C++复习 试卷总结篇

一、选择题

1. 类 A 是一个包含纯虚函数的抽象类，下列说明语法正确的是__C____

A. A a;　　B. A f( );

C. A &f( );　　D. A f(A);

C:抽象类不允许定义实例,不能作为函数参数,但可以返回函数的引用

2. 使用 exit 退出程序 ,关于对象自动析构，哪个叙述正确__B____

A.不析构全局对象但析构局部对象

B.析构全局对象但不析构局部对象

C.全局对象和局部对象都不析构

D.全局对象和局部对象都析构

B:基础题

3. 对于如下程序：

```
#include <stdio.h>
struct A {
   A( ) { printf(别理我); }
    A(const cha r *s) { printf(s); }
} a("烦着呢");
A f( ) { printf("一边去"); return a; }
void main(void) { A f( ); }
```

关于程序的输出 ，哪个叙述是正确的 ___A____

A. 输出为 烦着呢 ；

B. 输出为 烦着呢 别理我；

C. 输出为 一边去 ；

D. 输出为 烦着呢 一边去；

A:首先是构造全局变量 a,调用 A(const cha r *s) { printf(s); },最后 main 函数内部的 A f()仅仅是函数声明,不作任何作用

4. 对于 int x；如下运算错误的是___A___

A. x++ ++　　B.++　++x

C. (++x)++　　D.++ (++x)

A:x++返回右值,不允许使用++(后缀或者前缀)

5. 对于 int x , int &y 最好引用如下哪个表达式__A____

A. x+=3;　　B. x+4 ;

C. x++;　　D. (--x)--;

A:只有 A 是左值元素

6. 对于如下程序：

```
#include <stdio.h>
struct A{
        virtual int f( ) { printf("A"); }
    virtual int g( ) { printf("B"); }
} a, *p;
struct B: A {
    int f( ) { A::f( ); printf("C"); }
    int g( ) { printf("D"); }
};
void main( ) { p=&b; p->f( ); p->g( ); }
```

则程序的输出为_____：

A. AB

B. CD

C. ACD

D. ABCD

C:首先先调用 f,根据虚函数的定义,p 虽然是 A 类指针,但是却调用 B 类的元素,先输出 A 和 C,再输出 D

7. 如下程序：

```
#include <stdlib.h>
struct A { A( ) { } }a;
void main( ) { A b; exit(0 ); }
```

则关于对象 a, b 的析构，如下哪个叙述正确_____：

A. 无析构函数都没有析构;

B. a 析构了但是 b 没析构;

C. b 析构了但是 a 没析构;

D. 都析构了;

B:基础题

8. 对于定义“const char *&g( );”，如下哪个语句是错误的_____:

A. g( )= "abcde";

B. *g(　)= 'A';

C. const char *p=g(　);

D. const char *&q =g(　);

B:const char *表示指针指向的元素是不允许改变的,但指针的指向是允许改变的,所以说选择 B

9. 对于如下定义：

```
struct A { virtual int f( ) { return 1;} }a;
```

```
struct B: A {
        int f( ) const { return 2; }
    int f( ) volatile { return 3; }
        int f( ) const volatile { return 4; }
} c;
int main(int argc, char* argv[ ]) { A *p = &c;
return p->f( ); }
```

主函数 main 的返回值是_____:

A. 1

B. 2

C. 3

D. 4

A:这不是虚函数,这是重载函数,f 的 this 指针只与 A 内的 f 匹配

10. 关于运算符重载的叙述哪个正确_____:

A. 可以改变优先级和结合性

B. 可以改变优先级，不能改变结合性

C. 不能改变优先级，可改变结合性

D. 都不能改变

D:基础题

二、判断作用域

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

A:
private: a
protected: b, f
public: c, d

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

B:
private: d
protected: c, e, A::(b, c, d, f)
public: f

```
class C: public A {
    int   g;
protected:
    int   h, d;
public:
    int   c, i;
};
```

C:
private: g
protected: h, d, A::(b, f)
public: c, i, A::(c, d)

```
struct D: B, public C {
    int   j;
protected:
    int   k, c;
public:
    int   n;
};
```

D:
private:
protected: k, c, B::(c, e, A::(b, c, d, f)), C::(h, d, A::(b, f))
public: j, n, B::(f), C::(c, i, A::(c, d))

就照抄即可,利用 A::这个运算符用于区分

三、多继承,虚基类构造

构造的顺序:

1、先构造构造树内的虚基类,方向:左到右,上到下,构造过的虚基类不重复构造,对于虚基类的构造应该遵循递归原则,就是构造虚基类的时候,虚基类里面的普通基类也要构造

2、再构造构造树内的普通类,方向:左到右,构造过的虚基类不重复构造,就不进入虚基类里面进行访问了,也就是说 D 内含有虚基类 C,我们在访问 D 时除了 C 的部分都会输出,D:(AC)BD

3、在构造类内部声明的类,构造过的虚基类重复构造,就直接递归输出即可

```
#include <iostream>
struct A { A( ) { cout << 'A'; } };
struct B { B( ) { cout << 'B'; } };
struct C: A { C( ) { cout << 'C'; } };
struct D: B, virtual C { D( ) { cout << 'D'; } };
//struct E: virtual A, virtual D {
struct E: A, virtual D {
        D   d;
        E( ): A( ) { cout << 'E'; }
};
struct F: B, virtual C, E, virtual D {
        D   d;
        F( ) { cout << 'F'; }
};

void main( ) {
    A   a; ;   //输出=  A
    B   b;     //输出=  B
    C   c;     //输出=  AC
    D   d;     //输出=  ACBD
    E   e;     //输出=  ACBDAACBDE
    F   f;     //输出=ACBDBAACBDEACBDF
}
```

四、改错

```
    class A {
        int   a;
    protected:
        const   int   &b;
        ~A( ) { }
     public:
        int   c;
        virtual   A   (*g)(int);
//(1) g 是一个变量，不能定义为 virtual
        A(int x) { a = x; };
//(2) b 没有初始化
    } x = (4, 3);
    //(3) x 不能析构

    class B: A {
        int   d;
     public:
        A::b;            //(4) ?
        friend   int   operator( )(int) { return 2; };
        //(5) 不能声明为友元
        B(int x, int y, int z) { d = x + y + z; };
//(6) 没有说明基类的构造方法
    } b(5, 6, 7);

    //(7) 不能生成对象 b

    struct C: B {
        int z;
     public:
        ~C(int x) { z = x; };
    } c;
    //(8) 没有构造函数；(9) 不能生成对象 c
    void   main( ) {
        int   A::*p = &c.z;
//(10) 成员指针不能指向物理地址
        int   i = x.b;
//(11) 不能访问类 A 的保护成员
        i = x;
//(12) 类 A 没有强制类型转换函数
        i = b.b;          //(13) ?
        i = i + c.d;
//(14) 不能访问类 B 的私有成员 d
        i = b.*p;
//(15) p 不是类 A 的成员指针
    }
```

```
class A {
    int &a;
protected:
    const int &b;
    ~A( ) { }
public:
    int c;
    virtual A&   (*g)( );
//(1)virtual 不能用于定义数据成员
    A(int x) { b = x; };
//(2)没在函数体前初始化 a 和 b, (3)不能在体内初始 b
} a = (1, 2, 3);
//(4)a 不能调用私有的析构函数~A( )

class B: A {
    int d;
public:
    A::b;
//(5)不能改变访问权限，只能恢复（有问题）
    static int operator( )(int) { return 3; };
//(6)运算符( )只能为实例函数成员
    B(int x, int y, int z):A(x) { d=x+y+z; };
} b(2, 3, 7);

struct C: B{
    int z;
protected:
    virtual ~C( ) {   };
} c;
//(8)c 不能调用保护的析构函数~C( )

void   main( ) {
    int *A::*p = &c.z;
//(9)不能将 int *类型的值赋给 int *A::*类型的变量 p
    int   i = a.b;
//(10)main 不能访问保护成员 A::b（有问题）
    i = a;
//(11)对象 a 无法转换为整数
    i = b.b;
//(12) main 不能访问私有成员 B::b
    i = c.d;
//(13) main 不能访问私有成员 C::d
    i = b.*p;
//(14)不能将 int *类型的值赋值给整型变量 i
    return 1;
//(15)main 无需返回值
}
```

五、输出结果

```
int     x = 学号最后一位十进制数      ,
y=x+3;
struct A {
    int x;
    static int &y;
public:
    operator int( )const { return x + y; }
    int &v(int &x) {
        for(int y=1; x<201; x^=y, y++)
            if(x>200) { x -= 31; y -= 2;}
        return ++x;
    }
    A  &operator++( ) { ++x; ++y; return *this; }
    A(int x = ::x + 2, int y = ::y + 3) { A::x = x;   A::y = y; }
};
int & A::y = ::x;
void main( ) {
    A   a(2, 3),   b(a),   c;
    int   i,   &j=i,   A::*p=&A::x;
    i = a.y;                    //i=
    j = a.x++;                  //i=
    i = a.*p;                   //i=
    i = ++a;                    //i=
    i = b.y + ::y;              //i=
    (b.v(i) = 2) += 3;          //i=
}
i = a.y;
j = a.x++;      j 改变,i 随之改变  a.x=3,但返回 2
i = a.*p;
E.g M=1
A a(2,3) a.x=2,A::y=3 ::x=3,::y=4
b (a)     b.x=2,A::y=3,::x=3,::y=4
C          c.x=3+2,A::y=4+3=7,::x=3,::y=4
          //i=7
          //i=2
          //i=3(取 x)
```

```
i = ++a; a.x=4,A::y=8,(int)a=4+8=12          //i=12
i = b.y + ::y;          b.y=8,::y=4          //i=12
(b.v(i) = 2) += 3;                           //i=5
```

```
int   x = _____          //学号最后一位十进制数
int   y = x + 3;
struct A {
    int   x;
    static   int   &y;
public:
    operator int( ) const { return x + y; }
    int   &v(int &x) {
        for(int y = 1; x < 301; x ^= y++) {
            if(x > 300) { x -= 31; y -= 2;
            }
            return ++x;
        }
    A   &operator++(   )   {   ++x;      ++y;
return *this; }
    A(int x = A::y + 2,   int y = ::x + A::y)
{ A::x = x + 1;   A::y = y + 2; }
};
int   &A::y = ::y;
void main( ) {
    A   a(3, 4), b(a),   c;
    int   i, &j = i;
    int   A::*p = &A::x;
    j = a.x;                  // i =
    i = a.y;                  // i =
    i = a.*p;          // i =
    i = ++a;           // i =
    i = b.y + ::y;            // i =
    (b.v(i) = 3) += 2;        // i =
}
```

答案（假设学号最后一位 = m）：

```
void   main( ) {
    A a(3,4);          //a.x=4, ::y=6
    A b(a);            //b(a) 浅拷贝构造: b.x=4, ::y=6
    A c;
    //c(::y+2,::x+::y)=c(8,m+6), c.x=9, ::y=m+8
    int   i, &j = i;
    int A::*p = &A::x;
    j = a.x;           //i=4
    i = a.y;           //i=m+8
    i = a.*p;          //i=4
    i = ++a;           //++a: a.x=5,::y=m+9, i=m+14
    i = b.y + ::y;          //i=2m+18
    (b.v(i) = 3) += 2;      //i=5
};
```

六、程序设计

```
PER::PER(int p): n(p),f(new int [p])
{
    c = 0;
    for(int k = 1; k < n; k++)
    {
        if((n % k) == 0) f[c++] = k;
    }
    int sum = 0;
    for(int k = 0; k < c; k++) sum += f[k];
    if(sum != n) c = -1;
}

PER::~PER()
{   if(f) { delete[] f; *(int *)&f = 0;   }   }

//深拷贝构造函数
PER::PER(const PER &p): n(0),f(0)
{   *this = p;   }

//深拷贝赋值运算
PER &PER::operator=(const PER &p)
{
    this->~PER();
    *(int *)&n = p.n;
    *(int **)&f = new int[n];
    c = p.c;
```

```
    for(int k = 0; k < c; k++) f[k] = p.f[k];
    return *this;
}

//若不是完全数则返回 0，否则返回 c
PER::operator int( ) const
{   return c > 0 ? c : 0;   }

//返回 k 所指示的因子，若 k < 0 或 k >= c 返回 0
int PER::operator[](int k) const
{   return k < 0 || k >= c ? 0 : f[k];   }

MAP::MAP(int n): e(new int[n*(n-1)][2]),n(e?n:0), c(0) { }

MAP::MAP(const MAP& m): e(new int[m.n*(m.n-1)][2]),n(e?m.n:0){
    for(c=0; c<m.c; c++){
        e[c][0] = m.e[c][0];
        e[c][1] = m.e[c][1];
    }
}

MAP& MAP::operator=(const MAP &m) {
    if(e) delete e;
    *(int (**)[2])&e = new int[m.n*(m.n-1)][2];
    *(int *)&n = e?m.n:0;
    for(c=0; c<m.c; c++){
        e[c][0] = m.e[c][0];
        e[c][1] = m.e[c][1];
    }
    return *this;
}

MAP& MAP::operator( )(int v0, int v1) {//连接顶点 v1 和 v2 成为边
    int t=v0;
    if(v0==v1 || v0<0 || v0>=n || v1<0 || v1>=n) return *this;
    if(v0>v1) { v0 = v1; v1 = t; }
    for(t=0; t<c; t++)  if(e[t][0]==v0 && e[t][1]==v1) return *this;
    e[c][0] = v0;
    e[c++][1] = v1;
    return *this;
}
MAP::~MAP( ) {    //析构函数
    if(e) { delete e; *(int (**)[2])&e=0; *(int *)&n=0; c=0; }
}
int (*MAP::operator[ ](int x)) [2] {   return e + x;   }
```