******华中科技大学计算机科学与技术学院2022~2023第一学期**

解答内容不得超过装订线

**“ C++程序设计”考试试卷 (A卷)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **考试方式** | **开卷** | **考试日期** | **2022-11-26** | **考试时长** | **150 分钟** |
| **专业班级** |  | **学 号** |  | **姓 名** |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **题号** | **一** | **二** | **三** | **四** | **五** | **六** | **总分** | **核对人** |
| **分值** | 15 | 20 | 20 | 15 | 15 | 15 | 100 | 马光志 |
| **得分** |  |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| **分 数** |  |
| **评卷人** |  |

1. **单选题：请从4个选项中选择一个最合适的选项作为答案（15分：每小题3分）。**

1. 关于定义“union A { int x; const int y = 3; }a = { 5 };”，如下叙述哪个\_\_B\_\_正确:

A. a.x=3，a.y=3 B. a.x=5，a.y=5

C. a.x=3，a.y=5 D. a.x=5，a.y=3

2. 关于“void f(int&) {}; void f(const int&) {}; void f(int&&) {}; int x=2;”，如下叙述\_\_D\_\_正确:

A. f(x)调用void f(int&)，f(2)调用void f(const int&)

B. f(x)调用void f(const int&)，f(2)调用void f(int&&)

C. f(x)调用void f(int&&)，f(2)调用void f(int&)

D. f(x)调用void f(int&)，f(2)调用void f(int&&)

3. 对于说明“int &f( ); int \*&g( );”及其函数调用f( )和g( )，如下叙述\_\_C\_正确:

A. 调用f( )和调用g( )均不可被赋值 B.调用f( )不可被赋值，调用g( )可被赋值

C. 调用f( )和调用g( )均可被赋值 D.调用f( )可被赋值，调用g( )不可被赋值

4. 对于定义“struct A{ void f( ){}}a, \*p; struct B: A{ virtual void f( ){}}b, \*q=&b;”，若p=q如下叙述\_\_B\_\_正确:

A. p->f( )调用A::f( ), q->f( )调用A::f( ) B. p->f( )调用A::f( ), q->f( )调用B::f( )

C. p->f( )调用B::f( ), q->f( )调用B::f( ) D. p->f( )调用B::f( ), q->f( )调用A::f( )

5. 对于定义“struct A { const int x = 3; volatile int y = 4; }const a;”，如下叙述\_\_B\_\_正确:

A. a.x和a.y的类型均为const int; B. a.x类型为const int，a.y的为const volatile int;

C. a.x和a.y的类型均为const volatile int; D. a.x类型为const volatile int，a.y的为const int;

|  |  |
| --- | --- |
| **分 数** |  |
| **评卷人** |  |

1. **在最多使用单重作用域例如A::x的前提下，在空白处填写以下各类可被访问的成员及其访问权限（20分：根据正确回答的成员个数按比例计算给分）。**

class A { //类A的可访问成员：

int a; //私有成员：int a(A::a);\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

protected: //保护成员：int b(A::b), c(A::c);\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

int b, c; //公有成员：int d(A::d), e(A::e);\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

public:

int d, e;

};

class B: protected A{//类B的可访问成员：

int a; //私有成员：int a(B::a);\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

protected: //私有成员：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

int b, f; //保护成员：int A::b, b(B::b), c(A::c, B::c), d(A::d, B::d), A::e, f(B::f);\_\_\_

using A::d; //保护成员：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

public: //公有成员：int e(B::e), g(B::g)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

int e, g; //公有成员：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

};

struct C: A { //类C的可访问成员：

int a; //私有成员：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

protected: //私有成员：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

int b, c; //保护成员：int A::b, b(C::b), A::c, c(C::c), \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

public: //保护成员：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

int e, g; //公有成员：int a(C::a), d(A::d, C::d), A::e, e(C::e), g(C::g)\_\_\_\_\_\_\_\_\_\_\_\_

using A::d; //公有成员：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

};

struct D: B, C { //类D的可访问成员：

int a; //私有成员：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

protected: //私有成员：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

int b, c; //保护成员：int B::b, C::b, b(D::b), B::c, C::c, c(D::c), B::d, B::f\_\_\_\_\_\_\_\_

public: //保护成员：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

int e, f; //公有成员：int C::a, a(D::a), B::e, C::d, C::e, e(D::e), f(D::f), B::g, C::g\_\_

}; //公有成员：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| **分 数** |  |
| **评卷人** |  |

1. **回答main中每行语句的输出结果（20分：前四个语句的输出每个3分，后两个语句的输出每个4分）。**

#include <iostream>

using namespace std;

struct A { A( ) { cout << 'A'; } };

struct B { A a; B( ) { cout << 'B'; } };

struct C : virtual A { C( ) { cout << 'C'; } };

struct D : B, virtual C { D( ) { cout << 'D'; } };

struct E : virtual A, virtual D {

D d;

E( ) : A( ) { cout << 'E'; }

};

struct F : B, virtual C, virtual D, E{

D d; E e;

F( ) { cout << 'F'; }

};

int main( ) {

A a; cout << '\n'; //输出=A

B b; cout << '\n'; //输出=AB

C c; cout << '\n'; //输出=AC

D d; cout << '\n'; //输出=ACABD

E e; cout << '\n'; //输出=ACABDACABDE

F f; cout << '\n'; //输出=ACABDABACABDEACABDACABDACABDEF

}

|  |  |
| --- | --- |
| **分 数** |  |
| **评卷人** |  |

1. **综合分析并指出以下程序中下划线位置可能出现的语法错误及其原因 (共15分：每错约1分)。**

class A {

int a;

public:

const int b;

int c;

virtual A(\*g)( ) = 0; //错误1：数据成员g不能用virtual 说明\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A( ) { a=b=c= 0; } //错误2：( )后没对b初始化\_错误3：函数体中不能初始化b\_

virtual A(int x): a(x),b(x),c(x){ }; //错误4：构造函数不能定义为virtual\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

} a = (4, 3);

class B: A {

using A::a; //错误5：不能using私有的A::a\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

~B( ){ }

public:

int d;

friend int operator[ ](int) { return 2; };//错误6：[ ]只能重载为实例函数成员: friend说明非成员

B(int x, int y) { d = x + y ; };

} b(5, 6); //错误7：b无法调用私有的析构函数 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

class C: B {

namespace D{ int z; }; //错误8：不能在类体中定义名字空间\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

public:

~C(int x) { }; //错误9：析构函数不能有参数 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

friend void main( );

} c; //错误10：类B未提供无参构造函数,无法生成C::C( )\_\_\_\_\_\_\_\_

template<typename T> const T x = 3;

void main( ){

int i = a.a; //错误11：main不能访问私有成员a.a\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

int A::\* p = static\_cast<int A::\*>(&A::b); //错误12：不能去掉&A::b的const属性\_\_\_\_\_\_\_

char &y = x<char>; //错误13：只读变量x<char>不能被可写引用变量y引用\_\_\_\_\_

i = b.b + c.d; //错误14：main不能访问私用成员b.b \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

i = b.\*p; //错误15：A非B的父类且main不是B的友元，不能访问b.\*p

}

|  |  |
| --- | --- |
| **分 数** |  |
| **评卷人** |  |

1. **请填入自己学号的最后一位十进制数字，计算main函数中变量i在每条赋值语句执行后的值 (共15分: 每小题2.5分)。**

int x = 填写自己学号最后一位十进制数 , y = x + 3;

struct A {

int x = ::x + 2;

static int& y;

public:

operator int( )const { return x + y; }

int& v(int& x) {

for (int y = 1; x < 301; x ^= y, y++)

if (x > 300) { x -= 31; y -= 2; }

return ++x;

}

A& operator++( ) { ++x; ++y; return \*this; }

A(int x, int y = ::y + 3) { A::y = y; }

};

int& A::y = ::y;

void main( ) {

A a(2, 7), b(5);

int i, &j = i, A::\*p = &A::x;

auto f = [&i](int x) mutable {++i; return x + i; };

i = a.y; //i=

j = a.x; //i=

i = a.\*p; //i=

i = ++a; //i=

i = b.y + f(1); //i=

(b.v(i)=2) += f(3); //i=

}

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **学号** | i = a.y | j = a.x | i = a.\*p | i = ++a | i = b.y + f(1) | (b.v(i)=2) += f(3) |
| **0** | **10** | **2** | **2** | **14** | **27** | **33** |
| **1** | **10** | **3** | **3** | **15** | **28** | **34** |
| **2** | **10** | **4** | **4** | **16** | **29** | **35** |
| **3** | **10** | **5** | **5** | **17** | **30** | **36** |
| **4** | **10** | **6** | **6** | **18** | **31** | **37** |
| **5** | **10** | **7** | **7** | **19** | **32** | **38** |
| **6** | **10** | **8** | **8** | **20** | **33** | **39** |
| **7** | **10** | **9** | **9** | **21** | **34** | **40** |
| **8** | **10** | **10** | **10** | **22** | **35** | **41** |
| **9** | **10** | **11** | **11** | **23** | **36** | **42** |

|  |  |
| --- | --- |
| **分 数** |  |
| **评卷人** |  |

1. **对于一元多次函数，其中和皆为int类型的整数，注意n>=0, i为x的指数。现定义函数类F如下，试对其中的每个实例函数成员编程(共15分：每个函数1.5分)。**

class F {

int\*const a; //用于存放函数的多项式系数**，存放于a[0]**

const int n; //多项式的系数个数

public:

F(int n, ...); //创建n个系数的函数，n<0抛出异常，…为n个系数

F(const F& f); //根据已知函数f深拷贝构造新函数

F(F&& f)noexcept; //根据已知函数f移动构造新函数

F& operator=(const F& f); //深拷贝赋值运算符的重载

F& operator=(F&& f)noexcept; //移动赋值运算符的重载

F operator+(const F&f)const; //两个函数相加

operator int( )const noexcept; //获得函数多项式的系数个数n

int& operator[ ](int i); //获得第i个系数(i从0开始)，不存在则抛出异常

int operator( )(int x)const; //计算实参为x时的函数值

~F( )noexcept; //析构函数

};

**答：**

F::F(int n, ...): a(n<0?throw "n<0!": (n==0?nullptr: new int[n])), n(a ? n : 0) {

if (n>0 && a == nullptr) throw "memory not enough!";

int\* p = &n + 1;

for (int x = 0; x < n; x++) a[x] = p[x];

}

F::F(const F& f) : a(f.n>0?new int[f.n]:nullptr), n(a ? f.n : 0) {

if (n > 0 && a == nullptr) throw "memory not enough!";

for (int x = 0; x < n; x++) a[x] = f.a[x];

}

F::F(F&& f)noexcept : a(f.a), n(f.n) {

(int \*&)f.a = nullptr; //或\*(int\*\*)&f.a=nullptr;

(int&)f.n = 0; //\*(int\*)&f.n = 0;

}

F& F::operator=(const F& f) {

if (this == &f) return \*this;

if (a) delete a; //或delete []a;

if (f.n == 0) {

(int\*&)a = nullptr;

(int&)n = 0;

return \*this;

}

(int\*&)a = new int[(int&)n=f.n];

if (a == nullptr) throw "memory not enough!";

for (int x = 0; x < n; x++) a[x] = f.a[x];

return \*this;

}

F& F::operator=(F&& f)noexcept {

if (this == &f) return \*this;

if (a) delete a; //或delete []a;

(int\*&)a = f.a; //或\*(int\*\*)&a=f.a;

(int&)n = f.n; //\*(int\*)&n = f.n;

(int\*&)f.a = nullptr; //或\*(int\*\*)&f.a=nullptr;

(int&)f.n = 0; //\*(int\*)&f.n = 0;

return \*this;

}

F F::operator+(const F& f)const {

F r(n > f.n ? \*this: f);

if (n > f.n)

for (int x = 0; x < f.n; x++) r.a[x] += f.a[x];

else

for (int x = 0; x < n; x++) r.a[x] += a[x];

return r;

}

F::operator int( )const noexcept { return n; }

int& F::operator[ ](int i) {

if (i < 0 || i >= n) throw "subscrption error!";

return a[i];

}

int F::operator( )(int x)const {

if (n == 0) throw "can not call a null function!";

int s = a[n-1];

for (int k = n - 1; k >= 1; k--) s = s \* x + a[k - 1];

return s;

}

F::~F( )noexcept {

if (a!=nullptr) {

delete a; //或delete []a;

(int\*&)a = nullptr;

(int&)n = 0;

}

}