1.static的作用

1.1 类外

1. static修饰局部变量。静态局部变量在函数内定义，生存期为整个源程序，只能在定义该变量的函数内使用。退出该函数后，尽管该变量还继续存在，但不能使用它。对基本类型的静态局部变量若在说明时未赋初值，则系统自动初始化为0。

2. 修饰全局变量。全局变量本身就是静态存储方式，静态全局变量当然也是静态存储方式。但是他们的作用域不同，非静态全局变量的作用域是整个源程序（多个源文件可以共同使用）；而静态全局变量限制了其作用域，只在定义该变量的源文件内有效（隐藏的功能），在同一源程序的其他源文件中不能使用它

3. 修饰函数。只能被本文件中的函数调用，而不能被同一程序其他文件中的函数调用。区别于一般的非静态函数。

1.2 类内

可以使用静态成员变量在同类的多个对象之间实现数据共享。

1. 一个类中可以有一个或多个静态成员变量，**所有的对象都共享这些静态成员变量**，都可以引用它。

2. static成员变量和普通static变量一样，编译时在静态数据区分配内存，到程序结束时才释放。这就意味着，**static成员变量不随对象的创建而分配内存，也不随对象的销毁而释放内存。**而普通成员变量在对象创建时分配内存，在对象销毁时释放内存。

3. 静态成员变量必须初始化，而且只能在类外进行。初始化时可以赋值，也可以不赋值。如果不赋值，那么会被默认初始化，一般是0.静态数据区的变量都有默认的初始值，而动态数据区（堆区、栈区）的变量默认是垃圾值。static成员变量的内存空间既不是在声明类时分配，也不是在创建对象时分配，而是在初始化时分配。

4. static成员函数。普通成员函数可以访问所有成员变量，而静态成员函数只能访问静态成员变量。静态成员函数没有this指针，既然她没有指向某一对象，就无法对该对象中的非静态成员进行访问。如果要在类外调用public属性的静态成员函数，可以用类名和域解析符：：，也可以通过对象名调用静态成员函数。

2.const

2.1 修饰常量

1. 定义const常量，如const int Max=100

2. 替换#define功能，便于进行类型检查。const常量有数据类型，而宏常量没有数据类型。编译器可以对前者进行类型安全检查，而对后者只进行字符替换，没有类型安全检查，并且在字符替换是可能会产生意料不到的错误；使用const可以比#define产生更小的目标代码

3. 提高了效率，编译器通常不为普通const常量分配存储空间，而是将他们保存在符号表中这使得它成为一个编译期间的常量，没有存储与读内存的操作，使得效率很高。

2.2 修饰指针

指向const对象的指针（const double* ptr)，不允许用指针来改变所指向的const值

const指针（double* const ptr)，指针本身不可以改变，但是指向的值可以改变

2.3 修饰函数参数与返回值

修饰返回值，函数返回值（指针或引用）的内容不能被修改，该返回值只能被赋给加const修饰的同类型指针（如果返回值不是指针，则由于函数会把返回值复制到外部临时的存储单元中，加const修饰没有任何价值）

修饰函数参数，可以保护被修饰的东西，防止意外的修改，增强程序的健壮性

2.4 类中的应用

const成员函数，则该成员函数不能修改类中任何非const成员，一般写在函数的最后来修饰。const成员函数不能修改它所在对象的任何一个数据成员，能访问对象的const成员，而其他成员函数不可以。

const修饰类的成员变量，表示成员常量，不能被修改，同时只能在初始化列表中赋值。

3.volatile

volatile关键字是一种类型修饰符，用它声明的类型变量表示可以被某些编译器未知的因素更改，比如操作系统、硬件或者其他线程等。遇到这个关键字声明的变量，编译器对访问该变量的代码就不再进行优化，从而可以提供对特殊地址的稳定访问。声明时语法：int volatile vInt；当要求使用volatile声明的变量的值得时候，系统总是重新从它所在的内存读取数据，即使它前面的指令刚刚从该处读取过数据。而且读取的数据立刻被保存。

一般说来，volatile用在如下的几个地方：

- 中断服务程序中修改的供其他程序检测的变量需要加volatile
- 多任务环境下各任务间共享的标志应该加volatile
- 存储器映射的硬件寄存器通常也要加volatile说明，因为每次对他的读写都可能有不同意义

4.extern

1. 当它与"C"一起连用时，如 extern "C" void fun(int a,int b)，则告诉编译器在编译这个fun函数名时按着C的规则去翻译相应的函数名而不是C++的。C++语言在编译的时候为了解决函数的多态问题，会将函数名和参数联合起来生成一个中间的函数名称，而C语言则不会，因此会造成链接时找不到对应函数的情况，此时C函数就需要用extern "C"进行链接指定，这告诉编译器，请保持我的名称，不要给我生成用于链接的中间函数名。
2. 当extern不与"C"在一起修饰变量或函数时，如在头文件中：extern int g_Int，它的作用就是声明函数或全局变量的作用范围的关键字，其声明的函数和变量可以在本模块或其他模块中使用，记住他是一个声明而不是定义！以表示变量或者函数的定义在别的文件中，提示编译器遇到此变量和函数时在其他模块中寻找定义。

extern与static：

1. extern表明该变量在别的地方已经定义过了，在这里要使用那个变量
2. static表明静态的变量，分配内存的时候，存储在静态区，不存储在栈上面

extern与const：

1. c++中const修饰的全局常量具有跟Static相同的特性，即他们只能作用于本编译模块中，但是const可以与extern连用来声明该常量可以作用于其他编译模块中。如extern const char g_str[];