1. 链表反转作业
   1. 利用3个辅助指针，实现链表反转
   2. 提供返回链表节点个数 函数
2. 回调函数作业-实现对任意数据类型数组进行排序
   1. 利用选择排序实现大框架
   2. 对比功能交还给用户指定
   3. 利用回调函数技术来实现对比
3. 预处理指令
   1. 头文件 #include
      1. <> “”区别
      2. <> 包含系统头
      3. “” 包含自定义头
   2. 宏
      1. 宏常量
         1. 不重视作用域
         2. 没有数据类型
         3. 利用 #undef 卸载宏
      2. 宏函数
         1. 将频繁、短小函数写成宏函数
         2. 优点：以空间换时间
   3. 条件编译
      1. #ifdef #else #endif 测试存在
      2. #ifndef #else #endif 测试不存在
      3. #if #else #endif 自定义条件编译
   4. 特殊宏
      1. \_\_FILE\_\_ 宏所在文件路径
      2. \_\_LINE\_\_ 宏所在行
      3. \_\_DATE\_\_ 宏编译日期
      4. \_\_TIME\_\_ 宏编译时间
4. 静态库配置
   1. 右键项目->属性 ->常规->配置类型 ->静态库
   2. 生成项目 生成.lib文件
   3. 将.lib和 .h交给用户
   4. 测试
5. 动态库配置
   1. 右键项目->属性 ->常规->配置类型 ->动态库 .dll
   2. 生成项目 生成 .lib .dll
   3. 静态库中生成的.lib和动态库生成的.lib是不同的，动态库中的.lib只会放变量的声明和 导出函数的声明，函数实现体放在.dll中
   4. 导出函数/外部函数 ： \_\_declspec(dllexport)int mySub(int a, int b);
   5. 测试
      1. #pragma comment( lib,"./mydll.lib")
6. 递归函数
   1. 本质：函数自身调用自身
   2. 注意事项：递归函数必须有结束条件，函数有出口
7. 面向接口封装案例
   1. 甲乙两方设计接口
   2. 甲方实现代码
   3. 乙方实现代码
   4. 接口对接