



中山大學
SUN YAT-SEN UNIVERSITY

字符串与字符数组

中山大学计算机学院



主讲人：万海

目录

CONTENTS

01

字符与字符集

02

字符串概述

03

字符数组

04

字符串输入输出

05

字符串基本操作



字符 (Chars) - 整数

8 位
1 个字节

```
#include <stdio.h>

int main() {
    char c0 = 'A';           // 字符由单引号 ' ' 包围
    char c1 = 65;            // 字符是一个数字
    char c2 = '\n';          // 特殊字符可用转义
    char c3 = '\x41';        // 用16进制表示字符
    char c4 = '\101';        // 用8进制表示字符
    printf("%c,%c%c", c0, c1, c2);
    printf("%c,%c", c3, c4);
}
```

输出

A,A
A,A



ASCII表，字符与数字的映射

- 美国信息交换标准代码
- American Standard Code for Information Interchange / ASCII
- 共 128 个编码
- 历史
 - 用于电传打印机 (teletype)
 - 也称远程打印
- 原理
 - 按键 A, 发送 65
 - 收到 65, 打印 A
 - 按键 ^G, 发送 7
 - 收到 7, 响铃提示
 - 按键 ^M, 发送 13
 - 收到 13, 打印头回零位
 - 按键 ^J, 发送 10
 - 收到 10, 打印头下移一行
 - 按键 ^D, 发送 4
 - 收到 4, 电报传输结束

ASCII 字符代码表 一

高四位 低四位		ASCII非打印控制字符										ASCII 打印字符										
		0000					0001					0010	0011	0100	0101	0110	0111					
		0					1					2	3	4	5	6	7					
	+进制	字符	ctrl	代码	字符解释	+进制	字符	ctrl	代码	字符解释	+进制	字符	+进制	字符	+进制	字符	+进制	字符	+进制	字符	ctrl	
0000	0	0	BLANK NULL	^@	NUL 空	16	▶	^P	DLE	数据链路转意	32		48	0	64	@	80	P	96	`	112	p
0001	1	1	☺	^A	SOH 头标开始	17	◀	^Q	DC1	设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q
0010	2	2	☹	^B	STX 正文开始	18	↕	^R	DC2	设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r
0011	3	3	♥	^C	ETX 正文结束	19	!!	^S	DC3	设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s
0100	4	4	♦	^D	EOT 传输结束	20	¶	^T	DC4	设备控制 4	36	\$	52	4	68	D	84	T	100	d	116	t
0101	5	5	♣	^E	ENQ 查询	21	§	^U	NAK	反确认	37	%	53	5	69	E	85	U	101	e	117	u
0110	6	6	♠	^F	ACK 确认	22	■	^V	SYN	同步空闲	38	&	54	6	70	F	86	V	102	f	118	v
0111	7	7	●	^G	BEL 震铃	23	↑	^W	ETB	传输块结束	39	'	55	7	71	G	87	w	103	g	119	w
1000	8	8	◼	^H	BS 退格	24	↑	^X	CAN	取消	40	(56	8	72	H	88	X	104	h	120	x
1001	9	9	○	^I	TAB 水平制表符	25	↓	^Y	EM	媒体结束	41)	57	9	73	I	89	Y	105	i	121	y
1010	A	10	◻	^J	LF 换行/新行	26	→	^Z	SUB	替换	42	*	58	:	74	J	90	Z	106	j	122	z
1011	B	11	♂	^K	VT 竖直制表符	27	←	^[ESC	转意	43	+	59	;	75	K	91	[107	k	123	{
1100	C	12	♀	^L	FF 换页/新页	28	└	^\	FS	文件分隔符	44	,	60	<	76	L	92	\	108	l	124	
1101	D	13	🎵	^M	CR 回车	29	↔	^]	GS	组分隔符	45	-	61	=	77	M	93]	109	m	125	}
1110	E	14	🎵	^N	SO 移出	30	▲	^6	RS	记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~
1111	F	15	☼	^O	SI 移入	31	▼	^-	US	单元分隔符	47	/	63	?	79	O	95	_	111	o	127	Δ

Back space

注：表中的ASCII字符可以用:ALT + “小键盘上的数字键” 输入



ASCII表基本特征

- **控制符 (共33个)**

- 前 32 个 字 符 , 例 如 : 7, '\t'; 10, '\n'; 13, '\r' ...
- 127是控制符, DEL

- **可打印字符 (共95个)**

- 32是空格
- 48-57是'0'... '9'
- 65-90是'A'... 'Z'
- 97-122是'a'... 'z'

- **五个空白字符 (blank char)**

- 空格、水平制表、垂直制表、换页、换行

小知识:

不同操作系统下换行的表示:

\n: UNIX 系统行末结束符

\r\n: window 系统行末结束符

\r: MAC OS 系统行末结束符

我们经常遇到的一个问题就是, Unix/Mac系统下的文件在Windows里打开的话, 所有文字会变成一行; 而Windows里的文件在Unix/Mac下打开的话, 在每行的结尾可能会多出一个^M符号。



ASCII 的扩展字符集

- 扩展ASCII
 - 对 128-255 定义了西方常用字母
 - 例如: 'ä' 的编码 '\xe4'
- 大字符集, 16位编码
 - JIS 日文字母与日文汉字
 - Big5 港台繁体喊字
 - GB2312, GBK, GB18030 国标
 - ANSI windows 大字符集标准
 - 中国 CP936, 即 GBK
 - 日本 CP932, 即 JIS

小知识:

大字符集一般用两个**扩展ASCII**表示一个字母(汉字)。这样, ascii 和 汉字 混在一起也容易识别。

为了与文字传输、控制设备兼容, 仅可打印字符区域对应区域的扩展ASCII可用于大字符集的编码。

因历史原因, 各地编码并不统一。使用不当编码打开文档, 会产生乱码。

Unicode (单一编码) 字符集

- Unicode是一个字符集，它是所有其他广泛使用字符集的超集，包含了来自ISO/IEC 6937、ISO/IEC 8859家族、**Big5**、KS X 1001、JIS X 0213、**GB 2312**、**GBK**、GB 18030、HKSCS和CNS 11643等字符集的字符。
- 它包含三种编码形式
 - UTF-8, 变长格式
 - UTF-16, 16位宽字符
 - UTF-32, 32位宽字符

Code (Hex)	Character	Source
0041	A	English (Latin)
042F	Я	Russian (Cyrillic)
OE09	฿	Thai
13EA	Ꮝ	Cherokee
211E	℞	Letterlike Symbols
21CC	⇌	Arrows
282F	⠆	Braille
345F	低	Chinese/Japanese/ Korean (Common)

图: UTF-16 部分字符编码



常见字符集编码的区别（样例）

字符	ASCII	UTF-16	UTF—8	GBK
A	41	00 41	41	—
汉	—	6C 49	E6 B1 89	BA BA

因此，当字符串中包含汉字时，如果编译程序默认时 UTF-8 编码，每个汉字占3个 char。如果默认是 ANSI，即 GBK，则每个汉字 2 个字节。

3字节

如果你在2000年前出身，你出生证上无法打出“燊”这些字，因为DOS使用的是GB2312，只有6000个常用汉字。Windows 95 开始，GBK 包括了 GB2312，同时又增加了近20000个新的汉字（包括繁体字）和符号。



C语言字符类型 (type) 与常量

- 单字节字符, 表示英文或西方字符
 - 类型 `char`, 一个字节
 - 例如 `'a'` 或 `'\n'` 或 `'\13'`
- 16 位宽字符, 表示中文, 日文等
 - 类型 `char16_t`, 两个字节无符号整数
 - 例如 `u'字'` 但非 `u'字'` (`u'\U0001f34c'`)
- 32 位宽字符, 表示 emoji 等扩展
 - 类型 `char32_t`, 四个字节无符号整数
 - 例如 `U'字'` 或 `U'字'`
- 宽字符
 - 类型 `wchar_t`, 设定为16位或32位
 - 例如 `L'β'` 或 `L'字'`。

代码范例参见: [字符常量 - cppreference.com](https://en.cppreference.com/zh-cn/char_constants)

转义符	含义
<code>\\</code>	<code>\</code> 字符
<code>\'</code>	<code>'</code> 字符
<code>\"</code>	<code>"</code> 字符
<code>\?</code>	<code>?</code> 字符
<code>\a</code>	警报铃声
<code>\b</code>	退格键
<code>\f</code>	换页符
<code>\n</code>	换行符
<code>\r</code>	回车
<code>\t</code>	水平制表符
<code>\v</code>	垂直制表符
<code>\ooo</code>	一到三位的八进制数
<code>\xhh...</code>	一个或多个数字的十六进制数
<code>\uhhhh...</code>	UTF-16或UTF-32编码



字符串 (Strings) 概述

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

```
int main() {
```

```
    char* greeting = "Hello";
```

// 字符串由双引号" "包围

// 它由6个字符构成 'H' 'e' 'l' 'l' 'o' '\0'

```
    char your_name[] = "孙大圣";
```

// 由10个, 或7个字符构成?

```
    printf("%s %s!\n", greeting, your_name);
```

// 使用 %s 格式化

```
    char* no_zero = "孙\000大圣";
```

// 中间不能包含NULL字符

```
    printf("%s %s!\n", greeting, no_zero);
```

```
}
```

输出

3位八进制

Hello 孙大圣!

Hello 孙!

UTF-8 GBK



字符串概述——字符串的基本概念

定义：

由一串连续的字符组成的**字符序列**称为**字符串(string)**。

例如，“aabbccdd”，“wkfueh@ei~”，“t”，“Hello World.”均为字符串。

概念：

字符集（字母表）：字符串可能包含的字符集合，记为 Σ 。eg. GBK

字符串长度：一个字符串所包含字符的数目。

例如，“efl35”的长度为5，“wofskl6”的长度为7，“12345”的长度为5。

一般地，字符集 Σ 上长度为 n 的字符串共有 $|\Sigma|^n$ 个。

空串：长度为0的字符串，记为 \emptyset 。



字符串概述——字符串的基本概念

字符是组成字符串的基本单元。

例如，“Hello World.”中包含的字符依次为‘H’、‘e’、‘l’、‘l’、‘o’、‘ ’、‘W’、‘o’、‘r’、‘l’、‘d’、‘.’。

其中，“Hello World.”的第一个字符是H，第六个字符是空格，最后一个字符是英文句号。

因此，使用字符数组可以方便的存储字符串数据。

C语言没有字符串

只有字符数组



字符串概述——字符串的基本概念

子串：由某个字符串中连续的一段字符组成的字符串。

例如“abcd”的所有子串为 \emptyset , “a”, “b”, “c”, “d”, “ab”, “bc”, “cd”, “abc”, “bcd”, “abcd”。

PS: acd 不是子串

长度为 n 的字符串共有 $\binom{n+1}{2} + 1 = O(n^2)$ 个子串。

$$\frac{n(n+1)}{2} + 1$$



中山大學

SUN YAT-SEN UNIVERSITY

字符串概述——字符串的基本概念

前缀：某个字符串开头的一段子串。

例如“abcd”的所有前缀为 \emptyset , “a”, “ab”, “abc”, “abcd”。

长度为 n 的字符串共有 $n + 1$ 个前缀。



字符串概述——字符串的基本概念

后缀：某个字符串末尾的一段子串。

例如“abcd”的所有后缀为 \emptyset , “d”, “cd”, “bcd”, “abcd”。

长度为 n 的字符串共有 $n + 1$ 个后缀。



字符数组

在C语言中，可以使用char类型声明字符类型，其占用1个字节，按照ASCII码的顺序编码字符。

在C语言中，可以使用字符数组存储与表示一个字符串。字符数组可以用类似如下的语句定义：

```
char str[10];
```

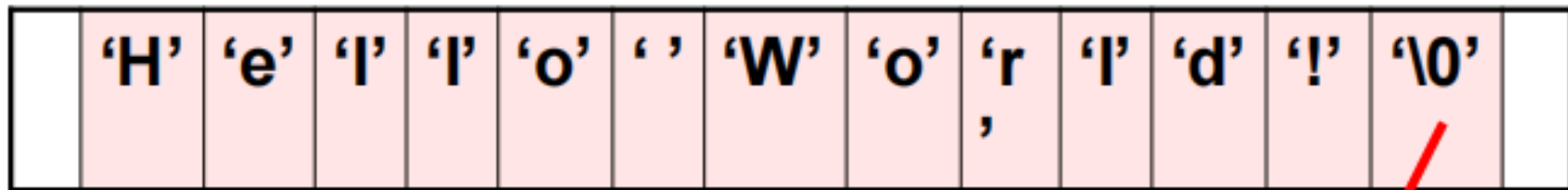
这一条语句定义了名为str的包含10个char的字符数组，可以用该字符数组存储字符串。

字符数组——字符串常量

双引号 “” 括起来的字符序列称为字符串常量。例如：

“Hello World!” “I am a teacher\n”

字符串常量在内存中的存储：



结束符



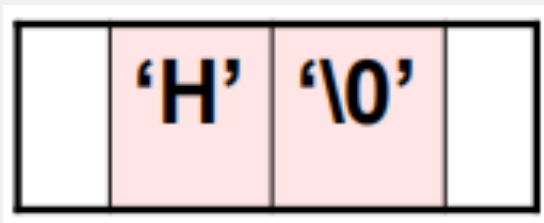
中山大學

SUN YAT-SEN UNIVERSITY

字符数组——字符串与字符的区别

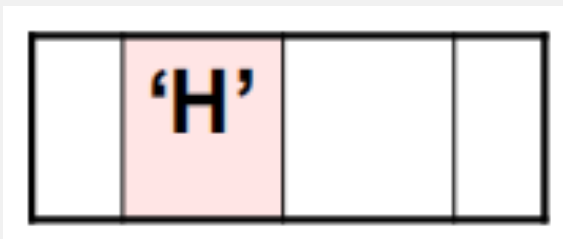
字符串 “H” 在内存中的存储为：

“ ”



而字符 ‘H’ 在内存中的存储为：

“ ”





字符数组——字符串常量的声明

`char ch = 'H' ; //正确`

`char ch = "H" ; //错误`

可以用多个字符或一个字符串对char类型的数组元素进行初始化。

- 1) `char msg[3] = { 'H' , 'i' , '!' } ;` //不是字符串 没有
- 2) `char msg[4] = { 'H' , 'i' , '!' , '\0' } ;` 结束符
- 3) `char msg[4] = "Hi!" ;`
- 4) `char msg[] = "Hi!" ;`

其中2)、3)、4)三种声明是等价的字符串。



中山大學

SUN YAT-SEN UNIVERSITY

字符数组——char类型数组、字符串与字符

地址

```
char msg[4];  
msg = "Hi!";
```



`char msg[4] = "Hi!";` ✓

```
char msg[4];  
msg[0] = 'H';  
msg[1] = 'i';  
msg[2] = '!';  
msg[3] = '\0';
```





字符数组——char类型数组与字符串

以下两种char类型的数组有区别：

1) `char msg[3] = { 'H' , 'i' , '!' };`

2) `char msg[4] = { 'H' , 'i' , '!' , '\0' };`

第一种没有结束符，是一般的char类型数组，不被认为是字符串。因为处理字符串的标准库函数都需要依靠结束符来判断该字符串是否结束，一般无法处理这种数组（或处理有误）。

第二种有结束符，被认为是字符串。标准库中有很多函数可以整体处理这种数组（及字符串常量）。



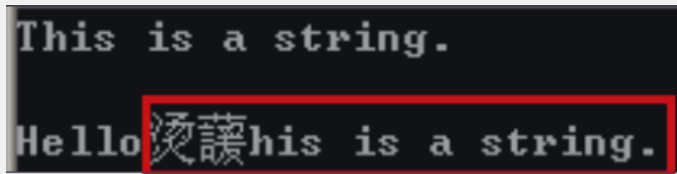
字符串输入输出——输出方法

`char str1[] = "This is a string.";` 自动加 `\0`

`char str2[] = {'H', 'e', 'l', 'l', 'o'};` 字符集不自动加 `\0`

`printf("%s\n\n", str1);` //字符串整体输出用 `%s`

`printf("%s\n\n", str2);`



↓
不是字符串

由于 `str2` 没有结束符，所以输出将一直进行直到遇到某个字节的值刚好为结束符为止（这具有随机性）。对于 `str2`，正确的输出方法应该是利用循环逐个输出元素。



字符串输入输出——输出方法

```
char str1[] = "This is a string.";
```

```
char str2[] = { 'H', 'e', 'l', 'l', 'o', '\0' };
```

```
printf( "%s\n\n", str1); //字符串整体输出用%s
```

```
printf( "%s\n\n", str2);
```

```
This is a string.
```

```
Hello
```

str1和str2都有结束符，输出至结束符处结束。



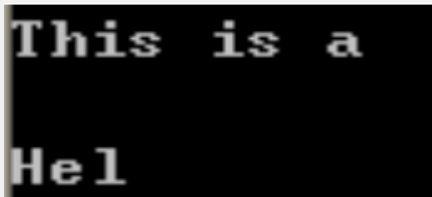
字符串输入输出——输出方法

```
char str1[] = "This is a\0 string.";
```

```
char str2[] = { 'H', 'e', 'l', '\0', 'l', 'o', '\0' };
```

```
printf( "%s\n\n", str1); //字符串整体输出用%s
```

```
printf( "%s\n\n", str2)
```



```
This is a
Hel
```

字符串的输出遇到第一个结束符时就结束。



字符串输入输出——输出方法

字符串可以利用%s整体输出，但对于其它类型的数组，例如
int\float\double等，不能整体输出，只能利用循环逐个元素输出。



字符串输入输出——输入方法

```
char str[10];
```

```
printf( "Please enter a string less than 9 characters:\n " );
```

```
scanf( "%s" , str );
```

//注意：不需要取地址运算符&

```
printf("The string you enter is: %s\n\n", str);
```

字符串可以利用%s整体输入，但对于其它类型的数组，例如

int\float\double等，不能整体输入，只能利用循环逐个元素输入。



字符串输入输出——输入方法

```
char str[10];
```

```
printf( "Please enter a string less than 9 characters:\n " );
```

```
scanf( "%s" , str ); //注意：不需要取地址运算符&
```

```
printf("The string you enter is: %s\n\n", str);
```

```
Please enter a string less than 9 characters:
student
The string you enter is: student
```

键盘输入到' t' 时敲enter，将结束输入，并在' t' 后置入结束符。

str在内存中(? 代表值不确定):

	's'	't'	'u'	'd'	'e'	'n'	't'	'\0'	?	?	
--	-----	-----	-----	-----	-----	-----	-----	------	---	---	--



字符串输入输出——输入方法

```
char str[10];  
  
printf( "Please enter a string less than 9 characters:\n " );  
  
scanf( "%s" , str ); //注意：不需要取地址运算符&  
  
printf("The string you enter is: %s\n\n", str);
```

```
Please enter a string less than 9 characters:  
stu dent  
The string you enter is: stu
```

在遇到空白字符时，仅获得空白字符之前的字符串。



字符串输入输出——gets/puts

```
char str[100];
```

```
puts( "Enter a string:\n" ); //输出字符串
```

```
gets( str ); //输入字符串
```

```
puts( "The string you enter is:\n" );
```

```
puts( str );
```

```
Enter a string:
I am a teacher
The string you enter is:
I am a teacher
```

(1) printf 用得更广泛，因此 puts 用得不多；(2) gets 可以获得一整行（包括空格）。



字符串操作——求字符串长度strlen

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

```
:
```

```
char str[]="Hello!";
```

```
unsigned int l1, l2;
```

```
l1=strlen(str);
```

```
l2=strlen("C PROGRAMMING!");
```

```
printf("l1 = %u, l2 = %u\n\n", l1, l2);
```

l1 = 6, l2 = 14

strlen返回字符串的长度，但不包括字符串的结尾空字符 ‘\0’ 。



字符串操作——字符串的连接strcat(toStr, fromStr)

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

```
:
```

```
char str1[15]= { "I love "};
```

```
char str2[ ] = { "China!" };
```

```
printf( "%s" , strcat( str1, str2 ) );
```

(1) 去掉str1的结束符，并将str2连接到str1上。(2) str1数组必须够长，否则出错。(3) 返回str1数组的首地址，即str1。

连接前的str1

'I'	' '	'I'	'o'	'v'	'e'	' '	'\0'	'?	'?	'?	'?	'?	'?	'?
-----	-----	-----	-----	-----	-----	-----	------	----	----	----	----	----	----	----

'C'	'h'	'i'	'n'	'a'	'!'	'\0'
-----	-----	-----	-----	-----	-----	------

 str2

连接后的str1

'I'	' '	'I'	'o'	'v'	'e'	' '	'C'	'h'	'i'	'n'	'a'	'!'	'\0'	'?
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	----



字符串操作——字符串的连接strncat(toStr, fromStr,n)

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

```
:
```

```
char str1[15]= { "I love "
```

```
char str2[ ] = { "China!" };
```

```
printf( "%s" , strncat( str1, str2, 2 ) );
```

把fromStr的前n个字符连接到toStr上，其余与strcat类似。

返回str1数组的首地址，即str1

连接前的str1

'I'	' '	'I'	'o'	'v'	'e'	' '	'\0'	?	?	?	?	?	?	?
-----	-----	-----	-----	-----	-----	-----	------	---	---	---	---	---	---	---

'C'	'h'	'i'	'n'	'a'	'!'	'\0'	str2
-----	-----	-----	-----	-----	-----	------	------

连接后的str1

'I'	' '	'I'	'o'	'v'	'e'	' '	'C'	'h'	'\0'	?	?	?	?	?
-----	-----	-----	-----	-----	-----	-----	-----	-----	------	---	---	---	---	---



字符串操作——字符串的拷贝strcpy(toStr, fromStr)

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

```
:
```

```
char str1[15];
```

```
char str2[] = { "I love China!" };
```

```
strcpy(str1, str2); //strcpy将fromStr包括空字符 '\0' 拷贝到toStr
```

```
strcpy(str1, "You are a student.");
```

```
strncpy(str1, str2, 4 ); //strncpy将fromStr的头n个字符拷贝到  
toStr相应的位置上
```



字符串操作——字符串的比较strcmp(str1, str2)

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

```
:
```

```
char str1[] = "Hell";
```

```
char str2[] = "Hello";
```

```
printf( "%d\n", strcmp(str1, str2) );
```

```
strcpy( str1, "ABcD" );
```

```
strcpy( str2, "ABCD" );
```

```
printf( "%d\n", strcmp(str1, str2) );
```

```
printf( "%d\n", strcmp("I am fine", "I am fine"));
```

-1
1
0

str1和str2按照字符在字符集中的表示值进行比较：从字符串第一个字符开始，逐个字符两两进行比较；第一对不同的字符出现时，较大字符所在的字符串就是大者。如果其中较短的字符串是较长字符串前面的子字符串，则较长的字符串大于较短的字符串。函数返回值：

str1 小于 str2时，返回小于0的整数值

str1 等于 str2时，返回整数值0

str1 大于 str2时，返回大于0的整数值



中山大學

SUN YAT-SEN UNIVERSITY

字符串操作——字符串的比较strncmp(str1, str2, n)

比较str1与str2头n个字符的大小，返回值与strcmp相同。



字符串操作——大小写转换

`strlwr(str)`: 将字符串`str`的字母全部转化为小写。

`strupr(str)`: 将字符串`str`的字母全部转化为大写。

```
char str1[] = "Hello"; char ch = 'a';
```

```
strlwr(str1); puts(str1);
```

```
strupr(str1); puts(str1);
```

```
strupr(&ch);
```

```
printf("%c\n\n", ch);
```





字符串操作——字符检索strchr(str, c)

C 库函数 `char *strchr(const char *str, int c)` 在参数 `str` 所指向的字符串中搜索指向第一次出现字符 `c`（一个无符号字符）位置的指针。

```
char str[] = "abcdedcbaf";
```

```
printf("%d\n", strchr(str, 'd') - str); // 'd' 在str中第一次出现的下标
```

```
printf("%s\n", strchr(str, 'd')); // 从第一次 'd' 出现开始的字符串
```

```
3  
dedcbaf
```



字符串操作——字符串检索strstr(str1, str2)

C 库函数 `char *strstr(const char *str1, const char *str2)` 在字符串 `str1` 中查找指向第一次出现字符串 `str2` 位置的指针(不包含终止符 `'\0'`)。

```
char str[] = "abcdedcbaf";
```

3
dedcbaf

```
printf("%d\n", strstr(str, "ded") - str); // “ded” 在str中第一次出现的下标
```

```
printf("%s\n", strstr(str, "ded")); // 从第一次 “ded” 出现开始的字符串
```



总结

操作	代码	操作	代码
输入	<code>scanf("%s", str);</code>	字符串的拷贝	<code>strcpy(toStr, fromStr)</code>
输出	<code>printf("%s", str);</code>	字符串的比较	<code>strcmp(str1, str2) /</code> <code>strncmp(str1, str2, n)</code>
整行输入	<code>gets(str);</code>	转换为小写	<code>strlwr(str)</code>
整行输出	<code>puts(str);</code>	转换为大写	<code>strupr(str)</code>
获取/操作第i个字符	<code>str[i]</code>	字符检索	<code>strchr(str, c)</code>
求字符串str长度	<code>strlen(str)</code>	字符串检索	<code>strstr(str1, str2)</code>
字符串的连接	<code>strcat(toStr, fromStr)</code> <code>/ strncat(toStr,</code> <code>fromStr, n)</code>	字符串结束标志	<code>'\0'</code> (ASCII码为0的字符)