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={1};”，如下叙述那个__D___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__B___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. 若派生类函数不是基类的友元，关于该函数访问基类成员___C__正确:
   A. 只有公有的可被派生类函数访问　　B. 都可以被派生类函数访问
   C. 公有和保护的可被派生类函数访问　　D. 都不对
4. 关于函数的所有缺省参数的叙述__B___正确:
   A. 只能出现在参数表的最左边　　B. 只能出现在参数表的最右边
   C. 必须用非缺省的参数隔开　　D. 都不对
5. 对于定义“char * &&f( );”，如下哪个语句是错误的__A___:
   A. f( )=(char*) "abcd";　　B. *f( )= 'A';
   C. char *p=f( );　　D. *f( )= "ABC"[1];

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) 。

```
class A{
      int   a;
protected:
      int   b;
public:
      int   c;
};
class B: protected A{
      int   d;
protected:
      int   e;
public:
     A::c;
      int   f;
};
```

```
class C:   A{
      int   g;
protected:
      int   h;
public:
      int   i;
};
class D: B, C{
      int   j;
protected:
     B::b;
      int   k;
private:
      int   n;
};
```

类 A 的可访问成员：

private:　　int a;
protected:　　int b;
public:　　int c;

类 B 的可访问成员：

private:　　int d;
protected:　　int b(或 A::b),e;
public:　　int c(或 A::c),f;

类 C 的可访问成员：

private:　　int b(或 A::b), c(或 A::c),g;
protected:　　int h;
public:　　int i;
类 D 的可访问成员：
private:　　int c(或 B::c),e(或 B::e), f(或 B::f),h(或 C::h), i(或 C::i),j,n;
protected:　　int b(或 B::b),k;

三. 指出 main 中每行的输出结果(20) 。

```
#include <iostream.h>
struct A{A( ){ cout<<'A';}};
struct B{B( ){ cout<<'B';}};
struct C: A{C( ){ cout<<'C';}};
struct D: virtual B, C{D( ){ cout<<'D';}};
struct E: A{
    C   c;
    E( ): c( ){ cout<<'E';}
};
struct F: virtual B, C, D, E{
    F( ){ cout<<'F';}
};
void main( ){
    A   a;   cout<<'\n';      //输出=A
    B   b;   cout<<'\n';      //输出=B
    C   c;   cout<<'\n';      //输出=AC
    D   d;   cout<<'\n';      //输出=BACD
    E   e;   cout<<'\n';      //输出=AACE
    F   f;   cout<<'\n';      //输出=BACACDAACEF
}
```

四. 指出以下程序的语法错误及其原因(15) 。

```
class A {
    static int  a=0;//(1)前面有inline或const或constexpr才能初始化,否则在类外初始化
protected:
    int  b;
public:
    int  c;
    A(int) {};
    inline constexpr operator int() { return b; };
}  a(1, 2);   //(2)没有两个参数的构造函数
class B: A {
    B(int m) { b = m; }; //(3)A不存在无参构造函数供“B(int m):”后调用
    using A::b;
    virtual int d;//(4)virtual不能用于说明数据成员
    int  e;
public:
```

```
    int b;
    friend B& operator =(const B& b){return *this;}//(5)等号仅单参，应为实例函数
    static B(int, int); //(6)不能用static定义构造函数
} b = 5; //(7)单参构造函数是私有的，无法访问
class C: B {
public:
    C operator++(double) { return *this; };//（8）后置运算必须int类型定义显式参数
}c;  //(9)类C无法生成无参构造函数初始化c
int main( ) {
    int* A::* p, i;
    i = a.a;  //(10)私有成员，main函数无法访问
    i = A(4);
    i = b.c;  //（11）A继承到B的成员B::c是私有的，main函数无法访问
    p = &A::c;//(12)&A::c的类型为int A::*，与p的类型int* A::*不同，不能赋值给p
    i = b;//(13)B未定义operator int函数，继承自A的operator int私有化，main不能访问
    return;//(14)main要求返回一个整型值
 }
```

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;
```

}

学号尾数	z = b.x	i = a.x	i=c.*p	i= a++	i=::x+c.y	i=a+b	b.h(i)=7
0	2	2	33	-1	33	-33	7
1	3	3	34	0	35	-32	7
2	4	4	35	1	37	-31	7
3	5	5	36	2	39	-30	7
4	6	6	37	3	41	-29	7
5	7	7	38	4	43	-28	7
6	8	8	39	5	45	-27	7
7	9	9	40	6	47	-26	7
8	10	10	41	7	49	-25	7
9	11	11	42	8	51	-24	7

六. 为了没有误差地表示分数，定义分数类 FRACTION 用来表示分数，用整型 numerator 存分子、整型 denominator 存分母；并用*重载分数约简运算、用>重载分数比较运算、用+重载分数加法运算、用*重载分数乘法运算，以及相关的构造函数；可运用最大公约数函数 cmd 约简 (15)。

```
int    cmd(int x, int y){
       int r;
       if(x<y){    r=x;    x=y;    y=r;    }
       while(y!=0){    y=x%(r=y);    x=r; }
       return x;
}
```

解：

```
class FRACTION{ //对于6/7，numerator 存分子 6，denominator 存分母 7
       int      numerator, denominator;
public:
       int operator>(const FRACTION&)const;            //大于比较，例6/7>2/3
       FRACTION(int num, int den=1);                   //num、den 各为分子和分母
       FRACTION operator*( )const;                          //分数约简，*30/36=5/6
       FRACTION operator+(const FRACTION&)const;       //加法，6/7+2/3=32/21
       FRACTION operator*(const FRACTION&)const;//乘法，6/7*2/3=12/21=4/7
};

FRACTION::FRACTION(int num, int den){
      numerator=num;
      denominator=den;
}
int FRACTION::operator>(const FRACTION&f)const{
     double d= denominator *f.denominator;
      return    numerator*f.denominator/d > denominator*f.numerator/d;
}
```

```
FRACTION FRACTION::operator*( )const{
    int    c=cmd(numerator, denominator);
    return FRACTION(numerator/c, denominator/c);
}
FRACTION FRACTION::operator+(const FRACTION&f)const{
    int    n= numerator*f.denominator+denominator*f.numerator;
    int    d= denominator*f.denominator;
    return *FRACTION(n, d);      //对运算结果进行约分运算
}
FRACTION FRACTION::operator*(const FRACTION&f)const{
    return *FRACTION(numerator*f.numerator,    denominator*f.denominator); //约分
}
```