南京信息工程大学 数据结构 | 实验(实习)报告

实验（实习）名称 串的应用 日期 2024.11. 得分 指导教师

学院 计院 专业 计科

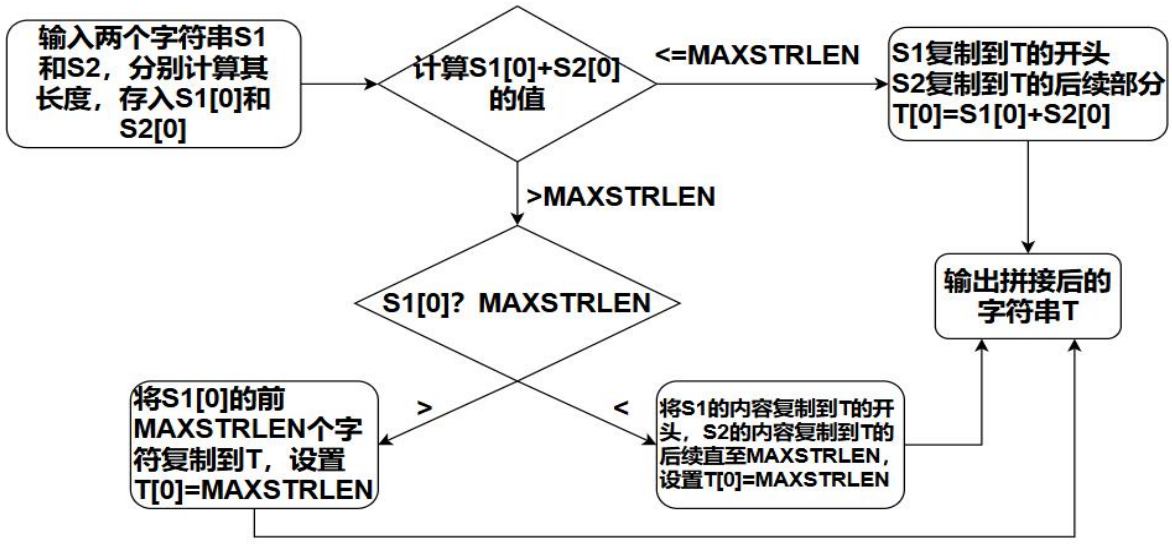
一、实验目的

1、了解串的三种存储结构；
2、掌握串的定长顺序存储结构；
3、掌握串的简单模式匹配算法并分析该算法的时间复杂度。

二、 实验内容与步骤

1、采用定长顺序存储结构（数据类型定义在教材 P73 页）存储两个长度不超过 255 的字符串，将上述两个字符串联接起来作为主串 S。（即实现教材中算法 4.2）

（1）（在电脑里）画出本题算法的流程图

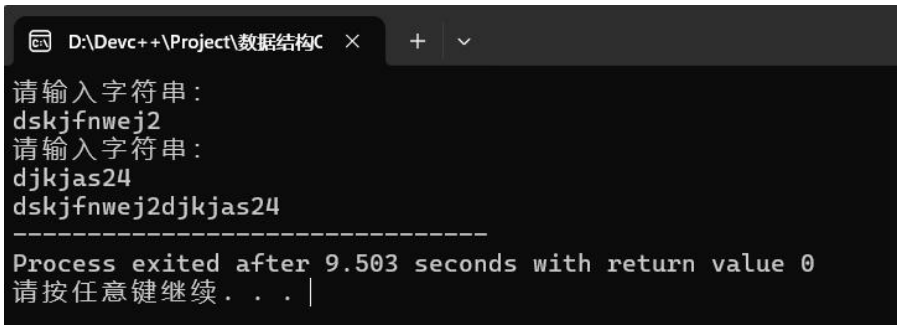


（2）实现程序源代码（文本）及运行结果（截图）

```
#include<iostream>
#include<cstring>
using namespace std;
#define MAXSTRLEN 255
#define OK 1
#define ERROR 0
typedef unsigned char SString[MAXSTRLEN+1];//0 号存放串的长度
typedef int Status;
Status Concat(SString &T,SString S1,SString S2){
    int i;
    if(S1[0]+S2[0]<=MAXSTRLEN){
        for(int i=1;i<=S1[0];i++)
        {
            T[i]=S1[i];
```

```
            }
            for(i=S1[0]+1;i<=S1[0]+S2[0];i++)
            {
                T[i]=S2[i-S1[0]];
            }
            T[0]=S1[0]+S2[0];
            return OK;
        }
        else if(S1[0]<MAXSTRLEN){
            for(i=1;i<=S1[0];i++)
            {
                T[i]=S1[i];
            }
            for(i=S1[0]+1;i<=MAXSTRLEN;i++)
            {
                T[i]=S2[i-S1[0]];
            }
            T[0]=MAXSTRLEN;
            return ERROR;
        }
        else{
            for(i=1;i<=MAXSTRLEN;i++)
            {
                T[i]=S1[i];
            }
            T[0]=MAXSTRLEN;
            return ERROR;
        }
}
void InputString(SString S)
{
        string temp;
        cout<<"请输入字符串："<<endl;
        getline(cin,temp);
        int len;
        // 截取至 MAXSTRLEN 长度
        if(temp.length()>MAXSTRLEN) len=MAXSTRLEN;
        else len=temp.length();
        S[0]=len;
        for(int i=1;i<=S[0];i++)
        {
            S[i]=temp[i-1];
        }
}
```

```
int main()
{
	SString s1,s2,T;
	InputString(s1);
	InputString(s2);
	Concat(T,s1,s2);
	for(int i=1;i<=T[0];i++)
	{
		cout<<T[i];
	}
	return 0;
}
```

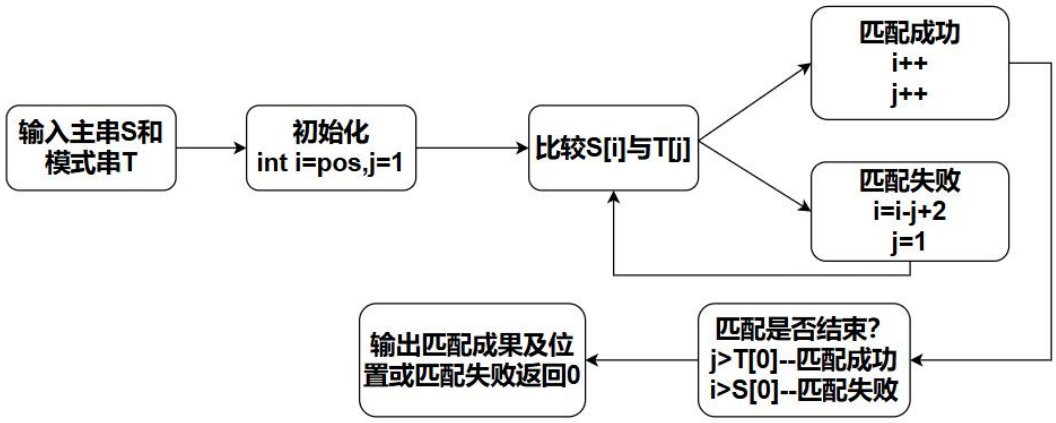


（3）分析本题算法的时间复杂度

此算法的时间复杂度主要受输入字符串的长度限制，其复杂度为 O（n+m），其中 n 和 m 分别为两个输入字符串的长度。由于算法的操作量不会超过定义的最大长度 MAXSTRLEN，即使拼接长度达到最大 MAXSTRLEN，复杂度仍为 O（MAXSTRLEN）。

2、采用定长顺序存储结构存储一个长度不超过 255 的字符串作为模式子串 T，利用简单模式匹配算法，从主串 S 的第 pos 个字符起和模式串 T 比较，如果模式串中的每个字符都和主串 S 中的字符序列匹配成功，则返回主串的 pos；如果匹配失败，则返回 0。（即实现教材中算法 4.5）

（1）（在电脑里）画出本题算法的流程图



（2）实现程序源代码（文本）及运行结果（截图）

```
#include<iostream>
```

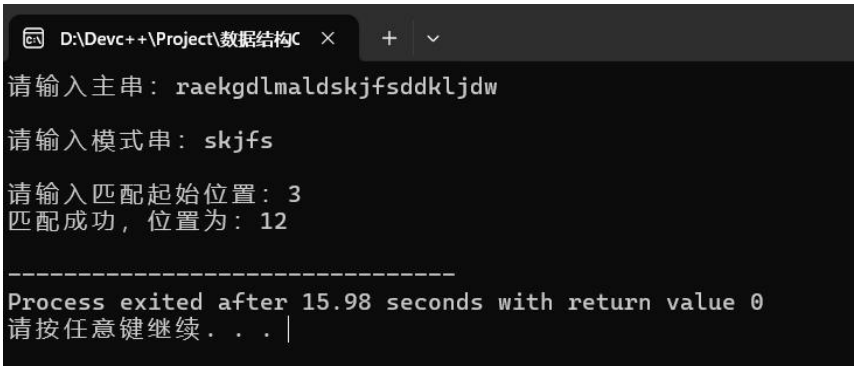
```
#include<cstring>
using namespace std;

#define MAXSTRLEN 255

typedef unsigned char SString[MAXSTRLEN+1];//0 号存放串的长度
typedef int Status;

void InputString(SString S)
{
   string temp;
   getline(cin,temp);
   int len;
   // 截取至 MAXSTRLEN 长度
   if(temp.length()>MAXSTRLEN) len=MAXSTRLEN;
   else len=temp.length();
   S[0]=len;
   for(int i=1;i<=S[0];i++)
   {
        S[i]=temp[i-1];
   }
}
int Index(SString S,SString T,int pos){
   int i=pos;
   int j=1;
   while(i<=S[0]&&j<=T[0])
   {
        if(S[i]==T[j])
        {
             i++;
             j++;
        }
        //指针后退开始重新匹配
        else{
             i=i-j+2;
             j=1;
        }
   }
   if(j>T[0]){
        return i-T[0];
   }
   else return 0;
}
int main()
```

```
{
    SString T,S;
    cout<<"请输入主串：";
    InputString(S);
    cout<<endl;
    cout<<"请输入模式串：";
    InputString(T);
    cout<<endl;
    int pos;
    cout<<"请输入匹配起始位置：";
    cin>>pos;
    int result=Index(S,T,pos);
    if(result){
        cout<<"匹配成功，位置为："<<result<<endl;
    }
    else cout<<"匹配失败"<<endl;
    return 0;
}
```



（4）分析本题算法的时间复杂度

在最坏情况下，当主串 S 和模式串 T 的内容接近但并不匹配时，算法可能会反复比较每一个字符。每次比较失败后，算法会将主串的位置 i 向后移动一位，并重新从模式串的第一个字符开始比较。因此，最坏情况下，时间复杂度为 O(n×m)，其中 n 是主串的长度，m 是模式串的长度。

三、 实验心得（必写）

本次实验中，通过实现串的定长顺序存储结构和 BF 算法，我对字符串的基本操作以及简单模式匹配算法有了更深入的理解。BF 算法虽然逻辑简单，但在大规模字符串匹配中效率较低。尤其在最坏情况下，其时间复杂度为 O(n×m)，随着主串和模式串的长度增加，算法性能会显著下降。通过本次实验，我不仅掌握了字符串的存储结构和基本操作，还了解到优化算法的重要性和必要性。