C++ const关键字

- static关键字的作用
  - 1-函数体内static变量的作用范围为该函数体，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全局变量和函数可以被模块内的函数访问，但不能被模块外其它函数访问；
  - 3-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；
  - 4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关键字的作用
  - 阻止一个变量被改变
  - 声明常量指针和指针常量
  - const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；
  - 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；
  - 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"。
- 修饰变量
  - 综述
    - 说 const 定义的是变量，但又相当于常量；说它定义的是常量，但又有变量的属性，所以叫常变量
    - const 修饰的变量，无论是全局变量还是局部变量，生存周期都是程序运行的整个过程
    - 使用 const 修饰过的局部变量就有了静态特性，它的生存周期也是程序运行的整个过程
    - 经过 const 修饰过的局部变量存储在内存中的"只读数据段"中
    - const 是 C++ 中的关键字，它会在编译期间（时机很重要），告诉编译器这个对象是不能被修改的。
  - 用法
    - const int/char/float/double/string XX=YY;
    - int/char/float/double/string const XX=YY;
      - const 在数值类型前、后都行，表示声明定义的变量是一个常量，不可以修改其值，只读变量；
      - 而且必须在定义的时候就给它赋初值
    - const 类型说明符 变量名
    - 类型说明符 const 变量名
- 修饰指针
  - 常量指针
    - 定义：类型说明符* const 指针名：int *const pInt = &someInt;
    - 综述
      - *pInt 不能改变： pInt是指针，但*pInt是指针指向的内存里面的数值
      - 指针不是常量，但是指针指向的内容是常量
  - 指针常量
    - 定义： const 类型说明符* 指针名：const int* pInt;
    - 综述
      - 是 pInt 不能改变 ： pInt是个指针
      - 指针不可改，指针指向的对象可改
      - 指针是常量，而指针指向的内容可以不是常量
  - 常量指针常量
    - 定义： const 数据类型 * const 指针变量=变量名；
数据类型 const *const 指针变量=变量名；
    - 指针不能改变，指针指向的值也不能改变
- 修饰函数
  - 1--修饰函数的参数
    - 综述
      - const修饰参数是为了防止函数体内可能会修改参数原始对象 （分三种情况）
        - 函数参数为值传递：值传递（pass-by-value）是传递一份参数的拷贝给函数，因此不论函数体代码如何运行，也只会修改拷贝而无法修改原始对象，这种情况不需要将参数声明为const。
        - 函数参数为指针：指针传递（pass-by-pointer）只会进行浅拷贝，拷贝一份指针给函数，而不会拷贝一份原始对象。因此，给指针参数加上顶层const可以防止指针指向被篡改，加上底层const可以防止指向对象被篡改。
          - void Fun(const A *in);
        - 函数参数为引用：引用传递（pass-by-reference）有一个很重要的作用，由于引用就是对象的一个别名，因此不需要拷贝对象，减小了开销。这同时也导致可以通过修改引用直接修改原始对象（毕竟引用和原始对象其实是同一个东西），因此，大多数时候，推荐函数参数设置为pass-by-reference-to-const。给引用加上底层const，既可以减小拷贝开销，又可以防止修改底层所引用的对象。
          - void Fun(const A &in);
      - 子主题 2
      - 子主题 3
      - 子主题 4
- 修饰引用
  - 定义：const 类型说明符 &引用名
  - 综述：
    - 非const引用只能绑定非const对象，const引用可以绑定任意对象，并且都当做常对象
    - 常引用经常用作形参，防止函数内对象被意外修改
    - 对于在函数中不会修改其值的参数，最好都声明为常引用
- 修饰类
  - 修饰类的成员函数
    - 综述
      - 由于C++会保护const对象不被更新，为了防止类的对象出现意外更新，禁止const对象调用类的非常成员函数。因此，常成员函数为常对象的唯一对外接口。
      - const对象只能访问const成员函数，而非const对象可以访问任意的成员函数，包括const成员函数
      - const对象的成员是不能修改的，而通过指针维护的对象确实可以修改的；
      - const成员函数不可以修改对象的数据，不管对象是否具有const性质。编译时以是否修改成员数据为依据进行检查。
    - 使用方法： 类型说明符 函数名(参数表) const
    - 要点：
      - 常成员函数的定义和声明都要含有const关键字；
      - 一个函数是否含有const关键字可以作为重载函数，const对象默认调用const函数，非const对象默认调用非const函数，如果没有非const函数，也可以调用const函数；
      - const函数中不能更新目的对象的任何成员(mutable修饰的变量除外，这里不展开阐述)，以此方法来保证const对象不被修改。
      - 如果const成员函数想修改成员变量值，可以用mutable修饰目标成员变量。
  - 修饰类的成员数据
    - 综述
      - 类的数据成员不能在任何函数中被赋值或修改，但必须在构造函数中使用初始化列表的方式赋初值。
      - 子主题 2
  - 修饰类对象
    - 综述
      - 用const修饰的类对象，该对象内的任何成员变量都不能被修改。
      - 不能调用该对象的任何非const成员函数，因为对非const成员函数的调用会有修改成员变量的企图
    - 用法： const A* b = new A();
- 修饰数组
- 辅助信息---指针
  - cosnt修饰指针，涉及到两个很重要的概念，顶层const和底层cosnt
  - 指针自身是一个对象，它的值为一个整数，表明指向对象的内存地址
  - 指针长度所指向对象类型无关，在32位系统下为4字节，64位系统下为8字节。
  - 进而，指针本身是否是常量以及所指向的对象是否是常量就是两个独立的问题。（重点---重点）指针本身是不是常量和指针所指向的对象是不是常量就是两个互相独立的问题
    - 用顶层表示指针本身是个常量
    - 顶层 const 可以表示任意的对象是常量，这一点对任何数据类型都适用；
    - 底层表示指针所指向的对象是个常量
    - 底层 const 则与指针和引用等复合类型有关