第4章 串

1．选择题

（1）串是一种特殊的线性表，其特殊性体现在（ ）。

A．可以顺序存储 B．数据元素是一个字符

C．可以链式存储 D．数据元素可以是多个字符

（2）串下面关于串的的叙述中，（ ）是不正确的？

A．串是字符的有限序列 B．空串是由空格构成的串

C．模式匹配是串的一种重要运算 D．串既可以采用顺序存储，也可以采用链式存储

（3）串“ababaaababaa”的next数组为（ ）。

A．012345678999 B．012121111212 C．011234223456 D．0123012322345

（4 ）串的长度是指（ ）。

A．串中所含不同字母的个数 B．串中所含字符的个数

C．串中所含不同字符的个数 D．串中所含非空格字符的个数

（5）设有两个串p和q，其中q是p的子串，求q在p中首次出现的位置的算法称为（    ）

A．求子串        B．联接

C．匹配          D．求串长

（6） 已知串S=‘aaab’,其Next数组值为（    ）。

A．0123         B．1123

C．1231           D．1211

（7）字符串‘ababaabab’ 的next 为（     ）

A．(0,1,0,1,4,1,0,1)       B．(0,1,0,1,0,2,1,0,1)

C．(0,1,0,1,0,0,0,1,1)       D．(0,1,1,2,3,4,2,3,4 )

答案：B B C B C A D

二．应用题与算法题

1．已知模式串t=‘abcaabbabcab’写出用KMP法求得的每个字符对应的next函数值。

答案：

模式串t的next值如下：

|  |  |
| --- | --- |
| j | 1 2 3 4 5 6 7 8 9 10 11 12 |
| t串 | a b c a a b b a b c a b |
| next[j] | 0 1 1 1 2 2 3 1 2 3 4 5 |

2．设主串为t=“abcaabbabcabaacbacba”,模式为p=“abcabaa”；

① 计算模式p的next函数值；

② 不写出算法,只画出利用KMP算法进行模式匹配时每一趟的匹配过程。

p的next函数值为0111232

利用KMP(next)算法，每趟匹配过程如下：(i不回溯)

第一趟匹配： abcaabbabcabaacbacba

abcab (i=5,j=5，失配，j=next[5]=2)

第二趟匹配： abcaabbabcabaacbacba

abc (i=5,j=2,失配，j=next[2]=1)

第三趟匹配： abcaabbabcabaacbacba

abc (i=7,j=3,失配，j=next[3]=1)

第四趟匹配： abcaabbabcabaac bacba

a (i=7,j=1,失配，j=next[1]=0 , i=8,j=1开始下一次)

第五趟匹配 abcaabbabcabaacbacba

abcabaa (i=8,j=1 开始匹配成功)

3.设s、t为两个字符串，分别放在两个一维数组中，m、n分别为其长度，判断t是否为s的子串。如果是，输出子串所在位置（第一个字符），否则输出0。（注：用程序实现）

**int**index(**char** s[],t[],**int**m,n)

//字符串s和t用一维数组存储，其长度分别为m和n。本算法求字符串t在字符串s中的第一次出现，如是，输出子串在s中的位置，否则输出0。

{**int**i=0,j=0;

**while** (i<=m-n && j<=n-1)

**if** (s[i]==t[j])

{i++;j++;}   //对应字符相等，指针后移。

**else** {i=i-j+1;j=0;}  //对应字符不相等，I回溯，j仍为0。

**if**(i<=m-n && j==n)

{printf(“t在s串中位置是%d”,i-n+1);

**return**(i-n+1);

}//匹配成功

**else**  **return**(0); //匹配失败

}//算法index结束

main ()//主函数

{**char** s[80],t[80];

**int**m,n,i;

 scanf(“%d%d”,&m,&n);  //输入两字符串的长度

 scanf(“%s”,s);        //输入主串

 scanf(“%s”,t);        //输入子串

 i=index(s,t,m,n);

}//程序结束

4．输入一个字符串，内有数字和非数字字符，如：ak123x456ccb17960?302gef4563，将其中连续的数字作为一个整体，依次存放到一数组ａ中，例如123放入a［0］,456放入ａ［1］，…　…　。编程统计其共有多少个整数，并输出这些数。

int CountInt（）

// 从键盘输入字符串，连续的数字字符算作一个整数，统计其中整数的个数。

｛int i=0，a[50]；     // 整数存储到数组a，i记整数个数

　scanf（“％c”，&ch）；// 从左到右读入字符串

　while（ch!=‘#’） //‘#’是字符串结束标记

　　if（isdigit（ch））// 是数字字符

　　｛num=0；       // 数初始化

　　　while（isdigit（ch）&& ch!=‘#’）// 拼数

　　　｛num=num＊10+‘ch’-‘０’；

　　　　scanf（“％c”，&ch）；

　　　 ｝

　　　a[i]=num；i++；

　　　if（ch!=‘#’）scanf（“%c”，&ch）； // 若拼数中输入了‘#’，则不再输入

　　 ｝// 结束while（ch！＝‘#’）

　printf（“共有%d个整数，它们是：”i）；

　for（j=0；j<i；j++）

　｛printf（“%6d”，a[j]）；

　 　if（（j+1）％10==0）printf（“\n”）；} // 每10个数输出在一行上

｝// 算法结束

5．编写算法，实现下面函数的功能。函数void insert(char\*s,char\*t,int pos)将字符串t插入到字符串s中，插入位置为pos。假设分配给字符串s的空间足够让字符串t插入。（说明：不得使用任何库函数）

void insert(char \*s,char \*t,int pos)

//将字符串t插入字符串s的第pos个位置。

{int i=1,x=0; char \*p=s,\*q=t; //p，q分别为字符串s和t的工作指针

if(pos<1)

{cout<<“pos参数位置非法”<<endl;exit(0);}

while(\*p!=’\0’&&i<pos)

{p++;i++;} //查pos位置

//若pos小于串s长度，则查到pos位置时，i=pos。

if(\*p == '/0')

{ cout<<pos<<"位置大于字符串s的长度";exit(0);}

else //查找字符串的尾

while(\*p!= '/0')

{p++; i++;} //查到尾时，i为字符‘\0’的下标，p也指向‘\0’。

while(\*q!= '\0')

{q++; x++; } //查找字符串t的长度x，循环结束时q指向'\0'。

for(j=i;j>=pos ;j--)

{\*(p+x)=\*p; p--;} //串s的pos后的子串右移，空出串t的位置。

q--; //指针q回退到串t的最后一个字符

for(j=1;j<=x;j++)

\*p--=\*q--; //将t串插入到s的pos位置上

[算法讨论] 串s的结束标记('\0')也后移了，而串t的结尾标记不应插入到s中。