1. 串

一.判断题

1. KMP算法的特点是在模式匹配时指示主串的指针不会变小。（ ）

2. 空串与空格串相同。（ ）  
3. 串是一种数据对象和操作都特殊的线性表。（ ）

4. 串长度是指串中不同字符的个数。（ ）

5. 字符串“aababaaaba”的next数组的值是-1 0 1 0 1 0 1 2 2 3。（ ）

二.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. 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．-1 0 1 2 3 4 5 6 7 8 8 8 B．-1 0 1 0 1 0 0 0 0 1 0 1

C．-1 0 0 1 2 3 1 1 2 3 4 5 D．-1 0 1 2 -1 0 1 2 1 1 2 3 4

4. 串“ababaaababaa”的next数组为\_\_\_\_\_\_\_\_。

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

5. 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

1. 已知字符串s为”abaabaabacacaabaabcc”，模式串t为”abaabc”，采用KMP算法进行匹配，第一次出现“失配”（s[i]!=t[i]）时,i=j=5，则下次开始匹配时，i和j的值分别是\_\_\_\_\_\_\_\_。  
   A. i=1,j=0 B. i=5,j=0 C. i=5,j=2 D. i=6,j=2  
   已知字符串s为”abacacaabaabcc”，模式串t为”abaabc”，采用KMP算法进行匹配，第一次出现“失配”（s[i]!=t[i]）时,i=j=3，则下次开始匹配时，i和j的值分别是\_\_\_\_\_\_\_\_。  
   A. i=1,j=0 B. i=3,j=0 C. i=3,j=1 D. i=4,j=1 C
2. 设有两个串S1和S2，求S2在S1中首次出现的位置的运算称作\_\_\_\_\_\_\_\_。  
   A. 求子串 B. 判断是否相等 C. 模式匹配 D. 连接
3. 字符串”ababaabab”的next数组为\_\_\_\_\_\_\_\_。  
   A. (0,1,0,1,0,4,1,0,1) B. (0,1,1,2,3,4,2,3,4)  
   C. (0,1,0,1,0,0,0,1,1) D. (0,1,0,1,0,1,0,1,1)
4. 若串S=“myself”，其子串的数目是\_\_\_\_\_\_\_\_。  
   A. 20 B. 21 C. 22 D. 23
5. 在下列表述中，\_\_\_\_\_\_\_\_是错误的。  
   A. 含有一个或多个空格字符的串称为空格串  
   B. 对n(n>0)个顶点的网，求出权最小的n-1条边便可构成其最小代价生成树  
   C. 选择排序算法是不稳定的  
   D. 平衡二叉树的左右子树的结点数之差的绝对值不超过1

三.填空题

1. 串的机内常用存储方式有三种：定长顺序串、 \_\_( )\_\_ 和块链串。
2. 子串的定位又称为串的\_\_( )\_\_。
3. 下列是简单的模式匹配算法程序，请补充完整（注：该函数要求返回子串T在主串S中第pos个字符之后的位置，若不存在，则函数的值为0）。  
   int Index(Sstring \*S, Sstring \*T,int pos) {  
    i=pos-1; j=0;  
    while(i<S->len&&j<T->len) {  
    if( T->ch[i]==S->ch[j] ) { \_\_( )\_\_ ; j++; }   
    else { i= \_\_( )\_\_ ; j=0; }   
    }  
    if(j==T->len) return i-T->len+1;  
    else return 0;  
   }
4. 已知串S=‘abcab’，则其next函数值为\_\_( )\_\_。
5. 两个字符串相等的充分必要条件是\_\_( )\_\_。
6. 组成串的数据元素只能是\_\_( )\_\_。
7. 模式串P=“abaabcac”的next函数值序列为\_\_( )\_\_。
8. 设目标串T=“abccdcdccbaa”，模式P=“cdcc”，按简单模式匹配算法，第\_\_( )\_\_趟匹配成功。
9. 实现字符串拷贝的函数strcpy为：  
   void strcpy(char \*s, char \*t ) /\*copy t to s\*/  
    { while\_\_( )\_\_ ; }
10. 下列程序判断字符串s是否对称，对称则返回1，否则返回0；如f(“abba”)返回1，f(“abab”)返回0。  
    int f(\_\_(1)\_\_)  
     {  
     int i=0,j=0;  
     while(s[j])\_\_(2)\_\_;  
     for(j--; i<j&&s[i]==s[j]; i++,j--);  
     return(\_\_(3)\_\_);  
     }

四. 算法应用题  
1. 给出以S=“aabcbabcaabcaaba”为目标串，T=“abcaaba”为模式串的KMP匹配过程。

2. 子串的定位运算通常称为串的模式匹配，是各种串处理系统中最重要的运算之一，其中KMP匹配算法是效率较高的方法。  
（1）请问KMP匹配算法的基本思想是什么？（2）求字符串“ababaababcabb”的NEXT数组。

五．算法设计题  
1. 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。

2. 函数void insert(char \*s,char \*t,int pos)将字符串t插入字符串s中，插入位置为pos。请用C语言实现该函数。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。）

参考答案：  
一. 判断题：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| √ | X | √ | X | √ |

二． 选择题：

1. 答案：B

2. 答案：B

解释：空格常常是串的字符集合中的一个元素，有一个或多个空格组成的串成为空格串，零个字符的串成为空串，其长度为零。

3. 答案：C

4. 答案：C

5. 答案：B

解释：串中字符的数目称为串的长度

1. 答案：C  
   解析：模式串t”abaabc”的next数组为：-1 0 0 1 1 2，也就是next[5]的值为2，所以：下次开始匹配时，i和j的值分别是i=5,j=2。  
   7. 答案：C
2. 答案：B
3. 答案：C
4. 答案：B

三． 填空题  
1. 堆串  
2. 模式匹配  
3. i++ i-j+1  
4. -1 0 0 0 1 或 0 1 1 1 2  
5. 串的长度相等并且两串对应字符相等  
6. 字符  
7. 01122312 或-10011201  
8. 6  
9. \*s++=\*t++或（\*s++=\*t++）!=’\0’  
10. (1)char s[]（或char \*s） (2)j++ (3)i>=j

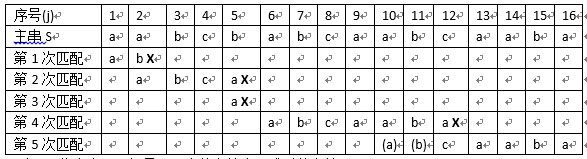
四. 算法应用题

1.（1） 模式串T的next数组：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 下标j | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 字符 | a | b | c | a | a | b | a |
| next[j] | -1 | 0 | 0 | 0 | 1 | 1 | 2 |

（2） KMP快速匹配过程：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 主串 | a | a | b | c | b | a | b | c | a | a | b | c | a | a | b | a |
| 第1趟 | a | b× | next[1]=0 | | | | | | | | | | | | | |
| 第2趟 |  | a | b | c | a× | next[3]=1 | | | | | | | | | | |
| 第3趟 |  |  |  |  | a× | next[0]=-1 | | | | | | | | | | |
| 第4趟 |  |  |  |  |  | a | b | c | a | a | b | a× | next[6]=2 | | | |
| 第5趟 |  |  |  |  |  |  |  |  |  |  |  | c | a | a | b | a |

或  
（1） 模式串T的next数组：  
  
（2） KMP快速匹配过程：  
（注：**X**代表失配，括号（）中的字符表示跳过的字符）  
（3）结果：模式串T在主串S中的位置为10。

2. （1）KMP匹配算法的基本思想是：当模式串字符和主串字符失配时，指向主串的指针不动，而是改变指向模式串指针的指向，使指向模式串的指针尽可能指向模式串靠右的字符。  
（2）0112342345123

五. 算法设计题  
1. 判断字符串t是否是字符串s的子串，称为串的模式匹配，其基本思想是对串s和t各设一个指针i和j,i的值域是0..m-n，j的值域是0..n-1。初始值i和j均为0。模式匹配从s0和t0开始，若s0=t0，则i和j指针增加1，若在某个位置Si!=tj，则主串指针i回溯到i=i-j+1，j仍从0开始，进行下一轮的比较，直到匹配成功（j>n-1），返回子串在主串的位置（i-j）。否则，当i>m-n则为匹配失败。核心语句段如下：

while(i<=m-n && j<=n-1)

if(s[i]==t[j]) { i++; j++; }

else { i=i-j+1; j=0; }

if(i<=m-n && j==n ) { cout<<”t串在s串中位置：”<<i-n+1; } //匹配成功

else return(0); //匹配失败

1. 首先应查找字符串s的pos位置，将第pos个字符到字符串s尾的子串向后移动字符串t的长度，然后将字符串t复制到字符串s的第pos位置后。对插入位置pos要验证其合法性，题目假设给字符串s的空间足够大，故对插入不必判溢出。核心语句段如下：

while(\*p!=’\0’&& i<pos) {p++;i++;}

if(\*p==’\0’) {cout<<”pos位置大于字符串s的长度”<<endl; exit(0);}  
 else

while(\*p!=’\0’) {p++;i++;}

while(\*q!=’\0’) {q++;x++;}

for(j=i;j>=pos;j--) {\*(p+x)=\*p;p--;}

q--;

for(j=1;j<=x;j++) \*p--=\*q--;