

软件学院设计性实验报告

学院：软件学院 专业： 智能应用 年级/班级： 大二/2 班
2024—2025 学年第一学期

课程名称	数据结构	指导教师	张磊
学号姓名	2328624035 李陈歆烨		
实验地点	向真楼 403	实验时间	11 月 13 日
项目名称	字符串的模式匹配	实验类型	综合性

一、实验目的利用串的定长顺序存储，

```
#define MAXLEN 255 //串最大长度
typedef struct
{ char ch[MAXLEN+1]; //存储串的一维数组,0 号单元空着
  int length; //串的当前长度
}SString
1) 编写完成下列功能的函数：(1) 创建一个串；(2) 实现 BF 模式匹配算法
   (3) 实现 KMP 模式匹配算法；(4) 调用创建串函数创建主串和模式串；(5)
   调用 BF 算法输出匹配结果；(6) 调用 KMP 算法输出匹配结果；
2) 用主函数调用你所编写的函数，并在每一步后有适当的输出，以验证你
   编程程序的正确性。
```

二、实验仪器或设备

学院提供公共机房，1 台微型计算机/学生

三、实验说明（设计方案）

```
利用串的定长顺序存储，
(1) 创建一个串；(2) 实现 BF 模式匹配算法；(3) 实现 KMP 模式匹配算法；(4) 调用
创建串函数创建主串和模式串；(5) 调用 BF 算法输出匹配结果；(6) 调用 KMP 算法输出
匹配结果；
2) 用主函数调用你所编写的函数，并在每一步后有适当的输出，以验证你编程程序的正
确性。
```

四、实验步骤（包括主要步骤、代码分析等）

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define MAXLEN 255 //串最大长度
// 串的定长顺序存储结构定义
typedef struct
{
    char ch[MAXLEN + 1]; //存储串的一维数组,0 号单元空着
```

```

        int length;                //串的当前长度
    } SString;

// 1. 创建一个串的函数
SString* createString(const char* str)
{
    SString* s = (SString*)malloc(sizeof(SString));
    if (s == NULL)
    {
        printf("内存分配失败! \n");
        return NULL;
    }
    int len = strlen(str);
    if (len > MAXLEN)
    {
        printf("输入字符串过长, 超出最大长度限制! \n");
        free(s);
        return NULL;
    }
    s->length = len;
    for (int i = 0; i < len; i++)
    {
        s->ch[i + 1] = str[i];
    }
    return s;
}

```

```

// 2. BF 模式匹配算法
int BF(SString* S, SString* T)
{
    int i = 1, j = 1;
    while (i <= S->length && j <= T->length)
    {
        if (S->ch[i] == T->ch[j])
        {
            i++;
            j++;
        }
        else
        {
            i = i - j + 2;
            j = 1;
        }
    }
}

```

```

        if (j > T->length)
            return i - T->length;
        return 0;
    }

// 3. 计算 next 数组，用于 KMP 算法
void getNext(SSString* T, int next[])
{
    int i = 1, j = 0;
    next[1] = 0;
    while (i < T->length)
    {
        if (j == 0 || T->ch[i + 1] == T->ch[j + 1])
        {
            i++;
            j++;
            next[i] = j;
        }
        else
            j = next[j];
    }
}

```

```

// 3. KMP 模式匹配算法
int KMP(SSString* S, SString* T)
{
    int next[T->length + 1];
    getNext(T, next);
    int i = 1, j = 1;
    while (i <= S->length && j <= T->length)
    {
        if (j == 0 || S->ch[i] == T->ch[j])
        {
            i++;
            j++;
        }
        else
            j = next[j];
    }
    if (j > T->length)
        return i - T->length;
    return 0;
}

```

```

int main()
{
    char mainStr[MAXLEN];
    char patternStr[MAXLEN];
    printf("请输入主串内容: ");
    scanf("%s", mainStr);
    printf("请输入模式串内容: ");
    scanf("%s", patternStr);

    // 4. 调用创建串函数创建主串和模式串
    SString* S = createString(mainStr);
    SString* T = createString(patternStr);
    if (S == NULL || T == NULL)
    {
        printf("串创建失败, 请检查输入! \n");
        return 1;
    }

    // 5. 调用 BF 算法输出匹配结果
    int bfIndex = BF(S, T);
    if (bfIndex > 0)
    {
        printf("BF 算法匹配成功, 模式串在主串中的起始位置为: %d\n",
bfIndex);
    }
    else
    {
        printf("BF 算法匹配失败, 主串中未找到模式串! \n");
    }

    // 6. 调用 KMP 算法输出匹配结果
    int kmpIndex = KMP(S, T);
    if (kmpIndex > 0)
    {
        printf("KMP 算法匹配成功, 模式串在主串中的起始位置为: %d\n",
kmpIndex);
    }
    else
    {
        printf("KMP 算法匹配失败, 主串中未找到模式串! \n");
    }

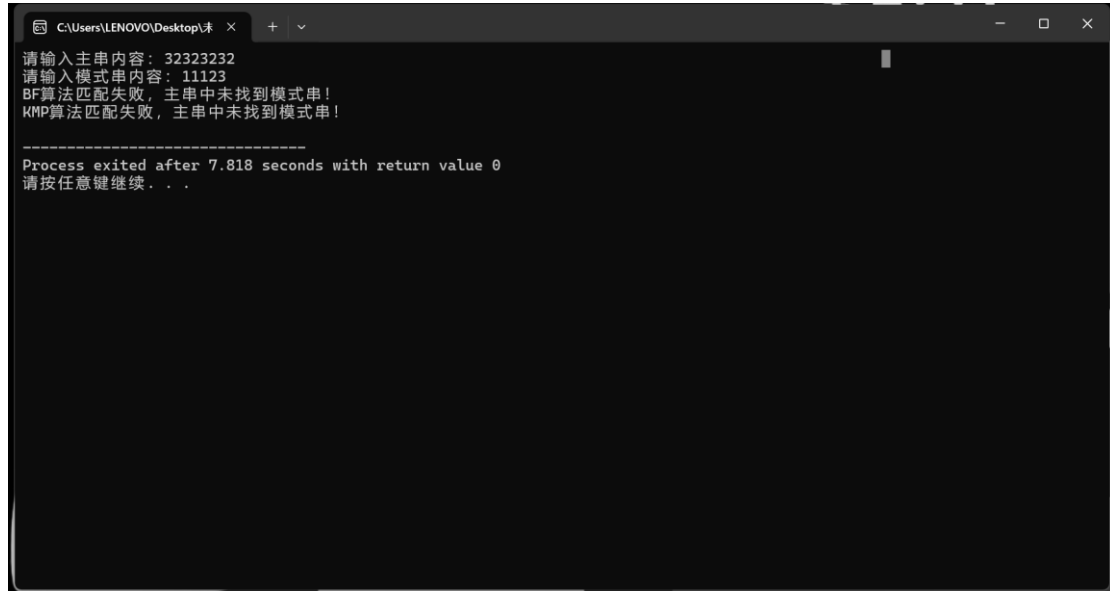
    free(S);
    free(T);
}

```

```
    return 0;  
}
```

五、结果分析与总结

(可参考以下内容：)



```
C:\Users\LENOVO\Desktop\未命名 >
请输入主串内容: 32323232
请输入模式串内容: 11123
BF算法匹配失败, 主串中未找到模式串!
KMP算法匹配失败, 主串中未找到模式串!

-----
Process exited after 7.818 seconds with return value 0
请按任意键继续...
```

根据用户页面提示输入模式串 主串所有元素, 结果正确

2024 年 10 月 22 日

实验报告格式说明:

1. 页脚插入页码: 宋体小五号, 居中
2. 课程名称标题: 黑体四号加粗
3. 表格内容: 宋体五号
4. 标题: 黑体小四
5. 正文: 宋体五号
6. 页面设置: 纸张A4, 页边距上下2cm, 左右2.8cm