1. 请说出static和const关键字尽可能多的作用

static关键字：

（1）函数体内static变量的作用范围为该函数体，不同于auto变量，该变量的内存只被分配一次，因此其值在下次调用时仍维持上次的值；
（2）在模块内的static全局变量可以被模块内所用函数访问，但不能被模块外其它函数访问；
（3）在模块内的static函数只可被这一模块内的其它函数调用，这个函数的使用范围被限制在声明它的模块内；
（4）在类中的static成员变量属于整个类所拥有，对类的所有对象只有一份拷贝；
（5）在类中的static成员函数属于整个类所拥有，这个函数不接收this指针，因而只能访问类的static成员变量。

const关键字：

（1）欲阻止一个变量被改变，可以使用const关键字。在定义该const变量时，**通常需要对它进行初始化**，因为以后就没有机会再去改变它了；
（2）对指针来说，可以指定指针本身为const，也可以指定指针所指的数据为const，或二者同时指定为const；
（3）在一个函数声明中，const可以修饰形参，表明它是一个输入参数，在函数内部不能改变其值；
（4）对于类的成员函数，若指定其为const类型，则表明其是一个常函数，不能修改类的成员变量；
（5）对于类的成员函数，有时候必须指定其返回值为const类型，以使得其返回值不为“左值”。例如：

```
const classA operator*(const classA& a1,const classA& a2);
```

operator *的返回结果必须是一个const对象。如果不是，这样的变态代码也不会编译出错：

```
classA a, b, c;
(a * b) = c; // 对a*b的结果赋值
```

操作(a * b) = c显然不符合编程者的初衷，也没有任何意义。

---

2. 构造函数不能声明为虚函数的原因及分析

1. 从**存储空间**角度，虚函数对应一个指向vtable虚函数表的指针，这大家都知道，可是这个指向vtable的指针其实是存储在对象的内存空间的。问题出来了，如果构造函数是虚的，就需要通过 vtable来调用，可是对象还没有实例化，也就是内存空间还没有，怎么找vtable呢？所以构造函数不能是虚函数。

2. 从**使用**角度，虚函数主要用于在信息不全的情况下，能使重载的函数得到对应的调用。构造函数本身就是要初始化实例，那使用虚函数也没有实际意义呀。所以构造函数没有必要是虚函数。虚函数的作用在于通过父类的指针或者引用来调用它的时候能够变成调用子类的那个成员函数。而构造函数是在创建对象时自动调用的，不可能通过父类的指针或者引用去调用，因此也就规定构造函数不能是虚函数。

3. 构造函数不需要是虚函数，也不允许是虚函数，因为**创建一个对象时我们总是要明确指定对象的类型**，尽管我们可能通过基类的指针或引用去访问它但析构却不一定，我们往往通过基类的指针来销毁对象。这时候如果析构函数不是虚函数，就不能正确识别对象类型从而不能正确调用析构函数。

4. 从实现上看，**vbtl在构造函数调用后才建立，因而构造函数不可能成为虚函数**。从实际含义上看，在调用构造函数时还不能确定对象的真实类型（因为子类会调父类的构造函数）；而且构造函数的作用是提供初始化，在对象生命期只执行一次，不是对象的动态行为，也没有必要成为虚函数。

5. 当一个构造函数被调用时，它做的首要的事情之一是初始化它的**VPTR（虚函数表指针）**。因此，它只能知道它是“当前”类的，而完全忽视这个对象后面是否还有继承者。当编译器为这个构造函数产生代码时，它是为这个类的构造函数产生代码——既不是为基类，也不是为它的派生类（因为类不知道谁继承它）。所以它使用的VPTR必须是对于这个类的VTABLE。而且，只要它是最后的构造函数调用，那么在这个对象的生命期内，VPTR将保持被初始化为指向这个VTABLE, 但如果接着还有一个更晚派生的构造函数被调用，这个构造函数又将设置VPTR指向它的VTABLE，等.直到最后的构造函数结束。VPTR的状态是由被最后调用的构造函数确定的。这就是为什么构造函数调用是从基类到更加派生类顺序的另一个理由。但是，当这一系列构造函数调用正发生时，每个构造函数都已经设置VPTR指向它自己的VTABLE。如果函数调用使用虚机制，它将只产生通过它自己的VTABLE的调用，而不是最后的VTABLE（所有构造函数被调用后才会有最后的VTABLE）。

---

3. 拷贝构造函数的写法，为什么要是 const A& , 不可以不用引用？

如果不用引用的话，相当于是按值传递，传值的方式会调用该类的拷贝构造函数，从而调用无穷无尽的拷贝构造函数。

4. 什么时候会调用拷贝构造函数？

- 显式或隐式地用同类型的一个对象来初始化另外一个对象。如上例中，用对象c初始化d；
- 作为实参(argument)传递给一个函数。如CClass(const CClass c_class)中，就会调用CClass的拷贝构造函数；
- 在函数体内返回一个对象时，也会调用返回值类型的拷贝构造函数；
- 初始化序列容器中的元素时。比如 vector svec(5)，string的缺省构造函数和拷贝构造函数都会被调用；
- 用列表的方式初始化数组元素时。string a[] = {string(“hello”), string(“world”)}; 会调用string的拷贝构造函数。

5. vector中迭代器的使用

问：假如说先用迭代器it=vector.begin()，那当push_back()一些元素之后，还能从it遍历vector吗？
答：of courese not。因为一旦push之后，迭代器理论上就无效了。

6. 能不能同时用static和const修饰类的成员函数？

不可以。C++编译器在实现const的成员函数的时候为了确保该函数不能修改类的实例的状态，会在函数中添加一个隐式的参数const this*。但当一个成员为static的时候，该函数是没有this指针的。也就是说此时const的用法和static是冲突的。

我们也可以这样理解：两者的语意是矛盾的。static的作用是表示该函数只作用在类型的静态变量上，与类的实例没有关系；而const的作用是确保函数不能修改类的实例的状态，与类型的静态变量没有关系。因此不能同时用它们。