

# 手把手教你学

# C语言

吴明杰 曹宇 吴丹 著



机械工业出版社  
China Machine Press

手把手教你学

# C语言

吴明杰 曹宇 吴丹 著



机械工业出版社  
China Machine Press

## 图书在版编目 (CIP) 数据

手把手教你学 C 语言 / 吴明杰, 曹宇, 吴丹著. —北京: 机械工业出版社, 2016.11

ISBN 978-7-111-55307-6

I. 手… II. ①吴… ②曹… ③吴… III. C 语言—程序设计 IV. TP312.8

中国版本图书馆 CIP 数据核字 (2016) 第 262349 号

## 手把手教你学 C 语言

---

出版发行: 机械工业出版社 (北京市西城区百万庄大街 22 号 邮政编码: 100037)

责任编辑: 余 洁

责任校对: 殷 虹

印 刷: 中国电影出版社印刷厂

版 次: 2016 年 11 月第 1 版第 1 次印刷

开 本: 186mm×240mm 1/16

印 张: 36.75

书 号: ISBN 978-7-111-55307-6

定 价: 79.00 元

凡购本书, 如有缺页、倒页、脱页, 由本社发行部调换

客服热线: (010) 88379426 88361066

投稿热线: (010) 88379604

购书热线: (010) 68326294 88379649 68995259

读者信箱: hzit@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问: 北京大成律师事务所 韩光 / 邹晓东

当你拿起本书翻到这一页时，不管最后买与不买，都要对你说声谢谢，相遇就是缘分。

## 为什么要写这本书

本书是我人生中写的第一本书，基于我学习 C 语言的笔记。我从大学本科的时候开始学习 C 语言，每每学到新的知识或有心得体会时便记下来，就同写日记一样。就这样断断续续一直持续到硕士研究生阶段，那时差不多写了 7 万多字。我会将自己的笔记分享给很多想学 C 语言的师弟师妹。在学习的过程中他们发现，我的笔记比其他 C 语言书籍都更易于理解，讲得通俗易懂，风趣幽默。虽然当时只有 7 万多字，内容有限，但他们都认为“绝对是入门的好书”。所以我的“前期读者”以及专业导师都希望我能出版这些笔记。但我觉得还不够好，内容还不够充实，也怕误人子弟，浪费读者时间，毕竟当时水平有限。但这却在我心里埋下了一颗想要写一本好书的种子。

硕士研究生毕业后我应聘到上海起策教育科技有限公司工作，而我教授的第一门课就是 C 语言。从此我正式开始了与 C 语言全天候、长时期的亲密接触，也正式开启了我要将这本书写好的历程。在工作中，我有很多心得体会，或跟同事交流，或得益于很多前辈的教导。于是通过不断地补充，原先的 7 万多字变为现在的 34 万多字。在这个过程中我不断地对它精雕细琢，就像培养一个优秀的孩子一样，只希望能展现给大家一份优秀的作品。现在我觉得时机到了，可以让它为更多想学好 C 语言的读者做贡献了。

## 本书内容

本书是学习 C 语言的入门书籍，所以一开始对 C 语言的铺垫很充分，循序渐进，目的是让大家对 C 语言不再陌生，轻松地学习。本书的内容对于入门来说是非常全面的，

包括 C 语言基础知识、流程控制、数组、函数、指针、字符串、结构体、链表、文件操作等主流知识。这几个知识点是学习 C 语言的主要框架，对于不同的 C 语言书籍，区别就在于讲得是否详细，是否能让每位读者都掌握。本书中这几个知识点都讲得非常详细、透彻，是经过无数学弟学妹检验过的，也期待着读者的检验。除此之外，本书还增加了很多在工作中需要用到的其他知识，如栈和队列、自定义头文件、多文件编译、Linux 下 C 文件的编译和链接、链接库等。

## 本书特色

因为本书基于我的学习笔记，所以本书更多的是以初学者的角度编写的，而且后来一直延续了这种风格，抛开“高大上”、生涩的专业术语，用词通俗易懂。

因为本书的很多内容是我在参加工作之后总结的，所以都是根据实际工作的需要整理而成。摒弃了脱离实际工作、过时的、不用的、“变态”的用法，大大减轻了读者学习的压力，除去了学习道路上的“杂草”，铺设了一条更好走的捷径。

此外，本书并不是单纯地讲理论，而是配有大量的程序。每个知识点都是配合程序讲解的，这样理解起来就更加容易。而且本书没有那种单独的、无答案的课后练习题，所有的练习都直接以程序的形式写在书中，读者在学习的时候直接练习那些程序即可，而且每个程序都是经过编译可以直接运行的。此外本书不会提供电子版的代码，因为学习 C 语言必须要多动手、多“敲”代码，所以我希望读者自己动手。

最后真切地希望本书能成为你编程路上的重要伙伴，为你的成长打下深厚的编程功底。“虽然我可能不是最好的，但我绝对是最用心的。”

限于作者水平有限，书中难免存在不当或疏漏之处，恳请读者批评指正，并多提出宝贵意见。希望在你的帮助下本书一步步接近完美，谢谢！

吴明杰

2016 年 9 月

前言	2.4 本章总结	11
<b>第 1 章 为什么要学习 C 语言</b>	<b>第 3 章 Microsoft Visual C++ 6.0 的使用</b>	12
1.1 C 的起源和发展	3.1 为什么要学习 VC++ 6.0	12
1.1.1 计算机语言发展的三个阶段	3.2 如何创建编程文件	12
1.1.2 语言运行速度的比较	3.3 编写一个最简单的程序	16
1.1.3 C 语言的演变过程	3.4 要养成时刻保存的习惯	18
1.2 C 的特点	3.5 编译-链接-执行	18
1.2.1 C 语言的优点	3.6 怎样运行第二个程序	20
1.2.2 C 语言的缺点	3.7 编译-链接-执行时保存路径下 的文件夹有什么变化	21
1.3 C 的应用领域	3.8 如何编写多文件程序	23
1.4 C 的重要性	3.9 如何用 VC++ 6.0 调试程序	28
1.5 本章总结	3.10 本章总结	30
<b>第 2 章 怎样学习 C 语言</b>	<b>第 4 章 从一个程序走进 C 语言</b>	31
2.1 学习 C 语言的心得	<b>第 5 章 预备知识</b>	35
2.2 学习 C 语言的目标	5.1 CPU、内存、硬盘、显卡、 主板、显示器之间的关系	35
2.3 常见问题答疑	5.1.1 电影是如何运行的	35
2.3.1 学习 Java 之前为什么建议 先学 C 语言	5.1.2 CPU 为什么不能直接操作 硬盘却能直接操作内存	35
2.3.2 没学过计算机专业课程 能够学懂 C 语言吗		
2.3.3 英语和数学不好能学好 C 语言吗		

5.1.3	内存的速度为什么比硬盘 的速度快 .....	36	5.8.1	原码和反码 .....	52
5.1.4	为什么不将内存造得跟 硬盘一样大 .....	36	5.8.2	补码的两个核心问题 .....	53
5.1.5	CPU 是如何操作内存的 .....	36	5.8.3	int 型变量所能存储的范围 .....	55
5.1.6	主板的作用 .....	37	5.8.4	int 型和 char 型变量是 如何相互赋值的 .....	56
5.2	HelloWorld 程序是如何运行 起来的 .....	37	5.9	什么是 ASCII .....	57
5.3	字节 .....	38	5.10	变量 .....	58
5.3.1	什么是字节 .....	38	5.10.1	如何定义变量 .....	58
5.3.2	字节换算 .....	38	5.10.2	变量的本质 .....	61
5.3.3	小结 .....	38	5.10.3	为什么要使用变量 .....	62
5.4	进制 .....	39	5.10.4	变量的命名规则 .....	62
5.4.1	什么是进制 .....	39	5.10.5	为什么必须要初始化 变量 .....	63
5.4.2	进制转换口算法 .....	40	5.10.6	小结 .....	66
5.4.3	进制转换公式法 .....	41	5.11	各类型数据之间的混合运算 .....	66
5.4.4	人类为什么最习惯用 十进制 .....	44	5.12	代码规范化 .....	68
5.4.5	计算机为什么用的是 二进制 .....	44	5.12.1	代码如何写才能规范 .....	68
5.4.6	小结 .....	44	5.12.2	代码规范化的好处 .....	68
5.5	数据类型 .....	45	5.12.3	代码规范化的七大原则 .....	69
5.5.1	数据类型的分类 .....	45	5.12.4	小结 .....	72
5.5.2	基本数据类型及其所占的 字节数 .....	46	5.13	本章总结 .....	72
5.6	常量 .....	48	<b>第 6 章 printf 的用法 .....</b>		<b>73</b>
5.6.1	整型常量 .....	49	6.1	printf 的格式 .....	74
5.6.2	浮点型常量 .....	49	6.2	输出控制符 .....	76
5.6.3	字符型常量 .....	51	6.3	%x、%X、%#x、%#X 的区别 .....	76
5.7	常量是以什么样的二进制代码 存储在计算机中的 .....	51	6.4	如何输出“%d”、“\”和双引号 .....	77
5.8	补码 .....	52	6.5	本章总结 .....	78
			<b>第 7 章 scanf 的用法 .....</b>		<b>79</b>
			7.1	概述 .....	79
			7.2	使用 scanf 的注意事项 .....	82

7.2.1	参数的个数一定要对应	82	<b>第 10 章 循环控制</b>	117	
7.2.2	输入的数据类型一定要与 所需要的数据类型一致	83	10.1	循环执行的定义和分类	117
7.2.3	在使用 scanf 之前使用 printf 提示输入	85	10.2	for 循环	117
7.3	本章总结	86	10.2.1	for 语句的格式	117
<b>第 8 章 运算符和表达式</b>		87	10.2.2	自增和自减	121
8.1	算术运算符	87	10.2.3	for 循环编程练习	122
8.2	关系运算符	89	10.2.4	for 和 if 的嵌套使用	122
8.3	逻辑运算符	90	10.2.5	强制类型转换	125
8.4	赋值运算符	93	10.2.6	浮点数的存储所带来的 问题	130
8.5	运算符的优先级	94	10.2.7	for 循环的嵌套使用	134
8.6	本章总结	95	10.2.8	for 循环嵌套编程练习	136
<b>第 9 章 选择结构程序设计</b>		96	10.3	while 循环	146
9.1	流程控制	96	10.3.1	while 的执行顺序	146
9.1.1	什么是流程控制	96	10.3.2	while 和 for 的比较	147
9.1.2	流程控制的分类	96	10.3.3	如何看懂一个程序—— “试数”	148
9.2	选择执行的定义和分类	97	10.3.4	do··while	151
9.3	if 语句	97	10.3.5	break 和 continue	156
9.3.1	if 的最简单用法	97	10.4	清空输入缓冲区	158
9.3.2	if 的控制范围问题	99	10.4.1	输入缓冲区	158
9.3.3	if··else 的用法	100	10.4.2	%d 和 %c 读取缓冲区 的差别	159
9.3.4	if··else if··else 的用法	103	10.4.3	用 scanf 吸收回车	161
9.3.5	练习——求分数的等级	104	10.4.4	getchar()	163
9.3.6	练习——三个整数从小到 大排序	107	10.4.5	fflush(stdin)	166
9.3.7	if 的常见问题解析	110	10.5	本章总结	168
9.4	switch 语句	113	<b>第 11 章 数组</b>		169
9.5	本章总结	115	11.1	一维数组的使用	169



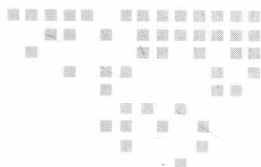
11.1.1	一维数组的定义	169	第 12 章 函数	215
11.1.2	一维数组的初始化	170	12.1 概述	215
11.1.3	一维数组元素的引用	173	12.1.1 什么是函数	215
11.1.4	如何将数组 <b>a</b> 赋给 数组 <b>b</b>	174	12.1.2 C 程序的组成和编译 单位	216
11.1.5	如何编程获取数组的 长度	175	12.1.3 库函数和自定义函数	216
11.1.6	练习	177	12.1.4 函数的调用	216
11.1.7	宏定义: <code>#define</code>	179	12.2 为什么需要函数	217
11.1.8	小结	182	12.3 有参函数	217
11.2	数组倒置算法	182	12.3.1 有参函数定义的一般 形式	218
11.3	数组查找算法	184	12.3.2 形参和实参	219
11.3.1	顺序查找	184	12.3.3 主调函数中对被调函数 的声明	220
11.3.2	折半查找	185	12.3.4 定义函数时应指定返回 值类型	221
11.4	数组插入、删除算法	188	12.3.5 函数的返回值	222
11.4.1	插入算法	188	12.3.6 <code>return</code> 是如何将值返回 给主调函数的	222
11.4.2	删除算法	189	12.3.7 函数的命名规则	223
11.5	数组排序算法	190	12.3.8 练习——判断一个自然 数是否是素数	223
11.5.1	冒泡排序	190	12.4 函数的递归调用	227
11.5.2	插入排序	193	12.4.1 什么是递归	227
11.5.3	选择排序	196	12.4.2 使用递归必须要满足 的两个条件	228
11.5.4	快速排序	198	12.4.3 如何学习递归	229
11.5.5	四种排序算法的 比较	204	12.4.4 递归和循环的关系	229
11.6	二维数组的使用	205	12.4.5 递归的优缺点	229
11.6.1	二维数组的定义	205	12.4.6 练习——用递归求 $n$ 的阶乘	230
11.6.2	二维数组的初始化	206		
11.6.3	二维数组如何输出	207		
11.6.4	练习——杨辉三角	208		
11.6.5	是否存在多维数组	213		
11.7	本章总结	213		

12.4.7	练习——用递归实现 1+2+3+...+100 的和	232			
12.4.8	练习——用递归求斐波 那契数列	233			
12.4.9	练习——用递归求两个 数的最大公约数	234			
12.4.10	小结	235			
12.5	数组名作为函数参数	236			
12.6	变量的作用域和存储方式	238			
12.6.1	局部变量	239			
12.6.2	全局变量	240			
12.6.3	为什么不建议使用全局 变量	242			
12.6.4	自动变量 (auto)	242			
12.6.5	静态变量 (static)	242			
12.6.6	寄存器变量 (register)	245			
12.6.7	外部变量 (extern)	246			
12.7	本章总结	248			
<b>第 13 章 指针</b>			249		
13.1	指针的重要性	249			
13.2	地址和指针的概念	250			
13.3	指针和指针变量	251			
13.3.1	指针变量的定义	252			
13.3.2	指针变量的初始化	254			
13.3.3	指针常见错误	257			
13.4	指针作为函数参数	260			
13.4.1	互换两个数	260			
13.4.2	函数参数传指针和传 数据的区别	262			
13.4.3	定义只读变量: const	263			
13.4.4	用 const 修饰指针变量 时的三种效果	264			
13.5	指针和一维数组的关系	266			
13.5.1	用指针引用数组元素	266			
13.5.2	指针的移动	267			
13.5.3	指针变量的自增运算	271			
13.5.4	两个参数确定一个数组	277			
13.5.5	指针变量占多少字节	279			
13.6	函数、数组、指针相结合的 程序练习	280			
13.7	动态内存分配	282			
13.7.1	传统数组的缺点	282			
13.7.2	malloc 函数的使用 (一)	283			
13.7.3	malloc 函数的使用 (二)	286			
13.7.4	free 函数的使用	289			
13.7.5	练习——动态数组的 构建	291			
13.7.6	动态数组长度的扩充 和缩小	294			
13.7.7	静态内存分配和动态 内存分配小结	296			
13.7.8	多级指针	296			
13.7.9	跨函数使用动态内存	298			
13.8	通过指针引用二维数组	302			
13.8.1	二维数组元素的地址	302			
13.8.2	二维数组的首地址和 数组名	303			
13.8.3	练习	307			
13.9	函数指针	309			
13.9.1	什么是函数指针	309			
13.9.2	如何用函数指针调用 函数	310			
13.10	本章总结	311			

<b>第 14 章 字符串</b> .....	313	14.6.11 sprintf().....	345
14.1 字符串常量.....	313	14.6.12 字符串比较函数 strcmp().....	347
14.2 不能将一个字符串常量赋给 一个字符变量.....	314	14.7 本章总结.....	348
14.3 字符数组.....	315	<b>第 15 章 自定义数据类型——       结构体</b> .....	350
14.3.1 字符数组的定义.....	315	15.1 为什么需要结构体.....	350
14.3.2 字符数组的初始化.....	315	15.2 定义和使用结构体变量.....	351
14.3.3 初始化内存函数： memset().....	318	15.2.1 声明结构体类型.....	351
14.3.4 用 scanf 输入字符串.....	320	15.2.2 定义结构体类型变量.....	352
14.4 字符串与指针.....	322	15.2.3 结构体变量可进行 哪些运算.....	354
14.5 如何用 scanf 给字符指针变量 所指向的内存单元初始化.....	323	15.2.4 结构体变量的初始化 ——定义时初始化.....	354
14.6 字符串处理函数.....	324	15.2.5 结构体变量的引用.....	355
14.6.1 字符串输入函数 gets().....	324	15.2.6 结构体变量的初始化 ——先定义后初始化.....	358
14.6.2 优先使用 fgets() 函数.....	326	15.2.7 结构体字节对齐.....	363
14.6.3 使用 gets() 和 fgets() 前注意吸收回车.....	329	15.3 结构体数组.....	369
14.6.4 练习——判断字符串 是否是回文字符串.....	331	15.3.1 结构体数组的定义和 引用.....	369
14.6.5 练习——将字符串中的 小写字母转换成大写 字母.....	336	15.3.2 结构体数组定义时 初始化.....	371
14.6.6 字符串输出函数 puts().....	337	15.4 结构体指针.....	372
14.6.7 字符串输出函数 fputs().....	339	15.4.1 指向结构体变量的指针.....	372
14.6.8 字符串复制函数 strcpy().....	341	15.4.2 指向结构体数组的指针.....	375
14.6.9 字符串复制函数 strncpy().....	343	15.5 结构体变量和结构体指针 变量作为函数参数.....	378
14.6.10 内存拷贝函数 memcpy().....	343	15.6 练习——动态构造存放学生 信息的结构体数组.....	382
		15.7 本章总结.....	385

第 16 章 链表	387	17.7 本章总结	454
16.1 为什么要学习链表	387	第 18 章 队列	455
16.1.1 数组的缺陷	388	18.1 队列的定义	455
16.1.2 链表的优点	388	18.2 顺序队列	455
16.2 链表的缺点	389	18.3 链式队列的基本操作	456
16.3 链表相关术语	389	18.4 链式队列程序演示	457
16.4 链表的定义和分类	390	18.5 队列的应用	464
16.4.1 链表的定义	390	18.6 本章总结	464
16.4.2 链表的分类	390	第 19 章 文件操作	465
16.5 编写一个链表程序	391	19.1 文件概述	466
16.6 练习——插入结点	396	19.1.1 文件的定义	466
16.7 练习——删除结点	404	19.1.2 文件的分类	466
16.8 练习——销毁链表	409	19.2 文件类型指针变量	467
16.9 链表排序算法	413	19.3 文件的打开	469
16.9.1 如何互换结点	413	19.3.1 打开文件	469
16.9.2 链表冒泡排序	414	19.3.2 文件的使用方式	470
16.9.3 链表插入排序	418	19.3.3 练习——打开文件	472
16.9.4 链表选择排序	421	19.4 文件的关闭	474
16.9.5 链表快速排序	422	19.5 文件读写函数概述	475
16.9.6 链表排序小结	425	19.6 fgetc() 和 fputc()	475
16.10 单循环链表	425	19.6.1 fputc()	475
16.11 双向链表	430	19.6.2 fgetc()	477
16.12 typedef 的用法	434	19.6.3 fgetc() 程序举例	478
16.13 本章总结	437	19.6.4 fgetc() 和 fputc() 结合 程序举例	482
第 17 章 栈	439	19.6.5 如何判断文件是否 为空	483
17.1 内存的分区	439	19.7 fgets() 和 fputs()	484
17.2 线性表	440	19.7.1 fgets()	484
17.3 栈的定义	441	19.7.2 fputs()	486
17.4 栈的基本操作	441		
17.5 链栈程序演示	442		
17.6 栈的应用	450		

19.8 移动文件位置指针: fseek() 和 rewind().....	488	20.3 “#include <>” 和 “#include ""” 的区别.....	518
19.8.1 fseek().....	488	20.4 如何自定义头文件.....	519
19.8.2 rewind().....	489	20.4.1 头文件的组成部分.....	519
19.9 格式化读写函数: fprintf() 和 fscanf().....	489	20.4.2 #ifndef/#define/#endif.....	520
19.10 数据块读写函数: fread() 和 fwrite().....	491	20.5 本章总结.....	522
19.10.1 fwrite() 函数.....	492	<b>第 21 章 位操作运算符</b> .....	523
19.10.2 fread() 函数.....	494	21.1 按位与 (&).....	523
19.10.3 ftell().....	500	21.2 按位或 ( ).....	524
19.10.4 如何用 fread() 和 fwrite() 删除文件中的数据块.....	503	21.3 按位异或 (^).....	524
19.10.5 练习——登录程序.....	509	21.4 左移运算符 (<<).....	526
19.11 本章总结.....	513	21.5 右移运算符 (>>).....	527
<b>第 20 章 头文件</b> .....	515	21.6 本章总结.....	528
20.1 程序是如何编译生成可执行文件的.....	515	<b>附录 A gcc 编译工具的使用</b> .....	530
20.2 概述.....	516	<b>附录 B make 自动化编译工具的使用</b> .....	537
20.2.1 什么是头文件.....	516	<b>附录 C gdb 调试工具的使用</b> .....	549
20.2.2 头文件的作用.....	517	<b>附录 D 链接库</b> .....	563
20.2.3 什么样的内容适合放在头文件中.....	517	<b>附录 E 运算符的优先级和结合性</b> .....	572
		<b>常用 ASCII 码表</b> .....	574



# 为什么要学习 C 语言

在学习 C 语言时，很多同学都有这样的疑问：为什么要学习 C 语言呢？这个问题很好，也很重要！因为学习任何一门课程都有难度，如果不知道为什么要学习的话，那么很可能学到半路就放弃了。所以学习 C 语言之前必须要先将这个问题弄清楚，这样学起来才有目的性，才有动力。

C 语言是一门编程语言，编程就是跟计算机进行对话。要与计算机进行对话就要学习一门能与计算机进行沟通的语言，C 语言就是这样一门语言。但是能与计算机进行沟通的语言很多，为什么要学习 C 语言呢？本章从四个方面来解释这个问题。

## 1.1 C 的起源和发展

### 1.1.1 计算机语言发展的三个阶段

如图 1-1 所示，计算机语言的发展主要分为三个阶段。

#### 1. 机器语言

第一代计算机语言称为机器语言。机器语言就是 0/1 代码。计算机只能识别 0 和 1。在计算机内部，无论是一部电影还是一首歌曲或是一张图片，最终保存的都是 0/1 代码，因为 CPU 只能执行 0/1 代码。那么这是不是就意味着我们编程一定要用 0/1 代码呢？首先这么编写肯定是可以的，但是这样太麻烦，而且很不好理解，所以后来就出现了汇编语言。

## 2. 汇编语言

汇编语言就是将一串很枯燥无味的机器语言转化成一个英文单词。比如说：

```
add 1, 2;
```

add 就是一个英文单词，这样看起来就稍微有一些含义了，即 1 和 2 相加。这个就是汇编语言。

如果直接用机器语言编写的话，这几乎是无法实现的。因为用机器语言太难记忆了，也没人能看得懂。所以后来就设计出了第二种语言，即将 0/1 代码翻译为英文单词，这些英文单词直接对应着一串 0/1 指令。这个就是汇编语言。通过专门的软件就可以将这些英文单词转化成 0/1 代码并由计算机执行，这种专门起翻译的作用的软件叫作编译器。这些英文单词和与它们对应的 0/1 代码之间的对应关系，以及语言的语法，在编写这个软件的时候就已经写在里面了。我们只要通过编译器就可以将这些都转化成 0/1 代码。这样大大方便了我们程序的编写。

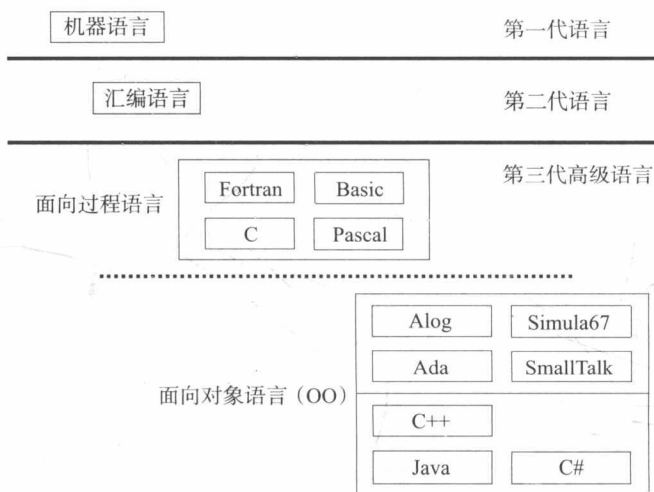


图 1-1 计算机语言发展的三个阶段

## 3. 高级语言

汇编语言之后又出现了第三代语言。第三代语言又叫“高级语言”。高级语言的发展分为两个阶段，以 1980 年为分界线，前一阶段属于结构化语言或者称为面向过程的语言，后一阶段属于面向对象的语言。

什么叫面向过程，什么叫面向对象？这是很难解释的一个问题，所以这个问题大家现在先不要考虑。等到将来你们学完 C 语言、C++、Java 或者 C# 之后才有可能理解。因为这个需要比较。

总之，面向过程语言中最经典、最重要的就是 C 语言。Fortran、Basic 和 Pascal 语言基本上已经很少有人使用了。但是 C 语言一直在用，因为 C 语言是计算机领域最重要的一门语言。但是 C 语言也有缺陷，它的缺陷只有在学完面向对象语言之后才能体会到。

所以从 20 世纪 80 年代开始又产生了另外一种“以面向对象”为思想的语言，其中最重要、最复杂的就是 C++。C++ 从易用性和安全性两个方面对 C 语言进行了升级。C++ 是一种较复杂、难学的语言，但是一旦学会了则非常有用。因为 C++ 太复杂，所以后来就对 C++ 进行了改装，产生了两种语言，一个是 Java，另一个是 C#。

Java 语言是现在最流行的语言之一。C# 则是微软公司看 Java 很流行而写的一个与 Java 语法相似的语言。因为 Java 和 C# 几乎是一模一样的，所以你只需要学习其中的一种语言就可以了。

### 1.1.2 语言运行速度的比较

计算机语言越是低级速度就越快，因为越低级就越符合计算机的思维。所以计算机语言中执行速度最快的是机器语言，汇编语言其次，高级语言的速度最慢。高级语言中 C 的速度最快，C++ 其次，最慢的是 Java 和 C#。Java 和 C# 虽然速度慢，但它们在任何机器上都可以运行，而且运行结果一模一样，这是它们的一个优点，也是它们流行的原因之一。

### 1.1.3 C 语言的演变过程

与 C 语言相关的语言很多。其中最早的一门语言叫 Algol 60，是 1960 年产生的，它是真正的第一门面向问题的语言。但是这门语言离硬件比较远，所以 1963 年剑桥大学在 Algol 60 的基础上研发出了 CPL。CPL 同 Algol 60 相比更接近硬件一些，但规模比较大，难以实现。1967 年剑桥大学的马丁·理查兹（Martin Richards）对 CPL 进行了简化，产生了 BCPL。BCPL 中的 B 就是 Basic 的缩写，即“简化的”。

1970 年，美国 AT&T 公司贝尔实验室（AT&T Bell Laboratory）的研究员肯·汤普森（Ken Thompson）以 BCPL 为基础，设计出了很简单而且很接近硬件的 B 语言（取 BCPL 的首字母）。B 语言是贝尔实验室开发的一种通用程序设计语言。虽然它没有流行起来，但是它很重要。肯·汤普森用 B 语言做了一件很重要的事情，一直影响至今，即他用 B 语言写出了世界上第一个操作系统——UNIX 操作系统。

1971 年，贝尔实验室的丹尼斯·里奇（Dennis Ritchie）加入了肯·汤普森的开发项目，合作开发 UNIX。他的主要工作是改造 B 语言，使其更加成熟。

1972 年，丹尼斯·里奇在 B 语言的基础上最终设计出了一种新的语言，他以 BCPL 的第二个字母作为这种语言的名字，即 C 语言。

1973 年年初，C 语言的主体完成。肯·汤普森和丹尼斯·里奇开始用 C 语言完全重写



UNIX，这就是 UNIX 第 5 版。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 语言作为其最主要的开发语言，其中还有不少继承肯·汤普森和丹尼斯·里奇之手的代码。

UNIX 系统是世界上第一个真正的操作系统。由于 UNIX 操作系统是用 C 语言编写的，而这个系统很流行，于是 C 语言也跟着流行起来。而 UNIX 操作系统是开源的，所以别人要想学习，就要先学 C 语言。

B 语言被 C 语言改写后，C 语言流行了而 B 语言就被淘汰了。而且后来发现，C 语言的确非常好，它是面向过程语言的代表，是有史以来最重要的一门计算机语言。

随后又出现了 C++。C++ 是本贾尼·斯特劳斯特卢普 (Bjarne Stroustrup) 编写的，他也来自贝尔实验室，是 C 语言创始人丹尼斯·里奇的下属。C++ 就是在 C 语言的基础上发明的。C++ 进一步扩充和完善了 C 语言，是一种面向对象的程序设计语言。

后来 Sun 公司又对 C++ 进行改写，产生了 Java。而微软公司发现 Java 很流行，就造出了一个类似的语言——C#。所以 Java 和 C# 都源自于 C++。

以上就是 C 语言演变的过程。从这个过程我们可以看出，如果以后要学习 C++、Java 或者 C# 的话，那么 C 语言就必须学！因为它们都源自于 C 语言。而且 C 语言中绝大部分的知识，在 C++、Java、C# 中几乎都会用到。C 语言里面有两个知识点是必须要学的，一个是函数，另一个是指针。这两个知识点是整个 C 语言的主体和核心。而且这两个知识点在其他语言中是学不到的，或者是同 C 语言中有差别。总之，C 语言是它们的“老祖宗”，学习其他语言之前最好要将 C 语言学好。

## 1.2 C 的特点

C 语言现在已经很成熟，它的各种语法规则、思想都已经确立起来了，并对现在的很多语言产生很大的影响。但是任何事物都有其优点和缺点，C 语言也不例外。下面我们分别来看一下。

### 1.2.1 C 语言的优点

C 语言的优点有三个：

- 1) 代码量小。
- 2) 运行速度快。
- 3) 功能强大。

我们先看第一个优点，C 语言的代码量很小，这是什么意思呢？也就是说如果你要完成同样一个功能，用 C 语言编写出来的程序的容量是很小的，而用其他语言编写容量就会