第 3 章

逻 辑 编 程

学习目标：

- 掌握各种分支结构
- 掌握各种循环结构
- 理解逻辑问题
- 掌握逻辑方法

3.1 引言

在计算机技术高速发展的今天，虽然计算机已经具备强大的计算能力，但依然是没有生命的机器。某些机器人具备独立解决问题的能力，但这也是人类赋予它的“人工智能”。因此，若要利用计算机解决实际问题，就必须告诉计算机如何解决问题，即把人类的逻辑分析结果、分析方法转换成计算机能识别的程序指令然后输入计算机从而解决问题。

3.2 分支结构

在解决实际问题过程中，常常会遇到很多不确定性，需根据不同的条件决定不同的解决方法。程序设计同样面临不确定性的问题，C 语言有两类分支结构：if 语句和 switch 语句。关系运算是实现分支语句的重要手段，以关系运算形成条件判断与结果走向。其运算符有：>=(大于等于)、>(大于)、<=(小于等于)、<(小于)、==(等于)、!=(不等于)。

【例 3-1】 开发远程控制系统，实现远程关机、发送远程消息、打开远程机器的控制面板、设置远程机器的分辨率等。

本例是一个网络程序，需要客户端和服务器端两个程序。远程计算机作为服务器端，本地计算机作为客户端，即从客户端发送不同的指令到服务器端，服务器端接收到指令后进行分析，根据不同指令执行不同的操作。客户端发送的指令即是自定义的网络协议，假设发送 1 时执行远程关机操作，发送 2 时接收文字信息，发送 3 时打开远程机器的控制面板，发送 4 时设置远程机器的分辨率。客户端程序相对简单，只起到了发送消息的作用，

在此不做演示。服务器端执行时的具体操作,如打开控制面板需要调用 API 函数 winexec("rundll32.exe shell32.dll,Control_RunDLL ",9)等细节问题不做详细介绍,目的在于说明分支语句的应用。服务器端程序清单如下。

```
//方法一
#include "stdafx.h"
int main(int argc, char * argv[])
{
   int ReceivedData;
    ⋮
    //假设从客户端发送的指令已接收到
    if(ReceivedData==1)
        printf("关机");
    if(ReceivedData==2)
        printf("接收消息");
    if(ReceivedData==3)
        printf("打开控制面板");
    if(ReceivedData==4)
        printf("设置分辨率");
    return 0;
}
```

也可以在 if 语句中使用 else 语句,如下所示:

```
//方法二
#include "stdafx.h"
int main(int argc, char * argv[])
{
    int ReceivedData;
    ⋮
    //假设从客户端发送的指令已接收到
    if(ReceivedData==1)
        printf("关机");
    else if(ReceivedData==2)
        printf("接收消息");
    else if(ReceivedData==3)
        printf("打开控制面板");
    else if(ReceivedData==4)
        printf("设置分辨率");
    return 0;
}
```

由于上述设计的自定义网络协议为整数,故也可使用 switch 语句来实现,如下所示:

```
//方法三
#include "stdafx.h"
int main(int argc, char * argv[])
{
    int ReceivedData;
    ⋮
```

```
    //假设从客户端发送的指令已接收到
    switch(ReceivedData)
    {
    case 1:
        printf("关机");
        break;
    case 2:
        printf("接收消息");
        break;
    case 3:
        printf("打开控制面板");
        break;
    default:
        printf("设置分辨率");
    }
    return 0;
}
```

3.3　循环结构

循环结构是程序设计中特别常用的一种结构，它能够充分发挥计算机高速重复运算的特点。在 C 语言中主要通过 for 语句、while 语句和 do-while 语句实现循环结构，完成具有一定规律性的处理过程。

【例 3-2】 求 1000 以内的水仙花数。水仙花数必须满足以下条件：这是一个三位数，数的个位、十位、百位的 3 次方的和等于这个数字本身。例如，$1^3+5^3+3^3=153$。

从该问题的条件得知，1000 以内的水仙花数是 3 位数的，即 0～99 无须考虑。实现水仙花数有两种方法：①3 位数都是数字组成的，数字为 0～9，则可在个位、十位、百位分别使用循环语句；②从 100 循环到 999，分离出个位、十位、百位上的数字。

```
//方法一
#include "stdafx.h"
int main(int argc, char * argv[])
{
    int i,j,k;
    for(i=1;i<=9;i++)
        for(j=0;j<=9;j++)
            for(k=0;k<=9;k++){
               if ((i * i * i+j * j * j+k * k * k)==(i * 100+j * 10+k))
                   printf("%d\n",i * 100+j * 10+k);
            }
    return 0;
}
//方法二
#include "stdafx.h"
int main(int argc, char * argv[])
{
```

```
    int i,j,k,n;
    n=100;
    while(n<1000)
    {
        i=n/100;          //分解出百位
        j=n/10%10;        //分解出十位
        k=n%10;           //分解出个位
        if(i*100+j*10+k==i*i*i+j*j*j+k*k*k)
            printf("%-5d",n);
        n++;
    }
    return 0;
}
```

【例 3-3】 求解 $\sin x$ 的近似值。

本例是一个数学问题,必须先找到解决该问题的方法(即算法),这与使用什么语言来编写程序无关,如果没有合适的算法再高明的程序员也无法解决问题。本例根据“泰勒公式”可得:

$$\sin x = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

分析上述表达式可知 x 的次方数和分母的阶乘数相同,并且都是由奇数组成的数列。每一数据项前的符号是正负号交替出现的,则可把这些数据项抽象成一个通用的表达式,如下所示:

$$(-1)^{\frac{n-1}{2}} \frac{x^n}{n!}, \quad \text{其中 } n \text{ 为奇数}(1, 3,5,7,9,\cdots)$$

从而 $\sin x$ 的值即转换为对上述表达式求和的操作。程序中处理 $(-1)^{\frac{n-1}{2}}$ 时,可先判断 $(n-1)/2$,如果是偶数则整个表达式值为正,如果是奇数整个表达式值为负。求解上述通用表达式可编写一个函数来实现。

```
#include "stdafx.h"
float GetData(int n,float x)
{
    float Total1=1;              //求次方
    float Total2=1;              //求阶乘
    for(int i=1;i <=n; i++)
    {
        Total1=Total1 * x;
        Total2=Total2 * i;
    }
    if(((n-1)/2)%2==0)           //偶数
        return Total1 / Total2;
    else                         //奇数
        return -Total1 / Total2;
}
int main(int argc, char * argv[])
{
```

```
    float sum=0;
    int n=7;                    //假设 n=7
    float x=1;                  //为了验证方便,假设 x=1
    for(int i=1;i <=n;i++)
    {
        if(i%2==0)
            continue;
        sum=sum+GetData(i,x);
    }
    printf("%f\n",sum);
    return 0;
}
```

3.4　逻辑问题

在实际应用中经常出现逻辑推理问题，即要同时满足多个条件，并且这些条件又很难立即确定，需要不断试探，不断推理。

【例 3-4】 在某小学，有位同学做了好事不留名。有一天班主任收到表扬信，根据信的内容将做好事的同学范围限定在某四位同学身上，问是谁做的好事。

同学 A 说：不是我。

同学 B 说：是同学 C 做的。

同学 C 说：是同学 D 做的。

同学 D 说：同学 C 瞎说。

已知其中三人说的是真话，一人说假话。如果你是班主任，请你找出做好事的人。

此类问题属于逻辑推理问题，根据题意不容易立即得到判断条件，需要经过严密的分析才能得出结论，解决此类问题可用枚举法。

步骤 1：由于只有一个人做好事，只有四种可能，如表 3-1 所示（假设四个人分别为 A、B、C、D，做了好事用 1 表示，否则用 0 表示）。

表 3-1　状态分析

学生 / 状态 / 编号	同学 A	同学 B	同学 C	同学 D	学生 / 状态 / 编号	同学 A	同学 B	同学 C	同学 D
1	1	0	0	0	3	0	0	1	0
2	0	1	0	0	4	0	0	0	1

定义一个变量 TheMan 表示做好事的人，把上述列表转换成表达式，如下所示。

```
TheMan='A';
TheMan='B';
TheMan='C';
TheMan='D';
```

步骤 2：再把四位同学说的话转换成表达式，如表 3-2 所示。

表 3-2 表达式分析

说话人	说的话	表达式	说话人	说的话	表达式
同学 A	不是我	TheMan != 'A'	同学 C	是同学 D 做的	TheMan == 'D'
同学 B	是同学 C 做的	TheMan == 'C'	同学 D	同学 C 瞎说	TheMan != 'D'

步骤 3：使用枚举法逐一测试，检查是否满足“其中三人说的是真话，一人说假话”的条件，即把每一种假设都带入，不断试探，直到满足条件为止。

(1) 假设做好事的人是同学 A，即 TheMan = 'A'。

表 3-3 结果分析(1)

说话人	说的话	表达式	试探	结果
同学 A	不是我	TheMan != 'A'	'A'! = 'A'	0
同学 B	是同学 C 做的	TheMan == 'C'	'A'== 'C'	0
同学 C	是同学 D 做的	TheMan == 'D'	'A'== 'D'	0
同学 D	同学 C 瞎说	TheMan != 'D'	'A'!= 'D'	1

从表 3-3 可以看出试探的结果只有一个是 1，其他三个是 0，不满足“其中三人说的是真话，一人说假话”的条件，因此一定不是同学 A 做的。

(2) 假设做好事的人是同学 B，即 TheMan = 'B'。

表 3-4 结果分析(2)

说话人	说的话	表达式	试探	结果
同学 A	不是我	TheMan != 'A'	'B'!= 'A'	1
同学 B	是同学 C 做的	TheMan == 'C'	'B'== 'C'	0
同学 C	是同学 D 做的	TheMan == 'D'	'B'== 'D'	0
同学 D	同学 C 瞎说	TheMan != 'D'	'B'!= 'D'	1

从表 3-4 可以看出试探的结果只有两个是 1，其他两个是 0，不满足“其中三人说的是真话，一人说假话”的条件，因此一定不是同学 B 做的。

(3) 假设做好事的人是同学 C，即 TheMan = 'C'。

表 3-5 结果分析(3)

说话人	说的话	表达式	试探	结果
同学 A	不是我	TheMan != 'A'	'C'!= 'A'	1
同学 B	是同学 C 做的	TheMan == 'C'	'C'== 'C'	1
同学 C	是同学 D 做的	TheMan == 'D'	'C'== 'D'	0
同学 D	同学 C 瞎说	TheMan != 'D'	'C'!= 'D'	1

从表 3-5 可以看出试探的结果有三个是 1，一个是 0，满足“其中三人说的是真话，一人说假话”的条件，因此一定是同学 C 做的。有兴趣的读者可以把第四种假设也列表，会发现只有两个 1，不满足条件。

通过以上透彻的分析不难看出，编程序时应有 4 次的探测(即需要循环语句，执行

4 次),在循环体内判断 4 个表达式是否有 3 个同时为真,如果是则找到做好事的人,否则相反。

```
#include "stdafx.h"
int main(int argc, char * argv[])
{
    char TheMan;
    int sum=0; //用于累加
    for(int i=0;i<4;i++)
    {
        TheMan='A'+i;
        sum=(TheMan!='A')+(TheMan=='C')+(TheMan=='D')+(TheMan!='D');
        if(sum==3)
        {
            printf("是同学%c做的好事",TheMan);
            break;
        }
    }
    return 0;
}
```

逻辑问题是一大类问题,属于逻辑学范畴,是离散数学的一个分支,注重逻辑推理。若读者如遇到更为复杂的逻辑问题,可把各种推理过程写成逻辑表达式,最后使用数学方法求解逻辑表达式,通过计算机程序得到结果。

3.5　实训案例

使用现有分支、循环结构知识编写一个猜数小游戏。游戏规则如下:

随机生成一个 20 以内的正整数,然后由用户猜测所生成的数,游戏后台自动判断是否猜中。用户有 5 次机会猜数,第一次猜中得 100 分,第二次猜中得 90 分,第三次猜中得 80 分,第四次猜中得 70 分,第五次猜中得 60 分并提示“终于猜中了!”。用户只有 5 次机会。

```
#include "stdafx.h"
#include <stdlib.h>
#include <time.h>
int main()
{
    int RandData,UserData;
    srand((unsigned)time(NULL));                    //为了防止生产相同的随机数
    RandData=rand() %20;                            //20 以内的随机数
    printf("请您猜猜,输入您猜的数\n");
    scanf("%d",&UserData);
    for(int i=0;i<6;i++)
    {
        if(RandData==UserData)
        {
```

```
            if(i==0)
                printf("您猜对了,得 100 分!\n");
            else if(i==1)
                printf("您猜对了,得 90 分!\n");
            else if(i==2)
                printf("您猜对了,得 80 分!\n");
            else if(i==3)
                printf("您猜对了,得 70 分!\n");
            else if(i==4)
                printf("您终于猜中了,得 60 分!\n");
            return 0;
        }
        else
        {
            if(i==5)
            {
                printf("对不起,您都猜错了!\n");
                return 0;
            }
            printf("您猜错了,请您继续猜,输入您猜的数!\n");
        }
        scanf("%d",&UserData);
    }
    return 0;
}
```

程序运行结果如图 3-1 所示。

图 3-1 猜数小游戏

3.6 编程理念

从例 3-1 可以看出：同一问题可以有多种解决方法；可以使用不同的程序设计语言来实现；程序设计语言只是一种工具，是解决问题的一种手段。方法一使用了 4 条 if 语句，方法二使用了一条 if 语句，方法三使用了一条 switch 语句。从执行效果上看方法二、方法三只执行了一次，方法一要执行 4 次，相比较而言方法二、方法三较优。

在例 3-2 中，两种方法执行的次数都是 900 次，似乎并无效果差异。但从算法性能来

分析，方法一执行了三重的循环，其时间复杂度为 $T(n)=O(n^2)$。方法二只执行了一重循环，其时间复杂度为 $T(n)=O(n)$，显然方法二较优。

例 3-3 是一类常见的问题。多数问题都有一定的规律，在分析问题时，需要从中发掘一种普遍存在和通用的规律，通过这个规律可以把复杂的问题简单化。在编写循环语句时要特别注意这样的规律，只有使用规律才能重复多次地执行循环，这是循环语句的本质。

例 3-4 是一大类逻辑问题，如果没有逻辑推理的方法则很难破解该类问题。此类问题的通用方法之一是“试探法”，首先把所有的可能性枚举出来，转换为逻辑表达式，再带入逐步试探，直到满足条件为止。

计算机具有强大的计算能力，但只是没有生命的电子设备，是人类发明的一种辅助工具。如果要让计算机解决问题，开发人员必须先知道如何解决问题，并把解决问题的步骤通过程序“告诉”计算机。因此程序必须具备很强的分析问题、解决问题的能力，并且还应善于设计，使用最优的算法程序来解决问题。