第 1 章　初识 C 语言

- 熟练掌握 C 语言开发环境 CodeBlocks 的使用
- 掌握运行 C 程序的基本步骤
- 了解 C 语言的历史及特点
- 熟悉算法的表示形式

1.1　C 语言概述

C 语言是国际上广泛流行的计算机高级语言，C 语言的祖先是 BCPL（Basic Combined Programming Language）语言。1970 年，美国 AT&T 贝尔实验室的 Ken Thompson 以 BCPL 为基础，设计出了简单且很接近硬件的 B 语言（取 BCPL 的第一个字母），但 B 语言过于简单，功能有限。1972 年至 1973 年间，美国贝尔实验室的 D.M. Ritchie 在 B 语言的基础上设计了 C 语言。C 语言既保持了 BCPL 和 B 语言的优点（接近硬件），又克服了它们的缺点（过于简单、无数据类型等）。最初的 C 语言只是为描述和实现 UNIX 操作系统提供一种工作语言而设计的。1973 年，Ken Thompson 和 D.M. Ritchie 合作把 UNIX 的 90%以上用 C 语言改写，即 UNIX 第 5 版。1978 年，Brian W.Kernighan 和 Dennis M.Ritchie 合著了影响深远的名著 *The C Programming Language*，它是第一个 C 语言标准。1978 年以后，C 语言先后被移植到大、中、小和微型计算机上。

C 语言是一种应用范围广泛，既可以用来编写系统应用程序，也可以用来编写不依赖计算机硬件的应用程序。C 语言问世后发展迅速，是目前最受欢迎的编程语言之一。

C 语言的主要特点如下：

① C 语言简洁、使用方便；源程序短，编辑程序时工作量小。

② C 语言可以对硬件编程，可以像汇编语言一样对位、字节和地址进行操作，在单片机和嵌入式系统中应用广泛。

③ C 语言是以函数形式提供给用户的，这些函数可方便调用，并具有多种循环、条件控制结构，使程序完全结构化。

④ C 语言具有各种数据类型和运算符，并引入指针概念，程序执行效率高。

⑤ 用 C 语言编写的程序可移植性好，适合多种操作系统、多种机型。

⑥ 生成目标代码质量高，程序执行效率高。

1.2　C 语言开发环境

C 语言开发环境有很多种，读者可根据需要选择 C 语言的开发环境，如 Visual C++ 6.0、

DEV C++等。本书采用 CodeBlocks 作为 C 语言的开发工具，CodeBlocks 是一个开源的全功能跨平台 C/C++集成开发环境，支持 Windows 和 GNU/Linux。

1.2.1 运行 C 语言程序的步骤和方法

要编辑一个 C 源程序，并通过 C 语言编程环境 CodeBlocks 进行编译、运行，一般要经过以下步骤，具体过程如图 1-1 所示。

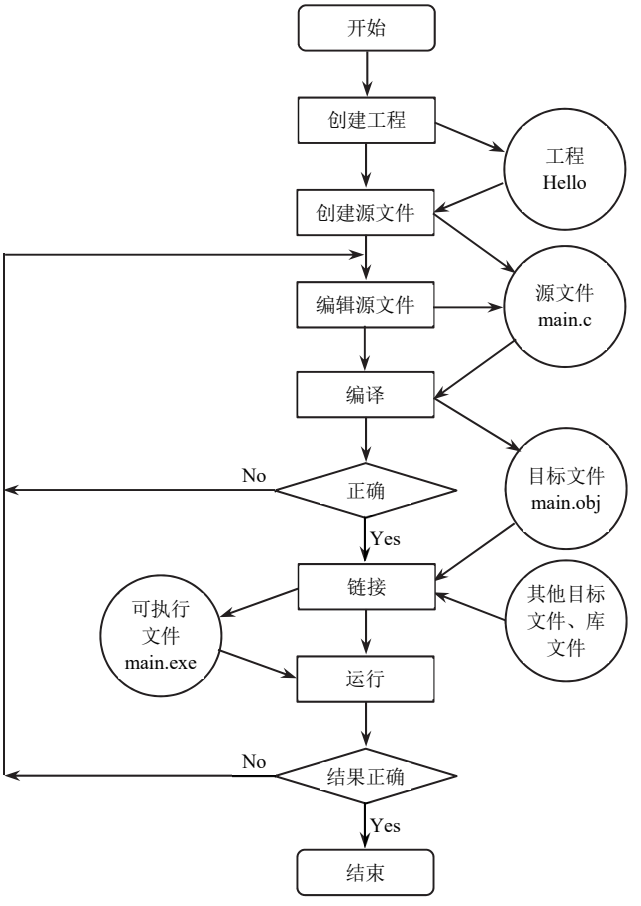

图 1-1 运行 C 程序的步骤和方法

（1）创建工程

通过创建一个工程为一个 C 程序提供工作环境。

（2）向工程添加源文件

一个工程可包含一个或多个源文件，可包含零或多个头文件。

（3）编辑源文件

根据 C 语言的语法规则，用文本编辑器编写的扩展名为 .c 的文件，如 first.c、hello.c 等。

（4）编译

计算机不能直接识别源文件，必须把源文件转化为计算机能够识别的机器指令。C 程序编译器将检查源文件中是否有语法错误，如果有语法错误，将提示有关错误，如果没有语法错误，

编译器会将源文件转化为一个二进制文件，该二进制文件被称为源文件的目标文件。目标文件的名字与源文件的名字相同，但扩展名为 .obj。

（5）链接

目标文件是供链接器使用的文件，也就是说，目标文件中含有待确定的链接信息，链接器必须把这些信息替换成真正的链接代码、形成完整的可执行的代码，即链接器负责产生一个可执行文件。可执行文件的名字与源文件的相同，但扩展名为 .exe。

（6）运行

将生成的可执行文件交给操作系统去执行。

1.2.2 最简单的 C 语言程序

学习并掌握 C 程序，首先要熟练使用 C 程序的开发环境，能用 CodeBlocks 编写简单的 C 源程序，并对源程序进行编译、链接和运行。

【例 1-1】 编写一个简单的程序，要求程序输出文字“Hello，C 程序设计——增量式项目驱动一体化教程！”。

（1）代码实现（chp1_1.c）

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    printf("Hello，C 程序设计——增量式项目驱动一体化教程！");
    getchar();
    return 0;
}
```

（2）运行结果

```
Hello，C程序设计——增量式项目驱动一体化教程!
```

（3）总结

一个 C 语言程序的结构具有以下特点。

① 一个程序由一个或多个源程序文件组成。一个源文件又包含三部分：

❖ 预处理命令。例如：

```
#include <stdio.h>
```

❖ 全局声明。例如，全局变量声明

```
int  num;
```

❖ 函数定义。例如，定义一个 max()函数，用来计算两个数的最大值。

② 函数是 C 程序的主要组成部分。一个 C 语言程序是由一个或多个函数组成的，其中必须且只能有一个 main()函数。

③ 一个函数包含如下两部分。

❖ 函数首部：包括函数类型、函数名、函数参数名、参数类型。

❖ 函数体：函数首部下面{ }中的部分。函数体又包含两部分：声明部分和执行部分。

④ 程序总是从 main()函数开始执行。

⑤ 程序对计算机的操作是由函数中的 C 语句完成的。

⑥ 每条语句结尾由“;”结束。

1.3 算法

通过以上内容的学习，可以发现，一个程序主要包括两方面的信息：

① 对数据的描述。程序中需要使用什么样的数据来描述具体问题，数据的类型、数据的组织形式分别如何表示，这就是数据结构（Data Structure）。

② 对操作的描述。程序中对数据进行什么样的处理，即要求计算机进行操作的步骤，这就是算法。

1.3.1 算法的定义

广义的算法是指“为解决一个问题而采取的方法和步骤”，也就是程序。计算机算法就是为了解决一个问题，计算机所需要执行的方法和步骤，也就是计算机程序。

在软件行业，程序的概念还要广一些，既包括算法，也包括算法操作的对象，即数据。数据是指所有能输入到计算机并被计算机程序处理的符号的介质的总称，各种字母、数字符号的组合、语音、图形、图像等统称为数据。数据经过加工后就成为信息，可由计算机进行处理。

算法与数据的关系是，算法操作的对象是数据。在同一应用环境下，不同的数据之间存在一定的联系，这些数据的组织形式就是数据结构。简单来说，程序 = 算法 + 数据结构。

对同一个问题，可能有不同的解题方法和步骤，不同的方法之间有优劣之分，有的方法较简单，有的较复杂，一般采用方法简单、运算步骤较少的方法。

计算机算法分为两大类：数值计算算法和非数值计算算法。数值计算问题一般可通过数学运算进行解决，如求方程的根、求微分方程等；非数值计算问题一般不能直接解决，要通过建立数学模型，设计合适的算法来解决，如图 1-2 所示。

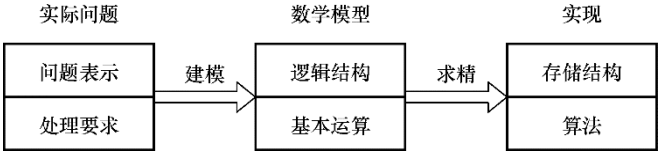

图 1-2 解决问题的步骤

1.3.2 算法的表示

常用的表示算法的方法有：自然语言、流程图、结构化（N/S）流程图和伪代码等。下面对每种方法进行简单介绍。

（1）自然语言

自然语言就是人们日常使用的语言，可以是汉语、英语或其他语言。用自然语言表示的算法通俗易懂，但是文字冗长、容易有歧义。自然语言表示的算法不太严格，要根据上下文才能判断其正确含义。用自然语言描述包含分支和循环结构的算法不太方便，因此，除了一些简单的问题，一般不用自然语言表示算法。

【例 1-2】 计算 1+ 2+ 3+ 4+ 5。

用自然语言描述算法。

步骤 1：先计算 1+ 2，得到结果 3。

步骤 2：将步骤 1 得到的结果再加上 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。

（2）流程图

流程图是用一些框图来表示各种操作，用图形表示算法，会更直观、易于理解。一些常用的流程图符号如图 1-3 所示。

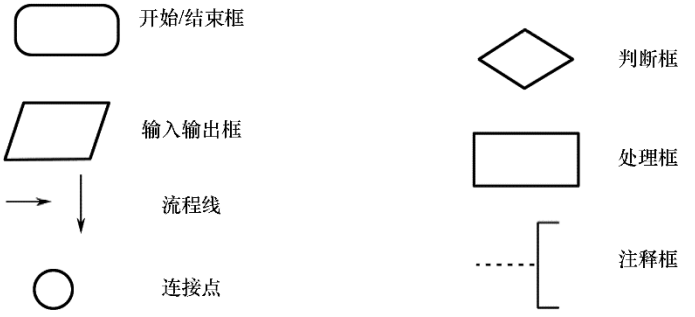

图 1-3　流程图符号

【例 1-3】 判断 x 是否为正数，画出解决该问题的流程图如图 1-4 所示。

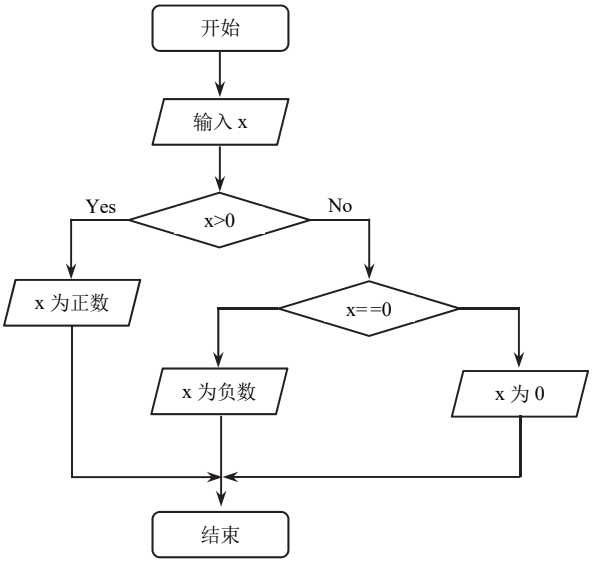

图 1-4　例 1-3 流程图

（3）N-S 结构化流程图

N-S 流程图用以下流程图符号来描述：

顺序结构用图 1-5 所示形式表示，A、B 两个框组成一个顺序结构。

选择结构用图 1-6 所示形式表示，P 表示条件，P 条件成立时执行 A 操作，否则执行 B 操作。

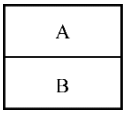

图 1-5　顺序结构

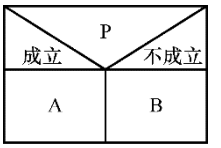

图 1-6　选择结构

循环结构用图 1-7 和图 1-8 所示的两种形式表示。图 1-7 为当型循环结构，当 P1 条件成立时反复执行 A 操作，直到 P1 条件不成立结束。图 1-8 为直到型循环结构，反复执行 A 操作，直到条件 P1 成立结束。

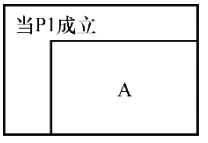

图 1-7　当型循环

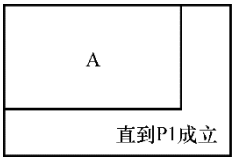

图 1-8　直到型循环

在图 1-5～图 1-8 中，A、B 可以是一个简单的操作，也可以是以上 3 种基本结构之一。

【例 1-4】 计算 5!，画出解决该问题的 N-S 流程图如图 1-9 所示。

图 1-9　例 1-4 的 N-S 流程图

（4）伪代码

伪代码是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。一般每一行（或几行）表示一个基本操作，伪代码不使用图形符号，书写方便，格式紧凑，容易看懂，便于向计算机程序过渡。用伪代码编写的算法并无固定的、严格的语法规则，可以使用任何语言，只要意思表达清楚，便于书写和阅读即可。

【例 1-5】 计算 5!，用伪代码表示该算法如下：

```
begin
    t:=1
    i:=2
    while i≤5
    {
        t:=t*i
        i:=i+1
    }
    print t
end
```

1.3.3 算法举例

【例 1-6】 计算两个整数 x 和 y 的最大值。

（1）问题分析

如果 x≥y，则 x 是较大者；否则，y 是较大者。

（2）流程图（如图 1-10 所示）

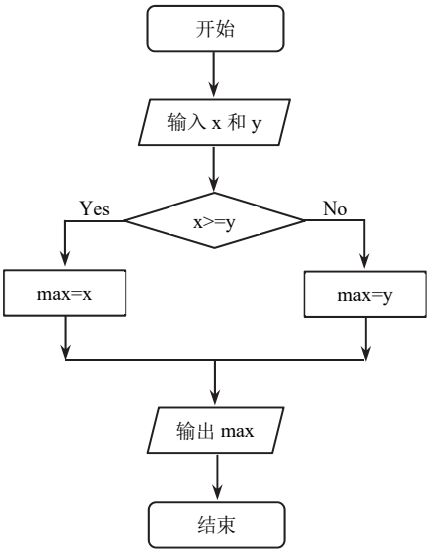

图 1-10　例 1-6 流程图

（3）代码实现（chp1_6.c）

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int x, y;
    int max;
    printf("输入 x 和 y 的值: \n");
    scanf("%d,%d", &x, &y);
    if(x>=y)                                    // 判断 x 与 y 的大小，最大值赋给 max
    {
        max=x;
    }
    else
    {
        max=y;
    }
    printf("较大的数是%d \n", max);             // 输出较大值
    return 0;
}
```

【例 1-7】 判断某一年是否为闰年，并将结果输出。

（1）问题分析

某年 x 为闰年的条件是：x 能被 4 整除并且不能被 100 整除，或者 x 能被 100 整除并且能被 400 整除。

（2）流程图（如图 1-11 所示）

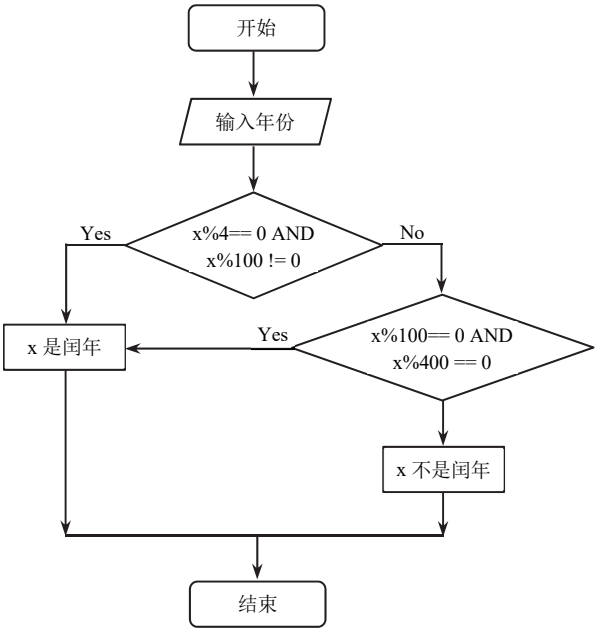

图 1-11　例 1-7 流程图

（3）代码实现（chp1_7.c）

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int year;                                    // 定义变量
    printf("输入年份:\n");
    scanf("%d", &year);
    // 判断是否是闰年的条件
    if((year%4==0 && year%100!=0) || (year%100==0 && year%400==0))
    {
        printf("%d 是闰年:  \n", year);
    }
    else
    {
        printf("%d 不是闰年:  \n", year);
    }
    return 0;
}
```

【例 1-8】 有 50 个学生，要求输出成绩在 80 分以上的学生的学号和成绩。

（1）问题分析

① 定义表示学生信息（学号和成绩）的数据类型。

② 定义长度为 50 的数组。

③ 依次取出每个学生的成绩，利用下面的循环进行处理，直到所有学生处理完为止：如果该学生成绩≥80，则输出该学生学号和成绩，然后取下一个学生的成绩

（2）流程图（如图 1-12 所示）

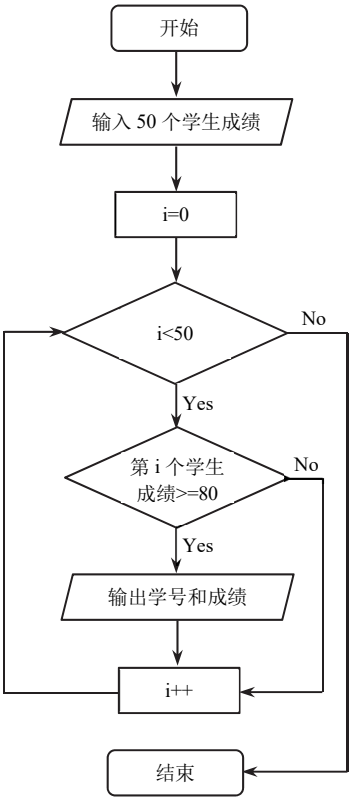

图 1-12　例 1-8 流程图

（3）代码实现（chp1_8.c）

```
#include <stdio.h>
#include <stdlib.h>
struct student
{
    long number;
    float score;
} stu[50];
int main()
{
    int   i;                                    // 定义变量
    for(i=0; i<50; i++)                         // 循环 50 次，输入 Number 学号和 score 成绩
    {
        printf("number 学号，score 成绩:\n");
        scanf("%ld,%f", &(stu[i].number), &(stu[i].score));
    }
    // for 循环判断输入的成绩是否大于等于 80
    for(i=0; i<50; i++)
```

```
    {
        if (stu[i].score>=80)
            printf("学号为%ld 的学生的成绩= %f 分\n", stu[i].number, stu[i].score);
    }
    return 0;
}
```

本章小结

本章介绍了 C 语言的发展史及特点；介绍了 C 语言的开发环境以及运行 C 程序的步骤和方法；介绍了算法的概念及表示形式。要求读者能熟练使用 C 语言开发环境，并能编写简单的 C 程序。

习 题 1

1. 了解 C 语言的发展历史。
2. 了解 C 语言的特点。
3. 写出 C 程序开发环境 CodeBlocks 的下载、安装及使用步骤。
4. 写出运行 C 程序的步骤与方法。
5. 什么是算法？一个算法有哪些表示形式？
6. 用流程图表示求解以下问题的算法：

（1）计算 1+2+…+100。

（2）判断一个数是否为偶数。

（3）计算一元二次方程 $ax^2+bx+c=0$。

7. 编写一个 C 程序，输出以下信息：

```
**********************************
    Hello, My name is Niuniu!
**********************************
```

8. 编写一个 C 程序，输入 a、b、c 三个值，输出其中最大者。