字符串

基本概念

数据结构中，字符串要单独用一种存储结构来存储，称为串存储结构。这里的串指的就是字符串。

严格意义上讲，串存储结构也是一种线性存储结构，因为字符串中的字符之间也具有"一对一"的逻辑关系。只不过，与之前所学的线性存储结构不同，串结构只用于存储字符类型的数据。

- 空串：存储 0 个字符的串，例如 S = ""（双引号紧挨着）；
- 空格串：只包含空格字符的串，例如 S = "　　"（双引号包含 5 个空格）；
- 子串和主串：假设有两个串 a 和 b，如果 a 中可以找到几个连续字符组成的串与 b 完全相同，则称 a 是 b 的主串，b 是 a 的子串。例如，若 a = "shujujiegou"，b = "shuju"，由于 a 中也包含 "shuju"，因此串 a 和串 b 是主串和子串的关系；

子串在主串中的位置，指的是子串首个字符在主串中的位置。

例如，串 a = "shujujiegou"，串 b = "jiegou"，通过观察，可以判断 a 和 b 是主串和子串的关系，同时子串 b 位于主串 a 中第 6 的位置，因为在串 a 中，串 b 首字符 'j' 的位置是 6。

实现串的模式匹配的算法主要有以下两种：

1. 普通的模式匹配算法；
2. 快速模式匹配算法；

需要掌握的东西：

1. 对文件进行操作
2. 暴力匹配算法
3. KMP算法

普通模式匹配（BF）算法

普通模式匹配算法，其实现过程没有任何技巧，就是简单粗暴地拿一个串同另一个串中的字符一一比对，得到最终结果。

例如，使用普通模式匹配算法判断串 A（"abcac"）是否为串 B（"ababcabacabab"）子串的判断过程如下：


```
int function(char* str1,char* str2){        //主串str1，子串str2，暴力匹配
    if(!str1||!str2)        //字符串不存在
        return -1;
    int i=0,j=0,k=0;
    while(str1[i]!='\0'&&str2[j]!='\0'){        //两个字符串都没有到末尾}
        if(str1[i]==str2[j]){
            ++i;
            ++j;
        }
        else{
            k++;
            i=k;
            j=0;
        }
    }
    if(str2[j]=='\0')
        return i-strlen(str2)+1;        //返回子串在主串中的位置
    else
        return -1;
}
```

该算法最理想的时间复杂度 `O(n)`，n 表示串 A 的长度，即第一次匹配就成功。

BF 算法最坏情况的时间复杂度为 `O(n*m)`

KMP算法

KMP算法：在一个文本串 S 内查找一个模式串 P 的出现位置

假设S匹配到了i位置，P匹配到了j位置

原来的普通匹配算法是如果有不匹配的情况：j 回到第一个位置，i 倒退为k+1。

KMP算法的改进：j 回到了 next[j] 的位置 i要么不变，要么加一

求next数组

首先理解前缀和后缀：

字符串：“bread”

前缀：b，br，bre，brea

后缀：read，ead，ad，d

next[i] 是前i-1个字符的前缀和后缀匹配的长度的最大值

尝试做一点例子：

模式串	A	A	B	A	B	A	B	B
下标	0	1	2	3	4	5	6	7
最长前后缀	0	0	0	1	2	3	4	-1
next值	-1	-1	0	0	1	2	3	4

模式串	A	B	C	S	A	B	C	D
顺序 i	0	1	2	3	4	5	6	7
最长前后缀 j	0	0	0	0	1	2	3	0
next值	-1	0	0	0	0	1	2	3

模式串	A	B	A	B	C	A	B	C
顺序	0	1	2	3	4	5	6	7
最长前后缀	0	0	1	2	0	1	2	0
next值	-1	0	0	1	2	0	1	2

next[i]的值等于已经匹配过的字符串的子串 Q 的最长匹配的前后缀的长度加1

- 算法流程

1. 初始化：next[0] = -1；从第一个字符开始遍历字串 i = 0，一个数用来记录当前匹配的前后缀的长度 j =-1；

2. 开始遍历模式串，即开始循环　while(i<strlen(P))

   如果 j == -1 或者　next[i] == next [j] ，说明下一个索引 (i+1) 的next数组的值已经确定，为 next[i+1] = j+1;

   否则回溯，重新确定下一个索引（i + 1）的值：

   j = next[j]

- 代码：

```
void Next(char* T,int * next){
    next[0] = -1;
    int i=0,j=-1;
    while(i<strlen(T)){
        if(j==-1||T[i]==T[j]){     //两种情况，当前情况匹配或者匹配长度为0了
            ++i;
            ++j;
            next[i] = j;
        }
        else        //回溯
            j = next[j];
    }
}
```

KMP算法

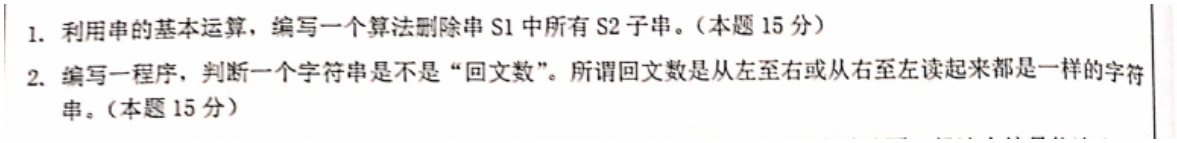
理解next数组怎么来的就应该会kmp算法了

```
int KMP(char const*S, char const*T) {
    int i = -1,j = -1;
    int next[20];
    mknext(T, next);
    int S_len, T_len;
    S_len = strlen(S);
    T_len = strlen(T);
    while (i < S_len && j < T_len) {
        if (S[i] == T[j] || j == -1) {
            i++;
            j++;
        }
        else
            j = next[j];
    }
    if (j >= T_len)
        return i - T_len;
```

```
    else
        return -1;
}
```

例题

1.


1. 利用串的基本运算，编写一个算法删除串 S1 中所有 S2 子串。（本题 15 分）
2. 编写一程序，判断一个字符串是不是“回文数”。所谓回文数是从左至右或从右至左读起来都是一样的字符串。（本题 15 分）

```
int function(char* str1,char* str2){        //主串str1，子串str2，暴力匹配
    if(!str1||!str2)
        return -1;
    int i=0,j=0,k=0;
    while(str1[i]!='\0'&&str2[j]!='\0'){        //两个字符串都没有到末尾}
        if(str1[i]==str2[j]){
            ++i;
            ++j;
        }
        else{
            k++;
            i=k;
            j=0;
        }
    }
    if(str2[j]=='\0')
        return i-strlen(str2)+1;        //返回子串在主串中的位置
    else
        return -1;
}
void delete(char* str1,int i,int length){        //删除子串
    if(!str1)
        return;
    int j;
    for(j=i;str1[j]!='\0';++j)  //一个一个替代
        str1[j] = str1[j+length];
    str1[j] = '\0';     //末尾变成\0
}
int main(){
```

```
    char str1[100],str2[20];
    printf("Enter str1,str2:\n");
    scanf("%s",str1);
    scanf("%s",str2);
    int length = strlen(str2);
    int i = function(str1,str2);
    while(i!=-1){        //找到所有子串并删除；
        delete(str1,i,length);
        i = function(str1,str2);
    }
    printf("result:%s",str1);
}
```

```

```

2.

3. 设定串采用顺序存储结构，求串 s1 和 s2 的一个最长公共子串的长度，并指出该最长公共子串分别在 s1 和 s2 中的起始位置。（本题 15 分）

4. 设单链表中存放 n 个字符，试设计一个算法，使用栈判断该字符串是否中心对称，如 xyzzyx 即为中心对称字符串。（小题 15 分）

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int match(char* str1,char* str2){        //判断str2是不是str1的子串
    if(!str1||!str2)
        return -1;
    int i=0,j=0,k=0;
    while(str1[i]!='\0'&&str2[j]!='\0'){
        if(str1[i]==str2[j]){
            i++;
            j++;
        }
        else{
            k++;
            i = k;
            j = 0;
        }
    }
    if(str2[j]=='\0')
        return i-strlen(str2);
    else
        return -1;
}
int main(){
```

```
    char s1[100],s2[100],s3[100];       //s1和s2字符串,s3保存子串
    printf("enter s1,s2:\n");
    scanf("%s%s",s1,s2);
    int i,j,len;
    for(len=strlen(s2);len>0;--len){        //子串的长度
        for(i=0;i+len<=strlen(s2);++i){     //该长度下的每一个子串
            for(j=0;j<len;++j)      //子串复制给s3
                s3[j] = s2[i+j];
            s3[j] = '\0';
            if(match(s1,s3)!=-1){
                len = 0;        //跳出整个循环
                break;
            }
        }
    }
    if(match(s1,s3)!=-1)        //有公共子串的情况
        printf("子串：%s\ns1：%d\ns2:%d",s3,match(s1,s3),i);
    else
        printf("no");
}
```

3.

2. 试采用递归函数实现将任意位数的整数转换为字符串输出，要求在主函数中输入整数并调用递归函数实现转换并输出结果，对于负数也能处理（ 15 分）
3. 以顺序存储结构表示串，设计算法，求串 S 中出现的第一个最长重复子串及其位置并分析算法的时间复杂度。（20 分）

```
void transform(int i,char* str,int p){      //i转化为字符数组
    if(i==0){       //递归结束条件
        str[p] = '\0';
        return;
    }
    str[p++] = i%10 + '0';
    transform(i/10,str,p);      //转换i/10的整数为字符串
}
void reverse(char* str,int n){      //翻转字符串
    int i = 0,j = strlen(str)-1;
    char temp;
    while(i<j){
        temp = str[i];
        str[i] = str[j];
        str[j] = temp;
    }
}
```

```c
int match(char* s1,char* s2,int len){        //判断s1和s2是否相等
    if(!s1||!s2)
        return 0;
    for(int i=0;i<len;++i)       //逐字符判断是否相等
        if(s1[i]!=s2[i])
            return 0;
    return 1;
}

int main(){
    char s[100];
    printf("Enter s:\n");
    scanf("%s",s);
    int len = strlen(s),i,j,k,judge=0;
    for(;len>0;--len){       //所有长度的子串
        for(i=0;i+len<=strlen(s);++i){       //找到一个子串
            for(j=i+1;j+len<=strlen(s);++j)      //找到其它的子串
                if(match(s+i,s+j,len)){      //如果匹配的情况
                    len = 0;         //为了跳出第一个循环
                    k = i;       //保留i值
                    i = strlen(s)+1;         //为了跳出第二个循环
                    judge = 1;       //有重复子串，标记一些
                    break;       //第三的个循环结束
                }
        }
    }
    if(judge){
        s[k+len] = '\0';         //为了输出子串
        printf("res:%s\ni:%d\nj:%d\n",s+k,k,j);
    }
    else
        printf("No.\n");
}
```

4.

1. 设 S 为一个长度为 n 的字符串，其中串的字符各不相同，写出具体程序并计算出 S 中互异的非平凡子串（非空且不同于 S 本身）的个数。（本题 15 分）。

5.

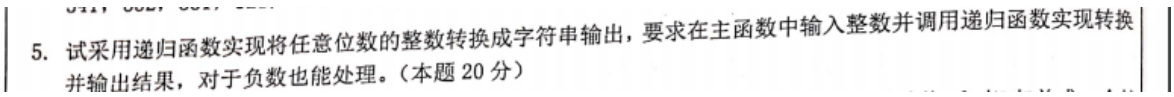
2. 采用顺序结构存储串，编写一个函数Substring(s1,s2)，用于判定s2 是否是 s1的子串。（本题15分）

6.

3. 编写程序用于统计字符串中最长单词的长度和在字符串中的位置，其中单词由字母组成。（本题 20 分）

```c
void longest(char* s,int* max,int* x){      //x，max保存字符串s中最长单词的位置和长度
    if(!s)
        return;
    int i=0,j=0;
    while(s[j]!=' ')          //j指向第一个空格
        ++j;
    *max = j-i-1;
    *x = i;
    while(s[j]!='\0'){        //i指向前一个空格，j指向后一个空格
        if(s[j]==' '){
            if(*max<(j-i)){
                *max = j-i-1;
                *x = i;
            }
            i = j;
            j++;
        }
        ++j;
    }
    if(*max<(j-i)){      //判断最后一个单词
                *max = j-i;
                *x = i;
    }
}
```

7.



5. 试采用递归函数实现将任意位数的整数转换成字符串输出，要求在主函数中输入整数并调用递归函数实现转换并输出结果，对于负数也能处理。（本题 20 分）

```c
void longest(char* s,int* max,int* x){      //x，max保存字符串s中最长单词的位置和长度
    if(!s)
        return;
    int i=0,j=0;
    while(s[j]!=' '||s[j]!='\0')
        ++j;
    *max = j-i;
    *x = i;
    while(s[j]!='\0'){
        if(s[j]==' '){
            if(max<j-i){
                *max = j-i;
                *x = i;
            }
            i = j;
            j++;
        }
```

```
    }
    if(max<j-i){
                *max = j-i;
                *x = i;
    }
}
```

```
void function(int n,int i,char a[]){//整数转为字符串
    if(n==0)         //结束条件
        return;
    a[i] = n%10 + '0';
    funcion(n/10,i++,a);
}
```

8.

2. 编写一个程序，利用递归法实现将用户输入的字符串逆序排列。（本题 15 分）

3. 找出所有 200 以内（含 200）满足 I，I+4，I+10 都是素数的整数 I（I+10 也在 200 以内）的个数以及这些数之和 sum。并把所有这些数、个数和 sum 按文本文件输出到文件 out.dat 中。（本题 20 分）

```
void function(char* s,int left,int right){
    if(left>right)
        return;
    char temp = s[left];
    s[left] = s[right];
    s[right] = temp;
    function(s,left+1,right-1);
}
```

9.

3. 已知 strcmp 的函数原型：int strcmp(char *s1, char *s2)。该函数的功能是比较字符串 s1 和 s2，当 s1<s2 时返回<0；s1=s2 时返回=0；s1>s2 时返回>0。编写程序实现函数 strcmp，不允许调用 C 语言库函数。（本题 15 分）

10.

2. 字符串和数值之间经常需要进行转换，C 库中的一个函数 atoi，可以实现字符串向整数的转换，其函数定义类似于：
int atoi(char *nptr);
请根据上述函数定义，实现此函数。提示：（1）字符’0’到’9’的 ASCII 码是连续的，为 48 到 57。（本题 15 分）

11.

8. 下面是一段英文文本，并且已经被读入到一个字符串中，请编写一段程序，统计其中的单词个数，每个单词出现的次数，可以利用你知道的C库函数辅助编写程序。在C库中可以使用的比较字符串的函数为int strcmp（const char *s1, const char *s2），s1和s2是两个需要比较的字符串，返回值为0时表示两个字符串相等。拷贝字符串的函数为char *strcpy(char *dest, char *src)，dest和src分别表示拷贝的源和目标。（可以将同一单词的不同时态、单复数等变化，视为不同的单词；不需要考虑示例文本中未出现的英语语法。）（本题20分）

Geovisualization is about working with maps and other views of the geographic information including interactive maps, 3D scenes, summary charts and tables, time-based views, and schematic views of network relations. A GIS includes interactive maps and other views that operate on the geographicdata set. Maps provide a powerful metaphor to define and standardize how people use and interact with geographic information.

```
typedef struct word{
    char str[100];       //单词
    int n;       //个数
}
int judge(char c){       //判断是否为字符
    ;
}
int search(word* arr,int n,char* s){//判断s是否再arr中
    ;
}
word arr[100];       //假设不超过100个单词
int n = 0;
void function(char* str){
    char s[100];
    int i,j,k;
    for(i=0;str[i]!='\0';++i){
        for(j=i+1;judge(str[j];++j){
            for(int k=i;k<j;++k)
                s[k-i] = str[k];
            s[k-i] = '\0';
        }
        i = j;
    }
    int temp = search(arr,n,s);
    if(temp==-1){
        strcpy(s,&arr[n].str);
        ++n;
    }else
            arr[temp].n++;
}
```

12.

7. （本题 15 分）在 C 语言中字符串是非常重要的的数据类型。请编写一个函数 char *strcat(char *a, char *b)，其功能是将字符串 a 和字符串 b 连接形成一个新的字符串，比如 char *a=" abc" ;char *b=" def" ；连接后的新字符串为“abcdef” 。

```
char* func(char* a,char* b){
    char* c  = (char*)malloc(sizeof(char)*100);
    int n = 0;
    for(int i=0;a[i]!='\0';++i)
        c[n++] = a[i];
    for(int i=0;b[i]!='\0';++i)
        c[n++] = b[i];
    c[n] = '\0';
    return c;
}
```

13.

2. 试编写程序：使用 KMP 算法实现子串 t 在主串 s 中定位。（本题 15 分）

14.

6、 试编写程序，能够计算一字符串中对称的子字符串的最大长度。例如：字符串“google”，由于该字符串里最长的对称子字符串是“goog”，因此输出 4。（25 分）

15.

6. （15 分）请设计一个 C 语言程序，从键盘输入一系列英文单词，单词用空格分割，结束输入后统计每个单词出现的次数，并按照出现的次数降序输出每个单词及其出现的次数。

提示：可用如下库函数辅助实现，也可自行实现或其他你了解的函数辅助实现

（1）int strcmp(char* s1, char* s2)是可使用的库函数，属于 string.h 头文件。该函数比较两个字符串，若 s1==s2 返回值为 0.

（2）void qsort(void *base, int nelem, int width, int (*fcmp)(const void *))是用于排序的库函数，在 stdlib.h 中定义，该函数的使用说明如下：

参数 1 是待排序数组的首地址

参数 2 是数组中待排序元素数量

参数 3 是数组中各元素的占用字节大小