C语言是一门面向过程的计算机编程语言，与C++、Java等面向对象编程语言有所不同。C语言的设计目标是提供一种能以简易的方式[编译](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91/1258343" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)、处理低级[存储器](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E5%99%A8/1583185" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)、仅产生少量的[机器码](https://baike.baidu.com/item/%E6%9C%BA%E5%99%A8%E7%A0%81/86125" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)以及不需要任何运行环境支持便能运行的编程语言。C语言描述问题比汇编语言迅速，工作量小、可读性好，易于[调试](https://baike.baidu.com/item/%E8%B0%83%E8%AF%95" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)、修改和移植，而[代码质量](https://baike.baidu.com/item/%E4%BB%A3%E7%A0%81%E8%B4%A8%E9%87%8F/8863758" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)与[汇编语言](https://baike.baidu.com/item/%E6%B1%87%E7%BC%96%E8%AF%AD%E8%A8%80/61826" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)相当。C语言一般只比[汇编语言](https://baike.baidu.com/item/%E6%B1%87%E7%BC%96%E8%AF%AD%E8%A8%80/61826" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)代码生成的[目标程序](https://baike.baidu.com/item/%E7%9B%AE%E6%A0%87%E7%A8%8B%E5%BA%8F/8082041" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)效率低10%～20%。因此，C语言可以编写系统软件。 [1]

二十世纪八十年代，[美国国家标准局](https://baike.baidu.com/item/%E7%BE%8E%E5%9B%BD%E5%9B%BD%E5%AE%B6%E6%A0%87%E5%87%86%E5%B1%80/3350884" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)为了避免各开发厂商用的C语言语法产生[差异](https://baike.baidu.com/item/%E5%B7%AE%E5%BC%82/2230102" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)，给C语言制定了一套完整的美国国家标准语法，称为[ANSI C](https://baike.baidu.com/item/ANSI C" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)。作为C语言最初的标准。 [2]  2011年12月8日，[国际标准化组织](https://baike.baidu.com/item/%E5%9B%BD%E9%99%85%E6%A0%87%E5%87%86%E5%8C%96%E7%BB%84%E7%BB%87/779832" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)（ISO）和[国际电工委员会](https://baike.baidu.com/item/%E5%9B%BD%E9%99%85%E7%94%B5%E5%B7%A5%E5%A7%94%E5%91%98%E4%BC%9A/2876390" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)（IEC）发布的**C11标准**是C语言的第三个官方标准，也是C语言的最新[标准](https://baike.baidu.com/item/%E6%A0%87%E5%87%86/219665" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)，该标准更好的支持了汉字函数名和汉字标识符，一定程度上实现了汉字编程。

C语言编译器普遍存在于各种不同的操作系统中，例如Microsoft Windows, Mac OS X, Linux, Unix等。C语言的设计影响了众多后来的编程语言，例如C++、Objective-C、Java、C#等。

C语言之所以命名为C，是因为 C语言源自[Ken Thompson](https://baike.baidu.com/item/Ken Thompson" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)发明的[B语言](https://baike.baidu.com/item/B%E8%AF%AD%E8%A8%80" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)，而 B语言则源自BCPL语言。

1967年，[剑桥大学](https://baike.baidu.com/item/%E5%89%91%E6%A1%A5%E5%A4%A7%E5%AD%A6" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)的Martin Richards对CPL语言进行了简化，于是产生了BCPL（Basic Combined Programming Language）语言。

20世纪60年代，美国[AT&T](https://baike.baidu.com/item/AT&T" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)公司[贝尔实验室](https://baike.baidu.com/item/%E8%B4%9D%E5%B0%94%E5%AE%9E%E9%AA%8C%E5%AE%A4" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)（AT&T Bell Laboratory）的研究员[Ken Thompson](https://baike.baidu.com/item/Ken Thompson" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)闲来无事，手痒难耐，想玩一个他自己编的，模拟在太阳系航行的电子游戏——Space Travel。他背着老板，找到了台空闲的机器——[PDP-7](https://baike.baidu.com/item/PDP-7" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)。但这台机器没有操作系统，而游戏必须使用操作系统的一些功能，于是他着手为PDP-7开发操作系统。后来，这个操作系统被命名为——[UNIX](https://baike.baidu.com/item/UNIX" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)。

1970年，美国贝尔实验室的 Ken Thompson，以[BCPL](https://baike.baidu.com/item/BCPL" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)语言为基础，设计出很简单且很接近硬件的B语言（取BCPL的首字母）。并且他用B语言写了第一个UNIX操作系统。

1971年，同样[酷爱](https://baike.baidu.com/item/%E9%85%B7%E7%88%B1/1371" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)Space Travel的[Dennis M.Ritchie](https://baike.baidu.com/item/Dennis M.Ritchie" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)为了能早点儿玩上游戏，加入了[Thompson](https://baike.baidu.com/item/Thompson" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)的开发项目，合作开发UNIX。他的主要工作是改造B语言，使其更[成熟](https://baike.baidu.com/item/%E6%88%90%E7%86%9F/80699" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)。 [2]

1972年，美国[贝尔实验室](https://baike.baidu.com/item/%E8%B4%9D%E5%B0%94%E5%AE%9E%E9%AA%8C%E5%AE%A4" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)的 D.M.Ritchie 在B语言的基础上最终设计出了一种新的[语言](https://baike.baidu.com/item/%E8%AF%AD%E8%A8%80/2291095" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)，他取了BCPL的第二个字母作为这种语言的名字，这就是C语言。

1973年初，C语言的主体完成。Thompson和Ritchie迫不及待地开始用它完全重写了[UNIX](https://baike.baidu.com/item/UNIX" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)。此时，编程的乐趣使他们已经完全忘记了那个"Space Travel"，一门心思地投入到了UNIX和C语言的开发中。随着UNIX的发展，C语言自身也在不断地完善。直到今天，各种版本的UNIX内核和周边工具仍然使用C语言作为最主要的开发语言，其中还有不少继承Thompson和Ritchie之手的代码。 [2]

在开发中，他们还考虑把[UNIX](https://baike.baidu.com/item/UNIX" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)移植到其他类型的计算机上使用。C语言强大的移植性（Portability）在此显现。机器语言和[汇编语言](https://baike.baidu.com/item/%E6%B1%87%E7%BC%96%E8%AF%AD%E8%A8%80/61826" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)都不具有移植性，为[x86](https://baike.baidu.com/item/x86" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)开发的程序，不可能在Alpha,[SPARC](https://baike.baidu.com/item/SPARC" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)和[ARM](https://baike.baidu.com/item/ARM/7518299" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)等机器上运行。而C语言程序则可以使用在任意架构的[处理器](https://baike.baidu.com/item/%E5%A4%84%E7%90%86%E5%99%A8" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)上，只要那种架构的处理器具有对应的C语言[编译器](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%99%A8" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)和库，然后将C源代码[编译](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)、[连接](https://baike.baidu.com/item/%E8%BF%9E%E6%8E%A5/8248019" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)成目标[二进制文件](https://baike.baidu.com/item/%E4%BA%8C%E8%BF%9B%E5%88%B6%E6%96%87%E4%BB%B6" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)之后即可运行。 [2]

1977年，[Dennis M.Ritchie](https://baike.baidu.com/item/Dennis M.Ritchie" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)发表了不依赖于具体机器系统的C语言编译文本《可移植的C语言编译程序》。 [3]

C语言继续发展，在1982年，很多有识之士和[美国国家标准协会](https://baike.baidu.com/item/%E7%BE%8E%E5%9B%BD%E5%9B%BD%E5%AE%B6%E6%A0%87%E5%87%86%E5%8D%8F%E4%BC%9A" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)为了使这个语言健康地发展下去，决定成立C标准委员会，建立C语言的标准。委员会由硬件厂商，编译器及其他软件工具生产商，软件设计师，顾问，学术界人士，C语言作者和应用程序员组成。1989年，[ANSI](https://baike.baidu.com/item/ANSI/14955" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)发布了第一个完整的C语言标准——ANSI X3.159—1989，简称“C89”，不过人们也习惯称其为“[ANSI C](https://baike.baidu.com/item/ANSI C" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)”。C89在1990年被[国际标准组织](https://baike.baidu.com/item/%E5%9B%BD%E9%99%85%E6%A0%87%E5%87%86%E7%BB%84%E7%BB%87" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)ISO(International Standard Organization)一字不改地采纳，ISO官方给予的名称为：ISO/IEC 9899，所以ISO/IEC9899: 1990也通常被简称为“C90”。1999年，在做了一些必要的修正和完善后，ISO发布了新的C语言标准，命名为ISO/IEC 9899：1999，简称“[C99](https://baike.baidu.com/item/C99" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)”。 [2]  在2011年12月8日，ISO又正式发布了新的[标准](https://baike.baidu.com/item/%E6%A0%87%E5%87%86/219665" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)，称为ISO/IEC9899: 2011，简称为“[C11](https://baike.baidu.com/item/C11" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)”。

首先贴出书上的例程：  
#include <stdio.h>  
int main (void)  
{  
printf("Hello world!\n");  
return 0;  
}  
一个程序，简单的可以理解为你让计算机去做一件事情。上面这个就是你让计算机在屏幕上输出hello world。

##### 第一句有什么卵用？

首先从字面上解释，include是包含的意思，那它要包含啥呢？后面有个stdio.h。至于# <>这两个约定俗成的符号，是为了让计算机识别的。stdio，其实是**st**an**d**ard **i**nput **o**utput的缩写，即**基本输入输出**。而stdio.h可以形象地理解为是一个**工具箱**，里面包含了各种造好的工具，当然，里面没有真正的锤子或是榔头。计算机要用到这个工具箱来进行基本输入输出操作，而后面用到的printf就相当于这里面的一把螺丝刀。

##### int main(void)又是什么鬼

首先来说一下函数，计算机里面的函数类似数学上的函数，但又不太一样。数学上，自变量x0经过f()的作用后得到了f(x0)。那么在上面的程序里，main相当于f，f(x0)是一个值（可以是整数、分数...），那么main()也是一个值，int是**int**eger(整数)的缩写，表明main作用出来的是一个**整数**。像f(x)一样，这个main函数也需要有自变量"x"的输入啊！void翻译过来是“无效的，空的”，放在main的自变量位置，说明main的自变量输入是**空**，即“我并不需要自变量”。

##### { 我是函数主体 }

像f一样，main也是有特定功能的，于是约定**要被执行的功能**被放在了一对**花括号**中。  
printf("Hello world!\n");  
printf()类似main()，也是一个函数。它从哪来的？？对，就是我们之前包含的stdio.h工具箱。printf()的自变量就是"Hello world!\n"。日常生活中双引号中间的是一句话。这边也是类似，不过它有个更高逼格的名字：字符串。  
\n，固定的表达，**n**ext line，表示换行。为什么要有 \ ? 不然n就识别不出来了！  
return 0;字面意思，返回0。前面说过，main()是一个整数，最后总会要有一个值的！return+值 就是表示给这个main()一个值。（0一般表示程序运行成功，所以通常main返回0）

##### 上课的void main()是什么鬼？

与int main(void)类比一下，int表示整数，也就是最后要返回一个整数值给main，而换成了void，表示“我不要返回值”。所以细心的人肯定发现了老师并没有return 0;这句。main()括号里面是空的，默认没有参数。不用管了。

#include：标准I/O头文件，下文中的printf需要使用这个文件

int main()：主函数，一个程序的入口。一个程序有且只有一个main()函数

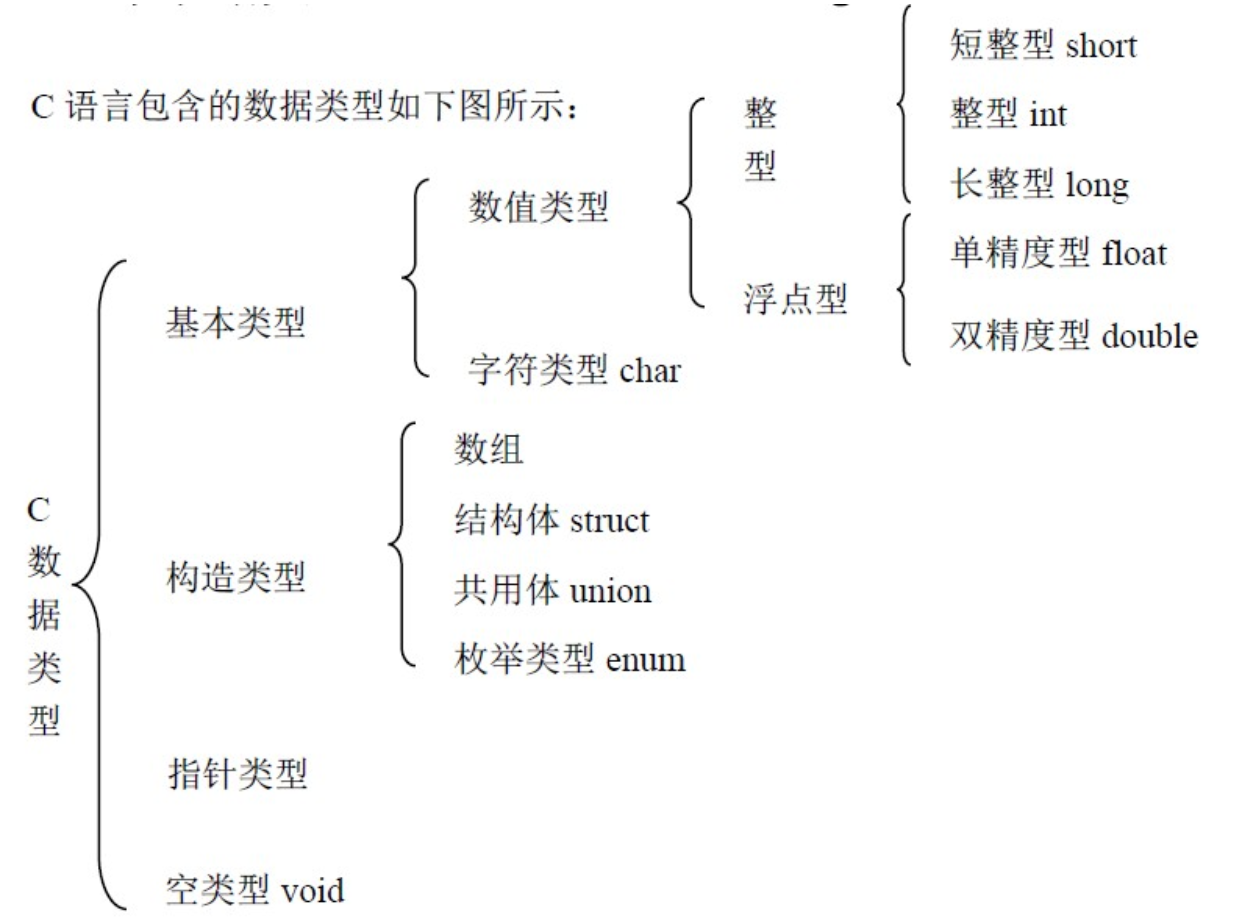
int：代表主函数的返回值类型是int型

printf()：格式化输出，将括号内的内容打印到显示器上，打印字符串需要用双引号""引上

return 0;：配合主函数类型使用，一般来说返回0的程序表示程序运行成功

打印的结果为：Hello World！

**C语言允许使用的数据类型有三类**。



(1)    **基本类型**。不可再分的最基本的数据类型，包括整型、浮点（单精度）型、双精度型、字符型、无值类型、逻辑型及复数型。基本类型通常代表单个数据。

(2)    **构造类型**。由已知的基本类型通过一定的构造方法构造出来的类型，包括数组、结构体、联合体、枚举类型等。构造类型通常代表一批数据。

(3)    **指针类型**。指针可以指向内存地址，访问效率高，用于构造各种形态的动态或递归数据结构，如链表、树等。

**1.1    基本数据类型**

**1．基本数据类型及其关键字**

C99标准提供的7种基本数据类型及其对应的关键字如表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| **数据类型** | **关键字** | **数据类型** | **关键字** |
| **字符型** | char | **无值类型** | void |
| **整型** | int | **逻辑型** | \_bool |
| **浮点(单精度)型** | float | **复数型** | \_complex\_imaginary |
| **双精度型** | double |  |  |

**字符型**：描述单个字符；

**整型**：描述整数，整数在计算机上是准确表示的；

**浮点型、双精度型**：描述实数，实数在计算机上一般是近似表示的，浮点型的近似程度比较低，而双精度型的近似程度比较高。

**无值类型**：没有具体的值，通常用来描述无形式参数或无返回值的C函数，以及无定向指针。

**逻辑型：**描述逻辑真（其值为1）与逻辑假（其值为0）。

**复数型：**描述复数（\_complex）和纯虚数（\_imaginary）。

使用逻辑型时必须包含头文件stdbool.h，使用复数型时必须包含头文件complex.h。

**c语言运算符**

算术运算符

下表显示了 C 语言支持的所有算术运算符。假设变量 **A** 的值为 10，变量 **B** 的值为 20，则：

|  |  |  |
| --- | --- | --- |
| **运算符** | **描述** | **实例** |
| + | 把两个操作数相加 | A + B 将得到 30 |
| - | 从第一个操作数中减去第二个操作数 | A - B 将得到 -10 |
| \* | 把两个操作数相乘 | A \* B 将得到 200 |
| / | 分子除以分母 | B / A 将得到 2 |
| % | 取模运算符，整除后的余数 | B % A 将得到 0 |
| ++ | 自增运算符，整数值增加 1 | A++ 将得到 11 |
| -- | 自减运算符，整数值减少 1 | A-- 将得到 9 |

## 关系运算符

下表显示了 C 语言支持的所有关系运算符。假设变量 **A** 的值为 10，变量 **B** 的值为 20，则：

|  |  |  |
| --- | --- | --- |
| **运算符** | **描述** | **实例** |
| == | 检查两个操作数的值是否相等，如果相等则条件为真。 | (A == B) 不为真。 |
| != | 检查两个操作数的值是否相等，如果不相等则条件为真。 | (A != B) 为真。 |
| > | 检查左操作数的值是否大于右操作数的值，如果是则条件为真。 | (A > B) 不为真。 |
| < | 检查左操作数的值是否小于右操作数的值，如果是则条件为真。 | (A < B) 为真。 |
| >= | 检查左操作数的值是否大于或等于右操作数的值，如果是则条件为真。 | (A >= B) 不为真。 |
| <= | 检查左操作数的值是否小于或等于右操作数的值，如果是则条件为真。 | (A <= B) 为真。 |

## 逻辑运算符

下表显示了 C 语言支持的所有关系逻辑运算符。假设变量 **A** 的值为 1，变量 **B** 的值为 0，则：

|  |  |  |
| --- | --- | --- |
| **运算符** | **描述** | **实例** |
| && | 称为逻辑与运算符。如果两个操作数都非零，则条件为真。 | (A && B) 为假。 |
| || | 称为逻辑或运算符。如果两个操作数中有任意一个非零，则条件为真。 | (A || B) 为真。 |
| ! | 称为逻辑非运算符。用来逆转操作数的逻辑状态。如果条件为真则逻辑非运算符将使其为假。 | !(A && B) 为真。 |

## 赋值运算符

下表列出了 C 语言支持的赋值运算符：

|  |  |  |
| --- | --- | --- |
| **运算符** | **描述** | **实例** |
| = | 简单的赋值运算符，把右边操作数的值赋给左边操作数 | C = A + B 将把 A + B 的值赋给 C |
| += | 加且赋值运算符，把右边操作数加上左边操作数的结果赋值给左边操作数 | C += A 相当于 C = C + A |
| -= | 减且赋值运算符，把左边操作数减去右边操作数的结果赋值给左边操作数 | C -= A 相当于 C = C - A |
| \*= | 乘且赋值运算符，把右边操作数乘以左边操作数的结果赋值给左边操作数 | C \*= A 相当于 C = C \* A |
| /= | 除且赋值运算符，把左边操作数除以右边操作数的结果赋值给左边操作数 | C /= A 相当于 C = C / A |
| %= | 求模且赋值运算符，求两个操作数的模赋值给左边操作数 | C %= A 相当于 C = C % A |

## ? : 运算符(三元运算符)

我们已经在前面的章节中讲解了 **条件运算符 ? :**，可以用来替代 **if...else** 语句。它的一般形式如下：

Exp1 ? Exp2 : Exp3;

### **实例**

以下实例通过输入一个数字来判断它是否为奇数或偶数

## 实例

#include<stdio.h>

int main() {

int num;

printf("输入一个数字 : ");

scanf("%d",&num);

(num%2==0)?printf("偶数"):printf("奇数");

}

## 判断语句

C 语言提供了以下类型的判断语句。点击链接查看每个语句的细节。

|  |  |
| --- | --- |
| **语句** | **描述** |
| [if 语句](https://www.runoob.com/cprogramming/c-if.html" \o "C 中的 if 语句" \t "https://www.runoob.com/cprogramming/_blank) | 一个 **if 语句** 由一个布尔表达式后跟一个或多个语句组成。 |
| [if...else 语句](https://www.runoob.com/cprogramming/c-if-else.html" \o "C 中的 if...else 语句" \t "https://www.runoob.com/cprogramming/_blank) | 一个 **if 语句** 后可跟一个可选的 **else 语句**，else 语句在布尔表达式为假时执行。 |
| [嵌套 if 语句](https://www.runoob.com/cprogramming/c-nested-if.html" \o "C 中的嵌套 if 语句" \t "https://www.runoob.com/cprogramming/_blank) | 您可以在一个 **if** 或 **else if** 语句内使用另一个 **if** 或 **else if** 语句。 |
| [switch 语句](https://www.runoob.com/cprogramming/c-switch.html" \o "C 中的 switch 语句" \t "https://www.runoob.com/cprogramming/_blank) | 一个 **switch** 语句允许测试一个变量等于多个值时的情况。 |
| [嵌套 switch 语句](https://www.runoob.com/cprogramming/c-nested-switch.html" \o "C 中的嵌套 switch 语句" \t "https://www.runoob.com/cprogramming/_blank) | 您可以在一个 **switch** 语句内使用另一个 **switch**语句。 |

#include <stdio.h>

#include <stdbool.h>

int main(void)

{

bool bTrue = true;

bool bFalse = false;

if (bFalse)

printf("if true!\n");

else if(bFalse)

printf("else if true!\n");

else

printf("else!\n");

return 0;

}

### **结果：**

else!

## 循环类型

C 语言提供了以下几种循环类型。点击链接查看每个类型的细节。

|  |  |
| --- | --- |
| **循环类型** | **描述** |
| [while 循环](https://www.runoob.com/cprogramming/c-while-loop.html" \o "C 中的 while 循环) | 当给定条件为真时，重复语句或语句组。它会在执行循环主体之前测试条件。 |
| [for 循环](https://www.runoob.com/cprogramming/c-for-loop.html" \o "C 中的 for 循环) | 多次执行一个语句序列，简化管理循环变量的代码。 |
| [do...while 循环](https://www.runoob.com/cprogramming/c-do-while-loop.html" \o "C 中的 do...while 循环) | 除了它是在循环主体结尾测试条件外，其他与 while 语句类似。 |
| [嵌套循环](https://www.runoob.com/cprogramming/c-nested-loops.html" \o "C 中的嵌套循环) | 您可以在 while、for 或 do..while 循环内使用一个或多个循环。 |

## 循环控制语句

循环控制语句改变你代码的执行顺序。通过它你可以实现代码的跳转。

C 提供了下列的循环控制语句。点击链接查看每个语句的细节。

|  |  |
| --- | --- |
| **控制语句** | **描述** |
| [break 语句](https://www.runoob.com/cprogramming/c-break-statement.html" \o "C 中的 break 语句) | 终止**循环**或 **switch** 语句，程序流将继续执行紧接着循环或 switch 的下一条语句。 |
| [continue 语句](https://www.runoob.com/cprogramming/c-continue-statement.html" \o "C 中的 continue 语句) | 告诉一个循环体立刻停止本次循环迭代，重新开始下次循环迭代。 |
| [goto 语句](https://www.runoob.com/cprogramming/c-goto-statement.html" \o "C 中的 goto 语句) | 将控制转移到被标记的语句。但是不建议在程序中使用 goto 语句。 |

1. #include <stdio.h>
2. int main(){
3. int i, sum=0;
4. i = 1; //语句①
5. **while**(i<=100 /\*语句②\*/ ){
6. sum+=i;
7. i++; //语句③
8. }
9. printf("%d\n",sum);
10. **return** 0;
11. }
12. #include <stdio.h>
13. int main(){
14. int i, sum=0;
15. **for**(i=1/\*语句①\*/; i<=100/\*语句②\*/; i++/\*语句③\*/){
16. sum+=i;
17. }
18. printf("%d\n",sum);
19. **return** 0;
20. }

数组遍历

#include<stdio.h>

#include<stdlib.h>

Int main(){

Int a[] = {1,9,10,78,0,5,10};

Int i;

For(i=0; i<7;i++)

{

Printf(“%d\n”,a[i]);

}

System(“pause”);

Return 0

}

continue 语句和 break 语句的区别是，continue 语句只结束本次循环，而不是终止整个循环。break 语句则是结束整个循环过程，不再判断执行循环的条件是否成立。而且，continue 只能在循环语句中使用，即只能在 for、while 和 do…while 中使用，除此之外 continue 不能在任何语句中使用。  
  
所以，再次强调：continue 不能在 switch 中使用，除非 switch 在循环体中。此时 continue 表示的也是结束循环体的本次循环，跟 switch 也没有关系。  
  
例如，下面为使用switch模拟电梯运行的代码：

1. # include <stdio.h>
2. int main(void)
3. {
4. int val; //variable的缩写, “变量”的意思
5. printf("请输入您想去的楼层:");
6. **while** (1)
7. {
8. scanf("%d", &val);
9. **switch** (val)
10. {
11. **case** 1:
12. printf("1层开!\n");
13. **break**; //跳出switch
14. **case** 2:
15. printf("2层开!\n");
16. **break**; //跳出switch
17. **case** 3:
18. printf("3层开!\n");
19. **break**; //跳出switch
20. **default**:
21. printf("该层不存在, 请重新输入:");
22. **continue**; //结束本次while循环
23. }
24. **break**; //跳出while
25. }
26. **return** 0;
27. }

输出结果是：  
请输入您想去的楼层:4  
该层不存在, 请重新输入:3  
3层开!  
  
此时在 default 中如果不使用 continue 的话，那么最后的“该层不存在，请重新输入：”仍然不起作用。它虽然能输出，但输出之后直接就执行后面的 break 了，用户仍然没有重新输入的机会。而加 continue 之后就会结束本次 while 循环，即后面的 break 语句本次就不会执行，而是直接进入 while 循环，这样用户就有“重新输入”的机会了。

函数

#include <stdio.h>

int sum(){

int i, sum=0;

for(i=1; i<=100; i++){

sum+=i;

}

return sum;

}

int main(){

int a = sum();

printf("The sum is %d\n", a);

return 0;

}

Python可以说是目前最火的语言之一了，人工智能的兴起让Python一夜之间变得家喻户晓，Python号称目前最最简单易学的语言，现在有不少高校开始将Python作为大一新生的入门语言。接触Python，发现Python与其他语言确实有很大的区别。Python是由C语言实现的，因此想把Python与C语言做一个简单的比较。

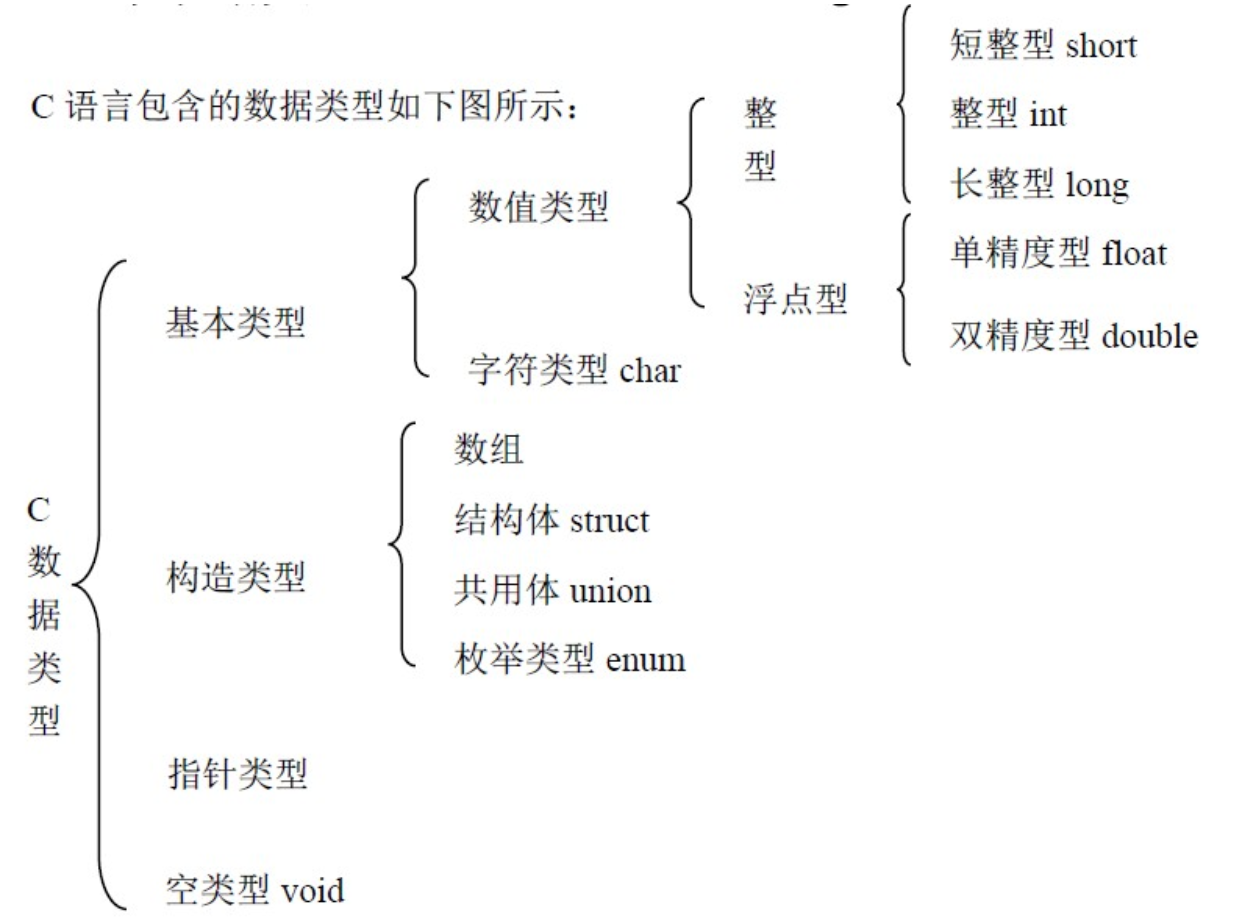
**一、**

C语言是编译型语言，经过编译后，生成机器码，然后再运行，执行速度快，不能跨平台，一般用于操作系统，驱动等底层开发。

Python是编译型还是解释型这个界限并不明显，但大致上可以理解为解释型语言，执行速度慢，由于Python虚拟机，Python是可以跨平台的，Python高度集成适合于软件的快速开发。

**二、**

C语言的数据类型如下图：



C语言中需要事先定义变量类型，以int类型为例，当定义一个int型变量后，就会在内存中开辟4个字节，再来进行初始化，由于长度是指定的，在运算过程中需要考虑，溢出，精度等问题。

Python中的数据类型：

1.Number：数字

  Int

  Float

  Bool

  Complex

2.String：字符串

3.List：列表

4.Tupel：元组

5.Sets：集合

6.Dictionary：字典

python不需要事先定义变量类型，以a=3为例，在内存中存放一个整数3，然后再用变量a指向3，变量a是没有类型的，我们所说的类型是指变量所指的内存中对象的类型。

从数据类型上就可以看出Python的友好性，基本数据类型变少了，没有烦人的指针，不需要考虑数据溢出和精度的问题，当在程序中需要使用某个变量时，就能够直接使用，而不需要在程序开头定义变量。除此之外，Python还提供了str,list,dict这些强大的数据类型，让程序开发变的更为简单。

**三、**

Python还提供了一个交互界面，输入python进入交互界面，输入exit()退出交互界面，类似于Linux终端，输入一行命令，执行一行，为学习Python提供了很大的便利。

**四、**

在运算符和优先级上面，两者并没有大的区别，但在python中没有自加和自减运算符，在逻辑运算符上Python也区别于C语言，Python中是and，or，not ，而C语言中则是&&，||，！

**六、**

Python中通过缩进来表示语句体，C语言通过{}来表示语句体，并且在Python中每一条语句结尾后没有分号，判断语句if else，这两者没有区别，循环语句while也没有，只是for循环，Python通过for  in来表示。

**七、**

Python有很多内置函数（build in function）,不需要写头文件，Python还有很多强大的模块，需要时导入便可。C语言在这一点上远不及Python,大多时候都需要自己手动实现。

**八、**

C语言中的函数，有着严格的顺序限制，如果要调用函数，该函数需要在本次调用之前就需要被实现，或者在程序开头事先声明，而Python中则没有这个限制，Python中还有高阶函数这一概念，即函数名也可当作函数参数，函数名也是一种变量，指向内存中的某个函数，这种写法可以大大减少代码长度。

python中还提供了可变参数和关键字参数，这样使得函数的功能大大提高，原来需要写多个函数，现在只需要一个函数就可以实现这些功能。

**九、**

C语言是面向过程的语言，很多时候都需要自己手动实现函数来完成某一功能。Python中引入了类和对象，是面向对象编程的语言，面向对象使得代码的可重用性大大提高，数据的封装性也更好。面向对象与面向过程的具体比较就不多说了，但有两句非常重要的话：**类是抽象的，而对象是具体的。**

**十、**

python中既有函数也有方法，常常让人疑惑，我个人觉得也没有必要区分的非常清楚，但是两者还是有较大的区别：**函数是自由的，而方法是受限的**。在编程的时候需要分清楚调用的是方法还是函数。

**总结**：Python可以说是非常`简单`的语言，高度集成，代码量少，简单是相对其他语言而言。但编程从来都不是一个简单活，需要我们不断学习，掌握底层实现原理，才是正道。