C++ 发展史

Dezeming Family

2023 年 2 月 23 日

DezemingFamily 系列文章和电子书**全部都有免费公开的电子版**，可以很方便地进行修改和重新发布。如果您获得了 DezemingFamily 的系列电子书，可以从我们的网站 [https://dezeming.top/] 找到最新的版本。对文章的内容建议和出现的错误也欢迎在网站留言。

本文大部分内容是来自维基百科和互联网博客，在整理时进行了丰富完善，并将涉及的内容的准确性进行了验证。

目录

一　从 C 语言开始

上个世纪 60 年代，全晶体管计算机已经开始盛行；到七十年代，已经开始出现集成电路计算机了。PDP-11 是 70 年代初开始出现的小型机，Dennis Ritchie 曾在 PDP-11 上开发 C 语言，在次之前，Kenneth Lane Thompson 曾经在 PDP-7 上开发了精简版 BCPL 语言，简称为 B 语言。Dennis Ritchie 和 Kenneth Lane Thompson 共同着手将 Unix 移植到 PDP-11 计算机上。

1975 年，Stephen C. Johnson 实现了一套容易移植的编译器，因此，C 语言开始被移植到其他各类机器中。

1978 年，Dennis Ritchie 和 Brian Wilson Kernighan 一起出版了《C 程序设计语言》，引入了一些特性，例如标准 I/O 库、结构体 struct 以及无符号整数等。这种标准被称为“K&R C”标准。不过后来第二版中（1988 出版），《C 程序设计语言》也加入了一些 ANSI C 的标准。

ANSI 是美国国家标准学会，它于 1918 年最初成立，并在 1969 年正式改名为 ANSI(American National Standards Institute)。1989 年，ANSI 将 C 语言标准化，扩充了一些 K&R C 的内容，增加了例如 void 类型等新特性。

国际标准化组织 ISO(International Organization for Standardization) 成立于 1947 年，用于指定各种标准。1990 年，ISO 成立 ISO/IEC JTC1/SC22/WG14 工作组，又修改了一些 ANSI C 标准，制定了新的标准，史称 C90，自此，ANSI 接受国际化标准，不再自己发展新的 C 标准。

之后，C 语言基本定型，但在 C++ 的标准化发展下，C 语言也带来了少量发展，并在 1999 年发布了新的 ISO 标准，史称 C99。

再之后，到 2011 年，ISO 又发布了新的 C 语言标准，新标准相当于为了增加对 C++ 的兼容性，从而增加了一些新特性，比如泛型等，但据我所知，新版本并没有使用的那么广泛，因为对于扩展功能，人们更倾向于使用 C++。

我们可以看到，C 语言和 C++ 的发展是有一些重叠的。C 语言更适用于小型程序或者操作系统内核，语言结构简洁明晰，所以相比于一直在更新迭代的 C++，C 语言的改动在后期一直都不是很大。

二　C++ 的发展史

网上 [3, 5] 一般会把 C++ 的发展分为三个主要阶段：

- 第一阶段：80 年代到 1995 年。这一阶段中，C++ 基本上属于传统类型上的面向对象语言，并且凭借着接近 C 语言的效率，在工业界使用的开发语言中占据了相当大份额。
- 第二阶段：从 1995 年到 2000 年，这一阶段由于标准模板库（STL）和后来的 Boost 等程序库的出现，泛型程序设计在 C++ 中占据了越来越多的比重。同时由于 Java、C# 等语言的出现和硬件价格的大规模下降，C++ 受到了一定的冲击。
- 第三阶段从 2000 年至今，由于以 Loki（《Modern C++ Design》一书的配套库）、MPL(Meta-Programming Library，后来成为了 Boost 的一员) 等程序库为代表的产生式编程和模板元编程的出现，C++ 出现了发展历史上又一个新的高峰，这些新技术的出现以及和原有技术的融合，使 C++ 已经成为当今主流程序设计语言中最复杂的一员。

1967 年，Simula 语言（顾名思义，Simula 语言的主要作用是仿真）中第一次出现了面向对象 (OO) 的概念（被称为 Simula 67，是 Simula 语言的一种变种），被公认是首款支持面向对象的语言。但由于当时软件规模不大且技术也不太成熟，Simula 语言执行效率也很低，面向对象的优势并未发挥出来。直到 1980 年，Smalltalk-80（Smalltalk 诞生于 70 年代，是一种纯面向对象的编程语言）的发展，使得面向对象技术开始逐步变得受欢迎。

1979 年，当时 Bjarne Stroustrup（C+++ 之父）正在准备他的博士毕业论文，他使用 Simula 语言时，发现面向对象的思想对于软件开发非常有用。不久之后，Stroustrup 开始着手“C with Classes”的研发工作，“C with Classes”表明这种新语言是在 C 基础上研发的，是 C 语言的超集。C 语言以其高可

移植性而广受好评，且程序执行速度以及底层函数的性能不受程序移植的影响，Stroustrup 要做的就是将面向对象的思想引入 C 语言。新语言的初始版本除了包括 C 语言的基本特征之外，还具备类、简单继承、内联机制、函数默认参数以及强类型检查等特性。

1979 年同年，Alexander Stepanov 创造了标准模板库 (STL)（我们可以在标准模板库中看到大量模板 template 操作），这是为了提高程序复用性而实现的库。第一个支持泛型概念的语言是 Ada，然而当时 Ada 并没有被广泛接受，而且当时 C++ 也并没有实现 STL。直到后来，STL 才逐步成为 C++ 不可分割的一部分。

Bjarne Stroustrup 的第一款“C with classes”编译器叫 Cfront，这个名字源自一个叫做 Cpre 的 C 编译器。Cfront 的机理是把“C with classes”的代码翻译成原生 C 代码。颇为有趣的一点是 Cfront 源码大部分是由“C with Classes”编写，这使得 Cfront 成为了一种自足执行的编译器（可以编译自身源码的编译器）。由于很难整合 C++ 的异常机制，Cfront 在 1993 年退出了历史的舞台，但是它对以后 C++ 编译器以及 Unix 操作系统的实现都产生了深远的影响。

1983 年，“C with Classes”语言更名为 C++。C 语言中“++”运算子的作用是对一个变量进行递增操作，由此我们多少可以知晓 Bjarne Stroustrup 对这种新语言的定位。这个时期，许多重要的特性被加入，其中包括虚函数、函数重载、引用机制（符号为 &）、const 关键字以及双斜线的单行注释（从 BCPL 语言引入）。

1985 年，Bjarne Stroustrup 的 C++ 参考手册《C++ Programming Language》出版，同年，C++ 的商业版本问世。由于当时 C++ 并没有正式的语言规范，因此《C++ Programming Language》成了业界的重要参考。1989 年，C++ 再次版本更新，这次更新引入了多重继承、保护成员以及静态成员等语言特性。

1990 年，《Annotated C++ Reference Manual》发布，同年，Borland 公司的商业版 Turbo C++ 编译器问世。Turbo C++ 附带了大量函数库，这一举措对 C++ 开发产生了极为深远的影响。虽然 Turbo C++ 上一个稳定的版本发布于 2006 年，之后就没有再更新过（截止到 2023 年），但当前该编译器仍被广泛使用。

1998 年，C++ 标准委员会发布了 C++ 语言的第一个国际标准—ISO/IEC 14882:1998，该标准即为大名鼎鼎的 C++98。C++98 的提出，《The Annotated C++ Reference Manual》功不可没。同时，1979 年开始研发的标准模板库（Standard Template Library，STL）也被纳入了该版标准中。

2000 年，Bjarne Stroustrup 推出了《The C++ Programming Language》特别版 (Special Edition)，书中内容根据 C++ 标准进行了更新。

2003 年，标准委员会针对 98 版本中存在的诸多问题进行了修订，修订后发布了 C++03。

2005 年，C++ 标准委员会发布了一份技术报告（Technical Report，TR1）详细说明了计划引入 C++ 的新特性。这个新标准被非正式地命名为 C++0x，因为其预计会在本世纪第一个十年的某个时间发布。有趣的是，直到 2011 年年中该标准才面世，相应的技术文档也随之出炉，一些编译器厂商也开始试验性地支持这些新特性。

2011 年中，新的 C++ 标准（C++11）面世。Boost 库对该版本影响很大，一些新的模块甚至直接衍生于 Boost 中相应的模块。一些新的语言特性，包括正则表达式（正则表达式详情）、完备的随机数生成函数库、新的时间相关函数，原子操作支持、标准线程库（2011 之前，C 和 C++ 语言均缺少对线程的支持，需要额外引入其他线程库来支持多线程，比如可以参考早期的 C++ 教材《Thinking in C++》）、一种能够和某些语言中 foreach 语句达到相同效果的新的 for 语法、auto 关键字、新的容器类、更好的 union 支持、数组初始化列表的支持以及变参模板的支持等等。

2014 年 8 月 18 日，经过 C++ 标准委员投票，C++14 标准获得一致通过。C++14 标准是‘ISO/IEC 14882:2014 Information technology – Programming languages – C++’的简称 。在标准正式通过之前，原名 C++1y。C++14 标准的委员会草案 N3690 于 2013 年 5 月 15 日发表。草案文档经过一定的文字润色和修正之后，将在年内提交给 ISO 作为正式的 C++ 标准发布。

C++17 是继 C++14 之后，C++ 编程语言 ISO/IEC 标准的下一次修订的非正式名称。而就在 2017-12-5，ISO C++ 委员会正式发布了 C++ 17 标准，官方名称为 ISO/IEC 14882:2017。基于 C++11，C++17 旨在使 C++ 成为一个不那么臃肿复杂的编程语言，以简化该语言的日常使用，使开发者可以更简单地编

写和维护代码。

2020 年，C++20 标准在布拉格的会议上由 WG21 进行了技术定稿，同年 9 月 4 日草案获得批准后，C++20 目前正处于最终批准过程中。相比 C++17，C++20 引入了新的语言特性，如概念、模块、操作符“<=>”、协程、指定初始化、新标准属性等。C++20 库标准还加入了范围、特性测试宏和位操作等。

目前，C++ 支持面向过程编程（C 语言等面向过程的语言的主要特性），数据抽象编程（隐藏后台实现过程的编程），面向对象编程，泛型编程以及函数式编程 (Functional programming，中文又被称为函数程序设计、泛函编程)。

三　编译器

有这么几种常用的 C/C++ 编译器：

- CLang：CLang 项目是为 LLVM 项目的 C 语言系列（C、C++、Objective C/C++、OpenCL、CUDA 和 RenderScript）中的语言提供了语言前端和工具基础设施。提供了与 GCC 兼容的编译器驱动程序（clang）和与 MSVC 兼容的编译器驱动器（clang-cl.exe）。
- MSVC(Microsoft Visual C++)：是 Microsoft 针对 C、C++ 和 C++/CX 编程语言的编译器。针对 C 语言的也被叫做 MSC 编译器。
- GCC(GNU Compiler Collection，GNU 编译器套装)：GNU 计划制作的一种优化编译器，支持各种编程语言、操作系统、计算机系统结构，后面会详细介绍。

这里我们详细讲解一下 GCC 和 GNU 之间的关系。有些人会好奇 GNU 似乎是一种开源协议，但又跟程序编译有一定的关系。

GNU 计划（英语：GNU Project，“GNU is Not Unix”的递归缩写），又译为革奴计划，是一个自由软体集体协作计划，1983 年 9 月 27 日由 Richard Matthew Stallman(通常被称为 RMS) 在麻省理工学院公开发起。它的目标是创建一套完全自由的操作系统，称为 GNU。RMS 最早在 net.unix-wizards 新闻组上公布该消息，并附带一份《GNU 宣言》等解释为何发起该计划的文章，其中一个理由就是要“重现当年软件界合作互助的团结精神”。

借助 GNU 项目，RMS 又发起了自由软件运动。他迄今为止一直是 GNU 项目的组织者，作为主要开发者的他开发了一些被广泛使用的 GNU 软件，其中就包括大名鼎鼎的 GCC、GDB、GNU Emacs。在 1985 年 10 月他创立了自由软件基金会。

RMS 为自由软件基金会的 GNU 项目所撰写了 GNU 通用公共许可协议（英语：GNU General Public License，缩写 GNU GPL 或 GPL），是被广泛使用的自由软件许可证，给予了终端用户运行、学习、共享和修改软件的自由，并授予计算机程序的用户自由软件定义（The Free Software Definition）的权利。

1983 年底，为了引导 GNU 操作系统，RMS 向阿姆斯特丹编译器套件（自由大学编译器套件）的作者 Andrew Stuart 请求在 GNU 上允许使用该编译器；但是 Andrew Stuart 告知他该编译器仅对大学免费。因此，他打算开发一个不同的编译器，GCC 就诞生了。GCC 原名为 GNU C 语言编译器（GNU C Compiler），因为它原本只能处理 C 语言。后来，新的 GCC 编译器可以编译 C++ 语言；后来又为 Fortran、Pascal、Objective-C、Java、Ada，Go 等其他语言开发了前端。C 和 C++ 编译器也支持 OpenMP 和 OpenACC 规范。

之后，GCC 改名为 GNU 编译器套装（英语：GNU Compiler Collection，缩写同样为 GCC）作为 GNU 计划制作的一种优化编译器，支持各种编程语言、操作系统、计算机系统结构。该编译器是以 GPL 及 LGPL 许可证所发行的自由软体，也是 GNU 计划的关键部分，还是 GNU 工具链的主要组成部份之一。GCC（特别是其中的 C 语言编译器）也常被认为是跨平台编译器的事实标准。截至 2019 年，GCC 大约有 1500 万行代码，是现存最大的自由程序之一。它在自由软件的发展中发挥了重要作用，不仅是一个工具，还是一个典例。

四　小结

那么关于 C++ 语言的发展史就基本介绍完了，在整理这段历史时，我也了解了不少其他方面的内容，比如各种同时期的语言与 C++ 之间的相互影响、各计算机大佬之间的博弈和斗争等，科研与程序的发展史也是一个非常有趣的、充满趣味性的历史。

参考文献

[1] https://zh.wikipedia.org/zh-hans/C 语言的历史

[2] http://c.biancheng.net/view/190.html

[3] https://cloud.tencent.com/developer/article/1518720

[4] https://www.jianshu.com/p/0490bf3fb893

[5] https://blog.csdn.net/m0_59718258/article/details/118251750

[6] https://www.51cto.com/article/697169.html

[7] https://zh.wikipedia.org/zh-hans/PDP-11

[8] https://blog.csdn.net/qq_41453285/article/details/103506107