1. 设计立方体类案例
   1. 设计class Cube
   2. 属性
      1. 长宽高
   3. 行为
      1. 设置长宽高
      2. 获取长宽高
      3. 获取面积
      4. 获取体积
   4. 通过全局函数和成员函数 判断两个立方体是否相等
2. 点和圆关系案例
   1. 设计点和圆类
   2. 点类 Point
      1. 属性 x y
      2. 行为 设置 获取 x y
   3. 圆类 Circle
      1. 属性： 圆心 Point m\_Center 半径 m\_R;
      2. 行为： 设置 获取 半径 圆心
   4. 通过成员函数 和 全局函数 判断点和圆关系
   5. 分文件编写 点和圆类
3. 构造函数和析构函数
   1. 构造函数
      1. //没有返回值 不用写void
      2. //函数名 与 类名相同
      3. //可以有参数 ，可以发生重载
      4. //构造函数 由编译器自动调用一次 无须手动调用
   2. 析构函数
      1. //没有返回值 不用写void
      2. 函数名 与类名相同 函数名前 加 ~
      3. 不可以有参数 ，不可以发生重载
      4. 析构函数 也是由编译器自动调用一次，无须手动调用
4. 构造函数的分类和调用
   1. 分类
      1. 按照参数分类： 有参 无参（默认）
      2. 按照类型分类： 普通 拷贝构造 ( const Person & p )
   2. 调用
      1. 括号法
      2. 显示法
      3. 隐式法
   3. 注意事项
      1. 不要用括号法 调用无参构造函数 Person p3(); 编译器认为代码是函数的声明
      2. 不要用拷贝构造函数 初始化 匿名对象 Person(p3); 编译器认为 Person p3对象实例化 如果已经有p3 p3就重定义
   4. 匿名对象 特点： 当前行执行完后 立即释放
5. 拷贝构造函数的调用时机
   1. 用已经创建好的对象来初始化新的对象
   2. 值传递的方式 给函数参数传值
   3. 以值方式 返回局部对象
6. 构造函数的调用规则
   1. //1、编译器会给一个类 至少添加3个函数 默认构造（空实现） 析构函数（空实现） 拷贝构造（值拷贝）
   2. //2、如果我们自己提供了 有参构造函数，编译器就不会提供默认构造函数，但是依然会提供拷贝构造函数
   3. //3、如果我们自己提供了 拷贝构造函数，编译器就不会提供其他构造函数
7. 深拷贝与浅拷贝的问题以及解决
   1. 如果有属性开辟到堆区，利用编译器提供拷贝构造函数会调用浅拷贝带来的析构重复释放堆区内存的问题
   2. 利用深拷贝解决浅拷贝问题
   3. 自己提供拷贝构造函数，实现深拷贝
8. 初始化列表
   1. 可以利用初始化列表语法 对类中属性进行初始化
   2. 语法：构造函数名称后 ： 属性(值), 属性（值）...
      1. Person(int a, int b, int c) : m\_A(a), m\_B(b), m\_C(c)
9. 类对象作为类中成员
   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，析构的顺序和构造相反
10. explicit关键字
    1. explicit用途： 防止利用隐式类型转换方式来构造对象
11. new和delete
    1. malloc 和 new 区别
       1. malloc 和 free 属于 库函数 new 和delete属于 运算符
       2. malloc不会调用构造函数 new会调用构造函数
       3. malloc返回void\* C++下要强转 new 返回创建的对象的指针
    2. 注意事项 不要用void\*去接受new出来的对象,利用void\*无法调用析构函数
    3. 利用new创建数组
       1. Person \* pPerson = new Person[10];
       2. 释放数组时候 需要加[]
       3. delete [] pPerson;
    4. 堆区开辟数组，一定会调用默认构造函数
    5. 栈上开辟数组，可不可以没有默认构造,可以没有默认构造