1. 链表作业
   1. 反转链表 通过3个辅助指针变量实现链表的翻转
   2. 统计链表长度 int size\_LinkList( struct LinkNode \* pHeader )
2. 回调函数案例 作业
   1. 对任意数据类型的数组进行排序
3. 预处理指令
   1. 头文件包含 #include
      1. 注意 “” <> 区别
      2. <> 系统头文件 “” 自定义头文件
   2. 宏定义
      1. 不重视作用域
      2. 可以利用 #undef 卸载宏
      3. 宏常量 没有数据类型
      4. 宏函数 要注意表达式完整性
   3. 条件编译
      1. 测试存在 #ifdef
      2. 测试不存在 #ifndef
      3. 自定义条件测试 #if
   4. 特殊宏
      1. 编译所在文件 \_\_FILE\_\_
      2. 编译所在行号 \_\_LINE\_\_
      3. 编译日期 \_\_DATE\_\_
      4. 编译时间 \_\_TIME\_\_
4. 静态库配置
   1. 创建项目 --- 配置属性 --- 常规 ----- 配置类型 --- 静态库
   2. 重新生成项目 ，创建出后缀名为 .lib的静态库文件
   3. 测试静态库
5. 动态库配置流程
   1. 静态库优缺点
      1. 优点 ： 生成的exe程序中包含了 静态库中的内容，与静态库无瓜葛
      2. 缺点 ： 浪费资源 ， 更新发布比较麻烦
   2. 动态库
      1. 运行阶段才去链接函数
      2. 配置流程： 创建项目 --- 配置属性 -- 常规 --- 配置类型 --- 动态库
      3. 重新生成解决方案，生成 .dll .lib 库文件
      4. 导入函数 只能在当前项目下使用
      5. 导出函数 可以在外部使用
      6. \_\_declspec (dllexport) int mySub(int a, int b);
      7. 测试 引入#pragma comment(lib,"./mydll.lib")
6. 递归函数
   1. 函数自身调用自身，必须有结束条件退出循环
   2. 案例
      1. 实现字符串逆序遍历
      2. 实现斐波那契数列
7. 面向接口编程
   1. 实现公司中编程方式
   2. 甲方和乙方商定好接口，分别实现自己的功能
   3. 最后对接成功后，实现游戏功能