**专题一：结构化程序设计与递归函数**

问题1： 六种排序算法

问题2： 共享子串问题

组号：21 教师：张引、田沈晶、熊海辉

成员：

学号 3170105860 姓名 彭子帆

学号 3160103848 姓名 陈稼诚

**报告提交时间：2018.6.17**

# 1.问题1名称（完成的其他问题也按此结构模板撰写）

## 1.1 问题描述

* 提供插入、归并、桶排序、基数排序、计数排序、快速排序的实现，可以让人选择使用的排序方式，从一个数据文件中读入N个整数（N<=100000），且把结果保存到另一个文本文件中，并显示排序所用的时间。

## 1.2 问题解决思路和关键点

### 1.2.1模块化

将几个不同功能单独编写，将插入、归并、桶排序、基数排序、计数排序、快速排序的函数分别编写，然后根据不同需要调用不用函数。

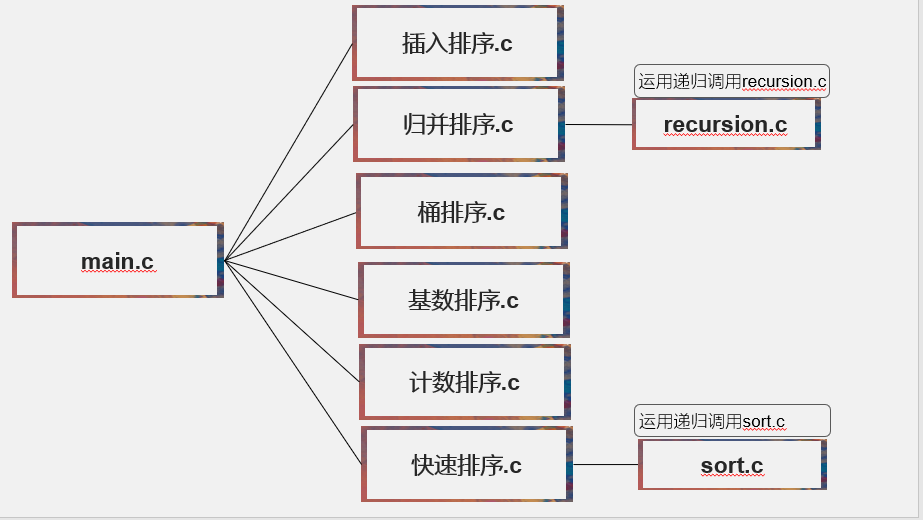
### 1.2.2 排序算法

通过查阅资料以及理解，逐步写出各个算法

## 1.3 程序结构

### 1.3.1 函数调用关系图

画出main函数及其下属各层调用的函数



### 1.3.2 函数说明

1）函数原型：void insert\_sort(int n)

功能描述：插入排序

参数描述： int n，排序数的个数

返回值描述：void

重要局部变量定义：i,j,temp

重要局部变量用途描述：计数以及中间变量

函数算法描述：PPT中有，遍历所有数插入到已排好数列

2）函数原型：void merge\_sort(int n)

功能描述；归并排序

参数描述： int n，排序数的个数

返回值描述：void

重要局部变量定义：i,j,temp

重要局部变量用途描述：计数以及中间变量

函数算法描述：分治到最后，递归到一个数为止，而后将两个已排好的数列合并

3）函数原型: void bucket\_sort(int n)

功能描述；桶排序

参数描述： int n，排序数的个数

返回值描述：void

重要局部变量定义：i,j,temp

重要局部变量用途描述：计数以及中间变量

函数算法描述：分成多个桶，桶内用其他方式排序，而后合并

4）函数原型：void radix\_sort(int n)

功能描述：基数排序

参数描述： int n，排序数的个数

返回值描述：void

重要局部变量定义：i,j,temp

重要局部变量用途描述：计数以及中间变量

函数算法描述：从个位到万位进行排序合并排序

5）函数原型：void counting\_sort(int n)

功能描述；计数排序

参数描述： int n，排序数的个数

返回值描述：void

重要局部变量定义：i,j,temp

重要局部变量用途描述：计数以及中间变量

函数算法描述：对于偏序集，可以计数小于某个数的个数，并把它放置到该个数的位置

6）函数原型：void quick\_sort(int n)

功能描述；快速排序

参数描述： int n，排序数的个数

返回值描述：void

重要局部变量定义：i,j,temp

重要局部变量用途描述：计数以及中间变量

函数算法描述：冒泡改良，进行交换

7）函数原型：void recursion (int \*a,int startindex,int endindex)、void sort (int \*a, int left, int right)

功能描述；归并排序和快速排序中的递归函数

参数描述： 数组，开始索引下标，结尾索引下标

返回值描述：void

重要局部变量定义：i,j,temp

重要局部变量用途描述：计数以及临时交换变量

函数算法描述：分成小串数列进行递归，分治，大问题化成小问题

### 1.3.3 程序文件结构

1）文件函数结构

插入排序.c，包含insert\_sort(int n)

归并排序.c 包含 merge(int n), recursion(int \*a, int startindex, int endindex)

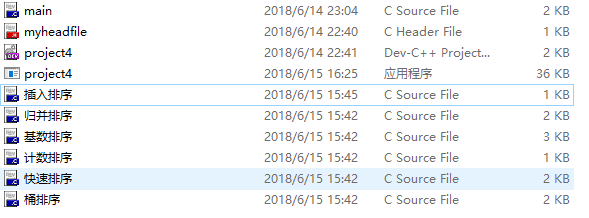
桶排序.c 包含 bucket\_sort(int n)

基数排序.c 包含 radix\_sort(int n)

计数排序.c 包含 counting\_sort(int n)

快速排序.c 包含quick\_sort(int n),sort(int \*a, int left, int right)

2）多文件构成机制



## 1.4 安装运行说明

直接运行exe文件，直接编译运行，无需安装

## 1.5 总结

### 1.5.1 实践过程中遇到的难点及解决方案

某些排序思想与算法较为复杂，通过一遍遍调试，在函数中加入printf()来判断哪些步骤出了问题

### 1.5.2 程序亮点或基本功能之外的拓展点

### 1.5.3 实践之不足

还行，实践挺足的吧

### 1.5.4 分工合作和各自的收获

我们写了不同的函数，收获了很多，对于链表的操作更加明了了。

# 2.共享子串

## 2.1 问题描述

## 英语单词可以用链表的形式存储。为了节省空间，可以将两个以相同子串结尾的英语单词，共享同一段链表空间。比如“loading”和“being”。

## 2.2 问题解决思路和关键点

### 2.2.1如何做到两个链表串联在一起？

先找出两个单词公共的部分。分别用三个链表存取两个单词不同的部分以及共享的部分。最后再拼接在一起。

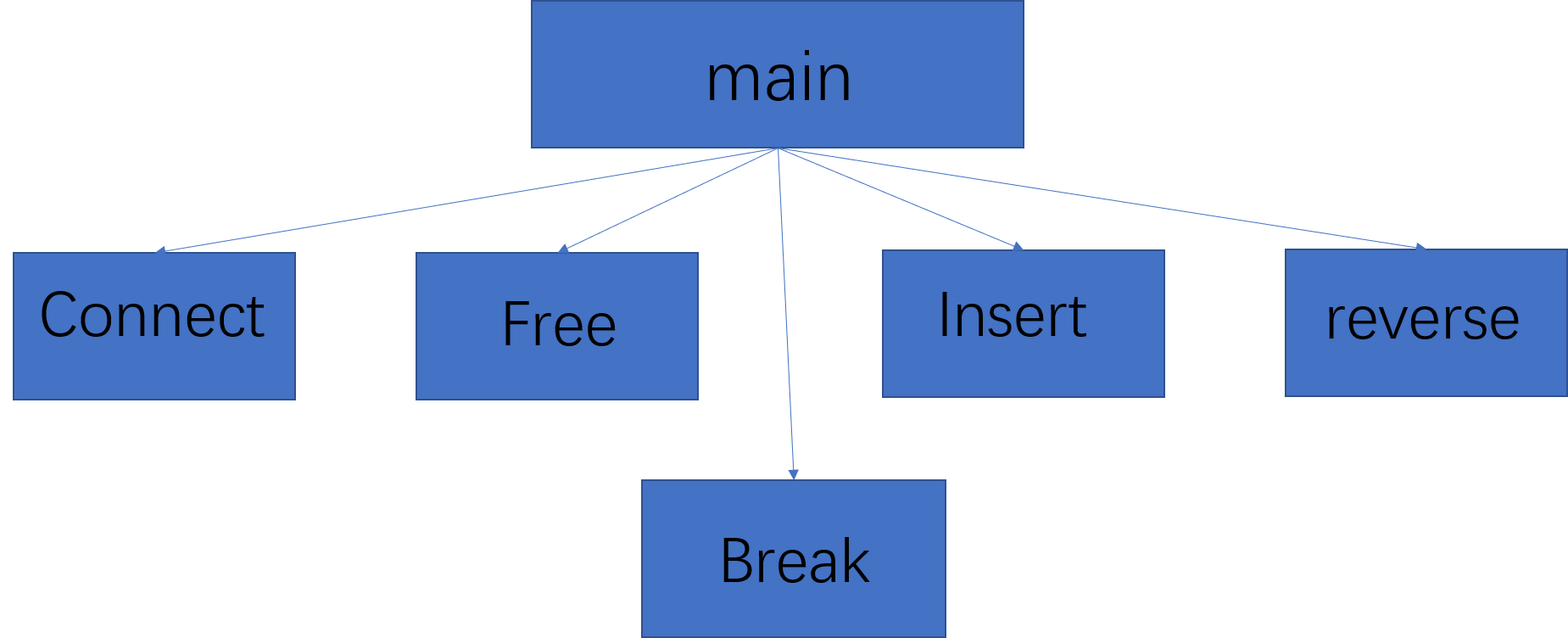
### 2.2.2如何找出两个单词的共享部分

将输入的两个单词的字符串倒置。一个一个对比字符是否相同。

## 2.3 程序结构

### 2.3.1 函数调用关系图

画出main函数及其下属各层调用的函数



### 2.3.2结构类型

在程序中，节点类型被定义为

typedef struct character{

char ch;

int tag;

struct character \*next;

}List;

### 2.3.3 函数说明

1）函数原型：List \*Insert(char c, int t, int i, List \*Ptrl);

功能描述：在链表中插入一个元素。

参数描述： 字符c，整数int，整型t，链表头Ptrl

返回值描述：本函数返回一个新的链表头节点。

重要局部变量定义：指针p；指针s；

重要局部变量用途描述：用于寻找第i-1个节点以方便插入节点的操作。

函数算法描述：首先判断插入位置是否为头结点。如果是即插入头结点。在本程序中只进行了插入头节点的操作。

2）函数原型： void reverse(char c[])

功能描述： 翻转字符串

参数描述： 字符数组c【】

返回值描述：本函数无返回值

重要局部变量定义：整型length，整型i

重要局部变量用途描述：length用于记录字符串长度，i用于循环。

函数算法描述：将数组的前半部分每个数与后半部分相加。再通过两次相减实现数据的交换。

3）函数原型：void Connect(List \*head, List \*tail)

功能描述： 拼接两个链表

参数描述：head为要放在前段的链表，tail表示放在后端的链表。

返回值描述：本函数无返回值。

重要局部变量定义：指针p

函数算法描述：首先判断head是否为空。若为空则返回tail。若不为空则利用p来寻找head中指向空指针的链表结。将其指向tail的首节；返回新的head链表首节点。

4）函数原型：void Free(List \*p)

功能描述：释放先前malloc申请的空间。

参数描述：需要释放的链表头指针。

返回值描述：无意义

重要局部变量定义：指针p；

函数算法描述：释放空间知道链表指向空指针。

5）函数原型：void Break(List \*head, List \*Breakpoint)

功能描述：将一个链表在指定处截断

参数描述：链表头指针，断点。

返回值描述：无意义

重要局部变量定义：指针p；

函数算法描述：释放空间知道链表指向空指针。

### 2.3.4 程序文件结构

1）文件函数结构

文件分为共享子串.c, function.c, function.h.

function.c中:

List \*FindKth(int K, List \*PtrL);

List \*Insert(char c, int t, int i, List \*Ptrl)

void Connect(List \*head, List \*tail);

void reverse(char c[]);

void Free(List \*p);

void Break(List \*head, List \*Breakpoint)

Function.h中包含函数原型。

2）多文件构成机制

在function,h中用ifndef防止重复定义

## 2.4 安装运行说明

运行时现将main.c、function.c文件编译为.o文件，再将.o文件以及头文件串联起来生成可执行的exe文件.

## 2.5 总结

### 2.5.1 实践过程中遇到的难点及解决方案

Free函数在释放空间时遇到程序崩溃对的情况。经过检查发现是因为两个链表衔接到同一个链表。第一个链表释放玩空间后并未将指针变为空指针。导致第二次释放时会尝试free掉一个已经被释放空间的指针，导致程序崩溃。后在释放空前之前先把链表拆分为两个独立的链表就可以顺利释放空间。

### 2.5.2 程序亮点或基本功能之外的拓展点

无。

### 2.5.3 实践之不足

程序功能较为单一。没有附加的功能。

### 2.5.4 分工合作和各自的收获

我们交流了各自的代码，在撰写代码的过程中获得了乐趣。

# n. 感想