## 面向对象程序设计模拟试卷

一. 单选题(15)。

1. 关于定义“struct A{int x; const int y=3;};volatile A a={1};”，如下叙述那个\_\_ \_\_\_正确:

A. a.x是int、a.y是const int类型 B. a.x是volatile int、a.y是const int类型

C. a.x是int、a.y是const volatile int类型D. a.x是volatile int、a.y是const volatile int类型

2. 关于inline constexpr int &f(int &&a)的调用叙述\_\_\_\_\_正确:

A.该用常量做实参调用，返回传统右值 B. 该用常量做实参调用，返回传统左值

C.可用变量做实参调用，返回传统右值 D. 可用变量做实参调用，返回传统左值

3. 若A是B的父类，B是C的父类，则catch应按如下哪种\_\_\_\_\_放置:

A. catch(A x)在catch(B x)的前面， catch(B x)在catch(C x)的前面

B. catch(A x)在catch(B x)的后面， catch(B x)在catch(C x)的后面

C. catch(A x)在catch(B x)的前面， catch(C x)在catch(A x)的前面

D. catch(A x)在catch(B x)的后面， catch(C x)在catch(A x)的后面

4. 对于定义struct A{int x; int y:2; }m;下列叙述\_\_\_\_\_正确:

A. m.x是有址左值, m.y是无址左值 B. m.x是无址左值, m.y是有址左值

C. m.x是有址右值, m.y是无址右值 B. m.x是无址右值, m.y是有址右值

5. 对于定义“char \* &&f( );”，如下哪个语句是错误的\_\_\_\_\_:

A. f( )=(char\*) "abcd"; B. \*f( )= 'A';

C. char \*p=f( ); D. \*f( )= 1 ["ABC"];

二. 在最多使用一级作用域“：：”访问如A::c的情况下，指出各类可访问的成员及其访问权限(20) 。

 

类A的可访问成员：

private:

protected:

public:

类B的可访问成员：

private:

protected:

public:

类C的可访问成员：

private:

protected:

public:

类D的可访问成员：

private:

protected:

三. 指出main中每行的输出结果(20) 。

#include <iostream>

using namespace std;

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

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

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

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

struct E : A {

C c;

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

};

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

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

};

void main() {

A a; cout << '\n'; //输出=

B b; cout << '\n'; //输出=

C c; cout << '\n'; //输出=

D d; cout << '\n'; //输出=

E e; cout << '\n'; //输出=

F f; cout << '\n'; //输出=

}

四. 指出以下程序的语法错误及其原因(15) 。

class A {

static int a=0;

protected:

int b;

public:

int c;

A(int) {};

inline constexpr operator int() { return b; };

} a(1, 2);

class B: A {

B(int m) { b = m; };

using A::b;

virtual int d;

int e;

public:

int b;

friend B& operator =(const B&b){return \*this;}

virtual B(int, int);

} b = 5;

class C: B {

public:

C operator++(double) { return \*this; };

}c;

int main( ) {

int \* A::\* p, i;

i = a.a;

i = A(4);

i = b.c;

p = &A::c;

i = b;

return;

}

五. 指出main变量i在每条赋值语句执行后的值(15) 。

int x=\_\_\_\_\_\_\_(请填入本人学号最后一位数字), y=x+30;

struct A{

static int x;

int y;

public:

operator int( ){ return x-y; }

A operator ++(int){ return A(x++, y++); }

A(int x=::x+2, int y=::y+3){ A::x=x; A::y=y; }

int &h(int &x);

};

int &A::h(int &x)

{

for(int y=1; y!=1|| x<201; x+=11, y++) if(x>200) { x-=21; y-=2;}

return x-=10;

}

int A::x=23;

void main( ){

A a(54, 3), b(65), c;

int i, &z=i, A::\*p=&A::y;

z=b.x;

i=a.x;

i=c.\*p;

i=a++;

i=::x+c.y;

i=a+b;

b.h(i)=7;

}

六. 英文字典的单词区分大小写，并且不能存放重复的单词，定义如下字典类DIC中的函数成员(15)。

#define \_CRT\_SECURE\_NO\_WARNINGS //防止strcpy等出现安全报警

#include <string.h>

class DIC {

    char\*\* const e; //用于存放单词，部分大小写

    const int m; //能够存放的最大单词个数

    int c; //已经存放的单词个数

public:

    DIC( )noexcept; //用于构造空字典

    DIC(int m); //用于构造单词个数最多为m的字典

    DIC(const DIC&); //字典深拷贝构造函数

    DIC(DIC&&)noexcept; //字典移动构造函数

    DIC& operator=(const DIC&); //字典深拷贝赋值运算符重载函数

    DIC& operator=(DIC&&)noexcept; //字典移动赋值运算符重载函数

    DIC& operator<<(const char\*); //将一个单词不重复放入字典

    int  where(const char\*); //查找单词在字典中的下标位置，没找到时返回-1

    char\* operator[](int x); //返回下标值为x的单词

    ~DIC()noexcept; //字典析构函数

};

解：