

# 第1讲：C语言常见概念

## 目录

1. C语言是什么？
2. C语言的历史和辉煌
3. 编译器的选择VS2022
4. VS项目和源文件、头文件介绍
5. 第一个C语言程序
6. main函数
7. printf和库函数
8. 关键字介绍
9. 字符和ASCII编码
10. 字符串和\0
11. 转义字符
12. 语句和语句分类
13. 注释是什么？为什么写注释？

---

正文开始

## 1. C语言是什么？

人和人交流使用的是自然语言，如：汉语、英语、日语

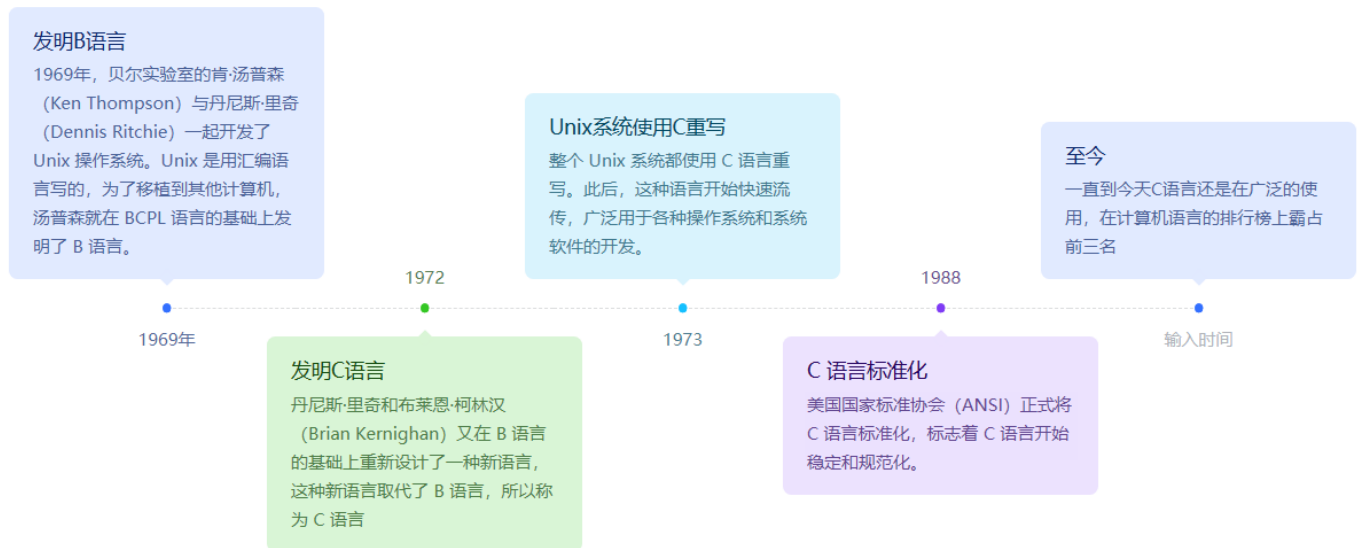
那人和计算机是怎么交流的呢？使用**计算机语言**。

目前已知已经有上千种计算机语言，人们是通过计算机语言写的程序，给计算机下达指令，让计算机工作的。

C语言就是众多计算机语言中的一种，当然C++/Java/Go/Python都是计算机语言。

## 2. C语言的历史和辉煌

C语言最初是作为 Unix 系统的开发工具而发明的。



<https://www.tiobe.com/tiobe-index/>

### 3. 编译器的选择VS2022

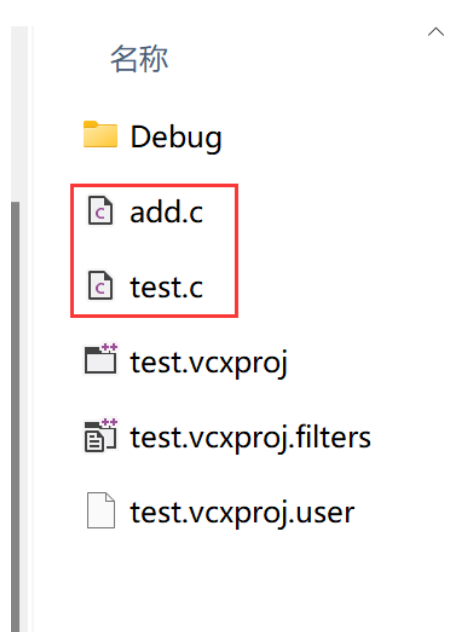
#### 3.1 编译和链接

C语言是一门**编译型**计算机语言，C语言源代码都是文本文件，文本文件本身无法执行，必须通过**编译器**翻译和**链接器**的链接，生成**二进制的可执行文件**，可执行文件才能执行。

C语言代码是放在 `.c` 为后缀的文件中的，要得到最终运行的可执行程序，中间要经过**编译**和**链接**2个过程。



VS2022项目中的.c文件



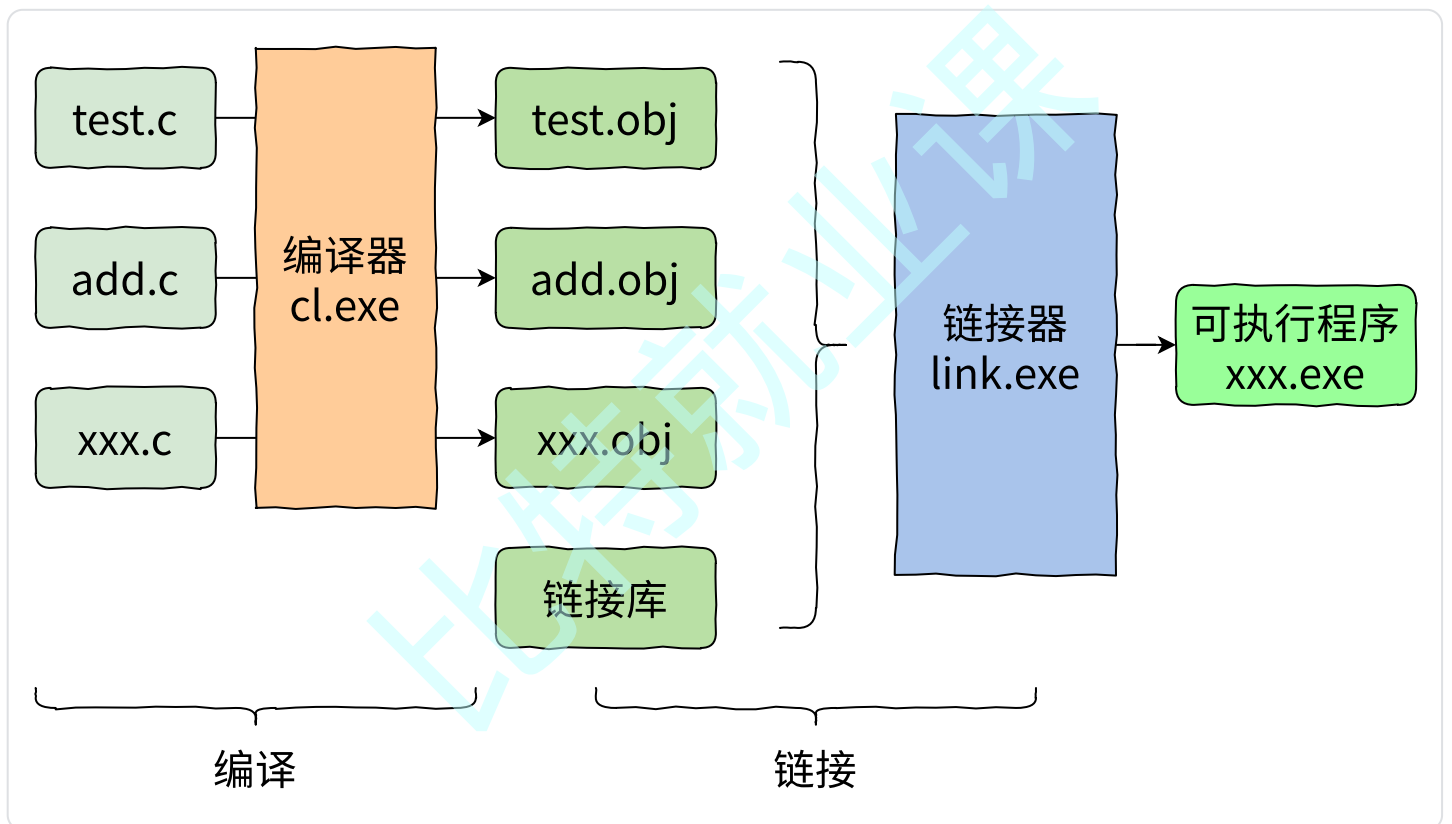
.c文件的展示

```
test.c  x
比特就业课主页：https://m.cctalk.com/inst/s9yewhfr
test (全局范围)
4 #include <stdio.h>
5
6 int main()
7 {
8     int a = 0;
9     int b = 0;
10    scanf("%d %d", &a, &b);
11    int c = Add(a, b);
12    printf("%d\n", c);
13
14    return 0;
15 }

add.c  x
add.c (全局范围)
1 #define _CRT_SECURE_NO_WARNINGS 1
2
3 int Add(int x, int y)
4 {
5     return x + y;
6 }
7
```

C语言代码

一个工程一般都会有多个源文件组成，如下图所示，演示了源程序经过编译器和链接器处理的过程。




注：

1. 每个源文件(.c)单独经过**编译器**处理生成对应的目标文件(.obj为后缀的文件)
2. 多个目标文件和库文件经过**链接器**处理生成对应的可执行程序(.exe文件)

这就是，在Windows电脑上C语言程序生成的exe可执行文件

比特就业课主页: <https://m.cctalk.com/inst/s9yewhfr>

名称	修改日期	类型	大小
 test_5_10.exe	2023/5/10 21:53	应用程序	39 KB
 test_5_10.pdb	2023/5/10 21:53	VisualStudio.pdb.d...	1,116 KB

可执行程序

## 3.2 编译器的对比

C语言是一门**编译型**的计算机语言，需要依赖编译器将计算机语言转换成机器能够执行的机器指令。

那我们常见的C语言编译器都有哪些呢？

比如：**msvc**、clang、gcc 就是一些常见的编译器，当然也有一些**集成开发环境** 如：VS2022、XCode、CodeBlocks、DevC++、Clion 等。

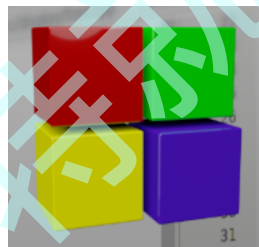
集成开发环境（IDE）用于提供程序开发环境的应用程序，一般包括**代码编辑器**、**编译器**、**调试器**和**图形用户界面**等工具。集成了**代码编写功能**、**分析功能**、**编译功能**、**调试功能**等一体化的开发软件服务套。



VS2022



XCode



CodeBlocks



DevC++



Clion

- **VS2022** 集成了MSVC（安装报包较大一些，安装简单，无需多余配置，使用起来非常方便）
- **XCode** 集成了clang（苹果电脑上的开发工具）
- **CodeBlocks** 集成了gcc（这个工具比较小众，需要配置环境，不太推荐）
- **DevC++** 集成了gcc（小巧，但是工具过于简单，对于代码风格的养成不好，一些竞赛使用）
- **Clion** 是默认使用CMake，编译器是可以配置的（工具是收费，所以暂时推荐大家使用）

整体考虑，**推荐大家安装 VS2022 的社区版本学习**，免费，使用方便，工作中常见。

VS2022 的安装教程：<https://www.bilibili.com/video/BV11R4y1s7jz/>

## 3.3 VS2022 的优缺点

## 优点：

- **VS2022** 是一个主流的集成开发环境，企业中使用较为普遍
- **VS2022** 包含了：编辑器+编译器+调试器，功能强大
- 直接安装即可使用，基本不用额外配置环境，上手容易
- 默认界面是中文的，初学者友好

## 缺点：

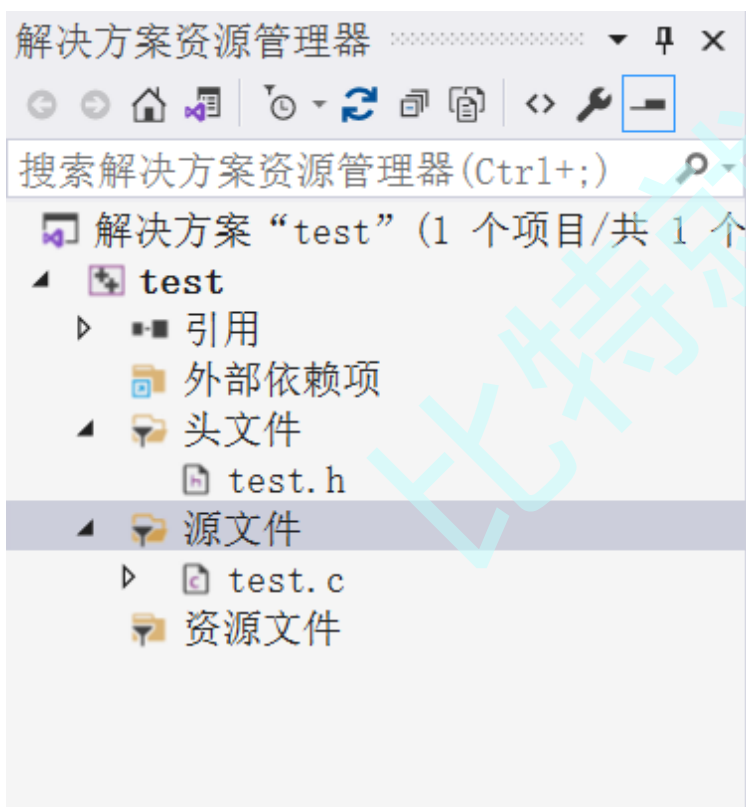
- 功能丰富，安装包大，占用空间多。

## 4. VS项目和 源文件、头文件介绍

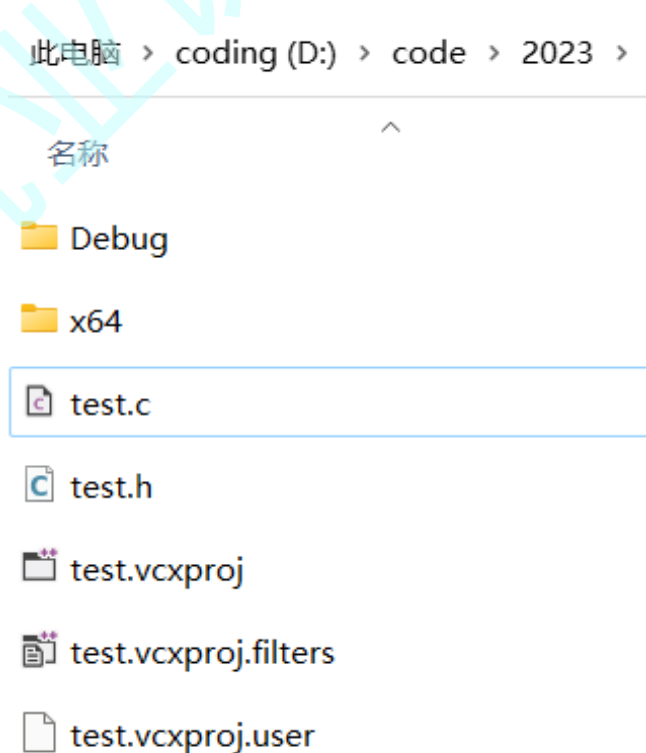
在VS上写代码，我们是需要创建项目的，直接新建项目就可以了。

在项目中就可以添加源文件和头文件。

C语言把 `.c` 为后缀的文件称为**源文件**，把 `.h` 为后缀的文件称为**头文件**。



头文件和源文件在VS中展示



头文件和源文件在磁盘上

## 5. 第一个C语言程序

```
1 #include <stdio.h>
2
```

```
3 int main()  
4 {  
5     printf("hello C\n");  
6     return 0;  
7 }
```

这里演示VS2022中创建项目和编写C代码的过程，并运行出结果。

在VS2022上运行代码的快捷键： `Ctrl+f5`

## 6. main函数

每个C语言程序不管有多少行代码，都是从 `main` 函数开始执行的，`main` 函数是程序的入口，`main` 函数也被叫做：**主函数**。`main` 前面的 `int` 表示 `main` 函数执行结束的时候返回一个整型类型的值。所以在 `main` 函数的最后写 `return 0;` 正好前后呼应。

- `main`函数是程序的入口
- `main`函数有且仅有一个
- 即使一个项目中有多个.c文件，但是只能有一个`main`函数（因为程序的入口只能有一个）

**第一次写代码，一些常见的错误总结：**

- `main` 被写成了 `mian`
- `main`后边的 `()`漏掉了
- 代码中不能使用中文符号，比如括号和分号
- 一条语句结束后，有分号

## 7. printf和库函数

在上面的代码中有一句代码如下：

```
1 printf("hello C\n");
```

代码中使用了 `printf` 函数，实现了在屏幕上的信息的打印。

这里简单的介绍一下 `printf`，`printf` 是一个**库函数**，它的功能是在标准输出设备（一般指屏幕）上进行信息的打印。上面的代码是使用 `printf` 函数打印字符串。只要把想要打印的一串字符放在双引号中并传递给`printf`函数就可以打印。

`printf`函数也可以用来打印其他类型的数据，比如：

```

1 int n = 100;
2 printf("%d\n", n);    //printf打印整型
3 printf("%c\n", 'q');  //printf打印字符
4 printf("%lf\n", 3.14); //printf打印双精度浮点型

```

这里的 `%d` , `%c` 等是**占位符**，会被后边的值替换。（后期课程再介绍）

同时我们在使用库函数的时候，是需要包含头文件的，比如：`printf` 函数需要包含的就是 `stdio.h` 这个头文件，具体的方法就是：

```

1 #include <stdio.h>

```

## 那什么是库函数呢？

为了不再重复实现常见的代码，让程序员提升开发效率，C语言标准规定了一组函数，这些函数再由不同的编译器厂商根据标准进行实现，提供给程序员使用。这些函数组成了一个函数库，被称为**标准库**，这些函数也被称为库函数。在这个基础上一些编译器厂商可能会额外扩展提供部分函数（这些函数其他编译器不一定支持）。

一个系列的库函数一般会声明在同一个头文件中，所以库函数的使用，要包含对应的头文件。

库函数比较多，后期慢慢来介绍，提前了解可参考链接：<https://cplusplus.com/reference/clibrary/>

## 8. 关键字介绍

C语言中有一批保留的名字的符号，比如：`int`、`if`、`return`，这些符号被称为**保留字**或者**关键字**。

- 关键字都有特殊的意义，是保留给C语言使用的
- 程序员自己在创建标识符的时候是不能和关键字重复的
- 关键字也是不能自己创建的。

C语言的32个关键字如下：

```

1 auto break case char const continue default do double else enum
  extern
2 float for goto if int long register return short signed sizeof
  static
3 struct switch typedef union unsigned void volatile while

```

注：在C99标准中加入了 `inline`、`restrict`、`_Bool`、`_Complex`、`_Imaginary` 等关键字。

一些关键字大家可以去了解一下，不过使用最多的还是上面的32个关键字。

注：<https://zh.cppreference.com/w/c/keyword>（C语言关键字的全部介绍）

后期我们讲课的过程中，会慢慢介绍这些常用的关键字的。

## 9. 字符和ASCII编码

在键盘上可以敲出各种字符，如：a，q，@，#等，这些符号都被称为**字符**，C语言中字符是用单引号括起来的，如：'a'，'b'，'@'。

我们知道在计算机中所有的数据都是以二进制的形式存储的，那这些字符在内存中分别以什么样的二进制存储的呢？如果我们每个人自己给这些字符中的每个字符编一个二进制序列，这个叫做**编码**，为了方便大家相互通信，不造成混乱，后来美国国家标准学会（ANSI）出台了一个标准 **ASCII 编码**，C语言中的字符就遵循了 ASCII 编码的方式。



# ASCII 码表

下列码表含有全部 128 个 ASCII 十进制 (**dec**)、八进制 (**oct**)、十六进制 (**hex**) 及字符 (**ch**) 编码。

dec	oct	hex	ch	dec	oct	hex	ch	dec	oct	hex	ch	dec	oct	hex	ch
0	0	00	NUL (空)	32	40	20	(空格)	64	100	40	@	96	140	60	`
1	1	01	SOH (标题开始)	33	41	21	!	65	101	41	A	97	141	61	a
2	2	02	STX (正文开始)	34	42	22	"	66	102	42	B	98	142	62	b
3	3	03	ETX (正文结束)	35	43	23	#	67	103	43	C	99	143	63	c
4	4	04	EOT (传送结束)	36	44	24	\$	68	104	44	D	100	144	64	d
5	5	05	ENQ (询问)	37	45	25	%	69	105	45	E	101	145	65	e
6	6	06	ACK (确认)	38	46	26	&	70	106	46	F	102	146	66	f
7	7	07	BEL (响铃)	39	47	27	'	71	107	47	G	103	147	67	g
8	10	08	BS (退格)	40	50	28	(	72	110	48	H	104	150	68	h
9	11	09	HT (横向制表)	41	51	29	)	73	111	49	I	105	151	69	i
10	12	0a	LF (换行)	42	52	2a	*	74	112	4a	J	106	152	6a	j
11	13	0b	VT (纵向制表)	43	53	2b	+	75	113	4b	K	107	153	6b	k
12	14	0c	FF (换页)	44	54	2c	,	76	114	4c	L	108	154	6c	l
13	15	0d	CR (回车)	45	55	2d	-	77	115	4d	M	109	155	6d	m
14	16	0e	SO (移出)	46	56	2e	.	78	116	4e	N	110	156	6e	n
15	17	0f	SI (移入)	47	57	2f	/	79	117	4f	O	111	157	6f	o
16	20	10	DLE (退出数据链)	48	60	30	0	80	120	50	P	112	160	70	p
17	21	11	DC1 (设备控制1)	49	61	31	1	81	121	51	Q	113	161	71	q
18	22	12	DC2 (设备控制2)	50	62	32	2	82	122	52	R	114	162	72	r
19	23	13	DC3 (设备控制3)	51	63	33	3	83	123	53	S	115	163	73	s
20	24	14	DC4 (设备控制4)	52	64	34	4	84	124	54	T	116	164	74	t
21	25	15	NAK (反确认)	53	65	35	5	85	125	55	U	117	165	75	u
22	26	16	SYN (同步空闲)	54	66	36	6	86	126	56	V	118	166	76	v
23	27	17	ETB (传输块结束)	55	67	37	7	87	127	57	W	119	167	77	w
24	30	18	CAN (取消)	56	70	38	8	88	130	58	X	120	170	78	x
25	31	19	EM (媒介结束)	57	71	39	9	89	131	59	Y	121	171	79	y
26	32	1a	SUB (替换)	58	72	3a	:	90	132	5a	Z	122	172	7a	z
27	33	1b	ESC (退出)	59	73	3b	;	91	133	5b	[	123	173	7b	{
28	34	1c	FS (文件分隔符)	60	74	3c	<	92	134	5c	\	124	174	7c	
29	35	1d	GS (组分分隔符)	61	75	3d	=	93	135	5d	]	125	175	7d	}
30	36	1e	RS (记录分隔符)	62	76	3e	>	94	136	5e	^	126	176	7e	~
31	37	1f	US (单元分隔符)	63	77	3f	?	95	137	5f	_	127	177	7f	DEL (删除)

ASCII编码表

参考：<https://zh.cppreference.com/w/cpp/language/ascii>

我们不需要记住所有的ASCII码表中的数字，使用时查看就可以，不过我们最好能掌握几组特殊的数据：

- 字符A~Z的ASCII码值从65~90
- 字符a~z的ASCII码值从97~122
- 对应的大小写字符(a和A)的ASCII码值的差值是32
- 数字字符0~9的ASCII码值从48~57
- 换行 `\n` 的ASCII值是：10
- 在这些字符中ASCII码值从0~31 这32个字符是不可打印字符，无法打印在屏幕上观察

单个字符的打印可以使用%c来指定格式：

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("%c\n", 'Q');
6     printf("%c\n", 81); //这里的81是字符Q的ASCII码值，也是可以正常打印的
7     return 0;
8 }
```

可打印字符展示：

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int i = 0;
6     for (i = 32; i <= 127; i++)
7     {
8         printf("%c ", i);
9         if (i % 16 == 15)
10             printf("\n");
11     }
12     return 0;
13 }
```

```
! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | }
```

## 10. 字符串和\0

C语言中如何表示字符串呢？使用双引号括起来的一串字符就被称为字符串，如："abcdef"，就是一个字符串。

字符串的打印格式可以使用 `%s` 来指定，也可以直接打印如下：

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("%s\n", "hello C");
6     printf("hello c");
7     return 0;
8 }
```

C语言字符串中一个特殊的知识，就是在字符串的末尾隐藏放着一个 `\0` 字符，这个 `\0` 字符是字符串的结束标志。

比特就业课主页：https://m.cctalk.com/inst/s9yewhfr

监视 1

搜索 (Ctrl+E) 搜索深度: 3

名称	值	类型
"abcdef"	"abcdef"	char[7]
[0]	97 'a'	char
[1]	98 'b'	char
[2]	99 'c'	char
[3]	100 'd'	char
[4]	101 'e'	char
[5]	102 'f'	char
[6]	0 '\0'	char
添加要监视的项		

VS2022的监视窗口观察字符串

对于字符串"abcdef"，我们实际上看到了6个字符：a,b,c,d,e,f，但是实际上在末尾还隐藏一个 `\0` 的转义字符，`\0` 是字符串的结束标志。所以我们在使用库函数 `printf()` 打印字符串或者 `strlen()` 计算字符串长度的时候，遇到 `\0` 的时候就自动停止了。

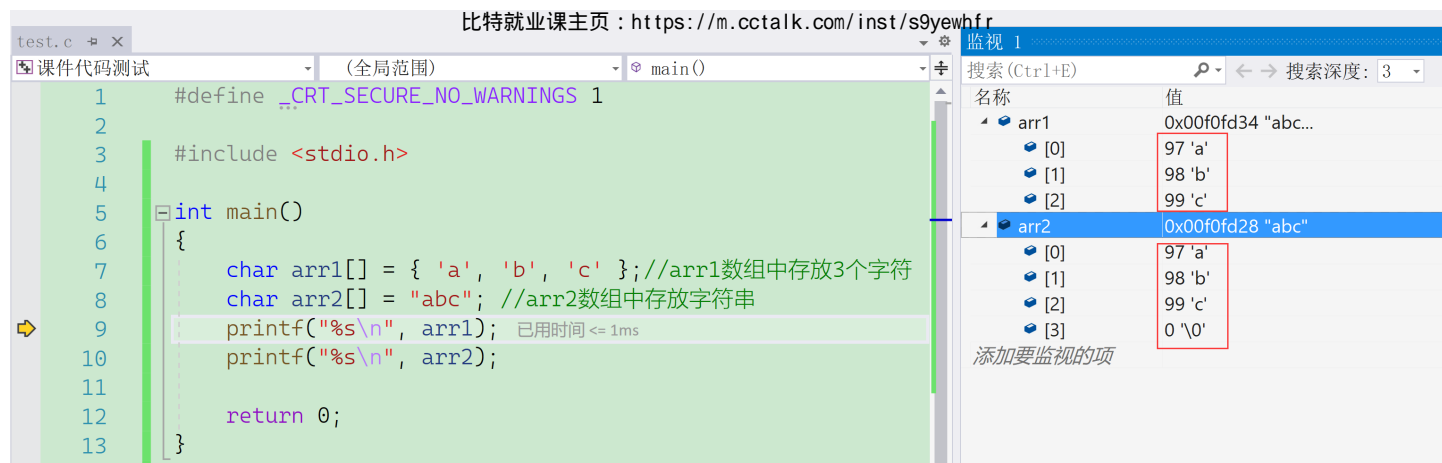
C语言中也可以把一个字符串放在一个字符数组中，我们在这里利用下面的代码验证一下 `\0` 的功能。

```

1 #include <stdio.h>
2
3 int main()
4 {
5     char arr1[] = {'a', 'b', 'c'}; //arr1数组中存放3个字符
6     char arr2[] = "abc"; //arr2数组中存放字符串
7     printf("%s\n", arr1);
8     printf("%s\n", arr2);
9
10    return 0;
11 }

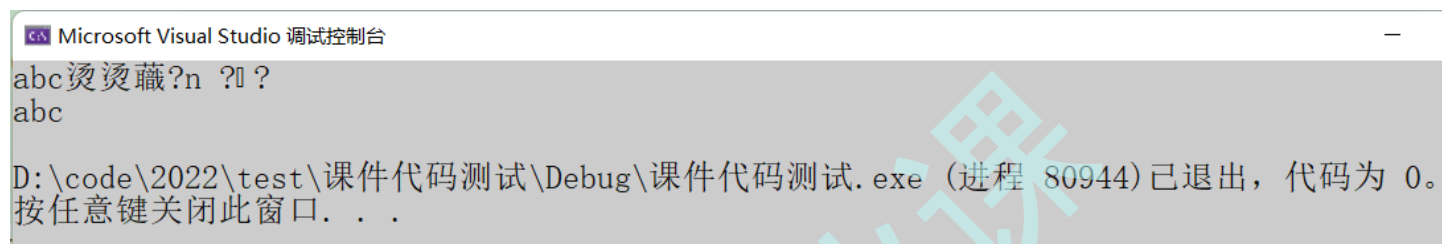
```

这样的代码，我调试的时候，观察一下 `arr1` 和 `arr2` 的内容：



arr1和arr2中内容的对比

运行结果：



我们可以看到，arr1 字符数组在打印的时候，打印了 a、b、c 后还打印了一些随机值，这就是因为 arr1 在末尾的地方没有 \0 字符作为结束标志，在打印的时候没有停止。

但是 arr2 的打印就是完全正常的，就是因为 arr2 数组是使用字符串常量初始化的，数组中有 \0 作为结束标志，打印可以正常停止。

如果我们在arr1数组中单独放一个 '\0' 字符会怎么样呢？

```
1 #include <stdio.h>
2
3 int main()
4 {
5     char arr1[] = {'a', 'b', 'c', '\0'};
6     char arr2[] = "abc";
7     printf("%s\n", arr1);
8     printf("%s\n", arr2);
9     printf("%s\n", "abc\0def");
10    return 0;
11 }
```

```
abc
abc
abc
```

看到三次打印的结果是一样的了，都是打印到 `\0` 的时候就停止了，那从上述的例子我们确实能够观察到 `\0` 的作用和重要性的。

## 11. 转义字符

也许在前面的代码中你看到 `\n`, `\0` 很纳闷是啥。其实在字符中有一组特殊的字符是**转义字符**，转义字符顾名思义：转变原来的意思的字符。

比如：我们有字符 `n`，在字符串中打印的时候自然能打印出这个字符，如下：

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("abcndef");
6     return 0;
7 }
```

输出的结果：

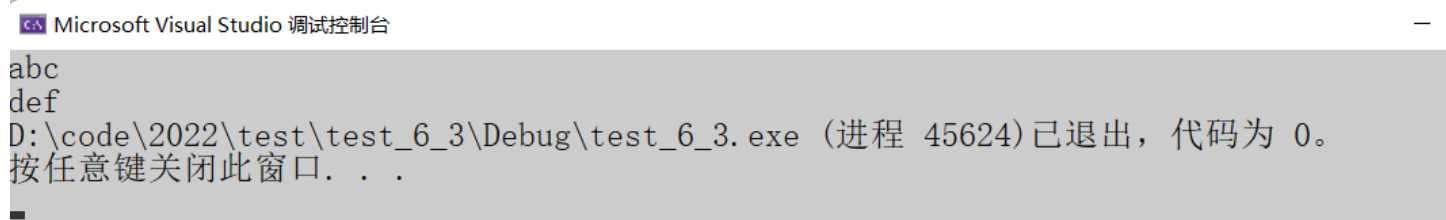
Microsoft Visual Studio 调试控制台

abcndef

D:\code\2022\test\test\_6\_3\Debug\test\_6\_3.exe (进程 52864) 已退出，代码为 0。  
按任意键关闭此窗口. . .

如果我们修改一下代码，在 `n` 的前面加上 `\`，变成如下代码：

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("abc\\ndef");
6     return 0;
7 }
```

The screenshot shows the Microsoft Visual Studio Debug Console. It displays the output of a program: 'abc' followed by 'def' on a new line. Below this, it states 'D:\code\2022\test\test\_6\_3\Debug\test\_6\_3.exe (进程 45624) 已退出，代码为 0。' and '按任意键关闭此窗口。 . . .'. The console has a dark background with light-colored text.

我们可以看到修改的前后代码输出的结果，截然不同的，那这是为什么呢？

这就是转义字符的问题，`\n` 是一个转义字符表示**换行**的意思，我们可以简单的理解为 `\` 让 `n` 的意思发生了转变，`n` 本来是一个普通的字符，被 `\` 转义为换行的意思。

C语言中像这样的转义字符还有一些，具体如下：

- `\?`：在书写连续多个问号时使用，防止他们被解析成三字母词，在新的编译器上没法验证了。
- `\'`：用于表示字符常量'
- `\"`：用于表示一个字符串内部的双引号
- `\\`：用于表示一个反斜杠，防止它被解释为一个转义序列符。
- `\a`：警报，这会使得终端发出警报声或出现闪烁，或者两者同时发生。
- `\b`：退格键，光标回退一个字符，但不删除字符。
- `\f`：换页符，光标移到下一页。在现代系统上，这已经反映不出来了，行为改成类似于 `\v`。
- `\n`：换行符。
- `\r`：回车符，光标移到同一行的开头。
- `\t`：制表符，光标移到下一个水平制表位，通常是下一个4/8的倍数。
- `\v`：垂直分隔符，光标移到下一个垂直制表位，通常是下一行的同一列。

下面2种转义字符可以理解为：字符的8进制或者16进制表示形式

- `\ddd`：d d d表示1~3个八进制的数字。 如： `\130` 表示字符X
- `\xdd`：d d表示2个十六进制数字。 如： `\x30` 表示字符0

`\0`：null 字符，代表没有内容，`\0` 就是 `\ddd` 这类转义字符的一种，用于字符串的结束标志，其ASCII码值是0。

代码演示：

```
1 #include <stdio.h>
2
3 int main()
4 {
```

```

5     printf("%c\n", '\x');
6     printf("%s\n", "\\");
7     printf("c:\\test\\code\\test.c\n");
8     printf("a");
9     printf("%c\n", '\130'); //130是8进制, 转换成10进制是88, 以88作为ASCII码值的字符
    是'X'
10    printf("%c\n", '\x30'); //x30中的30是16进制, 转换成10进制是48, 以48作为ASCII码值
    的字符是'o'
11
12    return 0;
13 }
```

这些ASCII码值是可以自己写代码验证的, 大家也可以自己验证。

关于转义字符我们首先要了解, 然后要能在字符串中识别出来。

转义字符参考: <https://zh.cppreference.com/w/c/language/escape>

## 12. 语句和语句分类

C语言的代码是由一条一条的**语句**构成的, C语言中的语句可为以下五类:

- 空语句
- 表达式语句
- 函数调用语句
- 复合语句
- 控制语句

### 12.1 空语句

空语句是最简单的, 一个分号就是一条语句, 是空语句。

```

1  #include <stdio.h>
2  int main()
3  {
4      ;//空语句
5      return 0;
6  }
```



空语句，一般出现的地方是：这里需要一条语句，但是这个语句不需要做任何事，就可以写一个空语句。

## 12.2 表达式语句

表达式语句就是在表达式的后边加上分号。如下所示：

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int a = 20;
6     int b = 0;
7     b = a + 5; //表达式语句
8     return 0;
9 }
```

## 12.3 函数调用语句

函数调用的时候，也会加上分号，就是函数调用语句。

```
1 #include <stdio.h>
2
3 int Add(int x, int y)
4 {
5     return x+y;
6 }
7
8 int main()
9 {
10     printf("hehe\n");//函数调用语句
11     int ret = Add(2, 3);//函数调用语句
12     return 0;
13 }
```

## 12.4 复合语句

复合语句其实就是前面讲过的代码块，成对括号中的代码就构成一个代码块，也被称为复合语句。

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

```

2
3 void print(int arr[], int sz) //函数的大括号中的代码也构成复合语句
4 {
5     int i = 0;
6     for(i=0; i<sz; i++)
7     {
8         printf("%d ", arr[i]);
9     }
10 }
11
12 int main()
13 {
14     int i = 0;
15     int arr[10] = {0};
16     for(i=0; i<10; i++) //for循环的循环体的大括号中的就是复合语句
17     {
18         arr[i] = 10-i;
19         printf("%d\n", arr[i]);
20     }
21     return 0;
22 }

```

## 12.5 控制语句

**控制语句**用于控制程序的执行流程，以实现程序的各种结构方式（C语言支持三种结构：顺序结构、选择结构、循环结构），它们由特定的语句定义符组成，C语言有**九种控制语句**。

可分成以下三类：

1. **条件判断语句也叫分支语句**：if语句、switch语句；
2. **循环执行语句**：do while语句、while语句、for语句；
3. **转向语句**：break语句、goto语句、continue语句、return语句。

后期会给大家一一介绍控制语句。

## 13. 注释是什么？为什么写注释？

注释是对代码的说明，编译器会忽略注释，也就是说，注释对实际代码没有影响。

注释是给程序员自己，或者其他程序员看的。

好的注释可以帮我们更好的理解代码，但是也不要过度注释，不要写没必要的注释。

当然不写注释可能会让后期阅读代码的人抓狂。

写注释一定程度上反应了程序作者的素质，建议大家写必要的注释，在未来找工作的时候，写代码时留下必要的注释也会给面试官留下更好的印象。

## 13.1 注释的2种形式

C 语言的注释有两种表示方法。

### 13.1.1 `/**/` 的形式

第一种方法是将注释放在 `/*...*/` 之间，内部可以分行。

```
1  /* 注释 */
2
3  /*
4     这是一行注释
5  */
```

这种注释可以插在行内。

```
1  int fopen(char* s /* file name */, int mode);
```

上面示例中，`/* file name */` 用来对函数参数进行说明，跟在它后面的代码依然会有效执行。

这种注释一定不能忘记写结束符号 `*/`，否则很容易导致错误。

```
1  printf("a "); /* 注释一
2  printf("b ");
3  printf("c "); /* 注释二 */
4  printf("d ");
```

上面示例的原意是，第一行和第三行代码的尾部，有两个注释。

但是，第一行注释忘记写结束符号，导致注释一延续到第三行结束。

`/**/` 的这个注释也不支持嵌套注释，`/*` 开始注释后，遇到第一个 `*/` 就认为注释结束了。

```
1  /*
2  printf("a ");
3  printf("b ");
4  printf("c "); /* 注释二 */
5  printf("d ");
```

### 13.1.2 // 的形式

第二种写法是将注释放在双斜杠 `//` 后面，从双斜杠到行尾都属于注释。这种注释只能是单行，可以放在行首，也可以放在一行语句的结尾。这是 C99 标准新增的语法。

```
1 // 这是一行注释
2
3 int x = 1; // 这也是注释
```

不管是哪一种注释，都不能放在双引号里面。

双引号里面的注释符号，会成为字符串的一部分，解释为普通符号，失去注释作用。

```
1 printf("// hello /* world */ ");
```

上面示例中，双引号里面的注释符号，都会被视为普通字符，没有注释作用。

## 13.2 注释会被替换

编译时，注释会被替换成一个空格，所以 `min/* 这里是注释 */Value` 会变成 `min Value`，而不是 `minValue`。

完