1. **预备知识**
   1. **C++简介**

C++融合了3种不同的编程方式:C语言代表的过程性语言、C++在C语言基础上添加的类代表的面向对象编程、C++模板支持的泛型编程。

* 1. **C++简史**
     1. **C语言**

20世纪70年代早期，贝尔实验室的Dennis Ritchie致力于开发UNIX操作系统(操作系统是能够管理计算机资源、处理计算机与用户之间交互的一组程序。)为完成这项工作，Ritchie需要一种语言，它必须简洁，能够生成简洁、快速的程序，并能有效地控制硬件。

传统上，程序员使用汇编语言来满足这些需求，汇编语言依赖于计算机的内部机器语言。然而，汇编语言是低级(low-level)语言，即直接操作硬件。因此汇编语言针对于特定的计算机处理器，要将汇编语言移植到另一台计算机上，必须使用不同的汇编语言重新编写程序。

然而，UNIX是为在不同的计算机(或平台)上工作而设计的，这意味着它是一种高级语言。高级(high-level)语言致力于解决问题，而不针对特定的硬件。一种被称为编译器的特殊程序将高级语言翻译成特定计算机的内部语言。这样，就可以通过对每个平台使用不同的编译器来在不同的平台上使用同一个高级语言程序了。Ritchie希望有一种语言能将低级语言的效率、硬件访问能力和高级语言的通用性、可移植性融合在一起，于是他在旧语言的基础上开发了C语言。

* + 1. **C语言编程原理**

由于C++在C语言的基础上移植了新的编程理念，所以先看一下C语言所遵循的旧的理念。

一般来说，计算机语言要处理两个概念——数据和算法。数据是程序使用和处理的信息，而算法是程序使用的方法。C语言与当前主流的语言一样，在最初面世时也是过程性(procedural)语言，这意味着它强调的是编程的算法方面。从概念上说，过程化编程首先要确定计算机应采取的操作，然后使用编程语言来实现这些操作。程序命令计算机按一系列流程生成特定的结果。

随着程序规模的扩大，早期的程序语言(如FORTRAN和BASIC)都会遇到组织方面的问题。如，程序经常使用分支语句，根据某种测试的结果，执行一组和另一组指令。很多旧式程序的执行路径很混乱，几乎不可能通过阅读程序来理解它，修改这种程序简直是一场灾难。为了解决这种问题，计算机科学家开发了一种更有序的编程方式——结构化编程(structured programming)。C语言具有使用这种方法的特性。如，结构化编程将分支(决定接下来应执行哪个指令)限制为一小组行为良好的结构。C语言的词汇表中就包含了这些结构(for循环、while循环、do while循环和if else语句)。

另一个新原则是自顶向下(top-down)的设计。在C语言中，其理念是将大型程序分解成小型、便于管理的任务。如果其中的一项任务仍然过大，则将它分解为更小的任务。这一过程将一直持续下去，直到将程序划分为小型的、易于编写的模块。C语言的设计有助于使用这种方法，它推荐开发程序单元(函数)来表示各个任务模块。结构化编程技术反映了过程性编程的思想，根据执行的操作来构思一个程序。

* + 1. **面向对象编程**

虽然结构化编程的理念提高了程序的清晰度、可靠性，并使之便于维护，但它在编写大型程序，仍面临着挑战。为应对这种挑战，OOP提供了一种新方法。与强调算法的过程性编程不同的是，OOP强调的是数据。OOP不像过程性编程那样，试图使用问题满足语言的过程性方法，而是试图让语言来满足问题的要求。其理念是设计与问题的本质特性相对应的数据格式。

在C++中，类是一种规范，它描述了这种新型数据格式，对象是根据这种规范构造的特定数据结构。通常，类规定了可使用哪些数据来表示对象以及可以对这些数据执行哪些操作。

OOP程序设计方法首先设计类，它们准确地表示了程序要处理的东西。类定义描述了对每个类可执行的操作。然后便可以设计一个使用这些类的对象和程序。从低级组织(如类)到高级组织(如程序)的处理过程叫作自下向上(bottom-up)的编程。

OOP编程并不仅仅是将数据和方法合并为类定义。还有助于创建可重用的代码，这将减少大量的工作。信息隐藏可以保护数据，使其免遭不适当的访问。多态为运算符和函数创建多个定义，通过编程上下文来确定使用哪个定义，继承能够使用旧类派生出新类。OOP引入了很多新的理念，使用的编程方法不同于过程性编程。它不是将重点放在任务上，而是放在表示概念上。

* + 1. **C++和泛型编程**

泛型编程(generic programming)是C++支持的另一种编程模式。它与OOP的目标相同，即使重用代码和抽象通用概念的技术更简单。不过OOP强调的是编程的数据方面，而泛型编程强调的是独立于特定数据类型。它们的侧重点不同。OOP是一个管理大型项目的工具，而泛型编程提供了执行常见任务的工具，术语泛型(generic)指的是创建独立于类型的代码。C++的数据表示有多种类型——整数、小数、字符、字符串、用户定义的、由多种类型组成的复合结构。如，要对不同类型的数据进行排序，通常必须为每种类型创建一个排序函数。泛型编程需要对语言进行扩展，以便可以只编写一个泛型(即不是特定类型的)函数，并将其用于各种实际类型。

* + 1. **C++的起源**

与C语言一样，C++也是在贝尔实验室诞生的，Bjarne Stroustrup于20世纪80年代在这里开发出了这种语言。

Stroustrup比较关心的是让C++更有用，而不是实施特定的编程原理或风格。在确定C++语言特性方面，真正的编程需要比纯粹的原理更重要。Stroustrup之所以在C的基础上创建C++，是因为C语言简洁、适合系统编程、使用广泛且与UNIX操作系统联系紧密。C++的OOP方面是受到了计算机模拟语言Simula67的启发。Stroustrup加入了OOP特性和对C的泛型编程支持，但并没有对C的组件作很大的改动。因此，C++是C语言的超集。这意味着任何有效的C程序都是有效的C++程序。它们之间有些细微的差距，但无足轻重。C++程序可以使用已有的C软件库。库是编程模块的集合，可以从程序中调用它们。库对很多常见的编程问题提供了可靠的解决方法。

名称C++来自C语言的递增运算符++，该运算符将变量加1。名称C++表明，它是C的扩充版本。

计算机程序将实际问题转换为计算机能够执行的一系列操作。OOP部分赋予了C++语言将问题所涉及的概念联系起来的能力，C部分则赋予了C++语言紧密联系硬件练习硬件的能力。

在C++获得一定程度的成功后，Stroustrup才添加了模板，这使得进行泛型编程成为可能。C++融合了OOP、泛型编程和传统的过程性方法，这表明C++强调的是实用价值，而不是意识形态方法，这也是该语言获得成功的原因之一。

* 1. **可移植性和标准**