1. 当函数为const类型 时，不能修改调用对象的非静态数据成员，但如果该数据成员的存储类为 mutable，则该数据成员就可以被修改，因此mutable一般用于在const函数中需要修改的非静态数据成员前；mutable 说明数据成员为机动数据成员，不能用于静态数据成员，因此不能同时用 const、volatile 或 static 修饰。
2. 静态成员指针存放成员地址，实例成员指针存放成员偏移；静态成员指针可以移动，普通成员指针不能移动；实例成员指针可以强制转换类型，实例成员指针不能强制转换类型。
3. （1）a.p = &A::a;错误，右边是char \*A::\*类型，左边是char A::\*，不能将右式转换为左式；

（2）a.q = &A::geta;错误，a.q既不是取函数q的地址，也不是调用该函数；

（3）a.r = a.geta;错误，a.geta既不是取该函数地址，也不是调用该函数；

（4）a.r = &a.geta;错误。

1. （1）static int x = 1;和static volatile int w = 4;错误，在类内初始化需要为常量；

（2）static const float u = 1.0f;错误，const float类型只能在类外初始化；

（3）int A::y = 22;声明与const int y冲突；int volatile A::z = 33;同理

1. 均正确
2. （1）static friend int f( );中static和friend不能同时出现；

（2）friend int A( );错误，因为friend int A::A()显然错误；

STACK::STACK(int max):max((stk = new char[max]) ? max : 0),top(-1)

{

}

STACK::~STACK()

{

char ch;

for (int i = 0; i < max; i++)

pop(ch);

delete stk;

\*(const\_cast<int\*>(&max)) = 0;

top = 0;

}

int STACK::push(char v)

{

if (top==max) return 0;

else

{

stk[++top] = v;

return 1;

}

}

int STACK::pop(char& v)

{

if (top==-1) return 0;

else

{

v = stk[top--];

printf("%c", v);

return 1;

}

}

REVERSE::REVERSE(char\* str):STACK(strlen(str))

{

int len = strlen(str);

for (int i = 0; i < len; i++)

{

push(str[i]);

}

}

REVERSE::~REVERSE()

{

}

1. （1）B中A::a1;错误，其不能访问

（2）cout << c.B::b2;错误，c.B::b2为protected

其中A的可访问成员及权限如下：

**private:**

int a1;

**protected:**

int a2;

**public:**

int a3;

~A( ) { };

B的可访问成员及权限如下：

**private:**

int b1;

**protected:**

int a2;

int b2;

int a3;

~A( ){ };

**public:**

A::a2;

int b3;

~B( ) { };

C的可访问成员及权限如下：

**private:**

int c1;

int a2;

int b2;

int a3;

~A( ){ };

A::a2;

int b3;

~B( ) { };

**protected:**

int c2;

B::A::a2;

A::a3;

**public:**

using B::b2;

int c3;

int a3;

~C( ) { };