

串

一、串的定义和基本操作

• 串的定义

串，即字符串(String)是由零个或多个字符组成的有限序列。一般记为 $S = 'a_1a_2\dots\dots a_n'$ ($n \geq 0$)；其中， S 是串名，单引号括起来的字符序列是串的值； a_i 可以是字母、数字或其他字符；串中字符的个数 n 称为串的长度； $n = 0$ 时的串称为空串（用 \emptyset 表示）

串的定义

串，即字符串（String）是由零个或多个字符组成的有限序列。一般记为
 $S = 'a_1a_2\dots\dots a_n'$ ($n \geq 0$)

其中， S 是串名，单引号括起来的字符序列是串的值； a_i 可以是字母、数字或其他字符；串中字符的个数 n 称为串的长度。 $n = 0$ 时的串称为空串（用 \emptyset 表示）。

例：

$S = "HelloWorld!"$

$T = 'iPhone 11 Pro Max?'$

注：有的地方用双引号（如Java、C）
有的地方用单引号（如Python）

子串：串中任意个连续的字符组成的子序列。

主串：包含子串的串。

字符在主串中的位置：字符在串中的序号。

子串在主串中的位置：子串的第一个字符在主串中的位置。

Eg: 'iPhone'，'Pro M' 是串 T 的子串

Eg: T 是子串 'iPhone' 的主串

Eg: '1' 在 T 中的位置是 8 (第一次出现)

Eg: '11 Pro' 在 T 中的位置为 8

空串 V.S 空格串：

$M = ""$

M 是空串

$N = ' '$

3

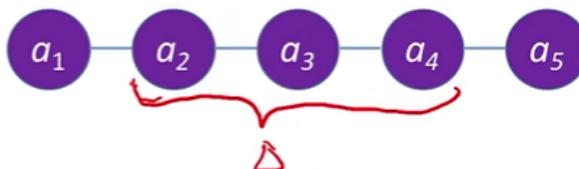
N 是由三个空格字符组成的空格串，每个空格字符占 1B

注意：位序从 1 开始
而不是从 0 开始

• 串 V.S 线性表

串 V.S 线性表

串是一种特殊的线性表，数据元素之间呈线性关系



有啥不一样~~

串的数据对象限定为字符集（如中文字符、英文字符、数字字符、标点字符等）

串的基本操作，如增删改查等通常以子串为操作对象



我们不一样

• 串的基本操作

串的基本操作

假设有串T="", S="iPhone 11 Pro Max?", W="Pro"

StrAssign(&T,chars): 赋值操作。把串T赋值为chars。

StrCopy(&T,S): 复制操作。由串S复制得到串T。

StrEmpty(S): 判空操作。若S为空串，则返回TRUE，否则返回FALSE。

StrLength(S): 求串长。返回串S的元素个数。

ClearString(&S): 清空操作。将S清为空串。

DestroyString(&S): 销毁串。将串S销毁（回收存储空间）。

Concat(&T,S1,S2): 串联接。用T返回由S1和S2联接而成的新串。

SubString(&Sub,S,pos,len): 求子串。用Sub返回串S的第pos个字符起长度为len的子串。

Index(S,T): 定位操作。若主串S中存在与串T值相同的子串，则返回它在主串S中第一次出现的位置；否则函数值为0。

StrCompare(S,T): 比较操作。若S>T，则返回值>0；若S=T，则返回值=0；若S<T，则返回值<0。

Eg: 执行基本操作 Concat(&T, S, W) 后, T="iPhone 11 Pro Max?Pro" 存储空间扩展?

执行基本操作 SubString(&T, S, 4, 6)后, T="one 11"

执行基本操作 Index(S, W)后, 返回值为 11

• 串的比较操作

串的比较操作

StrCompare(S,T): 比较操作。若S>T，则返回值>0；若S=T，则返回值=0；若S<T，则返回值<0。

A

a < o

abandon/ ə'bændən/ vt.丢弃;放弃, 抛弃

"abandon" < "aboard"

aboard/ ə'bɔ:d/ ad.在船(车)上;上船

absolute/ 'æbsəlju:t/ a.绝对的;纯粹的

absolutely/ 'æbsəlju:tli/ ad.完全地;绝对地

absorb/ ə'bɔ:sb/ vt.吸收;使专心

abstract/ 'æbstræk:t/ n.摘要

abundant/ ə'bʌndənt/ a.丰富的;大量的

abuse/ ə'bjuz, ə'bjus/ vt.滥用;虐待 n.滥用

academic/ ə'kædə'me:k/ a.学院的;学术的

accelerate/ ə'kɛlə'reit/ vt.(使)加快;促进

从第一个字符开始往后依次对比,
先出现更大字符的串就更大

长串的前缀与短串相
同时, 长串更大

"abstract" < "abstraction"

"abstract" < "abstract "

只有两个串完全相
同时, 才相等

"academic" > "abuse"

"academic" = "academic"

• 字符集编码

字符集编码

y = f(x)

字符集：函数定义域

编码：函数映射规则 f

y：对应的二进制数

任何数据存到计算机中
一定是二进制数。

需要确定一个字符和二
进制数的对应规则
这就是“编码”

“字符集”：
英文字符——ASCII字符集
中英文——Unicode字符集

基于同一个字符集，
可以有多种编码方案，
如：UTF-8, UTF-16

注：采用不同的编码方
式，每个字符所占空间
不同，考研中只需默认
每个字符占1B即可

ASCII 字符代码表 一

高四位	ASCII 非打印控制字符								ASCII 打印字符								
	0000		0001		0010		0011		0100		0101		0110		0111		
十进制	字符	ctrl	代码	字符解释	十进制	字符	ctrl	代码	字符解释	十进制	字符	ctrl	代码	字符解释	十进制	字符	
0000 0	0	BLANK	0	NULL	16	▶	P	DLE	数据链路终止	32	48	0	@	80	P	96	112 p
0001 1	1	②	1	A SOH	17	◀	Q	DC1	设备控制 1	33	! 49	1	65	A	81	Q	97 a 113 q
0010 2	2	③	2	B STX	18	↑	R	DC2	设备控制 2	34	" 50	2	66	B	82	R	98 b 114 r
0011 3	3	④	3	C ETX	19	!!	S	DC3	设备控制 3	35	# 51	3	67	C	83	S	99 c 115 s
0100 4	4	⑤	4	D ETB	20	¶	T	DC4	设备控制 4	36	\$ 52	4	68	D	84	T	100 d 116 t
0101 5	5	⑥	5	E ENQ	21	ֆ	U	NAK	数据确认	37	% 53	5	69	E	85	U	101 e 117 u
0110 6	6	⑦	6	F ACK	22	■	Y	STX	同步空间	38	& 54	6	70	F	86	V	102 f 118 v
0111 7	7	⑧	7	G BEL	23	↑	₩	ETB	传输块结束	39	' 55	7	71	G	87	W	103 g 119 w
1000 8	8	⑨	8	H BS	24	↑	X	CAN	取消	40	(56	8	72	H	88	X	104 h 120 x
1001 9	9	⑩	9	I TAB	25	↓	Y	EM	媒体结束	41) 57	9	73	I	89	Y	105 i 121 y
1010 A	10	⑪	10	J LF	26	→	Z	SUB	替换	42	* 58	:	74	J	90	Z	106 j 122 z
1011 B	11	⑫	11	K VT	27	←	[ESC	转义	43	+ 59	:	75	K	91	{	107 k 123 {
1100 C	12	⑬	12	L FF	28	↖	\	FS	文件分隔符	44	, 60	<	76	L	92	\	108 l 124 l
1101 D	13	⑭	13	M CR	29	↔]	GS	组分隔符	45	- 61	=	77	M	93] 109 m 125 }	
1110 E	14	⑮	14	N SO	30	▲	^	RS	记录分隔符	46	. 62	>	78	N	94	^	110 n 126 ~
1111 F	15	⑯	15	O SI	31	▼	-	US	单元分隔符	47	/ 63	?	79	O	95	_	111 o 127 △ Backspace

注：表中的ASCII字符可以用：ALT + “小键盘上的数字键” 输入。

拓展：乱码问题

乱码问题的根源在于不同的系统或软件对同一字节序列的解释不同。假设你有一个文件，原本采用一套编码规则 $y=f(x)$ ，如：

$y = f(x)$

$x = f^{-1}(y)$

在你的文件中，原本采用某一套编码规则 $y=f(x)$ ，如：

打开文件时，你的软件以为你采用的是另一套编码规则 $y=g(x)$ ，如：

二、串的存储结构

• 串的顺序存储

```

1 #define MAXLEN 255      //预定义最大串长为255
2 typedef struct
3 {
4     char ch[MAXLEN];   //每个分量存储一个字符
5     int length;        //串的实际长度
6 }SString;
7
8 typedef struct
9 {
10    char *ch;           //按串长分配存储区，ch指向串的基地址
11    int length;         //串的长度
12 }HString;
13
14 HString S;
15 S.ch = (char *)malloc(MAXLEN * sizeof(char));
16 S.length = 0;

```

串的顺序存储

```

#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;

```

静态数组实现
(定长顺序存储)

```

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;

```

HString S;

S.ch = (char *)malloc(MAXLEN * sizeof(char));

S.length = 0;

用完需要手动free



串的顺序存储

char ch[10]

方案一：

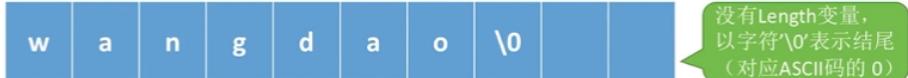


ch[0]充当 Length

方案二：



方案三：



ch[0]废弃不用

方案四：
(教材)

• 串的链式存储

```

1 typedef struct StringNode
2 {
3     char ch;           //每隔结点存1个字符
4     struct StringNode * next;
5 }StringNode, * String;

```



存储密度低：每个字符1B，每个指针4B

```

1 typedef struct StringNode
2 {
3     char ch[4];        //每个结点存多个字符
4     struct StringNode * next;
5 }StringNode, * String;

```



存储密度提高

• 基本操作的实现



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

StrAssign(&T,chars): 赋值操作。把串T赋值为chars。
 StrCopy(&T,S): 复制操作。由串S复制得到串T。
 StrEmpty(S): 判空操作。若S为空串，则返回TRUE，否则返回FALSE。
 StrLength(S): 求串长。返回串S的元素个数。
 ClearString(&S): 清空操作。将S清为空串。
 DestroyString(&S): 销毁串。将串S销毁（回收存储空间）。
 Concat(&T,S1,S2): 串联接。用T返回由S1和S2联接而成的新串

SubString(&Sub,S,pos,len): 求子串。用Sub返回串S的第pos个字符起长度为len的子串。

S.ch="wangdao"
S.length=7

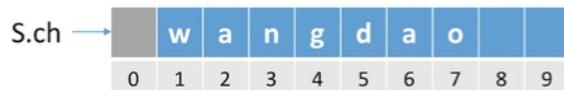


```
1 #define MAXLEN 255      //预定义最大串长为255
2 typedef struct
3 {
4     char ch[MAXLEN];   //每个分量存储一个字符
5     int length;         //串的实际长度
6 }SString;
7
8 //求子串
9 bool SubString(SString &Sub,SString s,int pos,int len)
10 {
11     //子串范围越界
12     if(pos + len - 1 > s.length)
13         return false;
14     for(int i = pos;i < pos + len;i++)
15         Sub.ch[i - pos + 1] = s.ch[i];
16     Sub.length = len;
17     return true;
18 }
```

基本操作的实现

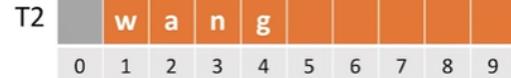
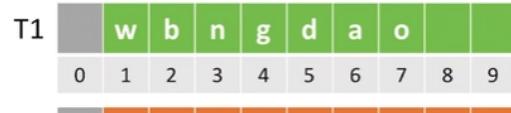
StrCompare(S,T): 比较操作。若S>T，则返回值>0；若S=T，则返回值=0；若S<T，则返回值<0。

S.ch="wangdao"
S.length=7



//比较操作。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 StrCompare(SSString S, SSString T) {
    for (int i=1; i<=S.length && i<=T.length; i++){
        if (S.ch[i]!=T.ch[i])
            return S.ch[i]-T.ch[i];
    }
    //扫描过的所有字符都相同，则长度长的串更大
    return S.length-T.length;
}
```

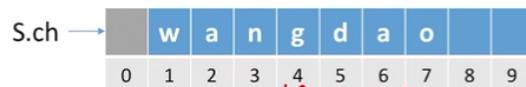


```
1 //比较操作。若S > T，则返回值>0；若S = T，则返回值=0；若S < T，则返回值<0
2 int StrCompare(SSString S,SSString T)
3 {
4     for(int i = 1;i <= s.length && i <= t.length;i++)
5     {
6         if(s.ch[i] != t.ch[i])
7             return s.ch[i] - t.ch[i];
8     }
9     //扫描过的所有字符都相同，则长度长的串更大
10    return s.length - t.length;
11 }
```

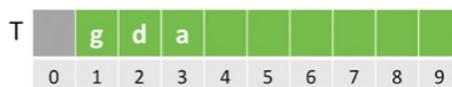
基本操作的实现

Index(S,T): 定位操作。若主串S中存在与串T值相同的子串，则返回它在主串S中第一次出现的位置；否则函数值为0。

S.ch="wangdao"
S.length=7



```
int Index(SSString S, SSString T){
    int i=1, n=StrLength(S), m=StrLength(T);
    SSString sub; //用于暂存子串
    while(i<=n-m+1){
        SubString(sub,S,i,m);
        if(StrCompare(sub, T)!=0) ++i;
        else return i; //返回子串在主串中的位置
    }
    return 0; //S中不存在与T相等的子串
}
```



```
1 int Index(SSString S,SSString T)
2 {
3     int i = 1,n = StrLength(S),m = StrLength(T);
4     SSString sub; //用于暂存子串
5     while(i <= n - m + 1)
6     {
7         SubString(sub,S,i,m);
8         if(StrCompare(sub, T) != 0)
9             ++i;
```

```

10         else
11             return i;           //返回子串在主串中的位置
12     }
13     return 0;                //S中不存在与T相等的子串
14 }

```

三、知识总结

