工业和信息化普通高等教育“十二五”规划教材立项项目
21世纪高等学校计算机规划教材
21st Century University Planned Textbooks of Computer Science

C语言程序设计

C Language

宁爱军 熊聪聪 主编
张艳华 满春雷 赵奇 编著

- 以Visual C++6.0为编程环境，内容涵盖C及C++基础
- 分析问题、设计算法、编写和调试程序为教学步骤
- 重点培养学生设计算法、编程解决实际问题的能力

高校系列

人民邮电出版社
POSTS & TELECOM PRESS

工业和信息化普通高等教育“十二五”规划教材立项项目
21世纪高等学校计算机规划教材
21st Century University Planned Textbooks of Computer Science

C语言程序设计

C Language

宁爱军 熊聪聪 主编
张艳华 满春雷 赵奇 编著

高校系列

人民邮电出版社
北京

图书在版编目（CIP）数据

C语言程序设计 / 宁爱军，熊聪聪主编 ；张艳华，满春雷，赵奇编著. -- 北京 ：人民邮电出版社，2011.2（2012.1 重印）
21世纪高等学校计算机规划教材. 高校系列
ISBN 978-7-115-24676-9

Ⅰ. ①C… Ⅱ. ①宁… ②熊… ③张… ④满… ⑤赵… Ⅲ. ①C语言－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2011)第009073号

内 容 提 要

本书介绍 C 语言的基础知识，以 Visual C ++ 6.0 为编程环境，通过分析问题、设计算法、编写和调试程序这些步骤，力求让读者掌握分析问题的方法，培养设计算法的能力。

全书共 17 章。第 1 章～第 3 章介绍程序设计与 C 语言的基础知识；第 4 章～第 7 章介绍 4 种基本的算法与程序设计；第 8 章～第 11 章介绍函数、编译预处理、指针、结构体和链表等；第 12、第 13 章介绍位运算与文件；第 14 章～第 16 章介绍面向对象程序设计的类与对象、继承与派生、多态性与虚函数；第 17 章介绍几个综合实例。

本书内容由浅入深，具有较强的可读性，适合大学生作为程序设计课程教材，也可作为 C 语言爱好者编程的参考书。

21 世纪高等学校计算机规划教材——高等系列

C 语言程序设计

◆ 主　　编　宁爱军　熊聪聪
编　　著　张艳华　满春雷　赵　奇
责任编辑　蒋　亮

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市海波印务有限公司印刷

◆ 开本：787×1092　1/16
印张：24.75　　2011 年 2 月第 1 版
字数：654 千字　　2012 年 1 月河北第 2 次印刷

ISBN 978-7-115-24676-9

定价：42.00 元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154
广告经营许可证：京崇工商广字第 0021 号

前　言

C 语言是结构化的程序设计语言，它功能丰富，使用灵活，可移植性好，广泛应用于科学计算、工程控制、网络通信、图像处理等领域。C 语言是学习程序设计的首选语言，也是实用性较强的编程语言。

本书以 Visual C ++ 6.0 为编程环境，通过分析问题、设计算法、编写和调试程序的步骤，重点培养学生分析问题和算法设计的能力，以及语言和语法知识的理解和掌握。力求弥补以往在程序设计课程教学中，学生不但能掌握语言语法知识，而且要培养自己设计算法、编写程序解决实际问题的能力。

本书共分为 17 章。其中第 1 章、第 2 章介绍程序设计的基础和 Visual C++ 6.0 集成开发环境；第 3 章介绍 C 语言的基础知识；第 4 章、第 5 章、第 6 章、第 7 章介绍顺序、选择、循环、数组的算法设计和程序设计；第 8 章介绍函数；第 9 章、第 10 章、第 11 章介绍编译预处理、指针、结构体和链表等；第 12 章、第 13 章介绍位运算和文件；第 14 章、第 15 和第 16 章介绍面向对象程序设计的类和对象、继承与派生、多态性与虚函数；第 17 章介绍几个综合实例；每章后配有丰富的选择题、填空题和编程题。

全书内容由浅入深、循序渐进，可读性强，是适合大学生阅读的程序设计课程教材，也可以作为 C 语言编程的参考书。

教师选用本书作为教材，可以根据授课学时情况适当取舍教学内容。教学建议如下：

（1）如果学时充分，建议系统学习全部内容。如果学时较少，建议以第 1 章～第 11 章为教学重点。后续章节可以在选修课或课程设计中介绍，也可以建议学生自学。

（2）第 3 章先重点掌握简单格式的输入输出方法，在需要使用复杂格式的输入输出时再回来学习。

（3）第 4 章～第 7 章应该先进行问题分析、算法设计，后进行程序设计和程序调试。努力培养分析问题、解决问题的能力。

（4）学生应该认真完成课后习题，以巩固语言和语法知识，培养实际编程能力，力求达到全国计算机等级考试（二级 C 语言）要求的水平。

本书的作者都是长期从事软件开发和大学程序设计课程教学的一线教师，具有丰富的软件开发和教学经验。本书由宁爱军和熊聪聪任主编，并负责全书的总体策划、统稿和定稿。第 1 章、第 2 章、第 3 章由张艳华编写，第 4 章、第 5 章、第 6 章、第 7 章、第 8 章由宁爱军编写，第 9 章、第 10 章、第 11 章由满春雷编写，第 12 章、第 13 和第 17 章由赵奇编写，第 14 章、第 15 章、第 16 章由熊聪聪编写，参加本书编写工作的还有杨光磊、李伟、窦若菲、王燕、王辉、张浥楠等。本书的编写和出版，还得到了天津科技大学各级领导的关怀和老师的指导，在此一并表示感谢。

本书的成稿是作者多年软件研发和教学经验的总结，但是由于水平有限，书中肯定还存在很多缺点和不足，恳请专家和读者批评指正。联系信箱：ningaijun@sina.com。

编　者

2010 年 12 月

目　录

第 1 章 程序设计基础

本章主要介绍程序设计语言及其分类、C 语言的历史与特点、程序设计的本质、算法及其表示方法、结构化的程序设计方法等内容。主要目的是使读者对程序设计有一个初步的了解。

1.1 程序设计语言

计算机和人之间不能完全使用自然语言进行交流，还需要借助计算机能够理解并执行的“计算机语言”。和人类语言类似，计算机语言是语法、语义与词汇的集合，它用来表达计算机程序。计算机语言也称为程序设计语言。

C 语言是最常用的程序设计语言之一，通过对 C 语言的扩充还产生了如 C++、Java、C#等语言。程序设计语言种类较多，但是不同语言具有相通之处。学好 C 语言可以为学习其他语言打下基础。

1.1.1 什么是程序

人们操作计算机完成各项工作，实际上是执行计算机中的各种程序，如操作系统、文字处理程序、手机内置的各类应用程序等。

程序是用来完成特定功能的一系列指令。通过向计算机发布指令，程序设计人员可以控制其执行某个操作或进行某种运算。一组指令构成一个程序，可以用来解决一个具体问题。简单的程序可能仅仅向屏幕输出一段符号，而复杂的程序可以完成更多功能。

一个程序总是按照既定顺序执行，完成编程人员设计的任务。虽然每个程序内部执行顺序可能不同，完成的任务有大有小，但程序编译成功进入执行状态后，其功能是不能被随心所欲修改的，除非重新编写程序、编译并执行程序。

1.1.2 语言的分类

自计算机诞生以来，产生了上千种程序设计语言，有些已被淘汰，有些得到了推广和发展。程序设计语言经历了由低级到高级的发展过程，可以分为机器语言、汇编语言、高级语言和面向对象的语言。低级语言包括机器语言和汇编语言，高级语言有很多种，如 C、Basic、Fortran 等，面向对象的语言如 C++、Visual Basic、Java 等。越低级的语言越接近计算机的二进制指令，越高级的语言越接近人类的思维方式。

1. 机器语言

机器语言是计算机能够直接识别并执行的二进制指令。机器语言指令由计算机的指令系统提供，阅读与编写比较费事，浪费时间，也容易出错。不同计算机的指令系统也不同，因此使用机器指令编写的程序通用性较差。

2. 汇编语言

汇编语言用助记符来代替机器语言的指令码，使机器语言符号化。如加法运算表示为“ADD AX, DX”。汇编语言比机器语言更进一步，但是编程人员仍然需要对机器硬件有深入了解，没有摆脱对具体机器的依赖，编程仍然具有较大难度。

3. 高级语言

为了解决计算机硬件的高速度和程序编制的低效率之间的矛盾，20 世纪 50 年代末期产生了“程序设计语言”，也称为高级语言。高级语言比较接近自然语言，直观、精确、通用、易学、易懂，编程效率高，便于移植。例如，语句“c = a + b”表示“求 a + b 的值并赋给 c”。高级语言有上千种，但实际应用的仅有十几种，如 BASIC、PASCAL、C、FORTRAN、ADA、COBOL、PL/I 等。

4. 面向对象的程序设计语言

面向对象的程序设计语言更接近人们的思维习惯。它将事物或某个操作抽象成类，将事物的属性抽象为类的属性，事物所能执行的操作抽象为方法。常用的面向对象语言有 Visual C++、Visual Basic、Java 等。

因为计算机不能直接识别高级语言，因此用高级语言编写的程序必须被翻译成计算机能识别的目标程序。程序执行有编译执行和解释执行两种方式。

（1）编译执行是将源程序翻译生成一个可执行的目标程序，该目标程序可以脱离编译环境和源程序独立存在和执行。

（2）解释执行是将源程序逐句解释成二进制指令，解释一句执行一句，不生成可执行文件，它的执行速度比编译方式慢。

1.1.3 C 语言简介

C 语言的诞生源于系统程序设计的深入研究和发展。它作为书写 UNIX 操作系统的语言，伴随着 UNIX 的发展和流行而得到发展与普及。

（1）1967 年，英国剑桥大学的 M.Richards 在 CPL（Combined Programming Language）语言的基础上，实现了 BCPL（Basic Combined Programming Language）语言。

（2）1970 年，美国贝尔实验室的 K.Thompson 以 BCPL 语言为基础，设计了一种类似于 BCPL 的语言，称为 B 语言。他用 B 语言在 PDP-7 机上实现了第一个实验性的 UNIX 操作系统。

（3）1972 年，贝尔实验室的 D.M.Ritchie 为克服 B 语言的诸多不足，在 B 语言的基础上重新设计了一种语言，由于是 B 的后继，故称为 C 语言。

（4）1973 年，贝尔实验室的 K.Thompson 和 D.M.Ritchie 合作，用 C 语言重新改写了 UNIX 操作系统。此后随着 UNIX 操作系统的发展，C 语言的应用越来广泛，影响越来越大。此时的 C 语言主要还是作为实验室产品在使用，并依赖于具体的机器，直到 1977 年才出现了独立于具体机器的 C 语言编译版本。

随着微型计算机的普及，出现了许多 C 语言版本。由于没有统一标准，这些语言之间出现了许多不一致的地方。为了改变这种情况，美国国家标准学会（ANSI）为 C 语言制定了一套 ANSI

标准，就是标准 C 语言。1983 年，美国国家标准学会颁布了 C 语言的新标准版本“ANSI C”。“ANSI C”比标准 C 语言有了很大的补充和发展。目前 C 语言的最新版本为 ANSI C99，VC ++ 也支持该标准。

C 语言使用灵活，并具有强大生命力，已经广泛应用于科学计算、工程控制、网络通信、图像处理等领域。C 语言是结构化的程序设计语言，具有如下特点：

（1）语言简洁、使用灵活，便于学习和应用。C 语言的书写形式较其他语言更为直观、精炼。

（2）语言表达能力强。运算符达 30 多种，涉及的范围广，功能强。

（3）数据结构系统化。C 语言具有现代语言的各种数据结构，并具有数据类型的构造功能，因此便于实现各种复杂的数据结构的运算。

（4）控制流结构化。C 语言提供了功能很强的各种控制流语句（if、while、for、switch 等），并以函数作为主要结构，便于程序模块化，符合现代程序设计风格。

（5）C 语言生成的程序质量高，程序运行效率高。实验表明，C 语言源程序生成的可执行程序的运行的效率仅比汇编语言的效率低 10%～20%。C 语言编程速度快，程序可读性好，易于调试、修改和移植，这些优点是汇编语言无法比拟的。

（6）可移植性好。统计资料表明。C 语言程序 80%以上的代码是公共的，因此稍加修改就能移植到各种不同型号的计算机上。

C 语言也存在一些不足之处，如编程自由度比较大。但总的来说，C 语言是一个出色而有效的现代通用程序设计语言。

1.1.4　C 语言组成

C 程序由函数构成。一个 C 程序至少由一个函数构成，而且至少包含一个名为 main 的主函数。

【例 1.1】 一个简单的 C 程序，向屏幕输出一句话“I like Programming!”。

```
#include<stdio.h>
void main( ) //函数首部
{    printf("I like Programming!\n"); //输出一串字符
}
```

程序的运行结果如下。

```
I like Programming!
Press any key to continue
```

说明：

（1）函数可以分成函数首部和函数体。花括弧“{}”括起来的是函数体。

（2）语句由一些基本字符和定义符按照 C 语言的语法规定组成。每个语句以分号结束。

（3）“//”后边的文字为注释，它们不执行，不影响程序的运行。

1.2　计算机的组成与程序设计的本质

程序设计与计算机组成有密切关系，学习计算机组成方面的知识，可以更好地理解程序设计的本质。

1.2.1 计算机系统结构

“计算机之父”冯·诺依曼提出的计算机系统结构如下：

（1）计算机由控制器、运算器、存储器、输入设备和输出设备 5 个部分构成。

（2）计算机采用二进制，指令和数据均以二进制数形式表示和存放。

（3）计算机按照程序规定的顺序将指令从存储器中取出，并逐条执行。

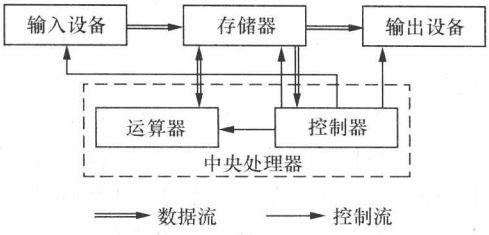


图 1-1 冯·诺依曼体系结构

如图 1-1 所示，控制器集中控制其他设备。信息分为数据信息和控制信息两种。在控制指令的控制下，数据按照如下方式“流动”：由输入设备输入数据，存储在存储器中，控制器和运算器直接从存储器中取出数据（包括程序代码和运算对象）进行处理，结果存储在存储器内，并由输出设备输出。

1.2.2 程序设计的本质

程序设计的本质是设计能够利用计算机的 5 个部件完成特定任务的指令序列。

【例 1.2】 用键盘输入价格与斤数，计算樱桃的总价。

```
#include<stdio.h>
void main()
{  int price,number,total;
   scanf("%d%d",&price,&number);                    //输入
   total=price*number;                              //计算
   printf("total=%d\n",total);                      //输出
}
```

在运行程序时输入数据“10 3”后按“回车”，显示总钱数为 30，程序的运行结果如下。

```
10 3
total=30
Press any key to continue
```

说明：

（1）整个程序保存在计算机的存储器中。

（2）数据存储在存储器中。定义 3 个变量 price、number 和 total，分别占用一块存储空间，用于存放价格、斤数和总价。

（3）通过键盘输入价格与斤数。

（4）由运算器来执行乘法，求出总价。

（5）通过输出设备输出并显示程序执行的结果。

通过本例可见一个程序离不开 5 个部件的配合。并且一个程序可以没有输入，但是一定要有输出才能知道程序的运行结果。

1.2.3 程序设计的过程

程序设计的一般过程如表 1.1 所示，在编写程序解决实际问题时，一般应按照这 6 个步骤，

逐一实施来完成程序。

表 1.1　　程序设计过程

1	分析和定义实际问题	做什么！
2	建立处理模型	如何做！
3	设计算法	
4	设计流程图	
5	编写程序	实现程序！
6	调试程序和运行程序	

1. 分析和定义实际问题

通过对实际问题的深入分析，准确地提炼、描述要解决的问题，明确要求。

2. 建立处理模型

实际问题都是有一定规律的数学、物理等过程，用特定方法描述问题的规律和其中的数值关系，是为确定计算机实现算法而做的理论准备。如求解图形面积一类的问题，可以归结为数值积分，积分公式就是为解决这类问题而建立的数学模型。

3. 设计算法

将要处理的问题分解成计算机能够执行的若干特定操作，也就是确定解决问题的算法。例如，由于计算机不能识别积分公式，需要将公式转换为计算机能够接受的运算，如选择梯形公式或辛普森（Simpson）公式等。

4. 设计流程图

在编写程序前给出处理步骤的流程图，能直观地反映出所处理问题中较复杂的关系，从而使编程时思路清晰，避免出错。流程图是程序设计的良好辅助工具，它作为程序设计资料也便于交流。

5. 编写程序

编程是指用某种高级语言按照流程图描述的步骤写出程序，也叫做编码。使用某种语言编写的程序叫源程序。

6. 调试程序和运行程序

将写好的程序上机检查、编译、调试和运行，并纠正程序中的错误。

1.3　算法的概念和特性

编写程序之前，首先要找出解决问题的方法，并将其转换成计算机能够理解并执行的步骤，即算法。算法设计是程序设计过程中的一个重要步骤。

1.3.1　什么是算法

算法即解决一个问题所采取的一系列步骤。著名的计算机科学家 Nikiklaus Wirth 提出如下公式：

程序 = 数据结构 + 算法

其中，数据结构是指程序中数据的类型和组织形式。

算法给出了解决问题的方法和步骤，是程序的灵魂，决定如何操作数据，如何解决问题。同一个问题可以有多种不同算法。

1.3.2 算法举例

计算机程序的算法，必须是计算机能够运行的方法。理发、吃饭等动作计算机不能运行，而加、减、乘、除、比较和逻辑运算等就是计算机能够执行的操作。

【例 1.3】 求 1 + 2 + 3 + 4 +…+ 100。

第一种算法是书写形如“1 + 2 + 3 + 4 + 5 + 6 +…+ 100”的表达式，其中不能使用省略号。这种算法太长，写起来很费时，且经常出错。

第二种算法是利用数学公式：

$$\sum_{n=1}^{100} n = (1+100)\times 100/2$$

相比之下，第二种算法要简单得多。但是，并非每个问题都有现成的公式可用，如求 100! = 1 × 2 × 3 × 4 × 5 × … × 100。

【例 1.4】 求 5!=1 × 2 × 3 × 4 × 5。

```
step1:  p=1
step2:  i=2
step3:  p=p × i
step4:  i=i+1
step5:  如果 i<=5，那么转入 step3 执行
step6:  输出 p，算法结束
```

其中 p 和 i 是变量，它们各占用一块内存，如图 1-2 所示。变量可以被赋值，也可以取出值参加运算。本例通过循环条件“i<=5”，使得乘法操作执行 4 次。

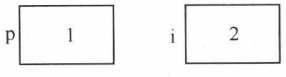


图 1-2 变量示意

【例 1.5】 求 1 × 2 × 3 ×… × 100。

```
step1:  p=1
step2:  i=2
step3:  p=p × i
step4:  i=i+1
step5:  如果 i<=100，那么转入 step3 执行
step6:  输出 p，算法结束
```

只需要在【例 1.4】算法的基础上，将循环条件改为“i<=100”，使得乘法操作执行 99 次就可以求出 100 个数的乘积。

【例 1.6】 求 1 × 3 × 5 × … × 101。

```
step1:  p=1
step2:  i=1
step3:  p=p×i
step4:  i=i+2
step5:  如果 i<=101，那么转入 step3 执行
step6:  输出 p，算法结束
```

只需要将 i 的初值改为 1、每次循环增加 2 就可以了。读者在学习过程中要多观摩已有的程

序，分析其算法，并力求有所创新。

1.3.3　算法的特性

算法应该具有以下特性：

（1）有穷性。算法经过有限次的运算就能得到结果，而不能无限执行或超出实际可以接受的时间。如果一个程序需要执行 1000 年才能得到结果，对于程序执行者而言，基本就没有什么意义了。

（2）确定性。算法中的每一个步骤都是确定的，不能含糊、模棱两可。算法中的每一个步骤不应当被解释为多种含义，而应当十分明确。比如描述“小王递给小李一件他的衣服”，这里，衣服究竟是小王的，还是小李的呢？

（3）输入。算法可以有输入，也可以没有输入，即有 0 个或多个输入。

（4）输出。算法必须有一个或多个输出，用于显示程序的运行结果。

（5）可行性。算法中的每一个步骤都是可以执行的，都能得到确定的结果，而不能无法执行。比如用 0 作为除数就不能执行。

1.4　算法的表示方法

算法的表示方法有很多种，常用的有自然语言、伪代码、传统流程图、N-S 流程图、PAD 图等。本节主要讲述常用的算法表示方法，其中流程图是学习和掌握的重点。

1.4.1　自然语言

使用自然语言表示，就是采用人们日常生活中的语言。如求两个数的最大值，可以表示为：如果 A 大于 B，那么最大值为 A，否则最大值为 B。但在描述“陶陶告诉贝贝她的小猫丢了”时，表示的是陶陶的小猫丢了还是贝贝的小猫丢了呢？就出现了歧义。可见使用自然语言表示算法时拖沓冗长，容易出现歧义，因此不常使用。

1.4.2　伪代码

伪代码用介于自然语言和计算机语言之间的文字和符号来描述算法。例如，求两个数的最大值可以表示为：

if　A 大于 B，then 最大值为 A，else 最大值为 B。

伪代码的描述方法比较灵活，修改方便，易于转变为程序，但是当情况比较复杂时，不够直观，而且容易出现逻辑错误。软件专业人员一般习惯使用伪代码，而初学者最好使用流程图。

1.4.3　传统流程图

流程图表示算法比较直观，它使用一些图框来表示各种操作，用箭头表示语句的执行顺序。传统流程图的常用符号如图 1-3 所示。将【例 1.5】求 $1 \times 2 \times 3 \times \cdots \times 100$ 的算法描述为传统流程图如图 1-4 所示。用传统流程图表示复杂的算法时不够方便，也不便于修改。

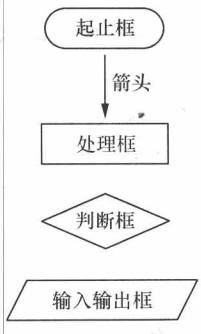


图 1-3　传统流程图的常用符号

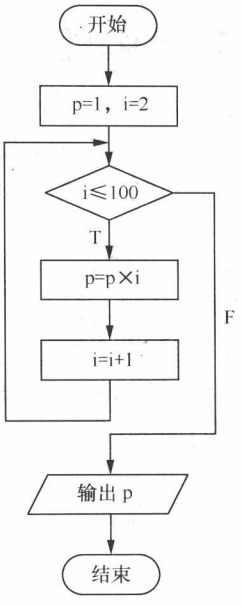


图 1-4　求 1 × 2 × 3 × … × 100 的传统流程图

1.4.4　N–S 流程图

N-S 流程图又称盒图，其特点是所有的程序结构均用方框表示。N-S 流程图的绘制比较节省空间，它避免了使用箭头任意跳转程序所造成的混乱，更加符合结构化程序设计的原则。它按照从上往下的顺序执行语句。例 1.5 求 1 × 2 × 3 × … × 100 算法的 N-S 流程图如图 1-5 所示。

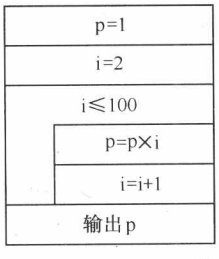


图 1-5　求 1 × 2 × 3 × … × 100 的 N-S 流程图

1.5　结构化的程序设计方法

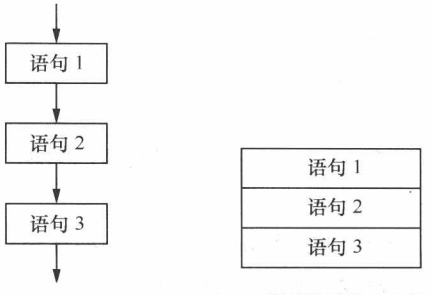
编出程序、得到运行结果仅是学习程序设计的基本要求，要全面提高编程的质量和效率，就必须掌握正确的程序设计方法和技巧，培养良好的程序设计风格，使程序具有良好的可读性、可修改性、可维护性。结构化程序设计方法是目前程序设计方法的主流之一。

1.5.1　结构化程序设计

1966 年，Bohra 和 Jacopini 提出了顺序结构、选择结构和循环结构 3 种基本结构，结构化程序设计方法使用这 3 种基本结构组成算法。已经证明，用 3 种基本结构可以组成解决所有编程问题的算法。

1. 顺序结构

顺序结构是指按照语句在程序中出现的先后次序执行，其流程图如图 1-6 所示。顺序结构里的语句可以是单条语句，也可以是一个选择结构或一个循环结构。



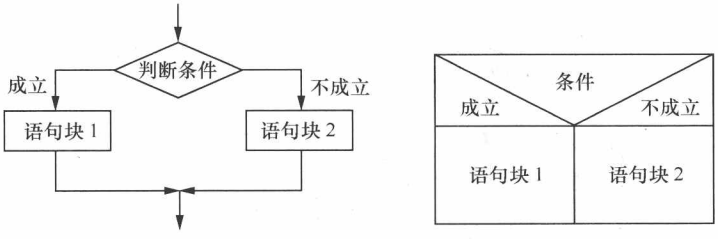
（a）顺序结构的传统流程图　　（b）顺序结构的 N-S 图

图 1-6　顺序结构

2. 选择结构

选择结构是指根据条件选择程序的执行顺序。

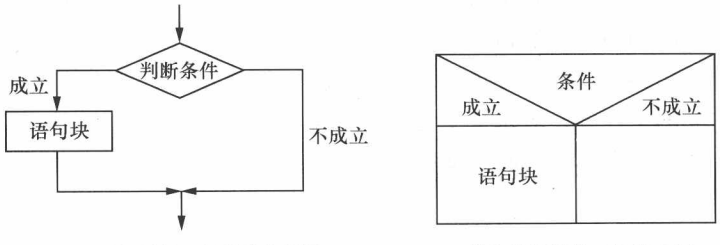
（1）选择结构一：流程图如图 1-7 所示，当条件成立时执行语句块①，否则执行语句块②。不管执行哪一个语句块，完成后继续执行选择结构后的语句。选择结构里的语句块可以是顺序语句，也可以是一个选择结构或一个循环结构。



（a）选择结构一的传统流程图　　（b）选择结构一的 N-S 图

图 1-7　选择结构一

（2）选择结构二：流程图如图 1-8 所示，当条件成立的时候执行语句块，否则什么都不执行。不管执行或不执行语句块，完成后继续执行选择结构后的语句。



（a）选择结构二的传统流程图　　（b）选择结构二的 N-S 图

图 1-8　选择结构二

3. 循环结构

循环结构是指设定循环条件，在满足该条件时反复执行程序中的某部分语句，即反复执行循环体。