内联函数：

原理：用被调函数的函数体代码替换掉函数调用语句，该替换是在编译期间进行

用空间换取时间上的效率，增大了空间上的内存消耗，所以内联函数要尽量结构简单，代码短小

重载：函数名相同，参数个数和参数类型不同，与返回值类型无关

构造函数：在类外创建对象时，构造函数必须公有

函数名与类名一致

参数可有可无

无返回值

作用：创建对象时，初始化对象成员

类型：不带参数的默认构造函数、参数被赋值的构造函数

初始化列表形式：常规初始化、初始化列表

A(int a){\_a=a;} A(int a):\_a(a){}

使用范围：构造函数的函数头，需要对类的成员变量初始化、拷贝构造函数

当类中有常量或引用类型的成员时，需要对成员变量初始化

当类中的一个对象成为另一个类的成员，若在该类中没有默认构造函数，则在另一个类中需要对该成员变量初始化

在继承中，如果父类没有默认构造函数，子类在定义时需要对父类部分的成员变量初始化

成员初始化的顺序：按照类中成员变量声明的顺序初始化

注意避免用类中的一个成员变量去初始化另一个成员变量

malloc/free和new/delet都可用于动态的内存申请和释放

malloc/free是库函数，而new/delete是运算符

new用于构造函数

delete用于析构函数

浅拷贝：即逻辑拷贝，将一个对象一对一地赋值给另一个对象

如果对象中有指针成员变量，且指向堆上的内存空间，在拷贝时，将只拷贝指针成员变量本身的值（即地址），导致两个对象的指针指向同一块堆空间，在删除时，会造成二次删除和内存泄漏

（1）、const int & fun (int &a);

函数返回值是const引用，函数的返回值将不能被修改

（2）、int fun(const int & a);

函数的参数类型是const引用，函数体内的参数的值将不能被修改

（3）、int fun() const;

常成员函数，只限于类的定义，成员函数将不能修改成员变量的值

继承：允许在不改动原程序的基础上，对其进行扩充，这样使得原程序的功能得以保存，而新功能得以扩展，减少了重复编码，大大提高了程序的运行效率。且便于维护，即只需要修改父类中的代码内容，子类中的内容也会随之一起改变

重载：函数名相同，参数个数和参数类型不同，重写：函数名和参数必须相同

重写必须继承，重载不用

析构函数：公有、唯一、无參、无返回值

作用：释放申请的内存空间

父类中声明为析构函数可以避免潜在内存泄漏。父类的指针或引用指向子类对象，当删除一个指向子类对象的父类指针时，将调用父类的析构函数，子类只析构了来自父类部分的成员变量，却没有析构子类自己扩展的成员变量，造成内存泄漏

多态：同一个操作作用于不同的对象，会产生不同的解释，造成不同的结果

条件：必须在继承关系中，子类必须覆盖父类的虚函数，父类的指针或引用必须指向子类，必须通过父类的指针或引用调用虚函数，会发生动态绑定

class 成员变量默认私有，继承访问权限默认私有

struct 成员变量默认公有，继承访问权限默认公有

构造函数不能是虚函数。构造函数是为了初始化对象成员，而虚函数里面有一个虚表，在对象初始化完成之后，将产生一个虚指针，虚指针指向虚表，通过虚表来调用虚函数。如果构造函数是虚函数，就必须通过虚表来找到虚构造函数的入口地址，而构造函数还没有初始化，不可能有虚表

类模版不是一个具体的、实际的类，而是代表一类类，将类模版实例化成一个具体的类，即模版类

多重继承的内存分配问题：  
比如有classA ：public class B , public classC , {} 那么A的内存结构大致是怎么样的。

A的内存大概分为三个部分，第一部分为继承基类B的成员，第二部分为继承基类C的成员，第三部分为自己类中定义的成员

多继承中，若子类没有覆盖父类，则先存放父类1的虚函数指针，再存放子类的虚函数指针，其余父类只存储自己的指针

若子类覆盖父类，则虚函数表的每部分的开始元素先存放子类覆盖父类的虚函数指针，再存放父类1中剩下的，然后是子类自己扩展的，其余父类在其后只存储自己剩下的虚函数指针

容器：存储相同类型数据的集合 用类模版来实现

vector 向量 连续存储，在尾部增删元素，随机访问任一元素

map 映射 一对一的键值对，用一个key查找对应的value，且key不能重复

list 双向链表 不连续存储，在任意位置增删元素，顺序访问