//串的定长顺序存储表示  
 #define MAXSTRLEN 40 // 用户可在255以内定义最大串长（1个字节）  
 typedef char SString[MAXSTRLEN+1]; // 0号单元存放串的长度  
  
//串采用定长顺序存储结构的基本操作(14个)  
 // SString是数组，故不需引用类型。此基本操作包括算法4.2,4.3,4.5  
 Status StrAssign(SString T,char \*chars)  
 { // 生成一个其值等于chars的串T  
   int i;  
   if(strlen(chars)>MAXSTRLEN)  
     return ERROR;  
   else  
   {  
     T[0]=strlen(chars);  
     for(i=1;i<=T[0];i++)  
       T[i]=\*(chars+i-1);  
     return OK;  
   }  
 }  
  
 Status StrCopy(SString T,SString S)  
 { // 由串S复制得串T  
   int i;  
   for(i=0;i<=S[0];i++)  
     T[i]=S[i];  
   return OK;  
 }  
  
 Status StrEmpty(SString S)  
 { // 若S为空串,则返回TRUE,否则返回FALSE  
   if(S[0]==0)  
     return TRUE;  
   else  
     return FALSE;  
 }  
  
 int StrCompare(SString S,SString T)  
 { // 初始条件: 串S和T存在  
   // 操作结果: 若S>T,则返回值>0;若S=T,则返回值=0;若S<T,则返回值<0  
   int i;  
   for(i=1;i<=S[0]&&i<=T[0];++i)  
     if(S[i]!=T[i])  
       return S[i]-T[i];  
   return S[0]-T[0];  
 }  
  
 int StrLength(SString S)  
 { // 返回串的元素个数  
   return S[0];  
 }  
  
 Status ClearString(SString S)  
 { // 初始条件:串S存在。操作结果:将S清为空串  
   S[0]=0;// 令串长为零  
   return OK;  
 }  
  
 Status Concat(SString T,SString S1,SString S2) // 算法4.2改  
 { // 用T返回S1和S2联接而成的新串。若未截断，则返回TRUE，否则FALSE  
   int i;  
   if(S1[0]+S2[0]<=MAXSTRLEN)  
   { // 未截断  
     for(i=1;i<=S1[0];i++)  
       T[i]=S1[i];  
     for(i=1;i<=S2[0];i++)  
       T[S1[0]+i]=S2[i];  
     T[0]=S1[0]+S2[0];  
     return TRUE;  
   }  
   else  
   { // 截断S2  
     for(i=1;i<=S1[0];i++)  
       T[i]=S1[i];  
     for(i=1;i<=MAXSTRLEN-S1[0];i++)  
       T[S1[0]+i]=S2[i];  
     T[0]=MAXSTRLEN;  
     return FALSE;  
   }  
 }  
  
 Status SubString(SString Sub,SString S,int pos,int len)  
 { // 用Sub返回串S的第pos个字符起长度为len的子串。算法4.3  
   int i;  
   if(pos<1||pos>S[0]||len<0||len>S[0]-pos+1)  
     return ERROR;  
   for(i=1;i<=len;i++)  
     Sub[i]=S[pos+i-1];  
   Sub[0]=len;  
   return OK;  
 }  
  
 int Index(SString S,SString T,int pos)  
 { // 返回子串T在主串S中第pos个字符之后的位置。若不存在,则函数值为0。  
   // 其中,T非空,1≤pos≤StrLength(S)。算法4.5  
   int i,j;  
   if(1<=pos&&pos<=S[0])  
   {  
     i=pos;  
     j=1;  
     while(i<=S[0]&&j<=T[0])  
       if(S[i]==T[j]) // 继续比较后继字符  
       {  
         ++i;  
         ++j;  
       }  
       else // 指针后退重新开始匹配  
       {  
     i=i-j+2;  
         j=1;  
       }  
     if(j>T[0])  
       return i-T[0];  
     else  
       return 0;  
   }  
   else  
     return 0;  
 }  
  
 Status StrInsert(SString S,int pos,SString T)  
 { // 初始条件: 串S和T存在,1≤pos≤StrLength(S)+1  
   // 操作结果: 在串S的第pos个字符之前插入串T。完全插入返回TRUE,部分插入返回FALSE  
   int i;  
   if(pos<1||pos>S[0]+1)  
     return ERROR;  
   if(S[0]+T[0]<=MAXSTRLEN)  
   { // 完全插入  
     for(i=S[0];i>=pos;i--)  
       S[i+T[0]]=S[i];  
     for(i=pos;i<pos+T[0];i++)  
       S[i]=T[i-pos+1];  
     S[0]=S[0]+T[0];  
     return TRUE;  
   }  
   else  
   { // 部分插入  
     for(i=MAXSTRLEN;i<=pos;i--)  
       S[i]=S[i-T[0]];  
     for(i=pos;i<pos+T[0];i++)  
       S[i]=T[i-pos+1];  
     S[0]=MAXSTRLEN;  
     return FALSE;  
   }  
 }  
  
 Status StrDelete(SString S,int pos,int len)  
 { // 初始条件: 串S存在,1≤pos≤StrLength(S)-len+1  
   // 操作结果: 从串S中删除第pos个字符起长度为len的子串  
   int i;  
   if(pos<1||pos>S[0]-len+1||len<0)  
     return ERROR;  
   for(i=pos+len;i<=S[0];i++)  
     S[i-len]=S[i];  
   S[0]-=len;  
   return OK;  
 }  
  
 Status Replace(SString S,SString T,SString V)  
 { // 初始条件: 串S,T和V存在,T是非空串（此函数与串的存储结构无关）  
   // 操作结果: 用V替换主串S中出现的所有与T相等的不重叠的子串  
   int i=1; // 从串S的第一个字符起查找串T  
   if(StrEmpty(T)) // T是空串  
     return ERROR;  
   do  
   {  
     i=Index(S,T,i); // 结果i为从上一个i之后找到的子串T的位置  
     if(i) // 串S中存在串T  
     {  
       StrDelete(S,i,StrLength(T)); // 删除该串T  
       StrInsert(S,i,V); // 在原串T的位置插入串V  
       i+=StrLength(V); // 在插入的串V后面继续查找串T  
     }  
   }while(i);  
   return OK;  
 }  
  
 void DestroyString()  
 { // 由于SString是定长类型,无法销毁  
 }  
  
 void StrPrint(SString T)  
 { // 输出字符串T。另加  
   int i;  
   for(i=1;i<=T[0];i++)  
     printf("%c",T[i]);  
   printf("\n");  
 }