JIXUJIAOYU（HANSHOU）
ZHUANSHENGBEN
GONGGONGKE
XILIEJIAOCAI

继续教育（函授）专升本公共课系列教材

C语言程序设计

林碧英　吴耀红　主编

中国电力出版社
www.cepp.com.cn

继续教育（函授）专升本公共课系列教材

C语言程序设计

主　编　林碧英　吴耀红
副主编　王素琴　王默玉

中国电力出版社
www.cepp.com.cn

内 容 提 要

本教材是针对继续教育（函授）专升本学生的特点而编写的，对内容的选择安排到问题的阐述、分析及解决方法都进行了精心的设计，由浅入深、通俗易懂，书中举有大量的例题。

本书共分11章，分别讨论：C语言程序的构成和书写格式；常量和变量；运算符与表达式；顺序结构程序设计；选择结构程序设计；循环结构程序设计；数组；指针；函数；结构体和文件等。

本书可作为继续教育（函授）专升本学生的教材，也可作为相关专业的本科生、专科生及工程技术人员学习和参考用书。

图书在版编目（CIP）数据

C语言程序设计/林碧英，吴耀红主编．—北京：中国电力出版社，2005

（继续教育（函授）专升本公共课系列教材）

ISBN 7-5083-3215-6

Ⅰ.C...　Ⅱ.①林...　②吴...　Ⅲ.C语言-程序设计-函授大学-教材　Ⅳ.TP312

中国版本图书馆CIP数据核字（2005）第026685号

中国电力出版社出版、发行

（北京三里河路6号　100044　http://www.cepp.com.cn）

北京同江印刷厂印刷

各地新华书店经售

*

2005年5月第一版　2005年5月北京第一次印刷

787毫米×1092毫米　16开本　14.25印张　371千字

印数0001—4000册　定价20.80元

前言

随着计算机技术的飞速发展，计算机已经应用在各个领域。为了更好地发挥计算机的作用，越来越多的人需要从过去的单纯会操作计算机变为今后会应用计算机解决实际问题。因此，学习一门高级程序设计语言，掌握用计算机解决问题的方法步骤，学会编写程序是非常重要的。C语言是面向过程的高级程序设计语言，它不仅具有高级语言的特点，又能够实现低级语言所能完成的操作。它不仅可以编写应用软件，还可以编写系统软件。因此，现在不同层次的高等教育，都开设了“C程序设计”这门课。为了帮助成人教育和函授教育的不同层次的学生，更好地掌握一门高级程序设计语言，我们特意编写了《C语言程序设计》这本书。编者均具有多年本科教学经历，更有多年成人、函授教学的经验，并且非常熟悉成人、函授教育的各个环节。我们在编写此书的过程中，根据学生以自学为主的特点，无论从内容的选择，还是问题的引入，以及问题的解决办法，都充分体现由浅入深、循序渐进、通俗易懂的原则。各个章节都配有大量例题，大多数例题都有分析过程，一部分还配有图表。总而言之，我们的编写目的是使学生通过以自学为主的方式，掌握C语言的语法规则和基本的编程方法，达到初中级程序设计的能力。

本书的内容分为十一章，其中第一章、第二章和第三章由王素琴老师编写，第四章、第五章、第六章和第十章由王默玉老师编写，第七章、第八章和第九章由吴耀红老师编写，第十一章由林碧英老师编写。最后由林碧英老师审阅定稿。

由于编写时间比较仓促，难免存在不足之处。欢迎广大读者批评指正。

《C语言程序设计》编写组

2004.11

目录

前言

第一章 概述 …… 1

1.1 算法的概念及描述 …… 1
1.1.1 算法的概念 …… 1
1.1.2 结构化程序设计 …… 2
1.1.3 算法的描述 …… 2
1.2 C语言的特点 …… 6
1.3 C程序的构成和书写格式 …… 7
1.3.1 C程序的构成 …… 7
1.3.2 C程序的书写格式 …… 9
1.4 C程序的上机步骤 …… 10
1.4.1 程序调试运行的基本过程 …… 10
1.4.2 在Turbo C 2.0环境下运行C程序 …… 11
习题 …… 15

第二章 常量及变量 …… 16

2.1 标识符与保留字 …… 16
2.1.1 标识符的构成规则 …… 16
2.1.2 保留字 …… 17
2.2 字符集 …… 17
2.3 基本数据类型 …… 17
2.3.1 C语言的数据类型 …… 17
2.3.2 基本数据类型的宽度及范围 …… 18
2.4 常量 …… 18
2.4.1 常量的概念 …… 18
2.4.2 整型常量 …… 19
2.4.3 实型常量 …… 19
2.4.4 字符常量 …… 20
2.4.5 转义字符常量 …… 20
2.4.6 字符串常量 …… 21
2.4.7 符号常量 …… 21
2.5 变量 …… 22
2.5.1 变量的概念 …… 22
2.5.2 变量的说明 …… 23
2.5.3 变量的初始化 …… 26
习题 …… 26

第三章 运算符及表达式 …… 29

3.1 概述 …… 29
3.2 算术运算符和算术表达式 …… 30
3.2.1 算术运算符 …… 30
3.2.2 算术表达式 …… 30
3.2.3 算术运算符的优先级与结合性 …… 31
3.3 关系运算符和关系表达式 …… 31
3.3.1 关系运算符 …… 31
3.3.2 关系表达式 …… 31
3.3.3 关系运算符的优先级与结合性 …… 32
3.4 逻辑运算符和逻辑表达式 …… 32
3.4.1 逻辑运算符 …… 32
3.4.2 逻辑表达式 …… 33
3.4.3 逻辑运算符的优先级与结合性 …… 33
3.5 赋值运算符 …… 35
3.5.1 赋值运算符和赋值表达式 …… 35
3.5.2 复合的算术赋值运算符 …… 35
3.5.3 赋值运算符的优先级与结合性 …… 36
3.6 自增自减运算符 …… 36
3.7 位运算符 …… 37
3.7.1 位运算符 …… 37
3.7.2 位运算符的优先级

与结合性 …… 38
3.8 条件运算符和条件表达式 …… 39
3.8.1 条件运算符 …… 39
3.8.2 条件运算符的优先级与结合性 …… 39
3.9 逗号运算符和逗号表达式 …… 39
3.9.1 逗号运算符和逗号表达式 …… 39
3.9.2 逗号运算符的优先级与结合性 …… 40
3.10 不同数据类型数据间的类型转换 …… 40
3.10.1 表达式中不同数据类型数据的混合运算 …… 40
3.10.2 强制数据类型转换 …… 41
3.10.3 赋值表达式的类型转换 …… 42
习题 …… 43
第四章 顺序结构程序设计 …… 46
4.1 C程序的基本结构及语句种类 …… 46
4.1.1 C程序的基本结构 …… 46
4.1.2 C语句种类 …… 46
4.2 数据的输入输出 …… 48
4.3 数据输出 …… 49
4.3.1 putchar函数（字符输出函数） …… 49
4.3.2 printf函数（格式输出函数） …… 50
4.4 数据输入 …… 56
4.4.1 getchar函数（字符输入函数） …… 56
4.4.2 scanf函数（格式输入函数） …… 57
4.5 顺序结构程序设计举例 …… 60
习题 …… 62
第五章 选择结构程序设计 …… 65
5.1 条件语句 …… 65
5.1.1 if语句的三种形式 …… 65
5.1.2 if语句的嵌套 …… 71
5.1.3 条件运算符（?:）代替条件语句 …… 73
5.2 switch语句 …… 74
5.3 选择结构程序举例 …… 77
习题 …… 78
第六章 循环结构程序设计 …… 82
6.1 while循环语句 …… 82
6.2 do－while循环 …… 85
6.3 for循环 …… 86
6.4 循环的嵌套 …… 90
6.5 几种循环的比较 …… 91
6.6 循环的中途退出 …… 91
6.6.1 break语句 …… 91
6.6.2 continue语句 …… 93
6.6.3 goto语句及用goto语句构成的循环 …… 94
6.7 循环程序举例 …… 95
习题 …… 100
第七章 数组 …… 107
7.1 一维数组说明及初始化 …… 107
7.1.1 一维数组的说明 …… 107
7.1.2 一维数组的引用 …… 108
7.1.3 一维数组的初始化 …… 109
7.2 一维数组应用举例 …… 110
7.3 字符数组与字符串 …… 115
7.3.1 字符串的存储 …… 115
7.3.2 字符型数组的初始化 …… 115
7.3.3 字符串的输入输出 …… 116
7.3.4 常用的字符串函数 …… 118
7.4 二维数组与双下标变量 …… 122
7.4.1 二维数组的说明 …… 122
7.4.2 二维数组的初始化 …… 123
7.4.3 二维数组程序设计举例 …… 124
习题 …… 127
第八章 指针 …… 130
8.1 指针的概念 …… 130

8.1.1 内存单元与地址…… 130
8.1.2 变量与地址…… 130
8.1.3 数组与地址…… 131
8.1.4 直接访问与间接访问…… 132
8.1.5 指针变量…… 133
8.2 指针变量的说明和指针运算符 …… 133
8.2.1 指针变量的说明…… 133
8.2.2 指针运算符 & …… 134
8.2.3 指针运算符 * …… 135
8.2.4 指针的初始化…… 135
8.2.5 空指针和悬挂指针…… 135
8.3 指针运算表达式 …… 137
8.3.1 指针的算术运算表达式…… 137
8.3.2 指针的关系运算表达式…… 138
8.3.3 指针的赋值运算表达式…… 139
8.3.4 混合运算中注意的问题…… 139
8.3.5 指针运算表达式的应用举例…… 140
8.4 指针与数组 …… 141
8.4.1 两种方法访问数组…… 141
8.4.2 用指针处理数组的例子…… 142
8.5 指针与字符串 …… 145
8.5.1 使用字符型指针处理字符串…… 145
8.5.2 使用字符型指针与字符数组处理字符串的区别…… 146
习题 …… 147

第九章 函数 …… 151

9.1 函数的概念 …… 151
9.2 函数的定义 …… 152
9.2.1 函数定义的一般格式…… 152
9.2.2 函数的返回…… 153
9.3 函数的调用 …… 154
9.3.1 函数调用的一般格式…… 154
9.3.2 函数的原型声明…… 156
9.4 函数参数的传递 …… 157
9.4.1 参数的“值”传递方式…… 158
9.4.2 参数的地址传递方式…… 159
9.4.3 利用参数返回结果…… 160
9.5 数组参数的传递 …… 161
9.5.1 向函数传递一维数组…… 161
9.5.2 向函数传递二维数组…… 165
9.6 字符串参数的传递 …… 166
9.7 局部变量与全局变量 …… 167
9.7.1 局部变量…… 168
9.7.2 全局变量…… 168
9.8 变量的存储类型 …… 170
9.8.1 变量的存储类型…… 170
9.8.2 不同存储类型变量初始化的规定…… 171
习题 …… 172

第十章 结构体 …… 178

10.1 结构体定义及结构体类型变量 …… 178
10.1.1 结构体定义 …… 178
10.1.2 结构体类型变量的说明…… 180
10.1.3 结构体类型变量的初始化 …… 182
10.1.4 结构体类型变量成员的引用 …… 183
10.2 结构体类型数组 …… 184
10.2.1 结构体类型数组的说明及初始化 …… 184
10.2.2 结构体类型数组的应用 …… 186
10.3 指向结构体类型数据的指针 …… 187
10.3.1 指向结构体类型变量的指针 …… 187
10.3.2 指向结构体类型数组的指针 …… 190
习题 …… 191

第十一章 文件 …… 194

11.1 文件的概述 …… 194
11.2 文件的打开与关闭 …… 195
11.3 文件的读写 …… 197
习题 …… 207

附录Ⅰ　ASCII 字符编码
一览表 …… 210
附录Ⅱ　关键字及其用途 …… 211
附录Ⅲ　运算符的优先级别和
结合方向 …… 211
附录Ⅳ　Turbo C 常用库函数 …… 212
参考文献 …… 218

第一章 概　述

要利用计算机处理实际问题，首先需要选择一门程序设计语言，然后编写出使计算机能够按照人的意愿工作的计算机程序。

C语言是优秀的结构化程序设计语言之一，已经广泛应用在各个领域中。为了有效地利用C语言进行程序设计，至少应该具有两方面的知识，即：

(1) 掌握C语言的语法规则。

(2) 掌握解决实际问题的方法和步骤，即算法。

本章首先介绍算法的概念和算法的描述方法，然后对C语言的特点、C程序的构成和书写格式以及上机步骤做了综述，为后续章节打下基础。

1.1 算法的概念及描述

人们学习C语言，是为了使用C语言编程序来解决实际问题。一个完整的程序应包括以下两方面的内容：

(1) 数据的描述。在程序中要指定数据的类型和数据的组织形式，即数据结构（data structure）。

(2) 操作的描述。即操作步骤，也就是算法（algorithm）。

数据是操作的对象，操作是对数据进行加工处理。数据和操作互相依存、共同作用得到期望的结果。因此，著名的计算机科学家Nikiklaus Wirth提出了一个公式

数据结构 + 算法 = 程序

作为程序设计人员，在编程之前，必须认真分析和设计程序的数据结构和算法。

1.1.1 算法的概念

人们无论做什么事情，必须按步骤循序渐进地进行。例如，学校的期末考试，首先，教务处统一安排考试时间、教室和监考教师，然后教师出考卷，接着教务处工作人员印出适量的试卷。考试开始前，监考教师取走试卷到指定教室组织学生进行考试，考试结束后，教师阅卷并给出成绩。这些步骤缺一不可，顺序错了也会产生严重的后果。因此，做任何事情之前，都必须考虑好实施的步骤，然后按部就班地进行，才能避免产生错误和混乱。

广义地说，为解决一个问题而采取的方法和步骤，就称为算法。

当然，同一个问题，可以有多种解决方法或操作步骤。例如：从北京到大连去旅游，可以选择不同的交通工具，如火车、汽车或飞机等。乘坐不同的交通工具，所花费的时间

和金钱是大不相同的。人们可以根据自己的实际情况选择最合适的交通工具。再如，求 1 +2+3+...+100，可以先算 1+2，再加 3，再加 4，一直加到 100，做 99 次加法运算，得到结果 5050；也可以用等差数列的求和公式：(1+100) ×100/2，做三步运算，同样得到结果 5050。从中可以看出，同一个问题，可以有多种算法，而算法不同，效率也不同。因此，我们不仅要保证算法的正确性，还要考虑算法的优劣，根据实际情况，选择合适的算法。

本书讨论的是计算机算法，所谓计算机算法，是指用计算机解决实际问题的方法和步骤。计算机算法分为两大类：数值运算算法和非数值运算算法。数值运算的目的是求解数值，如，求方程的根，求图形的面积等；非数值运算包含的范围很广，如人员管理、网上购物等。最初，计算机主要应用在数值运算领域，人们对算法的研究非常深入，对各种数值运算都有比较成熟的算法可供选用。而目前计算机更多的是用来解决非数值运算方面的问题。

1.1.2 结构化程序设计

目前，程序设计方法有两种，一种是面向对象的程序设计方法，另一种是面向过程的结构化的程序设计方法。本书只介绍结构化程序设计方法。因此，必须首先了解结构化程序设计的要求。

长时间以来，人们一直认为程序是给机器读的，只要程序没有错误，能够给出正确的结果就可以了。但随着计算机技术的迅速发展，程序的规模越来越大，人们发现读程序的时间可能比写程序的时间要长得多。不仅程序员需要读程序，测试人员和维护人员也需要读程序。如果程序很难读懂，也就无法进行相应的测试和维护。因此，衡量程序的质量不仅要看它的逻辑是否正确，性能是否满足要求，更主要的是看它是否容易阅读和理解，即程序的清晰性是第一位要考虑的。结构化程序设计是实现程序清晰易懂的关键技术。

1965 年，E.W.Dijkstra 提出了结构化程序设计的概念，1966 年 Bhm 和 Jacopini 证明了，只用三种基本的控制结构就能实现任何单入口单出口的程序。这三种基本控制结构是顺序结构、选择结构和循环结构。顺序结构是指各操作顺序执行，是最简单的一种基本结构。选择结构，也称为“分支结构”，先判断给定的条件是否成立，再决定执行哪种操作。循环结构，又称“重复结构”，即反复执行某一部分的操作。一个程序无论功能多么强大、逻辑多么复杂，都可以用这三种基本控制结构组合而成。

归纳起来，结构化程序设计主要有两个方面的含义：

(1) 使用顺序结构、选择结构和循环结构以及它们的组合来组织一个程序，各个结构之间只有一个入口和一个出口。

(2) 采用自顶向下逐步求精的方法编写程序。

依据结构化思想编写出来的程序称为“结构化程序”，结构化程序容易理解、容易维护、可靠性高。

1.1.3 算法的描述

在分析、设计算法的过程中，我们需要将算法描述出来。描述算法的工具很多，常用的有自然语言、程序流程图、N-S 图、PAD 图和伪代码等。

1. 用自然语言表示算法

自然语言就是人们日常使用的语言，如汉语、英语等等。

【例 1.1】 求 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10。

可以用最直接方法进行计算：

步骤 1：先求 1 + 2，得到结果 3。

步骤 2：将步骤 1 得到的和加上 3，得到结果 6。

步骤 3：将 6 再加上 4，得 10。

步骤 4：将 10 再加上 5，得 15。

… …

步骤 8：将上步得到的结果 36 再加上 9，得 45。

步骤 9：将 45 再加上 10，得 55。

这样的算法虽然正确，但太繁琐。如果要求 1 + 2 + 3 + … + 100，则要写 99 个步骤，显然是不可取的。而且每次都直接使用上一步的计算结果，不太方便。应该找一个通用的表示方法。

可以设两个变量，代表两个加数。直接将每一步骤的和放在其中一个加数变量中。现设 sum 和 i 分别代表两个加数，每一步骤的和也存放在 sum 中。用循环算法来求结果。改进的算法如下：

步骤 1：使 sum = 0

步骤 2：使 i = 1

步骤 3：使 sum + i，和仍然放在变量 sum 中，可表示为 sum + i→sum

步骤 4：使 i 的值加 1，即 i + 1→i

步骤 5：如果 i≤10，返回重新执行步骤 3、步骤 4 和步骤 5；否则，算法结束。

可以看出，步骤 3、4、5 组成一个循环，在实现算法时，要反复多次执行这三个步骤，直到执行步骤 5 时，经过判断，i 已超过规定的数值而不返回步骤 3 为止。此时算法结束，变量 sum 就是所求的结果。

用这种方法表示的算法具有通用性、灵活性。如果要计算 1 + 2 + 3 + … + 100，只需将步骤 5：i≤10 改成 i≤100 即可。

用自然语言描述算法通俗易懂，但具体实现时有较大差距，而且往往文字冗长，表示的含义不太严格，容易产生“二义性”。例如有这样一句话：“当 i 为 0，j 为 0 时要加 1。”，通过这句话无法判断要给哪个变量加 1，可能是 i，也可能是 j，或者是其它变量。因此，除了很简单的问题，一般不用自然语言表示算法。

2. 用程序流程图表示算法

程序流程图又称为程序框图，它是历史最悠久使用最广泛的算法描述工具，然而也是用的最混乱的一种方法。

图 1.1 列出了程序流程图中常用的图形符号。

起止框表示程序的开始和结束。

输入输出框表示程序输入数据或输出运行结果。

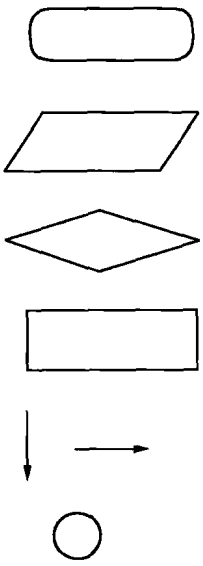

图 1.1　流程图的常用图形符号

处理框表示程序中的各种操作。

流程线用箭头表示，指出各框的执行顺序。

用菱形表示的判断框，是对一个给定的条件进行判断，根据给定的条件是否成立决定如何进行其后的操作。它有一个入口，两个出口。

连接点（小圆圈）是用于将画在不同地方的流程线连接起来。避免流程线的交叉或过长，使流程图更清晰。

用程序流程图表示算法时，必须考虑结构化程序设计的要求。图 1.2 显示了在程序流程图中，基本控制结构的表示方法。

顺序结构，如图 1.2 中（a）所示。A、B 两个操作是顺序执行的，先执行 A，再执行 B。

选择结构，分为两种：两个分支的选择结构（如图 1.2 中（b）所示）和多分支选择结构（如图 1.2（e）所示）。

两个分支的选择结构：首先判断条件 P 是否成立，如果成立，则执行 A 操作，否则，执行 B 操作。

多分支选择结构：根据条件 P 的取值，选择其中一个分支执行。

循环结构，分为当型（WHILE 型）循环结构（如图 1.2 中（c）所示）和直到型（UNTIL 型）循环结构（如图 1.2 中（d）所示）两种。

当型循环结构：首先判断条件 P 是否成立，如果成立，则执行 S 操作。执行 S 操作后，再判断条件 P 是否成立，如果仍然成立，再执行 S 操作，如此反复执行 S 操作，直到条件 P 不成立为止，退出循环结构。其中，P 称为循环条件，S 称为循环体（一组重复执行的操作）。

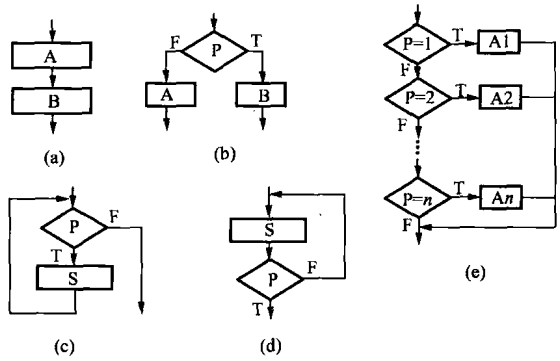

图 1.2　用流程图符号表示的基本控制结构

（a）顺序型；（b）选择型；（c）先判定型循环（DO－WHILE）；（d）后判定型循环（DO－UNTIL）；（e）多情况选择型（CASE 型）

直到型循环结构：首先执行 S 操作，然后判断条件 P 是否成立，如果不成立，则再执行 S 操作，然后再判断条件 P 是否成立，如果仍然不成立，再执行 S 操作，如此反复执行 S 操作，直到条件 P 成立为止，退出循环结构。同当型循环一样，P 称为循环条件，S 称为循环体。

【例 1.2】　将例 1.1 求 1＋2＋3＋4＋5＋6＋7＋8＋9＋10 的算法用程序流程图表示，如图 1.3 所示。

一个流程图包括：表示相应操作的框、带箭头的流程线和框内外必要的文字说明。其中，流程线的箭头表示各操作框的执行顺序。

用流程图表示算法形象直观、易于理解。但流程图也存在一些严重的缺点，如所使用的符号不够规范，常常使用一些习惯性用法，特别是对流程线的使用没有严格的限制，可以不受任何约束，随意转移控制。所以，使用程序流程图描述的算法不一定满足结构化程序设计的要求，往往难以理解，难以修改。许多人建议停止使用它，但至今仍在广泛使用。不过总的趋势是越来越多的人不再使用程序流程图了。

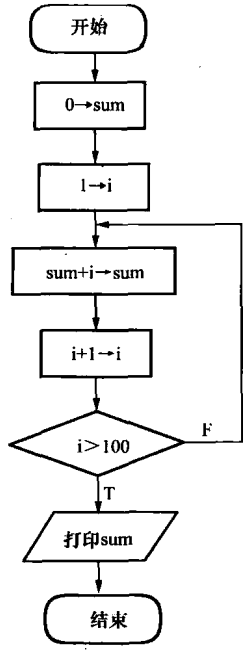

图 1.3　求 1+2+3+4+5+6+7+8+9+10 的流程图

3. 用 N-S 图表示算法

既然用基本结构的嵌套组合可以表示任何复杂的算法，那么基本结构之间的流程线就属多余了。1973 年，Nassi 和 Shneiderman 提出了一种结构化的流程图：N-S 流程图。在这种流程图中，完全去掉了带箭头的流程线，全部算法写在一个矩形框中，在该框内还可以包含其他从属于它的矩形框，即 N-S 图由矩形框组合嵌套而成，因此又称为盒图。图 1.4 列出了 N-S 图的五种基本控制结构的画法。

(a) 表示顺序结构，先执行 A，再执行 B。

(b) 表示两个分支的选择结构：首先判断条件 P 是否成立，如果成立，则执行 A 操作，否则，执行 B 操作。若 B 是空操作，则拉下一个箭头“↓”。

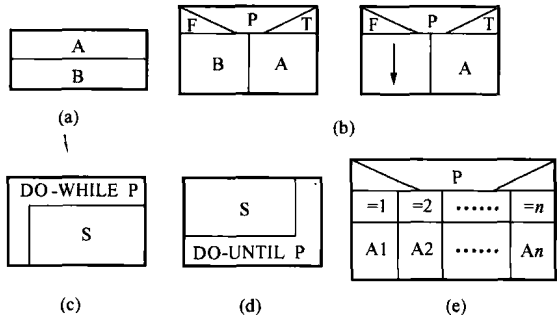

图 1.4　N-S 图的五种基本控制结构

(a) 顺序型；(b) 选择型；(c) while 重复型；(d) until 重复型；(e) 多分支选择型（CASE 型）

(c) 和 (d) 表示两种类型的循环结构：P 是循环条件，S 是循环体。其中，(c) 表示当型循环结构，先判断 P 是否成立，再执行 S 操作；(d) 表示直到型循环结构，执行 S 操作后，再判断条件 P 是否成立。

(e) 表示多分支选择结构：根据条件 P 的取值，相应地执行其值下面各框的内容。

【例 1.3】　将例 1.1 求 1+2+3+4+5+6+7+8+9+10 的算法用 N-S 图表示，如图 1.5 所示。

用 N-S 图描述算法结构清晰，容易理解，完全符合结构化程序设计的要求。

4. 用伪代码表示算法

使用程序流程图和 N-S 图表示算法，清晰易懂，但如果要修改，工作量会很大。通常，在设计算法的过程中，需要反复修改，不断完善。因此，流程图适合表示一个最终的

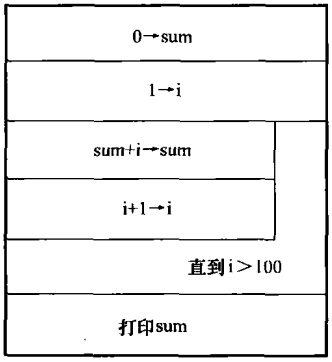

图 1.5　用 N-S 图表示求 1+2+3+4+5+6+7+8+9+10 的算法

算法，而在设计算法过程中使用却不太理想。为了设计算法时方便，经常使用伪代码作为描述工具。

伪代码是一种介于自然语言和计算机语言之间的，用来描述算法的文字和符号。它是在自然语言的基础上加了一些限制而得到的。语言的正文用基本控制结构进行分割，具体操作用自然语言来表示。它如同一篇文章，自上而下逐行写下来，每一行（或几行）表示一个操作。

【例 1.4】　将例 1.1 求 1+2+3+4+5+6+7+8+9+10 的算法用伪代码表示。

```
sum 赋初值为 0
for (i = 1; i < = 10; i + + )
    使 sum = sum + i
打印 sum 的值
```

从上例可以看出，伪代码书写格式比较自由，容易修改，但伪代码表示的算法没有流程图直观。

5. 用计算机语言表示算法

要完成一项工作，不仅要设计算法，还要实现算法。例如：我们已经清楚求 1+2+3+4+5+6+7+8+9+10 的算法，但并没有真正求出确切的结果。

实现算法的方法很多，如可以用心算、笔算、算盘或计算器求出结果，但我们的任务是用计算机实现算法。计算机是无法识别流程图和伪代码的，只有计算机语言编写的程序才能被计算机编译执行。因此，描述一个算法后，还要将它转换成计算机语言程序。

【例 1.5】　将例 1.1 求 1+2+3+4+5+6+7+8+9+10 的算法用 C 语言表示。

```
main ()
{int i, sum;
sum = 0;
for (i = 1; i < = 10; i + + )
   sum = sum + i;
printf ( "%d", sum);
}
```

用计算机语言表示算法必须严格遵循所用语言的语法规则，这是和伪代码不同的。

1.2　C 语言的特点

C 语言诞生于 1972 年，由美国电话电报公司（AT&T）贝尔实验室设计。经过不断改进，1978 年后，C 语言已先后被移植到大、中、小及微机上。同时由 B.W.Kernighan 和 D.M.Ritchit 合著了著名的“THE C PROGRAMMING LANGUAGE”一书，通常简称为《K&R》，也有人称之为《K&R》标准。但是，在《K&R》中并没有定义一个完整的标准 C

语言。1983 年，美国国家标准化协会在此基础上制定了一个 C 语言标准，称为 ANSI C。目前流行的 C 编译系统都是以它为基础的。

C 语言是面向过程的计算机程序设计语言，它既具有高级语言的优点，又具有低级语言的特点，功能强大，使用灵活方便，非常适合编写系统软件。同其他计算机语言相比，C 语言具有如下的特点：

（1）语言简练，使用灵活方便。ANSI C 一共只有 32 个关键字（见附录 I），9 种控制语句，程序书写格式自由，主要用小写字母表示，压缩了一切不必要的成分。

（2）运算符丰富。C 的运算符共有 34 种，把括号、赋值、逗号等都作为运算符处理。应用这些运算符可以构成形式多样的表达式，能够实现其他高级语言难以实现的运算。

（3）数据结构类型丰富。C 语言除了具有整型、实型、字符型等基本数据类型之外，还有数组、指针、结构体、共用体等多种构造数据类型。

（4）C 语言是结构化的程序设计语言。具有结构化的控制语句，可以很容易地实现各种基本控制结构；用函数作为程序模块，实现了模块化。符合现代编程风格的要求。

（5）语法限制不太严格，程序设计自由度大。

（6）C 语言允许直接访问物理地址，能进行位（bit）运算，能实现汇编语言的大部分功能，可以直接对硬件进行操作。因此有人把它称为中级语言。

（7）与其它高级语言相比，C 语言生成的目标代码质量好，程序执行效率高；与汇编语言相比，用 C 语言编写的程序可移植性好。

总之，C 语言具有很多优于其它语言的特点，适用于很多领域，已经得到广泛的应用。但是，C 语言对程序员要求也高，程序员用 C 写程序会感到限制少、灵活性大，功能强，但较其他高级语言在学习上要困难一些。

1.3 C 程序的构成和书写格式

1.3.1 C 程序的构成

为了说明 C 语言源程序的结构特点，先看以下几个程序。虽然有关内容还未介绍，但可从这些例子中了解到 C 源程序的基本组成和书写格式。

【例 1.6】 计算两个整数的和。

```
main ()                          /* 主函数 */
{
int x, y, sum;                   /* 定义变量 */
x = 10;                          /* 给变量赋值 */
y = 20;
sum = x + y;                     /* 求和 */
printf ( "The sum is %d", sum);  /* 输出结果 */
}
```

这个程序是求 x 和 y 的和 sum 并将结果输出到屏幕上。运行结果是：

The sum is 30

其中，main（）是主函数，表示 C 程序从这里开始执行。每一个 C 源程序都必须有且只能有一个主函数（main 函数），函数体由大括号 {} 括起来。{} 内的部分（第 3～7 行）称为函数体。

/* …… */表示注释，“…… ……”是注释的具体内容，用汉字、拼音、英文来写均可。注释可以加在程序中任意位置，可以单独占一行，也可以加在程序语句的后面，注释是用来帮助人们阅读程序的，不参与也不影响程序的运行。

第 3 行是变量定义，声明了三个整型变量。

第 4，5 行是两个赋值语句，分别给变量 x 和 y 赋值为 10 和 20。

第 6 行将 x 和 y 的和送入变量 sum 中。

第 7 行是调用系统输出函数 printf，按规定格式输出运算结果。其中 %d 表示在这个位置上将用一个整型的数值代替，这个程序中是用 sum 的值 30 来代替。

【例 1.7】 任意输入两个数，将两个数中最大的一个输出。

```
main ()                                  /* 主函数 */
{
int x, y, z;                             /* 定义变量 */
scanf ( "%d,%d", &x, &y);                /* 从键盘输入两个数 */
z = max (x, y);                          /* 调用 max 函数，求最大的数赋给 z */
printf ( "The Max number is %d", z);     /* 输出 z 的值 */
}
int max (int x, int y) /* 定义 max 函数，函数值为整型，形式参数 x，y 为整型 */
{
int z;
if (x > y) z = x;                        /* 如果 x 大于 y 就将 x 的值赋给 z */
else z = y;                              /* 否则就将 y 的值赋给 z */
return (z);                              /* 将 z 的值返回，通过 max 带回调用处 */
}
```

程序运行时首先要输入两个整数：

-30，29↙　　（注：“↙”表示输入数据后回车）

运行结果是：

The Max number is 29

这个程序包括两个函数，一个是主函数，另一个是求最大值的 max 函数。

主函数中首先定义了 3 个整型变量，然后调用系统输入函数 scanf 输入变量 x，y 的值。在执行时，先由 scanf 函数从键盘读取两个数字，这时由用户从键盘上输入 -30，29 并回车。此时 x 被赋值为 -30，y 被赋值为 29。第五行调用 max 函数，将 x 和 y 的值传入 max 函数中，在 max 函数中经过判断后，将 x 和 y 中的最大值赋给 z，用 return 语句将 z 的值作为函数的返回值带回主函数，并赋值给变量 z。第六行，将变量 z 的值输出到屏幕上。

本例所涉及的函数调用、参数传递等概念，只需简单了解即可，后续章节会详细阐述。主要目的是通过此例来了解 C 程序的组成和形式。

通过以上两例，不难看出：

(1) C 程序是由一个或多个函数构成的。一个程序至少包含 main () 函数，用户也可以根据需要设计自己的函数，如【例 1.7】中的 max 函数。此外，C 语言还提供了丰富的标准库函数。利用这些现成的函数，我们可以非常轻松的编写功能强大的程序。可以说 C 是函数式的语言，程序中的全部工作都是由各个函数分工合作完成的。

(2) 一个函数由函数说明和函数体两大部分组成。

函数的说明部分：又称函数首部，包括函数名、函数的返回值类型、函数的参数及类型。

例如，【例 1.7】中的 max 函数的首部为：

int	max	(int x, int y)
函数的返回值类型	函数名	函数的参数及类型

一个函数名后面必须有一对圆括号 ()，括号内可以有也可以没有函数参数，如 main ()。

函数体部分：函数首部下面的最外层花括弧 {……} 内的部分称为函数体，由若干变量定义语句和可执行语句构成。

(3) C 程序总是从 main () 函数开始执行，不论 main () 函数在程序的什么地方，也就是说，可以将 main () 函数放在程序的任何位置。

(4) C 语言中没有专门的输入、输出语句，输入输出是通过调用 scanf、printf 等库函数实现的。

1.3.2 C 程序的书写格式

C 语言书写格式比较灵活自由，没有太多的限制，但必须考虑程序的清晰性。从结构清晰，便于理解的角度出发，在书写程序时，应遵循以下规则：

(1) 可以在一行上写若干语句，一个语句也可以分写在多行上。但是在有些情况下，语句中的某些部分不能随意断开。

但为了提高程序的可读性，最好一行只写一个语句。

(2) 每个语句的后面都要有一个分号，这个分号是必不可少的，即使是程序的最后一个语句，也要加上分号。

(3) 低一层次的语句或说明可比高一层次的语句或说明缩进若干格后书写。以便看起来更加清晰，增加程序的可读性。见下例：

```
main ( )
{int sum = 0, i = 1;
while (i < = 100)
    { sum + = i;
      i + + ;
    }
```