1. 立方体设计案例
   1. 属性
      1. 长 、 宽 、高
   2. 行为
      1. 设置 获取 长、宽 、高
      2. 求立方体 面积、体积
      3. 判断两个立方体是否相等
   3. 全局函数 也做了判断立方体是否相等
2. 点和圆关系案例
   1. 点类
      1. 属性 x坐标 y坐标
      2. 行为 设置 获取 x、y坐标
   2. 圆类
      1. 属性 半径、圆心 （Point 点类）
      2. 行为 设置 获取 半径 、 圆心
   3. 分别利用全局函数 和 成员函数 判断点和圆的关系
   4. 分文件编写
3. 构造函数和析构函数
   1. 构造函数语法
      1. 不写返回值 ，不用void，名称与类型相同
      2. 可以有参数，可以发生重载
      3. 由编译器自动调用，只调用一次就可以
   2. 析构函数语法
      1. 不写返回值，不写void，名称与类名相同，名称前 + ~
      2. 不可以有参数， 不可以发生重载
      3. 由编译器自动调用，只调用一次就可以
4. 构造函数的分类以及调用
   1. 分类
      1. 按照参数分类 ： 有参构造 无参构造（默认构造）
      2. 按照类型分类 ： 普通构造 拷贝构造 （ 类名（ const 类名 & aaa ））
   2. 调用
      1. 默认构造调用 Person p ; 不要写成 Person p() ，编译器认为是一个函数声明
      2. 括号法 Person p1(10) Person p2(p1)
      3. 显示法 Person p1 = Person(10) Person p2 = Person(p1)
         1. Person(10) 匿名函数对象 特点：当前行执行完后系统就回收
         2. Person(p1) 不可以用拷贝构造函数来初始化 匿名函数对象，编译器认为是对象声明
      4. 隐式法
         1. Person p1 = 10; 相当于 编译器隐式将代码转为 Person p1 = Person(10)
5. 拷贝构造函数的调用时机
   1. 使用一个已经创建好的对象来初始化另一个对象
   2. 以值传递的方式 给函数的参数传值
      1. void doWork( Person p)
   3. 以值的方式返回局部对象
      1. Person doWork2()
6. 构造函数的调用规则
   1. 系统会默认给一个类添加至少3个函数：默认构造函数、拷贝构造函数、析构函数
   2. 如果提供有参构造、系统不会提供默认构造，但是会提供拷贝构造
   3. 如果提供拷贝构造，其他构造函数都不提供了
7. 深拷贝与浅拷贝
   1. 系统提供拷贝构造函数 只会做简单的值拷贝
   2. 如果类中有属性开辟到堆区，那么在释放时候 ，由于浅拷贝问题导致堆区内容会重复释放，程序down掉
   3. 利用深拷贝 来解决浅拷贝的问题
8. 初始化列表
   1. 构造函数名称 ( ) : 属性 (值) , 属性(值) ….
9. 类对象作为类成员
   1. 当其他类对象作为本类成员，先构造其他类对象，再构造自身，释放的顺序与构造是相反
10. explicit关键字
    1. explicit 关键字用途 ：防止隐式类型转换方式来初始化对象
11. new 和 delete 操作符
    1. // new delete 是一个运算符 malloc free 是一个库函数
    2. // malloc 返回值 void \* new 返回的是new出来的对象的指针
    3. // malloc 需要判断是否开辟成功 而new内部做好了操作(内部会malloc数据在堆区，判断内存是否分配成，调用构造函数)
    4. // malloc 不会调用构造函数 而 new调用构造函数
    5. // malloc 对应的释放是 free new 对应的释放是 delete
    6. 不要用void \* 去接受new出来的对象，原因是不能够释放
    7. 利用new 在堆区创建数组，类中必须要存在默认构造函数，否则无法创建
    8. 如果是数组 释放时候要在 delete后加 []
       1. delete **[]** persons;