串

本节内容的数组均从1开始

4.1 串类型的定义

串（字符串）

- 一般形式： $s =' a_1 a_2 \ldots a_n'$
- 空串
  - n=0
  - ∅
  - 区别于 空格串' ' ——— 一个或多个空格组成
- 长度 ——— n
- 子串 ——— 串中任意连续的字符
- 串相等
  - 长度相等
  - 每个位置都相等
- 必须用 单引号 括起来
- 最小操作子集
  （不能用其他串操作来实现）
  - ① 串赋值StrAssign
  - ② 串比较StrCompare
  - ③ 求串长StrLength
  - ④ 串联接StrCat
  - ⑤ 求子串SubString

4.2 串的表示和实现

① 定长顺序存储表示

- 用地址连续的存储单元存储
- 最大长度固定，超出长度的“截断”
- 串长
  - PASCAL：数组第0个位置，存储长度
  - c语言：'\0'表示串值终结
- **算法**
  - ① 串联接
    Status Concat(SString &T, SString S1， SString S2)
  - ② 求子串
    Status SubString(SString &Sub, SString S, int pos, int len)

② 堆分配存储表示

- 为了解决上面 最大长度 的限制 ——— 先分配空间，再复制
- **算法**
  - ① 串的插入操作
    Status StrInsert(HString &S, int pos, HString T)
  - ② 串的最小操作子集算法

③ 串的块链存储

- 可以存一个
- 也可以存多个 ——— 最后一块，不满的用#补上 ——— 串联接的时候，注意处理#
- 存储密度= 串值所占的存储位 / 实际分配的存储位
- 总的来说，没有前两种存储结构灵活，占用存储量大且操作复杂

4.3 串的模式匹配算法

模式匹配

- 子串定位 ——— 子串T称为模式串
- **算法**
  - ① 求子串位置的定位函数
    int Index(SString S, SString T, int pos)
    - 一般情况下，O(n+m)，所以至今仍被采用
    - 最坏时间复杂度O(m*n)
  - ② KMP算法
    int Index_KMP(SString S, SString T, int pos)
    - O(n+m)
    - 对处理外设输入的庞大文件很有效，可以边读入边匹配，而无需回头重读
    - 思想：不等时，只需要回退模式串指针j
      - next[j]：j处发生不等时，应该回退到模式串的哪个位置，再重新比较；
        next[j]的值表示，1~(j-1)的部分匹配值PM + 1；就是说，前有PM已经匹配，现在再看PM + 1位置，
        但是第一个位置特别注意，因为第一个都不匹配，直接i和j同时往后移，比较下一个就好了
      - $\text{next}[j] = \begin{cases} 0 & \text{当 } j = 1 \text{ 时} \\ \text{Max}\{k \mid 1 < k < j \text{ 且 } 'p_1 \cdots p_{k-1}' = 'p_{j-k+1} \cdots p_{j-1}'\} & \text{当此集合不空时} \\ 1 & \text{其他情况} \end{cases}$
      - eg:

j	1	2	3	4	5	6	7	8
模式串	a	b	a	a	b	c	a	c
$next[j]$	0	1	1	2	2	3	1	2

      - ③ 求next数组
        - O(m)
        - 思想，看成一个小KMP，模式串是子串也是主串
          但是第一个位置特别注意，因为第一个都不匹配，直接i和j同时往后移，比较下一个就好了（同样的）
      - ④ 改进next数组 ——— 如果j位置的值next[j]位置的值相等，那么next[j]=next[next[j]]

4.4 串操作应用举例

文本编辑

- 行表
  - 结构
    - 行号
    - 起始地址
    - 本行长度
  - 按行号递增存储
- 页表

删除行/页，可以只修改行表和页表内容，节省时间

建立词索引表 ——— P86