int型的

4. 以下语法正确的是：B

A. void foo(int a=0, int b){}//错误 不能第一个初始化第二个不初始化

B. int arr[]{1,2,3};//正确的

C. int & r = 2+3;//不能引用常值

D. template < class T1, T2>//T2前要加class

二、填空题

1. 填写代码，按要求输出结果

#include <cstdio>

#include <iostream>

using namespace std;

class f {

// 在此处补充你的代码

};

int main() {

cout << f(3)(5) << endl;

cout << f(4)(10) << endl;

cout << f(114)(514) << endl;

cout << f(9,7) << endl;

cout << f(2,3) << endl;

cout << f(2,5) << endl;

}

样例输出

2

6

400

63

6

10

2. 程序填空，编写Dog类，使得程序按样例进行输出

#include <iostream>

using namespace std;

class Mammal {

public:

Mammal() { cout << "Mammal constructor...\n"; }

virtual ~Mammal() { cout << "Mammal destructor...\n"; }

Mammal (const Mammal & rhs);

virtual void Speak() const { cout << "Mammal speak!\n"; }

virtual Mammal\* Clone() { return new Mammal(\*this); }

};

class Dog : public Mammal {

// 在此处补充你的代码

};

Mammal::Mammal (const Mammal & rhs) {

cout << "Mammal Copy Constructor...\n";

}

Dog::Dog(){ cout << "Dog constructor...\n"; }

Dog::~Dog(){ cout << "Dog destructor...\n"; }

void Dog::Speak()const { cout << "Woof!\n"; }

int main() {

Mammal\* x=new Dog, \*y=new Mammal;

Mammal\* x1=x->Clone(), \*y1=y->Clone();

x1->Speak(); y1->Speak();

return 0;

}

样例输出

Mammal constructor...

Dog constructor...

Mammal constructor...

Mammal Copy Constructor...

Dog copy constructor...

Mammal Copy Constructor...

Woof!

Mammal speak!

#include <cstdio>

#include <iostream>

using namespace std;

class f {

// 在此处补充你的代码 f(3)(5)前一个是初始化 后一个是重载小括号 当伪函数用

private:

int x;

int ans;

//int (\*p)(int x);//函数指针

public:

f(int xx, int yy) {

ans = xx \* yy;

}

f(int xx) {

x = xx;

}

friend ostream& operator<<(ostream& os, f ff) {

os << ff.ans;

return os;

}

int operator()(int xx) {

return xx - x;

}

};

int main() {

cout << f(3)(5) << endl;

cout << f(4)(10) << endl;

cout << f(114)(514) << endl;

cout << f(9, 7) << endl;

cout << f(2, 3) << endl;

cout << f(2, 5) << endl;

}

#include <iostream>

using namespace std;

class Mammal {

public:

Mammal() { cout << "Mammal constructor...\n"; }

virtual ~Mammal() { cout << "Mammal destructor...\n"; }

Mammal(const Mammal& rhs);

virtual void Speak() const { cout << "Mammal speak!\n"; }

virtual Mammal\* Clone() { return new Mammal(\*this); }

};

class Dog : public Mammal {

// 在此处补充你的代码

public:

Dog();

~Dog();

void Speak()const;

Dog(const Dog& rhs):Mammal(\*this) {

cout << "Dog copy Constructor...\n";

}

Mammal\* Clone() { return new Dog(\*this); }

};

Mammal::Mammal(const Mammal& rhs) {

cout << "Mammal Copy Constructor...\n";

}

Dog::Dog() { cout << "Dog constructor...\n"; }

Dog::~Dog() { cout << "Dog destructor...\n"; }

void Dog::Speak()const { cout << "Woof!\n"; }

int main() {

Mammal\* x = new Dog, \* y = new Mammal;

Mammal\* x1 = x->Clone(), \* y1 = y->Clone();

x1->Speak(); y1->Speak();

return 0;

}