```
#include"标头.h"
/*
2017-7-26 19:48:34
比较 串 和 单链表 两种数据结构参数的不同点
串的存储单元是刚好足够的，不定长的
单链表的存储单元是定长的，可能有剩余
从而，二者的初始化和录入数据函数存在差异，应当注意！
*/
typedef struct
{
    char* ch;
    int length;
}string;
Status input_data(string* S,char* ch)
{
    int j;
    if (!S->ch)
    free(S->ch);
    int i = strlen(ch);// i = 串长
    if (i == 0)
        return error;//空串
    else
    {
        S->ch = (char*)malloc(sizeof(char)*i);
        //分配与字符个数等长的存储单元
        if (S->ch == NULL) exit(-1);
        for (j = 0; j <= i - 1; j++)
        {
            S->ch[j] = ch[j];
        }//拷贝字符串
        S->length = i;//更新串长参数
        return ok;
    }
}
/* 初始条件:串S存在。操作结果: 由串S复制得串T */
Status copy(string S, string* T)
{
    int i;
    if (!T->ch)
        free(T->ch);//防止传出接收者不是新串
    T->ch = (char*)malloc(sizeof(char)*S.length);
    if (!T->ch) exit(-1);
    for (i = 0; i <= S.length - 1; i++)
    {
        T->ch[i] = S.ch[i];//拷贝串
    }
    T->length = S.length;//更新参数
    return ok;
}
/* 初始条件: 串S存在。操作结果: 若S为空串,则返回TRUE,否则返回FALSE */
Status is_empty(string S)
{
    return S.length == 0&&S.ch==NULL ? true : false;
}
/* 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 compare_two_len(string S, string T)
{
    int i = 0;
    while (i <= S.length - 1 && i <= T.length)
    {
        if (S.ch[i] != T.ch[i])
            return S.ch[i] - T.ch[i];
```

```
63          i++;
64      }
65      return S.length - T.length;//返回0时说明串相同
66  }
67  /* 将S清为空串 */
68  void clear(string S)
69  {
70      free(S.ch);
71      S.ch = NULL;
72      S.length = 0;
73  }
74  /* 用sub返回串S的第pos个字符起长度为len的子串*/
75  Status sub_str(string S, string* sub,int pos, int len)
76  {
77      int j;
78      if (sub->ch)
79          free(sub->ch);
80      int i = pos + len - 1;
81      if (i <= S.length)
82      {
83          sub->ch = (char*)malloc(sizeof(char)*len);
84          if (!sub->ch) exit(-1);
85          for (j = 0; j <= len - 1; j++)
86          {
87              sub->ch[j] = S.ch[pos - 1 + j];
88          }
89          sub->length = len;
90          return ok;
91      }
92      else
93          return error;
94  }
95  /* 初始化(产生空串)字符串T*/
96  void init_str(string S)
97  {
98      S.ch = NULL;
99      S.length = 0;
100 }
101 /* 返回子串T在主串S中第一次出现的位置,否则返回0*/
102 int index(string S, string T)
103 {
104
105 }
106 /*在串S的第pos个字符之前插入串T*/
107 Status insert_str(string* S, string T, int pos)
108 {
109     int i;
110     if (pos >= 1 && pos <= (*S).length + 1)
111     {
112         (*S).ch = (char*)realloc((*S).ch, sizeof(char)*((*S).length + T.length));
113         if (!(*S).ch) exit(-1);
114         for (i = S->length - 1; i >= pos - 1; i--)
115         {
116             S->ch[i + T.length] = S->ch[i];
117         }
118         for (i = 0; i <= T.length - 1; i++)
119         {
120             S->ch[pos-1+i] = T.ch[i];
121         }
122         (*S).length += T.length;
123         printf("串S插入串T后的长度是 %d \n", (*S).length);
124         return ok;
```

```
125     }
126         return error;
127 }
128 /*输出串*/
129 printf_str(string S)
130 {
131     int i = 0;
132     while (i <= S.length - 1)
133     {
134         printf("%c", S.ch[i]);
135         i++;
136     }
137     printf("\n");
138 }
139 int main()
140 {
141     string T;
142     string S;
143     string sub;
144     sub.ch = NULL;
145     sub.length = 0;
146     char*ch = "ABCD";
147     input_data(&S, ch);
148     printf("输出字符串 S:\n");
149     printf_str(S);
150     copy(S, &T);
151     printf("复制串S得到的串T：\n");
152     printf_str(T);
153     if (is_empty(T))
154         printf("串T是空串！\n");
155     else
156         printf("串T不是空串!\n");
157     if (compare_two_len(S, T))
158         printf("串S和串T不同！\n");
159     else
160         printf("串S和串T相同!\n");
161     printf("取出串S第5至第10个字符组成的子串:\n");
162     sub_str(S, &sub, 10, 10);
163     printf_str(sub);
164     printf("在串S的第5个字符前插入串T，插入后串S：\n");
165     insert_str(&S, T, 5);
166     printf_str(S);
167     getchar();
168     return 0;
169 }
```