# 实验四 串

## 【实验目的】

1、掌握串的存储表示及基本操作；

2、掌握串的两种模式匹配算法：BF和KMP。

3、了解串的应用。

## 【实验学时】

2学时

## 【实验预习】

回答以下问题：

1、串和子串的定义

串的定义： 串是零个或多个字符组成的有限序列。

子串的定义： 一个串中任意个连续字符组成的子序列（含空串，但不含串本身）

2、串的模式匹配

串的模式匹配即子串定位是一种重要的串运算。定义：

## 【实验内容和要求】

1、按照要求完成程序exp4\_1.c，实现串的相关操作。调试并运行如下测试数据给出运行结果：

* **求“This is a boy”的串长；**
* **比较”abc3”和“abcde“；表示空格**
* **比较”english”和“student“；**
* **比较”abc”和“abc“；**
* **截取串”white”，起始2，长度2；**
* **截取串”white”，起始1，长度7；**
* **截取串”white”，起始6，长度2；**
* **连接串”asddffgh”和”12344”；**

#include<stdio.h>

#include<string.h>

#define MAXSIZE 100

#define ERROR 0

#define OK 1

/\*串的定长顺序存储表示\*/

typedef struct

{

char data[MAXSIZE];

int length;

} SqString;

int strInit(SqString \*s); /\*初始化串\*/

int strCreate(SqString \*s); /\*生成一个串\*/

int strLength(SqString \*s); /\*求串的长度\*/

int strCompare(SqString \*s1,SqString \*s2); /\*两个串的比较\*/

int subString(SqString \*sub,SqString \*s,int pos,int len); /\*求子串\*/

int strConcat(SqString \*t,SqString \*s1,SqString \*s2); /\*两个串的连接\*/

/\*初始化串\*/

int strInit(SqString \*s)

{

s->length=0;

s->data[0]='\0';

return OK;

}/\*strInit\*/

/\*生成一个串\*/

int strCreate(SqString \*s)

{

printf("input string :");

gets(s->data);

s->length=strlen(s->data);

return OK;

}/\*strCreate\*/

/\*（1）---求串的长度\*/

int strLength(SqString \*s)

{

return s->length;

}/\*strLength\*/

/\*（2）---两个串的比较，S1>S2返回>0，s1<s2返回<0，s1==s2返回0\*/

int strCompare(SqString \*s1,SqString \*s2)

{

int i;

for(i=0;i<s1->length&&i<s2->length;i++)

{

if(s1->data[i]>s2->data[i])

{

return 1;

}

if(s1->data[i]<s2->data[i])

{

return -1;

}

}

return 0;

}/\*strCompare\*/

/\*（3）---求子串，sub为返回的子串，pos为子串的起始位置，len为子串的长度\*/

int subString(SqString \*sub,SqString \*s,int pos,int len)

{

int i;

if(pos<1||pos>s->length||len<0||len>s->length-pos+1)

{

return ERROR;

}

sub->length=0;

for(i=0;i<len;i++)

{

sub->data[i]=s->data[i+pos-1];

sub->length++;

}

sub->data[i]='\0';

return OK;

}/\*subString\*/

/\*（4）---两个串连接，s2连接在s1后，连接后的结果串放在t中\*/

int strConcat(SqString \*t,SqString \*s1,SqString \*s2)

{

int i=0,j=0;

while(i<s1->length)

{

t->data[i]=s1->data[i];

i++;

}

while(j<s2->length)

{

t->data[i++]=s2->data[j++];

}

t->data[i]='\0';

t->length=s1->length+s2->length;

return OK;

}/\*strConcat\*/

int main()

{

int n,k,pos,len;

SqString s,t,x;

do

{

printf("\n ---String--- \n");

printf(" 1. strLentgh\n");

printf(" 2. strCompare\n");

printf(" 3. subString\n");

printf(" 4. strConcat\n");

printf(" 0. EXIT\n");

printf("\n ---String---\n");

printf("\ninput choice:");

scanf("%d",&n);

getchar();

switch(n)

{

case 1:

printf("\n\*\*\*show strLength\*\*\*\n");

strCreate(&s);

printf("strLength is %d\n",strLength(&s));

break;

case 2:

printf("\n\*\*\*show strCompare\*\*\*\n");

strCreate(&s);

strCreate(&t);

k=strCompare(&s,&t); /\*（5）---调用串比较函数比较s，t\*/

if(k==0)

printf("two string equal!\n");

else if(k<0)

printf("first string<second string!\n");

else

printf("first string>second string!\n");

break;

case 3:

printf("\n\*\*\*show subString\*\*\*\n");

strCreate(&s);

printf("input substring pos,len:");

scanf("%d,%d",&pos,&len);

if(subString(&t,&s,pos,len))

printf("subString is %s\n",t.data);

else

printf("pos or len ERROR!\n");

break;

case 4:

printf("\n\*\*\*show subConcat\*\*\*\n");

strCreate(&s);

strCreate(&t);

if(strConcat(&x,&s,&t)) /\*（6）---调用串连接函数连接s&t\*/

printf("Concat string is %s",x.data);

else

printf("Concat ERROR!\n");

break;

case 0:

exit(0);

default:

break;

}

}

while(n);

return 0;

}2、按照要求完成程序exp4\_2.c，实现BF串的模式匹配算法。调试及测试数据并给出结果：

* **应用BF算法求子串”JING”在主串”BEIJING”中的位置，测试起始位置分别为1和5的情况；**

exp4\_2.c部分代码如下：

#include<stdio.h>

#include<string.h>

#define MAXSIZE 100

#define ERROR 0

#define OK 1

/\*串的定长顺序存储表示\*/

typedef struct

{

char data[MAXSIZE];

int length;

} SqString;

int strCreate(SqString \*s);

int indexBf(SqString \*s,SqString \*t,int pos); /\*串的模式匹配BF\*/

/\*生成一个串\*/

int strCreate(SqString \*s)

{

printf("input string :");

gets(s->data);

s->length=strlen(s->data);

return OK;

}/\*strCreate\*/

/\*---串的模式匹配BF\*/

int indexBf(SqString \*s,SqString \*t,int pos)

{

int i=pos-1,j=0;

while(i<s->length&&j<t->length)

{

if(s->data[i]==t->data[j])

{

i++ ;

j++;

;

}

else

{

i=i-j+1 ;

j=0;

}

}

if(j>=t->length)

{

return i-t->length+1;

}

else

{

return 0;

}

}/\*index\_bf\*/

/\*（2）---KMP求next值\*/

void getNext(SqString \*t,int next[])

{

int i=0,j=-1;

next[0]=-1;

while(i<t->length)

{

if((j==-1)||(t->data[i]==t->data[j]))

{

j++;

i++;

next[i]=j;

}

else

{

j=next[j];

}

}

}/\*getNext\*/

/\*（3）---KMP模式匹配\*/

int indexKmp(SqString \*s,SqString \*t,int start,int next[])

{

int i=start-1,j=0;

while(i<s->length&&j<t->length)

{

if(j==-1||s->data[i]==t->data[j])

{

i++;

j++;

}

else

{

j=next[j];

}

}

if(j>=t->length)

{

return i-t->length+1;

}

else

{

return 0;

}

}/\*index\_kmp\*/

int main()

{

int n,i,pos,next[MAXSIZE];

SqString s,t;

do

{

printf("\n ---String--- \n");

printf(" 1. Index\_BF\n");

printf(" 2. INdex\_KMP\n");

printf(" 0. EXIT\n");

printf("\n ---String---\n");

printf("\ninput choice:");

scanf("%d",&n);

getchar();

switch(n)

{

case 1:

printf("\n\*\*\*show Index\_BF\*\*\*\n");

printf(" s:");

strCreate(&s);

printf(" t:");

strCreate(&t);

printf("input start position:");

scanf("%d",&pos);

printf("BF:index is %d\n",indexBf(&s,&t,pos));

break;

case 2:

printf("\n\*\*\*show Index\_KMP\*\*\*\n");

printf(" s:");

strCreate(&s);

printf(" t:");

strCreate(&t);

printf("input start position:");

scanf("%d",&pos);

getNext(&t,next);

printf("KMP:\n");

printf("next[]:");

for(i=0; i<t.length; i++)

printf("%3d",next[i]+1);

printf("\n");

printf("index is %d\n",indexKmp(&s,&t,pos,next));

break;

case 0:

exit(0);

default:

break;

}

}

while(n);

return 0;

}

## 【实验小结】

















