

# 1.1初识C语言

CSDN: <https://blog.csdn.net/wzzzz6423>

Gitee: <https://gitee.com/wzz6423>

Github: <https://github.com/wzz6423>

## 1.1初识C语言

### 目录

1. 什么是C语言
2. C语言的历史
3. 编译器介绍与选择
4. 项目和源文件、头文件
5. 第一个C语言程序
6. main函数
7. 库函数与printf
8. 关键字介绍
9. 字符和ASCII码
10. 字符串与\0
11. 转义字符
12. 语句和语句分类
13. 注释

gitee链接: <https://gitee.com/wzz6423>

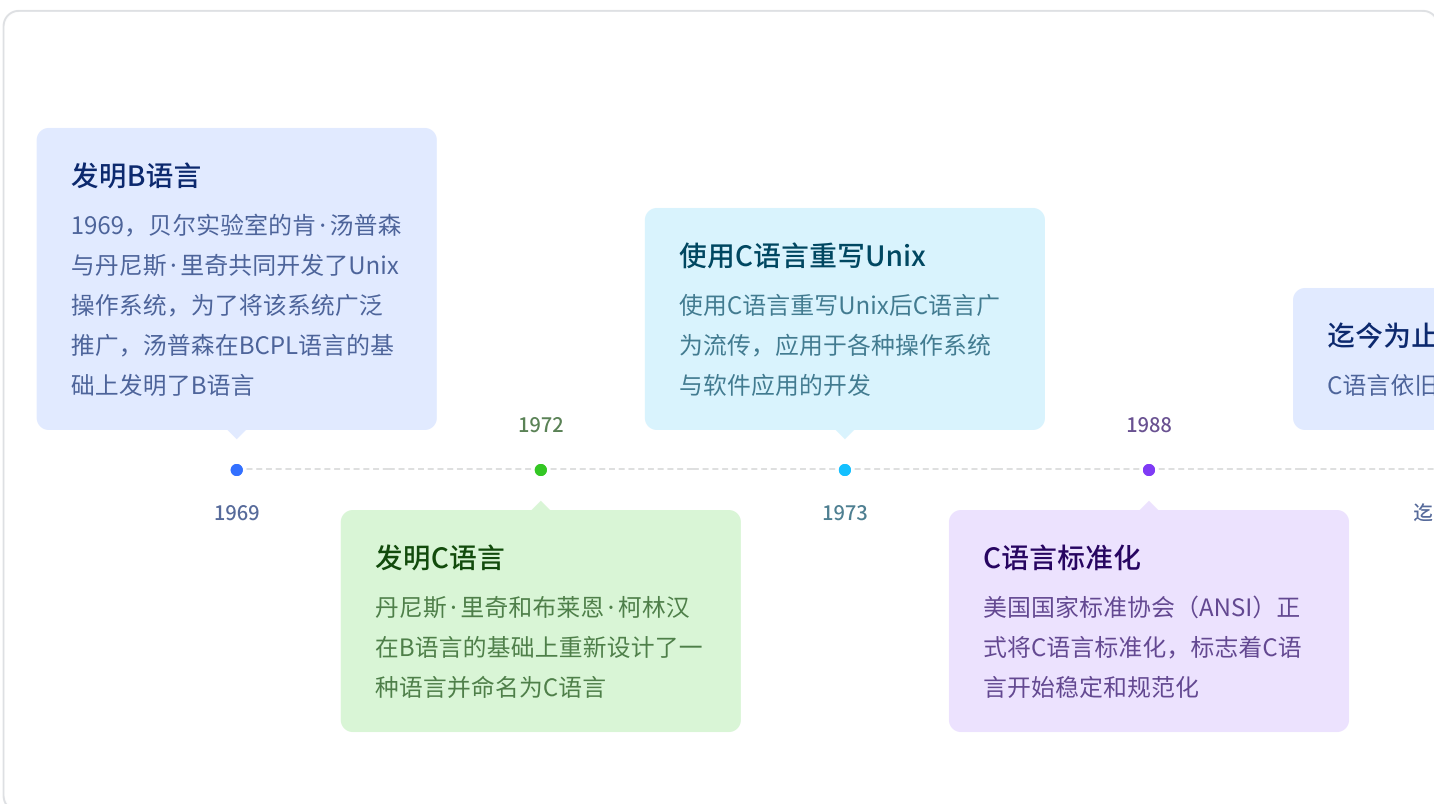
github链接: <https://github.com/wzz6423>

## 1. 什么是C语言

- C语言是一种较早的程序设计语言, 诞生于1972年的贝尔实验室。1972年, Dennis Ritchie 设计了C语言, 它继承了B语言的许多思想, 并加入了数据类型的概念及其他特性。尽管C语言是与UNIX操作系统一起被开发出来的, 但它不只支持UNIX。C是一种通用(广泛可用)的编程语言。
- 广泛应用于底层开发。C语言能以简易的方式编译、处理低级存储器。C语言是仅产生少量的机器语言以及不需要任何运行环境支持便能运行的高效率程序设计语言。尽管C语言提供了许多低级处理的功能, 但仍然保持着跨平台的特性, 以一个标准规格写出的C语言程序可在包括类似嵌入式处理器以及超级计算机等作业平台的许多计算机平台上进行编译。
- **总结:** C语言是一种 人与计算机进行交流 用来写程序、给计算机下达指令、让计算机工作 的一种语言

## 2. C语言的历史

- C语言之父: 丹尼斯·里奇

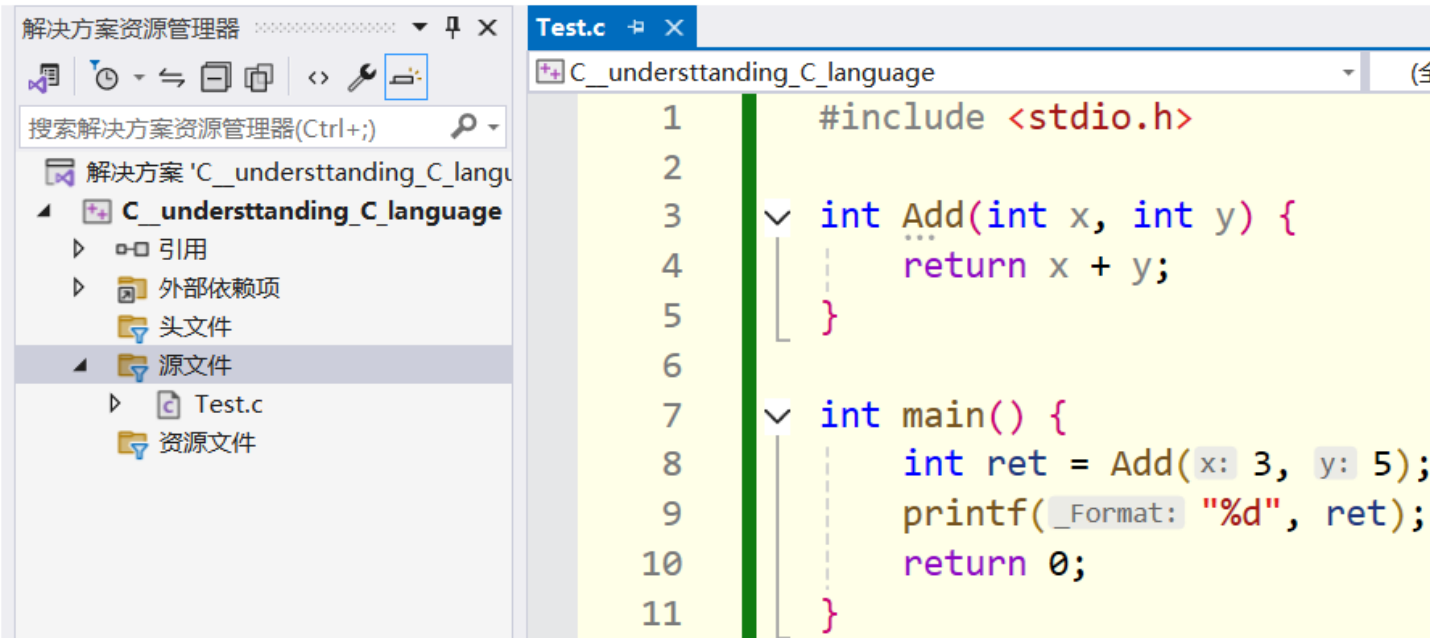


计算机语言排行榜: <https://www.tiobe.com/tiobe-index/>

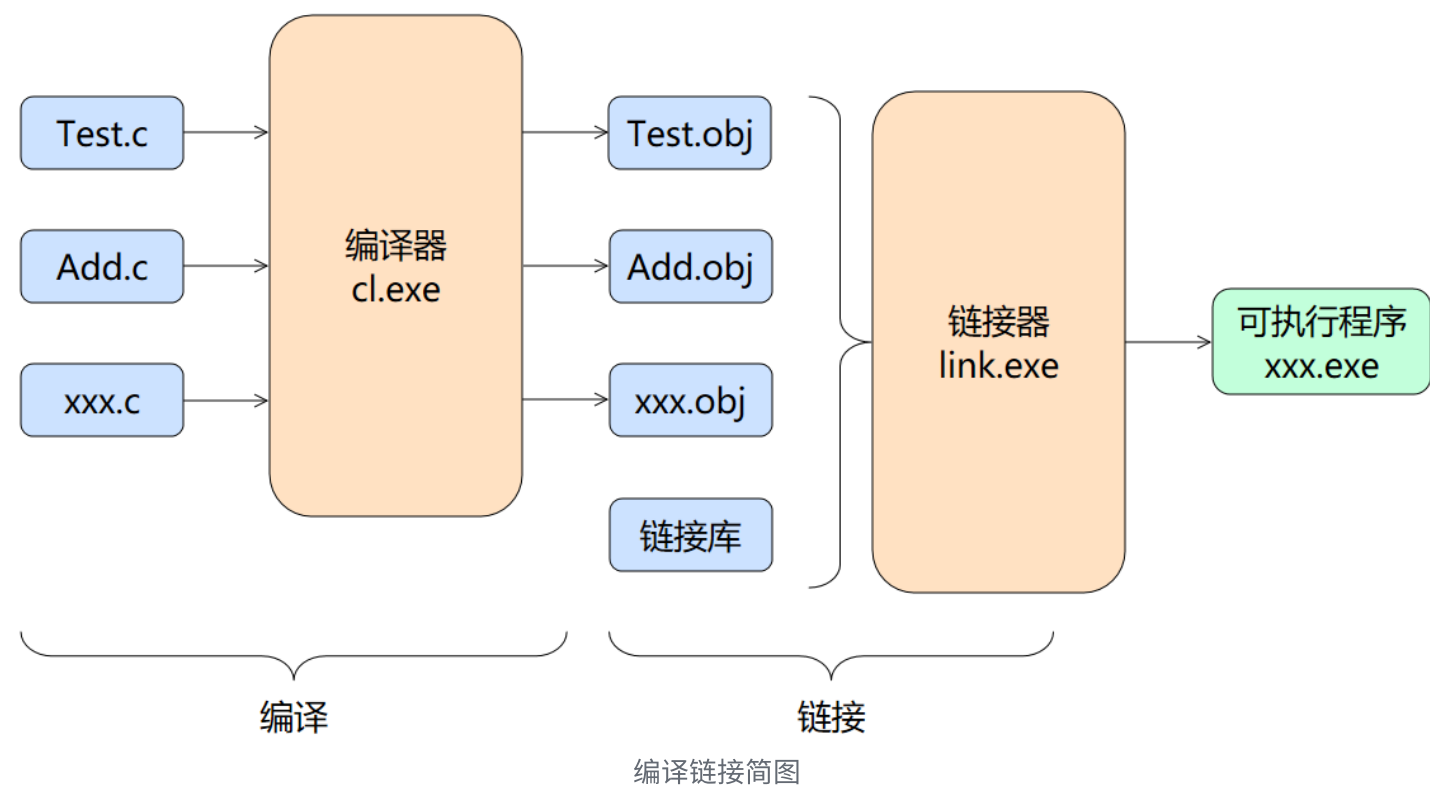
### 3. 编译器介绍与选择

#### 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**才能执行
- C语言代码放在以**.c**为后缀的文件中，要得到最终可以运行的可执行程序中间需要经过**编译**和**链接**2个过程



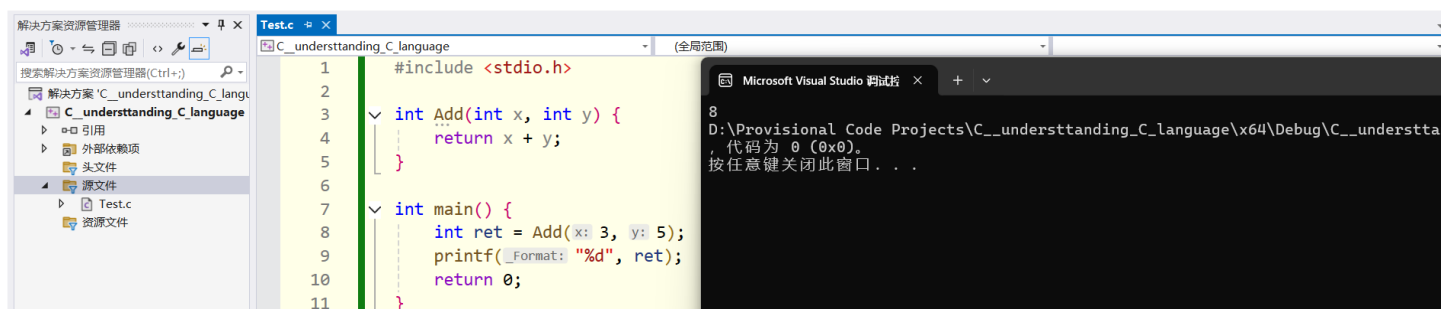
C语言源文件（Test.c）及代码



- 注意：（以windows系统下的文件后缀为例）
  - 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）
  - 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文件（.exe）

 C\_understanding\_C\_language.exe      2025/2/11 18:25      应用程序      62 KB

形成的可执行文件




形成的可执行文件运行结果

## 3.2 编译器

- 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及其指令
- 常见的C语言编译器：MSVC、Clang、gcc
- 常见的集成开发环境：Visual Studio2022、Clion、XCode、DevC++、小熊猫C++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、图形化用户界面等工具。集成了代码编写功能、分析功能、编译功能、调试功能等于一体的开发软件和服务套
- 编辑器不是IDE，编辑器仅具备文本编辑功能（如：记事本等）

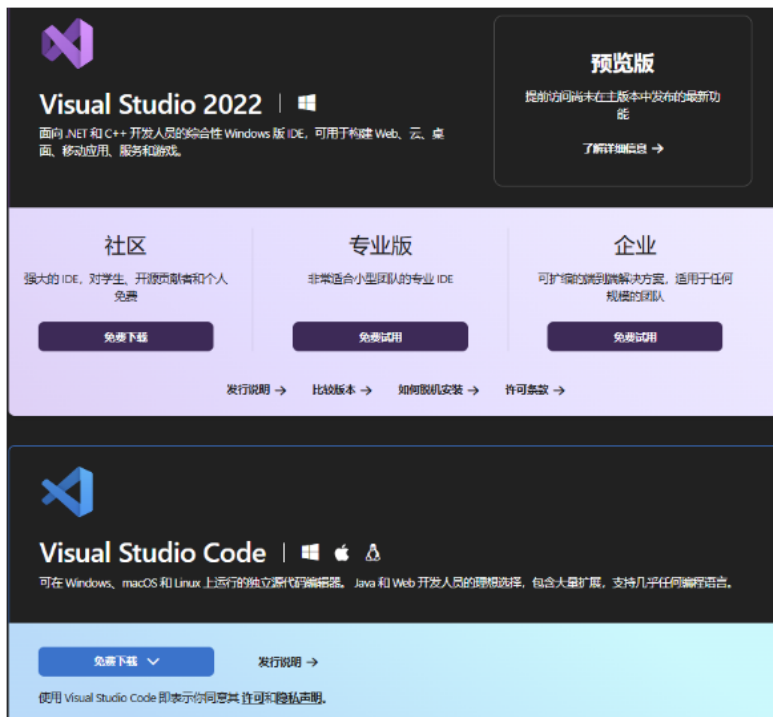
 Visual Studio2022：集成MSVC（**最推荐初学者使用**的IDE、安装包较大、使用简单、方便）  
（后续大部分样例使用的IDE是VS2022）

Clion：集成Clang，默认使用CMake（后续样例中会使用这个IDE、并简单介绍CMake、但初学者不建议使用、存储空间足够可以先安装上）

DevC++：集成gcc（小巧、竞赛用、日常不建议、对学习与工作使用不友好）

XCode：集成Clang（Mac上使用、在appstore直接下载即可）

小熊猫C++：DevC++的国内魔改版，内置easyx库，网络上一些比较好玩的代码可以用这个跑（很火的爱心代码等等，用其它IDE需要自行安装edasyx库）



上述工具都有免费版和付费版，下载免费版使用即可（功能足够用了）



正版下载链接：

- 火绒：<https://www.huorong.cn/>（火绒自带的应用商店可以解决所有的问题）
- Visual Studio2022：<https://visualstudio.microsoft.com/zh-hans/downloads/>
- Clion：<https://www.jetbrains.com.cn/clion/>
- DevC++：<https://sourceforge.net/projects/orwelldevcpp/>
- XCode：苹果appstore
- 小熊猫C++：火绒应用商店下载

#### • 温馨提示

- 今后会提起的一个概念MSVC编译器是不支持的，可以考虑使用Clion或者在VS2022安装时选择上Clang编译器
- VS2022和Clion等安装所需存储空间较大，请自行更改安装地址到其他盘（不要安装在C盘）
- 安装路径（文件夹的名字）不要出现中文及特殊字符，普普通通的全英文就好，也不要吧路径安装的太深

- d. Visual Studio Code不是Visual Studio2022，不要安装错了，但是也可以安装VSCode，是一个功能非常强大的文本编辑器，支持许多插件，在后续学习Linux连接远程服务器、编辑论文配置Latex、查看源码等许多方面有较大的优势
- e. 初学者非常不建议使用Visual Studi Code自己配置环境或者使用Linux服务器，没有必要，只会增加上手难度



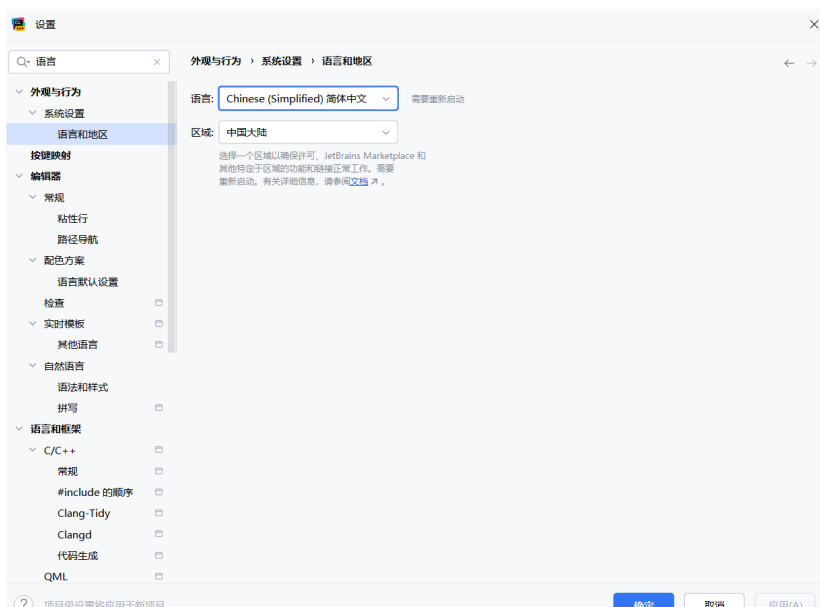
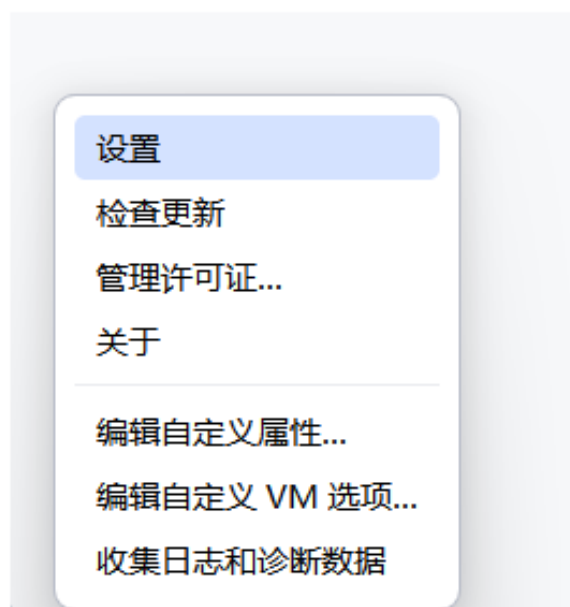
左侧选择“使用C++的桌面开发”，右侧自行选择下载Clang编译器（其余不用管）





- tips:
  - C++ (Cpp) 是后续以C语言为基础发展出来的OOP（面向对象）的语言，C++语法兼容C语言，因此上面选择的是“使用C++的桌面开发”
  - 同样学习C语言的过程中把源文件后缀使用.cpp也是可以正常使用的（但是在一些细节上有一定差异，建议现在就写成.c即可）
  - 其他编译器的安装方法差异不大甚至更加简单这里不多赘述
  - 部分IDE默认语言是英文，如果不熟悉可以自行更改语言为中文（VS2022安装语言选择中文后打开时应该就是中文了）

- Clion改中文：

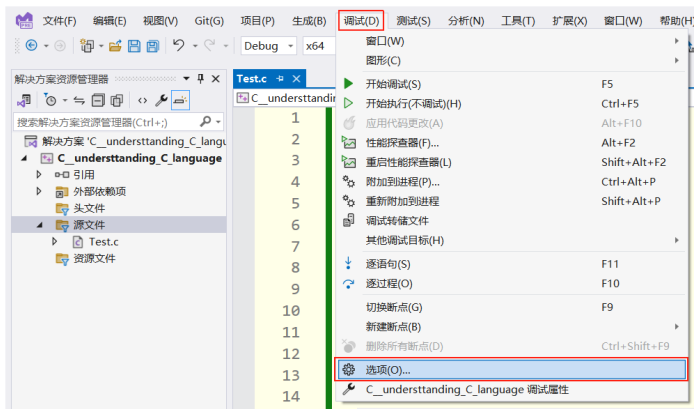


1.左下角设置，最上面设置（setting）

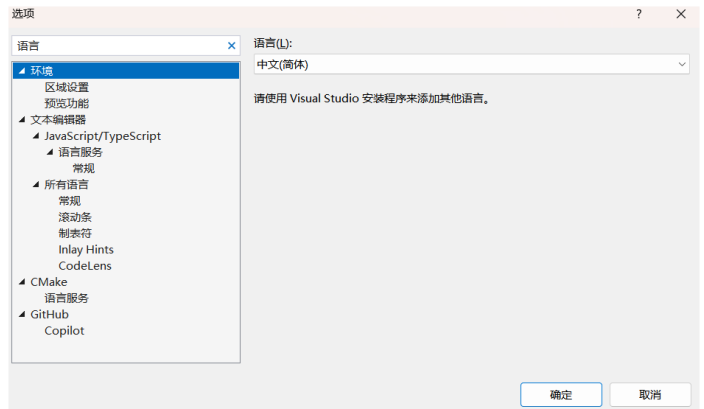
2.直接搜索语言（language）进行更改

- VS2022改中文：





1.最上方工具栏第7个，倒数第二个选项



2.直接搜索语言（language）进行更改

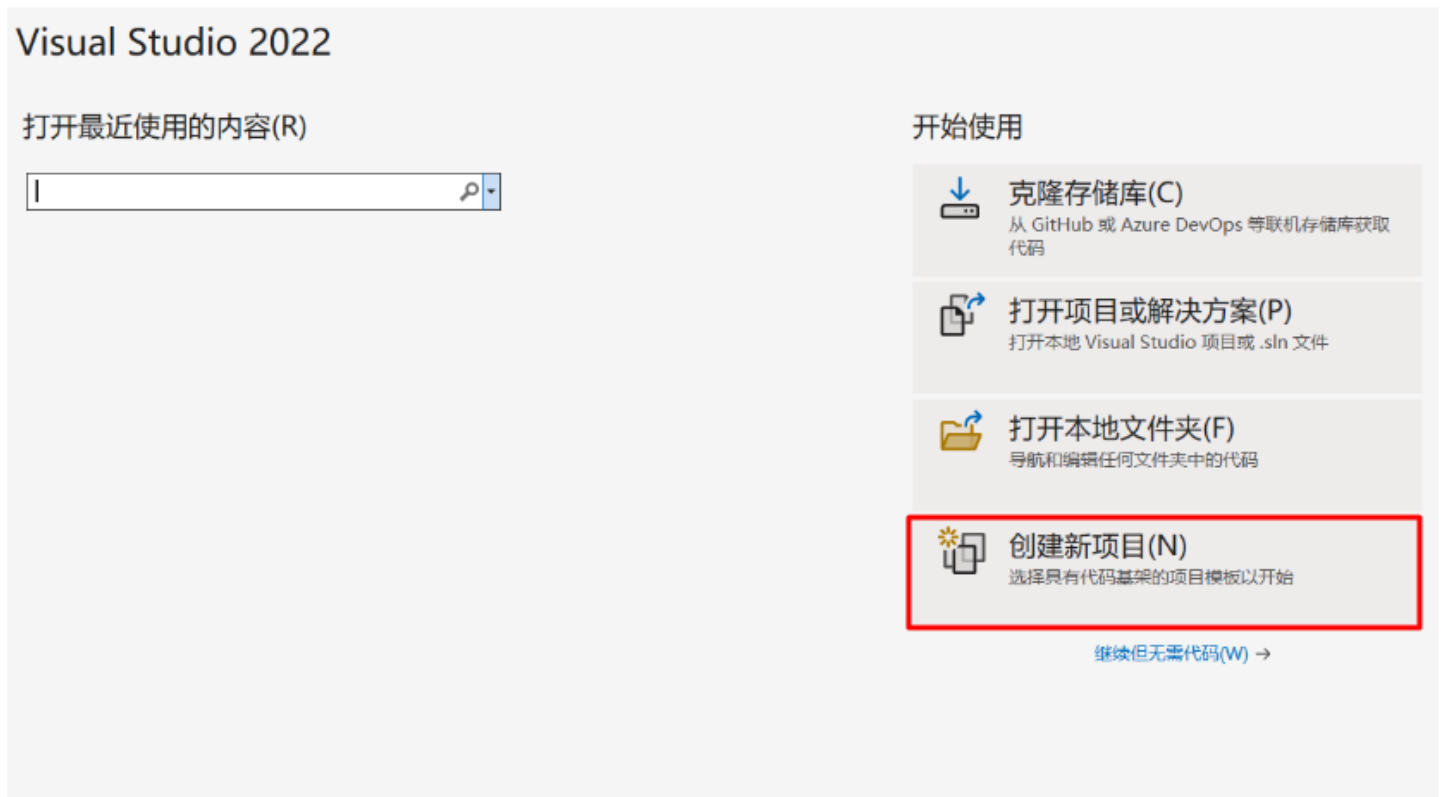
## 4. 项目和源文件、头文件

### 4.1 文件

- C/Cpp项目中将以.c/.cpp结尾的文件称为源文件，.h结尾的文件称为头文件（cpp中还会有.hpp结尾的文件，本质也是头文件，后续会讲，C语言用不到）

### 4.2 在VS上写代码需要创建项目

- 打开VS2022
- 选择“创建新项目”



- 选择“空项目”，“下一步”



## 创建新项目

### 最近使用的项目模板(R)

空项目

C++

搜索模板(Alt+S)(S)

所有语言(L)

所有平台(P)

所有项目类型(T)



空项目

使用适用于 Windows 的 C++ 从头开始操作。不提供基础文件。

C++

Windows

控制台



控制台应用

在 Windows 终端运行代码。默认打印 "Hello World"。

C++

Windows

控制台



CMake 项目

生成不依赖于 .sln 或 .vcxproj 文件的新式跨平台 C++ 应用。

C++

Windows

Linux

控制台



Windows 桌面向导

使用向导自行创建 Windows 应用。

C++

Windows

桌面

控制台

库



Windows 桌面应用程序

具有在 Windows 上运行的图形用户界面的应用程序的项目。

C++

Windows

桌面



Blank TypeScript Project

A blank TypeScript Project that can be used as a console application (using

上一步(B)

下一步(N)

- 自己进行命名并修改存储位置，点击“创建”（不要随便命名，建议使用下划线“\_”配合一些关键信息如日期进行组合（后面内容有针对性的直接使用对应名称），同时建议建立一个专门的文件夹保存自己写的所有需要保存的代码）

## 配置新项目

空项目

C++

Windows

控制台

项目名称(J)

Test\_2\_11

位置(L)

D:\Provisional Code Projects

解决方案名称(M) ⓘ

Test\_2\_11

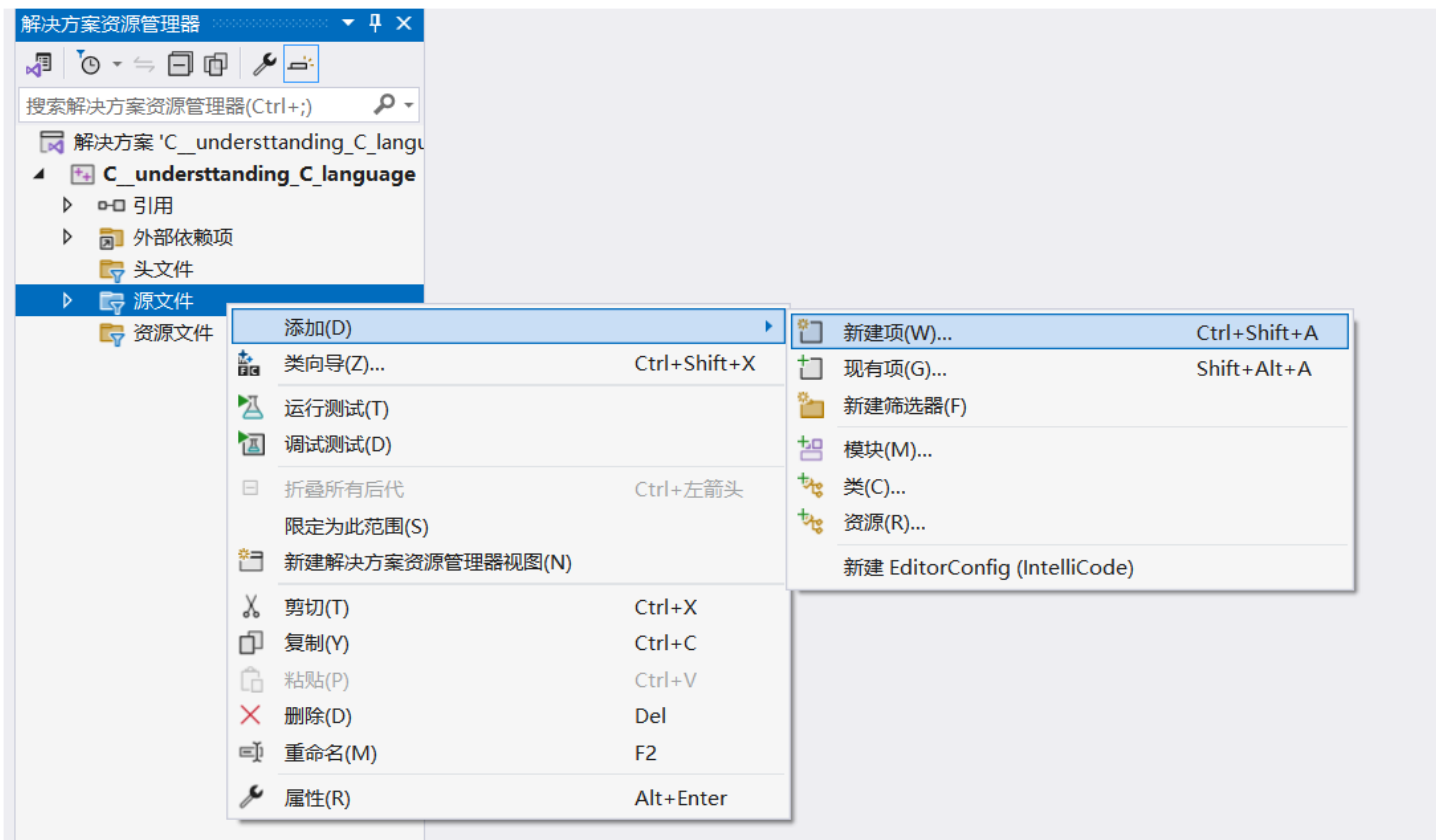
☐ 将解决方案和项目放在同一目录中(D)

项目 将在“D:\Provisional Code Projects\Test\_2\_11\Test\_2\_11\”中创建

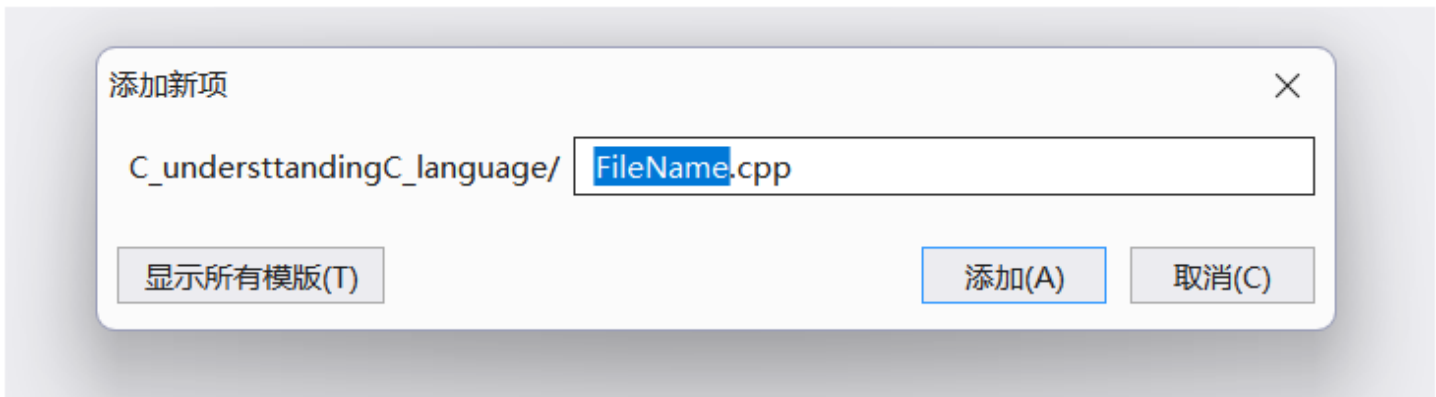
上一步(B)

创建(C)

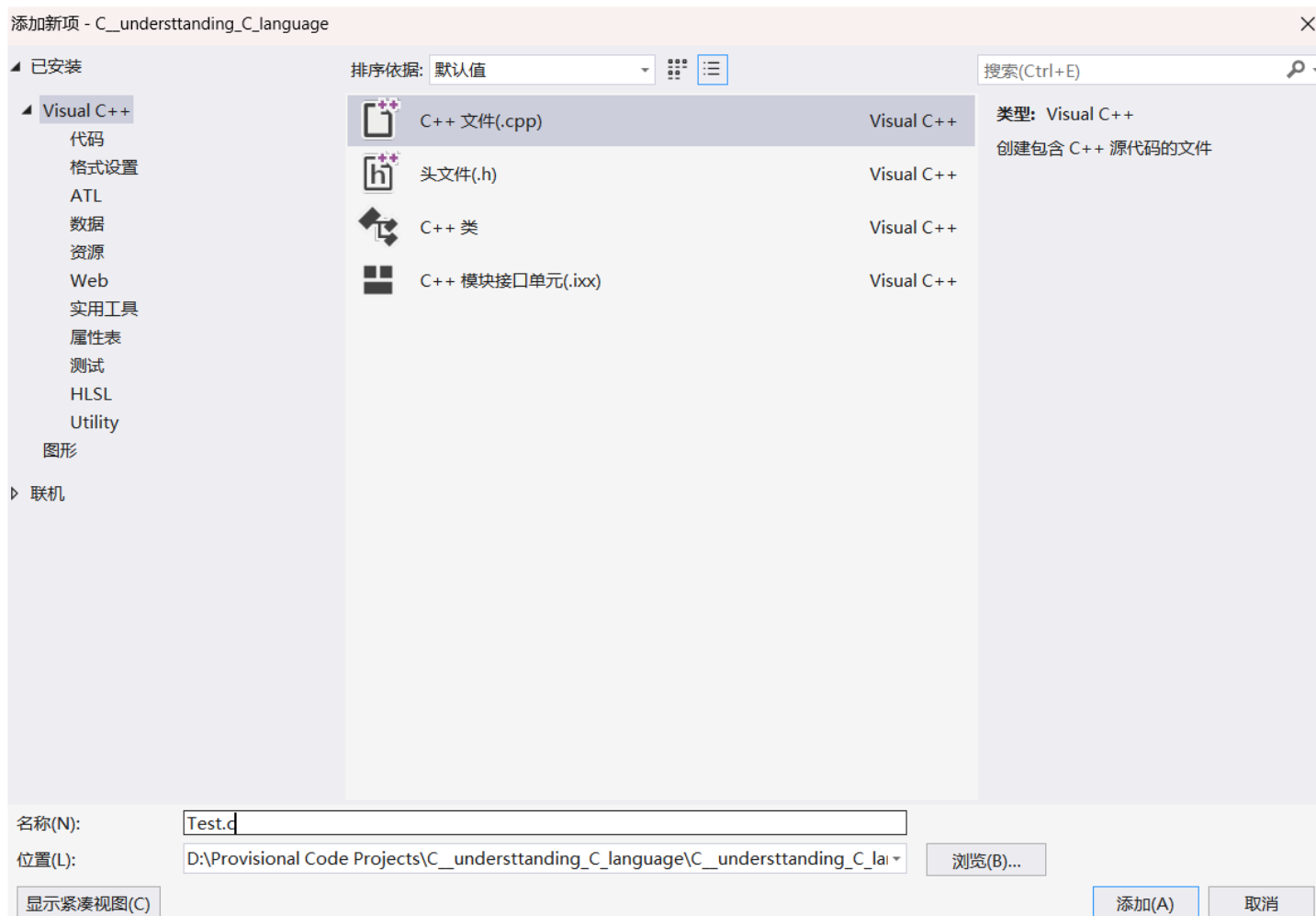
- 右键“源文件”，“添加”，“新建项”（如果创建头文件就在“头文件”上点击右键）



- 选择“显示所有模板”



- 选择“C++文件”（更改名称，后缀以.c结尾），“添加”（创建头文件则选择“头文件（.h）”，后缀不需要自己更改）



## 4.3 打开曾经的项目

- 在VS2022上想要打开曾经的项目不能直接点击源文件/头文件（.c/.cpp/.h）结尾的文件，否则会显示为“杂项文件”且无法正常进行编译
  - 但一个项目中对于程序员真正有价值的只有源文件（.c/.cpp）和头文件（.h），其他的文件程序员是读不懂的、给电脑看的，因此在上传源码/分享给别人的时候仅需上传源文件和头文件

Test.c [X] (全局范围)

1

// 可执行文件及运行结果

2

// #include <stdio.h>

3

//

4

// int Add(int x, int y) {

5

// return x + y;

6

// }

7

//

8

// int main() {

9

// int ret = Add(3, 5);

10

// printf("%d", ret);

11

// return 0;

12

// }

13

14

////////////////////////////////////

15

// 第一个c语言程序

16

// #include <stdio.h>

17

//

18

// int main() {

19

// printf("Hello World\n");

20

// return 0;

21

// }

22

23

////////////////////////////////////

24

// 打印一个字符

25

// #include <stdio.h>

26

//

27

// int main() {

28

// printf("%c\n", 'W');

29

// }

145 % 未找到相关问题 行: 1 字符: 1 制表符 CRLF

新站 添加到源代码管理 选择仓库

杂项文件

x64	2025/2/11 18:25	文件夹		C_under.7081de6c.tlog	2025/2/11 22:06	文件夹	
C_understanding_C_language.vcxproj	2025/2/11 18:25	VC++ Project	7 KB	C_understanding_C_language.exe.r...	2025/2/11 22:06	RECIPE 文件	1 KB
C_understanding_C_language.vcxproj.filters	2025/2/11 20:07	VC++ Project Fil...	1 KB	C_understanding_C_language.ilc	2025/2/11 22:06	Incremental Link...	638 KB
C_understanding_C_language.vcxproj.user	2025/2/11 18:04	USER 文件	1 KB	C_understanding_C_language.log	2025/2/11 22:06	文本文档	1 KB
Test.c	2025/2/24 15:21	C Source	4 KB	Test.obj	2025/2/11 22:06	VisualStudio.obj...	10 KB
				vc143.idb	2025/2/11 22:05	VC++ Minimum ...	27 KB
				vc143.pdb	2025/2/11 22:06	Program Debug ...	76 KB

对程序员无用的其他文件1

对程序员无用的其他文件2

- 想要打开曾经的项目需要在路径下打开.sln结尾的文件（解决方案），一般就存在于该文件夹点进去的一级目录

C_understtanding_C_language	2025/2/24 15:21	文件夹	
x64	2025/2/11 18:25	文件夹	
C_understtanding_C_language.sln	2025/2/11 18:04	Visual Studio Sol...	2 KB

解决方案

# 4.4 添加源文件或头文件到已有项目中

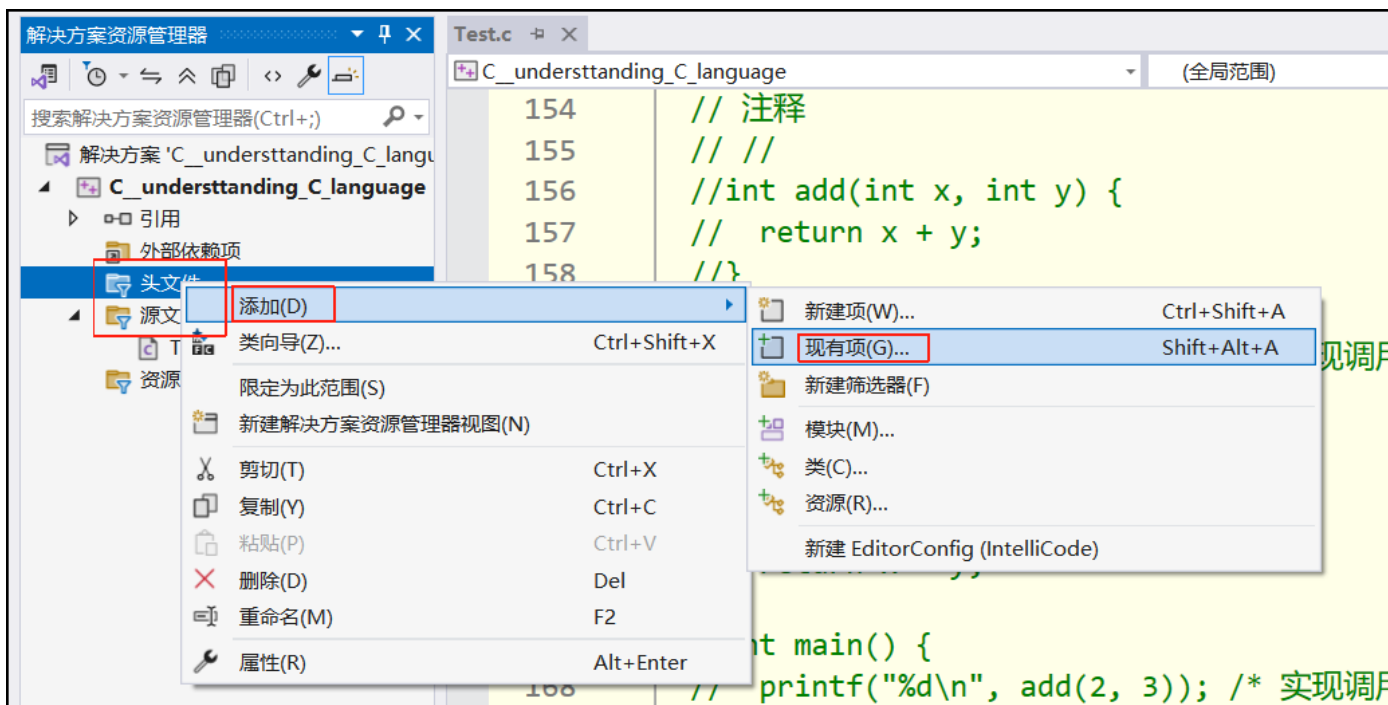
## 1. 将需要添加的源文件或头文件点击复制/剪切

- 打开文件夹并寻找到存放源码的位置（.c/.cpp/.h文件存在的目录），并粘贴

📁 x64	2025/2/11 18:25	文件夹	
📄 C_understtanding_C_language.vcxproj	2025/2/11 18:25	VC++ Project	7 KB
📄 C_understtanding_C_language.vcxproj.filters	2025/2/11 20:07	VC++ Project Fil...	1 KB
📄 C_understtanding_C_language.vcxproj.user	2025/2/11 18:04	USER 文件	1 KB
📄 <b>Test.c</b>	2025/2/24 15:21	C Source	4 KB

源文件/头文件存在的文件夹

## 2. 右键头文件/源文件，添加，现有项



添加现有项

## 3. 选择文件并添加：在弹出的窗口内选择刚刚粘贴的文件并确定即可（也可以省略第1步在此处直接粘贴并选择进行添加）

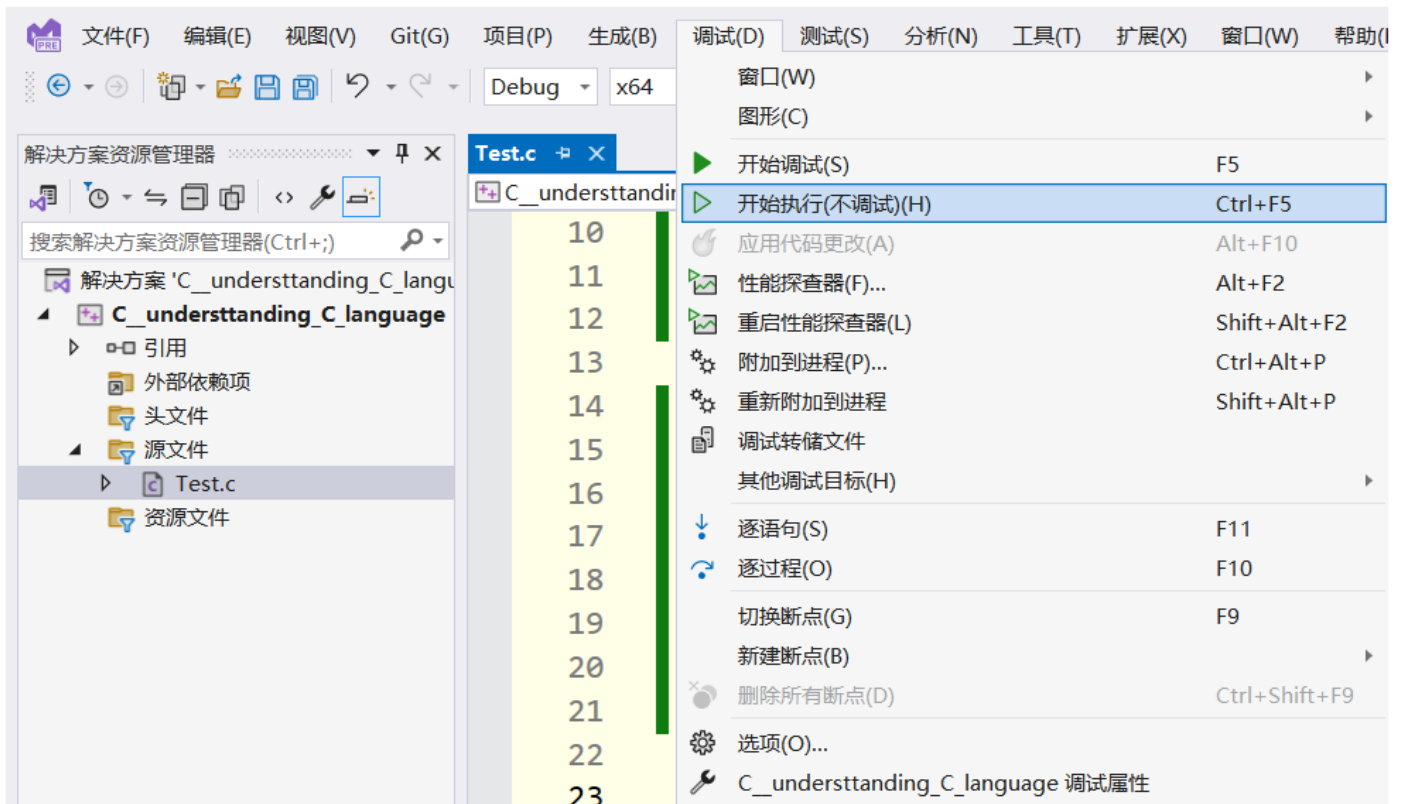
# 5. 第一个C语言程序

- 复制以下代码并粘贴在刚刚建好的源文件内

```

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

- 点击“调试”，“开始执行（不调试）”（快捷键：Ctrl+F5）



- 如果你出现的页面与这张图片相同（出现Hello World以及一行和空行、下方“代码为0”）就证明你的第一个C语言程序顺利运行成功了



## 6. main函数

- 每个C/Cpp项目不管有多少行代码，都是从**main**函数开始执行的，main函数是程序的入口。main函数也被称为“**主函数**”，每个C/Cpp项目只能有一个main函数（无论项目中有多少个.c/.cpp文件）
- main函数前面的int代表main函数运行完毕后会返回一个int类型（整型类型）的值，因此必须在main函数的最后写上return 0;
- 注意！！！所有代码（除了注释-- 本文最后一个板块）都只能使用输入法英文模式下的符号，不可以使用输入法中文模式下的符号，每一条语句的结束都要有分号（;）

## 7. 库函数与printf

- 简要介绍printf函数
  - printf函数是C语言标准库中提供给程序员使用的一个库函数，它的功能是在标准输出设备（显示器）上格式化打印出信息，具体的使用后面会介绍
- 库函数
  - 许多程序员都会有一些相同的需求例如利用程序在显示器上打印一些内容、往磁盘上的某个文件内写入一些数据内容等等。那么让每个程序员都去自己实现一个相关的代码效率非常慢、且没有标准规定有一定的安全隐患且重复冗余。因此C语言标准规定了一些函数，这些函数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。这些函数组成了一些函数库被称为标准库，这些函数称为库函数。
  - 除此之外，部分编译器厂商会扩展一些可以实现其它功能的函数，但是这些函数其他编译器不一定支持，因此使用这些库函数的代码缺失一定的可移植性。
  - 一个系列（功能相似）的函数一般会声明（告诉编译器有这个函数，但是不包括这个功能的实现方法，是一种承诺，实现方法通过其它方式给编译器，后续会讲）在一个头文件内，因此要使用库函数就要引入对应的头文件。
  - 库函数很多，不可能都讲，需要用的时候可以通过文档查找：在线文档，其中包括C/Cpp的库函数
    - <https://cplusplus.com/>
    - <https://en.cppreference.com/w/>
- 头文件
  - 使用库函数需要包含头文件，头文件中包含了例如printf等库函数的声明，如要使用printf这个库函数就要引入stdio.h这个头文件，格式如下：

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

- 使用 #include 代表引入后面的头文件，用 <> 包括住头文件的文件名

---

## 8. 关键字介绍

- 当你想要在代码中声明一个整数并赋值为1的时候怎么办？如果每个人都规定一个标准那肯定不可以。因此C语言中有一批保留名字的符号被称为**保留字/关键字**
  - 关键字有自己特殊的意义，是保留给C语言使用的
  - 程序员在创建标识符的时候不可以和关键字的命名重复



- 关键字程序员不能自己创建
- C语言有32个关键字：

```
1  auto break case char const continue default do double else enum extern
   float for goto if int long register return short signed sizeof struct
   switch typedef union unsigned void volatile while
2
3  // C99添加如下等关键字
4  inline restrict _Bool _Complex _Imaginary
```

- 使用最多的还是上面的32个关键字，不需要记忆，后续用的多了就记住了
- C语言所有关键字介绍：
  - <https://zh.cppreference.com/w/c/keyw>

---

## 9. 字符和ASCII码

- 我们可以在键盘上敲出各种字符如：a、@、#、¥、q、1等，这些符号都被称为字符，C语言中字符是使用单引号'括起来的例如：'a'、'b'、'@'
- 在计算机中所有的数据都是使用二进制的形式存储的（有关于二进制的内容后续会出新的文章来讲），如果每个人都给这些字符编一个二进制序列（编码）用于存储等作用那么大家是无法进行相互之间的交流、通信的。因此美国国家标准学会（ANSI）出台了一个标准**ASCII编码**，C语言中的字符遵循了该编码（后续有关编码的内容也会再次提起几次）
- ASCII码表：
  - <https://zh.cppreference.com/w/cpp/language/ascii>

# 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)

- 大概了解一些常见编码范围：
  - A~Z: 65~90
  - a~z: 97~122
  - 大小写英文字母差值: 32
  - 0~: 48~57
  - 换行符\n: 10

- 注意：ASCII码0~31这32个字符不可打印，无法在屏幕上观察（不可见字符）
- 可以编写代码打印字符：

```
1  #include <stdio.h>
2
3  int main() {
4      printf("%c\n", 'W');
5      printf("%c\n", 87);
6      return 0;
7  }
```

a. %c：占位符，此处为后面的字符'W'占位

- printf打印的时候如果有一些内容是此时不清楚的/不固定的/不想直接写在双引号内部的，可以使用占位符的方式来进行打印
- %d 代替有符号整型类型
- %c 代替字符类型
- %f %d 代替浮点类型（小数，这两个在精度上有区别，后面讲）
- %s 代替字符串类型

b. \n 转义字符，\n代表换行，其余转义字符后续会讲

c. 此处为什么写87也可以正确打印出W呢？

- 此处87是字符W的ASCII码值，可以被转换为字符正常打印



- 打印所有字符：

```
1  #include <stdio.h>
2
3  int main() {
4      for (int i = 32; i <= 127; ++i) {
5          printf("%c ", i);
```

```
6      }
7      printf("\n");
8      return 0;
9  }
```

- 如图



```
35  #include <stdio.h>
36
37  int main() {
38      for (int i = 32; i <= 127; ++i) {
39          printf(_Format: \"%c \", i);
40      }
41      printf(_Format: \"\\n\");
42      return 0;
43  }
```

## 10. 字符串与\0

### 10.1 字符串


- 字符串：字符串是由数字、字母、下划线组成的一串字符
- C语言中：使用双引号括起来的一串字符就被称为字符串，如："abcdefg"
- printf中打印使用%s占位，也可以直接进行打印

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

```
#include <stdio.h>

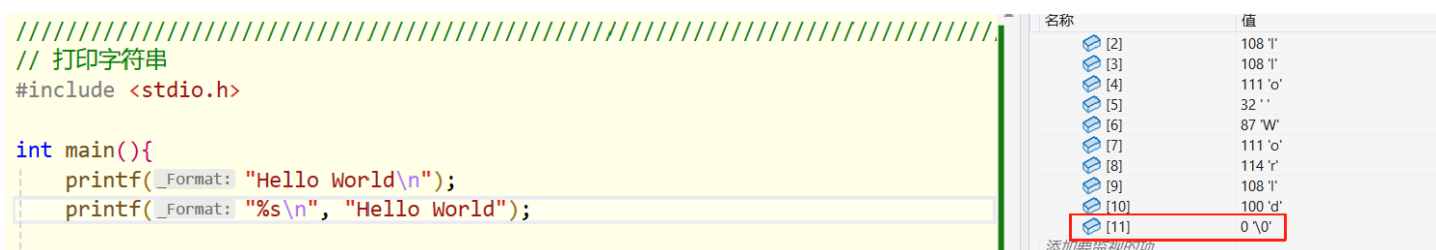
int main(){
    printf(_Format: "Hello World\n");
    printf(_Format: "%s\n", "Hello World");

    return 0;
}
```



## 10.2 '\0'

- 字符串的末尾存放一个 '\0' 字符以标记字符串的结束



```
//////////////////////////////////////
// 打印字符串
#include <stdio.h>

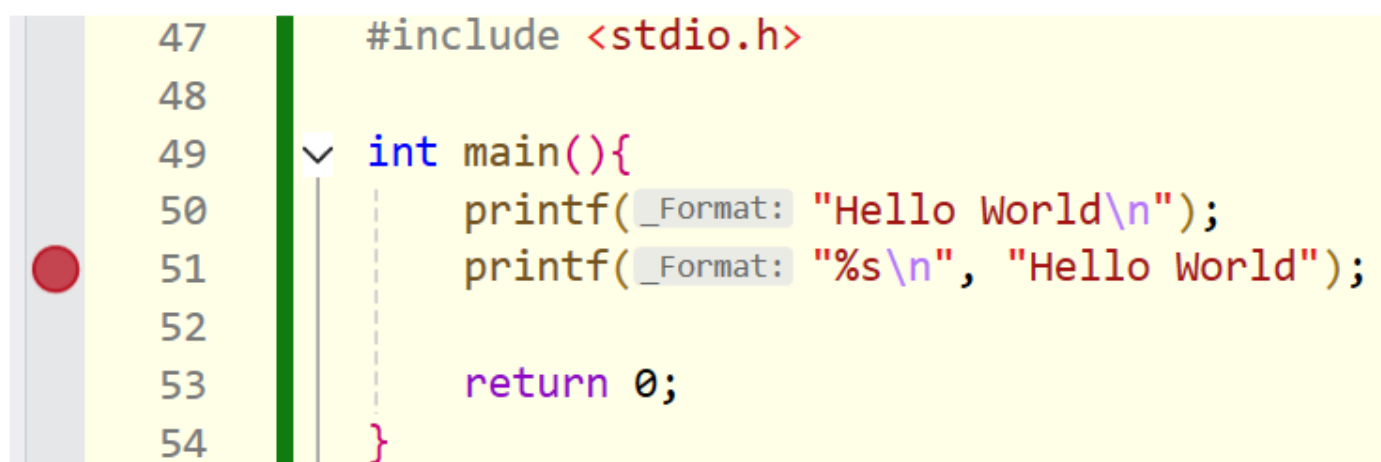
int main(){
    printf(_Format: "Hello World\n");
    printf(_Format: "%s\n", "Hello World");
}
```

名称	值
[2]	108 'I'
[3]	108 'I'
[4]	111 'o'
[5]	32 ''
[6]	87 'W'
[7]	111 'o'
[8]	114 'r'
[9]	108 'I'
[10]	100 'd'
[11]	0 '\0'

添加要监视的项

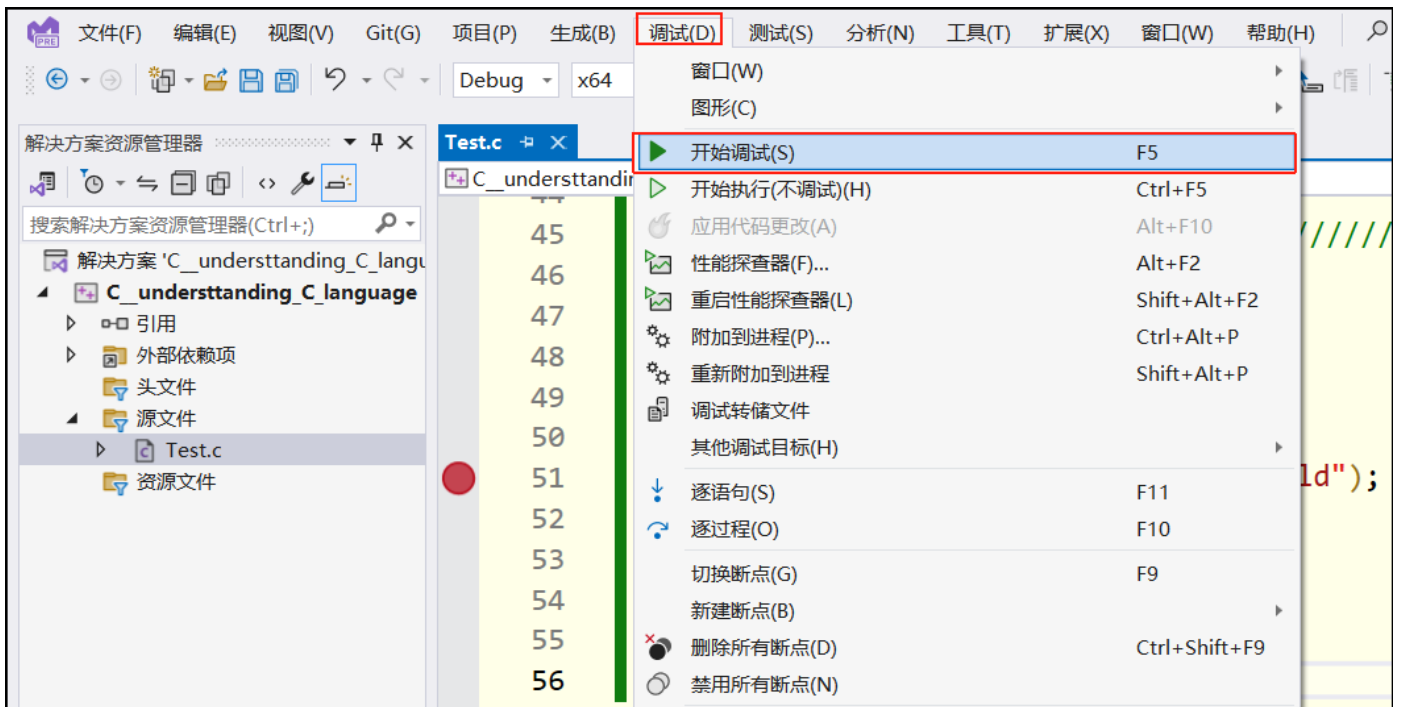
使用监视窗口观察常量字符串

- 调试方法：此处先跟着会用，后续有专门文章讲
  - 打断点：在想要让程序停下来的那一行最左侧点击，出现红色圆圈代表打断点成功（快捷键：F9）

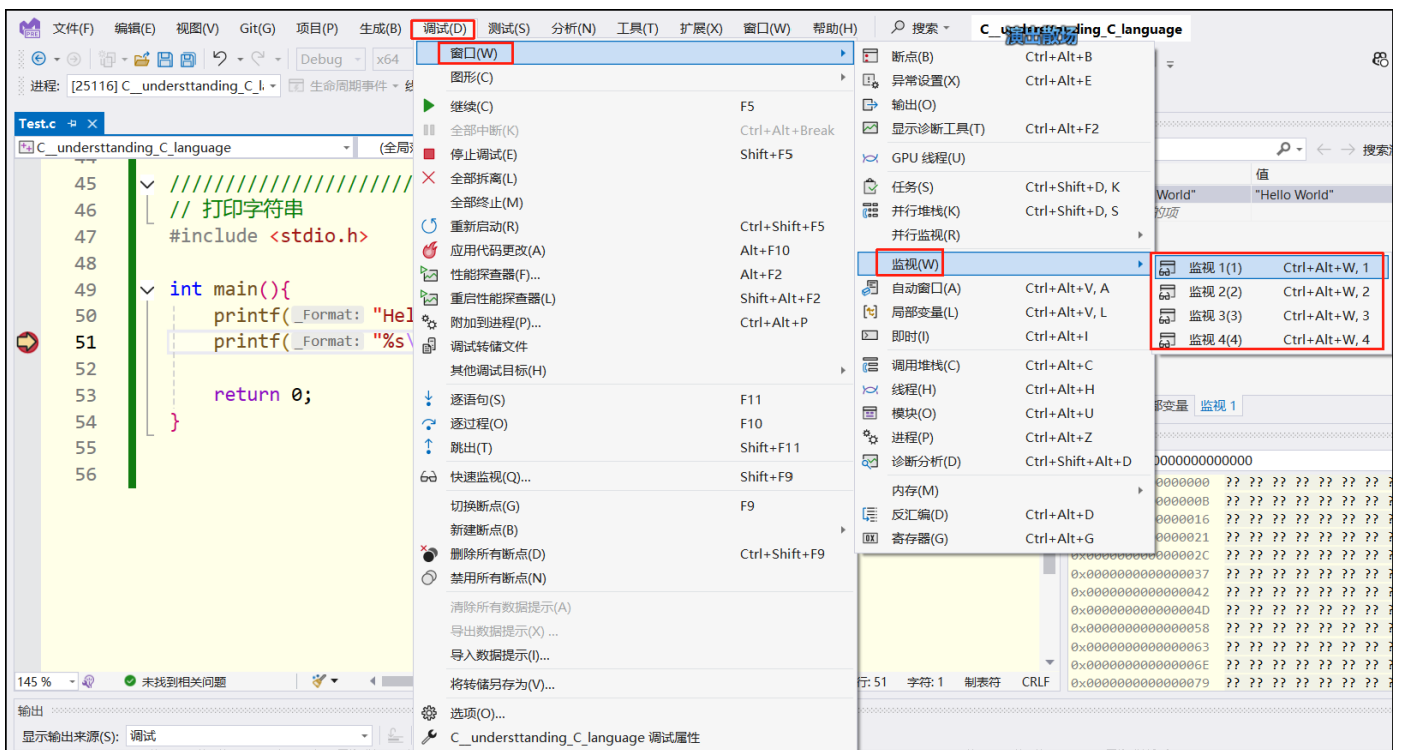


```
47      #include <stdio.h>
48
49      int main(){
50          printf(_Format: "Hello World\n");
51          printf(_Format: "%s\n", "Hello World");
52
53          return 0;
54      }
```













- 进入调试：“调试”，“开始调试”（快捷键：F5）



- 调出监视窗口：“调试”，“窗口”，“监视”，“监视（1/2/3/4均可）”（快捷键：Ctrl+Alt+W, 1/2/3/4任意一个 --> 表示先Ctrl+Alt+W, 再Ctrl+1/2/3/4）



- 双击监视窗口第一行输入要观察的字符串然后点击最左边的小三角形即可一个个字符观察

 "Hello World"	"Hello World"	查看	char[12]
 [0]	72 'H'		char
 [1]	101 'e'		char
 [2]	108 'l'		char
 [3]	108 'l'		char
 [4]	111 'o'		char
 [5]	32 ' '		char
 [6]	87 'W'		char
 [7]	111 'o'		char
 [8]	114 'r'		char
 [9]	108 'l'		char
 [10]	100 'd'		char

- printf打印遇到\0时就会停止：代码验证

```

1  #include <stdio.h>
2
3  int main(){
4      printf("Hello World\n");
5      printf("Hello \0World\n");
6
7      return 0;
8  }
```

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

```

int main(){
    printf(_Format: "Hello World\n");
    printf(_Format: "Hello \0World\n");

    return 0;
}
```



- 利用字符数组（先不用管是什么）验证\0在字符串中功能

```

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



```
10 }
```

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

```
int main(){
    char arr1[] = { 'a','b','c' };
    char arr2[] = "abc";

    return 0;
}
```

名称	值	类型
arr1	0x0000007415aff614 "abc..."	char[3]
[0]	97 'a'	char
[1]	98 'b'	char
[2]	99 'c'	char
arr2	0x0000007415aff634 "abc"	char[4]
[0]	97 'a'	char
[1]	98 'b'	char
[2]	99 'c'	char
[3]	0 '\0'	char

添加要监视的项

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

```
int main(){
    char arr1[] = { 'a','b','c' };
    char arr2[] = "abc";
    printf(_Format: "%s\n", arr1);
    printf(_Format: "%s\n", arr2);

    return 0;
}
```

Microsoft Visual Studio 调试控制台

```
abc烫烫烫烫烫烫烫烫烫烫烫烫烫烫bc
abc
```

- 发现arr1打印出来后面跟了很多随机字符“烫……”，就是因为arr1末尾没有\0作为结束标志打印时没有停止

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

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

```
int main() {  
    char arr1[] = { 'a', 'b', 'c', '\0' };  
    char arr2[] = "abc";  
    printf(_Format: "%s\n", arr1);  
    printf(_Format: "%s\n", arr2);  
  
    return 0;  
}
```



```
Microsoft Visual Studio 调试控制台 × +  
abc  
abc  
  
D:\Provisional Code Projects\  
, 代码为 0 (0x0)。  
按任意键关闭此窗口 . . . |
```

- 当手动添加上\0后发现可以正常打印了（与上面打印Hello World和Hello \0World是同一个道理）

## 11. 转义字符

- 前面讲到了\n是换行符，\0表示一个字符串的结束，这些特殊的字符被称为转义字符：转变了原本意思的字符
  - 代码举例：n与\n，0与\0

```
1  #include <stdio.h>  
2  
3  int main() {  
4      printf("abcnnnnnefg\n");  
5      printf("\n\n");  
6      printf("abc\nnnn\nefg\n");  
7      printf("\n\n");  
8  
9      printf("abc00efg\n");  
10     printf("abc0\0efg\n");  
11     printf("\n\n");  
12  
13     return 0;  
14 }
```

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

```
int main() {  
    printf(_Format: "abcnnnnnefg\n");  
    printf(_Format: "\n\n");  
    printf(_Format: "abc\nnnn\nefg\n");  
    printf(_Format: "\n\n");  
  
    printf(_Format: "abc00efg\n");  
    printf(_Format: "abc0\0efg\n");  
    printf(_Format: "\n\n");  
  
    return 0;  
}
```



Microsoft Visual Studio

abcnnnnnefg

abc  
nnn  
efg

abc00efg  
abc0

D:\Provisional Code  
, 代码为 0 (0x0)。

- 可以发现没有加\的时候可以正常打印出n (0) 字符，加了\后就变成了换行（结束打印）的功能
- 常见转义字符：
  - \?: 书写连续多个问号时使用，防止被解析为三字母词（现在新的编译器上无法验证）
  - \': 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，可以使终端发出警报声或出现闪烁或同时发生
  - \b: 退格键，光标回退一个字符，但不删除字符
  - \v: 垂直分割符，光标移动到下一个垂直制表位，通常为下一行的同一列
  - \f: 换页符，光标移动到下一页（现代系统反应不出来了，行为改为类似\v）
  - \n: 换行符
  - \r: 回车符，光标移动到同一行的开头
  - \t: 制表符：光标移动到下一个水平制表位，通常为下一个4/8的倍数
  - \ddd: ddd表示1~3个八进制的数字
    - 如：\130: 字符X
  - \xdd: dd表示2个十六进制数字

- 如：\x30：字符0
- \0：NULL字符，代表没有内容，\0属于\ddd这类转义字符的一种，用于字符串的结束标志，ASCII码值为0
- 代码验证：

```

1  #include <stdio.h>
2
3  int main() {
4      printf("%c\n", '\\');
5      printf("%s\n", "\\");
6      printf("D:\\Test\\Test.c\n");
7      printf("\a");
8      printf("%c\n", '\\130'); // 130是八进制 --> 十进制: 10 --> ASCII: X
9      printf("%c\n", '\\x30'); // x30是十六进制 --> 十进制: 48 --> ASCII: 0
10
11     return 0;
12 }

```

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

```

int main() {
    printf(_Format: "%c\n", '\\');
    printf(_Format: "%s\n", "\\");
    printf(_Format: "D:\\Test\\Test.c\n");
    printf(_Format: "\a");
    printf(_Format: "%c\n", '\\130');
    printf(_Format: "%c\n", '\\x30');

    return 0;
}

```

Microsoft Visual Studio 调试控制台

```

'
"
D:\Test\Test.c
X
0

D:\Provisional Code P
, 代码为 0 (0x0)。
按任意键关闭此窗口。

```

同时电脑会响一声

- 转义字符：<https://zh.cppreference.com/w/c/language/escape>
- 关于\?与三字母词
  - 三字母词：三个字母合成另一个字符，共有9个
    - ??= #
    - ??( [

- ??) ]
- ??< {
- ??> }
- ??/ /
- ??! |
- ??' ^
- ??- ~

- 历史：在以前的老式键盘中，类似 “[ ] { } ^ | ”等符号是没有的，为了解决这个问题，C语言中出现了“三字母词”，即在代码编译阶段用三个字符代替这些没有的符号
- 现在不再存在这种问题，了解即可，同时C23中取消了三字母词
- 关于回车、换行、回车换行
  - 在老式打字机中，回车和换行是不同的行为
    - 回车：回到当前行的起始位置 --- 对应转义字符\r
    - 换行：换到下一行 --- 对应转义字符\n
  - 但现代操作系统结合了回车和换行 --- 对应转义字符\n

## 12. 语句和语句分类

- C语言代码由一条条语句构成：

### 12.1 空语句

- 空语句就是由一个分号构成，不进行任何操作的语句。但实际开发中没有太多实际的价值

```
1  int main() {
2      ; // 空语句
3      return 0;
4  }
```

### 12.2 表达式语句

- 在表达式后加上分号即表达式语句

```
1  int main() {
2      int a = 0; // 表达式语句
}
```

```
3         return 0;
4     }
```

## 12.3 函数调用语句

- 调用函数的语句就是函数调用语句，后面也需要加上分号

```
1     int add(int x, int y) {
2         return x + y;
3     }
4     int main() {
5         add(2, 3); // 函数调用语句
6         return 0;
7     }
```

## 12.4 复合语句

- 复合语句即**代码块**，由花括号括起来的内容就是代码块

```
1     int add(int x, int y) {
2         return x + y;
3     } // 复合语句
4     int main() {
5         for (int i = 0; i < 10; ++i) {
6             printf("%d ", i);
7         } // 复合语句
8         printf("\n");
9         return 0;
10    }
```

## 12.5 控制语句

- 控制语句用于控制程序执行的流程，用来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构成，C语言支持9种控制语句
- 分类：
  - 条件判断语句/分支语句：if、switch
  - 循环执行语句：while、for、do-while
  - 转向语句：break、continue、goto、return

## 13. 注释

- 注释：注释是对代码功能、逻辑、关键点的说明，注释在编译时会被替换成为一个空格，对实际所写的代码没有任何影响。注意不要不写注释、也不要写无意义的注释
- 注释有两种方法
  - `/*注释内容*/`
  - 内部可分行，可以插入在行内部

```
1  int add(int x, int y) {
2      return x + y;
3  }
4  int main() {
5      printf("%d\n", add(2, 3)); /* 实现调用加法函数并在控制台打印出计算结果
   */
6      return 0;
7  }
```

- `// 注释内容`
- 从 `//` 开始到该行结束均为注释内容

```
1  int add(int x, int y) {
2      return x + y;
3  }
4  int main() {
5      printf("%d\n", add(2, 3)); // 实现调用加法函数并在控制台打印出计算结果
6      return 0;
7  }
```

- 注意：注释符号不能放在双引号内部，会被认为是转义字符或字符串内容

© 本文章内部部分资源来源于网络，侵权联系删除

本文章仅支持个人学习使用，不允许商用