第 3 章　程序的基本控制结构

程序的基本组成部分包括数据输入、数据处理和数据输出，它们可以分解为适当的模块。程序的基本组织结构是数据结构和程序控制结构。为了构建结构良好的程序，每个模块都应该由一些适当的语句组构成，每个模块最好只实现一个功能，它们除了应该具有良好的数据结构外，还要有良好的程序控制结构。本章主要介绍程序的 3 种基本控制结构及其简单的程序设计。

3.1　程序的基本控制结构介绍

为了构建较复杂的程序，需要提供能够组织控制结构的语句，以便能够使各部分语句组充分发挥作用，并将它们顺利地组合成一个整体。程序的控制结构是指影响程序执行的逻辑顺序。计算机科学家已经证明只需要使用 3 种基本控制结构就可以构建任何复杂程序或算法。这 3 种基本控制结构是顺序结构、选择结构和循环结构。

3.1.1　3 种基本控制结构

在 1.5 节介绍 N-S 结构化流程图时，已经提到了程序的 3 种基本控制结构：顺序结构、选择结构和循环结构，这里再进一步阐述。

一个程序包含一系列的执行语句，每个语句使计算机完成一定的操作。在编制程序时，要仔细周密地考虑各语句的排列次序，语句的排列次序不仅决定程序的正确性，而且影响程序的可读性和可维护性。要使程序清晰地表达算法的设计思想，具有良好的可读性，编制程序时应当遵循人们的思维习惯，尽量避免不必要的无条件转向，最好能使程序的执行顺序从上到下依次进行。1966 年，由 Bohra 和 Jacopini 提出了 3 种基本控制结构，如果只用这 3 种基本控制结构作为算法设计的基本组成部件，就能使所编制的程序实现从上到下依次执行的目的。

1. 顺序结构

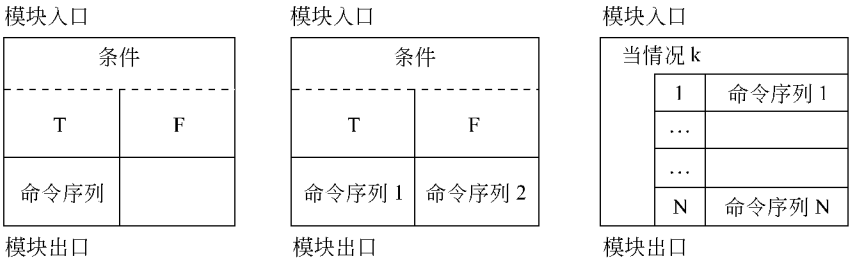
顺序结构是指程序的执行次序与程序的书写次序一致。任何程序从整体上看，都可以认为是顺序结构。程序的基本组成如图 3-1 所示，它是一个顺序结构，首先执行 a_1 模块输入数据，然后执行 a_2 模块处理数据，最后执行 a_3 模块输出数据。顺序结构是最简单也是最基本的结构。在进行算法设计和程序编制过程中，要树立这样的观念，顺序结构是整体的，选择结构和循环结构是局部的。

图 3-1　程序的基本组成

2. 选择结构

选择结构又称为判断结构或分支结构或条件结构，如图 3-2 所示，这个模块根据给定的条件是否成立而决定执行哪一个命令序列。选择结构包含有分支点和汇合点，分支点的条

件使程序产生不同的执行流程，当选择结构包含多个命令序列时，只有一个命令序列被执行，无论哪个命令序列被执行，执行流程都转到汇合点，即选择结构模块的出口处。



(a) 单向分支　　(b) 双向分支　　(c) 多向分支

图 3-2　N-S 图的选择结构框图

如图 3-2(a)所示是单向分支结构。程序设计语言中对应的是 if-then-endif 结构。若条件成立，则执行命令序列，否则什么也不做。

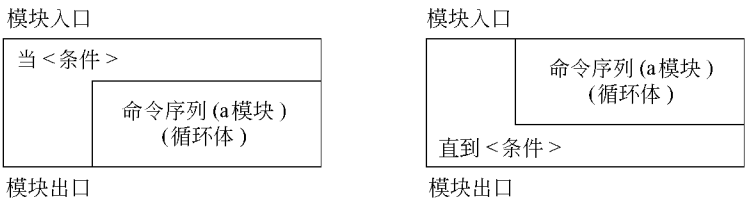
如图 3-2(b)所示是双向分支结构。程序设计语言中对应的是 if-then-else-endif 结构。若条件成立，则执行命令序列 1，否则执行命令序列 2。

如图 3-2(c)所示是多向分支结构。根据情况 k 决定执行哪一个命令序列，当 k 的取值在 1～N 时，就执行命令序列 k。在不同的程序设计语言中，对于组织多向分支结构的语句及其语句结构和规则有所差异，具体应用时要搞清楚语句结构和执行规则。

多向分支结构可以通过双向分支结构的嵌套来实现，不同的程序设计语言对允许嵌套的层次也不相同。在程序设计中，当需要使用选择结构时，应尽量减少选择结构的嵌套层次，同时使选择结构中的命令序列尽量简短。也就是说，在多个结构进行排列时，能够顺序排列的，绝不嵌套排列，并且使选择结构和循环结构的模块规模尽量地小。

3. 循环结构

循环结构又称为重复结构，如图 3-3 所示，这个模块使循环体在一定条件下重复执行。循环结构中的分支点和汇合点都在条件处，只要满足条件就执行循环体一次，然后判断条件，直到不满足条件时才退出循环结构。在不同的程序设计语言中，所提供的组织循环结构的语句及其语句结构和规则有所差异。实际应用时要搞清楚循环语句结构和执行规则。



(a) 当型循环结构框图　　(b) 直到型循环结构框图

图 3-3　N-S 图的循环结构框图

无论让计算机做什么事，如果它只能做一次，那么这样的计算机几乎是没有什么用处的。一次次重复做同样事情的能力，是程序设计中最基本的要求。好在计算机一次次重复

做同样的事没有觉得很枯燥。循环结构是最基本也是最重要的控制结构。

一个实际问题无论多么复杂,都可以由这 3 种基本控制结构组成。用这 3 种基本控制结构构造算法和编制程序,就如同搭积木盖房子一样方便,它使得程序结构清晰。这是结构化程序设计的基本要求,事实上,结构化程序设计的目标之一就是构建易于阅读和理解的程序。

3.1.2 关于对 GOTO 语句的认识

在现代编程语言方面,E. W. Dijkstra 以著名的反对(过分)使用 GOTO 语句的文章而著名。1968 年,E. W. Dijkstra 撰写了一篇题为《GOTO Statement Considered Harmful》的文章,这篇文章被认为是现代编程语言逐渐不鼓励使用 GOTO 语句,提倡使用编程控制结构的一个分水岭。一个有趣的插曲是 E. W. Dijkstra 的这篇文章的题目其实并不是他自己取的,而是 Communications of the ACM 的编辑 Niklaus Wirth 的杰作。这篇文章的发表,引起了软件界的一场大辩论。

主张在程序设计语言中取消 GOTO 语句的人们,其主要理由是:程序的执行是个动态的过程,如果程序中不加限制地使用 GOTO 语句,这就使程序的静态结构和动态执行情况差异甚大,使得程序难以阅读和理解,容易出错,也难以检查错误。由于破坏了程序基本结构的单入口、单出口原则,使程序的正确性证明复杂化。相反,取消 GOTO 语句,可以增强程序静态结构与动态执行情况的一致性。事实上,取消 GOTO 语句所带来的好处远不止这些。

另一些人提出不同的观点:认为 GOTO 语句从概念上来说是非常简明的;使用 GOTO 语句,可以提高程序的执行效率。

这场辩论一直持续了数年,直到 1974 年 Knuth 发表了一篇题为《带 GOTO 语句的结构程序设计》文章后,才算结束了这场争论。他概括地证实了 3 点:①滥用 GOTO 语句确实有害,应该尽量避免;②完全避免使用 GOTO 语句也并非是明智的方法,在有些程序的有些地方使用 GOTO 语句以后,将会使程序流程更清晰、效率更高;③争论的焦点不应放在要不要取消 GOTO 语句上,而应该放在采用什么样的程序结构上。因为只有良好的程序结构,才能使程序易于理解、易于维护。

滥用 GOTO 语句确实有害。在程序中使用 GOTO 语句使得程序文本与程序的动态执行不相对应,程序既不容易阅读,也不容易纠错和验证。E. W. Dijkstra 承认程序的可读性与程序的效率之间有着反作用。

如何避免滥用 GOTO 语句。①理论上取消 GOTO 语句;②消除 GOTO 语句的 3 种方法:重复编码方法、状态变量方法(转换成循环结构或选择结构)和布尔标志技术;③产生新的语言机制及新型程序设计语言。在进行算法设计和编制程序过程中重要的是要考虑程序的结构、程序的清晰度及可读性,在这个过程中关键是很少想到用 GOTO 语句的方式来编写程序。

可以使用 GOTO 语句的情况。程序员应该创建这样的算法 P,P 是容易理解的并且有良好的算法结构,然后对 P 进行优化,使之产生一个高效的程序 Q,Q 可以包含 GOTO 语句,不过由 P 到 Q 的变换应该是完全可靠的。P 是面向人的,而 Q 是面向机器的。

优化程序的方法可以通过将递归改为迭代，或者对于内层循环的循环体来说，应该通过建立适当的数据结构和控制结构，使内层循环的循环体更精练，因为只有内层循环的循环体运行时间愈少，程序效率才会愈高。

过早地优化是一切祸害的根源，应该忘掉进行过早优化。实质上，应该把程序开发分成几个层次，把变换前供人们进行阅读、交流的算法看成一个层次；把变换后面向机器能高效运行的程序看成一个层次；最后产生的目标代码又是一个层次。不同的层次目的不同，自然对它们的要求也应该不同。

一般来说，用删除 GOTO 语句的方法替换出的程序，不仅效率有所降低，而且可读性也不会得到改善。或者说，不能简单地认为，对带有 GOTO 语句的程序，只要简单地用其他语言成分替代了 GOTO 语句，就可以得到好结构程序。人们真正需要的是：在程序设计过程中就很少想到去使用 GOTO 语句。

怎样才能设计出好结构程序呢？这个问题与如何进行程序设计的技术和方法紧密相关。程序设计应该从方法学的角度进行根本性的变革。

程序结构良好是指程序结构清晰，易于理解，也易于验证。好结构程序从效率上看，不一定是好程序，但它便于阅读，提高了程序的可靠性和可维护性。结构化程序设计的基本要求是：宁可损失一些程序的执行效率，也要保持程序的好结构。采用程序的 3 种基本控制结构和单入口、单出口原则是设计好结构程序应该严守的信条。

3.2 顺序结构程序设计

顺序结构程序是最基本、最简单的一种程序结构。程序中的所有语句都是按照自上而下的顺序来执行，不会发生执行流程的跳转。虽然程序结构简单，但是在解决问题时，也应该按照程序设计步骤进行，做好问题分析、算法设计，不要急于写程序代码。在开始学习计算学科时就养成良好的程序设计风格。

例 3-1 计算应收款。试编写一个程序，用于水果店售货员结账。已知苹果每公斤 3.5 元，香蕉每公斤 4.2 元。输入顾客所买各种水果重量，计算应收款。再输入顾客付款额，计算应找顾客金额。

问题分析。这个问题虽然简单，但是也需要想好解题的方法和步骤。题目要求输出数据是应收款 Receivables 和需要找给顾客的金额；需要输入数据是苹果的重量 AppleWeight、香蕉的重量 BananaWeight 以及顾客付款额 Pay；水果单价定义为符号常量。根据程序的基本组成结构：输入数据，处理数据，输出结果。算法设计如图 3-4 所示，算法中的变量说明如下：

```
CONST
    ApplePrice=3.5
    BananaPrice=4.2
VAR AppleWeight,BananaWeight:real
    Pay:real
    Receivables:real
```

算法 3-1　CalcuReceivables

定义符号常量　ApplePrice=3.5;BananaPrice=4.2
输入数据　AppleWeight;BananaWeight
计算应收款　Receivables=ApplePrice * AppleWeight+BananaPrice * BananaWeight;
输出应收款　Receivables
输入数据　Pay
输出数据　Pay−Receivables
算法结束

图 3-4　计算应收款

根据算法 3-1 编写的 C 源程序如下：

```
//*******************************************************
//* 程序名称：CalcuReceivables.cpp                      *
//* 程序功能：计算应收款                                *
//* 作　　者：FENGJUN                                   *
//* 编制时间：2009 年 3 月 20 日                        *
//*******************************************************
#include<stdio.h>
#define ApplePrice 3.5
#define BananaPrice 4.2
void main()
{
    float AppleWeight,BananaWeight,Pay;
    float Receivables;
    printf("请输入苹果重量:");
    scanf("%f",&AppleWeight);
    printf("请输入香蕉重量:");
    scanf("%f",&BananaWeight);
    Receivables=ApplePrice* AppleWeight+BananaPrice*BananaWeight;
    printf("应收款 Receivables=%f\n", Receivables);
    printf("请输入顾客付款额:");
    scanf("%f",&Pay);
    printf("找零=%f \n", Pay-Receivables);
}
```

运行程序得到如下结果：

```
请输入苹果重量:12↙
请输入香蕉重量:8↙
应收款 Receivables=75.600000
请输入顾客付款额:80↙
找零=4.400000
```

例 3-2 输入三角形的 3 条边长，计算三角形的面积。

问题分析。

(1) 输入三角形的 3 条边长 a、b、c。为了方便起见，假设这 3 条边能够构成三角形。

(2) 已知三角形的 3 条边，求三角形面积的公式为

$$area=SQRT(s(s-a)(s-b)(s-c))$$

其中：SQRT()是求平方根函数，s=(a+b+c)/2。

(3) 输出三角形的面积。

N-S 图算法描述略。根据程序的基本组成结构：输入数据，处理数据，输出结果。编写 C 源程序如下：

```
//*******************************************************
//* 程序名称: Calcuarea.cpp                             *
//* 程序功能: 计算三角形面积                            *
//* 作    者: FENGJUN                                    *
//* 编制时间: 2009年 3月 20日                           *
//*******************************************************
#include<stdio.h>
#include<math.h>                          /* 因为要调用数学函数 sqrt() */
void main()
{
    float a,b,c;
    double s,area;
    printf("请输入三角形的 3条边长 a,b,c=" );
    scanf("%f%f%f",&a,&b,&c);
    s=(a+b+c)/2;
    area=sqrt (s * (s-a) * (s-b) * (s-c));
    printf("三角形的 3条边长 a,b,c=%f %f%f\n ",a,b,c );
    printf("则三角形的面积 area=%f\n ",area );
}
```

运行程序得到如下结果：

```
请输入三角形的 3条边长 a,b,c=4 5 6↙
三角形的 3条边长 a,b,c= 4.000000 5.000000 6.000000
则三角形的面积 area= 9.921567
```

这个程序不完善，或者说不健壮。当输入 3 个数 a、b、c，以它们为边长不能构成三角形时，程序中的计算将是没有意义的。因此，输入 3 个数后，应首先判断以它们为边长是否能构成三角形，只有能构成三角形，再计算三角形面积才有意义，这个问题将在 3.3 节中解决。

例 3-3 求一元二次方程 $ax^2+bx+c=0$ 的根。系数 a、b、c 由键盘输入。

问题分析。

(1) 输入一元二次方程的系数 a、b、c。为了方便起见，假设 $b^2-4ac>0$。

(2) 确定解方程的方法。已知一元二次方程的求根公式为

$$x1=\frac{-b+\sqrt{b^2-4ac}}{2a},\quad x2=\frac{-b-\sqrt{b^2-4ac}}{2a}$$

可以将上面的分式分为两项：

$$p = \frac{-b}{2a}, \quad q = \frac{\sqrt{b^2 - 4ac}}{2a}$$

则 x1＝p＋q，x2＝p－q。

(3) 输出一元二次方程 $ax^2+bx+c=0$ 的两个根 x1 和 x2。

N-S 图算法描述略。根据程序的基本组成结构：输入数据，处理数据，输出结果。编写 C 源程序如下：

```
//*****************************************************
//* 程序名称：SolutionEquation.cpp                    *
//* 程序功能：解一元二次方程                          *
//* 作　　者：FENGJUN                                 *
//* 编制时间：2009 年 3 月 20 日                      *
//*****************************************************
#include<stdio.h>
#include<math.h>                                  /* 因为要调用数学函数 SQRT() */
void main()
{
    float a,b,c;
    float p,q,x1,x2;
    printf("请输入一元二次方程的系数 a,b,c=" );
    scanf("%f%f%f",&a,&b,&c);
    p=-b/(2*a); q=sqrt(b*b-4*a*c);
    x1=p+q; x2=p-q;
    printf("一元二次方程%fx*x+%fx+%f=0\n ",a,b,c );
    printf("的两个解 x1=%f, x2=%f\n ",x1,x2 );
}
```

运行程序得到如下结果：

```
请输入一元二次方程的系数 a,b,c=3.5  7.6  2↙
一元二次方程 3.500000x*x+7.600000x+2.000000=0
的两个解 x1=-0.369558, x2=-6.540987
```

在程序中假定输入的 a、b、c 满足条件 $b^2-4ac>0$。事实上，所输入的 a、b、c 并不一定满足条件 $b^2-4ac>0$。因此，在利用求根公式解方程时，应首先判断条件 $b^2-4ac>0$ 是否成立，然后再做相应地处理，这个问题将在 3.3 节中解决。请读者给出这两个例子的 N-S 图算法描述。

3.3 选择结构程序设计

解决稍微复杂些的问题，就需要使用选择结构。选择结构包含一个测试条件和一个或多个命令序列，根据条件是否成立，决定执行哪个命令序列。在学习选择结构程序设计时，一定要搞清楚在不同条件下程序的各种不同执行流程。

3.3.1 单向分支选择结构程序设计

最简单的选择结构是单向分支选择结构。在程序设计语言中，它的一般形式为

```
if 条件 then
   命令序列
endif
```

分支点在条件处。若条件成立，则执行命令序列；否则跳过命令序列，直接执行 endif 的后续命令。汇合点在 endif 处。

例 3-4 输入 3 个数 a、b、c，输出最大数。

问题分析。

(1) 由键盘输入 3 个数 a、b、c，再定义一个变量 max 用于存放最大数。

(2) 首先将 a 的值赋给 max；然后测试关系表达式 b＞max，若成立，则将 b 的值赋给 max；最后测试关系表达式 c＞max，若成立，则将 c 的值赋给 max。

(3) 输出 max 的值。

因此，算法设计如图 3-5 所示，算法中的变量说明如下：

```
VAR a, b, c, max:real
```

算法 3-2 max(a,b,c,max)

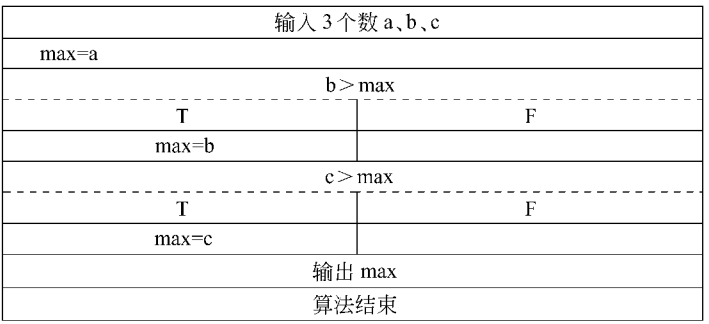


图 3-5 求最大数

根据算法 3-2 编写 C 源程序如下：

```
//*********************************************************
//* 程序名称：max.cpp                                     *
//* 程序功能：求最大数                                    *
//* 作　　者：FENGJUN                                     *
//* 编制时间：2009 年 3 月 20 日                          *
//*********************************************************
#include<stdio.h>
void main()
{
    float a,b,c, max;
    printf("请输入 3 个数 a,b,c=" );
    scanf("%f%f%f",&a,&b,&c);
```

```
    max=a;
    if(b>max) max=b;
    if(c>max) max=c;
    printf("3个数 a=%f b=%f c=%f\n ",a,b,c );
    printf("的最大数 max=%f\n ",max);
}
```

运行程序得到如下结果：

```
请输入 3 个数 a,b,c=35 76 188↙
3 个数 a=35.000000 b=76.000000 c=188.000000
的最大数 max=188.000000
```

例 3-5 输入 3 个数 a、b、c，要求由小到大排序。

问题分析。

(1) 由键盘输入 3 个数 a、b、c。

(2) 排序思想：排序后使 a 的值最小、c 的值最大。首先比较 a 与 b，若 b 小，则交换 a 与 b 的值；然后比较 a 与 c，若 c 小，则交换 a 与 c 的值；最后比较 b 与 c，若 c 小，则交换 b 与 c 的值。再定义一个中间变量 t 用于交换两个变量的值。

(3) 输出已排序 a、b、c 的值。

因此，算法设计如图 3-6 所示，算法中的变量说明如下：

```
VAR a, b, c, t:real
```

算法 3-3 sort(a,b,c)

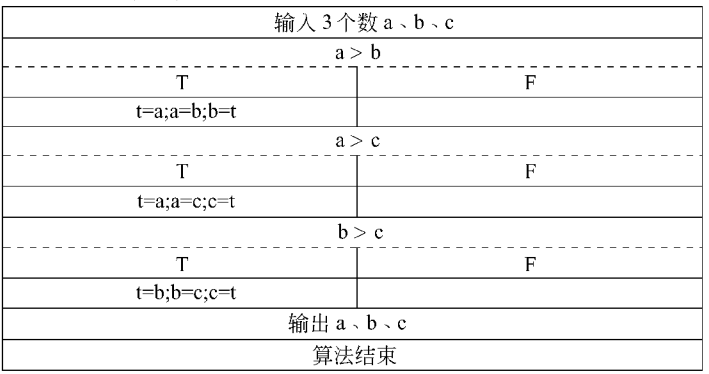


图 3-6 排序

根据算法 3-3 编写 C 源程序如下：

```
//*******************************************************
//* 程序名称：sort.cpp                                  *
//* 程序功能：排序                                      *
//* 作    者：FENGJUN                                   *
//* 编制时间：2009 年 3 月 20 日                        *
//*******************************************************
#include<stdio.h>
```

```
void main()
{
    float a,b,c,t;
    printf("请输入 3 个数 a,b,c=");
    scanf("%f%f%f",&a,&b,&c);
    if(a>b)
       {t=a;a=b;b=t;}
    if(a>c)
       {t=a;a=c;c=t;}
    if(b>c)
       {t=b;b=c;c=t;}
    printf("3 个数 a=%f b=%f c=%f\n ",a,b,c);
    printf("排序后的序列为:%6.2f, %6.2f, %6.2f\n", a,b,c);
}
```

运行程序得到如下结果：

```
请输入 3 个数 a,b,c=986.34 342 869.36↙
3 个数 a=986.340000 b=342.000000 c=869.360000
排序后的序列为:342.00,869.36,986.34
```

3.3.2 双向分支选择结构程序设计

双向分支选择结构是最完备的选择结构。在程序设计语言中，它的一般形式为

```
if  条件  then
    命令序列 1
else
    命令序列 2
endif
```

分支点在条件处。若条件成立，则执行命令序列 1，执行完毕跳过命令序列 2 执行 endif 的后续命令；否则，跳过命令序列 1，执行命令序列 2，再延续执行 endif 的后续命令。命令序列 1 与命令序列 2 有且仅有一个命令序列被执行。汇合点在 endif 处。

例 3-6 输入三角形的 3 条边长，计算三角形的面积。

问题分析。例 3-2 中的解法不完善，当输入 3 个数 a、b、c 后，首先判断以它们为边长是否能构成三角形，只有能构成三角形，再计算三角形面积才有意义。根据三角形知识，3 条边能构成三角形的条件是任意两边之和大于第 3 边。因此，例 3-2 中的 C 源程序可以修改如下：

```
//*****************************************************
//* 程序名称: Calcuarea1.cpp                          *
//* 程序功能: 计算三角形面积                          *
//* 作    者: FENGJUN                                  *
//* 编制时间: 2009 年 3 月 20 日                      *
//*****************************************************
#include<stdio.h>
#include<math.h>                          /* 因为要调用数学函数 sqrt() */
```