巍巍交大　百年书香

www.jiaodapress.com.cn
bookinfo@sjtu.edu.cn

策划编辑　李　勇
责任编辑　胡思佳
封面设计　刘文东

C++ 程序设计

扫描二维码
关注上海交通大学出版社
官方微信

ISBN 978-7-313-33743-6
9 787313 337436 >
定价：55.00元

C++程序设计

主编　孔德川

上海交通大学出版社
SHANGHAI JIAO TONG UNIVERSITY PRESS

河南省“十四五”普通高等教育规划教材
河南省本科高校新工科新形态教材

C++ 程序设计

主编　孔德川

上海交通大学出版社
SHANGHAI JIAO TONG UNIVERSITY PRESS

河南省“十四五”普通高等教育规划教材

河南省本科高校新工科新形态教材

C++程序设计

主　编　孔德川
副主编　王艳艳　张　磊
参　编　薛　然　王晓雷　张毛赛
王燕强　邱宏宙

上海交通大学出版社
SHANGHAI JIAO TONG UNIVERSITY PRESS

内容提要

本书是河南省“十四五”普通高等教育规划教材、河南省本科高校新工科新形态教材。本书通过理论讲解与案例实践相结合的方式，在系统阐述基础知识的同时强化实际应用，帮助学生深入理解C++语言中抽象的核心概念，清晰认识C++软件开发的实际流程，最终掌握使用C++进行软件开发的基本方法与关键技能。全书共分为12个模块，包括C++语言基础知识、基本数据类型与表达式、基本流程控制语句、函数、数组、指针、结构体和共用体、类与对象、继承与派生、输入/输出流、异常处理以及综合案例（图书管理系统）。

本书既适合作为普通高等教育C++程序设计课程的教材，也可作为计算机初学者自主学习的参考书。

图书在版编目（CIP）数据

C++程序设计 / 孔德川主编. -- 上海 ：上海交通大学出版社，2025. 10. -- ISBN 978-7-313-33743-6

Ⅰ. TP312. 8

中国国家版本馆CIP数据核字第2025G4B850号

C++程序设计
C++ CHENGXU SHEJI

主　　编：孔德川
出版发行：上海交通大学出版社　　地　　址：上海市番禺路951号
邮政编码：200030　　电　　话：021-64071208
印　　制：三河市龙大印装有限公司　　经　　销：全国新华书店
开　　本：850 mm×1 168 mm　1/16　　印　　张：15.5
字　　数：414千字
版　　次：2025年10月第1版　　印　　次：2025年10月第1次印刷
书　　号：ISBN 978-7-313-33743-6　　电子书号：ISBN 978-7-89564-511-0
定　　价：55.00元

前言

在数字化浪潮席卷全球的今天，C++ 作为计算机编程领域的中流砥柱，始终以其卓越的性能、高度的灵活性和广泛的适用性占据着不可替代的地位。从操作系统、大型游戏引擎到嵌入式设备控制程序、金融交易系统等，C++ 的身影无处不在，并驱动着无数复杂而关键的应用。无论是渴望探索计算机底层奥秘的技术爱好者，还是致力于开发高性能软件的专业开发者，掌握 C++ 编程能力都将成为他们开启技术大门的重要钥匙。这不仅是专业能力的提升，更是响应国家科技自立自强战略的号召、投身数字中国建设的重要实践，体现着新时代青年学子的责任担当。

本书正是为渴望系统学习 C++ 编程的读者精心打造的入门指南。它既适合零基础的编程新手，也能为有一定编程经验、希望深入理解 C++ 核心机制的学习者提供帮助。在内容编排上，本书秉持由浅入深、循序渐进的原则，从 C++ 语言的基本概念、语法规则入手，逐步深入面向对象编程、泛型编程等高级特性，确保读者能够扎实地构建知识体系，实现从编程“小白”到独立开发者的蜕变。

本书作为河南省本科高校新工科新形态教材，深度融合线上线下教学资源，以“人工智能 + 教育”模式重构学习生态。书中嵌入丰富的二维码链接，读者扫码即可获取知识讲解与动态演示视频，打破传统纸质教材的时空限制，助力读者实现自主探究式学习。同时，本书依托河南省课程思政样板课程建设成果，在知识传授中自然融入思政元素，并设置“育德树人”栏目，让读者在掌握编程技术的同时，厚植家国情怀，树立正确的价值观和职业操守。最后，超星平台设有与本书配套的在线智慧课程，可辅助读者学习 C++ 编程。

本书注重理论与实践紧密结合。每一个重要知识点都配有简洁易懂的代码示例，帮助读者快速理解和掌握该知识点；每一章都设置了丰富的习题，涵盖基础概念、应用实践等多个维度，助力读者巩固所学知识；教材末尾的综合案例更是将全书核心内容融会贯通，引导读者在实际项目中提升编程能力和培养解决问题的思维。此外，书中没有晦涩难懂的专业术语堆砌，采用生动、通俗的语言讲解复杂概念，让学习过程更加轻松顺畅。在案例设计中，融入具有社会价值的实践场景，引导读者关注技术对社会发展的推动作用，增强社会责任感。

学习编程是一场充满挑战与惊喜的旅程，C++ 的世界既深邃又迷人。希望本书能够成为读者可靠的伙伴，陪伴读者攻克每一个知识难关，领略 C++ 编程的独特魅力。编程之路没有

捷径，但只要保持好奇心并坚持学习，终将在代码的世界里收获满满的成就感。让我们一同开启这段精彩的 C++ 学习之旅，以代码为笔，书写属于新时代青年的科技强国篇章吧！

本书由河南科技学院孔德川任主编，河南科技学院王艳艳、新乡学院张磊任副主编，河南科技学院薛然、河南工学院王晓雷、河南科技学院张毛赛、河南科技学院王燕强和河南科技学院邱宏宙参与编写。

由于编者水平有限，书中不当之处恳请广大读者批评指正。

编　者

目录

模块 1　C++ 语言基础知识

模块 2　基本数据类型与表达式

模块 3 基本流程控制语句

模块 4 函 数

模块 5 数 组

模块 6 指 针

模块 7 结构体和共用体

模块 8 类 与 对 象

模块 9 继承与派生

模块 10 输入 / 输出流

模块 11 异 常 处 理

模块 12 综合案例（图书管理系统）

模块 1 C++ 语言基础知识

模块导读

在计算机编程语言的广阔天地中，C++ 以其独特的魅力和强大的功能占据着重要地位。C++ 脱胎于 C 语言，在保留对系统底层硬件的高效操控能力的同时，引入了面向对象编程理念，融合了过程式编程与面向对象编程范式，成为开发复杂软件系统的利器。在实际应用中，C++ 的身影无处不在：在游戏开发领域,《英雄联盟》《原神》等大型游戏的核心引擎模块，C++ 凭借其高执行效率，支撑游戏流畅的画面渲染与复杂的逻辑运算；在系统软件层面，Windows、Linux 等操作系统的部分组件也依赖 C++ 实现对硬件资源的精准管理。

本模块作为 C++ 学习的开篇，将带领读者追溯 C++ 语言的发展脉络，深入剖析其核心特性，详细讲解 C++ 编程环境的搭建流程，以及 C++ 程序从代码编写到成功运行的完整过程。通过本模块的学习，读者将建立对 C++ 语言的初步认知框架，为后续深入学习语法知识和开展编程实践筑牢根基。

学习目标

知识目标

（1）明晰 C++ 语言的发展历史、设计思想及其与其他编程语言的联系。

（2）掌握 C++ 面向对象编程（封装、继承、多态）与过程式编程的核心特点。

（3）了解常见 C++ 开发环境，熟悉程序编译、运行流程。

技能目标

（1）熟练完成 C++ 开发环境的安装与配置，独立编写、编译、运行简单程序。

（2）学会使用基础代码编辑与调试工具，具备初步的程序调试能力。

素质目标

（1）激发对 C++ 编程的学习兴趣，培养探索编程语言的热情。

（2）养成严谨规范的编程习惯，重视代码质量与可读性。

（3）提升自主学习和解决问题的能力，为后续进阶学习做好准备。

1.1 C++ 语言概述

视频
C++ 语言概述

C++ 语言是一种通用的编程语言，最初由本贾尼·斯特劳斯特卢普（Bjarne Stroustrup）在 20 世纪 80 年代初期开发。它是 C 语言的扩展，加入了面向对象编程（object-oriented programming，OOP）等高级特性，成为现代编程语言的重要组成部分。

1.1.1 C++ 语言的发展历史

C++ 语言的直接前身是 C 语言，后者是一种结构化编程语言，由丹尼斯·里奇（Dennis Ritchie）在 20 世纪 70 年代初期开发。1979 年，本贾尼·斯特劳斯特卢普在贝尔实验室开始了 C++ 的最初开发，称其为“C with Classes”（带有类的 C）。本贾尼·斯特劳斯特卢普当时的目标是通过在 C 语言中引入面向对象编程的特性，提高程序设计的抽象能力和可维护性。

1983 年，“C with Classes”语言正式改名为 C++，并首次提出了 C++ 的基本概念——类（class）和对象（object），使 C 语言具备了面向对象编程的功能。C++ 的关键特点之一是支持多态、继承和封装等面向对象特性，这些特性在 C 语言中无法实现。

随着 C++ 语言在软件开发和操作系统编程中的广泛应用，其功能不断扩展。1985 年，C++ 第一版（C++ 1.0）发布，它包含了基本的面向对象特性，如类、构造函数、继承、函数重载等。然而，最初的 C++ 语言并未进行严格的标准化，导致不同编译器存在差异。

为了解决这一问题，ISO/IEC（国际标准化组织 / 国际电工委员会）于 1990 年成立了一个标准委员会，开始对 C++ 进行标准化工作。经过长时间的讨论和修改，C++ 标准化委员会于 1998 年发布了 C++ 98 标准，这是第一个正式的 C++ 标准，定义了 C++ 语言的核心语法、库和运行时特性。

在 C++ 98 标准发布之后，C++ 语言并没有出现大的语法改变，只进行了小幅修正。2003 年发布了 C++ 03 标准，其目的是修复 C++ 98 标准中的一些错误和不一致之处，并没有引入新的特性。

2011 年发布的 C++ 11 标准是 C++ 历史上的重要版本，引入了大量的新特性，显著提升了 C++ 的现代化水平。其新特性如下。

（1）自动类型推导（auto）。

（2）智能指针（如 unique_ptr、shared_ptr）。

（3）并发编程支持（如 thread、mutex）。

（4）lambda 表达式（允许在函数内部定义匿名函数）。

（5）基于范围的 fol 循环（range-based for loop）。

（6）初始化列表。

C++ 11 标准的推出使得 C++ 语言在多核处理器和现代硬件上具有了更好的性能。

2014 年发布的 C++ 14 标准是对 C++ 11 标准的一个小修订版本，主要是对一些小错误的修复，并未引入太多新特性。2017 年的 C++ 17 标准进一步增强了语言的功能，引入文件系统（file system）库、结构化绑定声明等新的特性，优化模板库和编译器性能，提升语言的现代性和可移植性。

2020 年的 C++ 20 标准对 C++ 语言进行了重大扩展，增加了模块、概念、协程、范围（ranges）库等特性。C++ 20 标准的发布进一步增强了 C++ 在泛型编程、并发编程和异步编程等方面的能力。2024 年发布的 C++ 23 标准对 C++ 20 标准进行了修订和增强，加入了一些小的改进和新特性。

随着硬件和软件技术的发展，为适应现代编程的需求，C++ 语言将继续重点增加对并行计算、高性能计算和云计算的支持，进一步增强其语法简洁性与表达能力。

1.1.2 C++ 语言的特点

C++ 语言结合了面向对象编程和过程化编程的优点，是一门功能强大、灵活的语言，具备以下主要特点。

1）面向对象编程

C++ 语言支持面向对象编程的基本特性。

（1）封装（encapsulation）：C++ 支持将数据和操作数据的方法（函数）包装为类，提供更高的抽象能力。

（2）继承（inheritance）：C++ 支持类的继承，子类可以继承父类的成员和方法，支持多级继承和多重继承。

（3）多态（polymorphism）：通过虚函数机制实现运行时多态，允许不同类实现相同接口（interface）。

2）高效的底层控制

C++ 继承了 C 语言的高效性，允许开发者直接操作内存，使用指针、引用等底层功能。此外，C++ 支持内联函数、运算符重载等功能，可以对性能进行优化。

3）泛型编程

C++ 通过模板（template）支持泛型编程，使得代码可以处理多种数据类型而无须重复编写相同的代码。模板不仅支持函数模板，还支持类模板。

4）标准库

C++ 拥有强大的标准模板库（STL），在标准模板库中包括容器类（如 vector、list、map 等）、算法（如排序、查找等）和迭代器。在 C++ 11 标准及以上版本引入了其他库，如并发编程库、文件系统库、时间库等。

5）高效的编译时和运行时性能

C++ 提供了高效的编译时计算能力（如模板元编程）和运行时性能，特别适合开发高性能的应用程序。

1.1.3 C++ 语言的应用

C++ 语言的应用范围非常广泛，尤其在需要高性能的应用场景中，是理想的编程语言选择。以下是 C++ 语言的几个主要应用领域。

1）系统软件开发

C++ 经常用于开发操作系统、编译器和系统工具等底层软件。例如，Microsoft Windows 和 Linux 内核的部分组件都是使用 C++ 编写的。

2）游戏开发

C++ 的高性能和对硬件的直接控制，使它成了游戏开发的首选语言。许多大型游戏引擎（如 Unreal Engine、CryEngine）的核心模块都是用 C++ 开发的。

3）嵌入式系统和实时系统

C++ 可以用于嵌入式系统开发，特别是在需要高性能和实时响应的场合，如航空航天、汽车控制系统、智能硬件等领域。

4）金融与高频交易

在金融领域的高频交易系统中，C++ 的低延迟特性非常重要。许多交易平台和算法交易系统都使用 C++ 来实现高效的计算和数据处理。

5）科学计算和工程应用

C++ 强大的数值计算能力和并行计算能力，使其能应对复杂的科学计算和工程应用模拟，尤其在需要大量数值计算和并行计算的领域，如图像处理、气象模拟、物理仿真、计算流体力学等。

6）人工智能与机器学习

C++ 在人工智能和机器学习的应用中也有较为广泛的使用。许多 AI 库（如 TensorFlow、Caffe）都是用 C++ 开发的，C++ 的高性能对大规模数据的训练和处理至关重要。

7）Web 浏览器

许多浏览器（如 Google Chrome、Mozilla Firefox）都采用了 C++ 来开发底层渲染引擎和关键组件，C++ 高效的执行效率能够帮助浏览器处理大量的数据并优化响应速度。

育德树人

科技自主创新与民族担当

当前国家大力倡导科技自主创新，高级编程语言的发展正体现了这一趋势。我国自主研发的编程语言在一些特定领域崭露头角。在编程领域，我们不能依赖国外技术，要有自主研发的决心。就像老一辈科学家克服困难搞科研一样，我们要肩负起民族担当，为我国编程技术自主发展贡献力量，让中国的编程语言走向世界。

1.1.4 C++ 语言与 C 语言

虽然 C++ 是 C 语言的超集，但它引入了许多新的编程特性和理念，使得它在许多开发场景中

具备了 C 语言无法提供的优势。C 语言是一种过程式编程语言，强调通过函数来执行一系列操作。程序员将代码组织成一个个过程（函数），程序从上到下按顺序执行。C 语言不提供内置的面向对象编程机制，缺乏对类、继承、多态、封装等特性的支持。

C++ 引入了面向对象编程思想，它在 C 语言的基础上增加了对类、对象、继承、多态、封装等概念的支持。C++ 通过类封装数据和方法，使得数据和操作数据的函数能聚集在一起；而对象是类的实例，类为对象定义了属性和行为。同时，C++ 允许类之间的继承，使得代码可以重用和扩展。子类可以继承父类的成员变量和成员函数，并可以重写（override）父类的函数，也可以通过基类指针或引用调用子类函数，动态绑定（dynamic binding）到子类的实现，而不是静态绑定到基类函数。

C 语言不支持模板，也没有泛型编程机制，所有的函数和数据结构都需要针对具体的类型编写。C++ 通过模板实现允许函数和类在编译时接受类型作为参数，从而支持泛型编程。开发者可以编写不依赖特定数据类型的代码，实现通用算法和数据结构，使代码编写更加灵活和可重用。例如，C++ 的标准模板库就是基于模板实现的，提供了向量（vector）、链表（list）、映射（map）等数据结构和相关算法。

在 C 语言中，内存的管理需要开发者手动完成，需要显式地使用 malloc() 和 free() 来分配和释放内存。在使用过程中，经常会因为内存泄露、悬挂指针等问题，导致程序崩溃或内存浪费。在 C++ 中，在手动内存管理机制上，通过引入智能指针提供了更安全的内存管理方式，减少了内存泄露的风险。智能指针能够自动管理内存的释放，当指针超出作用域时，内存会自动释放。

C 语言的标准库主要包括一些基本的函数库，功能相对简单，而 C++ 则拥有强大的标准模板库，可以提供许多常见的容器类（如 vector、list、deque、map、set 等）和算法（如排序、查找等）。STL 通过模板实现泛型编程，使得数据结构和算法的组合更加灵活。STL 的使用极大简化了 C++ 程序的开发，使得程序员能够专注于更高层次的逻辑，而无需从头开始实现常用的功能。

C 语言不支持异常（exception）处理机制。错误处理一般依赖函数返回值，程序员需要手动检查每个函数调用的返回值，以判断是否发生了错误。C++ 引入了异常处理机制，将错误处理与正常代码分离，使得代码更清晰，且在出错时能够采取合适的处理措施，使错误的捕获与处理更加结构化和集中，可以提高程序的健壮性。

C 语言不支持函数重载，函数名必须唯一。C++ 支持函数重载和运算符重载，允许开发者根据参数类型或个数不同，定义多个版本的同名函数，也允许自定义运算符行为，使得自定义类型（如类）的实例可以直接使用这些运算符进行操作。

C++ 语言相较于 C 语言的优势在于其支持面向对象编程、引入了类和对象等高级特性，提供了强大的标准模板库，并且在内存管理、异常处理和类型安全等方面有更强的支持。C++ 的灵活性更高，能够在多种编程范式之间进行切换，适用于更为复杂的项目开发和更高效的程序设计。

1.2 认识 C++ 语言程序

视频
认识 C++ 语言程序

下面通过一个简单的程序来了解 C++ 语言程序的基本构成，该程序的功能是计算两个整数 a 和 b 的乘积。

例 1-1 计算两个整数 a 和 b 的乘积。

```
#include <iostream>
using namespace std;
int main()
{
    int a,b,c;                          //定义整型变量a、b、c
    cout<<"请输入a和b的数值: "<<endl;    //提示用户输入数值
    cin>>a>>b;                          //从键盘输入数值存入变量a、b中
    c=a*b;
    cout<<"a和b的乘积是: "<<c<<endl;
    return 0;
}
```

C++ 源程序扩展名为 .cpp，编辑完代码后，需要通过 C++ 编译器将源代码转换为目标代码（.o 或 .obj 文件），再转换为计算机能够理解的机器语言。编译时，编译器会检查代码的语法是否正确。如果有错误，编译器会报错并停止编译。编译生成的目标文件仍然是一个部分完成的程序，它不能直接执行。接下来，需要将目标文件与库文件（如标准库）链接，生成可执行文件。链接的过程是将目标文件与外部库进行连接，解决函数、变量等符号的引用。在这个例子中，链接器将把目标文件与所有必需的库（如 iostream）结合，生成一个可执行文件。

从例 1-1 中可以看出，C++ 程序通常由编译预处理、命名空间、主函数、输入和输出、注释等组成，下面逐一进行讲解。

1.2.1 编译预处理

#include 是 C++ 预处理指令之一，其作用是将指定的文件内容插入当前文件中。例 1-1 中的 #include <iostream> 语句通过引入 iostream（输入 / 输出流库），使程序具备输入和输出功能。include 指令后可以用尖括号形式或双引号形式进行文件引入，尖括号（< >）形式多用于包含标准库文件或系统文件，编译器首先会在标准库的路径中查找该文件，然后才会查找用户提供的路径；双引号形式（""）多用于包含自定义的头文件或当前目录中的文件，编译器首先会在当前文件所在目录中查找文件，然后才会在标准路径中查找。

1.2.2 命名空间

命名空间（namespace）是 C++ 中用于组织代码的机制，允许将不同的标识符（如变量、函数、

类等）放在一个逻辑组，避免不同部分代码中的名称冲突。命名空间使得代码更具结构性，便于管理大规模项目中的符号。命名空间通过 namespace 关键字进行定义，使用 :: 来访问命名空间中的元素。为了简化代码书写，可以使用 using 声明引入整个命名空间或单个元素。这样可以直接使用命名空间中的成员，而不必每次都写 namespace:: 前缀。

1.2.3 主函数

在 C++ 程序中，主函数（main）是程序的入口点，为程序的控制流程提供了基本框架，是每个 C++ 程序必须具备的基本结构。当程序启动时，操作系统首先会调用程序代码文件中的 main 函数，程序的执行从这里开始。每个 C++ 程序有且只有一个 main 函数，程序的执行流程从 main 函数开始，并在 main 函数中结束。当程序需要用户通过命令行来输入参数运行时，需要使用标准的 main 函数头部定义形式 int main(int argc，char *argv[]) 来进行定义，其中，argc 是一个整数，表示传递给程序的命令行参数的个数，因为 argv[0] 通常是程序的名称，所以 argc 至少为 1；argv 是一个字符串数组（array），主要用于给程序传递每个命令行参数，通常 argv[0] 是程序的名称，argv[1]、argv[2] 等是其他命令行参数。当程序不需要从命令行接收任何输入时，可以简化为不带 argc 和 argv 的形式，如例 1-1 所示。main 函数的返回值类型是 int，当 main 函数结束时通常会通过 return 语句返回一个整数值给操作系统，当返回值为 0 时，表示程序正常运行结束，返回值非零表示程序发生了错误或异常。

1.2.4 输入和输出

在 C++ 中，输入 / 输出（input/a4put,I/O）是程序与用户或外部环境进行交互的基本方式。C++ 通过标准库 <iostream> 中的输入 / 输出流对象（cin 和 cout）来实现输入和输出。在例 1-1 中，cin 对象通过 >> 运算符完成从键盘接收输入数据分别存入变量 a、b 中，cout 对象通过 << 运算符将字符串“a 和 b 的乘积是：”和变量 c 的值输出到屏幕，并通过 endl 实现换行。

1.2.5 注释

注释是解释代码功能的文本，主要用于帮助程序员更快理解代码的功能和设计，编译器在编译代码时会忽略这些文本。注释应该简洁、清晰，代码中应避免写过多的注释，尤其是那些显而易见的注释，正确、适当地使用注释可以显著提高代码的可读性和可维护性。C++ 中的注释方式主要有两种：单行注释和多行注释。单行注释用于注释掉一行代码的某一部分或整行。单行注释以 // 开始，后面的内容会被视为注释，直到该行结束。多行注释可以一次性注释掉多行代码或代码块，注释时可以跨越多行。以 /* 开始，以 */ 结束，被 /* 和 */ 包围的代码部分均会被视为注释，不会被编译。

1.2.6 代码编写规范

规范的 C++ 代码编写能够有效提高代码的可读性、可维护性、可扩展性，并确保代码的一致

性。从学习 C++ 编程伊始，养成良好的代码编写习惯，遵循良好一致的代码编写规范有助于后期团队协作开发和代码审查。C++ 代码常见的编写规范主要包括代码风格、命名规则、注释规范、代码结构和异常处理等方面。

1. 命名规范

类名：使用大驼峰命名法（Pascal Case），首字母大写，每个单词的首字母大写。例如，MyClass、EmployeeDetails。

函数名：使用小驼峰命名法（camel Case），首字母小写，后续单词首字母大写。例如，getName()、calculateSum()。

变量名：使用小驼峰命名法（camel Case），避免使用单字符变量名，除非在循环中或作为临时用途。例如，totalAmount、userAge。

常量（宏定义）：使用全大写字母，单词之间用下划线分隔。例如，MAX_BUFFER_SIZE、PI_CONSTANT。

类成员变量：通常使用 m_ 作为前缀，表示该变量是类的成员变量。例如，m_name、m_age。

函数参数：参数命名使用小驼峰命名法，避免使用简短的单字母命名，除非在局部的上下文中。例如，int totalAmount 而不是 int a。

2. 缩进

使用 4 个空格进行缩进，而不是 Tab 字符。Tab 字符的宽度在不同的编辑器中可能不同，使用空格更能保证一致性。

1.2.7 编译运行

编写完源程序后，将源程序代码文件存储为扩展名为 .cpp 的文件，代码文件须经过编译环境编译、链接后才能运行，输出结果。

编译的作用是将源程序转换成扩展名为 .obj 的二进制目标文件。编译的过程可以帮助用户发现在代码编辑过程中出现的语法问题，目前的 C++ 编译器均拥有完善、友好的语法提示方式，可以帮助用户快速修改语法问题。当所有语法问题修改完毕后，编译系统才可以进入下一步链接过程。

链接的作用是将编译得到的目标文件（.obj 文件）和必要的库（静态库、动态库）文件合并成一个可执行程序。编译阶段将源代码翻译成目标代码（机器代码），并生成目标文件。目标文件中包含了符号（如函数名、变量名等）的引用。这些符号可能在当前文件中被定义，也可能在其他文件或函数库中被定义。所以，链接的一个重要作用就是进行符号解析，链接过程就是查找所有未定义的符号，将所有函数调用、变量引用等符号正确地链接到它们所在的目标文件或库文件中的定义。链接完成后，将产生一个完整的可执行文件（.exe 文件），形成一个可以直接运行的程序。

例 1-1 的示例程序运行后，在计算机屏幕上的输出如下。

```
请输入a和b的数值:
12 13
a和b的乘积是: 156
--------------------------------
Process exited after 4.77 seconds with return value 0
请按任意键继续. . .
```

程序执行时，首先会先输出一行字符“请输入 a 和 b 的数值：”，等候用户输入需要运算的数据，输入数据后按 Enter 键确认，程序运行输出计算结果“a 和 b 的乘积是：156”，并给出程序运行的时间和“请按任意键继续 . . .”的提示，提醒用户在查看完运行结果后，按任意键程序将退出并返回到编程环境，继续编辑代码。

在运行阶段，可以通过输入一些特定的验证数据，来检验程序运行的结果是否正确。如果有错误出现，可以通过编译环境的程序跟踪功能来检查程序运行过程是否符合算法的设计，也可以通过观察程序运行中某些变量的值的变化情况来推断出错的原因，为后续的代码调试提供参考。

1.3 算法的基本概念

视频
算法的基本概念

算法是解决特定问题的一系列明确的、有限的指令。它是对特定问题求解步骤的一种描述，是指令的有限序列，其中每一条指令表示一个或多个操作。

1.3.1 算法的特性

一个有效的算法必须满足以下基本特征。

1. 有穷性

算法必须在执行有限个步骤后终止，且每个步骤都能在有限时间内完成。如果算法无限循环，将无法给出结果，失去实用价值。有穷性（finiteness）确保算法能在可接受的时间内解决问题。

2. 确定性

算法的每个步骤必须有明确、无二义的定义，无论谁执行都能得到相同的结果。只有拥有确定性（definiteness），才能确保算法可被正确实现和复现。算法中任何模糊的步骤会导致不同人实现的算法行为不一致。

3. 可行性

算法的每个步骤都必须是可行的，即能用基本运算（如加、减、乘、除等运算）在有限时间内完成。若步骤无法实现（如计算 π 的精确值），即算法没有可行性（effectiveness），则算法没有任何意义，同时也要确保算法在现有计算能力下可执行。

4. 输入

一个算法可以有零个或多个输入，用于提供问题的初始数据。输入使算法能够处理不同的问题实例（如排序不同数组），零输入的算法（如生成固定序列）也是合法的。

5. 输出

算法必须有一个或多个输出，用于反映问题的解或处理结果。如果算法运行后无输出结果，用户针对算法将无法验证其是否解决了问题，也可以说输出是算法价值的直接体现。

1.3.2 算法的评估

分析一个算法的优劣，主要是分析算法的效率（时间复杂度和空间复杂度），以便在不同的算法中选择最优的解决方案。分析时须着重从时间复杂度、空间复杂度、正确性、健壮性、可读性等方面进行评估。

1. 时间复杂度

时间复杂度是衡量算法执行效率的指标，它描述了算法执行时间随输入规模增长的变化趋势。时间复杂度关注的不是具体的执行时间（受硬件、程序语言影响），而是时间增长的量级，使用 O 表示。其核心思想是当输入规模 n 足够大时，算法的执行时间主要由最高阶项决定，低阶项和常数项可忽略。例如，$T(n) = 3n^2 + 5n + 2 \rightarrow$ 时间复杂度为 $O(n^2)$。常见时间复杂度及其应用如表 1-1 所示。

表 1-1　常见时间复杂度及其应用

复杂度	名称	含义	示例算法
$O(1)$	常数时间	执行时间与输入规模无关	数组元素访问、哈希表查询
$O(\log n)$	对数时间	每次操作将问题规模缩小一半	二分查找、平衡树操作
$O(n)$	线性时间	执行时间与输入规模成正比	数组遍历、单链表查找
$O(n \log n)$	线性对数	分解为规模为 n 的问题，递归求解并合并结果	快速排序、归并排序、堆排序
$O(n^2)$	平方时间	嵌套循环，每次处理 n 个元素	冒泡排序、选择排序、插入排序
$O(n^3)$	立方时间	三层嵌套循环	矩阵乘法、Floyd-Warshall 算法
$O(2^n)$	指数时间	问题规模每增加 1，计算量翻倍	子集枚举、递归斐波那契数列
$O(n!)$	阶乘时间	问题规模每增加 1，计算量乘以 $n+1$	旅行商问题（暴力枚举）

2. 空间复杂度

空间复杂度是衡量算法执行过程中临时占用存储空间大小的指标，它描述了存储空间随输入规模增长的变化趋势。空间复杂度主要关注除输入数据外，算法所需的额外空间（如临时变量、递归栈、辅助数组等），同样使用 O 表示，如 $O(1)$、$O(n)$ 等。其核心思想是当输入规模 n 足够大时，算

法的空间需求主要由最高阶项决定，低阶项和常数项可忽略。常见空间复杂度及其应用如表 1-2 所示。

表 1-2　常见空间复杂度及其应用

复杂度	名称	含义	示例算法
$O(1)$	常数空间	只使用固定的额外变量，与输入规模无关	冒泡排序、原地反转链表
$O(\log n)$	对数空间	递归调用栈的深度为对数级别	二分查找的递归实现、平衡树操作
$O(n)$	线性空间	需要与输入规模成正比的额外空间	复制数组、递归斐波那契数列（未优化）
$O(n^2)$	平方空间	需要二维数组或嵌套结构的空间	邻接矩阵表示图、动态规划的二维数组
$O(2^n)$	指数空间	需要存储所有可能的子集或组合	生成所有子集、回溯法求解

3. 正确性

算法必须能够正确地解决问题，这是算法的最基本要求。可以通过归纳法等严格的数学推导证明算法的正确性，也可以使用边界值（如空数组、单元素数组）等测试用例来验证算法的正确性。

4. 健壮性

算法在输入数据非法的情况下，也能适当地做出反应或进行处理，而不会产生莫名其妙的输出结果。可以使用检查数组是否为空、除数是否为零等特定输入来验证当程序出现错误时，是否会返回错误代码或抛出异常（throwing an exception）。

5. 可读性

优秀的算法应该易于理解和实现，以便后续的用户进行维护和修改，通常可读性高的代码更易发现和修复 bug。

1.3.3　算法的描述

设计算法时，需要用一种表达方式来描述一些问题的求解步骤，这就是算法描述。通过算法描述来了解算法设计的思路，常见的描述方法有自然语言描述、流程图描述、伪代码描述、程序代码描述。

1. 自然语言描述

自然语言描述就是使用日常语言（如中文、英文）描述算法步骤，适合向非专业人士讲解算法的初步构思或算法原理。但是自然语言的模糊性可能会导致不同人有不同的理解，当描述复杂算法时可能过于烦琐，且无法直接测试其正确性。

例如，描述计算两数之和。

（1）输入两个整数 a 和 b。

（2）计算它们的和 c = a + b。

（3）输出结果 c。

2. 流程图描述

流程图描述就是使用标准化图形符号表示算法的步骤和流程。这种方法的图形符号定义明确，误解少，可视化展示算法逻辑，清晰展示控制结构，便于理解整体流程。但是，规模较大、复杂的算法流程图过于庞大，修改难度高，不便于表达复杂的数据结构或数学公式。

例如，计算 1 到 100 的和，如图 1-1 所示。

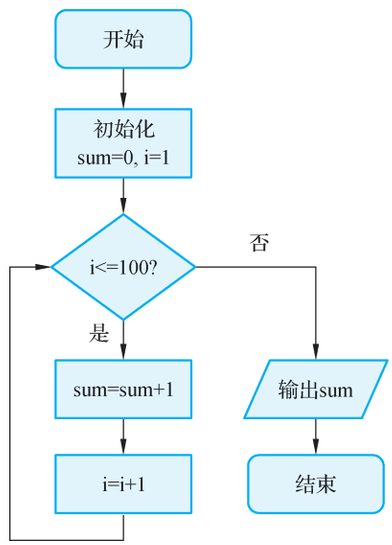

图 1-1　流程图描述算法过程

在图 1-1 中，用矩形表示处理步骤（如计算、赋值），用菱形表示判断条件（如是否小于等于 100），用圆角矩形表示算法的开始或结束，用平行四边形表示算法的输入或输出，用箭头表示算法的流程方向。

3. 伪代码描述

伪代码描述是介于自然语言和编程语言之间的描述方法，使用简化的语法结构，不依赖特定语言，使用通用的编程概念（如循环、条件），强调算法逻辑，忽略语法细节，聚焦算法核心思想，比自然语言更紧凑，比代码更抽象，不受编程语言限制，便于交流。

例如，计算 1 到 100 的和，伪代码描述如下。

```
算法: sum 1 To 100()
输出: 1到100的整数和

sum ← 0
for i ← 1 to 100 do
```

```
        sum ← sum + i
    end for

    return sum
```

4. 程序代码描述

程序代码描述就是使用具体的编程语言（如 C++、Python）实现算法。代码描述严格遵循语法规则，可直接运行代码验证算法的正确性，也可以利用语言特性优化性能。

育德树人

技术优势与国家科技竞争力

当下国际科技竞争激烈，编程语言的性能与特点对国家科技竞争力有着重要影响。C++ 具有高效、灵活等特点，在操作系统、游戏开发等众多关键领域发挥着不可替代的作用。我国在科技发展进程中，掌握 C++ 这样的核心技术至关重要。同学们要认识到，学好 C++ 不仅能提升个人编程能力，更能为增强国家科技竞争力添砖加瓦，提升使命感和责任感。

1.4 简单程序实例

下面通过一个简单的程序实例（包含用户交互、数据处理和结果输出），对本模块内容进行巩固。示例代码如下：

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    //程序欢迎信息
    cout << "=====================================" << endl;
    cout << "        C++基础程序示例        " << endl;
    cout << "=====================================" << endl;

    //1. 变量声明与数据输入
    string name;
    int age;
    double height;
    cout << "请输入您的姓名：";
```

```
getline(cin, name);  //读取包含空格的完整姓名
cout << "请输入您的年龄: ";
cin >> age;
cout << "请输入您的身高(米): ";
cin >> height;

//2. 数据处理与条件判断
bool isAdult = age >= 18;
double idealWeight = (height - 1.0) * 100;  //简单理想体重计算公式

//3. 结果输出
cout << "\n===== 个人信息汇总 =====" << endl;
cout << "姓名: " << name << endl;
cout << "年龄: " << age << " 岁" << endl;
cout << "身高: " << height << " 米" << endl;
cout << "是否成年: " << (isAdult ? "是" : "否") << endl;
cout << "理想体重: " << idealWeight << " 公斤" << endl;

//4. 循环与分支应用
int choice;
cout << "\n请选择要执行的操作: " << endl;
cout << "1. 查看年龄分类" << endl;
cout << "2. 计算BMI指数" << endl;
cout << "3. 退出程序" << endl;
cout << "请输入选项(1-3): ";
cin >> choice;

switch (choice) {
    case 1:
        cout << "\n年龄分类结果: " << endl;
        if (age < 12) cout << "儿童" << endl;
        else if (age < 18) cout << "青少年" << endl;
        else if (age < 60) cout << "成年人" << endl;
        else cout << "老年人" << endl;
        break;
```

```
        case 2: {
            double weight;
            cout << "请输入您的实际体重(公斤): ";
            cin >> weight;
            double bmi = weight / (height * height);
            cout << "\n您的BMI指数: " << bmi << endl;
            cout << "健康状况: ";
            if (bmi < 18.5) cout << "偏瘦" << endl;
            else if (bmi < 24) cout << "正常" << endl;
            else if (bmi < 28) cout << "过重" << endl;
            else cout << "肥胖" << endl;
            break;
            }
        case 3:
            cout << "感谢使用本程序，再见！" << endl;
            break;
        default:
            cout << "无效选项，程序退出。" << endl;
    }

    return 0;
}
```

育德树人

数字化转型下的严谨态度

如今各行业都在进行数字化转型，程序的质量至关重要。从“hello world”程序的编辑、编译、链接到运行，每一步都需要严谨对待。以金融行业数字化转型为例，一个小的代码错误可能导致巨额资金损失。同学们要明白，在数字化时代，严谨的编程态度是对自己和社会负责，是推动行业发展的基础。

1.5 模块小结

在本模块中，我们踏入了 C++ 程序设计的大门，对这门强大的编程语言有了初步的认识。C++ 作为一种多范式编程语言，融合了过程式、面向对象和泛型编程风格。C++ 在 C 语言基础上进行了扩展，既保持了对 C 语言的兼容性，又引入了许多新特性，使其在灵活性和表达能力等

方面都有显著提升，被广泛应用于系统软件、游戏、嵌入式系统、大型应用程序开发等众多领域。

通过本模块的学习，我们对 C++ 语言的整体框架和开发流程有了初步理解，已具备编写简单 C++ 程序的基本能力。后续将深入探讨数据类型、控制结构、函数、类与对象等更丰富的内容，逐步掌握 C++ 程序设计的精髓。

1.6 习　　题

一、选择题

1. C++ 是由计算机科学家（　　）开发的。

A. Bjarne Stroustrup　　B. James Gosling

C. Dennis Ritchie　　D. Donald Knuth

2. C++ 源文件的扩展名通常是（　　）。

A. .c　　B. .cpp

C. .java　　D. .py

3. 以下（　　）是 C++ 的注释符号。

A. //　　B. /* */

C. #　　D. 以上都是

4. C++ 程序的执行入口是（　　）。

A. main() 函数　　B. start() 函数

C. begin() 函数　　D. 任意函数

5. 若要在控制台输出“Hello, World!”，应使用（　　）。

A. printf("Hello, World!");　　B. cout << "Hello, World!";

C. System.out.println("Hello, World!");　　D. print("Hello, World!");

6. 以下关于 C++ 的说法错误的是（　　）。

A. C++ 是面向对象的编程语言　　B. C++ 可以直接操作硬件

C. C++ 不支持泛型编程　　D. C++ 兼容 C 语言

7. 编译 C++ 程序的过程不包括（　　）。

A. 预处理　　B. 编译

C. 链接　　D. 解释

8. 以下（　　）是 C++ 的标准输入流对象。

A. cout　　B. cin

C. cerr　　D. clog

9. 在 C++ 程序中，“using namespace std;”的作用是（　　）。

A. 引入标准库函数　　B. 引入标准命名空间

C. 定义新的命名空间　　D. 提高程序性能

10. 若要编译并运行 C++ 程序，通常需要使用（　　）。

A. 文本编辑器　　B. 集成开发环境（IDE）

C. 调试器　　D. 以上都是

11. 以下（　　）是 C++ 的标准输出流对象。

A. cin　　B. cout

C. cerr　　D. clog

12. 在 C++ 程序中，int main() 函数的返回值类型是（　　）。

A. void　　B. int

C. char　　D. 任意类型

13. 若要在 C++ 中使用字符串处理功能，须包含的头文件是（　　）。

A. <stdio.h>　　B. <string.h>

C. <string>　　D. <cstring>

14. 以下关于 C++ 注释的说法正确的是（　　）。

A. 单行注释以 /* 开始，以 */ 结束　　B. 多行注释以 // 开始

C. 注释会被编译器忽略　　D. 注释不能出现在代码中间

15. C++ 源文件经过编译后会生成（　　）。

A. .exe 文件　　B. .obj 文件（Windows）或 .o 文件（Linux）

C. .class 文件　　D. .pyc 文件

16. 以下（　　）是 C++ 的标准错误流对象。

A. cout　　B. cin

C. cerr　　D. clog

17. 用 C++ 语言编写的程序代码（　　）。

A. 可以直接执行　　B. 经过编译解释才能执行

C. 经过编辑可执行　　D. 以上说法都正确

18. C++ 程序的基本组成单位是（　　）。

A. 子程序　　B. 函数

C. 语句　　D. 行

19. 系统编译程序时，中间文件的扩展名是（　　）。

A. .cpp　　B. .obj

C. .exe　　D. .txt

20. 在 C++ 源程序中，函数体应该置于（　　）符号之内。

A. (　)　　B. < >

C. { }　　D. # #

二、简答题

1. 简述 C++ 语言的主要特点。
2. 什么是编译型语言？ C++ 属于编译型语言吗？
3. 编写一个完整的 C++ 程序，输出“Welcome to C++ Programming!”。
4. 什么是命名空间？为什么 C++ 引入命名空间的概念？
5. 简述 C++ 中 #include 预处理指令的作用。
6. 比较 C++ 和 C 语言的主要区别。
7. 什么是集成开发环境？列举几个常见的 C++ IDE。
8. 解释 C++ 程序的编译和链接过程。
9. 简述 C++ 标准库的作用和组成部分。

三、程序设计题

1. 输出你的姓名和学号。
2. 计算并输出两个整数的和。
3. 输入圆的半径，计算并输出圆的面积。
4. 输入两个整数，交换它们的值后输出。
5. 输入三个整数，输出其中的最小值。
6. 输入一个字符，判断其是否为大写字母。
7. 计算并输出 1 到 100 的整数之和。
8. 输入一个字符串，统计其中字符的个数。