		Description
4	串	串的定义和实现 串的模式匹配

```c
//定长顺序存储表示
#define MAXLEN 255//预定义最大串长为255
typedef struct {
    char ch[MAXLEN];//每个分量存储一个字符
    int length;//串的实际长度
} SString;

//堆分配存储表示
typedef struct {
    char *ch;//按串长分配存储区，ch指向串的基地址
    int length;//串的长度
} HString;

int Index(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;//继续比较后续字符
        } else {
            i = i - j + 2;
            j = 1;//指针后退重新开始匹配
        }
    }
    if (j > T.length)return i - T.length;
    else return 0;
}

void get_next(SString T, int next[]) {
    int i = 1, j = 0;
    next[1] = 0;
    while (i < T.length) {
        if (j == 0 || T.ch[i] == T.ch[j]) {
            ++i;
            ++j;
            next[i] = j;//若pi=pj，则next[j+1]=next[j]+1
        } else
            j = next[j];//否则令j=next[j],循环继续
    }
}

int Index_KMP(SString S, SString T, int next[]) {
    int i = 1, j = 1;
    while (i < S.length && j <= T.length) {
        if (j == 0 || S.ch[i] == T.ch[j]) {
            ++i;
            ++j;//继续比较后继字符
        } else
            j = next[j];//模式串向右移动
```

```
    }
    if (j > T.length)
        return i - T.length;//匹配成功
    else
        return 0;
}

void get_nextval(SString T, int nextval[]) {
    int i = 1, j = 0;
    nextval[1] = 0;
    while (i < T.length) {
        if (j == 0 || T.ch[i] == T.ch[j]) {
            ++i;
            ++j;
            if (T.ch[i] != T.ch[j])nextval[i] = j;
            else nextval[i] = nextval[j];
        } else
            j = nextval[j];
    }
}
```