

C语言中的字符串处理函数

- C语言提供了一系列字符串处理功能函数，所有这些函数只能处理字符串（即一定要有'\0'结尾）
- <stdio.h>，提供各类字符串输入输出转换函数
 - ◆ scanf、printf中提供%s操作字符串
 - ◆ puts、gets字符串输入和输出函数
 - ◆ sscanf、sprintf提供字符串和普通数据类型的转换
- <string.h>，以str开头，提供各类操作函数
 - ◆ 字符串长度：strlen
 - ◆ 字符串拷贝：strcpy、strncpy
 - ◆ 字符串连接：strcat、strncat
 - ◆ 字符串比较：strcmp、strncmp
 - ◆ 字符串查找：strstr、strchr、strrchr

C语言中的字符串处理函数

- 字符串基本构成单位为字符，字符处理函数可以简化字符处理
- <ctype.h>：字符处理函数
- 判断各类字符类型
 - ◆ 判断字母、数字等：isalpha、islower、isupper、isdigit、isalnum
 - ◆ 判断特殊字符：isprint、isspace、isxdigit、iscntrl、ispunct、isgraph
- 字符大小写转换
 - ◆ 将字符c转换为小写字母：tolower
 - ◆ 将字符c转换为大写字母：toupper

示例：C语言中字符串的定义和赋值

```
//定义一个长度为32的字符数组，可以最大长度为31的字符串
char arr1[32];
//定义一个字符串指针，需要指向某个有效的连续存储空间，该空间内容可看做字符串
char *str1;

strcpy(arr1, "Data Structure");
str1 = (char *) malloc( strlen(arr1) + 1 ); //根据arr1的大小(14个字符)分配字符串空间
strcpy(str1, arr1);
free(str1); //malloc分配的空间，必须通过free释放
//定义一个字符串数组，初始化数组成员
char arr2[] = "Data Structure";
//定义一个字符串指针，指向某个字符串常量
char *ptr2 = "Data Structure";
```

问题1：将数字字符串转换为整数

"123" → 123

$12 \times 10 + '3' - '0' = 123$
 $1 \times 10 + '2' - '0' = 12$
 $'1' - '0' = 1$

```
n = 0;
for(i=0; s[i]!='\0'; i++)
    n = 10*n + s[i] - '0';
```

字符	十进制 ASCII 码	十六进制 ASCII 码	字符	十进制 ASCII 码	十六进制 ASCII 码	字符	十进制 ASCII 码	十六进制 ASCII 码
0	48	30	A	65	41	a	97	61
1	49	31	B	66	42	b	98	62
2	50	32	C	67	43	c	99	63
3	51	33	D	68	44	d	100	64
4	52	34	E	69	45	e	101	65
5	53	35	F	70	46	f	102	66
6	54	36	G	71	47	g	103	67
7	55	37	H	72	48	h	104	68
8	56	38	I	73	49	i	105	69
9	57	39	J	74	4A	j	106	6A

问题1：代码实现

```
int atoi(char s[])
{
    int i, n, sign;
    for (i=0; s[i]!='\0' || s[i]!='\t'; i++)
        /* skip white space */
        sign = 1;
    if (s[i] == '+' || s[i] == '-')
        sign = (s[i++] == '+') ? 1 : -1;
    for (n = 0; s[i] >= '0' && s[i] <= '9'; i++)
        n = 10 * n + s[i] - '0';
    return (sign * n);
}
```

条件运算符：
 <表达式1> ? <表达式2> : <表达式3>
 先计算表达式1，若其值为非零，则整个表达式结果为表达式2的值，否则为表达式3的值

读入一个以空白字符分隔的字符串到s中，以'\0'结束（即由非空字符串组成的字符串）

为同字符串数组（字符串）作为函数参数传递时，通常不用传递数组长度？

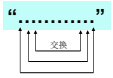
问题2：将整数转换为字符串

```
void itoa(int n, char s[])
{
    int i, sign;
    if ((sign = n) < 0)
        n = -n;
    i = 0;
    do
    {
        s[i++] = n % 10 + '0';
    } while ((n /= 10) > 0);
    if (sign < 0)
        s[i++] = '-';
    s[i] = '\0';
    reverse(s); //将字符串颠倒
}
```

```
#include <stdio.h>
void itoa(int n, char s[]);
int main()
{
    int n;
    char s[20];
    scanf("%d", &n);
    itoa(n, s);
    printf("%s\n", s);
    return 0;
}
```

提示：关于字符串和其他数据类型的转换，推荐使用sscanf和sprintf这两个函数

问题3：将字符串颠倒



```
void reverse(char s[])
{
    int c, i, j;
    for(i=0, j=strlen(s)-1; i<j; i++, j--){
        c = s[i];
        s[i] = s[j];
        s[j] = c;
    }
}
```

字符串组作为函数参数传递时，不需要同时传递数组长度，因为字符串中字符串是以 '\0' 结束的

提示：数组作为函数参数传递时，不要同时传递数组长度，取决于函数中是否知道数组中元素的个数

当有多个循环变量时，要用逗号隔开逗号表达式：如 `e1, e2` 顺序求 `e1` 和 `e2`，以 `e2` 值作为整个表达式结果的值。如，`a = (t = 3, t+2)`；结果为5

13

问题4：根据输入整数决定输出小数点位数

由参数确定输出的小数位数

从标准输入读入浮点数 x ($-10 < x < 10$) 和整数 m ($0 < m < 13$)，在标准输出上输出 $\sin(x)$ 的值，保留到小数点后 m 位数字，如：

- 输入：3.14 3，输出：0.002；
- 输入：3.14 10，输出：0.0015926529

问题分析

- 如果直接用 `printf("%.#f", sin(x))`，则需要用 `switch` 或 `if` 语句，有很多判断条件（这里 `#` 是常数，值跟输入的 `m` 相同）
- 利用 `sprintf` 函数，根据输入的 `m` 构造格式字符串

14

使用字符串转换函数：sprintf

将多个普通变量输出为sprintf：变量 -> 字符串

- `printf` 将不同变量输出到屏幕上，屏幕输出的本质就是字符串
- `sprintf` 用法与 `printf` 一样，只不过是输出到屏幕，而是直接保存到字符串中
- 比 `printf` 多一个参数（第一个参数），就是要存储的字符串数组，后面参数的使用与 `printf` 完全

```
//字符串转换函数，将其他变量转换为字符串
int sprintf(char *buf, char *format [, argument]...);
```

15

使用sprintf生成字符串

```
#include <stdio.h>
int main()
{
    int m; //小数点位数
    double x; //输入x
    char format[32]; //要存储的格式串
    scanf("%lf%d", &x, &m);
    //利用sprintf生成格式串
    sprintf(format, "%.df\n", m);
    //利用生成的格式串控制输出格式
    printf(format, sin(x));
}
```

若输入 `m` 为3，则 `"%.df\n"` 变为 `"%.3f\n"`，且该字符串存入字符串组 `format`，利用该字符串控制输出格式

说明：斜杠是编译器级别的转义，`%` 是 `printf` 内部的解析特殊符号，因此斜杠是不行的，只能是 `%%`，不能是 `\%`

sprintf 函数非常好用，任何可以输出到屏幕的数据都可以转换为字符串

问题5：从一行字符串中提取变量

提取日期和时间

- 计算机显示的时间通常有特殊的格式，比如计算机给出的格式 `17/Apr/2018:10:28:28 +0800`，表示北京时间2018年4月17日10时+28分28秒。给出一个这种格式表示的字符串，提取其中的每一项，并在屏幕上逐行显示出来

问题分析

- 定义各种类型的输入变量
- 编写格式控制符，从字符串中读入变量的值

17

使用字符串转换函数：sscanf

从字符串中转换出各种类型的变量，与sprintf对应

- `scanf` 从键盘读入一串符号，将其存入到不同变量中
- `sscanf` 用法与 `scanf` 一样，只不过不是从键盘输入，而是从一个给定的字符串中读入
- 比 `scanf` 多一个参数（第一个参数），就是要存储的要读取的字符串，后面参数的使用与 `printf` 完全

```
//字符串转换函数，从字符串中提取各种变量
int sscanf(const char *buf, char *format [, arg]...);
```

18

使用sscanf从字符串中提取变量

```
#include <stdio.h>
int main()
{
    int day, year, h, m, s;
    char mon[4], zone[6];
    char buf[] = "17/Apr/2018:10:28:28 +0800"; //要转换的字符串
    //从字符串中, 按照指定的格式读入日期的各个部分
    sscanf(buf, "%d/%3c/%d:%d:%d %s",
           &day, mon, &year, &h, &m, &s, zone);
    mon[3] = '\0'; //理解这句话的作用
    printf("%d\n%s\n%d\n%d\n%d\n%d\n",
           year, mon, day, h, m, s, zone); //按照格式输出
    return 0;
}
```

```
2018
Apr
4
17
28
+0800
```

19

问题6: 综合应用: 扩展字符

【问题描述】从键盘输入包含扩展字符“-”的字符串, 将其扩展为等价的完整字符, 例如将a-d扩展为abcd, 并输出扩展后的字符串。

- ◆要求: 只处理[a-z]、[A-Z]、[0-9]范围内的字符扩展, 即只有当扩展前后的字符同时是小写字母、大写字母或数字, 并且扩展后的字符大于扩展前的字符时才进行扩展, 其它情况不进行扩展, 原样输出。例如: a-R、D-e、0-b、4-8等字符串都不进行扩展。

【输入形式】: 从键盘输入包含扩展字符的字符串

【输出形式】: 输出扩展后的字符串

【输入样例1】: ADEa-g-m02

【输出样例1】: ADEabcd efghijklm02

【输入样例2】: cdeT-bcd

【输出样例2】: cdeT-bcd

【样例说明】

- ◆将样例1的输入ADEa-g-m02扩展为: ADEabcd efghijklm02; 样例2的输入cdeT-bcd中, 扩展前的字符为大写字母, 扩展后的字符为小写字母, 不在同一范围内, 所以不扩展

20

问题分析

基本字符串(串)处理

- ◆读写字符串: gets、puts

判断是否需要进行字符串扩展

```
//判断是否需要扩展字符串
int isexpand(char start, char end);
```

字符串扩展

```
//扩展字符串
for (ch = src[i] + 1; ch < src[i + 2]; ch++)
    dest[j++] = ch;
```

21

代码实现: 主函数和判断是否扩展函数

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#define LEN 1024
//判断是否需要扩展字符串
int isexpand(char start, char end);
//将源串扩展为目标串
char *expand(const char *src, char *dest);
int main()
{
    char src[LEN], dest[LEN];
    gets(src);
    expand(src, dest);
    puts(dest);
    return 0;
}

//判断是否需要扩展字符串
int isexpand(char start, char end)
{
    if (islower(start) && islower(end) && start < end)
        return 1;
    if (isupper(start) && isupper(end) && start < end)
        return 1;
    if (isdigit(start) && isdigit(end) && start < end)
        return 1;
    return 0;
}
```

代码实现: 扩展字符串

```
//将源串扩展为目标串
char *expand(const char *src, char *dest) {
    char ch;
    int i, j;
    for (i = 0; j = 0; src[i] != '\0'; i++){
        dest[j++] = src[i];
        if (src[i + 1] == '-') {
            // 如果需要扩展, 则扩展字符串
            if (isexpand(src[i], src[i + 2])){
                for (ch = src[i] + 1; ch < src[i + 2]; ch++){
                    dest[j++] = ch;
                }
            }
        }
    }
    dest[j] = '\0'; //最后一一定要补充\0
    return dest;
}
```

23

问题7: 字符串替换

问题描述

- ◆从标准输入读入数据中, 每行最多包含一个字符串 "xy_". 将输入行中的 "x_y" 替换为 "_abc_", 在标准输出上输出替换后的结果

问题分析

- ◆如何找到 "xy_" : strstr?
- ◆找到后, 如何替换为 "_abc_" : 数组操作、指针运算?

如何实现字符串查找函数?

24

字符串查找：库函数

字符串查找库函数：strstr

```
//在字符串中查询是否存在子字符串sub_str
//返回中第一次出现的位置指针
//如果不存在，则返回NULL。
char *strstr(char *s, char *sub_str);
```

28

问题7：字符串替换

提示：BUFSIZ是一个常用的说明缓冲区大小的符号常量，实际数量取决于编译系统的版本和编译选项，常见的数值为512或4096

```
#include <stdio.h>
#include <string.h>
int main(){
    char buf[BUFSIZ], *p, *str = "_xy_";
    while (gets(buf) != NULL){ //读入字符串
        p = strstr(buf, str); //字符串中查询子串
        if (p == NULL){ //找不到子串，直接输出
            printf("%s", buf);
            continue;
        }
        *p = '\0'; //将找到的位置设置为\0，从而输出前面的串
        //用%s输出时，通过buf只能输出前面的串
        //代码中直接输出_abc_子串
        //p+strlen(str)通过指针运算，输出_xy_后面的串
        printf("%s_abc_%s", buf, p + strlen(str));
    }
    return 0;
}
```

输出示例 abc_xy_ef

buf a b c _ x y _ e f \0

被替换的字符不输出
其它字符照常输出
用指针指向开始输出的位置

从子串 "_xy_" 的第一个字符将输入分为两部分

29

字符串查找：strstr函数的代码实现

使用C代码实现（在字符串s中查找t）

```
int index(char s[], char t[]){
    int i, j, k;
    for (i = 0; s[i] != '\0'; i++)
    {
        for (j = i, k = 0;
             t[k] != '\0' && s[j] == t[k];
             j++, k++);
        if (t[k] == '\0')
            return i;
    }
    return -1;
}
```

主串s

子串t

27

思考1-不区分大小写

- 目前实现了大小写相关的字符串查找，即字符串"the"和"The"是不同字符串
- 如何实现大小写无关的字符串查找？

在比较字符时，可将要比较字符均转换为小写或大写，即可实现大小写无关查找

设函数char tolower(char c)用于将字符c转换为相应小写字符，则上面index可改为

```
int index(char s[], char t[]){
    int i, j, k;
    for (i = 0; s[i] != '\0'; i++){
        for (j = i, k = 0;
             t[k] != '\0' && tolower(s[j]) == tolower(t[k]);
             j++, k++);
        if (t[k] == '\0')
            return (i);
    }
    return (-1);
}
```

28

思考2-字符串的查找算法

主串s: a b a b c a c c a b

模式串t: a b c a

匹配过程：i=0, j=0; i=1, j=1; i=2, j=2; i=3, j=3; i=4, j=4; i=5, j=5; i=6, j=6; i=7, j=7; i=8, j=8; i=9, j=9; i=10, j=10; i=11, j=11; i=12, j=12; i=13, j=13; i=14, j=14; i=15, j=15; i=16, j=16; i=17, j=17; i=18, j=18; i=19, j=19; i=20, j=20; i=21, j=21; i=22, j=22; i=23, j=23; i=24, j=24; i=25, j=25; i=26, j=26; i=27, j=27; i=28, j=28; i=29, j=29; i=30, j=30; i=31, j=31; i=32, j=32; i=33, j=33; i=34, j=34; i=35, j=35; i=36, j=36; i=37, j=37; i=38, j=38; i=39, j=39; i=40, j=40; i=41, j=41; i=42, j=42; i=43, j=43; i=44, j=44; i=45, j=45; i=46, j=46; i=47, j=47; i=48, j=48; i=49, j=49; i=50, j=50; i=51, j=51; i=52, j=52; i=53, j=53; i=54, j=54; i=55, j=55; i=56, j=56; i=57, j=57; i=58, j=58; i=59, j=59; i=60, j=60; i=61, j=61; i=62, j=62; i=63, j=63; i=64, j=64; i=65, j=65; i=66, j=66; i=67, j=67; i=68, j=68; i=69, j=69; i=70, j=70; i=71, j=71; i=72, j=72; i=73, j=73; i=74, j=74; i=75, j=75; i=76, j=76; i=77, j=77; i=78, j=78; i=79, j=79; i=80, j=80; i=81, j=81; i=82, j=82; i=83, j=83; i=84, j=84; i=85, j=85; i=86, j=86; i=87, j=87; i=88, j=88; i=89, j=89; i=90, j=90; i=91, j=91; i=92, j=92; i=93, j=93; i=94, j=94; i=95, j=95; i=96, j=96; i=97, j=97; i=98, j=98; i=99, j=99; i=100, j=100; i=101, j=101; i=102, j=102; i=103, j=103; i=104, j=104; i=105, j=105; i=106, j=106; i=107, j=107; i=108, j=108; i=109, j=109; i=110, j=110; i=111, j=111; i=112, j=112; i=113, j=113; i=114, j=114; i=115, j=115; i=116, j=116; i=117, j=117; i=118, j=118; i=119, j=119; i=120, j=120; i=121, j=121; i=122, j=122; i=123, j=123; i=124, j=124; i=125, j=125; i=126, j=126; i=127, j=127; i=128, j=128; i=129, j=129; i=130, j=130; i=131, j=131; i=132, j=132; i=133, j=133; i=134, j=134; i=135, j=135; i=136, j=136; i=137, j=137; i=138, j=138; i=139, j=139; i=140, j=140; i=141, j=141; i=142, j=142; i=143, j=143; i=144, j=144; i=145, j=145; i=146, j=146; i=147, j=147; i=148, j=148; i=149, j=149; i=150, j=150; i=151, j=151; i=152, j=152; i=153, j=153; i=154, j=154; i=155, j=155; i=156, j=156; i=157, j=157; i=158, j=158; i=159, j=159; i=160, j=160; i=161, j=161; i=162, j=162; i=163, j=163; i=164, j=164; i=165, j=165; i=166, j=166; i=167, j=167; i=168, j=168; i=169, j=169; i=170, j=170; i=171, j=171; i=172, j=172; i=173, j=173; i=174, j=174; i=175, j=175; i=176, j=176; i=177, j=177; i=178, j=178; i=179, j=179; i=180, j=180; i=181, j=181; i=182, j=182; i=183, j=183; i=184, j=184; i=185, j=185; i=186, j=186; i=187, j=187; i=188, j=188; i=189, j=189; i=190, j=190; i=191, j=191; i=192, j=192; i=193, j=193; i=194, j=194; i=195, j=195; i=196, j=196; i=197, j=197; i=198, j=198; i=199, j=199; i=200, j=200; i=201, j=201; i=202, j=202; i=203, j=203; i=204, j=204; i=205, j=205; i=206, j=206; i=207, j=207; i=208, j=208; i=209, j=209; i=210, j=210; i=211, j=211; i=212, j=212; i=213, j=213; i=214, j=214; i=215, j=215; i=216, j=216; i=217, j=217; i=218, j=218; i=219, j=219; i=220, j=220; i=221, j=221; i=222, j=222; i=223, j=223; i=224, j=224; i=225, j=225; i=226, j=226; i=227, j=227; i=228, j=228; i=229, j=229; i=230, j=230; i=231, j=231; i=232, j=232; i=233, j=233; i=234, j=234; i=235, j=235; i=236, j=236; i=237, j=237; i=238, j=238; i=239, j=239; i=240, j=240; i=241, j=241; i=242, j=242; i=243, j=243; i=244, j=244; i=245, j=245; i=246, j=246; i=247, j=247; i=248, j=248; i=249, j=249; i=250, j=250; i=251, j=251; i=252, j=252; i=253, j=253; i=254, j=254; i=255, j=255; i=256, j=256; i=257, j=257; i=258, j=258; i=259, j=259; i=260, j=260; i=261, j=261; i=262, j=262; i=263, j=263; i=264, j=264; i=265, j=265; i=266, j=266; i=267, j=267; i=268, j=268; i=269, j=269; i=270, j=270; i=271, j=271; i=272, j=272; i=273, j=273; i=274, j=274; i=275, j=275; i=276, j=276; i=277, j=277; i=278, j=278; i=279, j=279; i=280, j=280; i=281, j=281; i=282, j=282; i=283, j=283; i=284, j=284; i=285, j=285; i=286, j=286; i=287, j=287; i=288, j=288; i=289, j=289; i=290, j=290; i=291, j=291; i=292, j=292; i=293, j=293; i=294, j=294; i=295, j=295; i=296, j=296; i=297, j=297; i=298, j=298; i=299, j=299; i=300, j=300; i=301, j=301; i=302, j=302; i=303, j=303; i=304, j=304; i=305, j=305; i=306, j=306; i=307, j=307; i=308, j=308; i=309, j=309; i=310, j=310; i=311, j=311; i=312, j=312; i=313, j=313; i=314, j=314; i=315, j=315; i=316, j=316; i=317, j=317; i=318, j=318; i=319, j=319; i=320, j=320; i=321, j=321; i=322, j=322; i=323, j=323; i=324, j=324; i=325, j=325; i=326, j=326; i=327, j=327; i=328, j=328; i=329, j=329; i=330, j=330; i=331, j=331; i=332, j=332; i=333, j=333; i=334, j=334; i=335, j=335; i=336, j=336; i=337, j=337; i=338, j=338; i=339, j=339; i=340, j=340; i=341, j=341; i=342, j=342; i=343, j=343; i=344, j=344; i=345, j=345; i=346, j=346; i=347, j=347; i=348, j=348; i=349, j=349; i=350, j=350; i=351, j=351; i=352, j=352; i=353, j=353; i=354, j=354; i=355, j=355; i=356, j=356; i=357, j=357; i=358, j=358; i=359, j=359; i=360, j=360; i=361, j=361; i=362, j=362; i=363, j=363; i=364, j=364; i=365, j=365; i=366, j=366; i=367, j=367; i=368, j=368; i=369, j=369; i=370, j=370; i=371, j=371; i=372, j=372; i=373, j=373; i=374, j=374; i=375, j=375; i=376, j=376; i=377, j=377; i=378, j=378; i=379, j=379; i=380, j=380; i=381, j=381; i=382, j=382; i=383, j=383; i=384, j=384; i=385, j=385; i=386, j=386; i=387, j=387; i=388, j=388; i=389, j=389; i=390, j=390; i=391, j=391; i=392, j=392; i=393, j=393; i=394, j=394; i=395, j=395; i=396, j=396; i=397, j=397; i=398, j=398; i=399, j=399; i=400, j=400; i=401, j=401; i=402, j=402; i=403, j=403; i=404, j=404; i=405, j=405; i=406, j=406; i=407, j=407; i=408, j=408; i=409, j=409; i=410, j=410; i=411, j=411; i=412, j=412; i=413, j=413; i=414, j=414; i=415, j=415; i=416, j=416; i=417, j=417; i=418, j=418; i=419, j=419; i=420, j=420; i=421, j=421; i=422, j=422; i=423, j=423; i=424, j=424; i=425, j=425; i=426, j=426; i=427, j=427; i=428, j=428; i=429, j=429; i=430, j=430; i=431, j=431; i=432, j=432; i=433, j=433; i=434, j=434; i=435, j=435; i=436, j=436; i=437, j=437; i=438, j=438; i=439, j=439; i=440, j=440; i=441, j=441; i=442, j=442; i=443, j=443; i=444, j=444; i=445, j=445; i=446, j=446; i=447, j=447; i=448, j=448; i=449, j=449; i=450, j=450; i=451, j=451; i=452, j=452; i=453, j=453; i=454, j=454; i=455, j=455; i=456, j=456; i=457, j=457; i=458, j=458; i=459, j=459; i=460, j=460; i=461, j=461; i=462, j=462; i=463, j=463; i=464, j=464; i=465, j=465; i=466, j=466; i=467, j=467; i=468, j=468; i=469, j=469; i=470, j=470; i=471, j=471; i=472, j=472; i=473, j=473; i=474, j=474; i=475, j=475; i=476, j=476; i=477, j=477; i=478, j=478; i=479, j=479; i=480, j=480; i=481, j=481; i=482, j=482; i=483, j=483; i=484, j=484; i=485, j=485; i=486, j=486; i=487, j=487; i=488, j=488; i=489, j=489; i=490, j=490; i=491, j=491; i=492, j=492; i=493, j=493; i=494, j=494; i=495, j=495; i=496, j=496; i=497, j=497; i=498, j=498; i=499, j=499; i=500, j=500; i=501, j=501; i=502, j=502; i=503, j=503; i=504, j=504; i=505, j=505; i=506, j=506; i=507, j=507; i=508, j=508; i=509, j=509; i=510, j=510; i=511, j=511; i=512, j=512; i=513, j=513; i=514, j=514; i=515, j=515; i=516, j=516; i=517, j=517; i=518, j=518; i=519, j=519; i=520, j=520; i=521, j=521; i=522, j=522; i=523, j=523; i=524, j=524; i=525, j=525; i=526, j=526; i=527, j=527; i=528, j=528; i=529, j=529; i=530, j=530; i=531, j=531; i=532, j=532; i=533, j=533; i=534, j=534; i=535, j=535; i=536, j=536; i=537, j=537; i=538, j=538; i=539, j=539; i=540, j=540; i=541, j=541; i=542, j=542; i=543, j=543; i=544, j=544; i=545, j=545; i=546, j=546; i=547, j=547; i=548, j=548; i=549, j=549; i=550, j=550; i=551, j=551; i=552, j=552; i=553, j=553; i=554, j=554; i=555, j=555; i=556, j=556; i=557, j=557; i=558, j=558; i=559, j=559; i=560, j=560; i=561, j=561; i=562, j=562; i=563, j=563; i=564, j=564; i=565, j=565; i=566, j=566; i=567, j=567; i=568, j=568; i=569, j=569; i=570, j=570; i=571, j=571; i=572, j=572; i=573, j=573; i=574, j=574; i=575, j=575; i=576, j=576; i=577, j=577; i=578, j=578; i=579, j=579; i=580, j=580; i=581, j=581; i=582, j=582; i=583, j=583; i=584, j=584; i=585, j=585; i=586, j=586; i=587, j=587; i=588, j=588; i=589, j=589; i=590, j=590; i=591, j=591; i=592, j=592; i=593, j=593; i=594, j=594; i=595, j=595; i=596, j=596; i=597, j=597; i=598, j=598; i=599, j=599; i=600, j=600; i=601, j=601; i=602, j=602; i=603, j=603; i=604, j=604; i=605, j=605; i=606, j=606; i=607, j=607; i=608, j=608; i=609, j=609; i=610, j=610; i=611, j=611; i=612, j=612; i=613, j=613; i=614, j=614; i=615, j=615; i=616, j=616; i=617, j=617; i=618, j=618; i=619, j=619; i=620, j=620; i=621, j=621; i=622, j=622; i=623, j=623; i=624, j=624; i=625, j=625; i=626, j=626; i=627, j=627; i=628, j=628; i=629, j=629; i=630, j=630; i=631, j=631; i=632, j=632; i=633, j=633; i=634, j=634; i=635, j=635; i=636, j=636; i=637, j=637; i=638, j=638; i=639, j=639; i=640, j=640; i=641, j=641; i=642, j=642; i=643, j=643; i=644, j=644; i=645, j=645; i=646, j=646; i=647, j=647; i=648, j=648; i=649, j=649; i=650, j=650; i=651, j=651; i=652, j=652; i=653, j=653; i=654, j=654; i=655, j=655; i=656, j=656; i=657, j=657; i=658, j=658; i=659, j=659; i=660, j=660; i=661, j=661; i=662, j=662; i=663, j=663; i=664, j=664; i=665, j=665; i=666, j=666; i=667, j=667; i=668, j=668; i=669, j=669; i=670, j=670; i=671, j=671; i=672, j=672; i=673, j=673; i=674, j=674; i=675, j=675; i=676, j=676; i=677, j=677; i=678, j=678; i=679, j=679; i=680, j=680; i=681, j=681; i=682, j=682; i=683, j=683; i=684, j=684; i=685, j=685; i=686, j=686; i=687, j=687; i=688, j=688; i=689, j=689; i=690, j=690; i=691, j=691; i=692, j=692; i=693, j=693; i=694, j=694; i=695, j=695; i=696, j=696; i=697, j=697; i=698, j=698; i=699, j=699; i=700, j=700; i=701, j=701; i=702, j=702; i=703, j=703; i=704, j=704; i=705, j=705; i=706, j=706; i=707, j=707; i=708, j=708; i=709, j=709; i=710, j=710; i=711, j=711; i=712, j=712; i=713, j=713; i=714, j=714; i=715, j=715; i=716, j=716; i=717, j=717; i=718, j=718; i=719, j=719; i=720, j=720; i=721, j=721; i=722, j=722; i=723, j=723; i=724, j=724; i=725, j=725; i=726, j=726; i=727, j=727; i=728, j=728; i=729, j=729; i=730, j=730; i=731, j=731; i=732, j=732; i=733, j=733; i=734, j=734; i=735, j=735; i=736, j=736; i=737, j=737; i=738, j=738; i=739, j=739; i=740, j=740; i=741, j=741; i=742, j=742; i=743, j=743; i=744, j=744; i=745, j=745; i=746, j=746; i=747, j=747; i=748, j=748; i=749, j=749; i=750, j=750; i=751, j=751; i=752, j=752; i=753, j=753; i=754, j=754; i=755, j=755; i=756, j=756; i=757, j=757; i=758, j=758; i=759, j=759; i=760, j=760; i=761, j=761; i=762, j=762; i=763, j=763; i=764, j=764; i=765, j=765; i=766, j=766; i=767, j=767; i=768, j=768; i=769, j=769; i=770, j=770; i=771, j=771; i=772, j=772; i=773, j=773; i=774, j=774; i=775, j=775; i=776, j=776; i=777, j=777; i=778, j=778; i=779, j=779; i=780, j=780; i=781, j=781; i=782, j=782; i=783, j=783; i=784, j=784; i=785, j=785; i=786, j=786; i=787, j=787; i=788, j=788; i=789, j=789; i=790, j=790; i=791, j=791; i=792, j=792; i=793, j=793; i=794, j=794; i=795, j=795; i=796, j=796; i=797, j=797; i=798, j=798; i=799, j=799; i=800, j=800; i=801, j=801; i=802, j=802; i=803, j=803; i=804, j=804; i=805, j=805; i=806, j=806; i=807, j=807; i=808, j=808; i=809, j=809; i=810, j=810; i=811, j=811; i=812, j=812; i=813, j=813; i=814, j=814; i=815, j=815; i=816, j=816; i=817, j=817; i=818, j=818; i=819, j=819; i=820, j=820; i=821, j=821; i=822, j=822; i=823, j=823; i=824, j=824; i=825, j=825; i=826, j=826; i=827, j=827; i=828, j=828; i=829, j=829; i=830, j=830; i=831, j=831; i=832, j=832; i=833, j=833; i=834, j=834; i=835, j=835; i=836, j=836; i=837, j=837; i=838, j=838; i=839, j=839; i=840, j=840; i=841, j=841; i=842, j=842; i=843, j=843; i=844, j=844; i=845, j=845; i=846, j=846; i=847, j=847; i=848, j=848; i=849, j=849; i=850, j=850; i=851, j=851; i=852, j=852; i=853, j=853; i=854, j=854; i=855, j=855; i=856, j=856; i=857, j=857; i=858, j=858; i=859, j=859; i=860, j=860; i=861, j=861; i=862, j=862; i=863, j=863; i=864, j=864; i=865, j=865; i=866, j=866; i=867, j=867; i=868, j=868; i=869, j=869; i=870, j=870; i=871, j=871; i=872, j=872; i=873, j=873; i=874, j=874; i=875, j=875; i=876, j=876; i=877, j=877; i=878, j=878; i=879, j=879; i=880, j=880; i=881, j=881; i=882, j=882; i=883, j=883; i=884, j=884; i=885, j=885; i=886, j=886; i=887, j=887; i=888, j=888; i=889, j=889; i=890, j=890; i=891, j=891; i=892, j=892; i=893, j=893; i=894, j=894; i=895, j=895; i=896, j=896; i=897, j=897; i=898, j=898; i=899, j=899; i=900, j=900; i=901, j=901; i=902, j=902; i=903, j=903; i=904, j=904; i=905, j=905; i=906, j=906; i=907, j=907; i=908, j=908; i=909, j=909; i=910, j=910; i=911, j=911; i=912, j=912; i=913, j=913; i=914, j=914; i=915, j=915; i=916, j=916; i=917, j=917; i=918, j=9

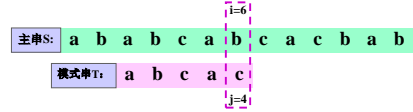
暴力破解 (BF, Brute Force) 法的另一种实现

- index查找函数使用两层循环, 可改为只是用一层循环的字符串查找算法, 但时间复杂度并没有变化

```
int index(char s[], char t[]) {
    int i=0, j=0; //设置查找的起始位置
    while(s[i] != '\0' && t[j] != '\0') {
        if(s[i] == t[j]) { //若字符相等, 继续查找下一个字符
            i++; j++;
        }
        else { //若字符不等, 则s中退回到上次查找开始的下一个位置
            i = i - j + 1;
            j = 0;
        }
    }
    if(t[j] == '\0') //查找到字符串t, 返回t在s中的起始位置
        return i - j;
    else
        return -1;
}
```

31

思考3-更高效的模式匹配算法: 串的模式匹配



KMP (Knuth-Morris-Pratt) 算法

- 源串称为主串, 定义为 S , 当前匹配位置为 i ; 目标串称为子串, 定义为 T , 当前匹配位置为 j
- 当前匹配在找到不匹配的字符后, 重新开始匹配时:
 - 主串当前位置 i 不回溯, 即不重置为上次匹配开始位置的一下位置
 - 子串当前位置 j 视情况回溯至起始串位置或子串中某一位置

32

KMP算法核心思想: 计算子串回溯位置

根据子串T当前匹配的规律: " $T_0 \dots T_{k-1}$ " = " $T_{j-k} \dots T_{j-1}$ "
 由当前失配位置 j (已知), 可以归纳计算下次匹配起点 k 的表达式。
 令 $k = \text{next}[j]$ (函数next用子串当前位置 j 来计算下次开始匹配位置 k , k 与 j 显然具有函数关系), 则

$$\text{next}[j] = \begin{cases} -1 & \text{当 } j=0 \text{ 时} \\ \max \{ k \mid 0 \leq k < j \text{ 且 } 'T_0 \dots T_{k-1}' = 'T_{j-k} \dots T_{j-1}' \} & \text{其他情况} \end{cases}$$

取当前位置j前序串首尾最大的相同子串长度

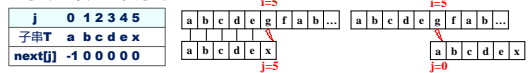
注意:

- (1) k 值仅取决于子串本身而与相匹配的主串无关。
- (2) k 值为子串从头向后及从 j 向前的两部分子串最大相同子串的长度。
- (3) 这里的两部分子串可以有部分重叠的字符, 但不可以全部重叠, 即 k 最大为 $j-1$ 。

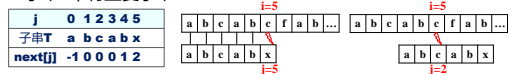
33

KMP算法核心思想

1. 子串T中无重复子串



2. 子串T中有重复子串



34

KMP算法C语言实现

```
int KMPindex(char S[], char T[]) {
    int i = 0, j = 0, *next;
    next = (int *)malloc(sizeof(int)*(strlen(T)+1));
    getnext(T, next); //求next
    while (S[i] != '\0' && T[j] != '\0') {
        if (S[i] == T[j]) {
            i++;
            j++;
        }
        else {
            //j退回到相应位置开始匹配, i不变
            (j == 0) ? i++ : (j = next[j]);
        }
        free(next);
    }
    if (T[j] == '\0') //匹配成功, 返回匹配位置
        return i - j;
    else
        return -1;
}
```

算法分析:

由于指针 i 无须回溯, 比较次数仅为 n , 加上计算 $\text{next}[j]$ 时所用的比较次数 m , 比较总次数也仅为 $n+m$, 时间复杂度为 $O(n+m)$, 大大优于朴素的Brute-Force算法

35

KMP算法C代码实现: 计算next

```
void getnext(char T[], int next[]) {
    int i = 0, j = -1;
    next[0] = -1;
    while (T[i] != '\0') {
        if (j == -1 || T[i] == T[j]) {
            i++;
            j++;
            next[i] = j;
        }
        else {
            j = next[j]; //若字符不同, 则j值回溯
        }
    }
}
```

36

问题4：思考4-更复杂的字符串查找

- ▢ 注意：index函数只能查找子字符串首次出现，如果一行中含有多个要查找的字符串怎么办？(类似Office软件中的**查找**功能)
 - ◆ 如何查找子字符串的最后一次出现？
 - ◆ 如果要查找一个字符串在一个文件中的所有出现（如给出所有出现的行列位置），如何实现？
- ▢ 在一个文件中查找给定串并用另一个替换串，如何实现？(Office软件中的**替换**功能)
- ▢ 如何实现模糊查找（如UNIX命令**grep**），如要查找的串形式为：
comp?ter, com*er (?单字符匹配, *多字符匹配)

37

单选题 1分

有如下变量声明语句：

```
char *name[] = {"Zhangsan", "Lisi", "Wangwu"};
则表达式 " *name[1] != *name[2]" 比较的是_____。
```

- ☐ A 字符串"Zhangsan"和"Lisi";
- ☐ B 字符'Z'和'L';
- ☐ C 字符串"Lisi"和"Wangwu";
- ☐ D 字符'L'和'W'。

38

找错题

```
Main()
{
    char string1[10] = "hello";
    char string[10] = {'h', 'e', 'l', 'l', 'o', '\0'};
    string1="aaaaaaa";    --错!
    .....
```

- ▢ 字符串以' \0' 作为结束标志
- ▢ 字符串在定义中可以整体初始化，但不能使用赋值语句整体赋值