第 3 章

顺序结构程序设计

Chapter 3

程序设计是指设计、编制、调试程序的方法和过程，是软件构造活动中的重要组成部分。常用的程序设计方法有面向过程的结构化程序设计和面向对象的程序设计等，而 C 语言是一种典型的结构化程序设计语言。它层次清晰，便于按模块化方式组织程序，易于调试和维护。而顺序结构是 C 程序中最简单、最基本、最常用的一种程序结构，也是进行复杂程序设计的基础。顺序结构程序设计中使用较多的有赋值语句和函数调用语句，特别是输入、输出函数调用语句使用较为频繁。

学习目标	(1) 了解程序设计的一些基础知识。 (2) 理解 C 程序语句的使用方法。 (3) 掌握数据的输入函数和输出函数。 (4) 掌握顺序结构程序设计的基本思想和设计方法。

3.1 程序设计基础

程序是指为完成某项活动或过程所规定的途径。对于计算机来说，程序就是为实现特定目标或解决特定问题而用计算机语言编写的指令序列。计算机按照程序中的指令逐条执行就可以完成相应的操作。著名的计算机科学家沃思(Nikiklaus Wirth)曾提出一个公式：程序＝数据结构＋算法。他认为程序是由数据结构和算法这两个关键成分组成，而语句是算法实现的程序表示，是算法实现的最小单位。

本节学习目标：

- 了解程序设计的一般过程。
- 领会结构化程序设计的基本思想。
- 熟悉算法的概念及算法的描述方法。
- 掌握 C 语言的语句。

【任务提出】

任务 3.1：从键盘上输入一个矩形的长为 a 和宽为 b 的值，画出计算矩形面积 s 的传统流程图和 N-S 图。

【任务分析】

本任务要求画出计算矩形面积 s 的程序流程图。首先要对任务进行分析，确定输入量和输出量；随后找出解决问题的办法，即用面积公式求出矩形面积；最后分别用两种流程图中规定的符号将解决问题的思路描述出来。

【任务实现】

根据任务分析，传统流程图和 N-S 流程图如图 3.1 所示。

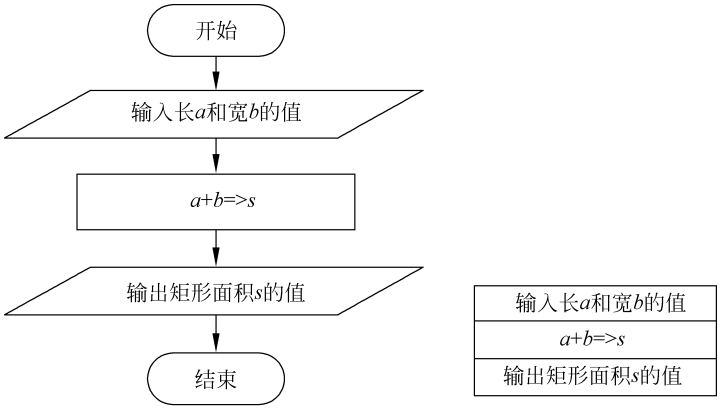

图 3.1　两种流程图的描述

【知识讲解】

1. 程序设计

程序设计是以某种程序设计语言为工具，给出为解决某种问题的程序。它是指从分析实际问题开始到计算机给出正确结果的整个过程，也就是通常所说的“编程”。如图 3.2 所示，其一般可包括以下几个步骤。

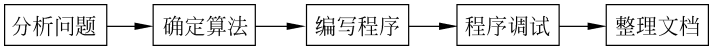

图 3.2　程序设计过程

(1) 分析问题

根据任务提出的要求，分析解决问题的方案，明确输入数据和输出结果，确定存放数据的数据结构。

(2) 确定算法

针对存放数据的数据结构来确定解决问题、完成任务的步骤。

(3) 编写程序

根据确定的数据结构和算法，使用选定的计算机语言编写程序代码，输入到计算机并保存在磁盘上，简称编程。

(4) 程序调试

消除由于疏忽而引起的语法错误或逻辑错误；用各种可能的输入数据对程序进行测试，使之对各种合理的数据都能得到正确的结果，对不合理的数据能进行适当的处理。

(5) 整理并写出文档资料

对文档资料进行整理并写出。

2. 结构化程序设计思想

结构化程序设计的概念最早由 E. W. Dijikstra 在 1965 年提出，是软件发展的一个重要里

程碑。它的主要观点是采用自顶向下、逐步细化、模块化设计、结构化编码的程序设计方法。结构化程序设计思想要求程序只能用三种基本结构来描述，复杂的程序可以用这三种基本结构组合而成。这三种基本结构就是顺序结构、选择结构和循环结构。

(1) 顺序结构

顺序结构就是一组逐条执行的可执行语句，按照书写顺序自上而下执行。顺序结构是最简单的一种结构。

(2) 选择结构

选择结构又称条件结构或分支结构，是一种先对给定条件进行判断，然后根据判断的结果执行相应命令的结构。

(3) 循环结构

循环结构又称重复结构，是指多次重复执行同一组命令的结构。在循环结构中最主要的问题是：什么情况下执行循环？哪些操作需要循环执行？循环结构的基本形式有两种：当型循环和直到型循环。

已经证明，由以上三种基本结构顺序组成的算法结构可以解决任何复杂的问题。由基本结构所构成的算法属于“结构化”算法，它不存在无规律的转向，只在一个基本结构内才允许存在分支和向前或向后的跳转。

3. 算法及算法表示

算法是为解决一个问题而采取的方法和步骤，是程序设计步骤中的重要环节。算法的表示方法有多种，如自然语言表示法、传统流程图表示法、N-S 流程图表示法、伪代码表示法及计算机语言表示法等，下面主要对传统流程图及 N-S 流程图两种表示方法进行介绍。

(1) 传统流程图表示法

传统流程图是一种使用很广的方法，它使用一些约定的几何图形来描述算法，直观形象，易于理解。美国标准化协会 ANSI 规定了一些常用的流程图符号，如图 3.3 所示，这些符号已被各国程序工作者采用。

(a) 起始框　(b) 处理框　(c) 输入/输出框　(d) 判断框　(e) 连接点　(f) 流程线

图 3.3　常用的流程图符号

(a) 起始框。表示算法的开始和结束。一般内部只写“开始”或“结束”。

(b) 处理框。表示算法的某个处理步骤，一般内部常常填写赋值操作。

(c) 输入/输出框。表示算法请求输入/输出需要的数据或算法将某些结果输出。一般内部常常填写“输入……”“打印/显示……”等。

(d) 判断框。主要是对一个给定条件进行判断，根据给定的条件是否成立来决定如何执行其后的操作。它有一个入口和两个出口。

(e) 连接点。用于将画在不同地方的流程线连接起来。同一个编号的点是相互连接在一起的，实际上同一个编号的点是同一个点，只是画不下才分开画。

用流程图来表示三种基本结构如下。

① 顺序结构。如图 3.4 所示，A 和 B 两个框是顺序执行的。即在执行完 A 框所指定的操作后，必然接着执行 B 框所指定的操作。

② 选择结构。如图 3.5 所示，选择结构必包含一个判断框，根据给定的条件 P 是否成立来进行选择。若 P 条件成立，则执行 A 框中的操作；否则，执行 B 框中的操作。

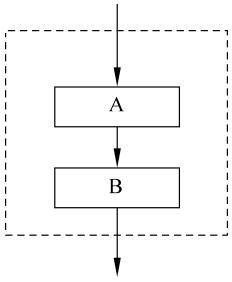

图 3.4　顺序结构图

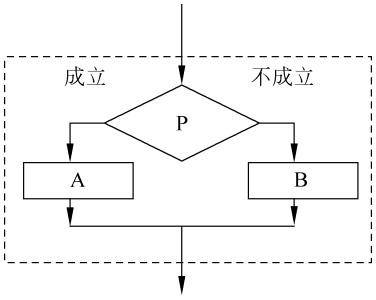

图 3.5　选择结构图

③ 循环结构。当型循环如图 3.6 所示，此结构表示当给定条件 P 成立时，反复执行 A 操作，直到条件 P 不成立为止，跳出循环。

直到型循环如图 3.7 所示，此结构表示先执行 A 操作，再判断 P 条件是否成立，如条件 P 不成立时，则继续执行 A 操作，再判断 P 条件是否成立，直到条件 P 成立为止，然后跳出循环。

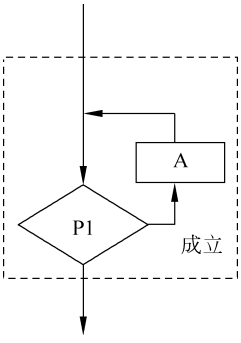

图 3.6　当型循环图

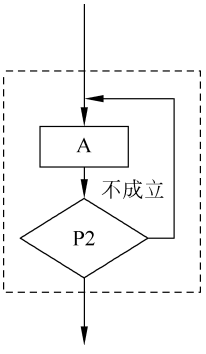

图 3.7　直到型循环图

用传统流程图表示算法比较直观形象，易于理解，能将设计者的思路清楚地表达出来。但是，这种流程图占用篇幅较多，尤其当算法比较复杂时，流程图之间的连线会使结构的清晰度变差。为此，人们设计了一种新的流程图——N-S 图。

(2) N-S 流程图

N-S 流程图将整个算法写在一个大框图内，这个大框图由若干个小的基本框图构成，基本框图符号如图 3.8～图 3.11 所示。在这种流程图中，完全去掉了带箭头的流程线。

① 顺序结构。如图 3.8 所示。A 和 B 两个框组成一个顺序结构。

② 选择结构。如图 3.9 所示。当 P 条件成立时执行 A 操作，P 不成立则执行 B 操作。请注意图 3.9 是一个整体，代表一个基本结构。

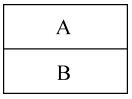

图 3.8　顺序结构 N-S 图

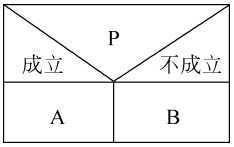

图 3.9　选择结构 N-S 图

③ 循环结构。当型循环结构如图 3.10 所示。图 3.10 表示当 P1 条件成立时反复执行 A 操作，直到 P1 条件不成立为止。直到型循环结构如图 3.11 所示。

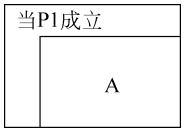

图 3.10　当型循环结构 N-S 图

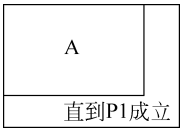

图 3.11　直到型循环结构 N-S 图

用以上三种结构的 N-S 流程图基本框可以组成复杂的 N-S 流程图，以表示算法。应当说明的是，在上述各图中的 A 框或 B 框，可以是一个简单的操作（如读入数据或打印输出等），也可以是 3 个基本结构之一。

用 N-S 图表示的算法比传统流程图紧凑易画，尤其是它废除了流程线，整个算法结构是由各个基本结构按顺序组成的，如同一个多层的盒子，又称盒图。

4. C 语言的语句

语句是算法实现的程序表示，是实现程序功能的最小单位。程序中的语句可分为表达式语句、函数调用语句、控制语句、复合语句和空语句五类。

(1) 表达式语句

表达式语句是由一个表达式加上分号“;”构成。

表达式语句的一般形式如下：

```
表达式;
```

执行表达式语句就是计算表达式的值。

如果构成该语句的表达式是赋值表达式，则该表达式语句又称为赋值语句。这是表达式语句最典型、使用最频繁的一种形式。例如：

```
x = y + z; x = 3;
```

需要注意的是赋值表达式和赋值语句的区别。赋值表达式是一种表达式，它可以出现在任何允许表达式出现的地方，而赋值语句则不能。

下述语句是合法的。

```
c = ((a = b + 3)> 0)?a:b;
```

语句的功能是，若 a＝b＋3 大于 0，则 c＝a，否则 c＝b。

而下述语句是非法的。

```
c = ((a = b + 3;)> 0)?a:b;
```

因为“a＝b＋3;”是语句，不能出现在表达式中。

(2) 函数调用语句

函数调用语句由一次函数调用加上分号“;”组成。

函数调用语句的一般形式如下：

```
函数名(实际参数表);
```

执行函数调用语句就是调用函数体并把实际参数赋予函数定义中的形式参数，然后执行被调函数体中的语句，求取函数值(在后面函数章节中将详细介绍)。例如：

```
printf("This is my first C program.\n");
```

该语句的作用是调用库函数，输出字符串。

(3) 控制语句

控制语句用于完成一定的控制功能，以实现程序的各种结构方式。它们由特定的语句定义符组成。C 语言只有九种控制语句，它们是：

- 选择结构控制语句。if 语句、switch 语句。
- 循环结构控制语句。do-while 语句、while 语句、for 语句。
- 转向语句。break 语句、goto 语句、continue 语句、return 语句。

(4) 复合语句

复合语句是由大括号{}括起来的一组语句构成，也称为分程序。在程序中应把复合语句看成是单条语句，而不是多条语句。

例如：

```
{
    t = x;
    X = y;
    y = t;
}
```

对复合语句说明如下。

- 复合语句在语法上和单一语句相同，即单一语句可以出现的地方，复合语句也可在此使用。
- 复合语句可以嵌套，即复合语句里还可以出现复合语句。

(5) 空语句

空语句仅由一个分号“;”构成。它什么也不做。有时用来做被转向点或循环体(此时循环体不执行任何操作)。例如：

```
while(getchar()!= '\n')
;
```

本语句的功能是，只要从键盘输入的字符不是回车则重新输入。这里的循环体为空语句。

【知识拓展】

拓展任务 3.1：试用传统流程图描述判断输入年份是否为闰年的算法。

任务分析：闰年的条件是能被 4 整除但不能被 100 整除或能被 400 整除。判断某条件是否成立，需要用到选择结构。参考流程图如图 3.12 所示。

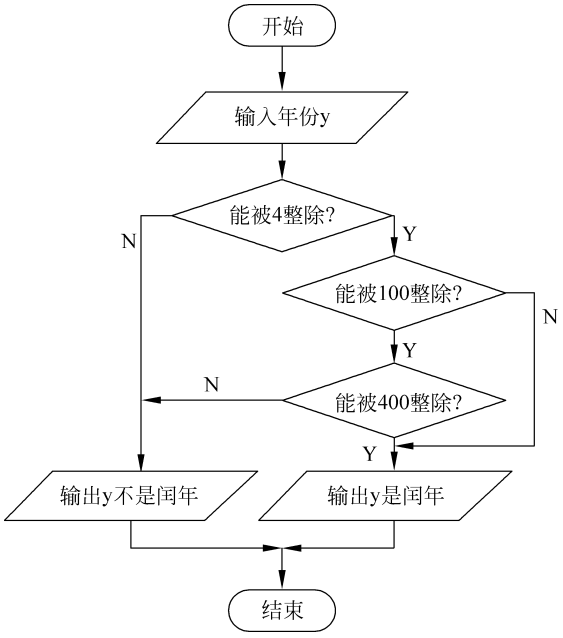

图 3.12　判断输入年份是否为闰年流程图

拓展任务 3.2：试用 N-S 图描述求解 1＋2＋…＋100 的算法。

任务分析：要求解 1＋2＋…＋100 的结果，可先设置一个初值为 0 的累加器 sum，再设置一个初值为 0 的变量 n，反复计算 sum＝sum＋n 的值，其中 n 的值依次取 1，2，…，100，从而求出 1＋2＋…＋100 的累加和。在一定条件下，反复执行同一操作，需要用到循环结构。参考流程图如图 3.13 所示。

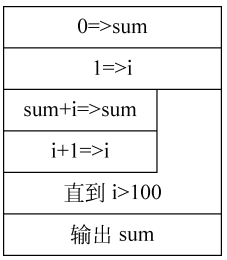

图 3.13　求解 1 至 100 的算法和 N-S 流程图

【知识小结】

（1）结构化程序设计三种基本结构分别为顺序结构、选择结构及循环结构。任何复杂的程序可以用这三种基本结构组合而成。

（2）算法的表示方法有自然语言表示法、传统流程图表示法、N-S 流程图及伪代码表示法等，其中传统流程图法的特点是画法简单、结构清晰、逻辑性强、便于描述且容易理解。但是流程图占用的篇幅多，而且当算法复杂时，每一个步骤要画一个框，比较费事。N-S 流程图法的特点是比文字描述更直观、更形象、更易于理解，比传统流程图紧凑易画，废除了流程线，整个算法结构是由各个基本结构按顺序组成的。

（3）程序中的语句一般可分为表达式语句、函数调用语句、控制语句、复合语句和空语句五类。

3.2　输入与输出函数的使用

所谓的输入是指输入设备向计算机输入数据；输出是指由计算机向外部设备输出数据。C 语言本身不提供输入/输出语句，输入和输出的操作是由调用库函数来实现的。在 C 标准函

数库中提供了一些输入/输出函数，例如 printf()函数、scanf()函数、putchar()函数和 getchar()函数等。

本节学习目标：

- 掌握格式输入与输出函数使用的方法与技巧。
- 掌握字符输入与输出函数使用的基本格式。

3.2.1 格式输出函数 printf()

【任务提出】

任务 3.2：已知圆的半径 $r=3$cm，计算并输出圆的面积。（保留小数点后面两位数字）

【任务分析】

本任务首先应定义两个变量用于存储圆的半径和面积的值，通过公式 $s=\pi r^2$ 计算圆的面积，并利用格式输出函数 printf()输出相应数据。

【任务实现】

参考代码如下：

```
1    #include <stdio.h>
2    int main()
3    {
4        int r = 3;
5        float s;
6        s = 3.14159*r*r;
7        printf("r = %d 厘米,s = %7.2f 平方厘米\n",r,s);
8        return 0;
9    }
```

程序运行结果如图 3.14 所示。

图 3.14 任务 3.2 程序运行结果

程序分析：程序第 4 行和第 5 行定义了整型变量 r 和实型变量 s；第 6 行通过圆的面积公式求出了圆的面积并保存在变量 s 中；第 7 行用 printf()函数输出了半径和面积的值，其中，半径的输出格式%d 是指以十进制整数输出，面积输出格式%7.2f 是指以最小宽度为 7 且小数点后面保留两位的十进制小数的格式输出，而其他的非格式字符原样输出。

【知识讲解】

格式输出函数 printf()是将输出项按指定的格式输出到标准输出终端上。

1. printf()函数调用的一般形式

printf()函数是一个标准库函数，它的函数原型包含在头文件 stdio.h 中。但作为一个特

例,有的编译器不要求在使用 printf()函数之前必须包含 stdio.h 文件。printf()函数调用的一般形式如下:

```
printf("格式控制字符串",输出表列);
```

其中格式控制字符串用于指定输出格式。格式控制字符串可由格式说明字符串和普通字符串两种组成。格式说明字符串是以%开头的字符串,在%后面跟有各种格式字符,以说明输出数据的类型、形式、长度、小数位数等。如%d 表示按十进制整型输出,%ld 表示按十进制长整型输出,%c 表示按字符型输出等。普通字符串在输出时将会原样输出,在显示中起提示作用。输出表列中给出了各个输出项,要求格式说明字符串和各输出项在数量和类型上应该一一对应。

2. 格式说明字符串

格式说明字符串的一般形式如下:

```
%[标志][输出最小宽度][.精度][长度]类型
```

其中方括号[]中的项为可选项。各项的意义介绍如下。

(1) 类型

类型字符用以表示输出数据的类型,其格式符和意义见表 3.1。

表 3.1 输出格式符和意义

格式字符	意　义
d	以十进制形式输出带符号整数(正数不输出符号)
o	以八进制形式输出无符号整数(不输出前缀 0)
x、X	以十六进制无符号形式输出整数(不输出前缀 0x),用 x 输出十六进制数 a～f 时以小写形式输出,用 X 时则以大写形式输出
u	以十进制形式输出无符号整数
f	以小数形式输出单、双精度实数
e、E	以指数形式输出实数,数字部分小数位数为 6 位,如用 E,则输出时指数以大写表示
g、G	选用%f 或%e 格式中输出宽度较短的一种格式,不输出无意义的 0,用 G 时,若以指数形式输出,则指数以大写表示
c	以字符形式输出,输出单个字符
s	输出字符串

(2) 标志

标志字符为－、＋、空格、＃四种,其意义见表 3.2。

表 3.2 输出标志和意义

标志	意　义
－	结果左对齐,右边填空格
＋	输出符号(正号或负号)
空格	输出值为正时冠以空格,输出值为负时冠以负号
＃	对 c、s、d、u 类无影响;对 o 类,在输出时加前缀 o;对 x 类,在输出时加前缀 0x;对 e、g、f 类,当结果有小数时才给出小数点

(3) 输出最小宽度

用十进制整数来表示输出的最少位数。若实际位数多于定义的宽度,则按实际位数输出,该数不起作用,若实际数据宽度小于输出最小宽度数值,且数值前无“-”号时,则结果右对齐,左边填空格。

(4) 精度

精度格式符以“.”开头,后跟十进制整数。本项的意义是:如果输出数字,则表示小数的位数;如果输出的是字符,则表示输出字符的个数;若实际位数大于所定义的精度数,则截去超过的部分。

(5) 长度

长度格式符为 h 和 l 两种,h 表示按短整型量输出,l 表示按长整型量输出。

例 3.1 分别以整型及字符型数据格式输出 65 和 66。

参考代码如下:

```
1  #include <stdio.h>
2  int  main()
3  {
4    int a=65,b=66;
5    printf("%d %d\n",a,b);
6    printf("%4d,%4d\n",a,b);
7    printf("%c,%c\n",a,b);
8    printf("a=%-4d,b=%-4d\n",a,b);
9    return  0;
10 }
```

程序运行结果如图 3.15 所示。

```
65 66
  65,  66
A,B
a=65  ,b=66
Press any key to continue
```

图 3.15 例 3.1 程序运行结果

程序分析:4 次输出了 a 和 b 的值,但由于格式控制字符串不同,输出的结果也不相同。第 5 行的 printf()函数格式控制串中,两格式串%d 之间加了一个空格(非格式字符),所以输出的 a 和 b 值之间有一个空格。第 6 行的 printf()函数格式控制字符串中的两格式说明符%4d 间加入普通字符逗号,因此输出的 a 和 b 值之间加了一个逗号,其中的 4d 表示输出的最小宽度为 4,如不足 4 位则左补空格。第 7 行的格式串要求按字符型输出 a、b,所以输出的是 ASCII 码值为 65、66 所对应的字符。第 8 行中对%4d 添加了标志符“-”,表示左对齐,不足最小宽度右补空格,增加的普通字符串用于对输出结果做提示说明。

例 3.2 分别以整型格式、无符号数据格式输出 2147483647 和 2147483648。

参考代码如下:

```
1  #include <stdio.h>
2  int main()
3  {
4    int a=2147483647,b;
5    b=a+1;
```