第 5 章　程序设计基础

本章学习目标

- 掌握程序设计的相关概念。
- 掌握程序设计语言的相关知识。
- 了解算法的相关内容及常用算法。
- 了解软件工程的相关内容。

本章首先介绍程序设计的相关知识，包括程序设计方法及程序设计语言；然后介绍算法和软件工程的相关内容，包括算法的概念、算法的表示方法、常用算法、软件危机和软件生存周期等内容。


程序设计概述

5.1　程序设计概述

程序设计是指编写程序的过程。程序设计是一门技术，需要相应的理论、技术、方法和工具来支持。程序设计不仅要保证设计的程序能正确地解决问题，还要求程序具有可读性、可维护性。

5.1.1　程序设计的基本过程

程序设计的基本过程包括分析问题、设计算法、编写程序、运行程序、分析结果和编写文档等不同阶段。

1. 分析问题

对于接受的任务要进行认真的分析，研究所给定的条件，分析最后应达到的目标，找出解决问题的规律，选择解题的方法，完成实际问题。

2. 设计算法

根据对接受任务的分析设计出解决的方法和具体步骤。

3. 编写程序

将算法翻译成计算机程序设计语言，对源程序进行编辑、编译和连接。

4. 运行程序、分析结果

运行可执行程序，得到运行结果。能得到运行结果并不意味着程序正确，要对结果进

行分析，看它是否合理，如果不合理，要对程序进行调试。

5. 编写文档

许多程序是提供给别人使用的，如同正式的产品应当提供产品说明书一样，正式提供给用户使用的程序必须向用户提供程序说明书。其内容应包括程序名称、程序功能、运行环境、程序的装入和启动、需要输入的数据以及使用注意事项等。

5.1.2 程序设计的方法

常用的程序设计方法有结构化程序设计和面向对象程序设计。

1.结构化程序设计

随着“软件危机”的出现，人们开始研究程序设计方法，其中最受关注的是结构化程序设计方法。20 世纪 70 年代，人们提出了“结构化程序设计”的思想和方法。结构化程序设计方法引入了工程思想和结构化思想，使大型软件的开发和编程都得到了极大改善。

1）结构化程序设计方法

结构化程序设计的主要观点是采用自顶向下、逐步求精、模块化、限制使用 goto 语句的程序设计方法。

（1）自顶向下。

将复杂的大问题分解为相对简单的小问题，找出每个问题的关键、重点所在，然后用精确的思维定性、定量地描述问题。其核心本质是“分解”。

（2）逐步求精。

程序设计时，应先考虑总体，后考虑细节；先考虑全局目标，后考虑局部目标。设置完全局的内容后再对局部的问题进行逐步细化和具体化。

（3）模块化。

一个复杂问题肯定是由若干稍简单的问题构成的。模块化是把程序要解决的总目标分解为子目标，再进一步分解为具体的小目标，把每一个小目标称为一个模块。

（4）限制使用 goto 语句。

goto 语句是程序设计语言中的一个无条件转移语句，一般用在模块中改变程序执行的顺序。在程序中过多地使用 goto 语句，会使程序变得难以理解。从提高程序清晰度考虑，一般建议不使用 goto 语句。

2）结构化程序设计的基本结构

结构化程序设计的 3 种基本结构是顺序结构、选择结构和循环结构。

（1）顺序结构。

顺序结构是最基本、最常用的结构，如图 5.1 所示。顺序结构就是按照程序语句的书写顺序一条一条地执行。

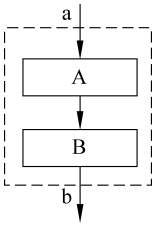
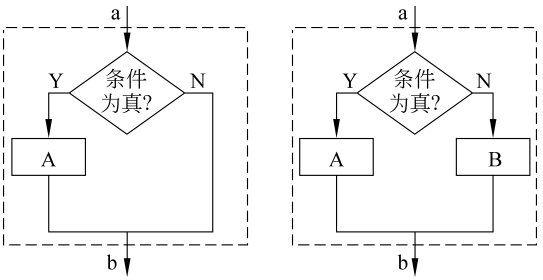


图 5.1 顺序结构

（2）选择结构。

选择结构又称为分支结构,包括单分支结构、双分支结构和多分支结构。选择结构是根据设定的条件,判断应该执行哪一条语句。单分支结构和双分支结构如图 5.2 所示。

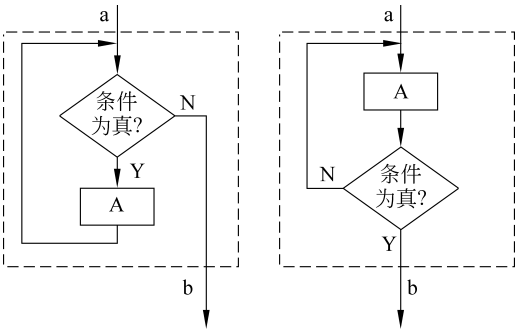


(a) 单分支结构　　(b) 双分支结构

图 5.2　单分支结构和双分支结构

(3) 循环结构。

循环结构又称为重复结构,是根据给定的条件,判断是否需要重复执行相同的程序段。利用循环结构可以简化大量的代码。循环结构分为当型循环结构和直到型循环结构,如图 5.3 所示。



(a) 当型循环结构　　(b) 直到型循环结构

图 5.3　循环结构

通过图 5.1～图 5.3 可知,结构化程序设计的每种结构只有一个入口和一个出口,这是结构化程序设计的一个原则。根据结构化程序设计方法设计出的程序易于阅读、理解和维护,同时还提高了编程的工作效率,降低了软件开发成本。

2. 面向对象程序设计

面向对象方法已经发展成为主流的软件开发方法。面向对象的本质就是主张从客观世界固有的事物出发来构造系统,提倡用人类在现实生活中常用的思维方法来认识、理解和描述客观事物,系统中的对象以及对象之间的关系能够如实地反映固有事物及其关系。

1) 面向对象方法的基本概念

(1) 对象。

对象是面向对象方法中最基本的概念。对象可以用来表示客观世界中的任何实体。也就是说,应用领域中任何有意义的、与所要解决的问题有关系的事物都可以作为对象。

它既可以是具体的物理实体的抽象，也可以是人为的概念，或者是任何有明确边界和意义的东西。例如，一个学生、一张桌子、一笔贷款等都可以作为一个对象。

面向对象程序设计方法中涉及的对象是系统中用来描述客观事物的一个实体，是构成系统的一个基本单位，由一组表示其静态特征的属性和它可执行的一组操作组成。例如，一辆汽车是一个对象，它包含了汽车的属性（如颜色、型号、载重量等）及其操作（如发动、转弯、刹车等）。

对象具有的特点是标识唯一性、分类性、多态性、封装性、模块独立性好等。

（2）类和实例。

将属性、操作相似的对象归为类。也就是说，类是具有共同属性、共同方法的对象的集合。类是对象的抽象，它描述了该对象类型的所有对象的性质，而一个对象则是其对应类的一个实例。注意，当使用“对象”这个术语时，既可以指一个具体的对象，也可以泛指一般的对象；但是当使用“实例”这个术语时，对象指的是一个具体的对象。

例如，人类是一个类，它描述了所有人的性质，因此任何人都是人类的对象，而一个具体的人，如张三，就是人类的一个实例。

（3）消息。

面向对象的世界是通过对象与对象间彼此的相互合作来推动的，对象间的这种相互合作需要一个机制来协助完成，这种机制称为“消息”。消息是一个实例与另一个实例之间传递的信息，它是请求对象执行某一处理或回答某一要求的消息。

例如，一个汽车对象具有“行驶”这项操作，那么让汽车以 60km 的速度行驶时，需要传递给汽车“行驶”及 60km 的消息。

（4）继承。

继承是面向对象方法的一个主要特征。继承是使用已有的类定义作为基础建立新类的定义技术。已有的类可被当作基类来引用，新类可被当作派生类来引用。从广义上说，继承是指能够直接获得已有的性质和特征，而不必重复定义它们的一种技术。图 5.4 给出了实现继承机制的原理。

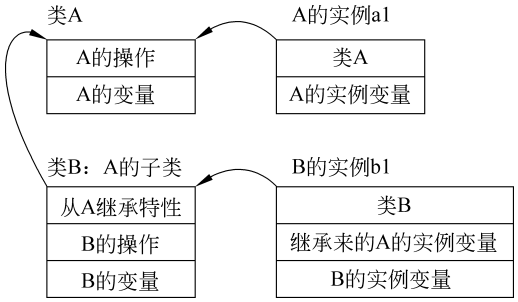


图 5.4 实现继承机制的原理

继承具有传递性，若类 C 继承类 B，类 B 继承类 A，则类 C 继承类 A。因此，一个类实际上继承了它上层的全部基类的特性。也就是说，属于某类的对象除了具有该类所定义的特性外，还具有该类上层全部基类定义的特性。

(5) 封装性。

封装指的是类中定义的数据只能被类中定义的方法访问,除此之外别无他法。封装的结果使对象以外的部分不能随意存取对象的内部属性,从而有效地避免了外部错误对它的影响,大大减小了查错和排错的难度。另外,当对对象内部进行修改时,因为它只通过少量的外部接口对外提供服务,所以减小了内部的修改对外部的影响。封装机制将对象的使用者与设计者分开,使用者不必知道对象行为实现的细节,只需用设计者提供的外部接口让对象去做。封装的结果实际上隐蔽了复杂性,并提供了代码重用性,从而降低了软件开发的难度。

(6) 多态性。

对象根据所接收的消息而做出动作,同样的消息被不同的对象接收时,可导致完全不同的行为,该现象称为多态性。多态性是指对象可以像父类那样使用,同样的消息既可以发送给父类对象,也可以发送给子类对象。多态性机制不仅增加了面向对象软件系统的灵活性,还进一步减少了信息冗余,显著地提高了软件的可重用性和扩充性。

2) 面向对象方法的优点

面向对象方法日益受到人们的重视和应用,成为主流的软件开发方法,主要源于面向对象方法的以下优点。

(1) 与人类习惯的思维方法一致。

用计算机解决的问题都是现实世界中的问题,这些问题都由一些相互之间存在一定联系的事物组成,每个具体的事物都具有行为和属性两方面的特征。因此,把描述事物静态属性的数据结构和表示事物动态行为的操作放在一起构成一个整体,才能完整、自然地表示客观世界中的实体。

(2) 稳定性好。

面向对象方法基于对象模型,以对象为中心构造软件系统。它的基本做法是用对象模型模拟现实生活中的实体,以对象间的联系刻画实体间的联系。因为现实世界中的实体是相对稳定的,所以以对象为中心构造的软件系统也是比较稳定的。

(3) 可重用性好。

软件重用是指在不同的软件开发过程中重复使用相同或相似软件的过程。重用是提高软件生产效率的主要方法。

面向对象的软件开发技术在利用可重用的软件成分构造新的系统软件时有很大的灵活性。继承性机制使得子类不仅可以重用其父类的数据结构和程序代码,而且还可以在父类代码的基础上方便地修改和扩充,而这种修改并不影响对原有类的使用。

(4) 易于开发大型软件产品。

用面向对象方法开发软件时,可以把一个大型产品看作一系列本质上相互独立的小产品来处理,这样可以降低开发的技术难度,还使得对开发工作的管理变得容易。

(5) 可维护性好。

用面向对象方法开发的软件可维护性好,这主要是因为:用面向对象方法开发的软件稳定性比较好,比较容易修改,比较容易理解,而且易于测试和调试。

5.1.3 程序设计语言

程序设计语言就是用于书写计算机程序的一组记号和一组规则。程序设计人员把计划让计算机完成的工作用这些记号编排好程序，再交给计算机去执行。

1. 程序设计语言的分类

对程序设计语言可以从不同的角度进行分类，其中最常见的分类方法是根据程序设计语言与计算机硬件的联系程度将其分为3类，即机器语言、汇编语言和高级语言。前两类依赖于计算机硬件，被统称为低级语言；而高级语言与计算机硬件依赖关系较小。

1）机器语言

机器语言是计算机硬件系统能够直接识别的计算机语言，不需要翻译。机器语言中的每一条语句实际上是一条二进制数形式的指令代码，由操作码和操作数组成。操作码指出应该进行什么样的操作，操作数指出参与操作的数据本身或它在内存中的地址。对于不同的计算机硬件，其机器语言是不同的。因此，针对某一种计算机所编写的机器语言，其程序多数不能在另一种计算机上运行。

例如，计算累加器A＝8＋10的机器语言程序及注释如表5.1所示。

表5.1 计算A＝8＋10的机器语言程序及注释

机器语言程序	注　　释
10110000 00001000	把8存放到累加器A中
00101100 00001010	将10与累加器中的8相加，结果存在A中
11110100	程序结束

由于机器语言程序是直接针对计算机硬件所编写的，因此它的执行效率比较高，能充分发挥计算机的速度性能。但是，用机器语言编写程序，工作量大，难于记忆，容易出错，调试修改麻烦，而且程序的直观性差，不容易移植。

2）汇编语言

为了克服机器语言的缺点，汇编语言用助记符来代替机器指令的操作码，用地址符号代替操作数。由于这种符号化的做法，所以汇编语言也被称为符号语言。汇编语言要比机器语言直观，容易理解和记忆。例如，ADD表示加，SUB表示减，JMP表示跳转，MOV表示数据的传送指令等。

例如，8＋10加法题的汇编语言程序为

```
MOV  AX, 8H
MOV  BX, 0AH
ADD  BX, AX
```

用汇编语言编写的程序称为汇编语言源程序。由于计算机能够直接识别的只有机器语言，因此汇编语言的源程序是不能在计算机上直接运行的，而是需要用汇编程序把它翻

译成机器语言程序后方可运行。

用汇编语言编写的程序比机器语言编写的程序易读、易检查、易修改,同时保持了机器语言执行速度快、占用存储空间少的优点。但是,汇编语言也是面向机器的语言,不具备通用性和可移植性。

3) 高级语言

高级语言是一类面向问题或面向对象的语言,它并不面向机器,不依赖于具体的计算机。在不同的平台上高级语言会被编译成不同的机器语言,而不是直接被计算机执行。

由于高级语言采用自然词汇,并使用与自然语言语法相近的语法体系,所以它的程序设计方法比较接近人们的思维习惯,编写出的程序更容易阅读和理解。

例如,计算 8+10,并把结果赋值给变量 c,在 C 语言中可将它表示成:

```
c=8+10;
```

高级语言的特点是易学、易用、易维护,人们可以更有效、更方便地利用它编写各种用途的计算机程序。高级语言独立于具体的计算机硬件,通用性和可移植性都比较好。

2. 常用的程序设计语言

常用的程序设计语言被分为两类:面向过程语言和面向对象语言。

1) 面向过程语言

面向过程语言使用传统的方法编程。它采用与计算机硬件程序相同的方法来执行程序。当程序员需要使用某一面向过程语言来解决问题时,他必须仔细设计算法,然后将算法翻译成指令。

常见的面向过程语言有 FORTRAN、Pascal、C 等,其中 C 语言是使用最广泛的高级语言之一。

(1) FORTRAN 语言。

FORTRAN 语言是世界上第一个被正式推广使用的高级语言。它是 1954 年被提出来的,1956 年开始正式使用,目前它仍然是数值计算领域所使用的主要语言。

FORTRAN 是 FORmula TRANslation 的缩写,意为“公式翻译”。它是为科学、工程问题或企事业管理中的那些能够用数学公式表达的问题而设计的,其数值计算的功能较强。

(2) Pascal 语言。

Pascal 语言是一种计算机通用的高级程序设计语言。它由瑞士 Niklaus Wirth 教授于 20 世纪 60 年代末设计并创立。Pascal 源于人名,是为了纪念 17 世纪法国著名哲学家和数学家 Blaise Pascal。Pascal 语言现已成为使用最广泛的基于 DOS 的语言之一,主要特点有:严格的结构化形式,丰富完备的数据类型,运行效率高,查错能力强。Pascal 语言还是一种自编译语言,这就使其大大提高了可靠性。Pascal 是结构化编程语言,具有简洁的语法和结构化的程序结构。因此,以前许多学校开设了 Pascal 这门计算机语言课。

Pascal 语言是最早出现的结构化编程语言,具有丰富的数据类型和简洁灵活的操作语句,适于描述数值和非数值的问题。

(3) C 语言。

C 语言是一种面向过程的计算机程序设计语言,它是目前众多计算机语言中举世公认的优秀的结构程序设计语言之一。它由美国贝尔实验室的 Dennis M. Ritchie 于 1972 年推出,1978 年后 C 语言已先后被移植到大、中、小及微型计算机上。C 语言可以作为工作系统设计语言,编写系统应用程序;也可以作为应用程序设计语言,编写不依赖计算机硬件的应用程序。C 语言具有很强的数据处理能力,应用范围广泛,不仅在软件开发方面,而且各类科研都需要用到 C 语言,适于编写系统软件以及二维、三维图形和动画制作软件等。

C 语言的优点:语法简洁紧凑,运算符和数据类型丰富,表达方式灵活实用,允许直接访问物理地址对硬件进行操作,生成目标代码质量高,程序执行效率高,可移植性好,表达力强。

2) 面向对象语言

一种计算机语言要称为面向对象语言必须支持几个面向对象的概念:类、对象、继承和多态。面向对象语言中,有一部分是新发明的语言,如 Smalltalk、Java 等。这些语言吸取了其他语言的精华,而又尽量剔除了它们的不足。因此,面向对象的特征明显,充满了生机。另外一些语言则是通过对现有的语言进行改造,增加了面向对象的特征演化而来的,如 Object Pascal、C++ 、Ada 95 等,其保留着对原有语言的兼容。

(1) C++ 语言。

C++ 语言是一种优秀的面向对象的程序设计语言。它是在 C 语言的基础上发展过来的,但是它比 C 语言更容易被人们学习和掌握。C++ 语言是由贝尔实验室的 Bjarne Stroustrup 设计和实现的,它兼具 Simula 语言和 C 语言的特性。C++ 最初的版本被称为带类的 C,在 1980 年被第一次投入使用,当时它只支持系统程序设计和数据抽象技术。支持面向对象程序设计的语言设施在 1983 年被加入 C++ 。在此之后,面向对象设计方法和面向对象程序设计技术逐渐进入了 C++ 领域。在 1985 年,C++ 第一次投入商业市场。1987—1989 年,支持泛型程序设计的语言设施也被加入了 C++ 。

C++ 语言的优点如下。

① C++ 直接、广泛地支持多种程序设计风格。

② C++ 给程序设计者更多的选择。

③ C++ 尽可能与 C 兼容,提供了一个从 C 到 C++ 的平滑过渡。

④ C++ 避免平台限定或没有普遍用途的特性。

⑤ C++ 不使用会带来额外开销的特性。

⑥ C++ 不需要复杂的程序设计环境。

(2) Java 语言。

Java 语言是一种简单的、跨平台的、面向对象的、分布式的、解释的、健壮的、安全的、结构的、中立的、可移植的、性能优异的、多线程的、动态的语言。Java 是由 Sun 公司于 1995 年 5 月推出的面向对象程序设计语言。

Java 语言的特征如下。

① Java 语言是简单的。Java 语言的语法与 C 语言和 C++ 语言很接近,使得大多数

程序员很容易学习和使用Java。另外,Java丢弃了C++中很少使用的、很难理解的那些特性。

② Java语言是面向对象的。Java语言提供类、接口和继承等原语,为了简单起见,只支持类之间的单继承,但支持接口之间的多继承,并支持类与接口之间的实现机制。

③ Java语言是分布式的。Java建立在扩展TCP/IP网络平台上。库函数提供了用HTTP和FTP传送和接收信息的方法,这使得程序员使用网络上的文件和使用本机文件一样容易。

④ Java语言是健壮的。Java的强类型机制、异常处理、垃圾的自动收集等是Java程序健壮性的重要保证。

⑤ Java语言是安全的。Java舍弃了C++的指针对存储器地址的直接操作,程序运行时,内存由操作系统分配,这样可以避免病毒通过指针侵入系统。Java对程序提供了安全管理器,防止程序的非法访问。

⑥ Java语言是体系结构中立的。Java程序在Java平台上被编译为体系结构中立的字节码格式,然后可以在实现这个Java平台的任何系统中运行。这种途径适合于异构的网络环境和软件的分发。

⑦ Java语言是可移植的。这种可移植性来源于体系结构中立性。另外,Java还严格规定了各个基本数据类型的长度。Java系统本身也具有很强的可移植性。Java编译器是用Java实现的,而Java的运行环境是用ANSI C实现的。

⑧ Java语言是多线程的。Java语言支持多个线程的同时执行,并提供多线程之间的同步机制。

3. 程序设计语言处理系统

用机器语言编写的程序可以直接在计算机上执行,而用其他程序设计语言编写的程序都不能直接在计算机上执行,需要对其进行适当的变换。语言处理系统的作用就是把用程序设计语言编写的程序变换成在计算机上可执行的程序。负责完成这些功能的软件是汇编程序、解释程序和编译程序,它们统称为程序设计语言处理系统。

1) 汇编程序

用汇编语言编写的程序称为汇编语言源程序,转换成机器语言的程序为目标程序。汇编语言源程序需要由一种“翻译”程序来将源程序转换为机器语言程序,这种翻译程序被称为汇编程序。

2) 编译程序

把用高级语言编写的源程序翻译成目标程序的过程称为编译。完成编译工作的软件称为编译程序。

源程序经过编译后,若无错误就会生成一个等价的目标程序,对目标程序进行连接、装配后,便得到执行程序,最后运行执行程序。执行程序全部由机器指令组成,运行时不依附于源程序,运行速度快。但这种方式不够灵活,每次修改源程序后,都必须重新编译和连接。目前使用的FORTRAN、C、Pascal等高级语言都采用这种方式。

3）解释程序

解释程序也是将高级语言转换为机器能够识别的语言。与编译程序不同的是，解释程序是边扫描源程序边进行翻译，然后执行，即解释一句，执行一句，不生成目标程序。这种方式运行速度慢，但执行中可以进行人机对话，随时改正源程序中的错误。以前流行的BASIC语言大都采用这种方式处理。编译程序与解释程序的区别如图5.5所示。

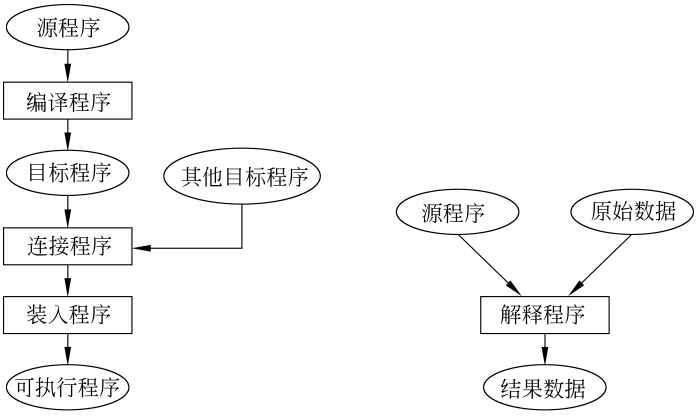


图 5.5 编译程序与解释程序的区别

5.2 算法概述


算法概述

计算机系统中的任何软件都是由大大小小的各种程序模块组成的，它们按照特定的算法来实现，算法的好坏直接决定了所实现软件性能的优劣。用什么样的方法来设计算法，所设计的算法需要什么样的资源，需要多少运行时间和多少存储资源等问题，都是在实现一个软件时所必须要解决的。计算机中的各类软件，包括操作系统、语言编译系统和数据库管理系统等，都必须用一个个的具体算法来实现。因此，算法设计是计算机科学的一个核心问题。

5.2.1 算法的概念

对于算法的概念，不同的专家有不同的定义方法，但这些定义的内涵基本是一致的。这些定义中最为著名的是计算机科学家 Donald E. Knuth 在其经典著作《计算机程序设计艺术(卷1)：基本算法》中对算法的定义和特性所做的有关描述：算法就是一个有穷规则的集合，其中的规则确定了一个解决某一特定类型问题的运算序列。此外，算法的规则序列需要满足以下5个重要特性。

(1) 有限性：算法在执行有限步之后必须终止。

(2) 确定性：算法的每一个步骤都有精确的定义，要执行的每一个动作都是清晰的、无歧义的。