1. 设计类、求圆的周长
   1. 关键字 class
   2. class + 类名 { 成员 }
   3. 类的构成 ： 成员函数 + 成员属性
   4. 作用域 public ： 公共作用域
   5. 属性 ： 半径
   6. 行为 ： 求圆周长、设置圆半径 、获取圆半径
   7. 测试 ： 实例化对象 通过类创建对象 Circle c1;
   8. 给对象赋值 c1.setR(10) 获取半径 c1.getR(); 获取周长 c1.calculateZC();
   9. 设计一个学生类，有姓名和学号，给姓名学号赋值，并且打印学生的信息
2. 内联函数
   1. 内联函数的引出 --- 宏函数缺陷
      1. 必须保证运算完整性，加括号
      2. 即使加了括号，有些情况依然和预期结果不符
      3. 宏函数 也不重视作用域
   2. 内联函数 inline
   3. 函数声明和实现必须同时加inline 才算内联函数
   4. 类内部成员函数 默认前面加了inline关键字
   5. 内联函数和编译器关系
      1. 加入内联后只是对编译器的一个建议，有些情况下编译器依然不会按照内联方式处理函数
      2. 循环、 判断、 函数体过于庞大、对函数取地址 ，不会按照内联处理
      3. 一个好的编译器，会将短小的函数 隐藏的加入inline 关键字
3. 函数的默认参数以及占位参数
   1. 默认参数
      1. 在形参后面 = 默认值
      2. 如果参数列表中 有一位置有了默认参数，那么从这个位置起从左到右都必须有默认参数
      3. 函数的声明和实现 只能有一个有默认参数
   2. 占位参数
      1. 函数参数列表中 ( int )只写类型，调用必须要传入参数调用
      2. 占位参数也可以有默认参数 （int = 0）
      3. 目前用不到占位参数，后面学习符号重载才会用到
4. 函数重载
   1. 满足条件
      1. 同一个作用域，函数名称相同
      2. 函数参数 类型 、 个数、 顺序不同
   2. 函数的返回值 不可以作为函数重载条件
   3. 引用 加入const 和不加const也可以作为重载条件
   4. 函数重载碰到函数默认参数 也需要注意避免二义性
5. extern C 浅析
   1. 函数重载原理 编译器在底层会将函数名字做两次修饰，方便内部访问函数体
   2. 用途： 在C++下可以运行C语言的文件
   3. #ifdef \_\_cplusplus 两个下划线 + c plus plus
      1. extern "C" {
   4. #endif
   5. #ifdef \_\_cplusplus
      1. }
   6. #endif
6. 类的封装
   1. C语言的封装
      1. 缺点：C语言下没有做类型转换的检测
      2. 缺点2： 将属性和行为分离
   2. C++封装
      1. 将属性和行为作为一个整体，来表现生活中的事物
      2. 将这些成员 加以权限控制
   3. 权限
      1. public 公有权限
      2. protected 保护权限
      3. private 私有权限
   4. C++中struct 和 class 区别
      1. struct 默认权限 public
      2. class 默认权限 private
7. 尽量将成员属性设置为私有
   1. 设置为私有权限的属性，可以自己控制属性的读写操作
   2. 可以将传入数据，进行有效性的检测