中国地质大学
China University of Geosciences

绪论——C++ Introduction

主 讲：王红平

Email：wanghp@cug.edu.cn

内容提要

- 程序语言简介 - Introduction to Programming Languages
- C/C++简介 - Introduction to C++
- C++开发简介 - Introduction to C++ Development

1、Introduction to Programming Language

- 计算机程序（通常也被称为应用程序）是一组计算机可以执行一些任务的指令。创建程序的过程称为编程。程序员通常通过生成源代码（通常缩写为代码）来创建程序，源代码是键入一个或多个文本文件的命令列表。
- 组成计算机和执行程序的物理计算机部件的集合称为硬件。当计算机程序被加载到存储器中并且硬件顺序执行每个指令时，这称为运行或执行程序。

1、Introduction to Programming Language

- 计算机程序与源代码
- 机器语言
- 汇编语言
- 高级语言
- 编译与解释

计算机程序和源代码

- 计算机程序（通常也被称为应用程序）是一组计算机可以执行一些任务的指令。创建程序的过程称为编程。程序员通常通过生成源代码（通常缩写为代码）来创建程序，源代码是键入一个或多个文本文件的命令列表。
- 组成计算机和执行程序的物理计算机部件的集合称为硬件。当计算机程序被加载到存储器中并且硬件顺序执行每个指令时，这称为运行或执行程序。

机器语言

- 计算机CPU是无法解读C++语言的。CPU可以直接理解的有限指令集称为机器代码（机器语言或指令集）。
- 这是一个示例机器语言指令： 10110000 01100001
- 当计算机最初发明时，程序员必须直接用机器语言编写程序，这是一件非常困难和耗时的事情。
- 机器语言Wikipedia

汇编语言

- 因为人类对阅读和理解机器语言很难理解，所以就发明了汇编语言。在汇编语言中，每个指令由短缩写（而不是一组位）标识，并且可以使用名称和其他数字。
- 以下是汇编语言中与上述相同的指令： mov al, 061h
- 汇编语言比机器语言更容易读取和写入。但是，CPU无法直接理解汇编语言。
- 汇编语言也有缺点
  - 汇编语言仍需要大量指令来完成简单的任务。虽然单个指令本身在某种程度上是人类可读的，但了解整个程序正在做什么可能具有挑战性（这有点像试图通过单独查看每个字母来理解句子）。
  - 汇编语言是面向指令集的，针对不同指令集的硬件，需要重写或修改相应的代码。如：Intel CPU 和 AMD CPU的指令集就不完全相同。

高级语言

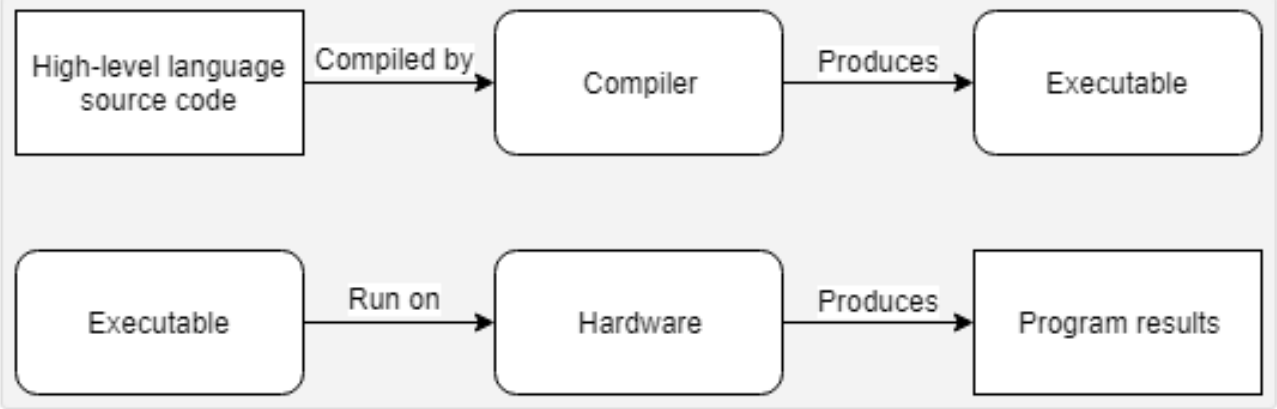
- 为了解决可读性和可移植性问题，出现了新的编程语言，如C，C ++，Pascal（以及后来的Java，Javascript和Perl等语言）。这些语言被称为高级语言，它们旨在允许程序员编写程序而无需关心程序将运行在何种类型的计算机上。
- 这是与C/C ++中相同的指令： a = 97;
- 与汇编程序非常相似，用高级语言编写的程序必须翻译成计算机可以运行的格式。两种主要方式：编译和解释。

不同类程序语言间的对比

语言	执行效率	可读性	移植性
机器语言	高	晦涩难懂 👎👎	👎👎
汇编语言	高	指令别名，程序逻辑仍难理解 👎	👎
高级语言	高	可读性好 👍	👍
	中	接近自然语言 👍👍	👍👍
自然语言	？	自然语言 👍👍👍	👍👍

编译器VS.解释器

- 编译过程的简化表示

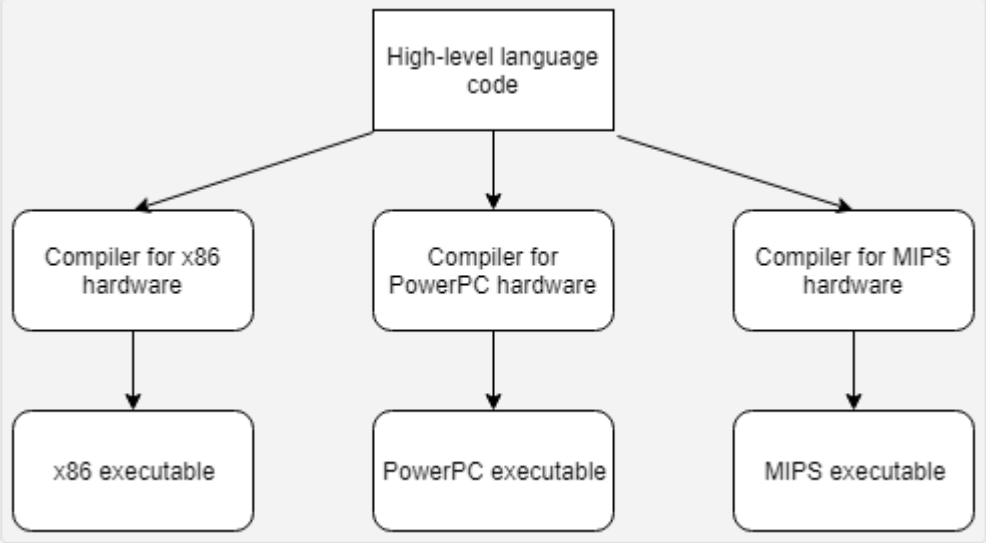


- 解释过程的简化表示

- 有关编译与解释的讨论

高级语言的特性

- 高级语言更容易读写，因为命令更接近我们每天使用的自然语言。
- 高级语言程序更为简洁，也更容易理解，相同的任务所需的指令比与低级语言少。如在C++中，a=b*2+5;在汇编语言中，这将需要5或6个不同的指令。
- 可以为许多不同的系统编译（或解释）程序，并且不必将程序更改为在不同的CPU上运行（您只需为该CPU重新编译）。如：

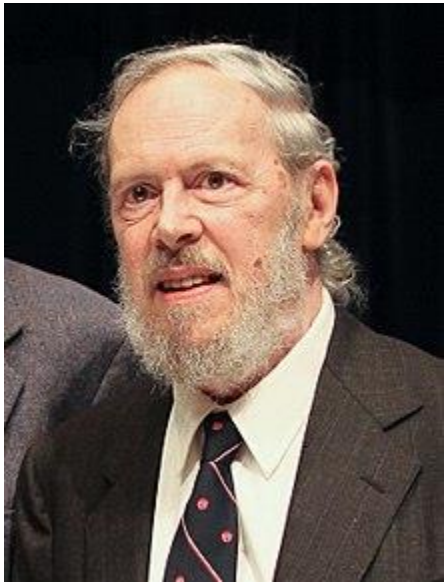


2、Introduction to C/C++

- C++的前世C
- C++的诞生
- C和C++的哲学
- Question about C++?

C++的前世C

- C语言由贝尔电话实验室的Dennis Ritchie于1972年开发而成，主要用于系统编程语言（编写操作系统的语言）
- C最终变得如此高效和灵活，以至于1973年，Ritchie和Ken Thompson使用C重写了大部分UNIX操作系统。
- 1978年，Brian Kernighan和Dennis Ritchie出版了一本名为“The C Programming Language”的书。K&R书中提供了C语言的非正式规范，并成为事实上的标准。当需要最大的可移植性时，程序员会坚持K&R中的建议，因为当时大多数编译器都是按照K&R标准实现的。



Dennis Ritchie

Brian Kernighan

C++的诞生

- C++（发音为see plus plus）是由Bjarne Stroustrup在贝尔实验室从1979年开始开发的。作为C的扩展，C ++为C语言添加了许多新功能。C++声名鹊起的主要原因在于它是一种面向对象的语言。
- C++于1998年由ISO委员会批准，并于2003年再次批准（称为C++ 03）。自那时起，C ++语言（C++ 11，C++ 14和C++ 17，在2011年，2014年和2017年得到批准）的三个主要更新已经完成，为该语言添加了额外的功能。特别是C++ 11为该语言添加了大量新功能。

Bjarne Stroustrup

C和C++的哲学

- C和C++的基本设计理念可以概括为：“信任程序员” —— 既美妙又危险。
- C++旨在让程序员可以高度自由地完成他们想要的任务。然而，这也意味着语言通常不会阻止你做一些没有意义的事情。如果没有意识到，新程序员可能会陷入相当多的陷阱。这就是为什么你应该知道在C/C ++中做什么和不能做什么几乎一样重要的原因之一。

Question：C++擅长什么？

- 答：C++在需要高性能和精确控制内存和其他资源的情况下表现出色。以下是一些常见的应用程序类型，最有可能用C ++编写：
  - 视频游戏
  - 实时系统（例如运输，制造等.....）
  - 高性能金融应用（例如高频交易）
  - 图形应用程序和模拟
  - 生产力/办公应用
  - 嵌入式软件
  - 音视频处理

Question:是否需要先行了解C?

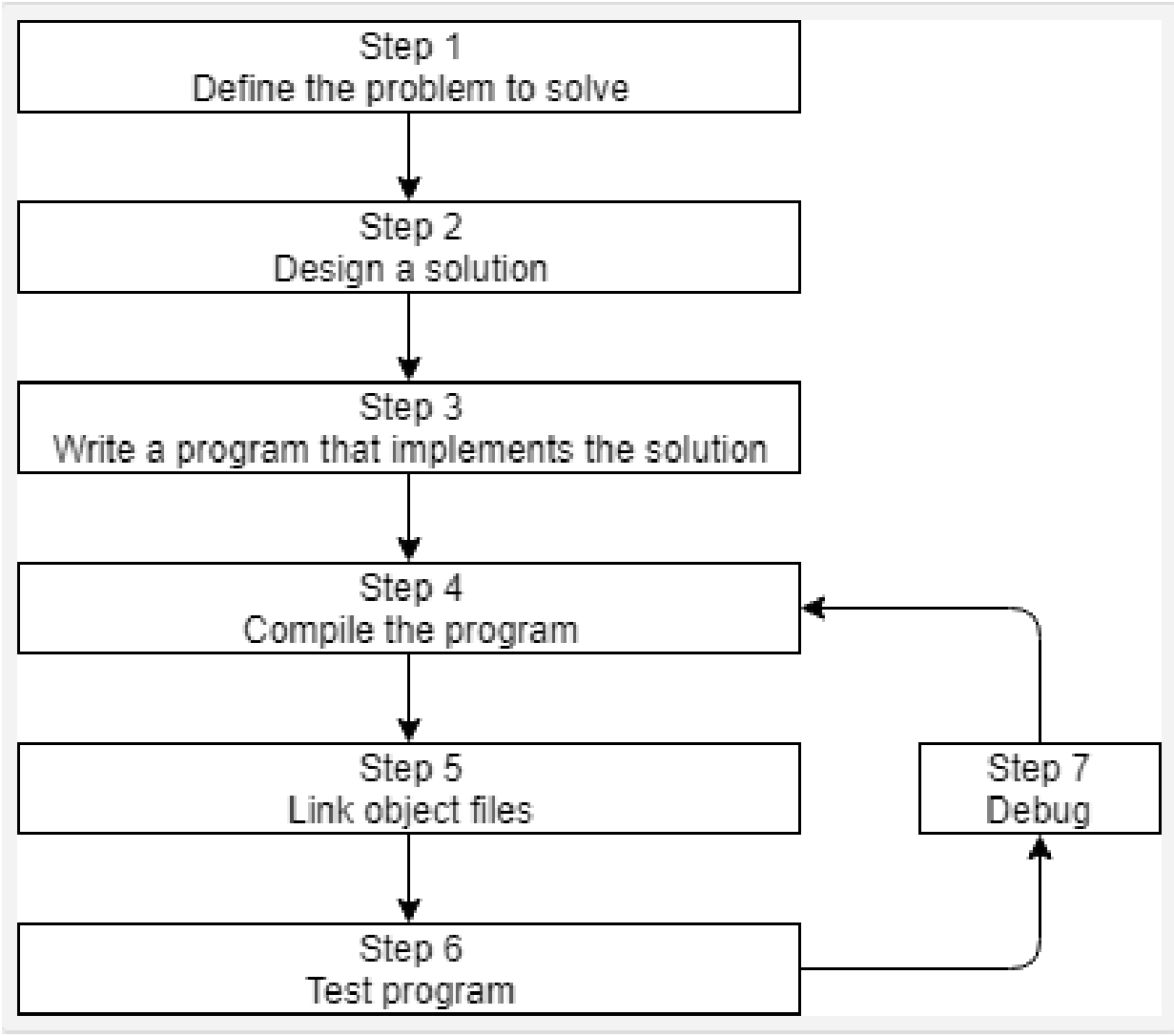
- 问：在学习C++之前，我是否需要了解C?
- 答：不！从C++开始是完美的，我们会教你一路上需要知道的一切（包括要避免的陷阱）。
- 一旦你了解了C ++，如果你有需要，学习标准C应该很容易。目前，C主要用于小众用例：在嵌入式设备上运行的代码，当您需要与只能与C接口等的其他语言进行交互时.....对于大多数其他情况，建议使用C ++。

3、Introduction to C++ development

- C++开发流程
  - ✓ Step 1：定义要解决的问题
  - ✓ Step 2：确定如何解决问题
  - ✓ Step 3：编写程序
  - ✓ Step 4：编译源代码
  - ✓ Step 5：链接目标文件和库
  - ✓ Step 6：调试
  - ✓ Step 7：测试

C++开发流程

- C++开发流程简图
  - 定义问题
  - 设计解决方案
  - 编码实现
  - 编译代码
  - 链接程序
  - 测试程序
  - 调试程序



第1步：定义要解决的问题

- 这是“什么”步骤，你需要找出需要解决的问题。提出你想要编程的最初想法可能是最简单的步骤，也可能是最难的步骤。但从概念上讲，它是最简单的。你只需要一个可以很好定义的想法，并为下一步做好准备。
- 这里有一些例子：
  - 我想编写一个程序，允许我输入许多数字，然后计算平均值。
  - 我想写一个生成二维迷宫的程序，让用户浏览它。如果用户到达目的地，用户将获胜。
  - 我想写一个程序，读取股票价格文件并预测股票是涨还是跌。

第2步：确定如何解决问题

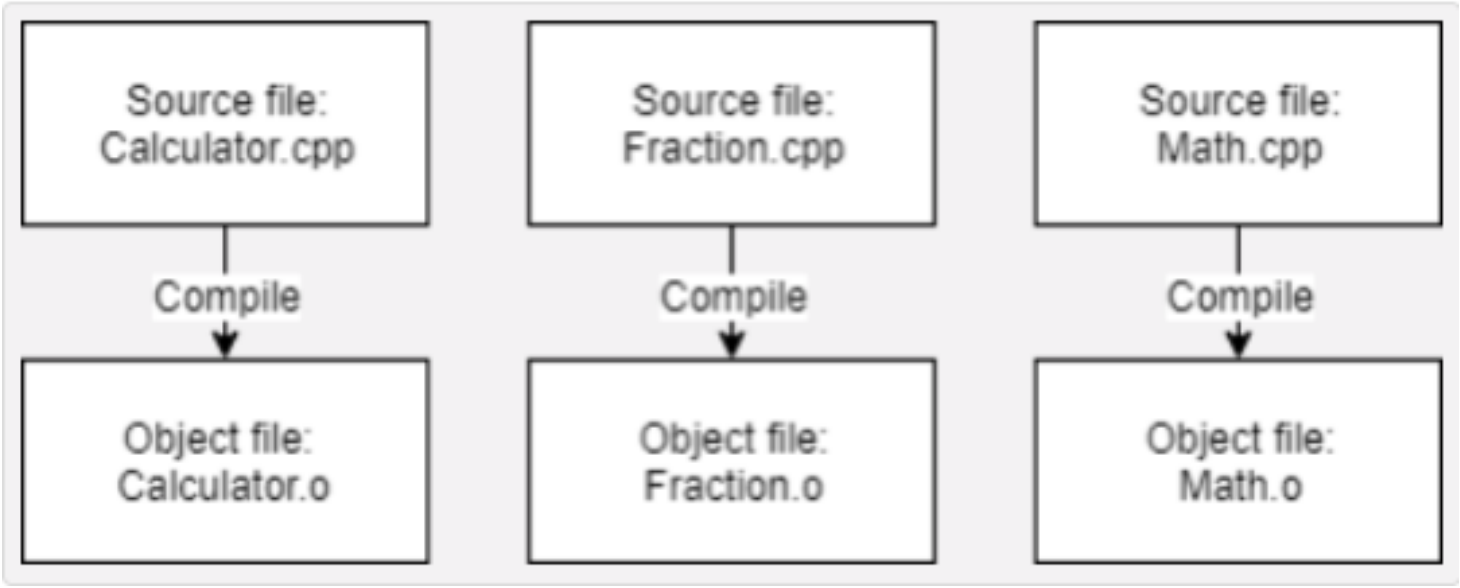
- 这是“如何”步骤，你可以在其中确定如何解决步骤1中提出的问题。这也是软件开发中最容易忽略的步骤。问题的关键在于有很多方法可以解决问题。但是，其中一些解决方案很好，而其中一些解决方案很糟糕。通常，程序员会得到一个想法，坐下来，并立即开始编写解决方案。这通常会产生一个属于坏类别的解决方案。（权衡、取舍）
- 通常，良好的解决方案具有以下特征：
  - 它们很简单（不会过于复杂或混乱）。
  - 它们有很好的文件记录（特别是在任何假设或限制的情况下）。
  - 它们是模块化构建的，因此可以在以后重复使用或更改部件，而不会影响程序的其他部分。
  - 它们非常稳健，可以在出现意外情况时恢复或提供有用的错误消息。

第3步：编写程序

- 为了编写程序，我们需要两件事：
  - 首先我们需要编程语言的知识，这正是本门课程所学的内容！
  - 其次，我们需要一名编辑。可以使用您想要的任何编辑器编写程序，甚至可以像记事本或Unix的vi或pico那样简单。
- 专为编码而设计的典型编辑器具有一些使编程更容易的功能，包括：
  - 行号。当编译器给我们一个错误时，行编号很有用，因为典型的编译器错误会说明：某些错误代码/消息，第64行。如果没有显示行号的编辑器，找到第64行可能会非常麻烦。
  - 语法高亮和着色。语法高亮和着色会更改程序各个部分的颜色，以便更容易识别程序的不同组件。这是一个包含行号和语法高亮的C ++程序示例：
  - 明确的字体。非编程字体通常使得难以区分数字0和字母O，或数字1，字母l（小写字母L）和字母I（大写字母i）之间。一个好的编程字体将区分这些符号，以确保不会意外地使用一个符号代替另一个符号。

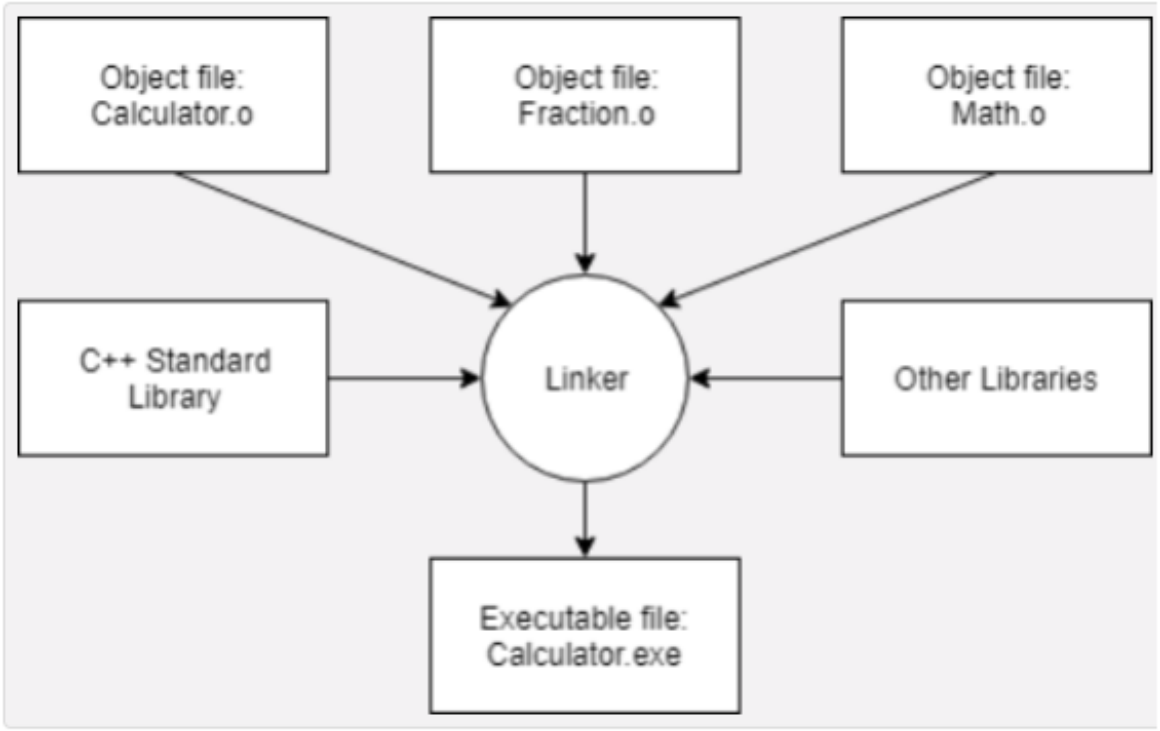
第4步：编译源代码

- C++编译器的工作是按顺序遍历程序中的每个源代码（.cpp）文件，并执行两项重要任务：
  - 1. 检查代码确保其遵循C ++语言的规则
  - 2. 将C++源代码转换为与机器语言对应的目标文件
- 如程序中有3个.cpp文件，编译器将生成3个目标文件：



第5步：链接目标文件和库

- 在编译器创建一个或多个目标文件之后，另一个称为链接器的程序启动。 链接器的工作有三个：
  - 1. 将编译器所生成的所有目标文件组合成一个可执行程序
  - 2. 除了能够链接目标文件之外，链接器还能够链接库文件
  - 3. 确保所有跨文件依赖关系的正确性



步骤6和7：调试和测试

- 这是有趣的部分（希望如此）！您可以运行可执行文件，看看它是否产生您期望的输出！
- 如果您的程序运行但无法正常工作，那么现在是时候进行一些调试以找出问题所在。我们将在后面的部分讨论如何测试程序以及如何更快地调试它们。
- 调试和测试工作特别重要。一般而言，调试和测试的工作量占到软件开发80%以上的工作量。

Home Work

- 1. 查找资料，整理一份C++发展史的里程碑
- 2. 编译器与解释器的各自特点与优势
- 3. 在你的机器上试着安装一个IDE环境，参考 https://www.learncpp.com/cpp-tutorial/installing-anintegrated-development-environment-ide/