第1章

程序设计基础

计算机是一种能高速运算，具有存储和记忆能力的、用程序控制的电子机械装置，是信息处理的基本工具。当操作者向计算机输入一定的信息（此信息必须是计算机能接受的），计算机才能按照操作者的要求处理信息（如何处理信息由操作者事先以程序形式确定），然后输出所需要的结果。计算机的工作机制如下：

（1）事先编制程序，将要求计算机处理的任务编制成一条条的指令，输入计算机中（存放于计算机的内存储器），这个过程称为编程序。

（2）计算机工作时，从内存中逐条取出指令，然后执行指令，这个过程称为运行程序，在程序的运行过程中，将要求用户提供必要的数据信息，程序处理的结果一般是输出有关数据信息。

本章将介绍有关结构化程序设计方法、模块化设计、面向对象技术的基本思想。

1.1 程序设计概述

1.1.1 程序设计方法的发展

程序设计初期，由于计算机硬件条件的限制，运算速度与存储空间都迫使程序员追求高效率，编写程序成为一种技巧与艺术，而程序的可理解性、可扩充性等因素被放到第二位。

随着计算机硬件与通信技术的发展，计算机应用领域越来越广泛，应用规模也越来越大，程序设计不再是一两个程序员可以完成的任务。在这种情况下，编写程序不再片面追求高效率，而是综合考虑程序的可靠性、可扩充性、可重用性和可理解性等因素。

正是这种需求促进了程序设计方法与程序设计语言的发展。

1. 早期程序设计

早期出现的高级程序设计语言有 FORTRAN、COBOL、ALGOL、BASIC 等语言。

这一时期，由于追求程序的高效率，程序员过分依赖技巧与天分，不太注重所编写程序的结构，这时期可以说是无固定程序设计方法的时期。

存在的一个典型问题是程序中的控制随意跳转，即不加限制地使用 goto 语句，这样的程序对别人来说是难以理解的，程序员自己也难以修改程序。

2. 结构化程序设计

随着程序规模与复杂性的不断增长，人们也在不断探索新的程序设计方法。证明了只用三种基本的控制结构（顺序、选择、循环）即可实现任何单入口-单出口的程序设计。

Dijkstra 建议从一切高级语言中取消 goto 语句，Mills 提出程序应该只有一个入口和一个出口。这些思想导致了结构化程序设计方法的诞生。

而 Pascal 语言则是由 Niklaus Wirth 根据结构化程序设计方法开发出来的语言。其特点是提炼出程序设计共同的特征并能将这些特征编译成高效的代码，因而成为结构化程序设计的有力工具。C 语言也是一种广为流行的结构化程序设计语言，它具有灵活方便、目标代码效率高、可移植性好等优点。

3. 面向对象程序设计

面向对象的方法是一种把面向对象的思想运用于软件开发过程中，指导开发活动的系统方法，简称 OO 方法，是建立在“对象”概念（对象、类和继承）基础上的方法学。对象是由数据和允许的操作组成的封装体，与客观实体有直接的对应关系。

面向对象的方法起源于面向对象的编程语言。20 世纪 60 年代后期就出现了类和对象的概念，类作为语言机制用来封装数据和相关操作。70 年代前期，Smalltalk 语言，奠定了面向对象程序设计的基础，1980 年 Smalltalk-80 标志着面向对象的程序设计已进入实用阶段。进入 80 年代相继出现了一系列面向对象的编程语言，如：C++、Java 等。自 80 年代中期到 90 年代，面向对象的研究重点已经从语言转移到设计方法学方面，尽管还不成熟，但已陆续提出了一些面向对象的开发方法和设计技术。

4. 程序设计方法学的研究

20 世纪 60 年代中期以后，计算机硬件技术日益进步，计算机的存储容量、运算速度和可靠性明显提高，生产硬件的成本不断降低。计算机价格的下跌为它的广泛应用创造了极好的条件。在这种形势下，迫切要求计算机软件也能与之相适应。因而，一些开发大型软件系统的要求提了出来。然而软件技术的进步一直未能满足形势发展的需要，在大型软件的开发过程中出现了三大难题：

- 复杂程度高。
- 研制周期长。
- 正确性难以保证。

由于遇到的问题找不到解决办法，使问题堆积起来，形成了人们难以控制的局面，出现了所谓的“软件危机”。

最为典型的例子是美国 IBM 公司于 1961—1964 年开发的 IBM 360 系列机的操作系统。该操作系统花了大约 5000 人/年的工作量，最多时，有 2000 人投入开发工作，写出近 100 万行的源程序。尽管投入了这么多的人力和物力，得到的结果却极其糟糕。据统计，这个操作系统每次发行的新版本都是从前一版本中找出 1000 个程序错误而修正的结果。可想而知，这样的软件质量糟到了什么地步。

IBM 360 操作系统的历史教训，已成为软件开发项目中的典型事例被记入历史史册。

如果开发的软件隐含错误，可靠性得不到保证，那么在运行过程中很可能对整个系统造成十分严重的后果。例如：1963年美国用于控制火星探测器的计算机软件中的一个“,”号被误写为“·”，而使飞往火星的探测器发生爆炸，造成高达数亿美元的损失。

为了克服这一危机，一方面需要对程序设计方法、程序的正确性和软件的可靠性等问题进行系列的研究；另一方面，也需要对软件的编制、测试、维护和管理的方法进行研究，从而产生了程序设计方法学。

程序设计方法学是讨论程序的性质、程序设计的理论和方法的一门学科。它包含的内容比较丰富，例如，结构程序设计、程序正确性证明、程序变换、程序的形式说明与推导、程序综合、自动程序设计等。

在程序设计方法学中，结构程序设计占有十分重要的地位，可以说，程序设计方法学是在结构程序设计的基础上逐步发展和完善起来的。

1.1.2　结构化程序设计

1. 结构化程序设计

结构化程序设计的思想是在20世纪60年代末、70年代初为解决“软件危机”而形成的。多年来的实践证明，结构化程序设计策略确实使程序执行效率提高，由于减少了程序的出错率，大大减少了维护费用。

结构化程序设计就是一种进行程序设计的原则和方法，按照这种原则和方法可设计出结构清晰、易理解、易修改、易验证的程序。即，结构化程序设计是按照一定的原则与原理组织和编写正确且易读的程序软件技术。结构化程序设计的目标在于使程序具有一个合理结构，以保证和验证程序的正确性，从而开发出正确、合理的程序。

按照结构化程序设计的要求，设计出的程序设计语言称为结构化程序设计语言。利用结构化程序设计语言，或者说按结构化程序设计的思想和原则编制的程序称为结构化程序。

2. 结构化程序设计的特征与风格

结构化程序设计的主要特征如下：

(1) 一个程序按结构化程序设计方式构造时，就是一个结构化程序，包括三种基本控制结构：顺序结构、选择结构和循环结构。这三种结构都是单入口-单出口的程序结构。已经证明，一个任意大而且复杂的程序总能转换成这三种标准形式的组合。

(2) 有限制地使用goto语句。鉴于goto语句的存在使程序的静态书写顺序与动态执行顺序十分不一致，导致程序难读、难理解，容易存在潜在的错误，难于证明正确性，有人主张程序中禁止使用goto语句，但有人则认为goto语句是一种有效的措施，不应全盘否定而完全禁止使用。结构化程序设计并不在于是否使用goto语句，因此作为一种折衷，允许在程序中有限制地使用goto语句。

(3) 借助于结构化程序设计思想的程序设计语言来书写结构化程序，并采用一定的书写格式以提高程序结构的清晰性，增进程序的易读性。

(4) 强调程序设计过程中人的思维方式与规律，是一种自顶向下的程序设计策略，它通

过一组规则、规律与特有的风格对程序设计细分和组织。对于小规模程序设计，它与逐步精化的设计策略相联系，即采用自顶向下、逐步求精的方法对其进行分析和设计；对于大规模程序设计，它则与模块化程序设计策略相结合，即将一个大规模的问题划分为几个模块，每一个模块完成一定的功能。

1.1.3 模块化程序设计的方法

用模块化方法进行程序设计的技术在20世纪50年代就出现雏形。在进行程序设计时把一个大的程序按照功能划分为若干个小的程序，每一个小的程序完成一个确定的功能，在这些小的程序之间建立必要的联系，互相协作完成整个程序要完成的功能。这些小的程序称为程序的模块。

用模块化的方法设计程序，其过程犹如搭积木的过程，选择不同的积木块或采用积木块不同的组合就可以搭出不同的造型来。同样，选择不同的程序块或程序模块的不同组合就可以完成不同的系统架构和功能。

将一个大的程序划分为若干不同的相对独立的小程序模块，正是体现了抽象的原则，这种方法已经被人们接受。把程序设计中的抽象结果转化成模块，不仅可以保证设计的逻辑正确性，而且更适合项目的集体开发。各个模块分别由不同的程序员编制，只要明确模块之间的接口关系，模块内部细节的具体实现可以由程序员自己随意设计，而模块之间不受影响。

具体到程序来说，模块通常是指可以用一个名字调用的一个程序段。对于不同的程序设计语言，模块的实现和名称也不相同。在BASIC、FORTRAN语言中的模块称作子程序；Pascal语言中的模块称为过程；C语言中的模块叫函数。

1.1.4 面向对象的程序设计

结构化程序设计的主要技术是自顶向下、逐步求精，采用单入口-单出口的控制结构。

自顶向下是一种分解问题的技术，与控制结构无关；逐步求精是指结构化程序设计连续分解，最终成为三种基本控制结构（顺序、选择、循环）的组合。

结构化程序设计的结果是使一个结构化程序最终由若干个过程组成，每一过程完成一个确定的工作。对于结构化的程序设计语言，其数据结构是解决问题的中心。一个软件系统的结构是围绕一个或几个关键数据结构为核心而组成的。在这种情况下，软件开发一直被两大问题所困扰：一是如何超越程序的复杂性障碍；二是计算机系统中如何自然地表示客观世界。

Niklaus Wirth提出的“算法＋数据结构＝程序设计”，在软件开发进程中产生了深远的影响，但软件系统的规模越来越大，复杂性不断增长，以致不得不对“关键数据结构”进行重新评价。在这种系统中，许多重要的过程和函数（子程序）的实现严格依赖于关键数据结构，如果这些“关键数据结构”的一个或几个数据有所改变，则涉及到整个软件系统，许多过程和函数必须重写，甚至因为几个关键数据结构改变，导致软件系统的彻底崩溃。

1. 面向对象的基本概念与特征

20世纪80年代末兴起的面向对象，按照人们通常的思维方式建立问题模型，设计出尽

可能自然的表示求解方法的软件。

所谓建立模型就是建立问题领域中事物间的相互关系，而表示求解问题的方法就是人们思维方式的描述方法。

面向对象的开发方法不仅为人们提供了较好的开发风格，而且在提高软件的生产率、可靠性、可重用性、可维护性等方面有明显的效果，已成为当今计算机界最为关注的一种开发方法。

2. 面向对象的主要特点

结构化方法强调过程抽象和模块化，将现实世界映射为数据流和过程，过程之间通过数据流进行通信，数据作为被动的实体，被主动的操作所加工，是以过程（或操作）为中心来构造系统和设计程序的。

面向对象方法把世界看成是独立对象的集合，对象将数据和操作封装在一起，提供有限的外部接口，其内部的实现细节、数据结构以及对它们的操作是外部不可见的，对象之间通过消息相互通信，当一个对象为完成其功能需要请求另一个对象的服务时，前者就向后者发出一条消息，后者在接收到这条消息后，识别该消息并按照自身的适当方式予以响应。

面向对象方法和结构化方法相比具有以下一些特点：

（1）模块化、信息隐藏与抽象；

（2）自然性与共享性；

（3）并发性；

（4）重用性。

面向对象方法是一种新的程序设计方法，它具有很多优点，已经被广泛使用。

3. 可视化的程序设计

在用传统程序设计语言来设计程序时，都是通过编写程序代码来设计输入输出的用户界面，在设计过程中看不到界面的实际显示效果，必须编译后运行程序才能观察。如果对界面的效果不满意，还要回到程序中去修改。有时候，这种编程修改的操作可能要反复多次，大大影响了软件开发效率。现在的可视化设计工具，把 Windows 界面设计的复杂性“封装”起来，定义为对象，编程人员不必为界面设计而编写大量程序代码，只需要按设计要求的屏幕布局，用系统提供的工具，在屏幕上画出各种“部件”，即图形对象，并设置这些图形对象的属性，系统就会自动产生界面设计代码。这样一来，程序设计人员只需要编写实现程序功能的那部分代码，从而可以大大提高程序设计的效率。

1.2　程序设计语言

1.2.1　程序设计的基本步骤

编制程序，解决问题的基本步骤如下。

1）分析问题以确定目的

设计程序是为了解决某些实际问题，所以，编制程序要有的放矢，要认真分析研究，确定程序要解决的实际问题。

2）提出算法

将问题分析的思路进一步明确化、详细化，建立解决问题的数学模型或者物理模型。把解题的步骤和方法一步一步地详细写下来，也就是提出算法，以便为下一步用计算机语言表达这些算法奠定基础。

3）编写程序代码

根据提出的算法，选择适当的程序设计语言，编写程序。程序设计语言是人们与计算机进行信息交流的重要工具之一。程序设计语言有许多不同的种类，主要包括机器语言、汇编语言、高级语言3种，人们可以根据实际的需要选择不同的程序设计语言。

4）程序调试

编写完程序后，要将程序输入到计算机进行调试，调试的目的是发现程序编写中的语法错误和程序设计算法上的逻辑错误，并将这些错误排除。一般在测试时，要选择一些有代表性的典型数据或者模拟某些特定的环境，对程序进行测试；现在的程序，还要求具备良好的人机交互界面，并且具备一定的容错功能，能发现输入数据中的错误，并给予提示，所有的这些都要通过程序调试加以解决。

5）投入运行

程序经过调试，纠正其中的错误后，就可以正式投入运行，实现程序设计的功能。

6）维护升级

由于实际的应用问题和客户的需求不是一成不变的，程序正式投入运行后，可能因为应用环境的变化（包括所使用硬件的更新）、客户需求的变化，要不断地进行维护，修改其中存在的漏洞，开发新增功能。当对某一个软件进行较大程度的修改，新增了较多功能，就称为软件的版本升级。一般而言，每一个软件都有一个版本号，版本号的变化反映了该软件产品的升级情况。

1.2.2 算法及其表示

1. 算法

算法就是对问题求解方法的精确描述。在进行程序设计时，最关键的问题是算法的提出。因为它直接关系到写出来程序的正确性、可靠性。如果没有认真地研究实际的问题，就草率地提出一些不成熟的算法，那么编写出来的程序就可能出现错误或疏忽。

算法作为对解题步骤的精确描述，应具备以下特点。

1）有穷性

一个算法必须在有限步骤之后结束，而不能无限制地进行下去。因此在算法中必须给出一个结束的条件。

例如：不能指望计算机算出圆周率的精确值，因为π是一个无穷不循环小数，无法求出它的精确值。

2）明确性

一个算法中的任何步骤都必须意义明确，不能模棱两可、含混不清，即不允许有二义性，

不能在计算机中使用诸如“老张对老李说：他的儿子考上了清华大学”这种歧义的表达方法。到底是老张的儿子上了大学还是老李的儿子上了大学？无法确知。

3）可执行性

所采用的算法必须能够在计算机上执行，因此，在算法中所有的运算必须是计算机能够执行的基本运算。要计算机执行的步骤，计算机应该能够实现，不能提出像“让计算机去煮饭，煮完饭之后再炒菜”之类的算法，至少它在目前无法实现。

4）有一定的输入与输出

要计算机解决问题时，总是需要输入一些原始的数据，计算机向用户报告结果时，总是要输出一些信息。因此，一个算法中必须有一定的输入与输出。

2. 算法的表示

算法可以用自然语言、图形、表格、伪代码（类似于某种高级语言）的形式来表示，不同的描述方法各有特点，其中流程图和 N-S 图是使用较多的算法描述方法。

1）文字描述

例如让计算机求整数 A，B 的最大公约数，表示为 GCD(A，B)。其计算方法如下。

① 输入 A，B 的值。

② 比较 A，B 的大小，如果 A＜B 则将 A，B 的值互换。

③ 计算 A－B 的值，并将 A 的值改为 A－B。

④ 比较新的 A 值和 B 值的大小，如果 A＝B，则 A（或者 B）就是所求的两个数的最大公约数，程序继续下一步；否则程序转到第②步。

⑤ 输出最大公约数。

⑥ 结束。

2）流程图描述

流程图是一种能够比较形象地描述“算法”的工具，它对于编制程序很有帮助。流程图又称为框图，是由几种不同的图形组合而成的，算法流程图如图 1.1 所示。流程图中的图形如图 1.2 所示。

流程图中元素表示意义如下：

① 起止框。如图 1.2(a)所示，它代表一个算法的开始与结束之处。

② 处理框。如图 1.2(b)所示，它表示算法中的一个或若干个步骤，这些步骤不涉及输入与输出。

③ 输入输出框。如图 1.2(c)所示，它表示一个算法中需要进行输入或输出处理的步骤。为了与一般的处理步骤区别开来，输入输出框采用了平行四边形的形式。

④ 条件判断框。如图 1.2(d)所示，当一个算法中需要依据某一条件来决定后续操作时，采用条件判断框。

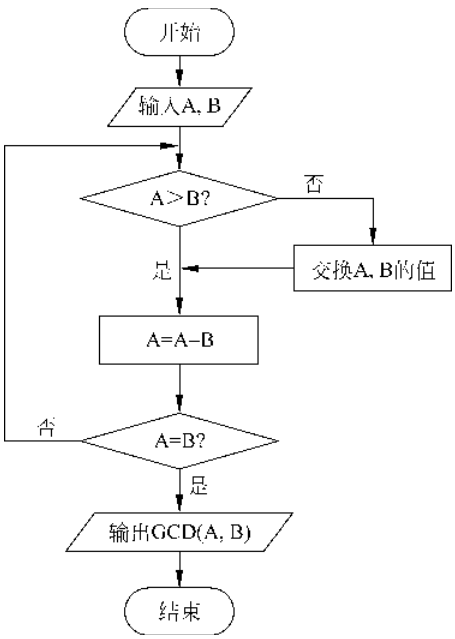

图 1.1　流程图的表示

⑤ 流程线。如图 1.2(e)所示，它表明每一步骤之间的先后顺序，表示一个数据的流向。

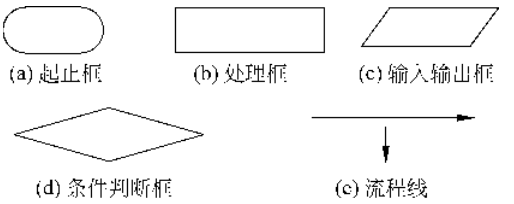

图 1.2　各种流程图元素

3）N-S 图描述

N-S 图是另一种算法表示法，是由美国人 I. Nassi 和 B. Shnei derman 共同提出的。它是根据程序是由三种基本结构组成提出，各基本结构之间的流程线就是多余的，可以省略。在 N-S 图中，一个算法就是一个大的矩形框，框内包含若干个基本框，三种基本结构 N-S 图描述如下。

① 顺序结构 N-S 图如图 1.3 所示，执行顺序为先 A 后 B。

② 选择结构 N-S 图如图 1.4 所示。图(a)表示条件为真时执行程序 A，条件为假时执行程序 B；图(b)表示当条件为真时执行程序 A，为假时什么都不做。

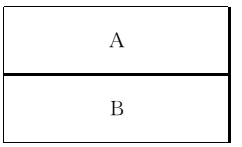

图 1.3　顺序结构 N-S 图

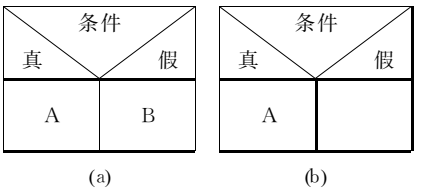

图 1.4　选择结构 N-S 图

③ while 型循环的 N-S 图如图 1.5 所示，条件为真时一直循环执行循环体 A，直到条件为假时才跳出循环。

do-while 型循环的 N-S 图如图 1.6 所示，一直循环执行循环体 A，直到条件为假时才跳出循环。

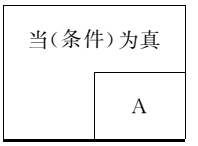

图 1.5　while 型循环 N-S 图

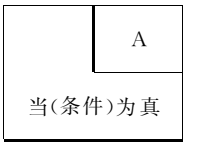

图 1.6　do-while 循环 N-S 图

4）PDA 图描述法

PDA(problem analysis diagram)，是近年来在软件开发中被广泛使用的一种算法表示方法。前面介绍的流程图、N-S 图、都是自上而下的顺序描述，而 PDA 图除了自上而下以外，还包含自左向右的展开，所以说 PDA 图是用二维来描述算法的。它能更好地展现算法的层次结构，更直观易懂。

下面介绍几种基本的 PDA 图。

① 顺序结构如图 1.7 所示。

② 选择结构包括单分支结构、双分支结构和多分支结构，如图 1.8 所示。

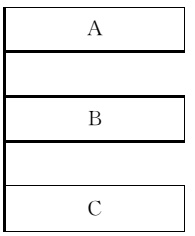

图 1.7　顺序结构 PDA 图

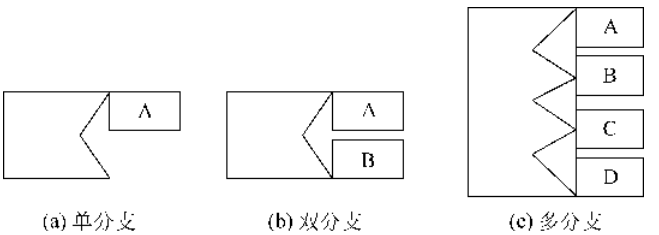

图 1.8　选择结构 PDA 图

③ while 型循环如图所示。do-while 型循环如图 1.9 和图 1.10 所示。

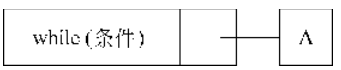

图 1.9　while 型循环 PDA 图

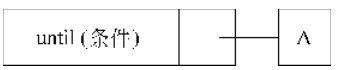

图 1.10　do-while 型循环 PDA 图

1.2.3　程序设计语言

要把算法转变为计算机能接受的形式输入计算机，就必须克服以下几个困难。

(1) 首先，计算机内部不接受自然语言表示的信息，只能接受二进制信息。

(2) 其次，由于自然语言往往和其具体环境有关，一个句子，一个单词在不同的环境中，含义可能不同，具有“二义性”。

所以需要一种能够准确表达编制程序思想的、能够被计算机所接受的、相当严谨，不具有二义性的表达方法的出现，这就是计算机语言。

计算机语言是人们与计算机之间进行信息沟通的工具，它是一种计算机能够接受的信息，它由一些表达符号(包括单词、符号、数字)和语法(规定这些表达符号的运用方式)组成，能准确地表示人们要求计算机进行的操作。

根据计算机语言和人类自然语言的接近程度，计算机语言分为：机器语言、汇编语言、高级语言三种。

1. 机器语言

计算机不能识别与执行人类自然语言。计算机内部存储数据和指令是采用二进制(0 和 1)方式的。计算机只能接受和识别由 0 和 1 组成的二进制信息。每一类型的计算机都分别规定了由若干个二进制位的信息(即若干个 0 或 1 组成的信息)组成一条指令。

例如，某种计算机规定以 1011011000000000 这样的指令作为“加法”指令，遇到这样的指令计算机就执行一次加法。以 1011010100000000 作为“减法”指令，遇到这样的指

令计算机就执行一次减法操作。以上两条指令第 6 和第 7 位不同（以最左边的一位作为第 0 位）。

这种计算机能直接识别和执行的二进制形式的指令称为“机器指令”。例如前面介绍的 1011011000000000 和 1011010100000000 就是两条机器指令。一条机器指令产生一个相应的机器操作。每一种计算机都确定有若干条指令（例如加法指令、减法指令、传送指令、取数指令、存数指令、输出指令等），以实现不同的操作。一种计算机的指令的集合称为该计算机的机器语言（machine language），或者说该计算机的指令。正如同用算盘算题一样，每一条珠算口诀就是一条“指令”，算盘全部口诀之和就是“珠算语言”。也就是说，“语言”是全部指令的总和。人们为了解决某一问题，可以从该“语言”中选择所需的指令，组成一个指令序列。这个指令序列称为“机器语言程序”。

不同的计算机系统的电路逻辑是不同的，因此，对不同的计算机，即使是执行同一种操作，它们的指令是不同的。或者说，不同的计算机有不同的指令系统。譬如有的计算机指令的长度为 16 位，有的计算机则为 32 位，假如用 A 机器上的机器语言编写了一个程序，拿到 B 机器上就不能用，需要重新编写程序。这显然是很不方便的。因此说，机器语言是依赖于具体计算机的，它是“面向机器”的语言。

用机器语言编写出的程序，计算机能直接识别和执行，执行效率比较高。但是，直接用 0 和 1 这样的二进制代码编程序，难学、难记、难写、难检查、难调试、难以推广。只有在计算机产生初期，计算机专业人员才用机器语言编写程序。

2. 汇编语言

为了克服机器语言的缺点，人们用一些容易记忆和辨别的英文单词或缩写符号代替机器指令，就产生了汇编语言，绝大多数的汇编语言指令和机器语言存在着直接对应关系，所以汇编语言实际上是一种符号语言。

例如，汇编语言中以 MOV（MOVE 的缩写）代表数据“传送”，ADD 代表“加”等。这些符号含义明确，容易记忆，所以又称为助记符。用这些助记符编程，可读性好，容易查错，修改也方便。然而，汇编语言对人来说固然是方便了，可是机器不认识了。为了解决这个问题，需要建立一个“符号与指令代码”对照表，将每个助记符逐个扫描查表，将它转换为对应的机器语言程序。这个工作由一个叫做“汇编程序”的语言处理程序来完成，翻译出的程序叫做“目标程序”。汇编语言也是一种面向机器的语言，仍然必须了解机器结构才能编程，但比机器语言易读、易改，执行速度与机器语言相仿，比高级语言快得多，所以直到现在仍广泛应用于实时控制、实时处理等领域中。

3. 高级语言

汇编语言虽然较机器语言有所改善，但并未从根本上摆脱指令系统的束缚，它与指令仍然是一一对应的，而且与自然语言相距甚远，很不符合人们的习惯。

为了从根本上改变计算机的语言体系，必须从两方面下功夫：一是力求接近自然语言，二是力求脱离于具体机型，使语言与指令系统无关，达到程序通用的目的。在长期实践的基础上，人们终于创造出独立于机型、表达方式接近于被描述问题、容易学习使用的高级语言。