C语言概述

One man's constant is another man's variable。[1]

:arrow_forward: 此符号表示该内容以后的章节会讲解，此章节内不要求理解。

本节内容

C语言的历史，C语言的优缺点以及如何高效的使用C语言

C语言还值得学习吗？C语言查错的工具

C语言的历史

起源

C语言是贝尔实验室的 Ken Thompson, Dennis Ritchie 等人开发的 UNIX 操作系统的“副产品”。

与同时代的其他操作系统一样，UNIX 系统最初也是用汇编语言写的。用汇编语言编写的程序往往难以调试和改进，UNIX 操作系统也不例外。Thompson 意识到需要用一种高级的编程语言来完成 UNIX 系统未来的开发，于是他设计了一种小型的 B语言。Thompson 的 B语言是在 BCPL语言（20世纪60年代中期产生的一种系统编程语言）的基础上开发的，而 BCPL语言又可以追溯到最早（且影响深远）的语言之一——Algol 60语言。

1970年，贝尔实验室为 UNIX 项目争取到了一台 PDP-11 计算机。当 B语言经过改进并能够在 PDP-11 计算机上成功运行后，Thompson 用 B语言重新编写了部分 UNIX 代码。

到了1971年，B语言已经明显不适合 PDP-11 计算机了，于是 Ritchie 着手开发 B语言的升级版。最初他将新开发的语言命名为 NB 语言（意味New B），但是后来新语言越来越偏离 B语言，于是他将其改名为 C语言。

到1973年，C语言已经足够稳定，可以用来重新编写 UNIX 系统了。

标准化

C语言在20世纪七十年代（尤其是1977年到1979）持续发展。这一时期出现了第一本有关 C语言的书。Brian Kernighan 和 Dennis Ritchie 合作编写的 *The C Programming Language* 于1978年出版，并迅速成为 C程序员必读的“圣经”。由于当时没有 C语言的正式标准，这本书就成为了事实上的标准，编程爱好者把它称为“K&R”或者“白皮书”。（公众号后台回复：【KR】即可获得）

随着C语言的迅速普及，一系列问题也接踵

而至。首先， K&R 对一些语言特性描述得非常模糊，以至于不同编译器对这些特性会做出不同的处理。而且，K&R 也没有对属于C语言的特性和属于 UNIX 系统的的特性进行明确的区分。更糟糕的是，K&R 出版后 C 语言仍然在不断变化，增加了一些新特性并除去了一些旧特性。很快，C语言需要一个全面，准确的最新描述开始成为共识。

C89/C90

1983年，在美国国家标准协会（ANSI）的推动下（ANSI 于此年组建了一个委员会称为 X3JII），美国开始制定本国的 C语言标准。

1988年完成并于1989年12月正式通过的 C 语言标准成为 ANSI 标准 X3.159-1989。

1990年，国际标准化组织（ISO）通过了此项标准，将其作为 ISO/IEC 9899:1990 国际标准（中国国家标准为 GB/T 15272—

1994）。

我们把这一C语言版本称为 **C89** 或 **C90**，以区别原始的 C语言版本。

委员会在制定的指导原则中的一条写道：保持 C 的精神。委员会在描述这一精神时列出了一下几点：

- 信任程序员
- 不要妨碍程序员做需要做的事
- 保持语言精炼简单
- 只提供一种方法执行一项操作
- 让程序运行更快，即使不能保持其可移植性

在最后一点上，标准委员会的用意是：作为实现，应该针对目标计算机来定义最合适的某特定操作，而不是强加一个抽象，统一的定义。在学习 C语言的过程中，许多方面都反映了这一哲学思想。

C99

1995 年，C语言发生了一些改变。

1999年通过的 ISO/IEC 9899：1999 新标准中包含了一些更重要的改变，这一标准所描述的语言通常称为 **C99**

此次改变，委员会的用意不是在C语言中添加新的特性，而是为了达到新的目标。

1. **支持国际化编程**。如：提供多种方法处理国际字符集
2. **调整现有实践致力于解决明显的缺陷**。因此，在遇到需要将 C移至64位处理器时，委员会根据现实生活中处理问题的经验来添加标准。
3. 为**适应科学和工程项目中的关键计算**，提高 C 的适应性，让 C 比 FORTRAN 更有竞争力。

其他方面的改变则更为保守，如，尽量让

C90，C++兼容，让语言在概念上保持简单。

虽然改标准已经发布了很长时间，但**并非所有编译器都完全支持C99**的所有改动。因此，你有可能发现 C99 的一些改动在自己的系统中不可用，或者需要改变编译器的设置才可用。

C11

2011年，**C11**标准问世。

基于C的语言

- C++：包含所有C的特性
- Java：基于C++，所以也继承了C的许多特性
- C#：由C++于java发展起来的较新的语言
- Perl：最初是一种简单的脚本语言，在发展过程中采用了C的许多特性

C语言还值得学吗?

答案是肯定的。

第一，学习C有助于更好的理解C++，Java，C#，Perl以及其他基于C的特性的语言。第一开始就学习其他语言的程序员往往不能很好的掌握继承自C语言的基本特性。

第二，目前仍有许多C程序，我们需要读懂并维护这些代码。

第三，C语言仍广泛应用于新软件的开发，特别是在内存或处理能力受限的情况下以及需要使用C语言简单特性的地方。

C语言的优缺点

与其他任何一种编程语言一样，C语言也有自己的优缺点。这些优缺点都源于该语言的最初用途（编写操作系统和其它系统软件）

和它自身的基础理论体系。

- **C语言是一种底层语言** 为了适应系统编程的需要，C语言提供了对机器级概念（例如，字节和地址）的访问，而这些都是其他编程语言试图隐藏的内容。
- **C语言是一种小型语言** 与许多其他编程语言相比，C语言提供了一套更有限特性集合。（在K&R第二版的参考手册中仅用49页就描述了整个C语言。）为了使特性较少，C语言在很大程度上依赖一个标准函数的“库”。
- **C是一种包容性语言** C假设用户知道自己在干什么，因此它提供了比其他许多语言更广阔的自由度。此外，C语言不像其他语言那样强制进行详细的错误检查。

C语言的优点

C语言的众多优点解释了C语言为何如此流

行。

- **高效** 高效性是C语言与生俱来的优点之一。发明C语言就是为了编写那些以往由汇编语言编写的程序，所以对C语言来说，能够在有限的内存空间快速运行就显得至关重要。

- **可移植** 当程序必须在多种机型（从个人计算机到超级计算机）上运行时，常常会用C语言来编写。

  原因一：C语言没有分裂成不兼容的多种分支。这主要归功于C语言早期与UNIX系统的结合以及后来的ANSI/ISO标准。

  原因二：C语言编译器规模小且容易编写，这使得它们得以广泛应用。

  原因三：C语言的自身特性也支持可移植性（尽管它没有阻止程序员编写不可

移植的程序）。

- **功能强大** C语言拥有一个庞大的数据类型和运算符集合，这个集合使得C语言具有强大的表达能力，往往寥寥几行代码就可以实现许多功能。

- **灵活** C语言最初设计是为了系统编程，但没有固有的约束将其限制在此范围内。C语言现在可以用于编写从嵌入式系统到商业数据处理的各种应用程序。

- **标准库** C语言的突出优点就是它具有标准库，该标准库包括了数百个可以用于输入/输出，字符串处理，储存分配以及其他实用操作的函数。

- **与UNIX的集成** C语言在与UNIX系统（包括Linux）结合方面特别强大。事实上，一些UNIX工具甚至假设用户是了解C语言的。

C语言的缺点

- **C语言容易隐藏错误** C语言的灵活性使得用它编程出错的概率极高。在用其他语言时可以发现的错误，C语言的编译器却无法检查到。更糟糕的是，C语言还包含大量不易察觉的隐患。
- **C程序可能难以理解** C程序的简明扼要与灵活性，可能导致程序员编写出除了自己别人无法读懂的代码。
- **C程序可能难以修改** 如果在设计中没有考虑到维护的问题，那么C编写的大型程序可能很难修改。现代的编程语言通常提供“类”和“包”之类的语言特性，这样的特性可以把大的程序分解成许多更容易管理的模块。遗憾的是，C语言恰恰缺少这样的特性。

高效的使用C语言

要高效的使用C语言，就需要利用C语言优

点的同时尽量避免它的缺点，一下给出一些建议。

- **学习如何规避C语言的缺陷**
- **使用软件工具使程序更可靠**（详细见下文）
- **利用现有的代码库** 使用C语言的一个好处是其他许多人也在使用C。把别人编写好的代码用于自己的程序是一个非常好多主意。C代码通常被打包成库（函数的集合）。获取适当的库既可以大大减少错误，也可以节省很多编程工作。
- **采用一套切合实际的编码规范** 良好的编码习惯和规范易于自己和他人对自己代码的阅读和修改。（公众号回复：【编码规范】，让你学会如何写出规范的代码。）
- **避免“投机取巧”和极度复杂的代码**。C语言鼓励使用编程技巧。但是，过犹不及，不要对技巧毫无节制，最简单的解

决方案往往也是最难理解的。

- **紧贴标准** 大多数编译器都提供不属于 C89/C99 标准的特征和库函数。为了程序的可移植性，若非确有必要，最好避免这些特性和库函数。

怎么让程序更加安全可靠?

- 分析错误工具——lint
- 越界检查工具——bounds-checker
- 内存泄漏监测工具——leak-finder
- 调高你的编译器的“警告级别”

以上就是本次的内容，感谢观看。

如果文章有错误欢迎指正和补充，感谢!

最后，如果你还有什么问题或者想知道到的，可以在评论区告诉我呦，我在后面的文章可以加上。

最后，**关注我**，看更多干货！

我是程序圆，我们下次再见。

参考资料：《C Primer Plus》《C语言程序设计：现代方法》

---

1. *吾之常量，彼之变量。摘自《epigrams-on-programming》* ↩