

字符串顺序表处理程序报告

一、整体功能概述

程序实现了一个基于“顺序表”（动态数组）存储的**分词结果管理与字符串操作系统**。它接收用户输入（控制台或文件读取）的一段文本，按照空白符、英文标点及常见中文标点对文本做**分词**，将每个词（Token）以 string 形式顺序存储在自建的顺序表 SqList 中。随后提供一组典型操作：

功能	描述
分词 String_cut	依据空格/标点拆分出词（支持部分中文标点，多字节分隔符长度感知）
打印 Print_String	可选择紧凑打印或词间加空格
插入 Word_Insert	在第 i 个位置前插入 1..n 个新词（支持一次性输入多词并再分词）
删除 Word_Delete	删除从位置 i 开始的连续 num 个词
反转 String_Reverse	原地首尾交换实现顺序表倒置
回文判断 Judge_Palindrome	比较对称位置词是否一致（按“词”为单位而非字符）
计数 String_Length	返回当前词数量
查找子串 Find_Word	支持查找单一词或一段词序列（相邻匹配）
词频统计 Count_Word	统计所有不同词出现次数
重新加载 Load_SqList	重新读取输入文本并替换当前顺序表
销毁 Destroy_List	手动释放动态数组内存，结束程序

二、顺序表设计

```
typedef struct {  
    Elemtype *elem; // 动态数组首地址  
    int length;      // 当前已存储的词数  
    int listsize;    // 当前分配的容量（可容纳的最大词数）  
} SqList;
```

设计特点：

1. 使用原始指针 + new[] 管理字符串数组。
2. 初始容量固定 LIST_SIZE=100，需要时按固定步长扩容（Expand_Capacity）。
3. 逻辑上是一个“词”级别的容器，每个 Token 是一个 string。

三、个人收获与思考

1. 我提高了对基本数据结构进行操作的熟练度，尤其是顺序表的增删改查和动态内存管理。
2. 实现各类功能与编写用户界面难度一致，让我明白了前后端开发同等重要。
3. 编码规范一致性也很重要。命名风格、常量集中、重复代码抽象，这些都直接影响后续维护成本。