第5章

选择结构程序设计

CHAPTER 5

◇ **学习导读**

在编写程序时，有时并不能保证程序一定执行某些指令，而是要根据一定的外部条件来判断哪些指令要执行。选择结构可以让程序有选择地执行，满足条件就执行，不满足条件就不执行。根据判断的结果来控制程序的流程，属于流程控制语句。如一个菜谱中包含西红柿这个食材，需要加入西红柿时，可能有如下的步骤：如果是用新鲜的西红柿，则去皮、切碎，开始放入；如果是用西红柿酱，则直接放入。这里，并不知道具体操作时执行哪段指令，而是根据菜谱给出的条件进行处理，计算机程序也是如此，可以根据不同的条件执行不同的代码，这就是选择结构。

程序是为解决某个实际问题而设计的，而问题往往包含多个方面，不同的情况需要进行不同的处理，所以选择结构在实际应用程序中可以说是无处不在，离开了选择结构，很多情况将无法处理，因此正确掌握选择结构程序的设计方法对于编写实际的应用问题来说尤为重要。本章介绍了C语言源程序的选择结构程序设计。

◇ **内容导学**

(1) 选择结构的含义。

(2) 关系运算符、逻辑运算符和条件运算符。

(3) if、switch 语句的使用方法。

(4) 选择结构程序设计的编写方法。

(5) 编写选择结构问题的程序。

◇ **育人目标**

《论语》的“知之者不如好之者，好之者不如乐之者。”意为：懂得它的人，不如爱好它的人；爱好它的人，又不如以它为乐的人——有比较，才有鉴别，选择对了，胜过百般努力。

大千世界，选择无处不在。人的一生可能面临诸多选择，如此时此刻，或许你正站在十字路口，向左走还是向右走？为了在关键时刻能够实现选择自由，平时就需要不断努力地创造条件，创造机遇，提高能力和水平，做到志存高远、德才并重、情理兼修、勇于开拓，自然就能作出正确的判断和正确的选择，实现“山重水复疑无路，柳暗花明又一村”。计算机在求解问题时，也要考虑所有可能发生的情况，做到严谨、完备、不出差错。


视频讲解

5.1　关系运算符、逻辑运算符和条件运算符

本节主要讨论以下问题。

（1）选择结构中的条件如何表示？如何判定一个 C 语言表达式的“真”和“假”？用什么值表示表达式的“真”和“假”？

（2）关系运算符、逻辑运算符和条件运算符的功能是什么？关系运算符、逻辑运算符和条件运算符有哪些？其优先级和结合性怎样？什么是关系表达式、逻辑表达式？如何计算关系表达式和逻辑表达式？

（3）关系运算符、逻辑运算符和条件运算符如何表达条件？

在日常的学习和生活中，经常会遇到需要作出选择，如高考填报志愿、出门是否需要带伞等。在这些事件中，都蕴含着一个称为条件的信息，如当今天会下雨时就带伞，否则就不带；当高考分数在第一梯队时志愿填报 985 或 211 大学等。这里出现的“下雨或第一梯队”就是条件。因此，用选择结构解决问题，首先要学会描述条件。C 语言提供了描述条件的运算符：关系运算符、逻辑运算符和条件运算符。前者适合描述简单的条件，后两者适合描述两个或两个以上更复杂的条件。下面从运算符的类型、优先级和结合性以及计算规则等方面进行详细介绍。

5.1.1　关系运算符及其表达式

1. 关系运算符

关系运算符是用于表达比较运算的运算符。C 语言提供的关系运算符有 6 种，这 6 种关系运算符都属于双目运算符，其写法和含义如表 5-1 所示（假设变量 a、b 已定义）。

表 5-1　关系运算符的写法和含义

类　型	运 算 符	含　义	举　例
双目	>	大于	a>b、5>2
	<	小于	a<b
	>=	大于或等于	a>=b
	<=	小于或等于	a<=b
	==	等于	a==b
	!=	不等于	a!=b

在这 6 种关系运算符中，“>”“<”“>=”“<=”的优先级相同，且高于优先级相同的“==”和“!=”。关系运算符的优先级低于算术运算符而高于赋值运算符，结合性为左结合性。

2. 关系表达式

由关系运算符将两个表达式连接起来的表达式,称为关系表达式。它的一般格式是:

<表达式><关系运算符><表达式>

例如:a>b,a+b>b+c 都是合法的关系表达式。其中,表达式可以是任何类型的 C 语言合法的表达式。关系表达式的结果是一个逻辑值,即“真”和“假”,在 C 语言中用“1”代表“真”,用“0”代表“假”。

在计算关系表达式时,必须考虑其运算符的优先级和结合性,先“>”、“<”、“>=”和“<=”,再“==”和“!=”,同级运算符的计算顺序是从左到右,即左结合性,最后经过关系运算后得到关系表达式值为 1 或 0。

【例 5-1】 计算以下关系表达式(假设 x=3,y=5,z=1)。

① x+z>y

② y>z==x>z

③ y>x>z

④ a=x+y==x+z<y+x!=z+1>x+1

计算过程分析:

① 先计算 x+z,结果为 4;然后计算 4>y,结果为假。因此,该关系表达式值为 0。

② 先计算 y>z,结果为真,值为 1;然后计算 x>z,结果为真,值为 1;最后计算 1==1,结果为真。因此,该关系表达式值为 1。

③ 先计算 y>x,结果为真,值为 1;然后计算 1>z,结果为假。因此,该关系表达式值为 0。

④ 运算步骤如下:

第 1 步计算 x+y,结果为 8,此时表达式变为 a=8==x+z<y+x!=z+1>x+1。

第 2 步计算 x+z,结果为 4,此时表达式变为 a=8==4<y+x!=z+1>x+1。

第 3 步计算 y+x,结果为 8,此时表达式变为 a=8==4<8!=z+1>x+1。

第 4 步计算 z+1,结果为 2,此时表达式变为 a=8==4<8!=2>x+1。

第 5 步计算 x+1,结果为 4,此时表达式变为 a=8==4<8!=2>4。

第 6 步计算 x+z<y+x ,即 4<8,结果为真,值为 1,此时表达式变为 a=8==1!=2>4。

第 7 步计算 z+1>x+1,即 2>4,结果为假,值为 0,此时表达式变为 a=8==1!=0。

第 8 步计算 x+y==x+z<y+x,即 8==1,结果为假,值为 0,此时表达式变为 a=0!=0。

第 9 步计算 x+y==x+z<y+x!=z+1>x+1,即 0!=0,结果为假,值为 0,此时表达式变为 a=0。

最后,a=x+y==x+z<y+x!=z+1>x+1 即 a=0。

因此,变量 a 的值为 0。

5.1.2 逻辑运算符及其表达式

1. 逻辑运算符

使用关系运算符可以比较两个数的大小。但是,对于一些比较复杂的问题,只有关系运

算符还不够，如表示一个数是否在某个范围内或者同时满足多个条件或者满足若干条件之一等诸如此类的问题。

为了更好地解决上述问题，C 语言提供了逻辑运算符。逻辑运算符共有三种。它们分别是“&&”(与)、“||”(或)和“!”(非)，其中“&&”(与)、“||”(或)是双目逻辑运算符，“!”(非)是单目逻辑运算符。

这些逻辑运算符中，优先级由高到低为“!”“&&”“||”。和其他运算符的优先级相比，从高到低依次为：“!”、算术运算符、关系运算符、“&&”、“||”、赋值运算符。逻辑运算符“&&”和“||”的结合性都为左结合性，“!”的结合性为右结合性。

2. 逻辑表达式

用逻辑运算符将表达式连接起来的表达式，称为逻辑表达式。它的一般格式是：

<表达式>　<逻辑运算符>　<表达式>

例如：a&&b，a+b||b+c 都是合法的逻辑表达式。其中，表达式可以是任何类型的 C 语言合法的表达式。逻辑表达式的值也是一个逻辑值。同时，C 语言在使用一个表达式作为条件使用时，如果这个表达式值为 0 表示“假”，为非 0 表示“真”。

在计算逻辑表达式时，需要掌握逻辑运算符的运算规则。逻辑运算符的运算规则如表 5-2 所示(假设 a 和 b 代表任意两个表达式)。

表 5-2　逻辑运算符的运算规则

a	b	!a	a&&b	a||b
真	真	假	真	真
真	假	假	假	真
假	真	真	假	真
假	假	真	假	假

根据逻辑运算符的运算规则表可知：a&&b，只有当表达式 a 并且表达式 b 都为真时，表达式值才为真，否则一律为假；a||b，当表达式 a 或者表达式 b 有一个为真时，表达式值为真；!a，a 为真时，表达式值为假，反之表达式值为真。

3. 逻辑表达式的应用

(1) 逻辑表达式的计算。

【例 5-2】 计算以下表达式的值(设 a=2，b=3，c=0)。

① a&&b

② !a+c&&b+c

③ !c+a==b||b<a

计算过程分析：

① a 和 b 都是一个变量且值为非 0，即为真。因此，该逻辑表达式值为 1。

② 先计算!a，值为 0；然后计算!a+c，值为 0。因此，该逻辑表达式值为 0。

③ 运算步骤如下。

第 1 步计算!c，结果为 1，此时表达式变为 1+a==b||b<a。

第 2 步计算 1+a，结果为 3，此时表达式变为 3==b||b<a。

第 3 步计算 b<a，结果为 0，此时表达式变为 3==b||0。

第4步计算3==b,结果为1,此时表达式变为1||0。

第5步计算1||0,结果为1,此时表达式值为1。

因此,表达式的值为1。

(2) 逻辑表达式的构造。

【例5-3】 设有数学表达式a≥b≥c,将其转换成C语言支持的表达式。

分析:

类似以上的数学表达式计算机不支持,会误认为一个关系表达式,因此,必须转换成计算机支持的表达式,在转换的过程中,需要借助关系运算符和逻辑运算符。

数学表达式a≥b≥c表示a大于或等于b且b大于或等于c,因此,转换为C语言支持的表达式为a>=b&&b>=c。

【例5-4】 地球绕太阳运行的周期为365天5小时48分46秒(合365.24219天),即一回归年(Tropical Year)。公历的平年只有365天,比回归年短约0.2422天,所余下的时间约为每四年累积一天,故在第四年的2月末加1天,使当年的时间长度变为366天,这一年就是闰年。现行公历中每400年有97个闰年。按照每四年一个闰年计算,平均每年就要多算出0.0078天,这样,每128年就会多算出1天,经过400年就会多算出3天多。因此,每400年中要减少3个闰年。所以公历规定:年份是整百数时,必须是400的倍数才是闰年;不是400的倍数的世纪年,即使是4的倍数也不是闰年。这就是通常说的:四年一闰,百年不闰,四百年再闰。

假设用year表示某一年,要求用逻辑表达式表示以上闰年的条件。

分析过程:

根据以上描述,闰年的条件应符合下面二者之一。

① 能被4整除,又能被400整除,如2000年。

② 能被4整除,但不能被100整除,如2008年。

在C语言中表示a能被b整除,就是指a除以b的余数等于0,符合两种条件之一或者满足两种条件,分别用逻辑运算符“||”(或)和“&&”(且)表示。因此,根据分析可以得出判断闰年的逻辑表达式为:

```
year % 400 == 0||(year % 4 == 0&&year % 100!= 0)
```

反之,判断非闰年的逻辑表达式为:

```
!(year % 400 == 0||(year % 4 == 0&&year % 100!= 0))
```

5.1.3 条件运算符及其表达式

条件运算符(?:)是C语言中唯一的三目运算符。条件表达式的一般格式为:

表达式1?表达式2:表达式3

在计算该表达式时,遵守以下规则:先求表达式1的值,如果非0(真),则求表达式2的值,且表达式2的值作为整个条件表达式的值;否则,表达式3的值为整个条件表达式的值。条件运算符的优先级低于关系运算符而高于赋值运算符,且条件运算符的结合性为右结合性。

【例5-5】 计算以下语句的输出结果。

```
printf("%d",a>b?b:a);
```

(1) 若 a=3,b=4,则有 a>b 为 0,因此,该语句输出 a 的值。输出结果为:3。

(2) 若 a=4,b=3,则有 a>b 为 1,因此,该语句输出 b 的值。输出结果为:3。

5.2　选择结构程序设计语句

本节主要讨论以下问题。

(1) 实现选择结构程序设计的语句有哪些?

(2) if 语句有哪些形式?不同形式的 if 语句的功能是什么?其格式、执行过程如何?在使用过程中应该注意哪些问题?

(3) switch 语句的功能是什么?其执行过程如何?在使用过程中应该注意哪些问题?

(4) 如何选择 if 语句和 switch 语句?

选择结构是指程序中的语句根据是否符合条件决定执行。其基本特点是:程序的流程由多路分支组成,在程序的执行过程中,根据不同的情况,只有满足条件的分支被选中执行,而其他不满足条件的分支则不执行。

按照分支的条数,选择结构分为单分支选择结构、双分支选择结构及多分支选择结构。C 语言提供了两条实现选择结构的语句,它们分别是 if 语句和 switch 语句。下面将从语句的格式、功能、执行和使用详细介绍实现选择结构的各条语句。

5.2.1　if 语句


视频讲解

C 语言提供了 if 语句,用来判定所给定的条件是否满足,根据判定的结果(真或假)决定是否执行对应的操作。if 语句的使用很灵活,根据功能可以分为 3 种不同的格式,分别用来实现单分支结构、双分支结构和多分支结构。

1. if 语句实现单分支结构

if 语句实现单分支结构的一般使用格式为:

```
if(表达式)  语句;
```

其含义是:当表达式值为真时,执行语句,否则执行 if 语句的下一条语句。其中,格式中的表达式称为条件表达式,条件表达式可以是任何类型的表达式,当表达式值为非 0 即为真时,就执行语句。该语句可以由一条或多条语句组成,当由多条语句组成时该语句称为语句块或复合语句。这些语句块或复合语句必须放在一对大括号({})中。

例如:

```
if (x>y) printf("%d", x);      /*如果 x>y,则输出 x*/
if (x>y) {x+=5;y-=5;}          /*如果 x>y,则执行语句块,由两条语句组成*/
```

if 语句实现单分支结构执行过程的流程图如图 5-1 所示。

图 5-1　单分支 if 语句 N-S 图

【例 5-6】　输入两个整型数据,编写按照从大到小的顺序输出两个整数的程序。

算法分析:

设有两个整型数据 a、b,要求按从大到小的顺序输出。可以看出,该问题的要求是比较这两个数 a 和 b 的大小,根据比较结果确定

输出 a 和 b 的顺序。因此,建立的算法模型为:若 a>b,则输出的顺序为 a、b,否则输出的顺序为 b、a。

从这种方法的描述中得知,不管是哪种情况都要进行输出操作。若输出顺序只有一种,即 a、b 呢?也就是说,a 中存放的是大的整数,b 中存放的是小的整数。那么,建立的算法模型为:当 a<b 时,就需要交换 a 和 b 的值。

根据分析的这两种思路,都可以编写源程序。下面选择根据后者的思路编写源程序。

源程序的编写步骤:

(1) 输入两个整型数据 a、b——用 scanf 函数。

(2) 当 a<b 时,交换 a 和 b 的值——用 if 语句“if(表达式) 语句;”的格式。其中表达式是 a<b,语句的操作是实现 a 和 b 的交换。怎么交换两个变量的值呢?设有两个杯子,一个编号为 a,装有苹果汁,另一个编号为 b,装有草莓汁。很显然,根据要求,要把其中一个杯子倒空才能装另一个杯子中的果汁,因此,需要借助中间杯子 t。有了杯子 t,就可以将杯子 a 中的果汁倒入杯子 t,然后将杯子 b 中的草莓汁倒入杯子 a,最后将杯子 t 中的果汁倒入杯子 b。这样,就实现了两个杯子果汁的互换。借助这种思想,也就可以实现两个数的互换了。用 C 语言中的语句描述如下:

将杯子 a 中的果汁倒出杯子 t⇔t=a;

将杯子 b 中的草莓汁倒入杯子 a⇔a=b;

将杯子 t 中的果汁倒入杯子 b⇔b=t。

(3) 输出两个整型数据 a、b——用 printf 函数。

源程序:

```
/*5-6.c*/
#include "stdio.h"
int main(void)
{
    int a,b,t;
    scanf("%d%d",&a,&b);                    /*输入两个整型数据a、b*/
    if(a<b)                                 /*当a<b时,交换a和b的值*/
    {
       t=a;
       a=b;
       b=t;
     }
     printf("a=%d,b=%d\n",a,b);             /*输出两个整型数据a、b*/
     return 0;
 }
```

程序运行结果如下。

```
20 30↙
a=30,b=20
```

2. if 语句实现双分支结构

从 if 语句实现单分支结构可以看出,当条件表达式为真时就执行对应操作,否则什么也不做,程序直接执行其他语句。若不管表达式是否为真都要去执行对应的操作,怎么办呢?这时,可以使用 if 语句的双分支结构形式。孟子在《鱼我所欲也》说道,“鱼,我所欲也,

熊掌亦我所欲也；二者不可得兼，舍鱼而取熊掌者也。”鱼和熊掌的故事就是双分支选择结构的执行过程，即鱼和熊掌必须二选一。

if 语句实现双分支结构的一般格式为：

```
if (表达式)
     语句 1;
  else
     语句 2;
```

其含义是当表达式值为真(非 0)时，执行语句 1，否则执行语句 2。其中表达式、语句 1 和语句 2 与 if 实现单分支结构中的表达式和语句含义相同。

例如：

```
if (x > y)                          /* 如果 x > y,则输出 x,否则输出 y */
     printf("%d",x);
  else
     printf("%d",y);
```

图 5-2　双分支 if 语句 N-S 图

if 语句实现双分支结构执行过程的流程图如图 5-2 所示。

【例 5-7】　编写判断某年是否为闰年的程序。

算法分析：

设要判断的某年用变量 year 来表示，最终的输出结果有两种情况，即是闰年和不是闰年。如果是闰年就输出“是闰年”，否则就输出“不是闰年”。因此，可以用 if 语句实现双分支结构的形式。那么条件表达式就是判断闰年的表达式，如何判断某年是否为闰年呢？判断是否是闰年的条件要符合以下两个条件之一。

① 能被 4 整除，但不能被 100 整除。

② 能被 4 整除，又能被 400 整除。

对于这个条件，可以用逻辑表达式表示为：(year%4==0&&year%100!=0)||year%400==0。

根据以上分析，建立的算法模型为：如果上述逻辑表达式为真则输出“year 是闰年”，否则输出“year 不是闰年”。

源程序的编写步骤：

(1) 输入数据 year——用 scanf 函数。

(2) 输出 year 是闰年或不是闰年——用 if 语句“if(表达式)语句 1；else 语句 2；”的格式，其中表达式是“(year%4==0&&year%100!=0)||year%400==0”，语句 1 的操作输出“year 是闰年”，语句 2 的操作输出“year 不是闰年”。

(3) 输出信息——用 printf 函数。

源程序：

```
1  /* 5-7.c */
2  #include <stdio.h>
3  int main(void)
4  {
5     int year;
6     scanf("%d",&year);                                  /* 输入要判断的年份 */
7     if (year % 4 == 0&&year % 100!= 0||year % 400 == 0)  /* 闰年判断 */
8           printf("%d 是闰年\n",year);                   /* 满足条件输出是闰年 */
```

```
    else
        printf("%d 不是闰年\n",year);     /* 否则输出不是闰年 */
    return 0;
}
```

程序运行结果如下。

```
2023 ↙
2023 不是闰年
```

3. 嵌套的 if 语句实现多分支结构

if 语句实现单分支结构可以根据条件表达式决定是否执行操作,if 语句实现双分支结构可以根据条件表达式决定执行两种操作之一。在这种基本形式中,语句都可以是复合语句。当复合语句中又包含 if 语句时,称为 if 语句的嵌套。if 语句的嵌套就是指条件成立或不成立后执行的语句仍然是一条 if 语句。这样,就需要进行多层判断方能执行其对应的操作。

if 语句的嵌套有以下几种不同的形式,如图 5-3 所示。

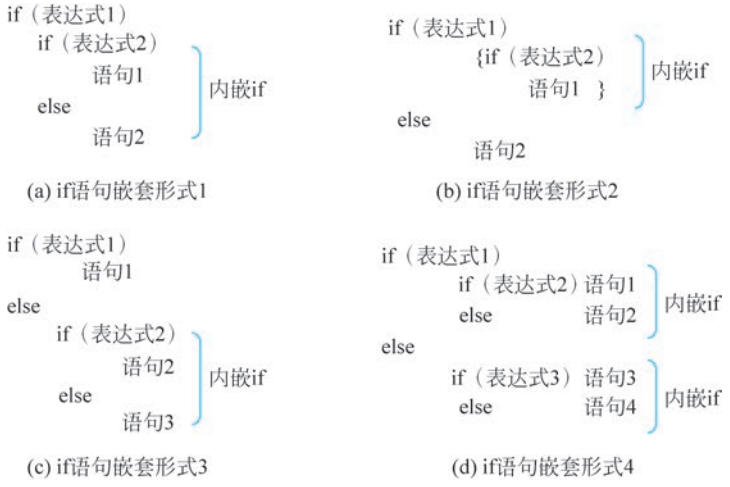

图 5-3 if 语句的嵌套形式

下面就以“if 语句嵌套格式 4”为例,介绍执行过程。if 语句嵌套格式 4 为:

```
if(表达式 1)
        if (表达式 2)
              语句 1;
        else  语句 2;
else
        if (表达式 3)
              语句 3;
        else 语句 4;
```

在该嵌套形式中,当表达式 1 和表达式 2 同时为真时,执行语句 1;当表达式 1 为真,表达式 2 为假时,执行语句 2,其执行过程如图 5-4 所示。

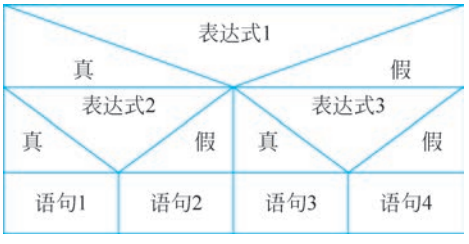

图 5-4 嵌套 if 语句 N-S 图

例如,以下又是 if 语句嵌套的另一种形式。

```
if(score == 100) printf("A");                    /* 成绩为 100 分时,输出等级 A */
  else if(score >= 90) printf("B");              /* 成绩为 90～99 分时,输出等级 B */
    else if(score >= 80) printf("C");            /* 成绩为 80～89 分时,输出等级 C */
        else if(score >= 70) printf("D");        /* 成绩为 70～79 分时,输出等级 D */
           else if(score >= 60) printf("E");     /* 成绩为 60～69 分时,输出等级 E */
               else printf("F");                 /* 成绩为 60 分以下时,输出等级 F */
```

在使用 if 语句的嵌套结构时,要注意 else 与 if 的配对。C 语言规定,else 总是与离它最近且尚未配对的 if 配对。因此,在写 if 嵌套语句时最好把嵌套的 if 语句用“{ }”括起来,以免出现 if、else 匹配出错及阅读困难的现象,并将程序源代码排成锯齿状,形成明显的结构层次,如图 5-5 所示。

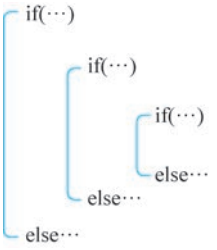

图 5-5　if 语句配对

【例 5-8】　输入一个百分制成绩,输出其等级。等级划分如下:A(90～100 分)、B(80～89 分)、C(70～79 分)、D(60～69 分)和 E(60 分以下)。

算法分析:

设百分制成绩用变量 score(0<=score<=100)表示,等级用变量 grade 表示,最终输出变量 grade 的值。变量 grade 的值是'A'、'B'、'C'、'D'或'E',到底是哪个值,取决于变量 score 符合哪种情况。从这里可以看出,情况有多种,因此,可以建立以下选择结构的多种不同的算法模型。

(1) 用 if 语句的单分支结构形式。依次列出变量 score 的各种情况,得出对应的 grade 的值,代码段如下。

```
if(score >= 90&&score <= 100) grade = 'A';       /* 成绩为 90～100 分时,grade 为 A */
if(score < 90&&score >= 80) grade = 'B';         /* 成绩为 80～89 分时,grade 为 B */
if(score < 80&&score >= 70) grade = 'C';         /* 成绩为 70～79 分时,grade 为 C */
if(score < 70&&score >= 60) grade = 'D';         /* 成绩为 60～69 分时,grade 为 D */
if(score < 60)   grade = 'E';                    /* 成绩为 60 分以下时,grade 为 E */
```

(2) 用 if 语句的嵌套形式 3。首先判断变量 score 是否为 90～100 分,符合则得出等级('A');否则,判断变量 score 是否为 80～89 分,符合则得出等级('B');否则,判断变量 score 是否为 70～79 分,符合则得出等级('C');否则,判断变量 score 是否为 60～69 分,符合则得出等级('D');否则,等级就是('E'),代码段如下。

```
if(score >= 90&&score <= 100)     grade = 'A';   /* 成绩为 90～100 分时,grade 为 A */
  else if(score >= 80)    grade = 'B';           /* 成绩为 80～89 分时,grade 为 B */
    else if(score >= 70)   grade = 'C';          /* 成绩为 70～79 分时,grade 为 C */
       else if(score >= 60) grade = 'D';         /* 成绩为 60～69 分时,grade 为 D */
          else   grade = 'E';                    /* 成绩为 60 分以下时,grade 为 E */
```

(3) 用 if 语句嵌套形式 4。首先判断变量 score 是否为 80～100 分,符合则判断变量 score 是否为 90～100 分,符合则得出等级('A'),否则得出等级('B');均不符合则判断变量 score 是否为 70～79 分,符合得出等级('C');否则判断变量 score 是否为 60～69 分,符合则得出等级('D');否则等级就是('E'),代码段如下。

```
if(score >= 80&&score <= 100)
     if(score >= 90)    grade = 'A';             /* 成绩为 90～100 分时,grade 为 B */
          else grade = 'B';                      /* 成绩为 80～89 分时,grade 为 B */
  else   if(score >= 70)   grade = 'C';          /* 成绩为 70～79 分时,grade 为 C */
```

```
    else  if(score>=60) grade='D';        /*成绩为60～69分时,grade为D*/
      else  grade='E';                    /*成绩为60分以下时,grade为E*/
```

其实,除了以上3种不同的方法外,可以根据方法(2)和方法(3)派生出其他类似的方法。不管用什么方法,尤其是使用if语句嵌套格式时,一定要注意条件的准确表示。实质上,二分支和多分支都可以转换成多个单分支。有时为了方便或减少书写条件时的重复,往往选择用多分支或二分支。

源程序的编写步骤:

(1) 输入成绩score——用scanf函数。

(2) 选取合适的方法进行成绩判断,得到对应的等级,代码段见算法分析。

(3) 输出等级grade——用printf函数或putchar函数。

源程序:

```
1  /*5-8.c*/
2  #include<stdio.h>
3  int main(void)
4  {
5   int score;
6   char grade;
7   scanf("%d",&score);                       /*输入成绩score*/
8   if(score>=90&&score<=100)grade='A';       /*用方法(2)进行成绩判断,得到对应等级 */
9     else if(score>=80) grade='B';
10      else if(score>=70) grade='C';
11         else if(score>=60) grade='D';
12            else grade='E';
13   printf("%c\n",grade);                     /*输出等级grade*/
14   return 0;
15   }
```

程序运行结果如下。

```
80↙
B
```


视频讲解

5.2.2　switch 语句

在例5-8中,可以发现条件中蕴含了一个规律,如等级'A'的分数段是90～100,除100之外,其他分数的最高位都为9;等级'B'的分数段是80～89,分数的最高位都为8;等级'C'的分数段是70～79,分数的最高位都为7;等级'D'的分数段是60～69,分数的最高位都为6;等级'E'的分数段是60分以下,分数的最高位可以是5、4、3、2、1和0。像这种分支结构中,若条件比较多且蕴含了一定的规律,用if语句的多分支结构可以完成,但嵌套的层数较多,程序显得冗长且可读性差。因此,对于多分支结构除了用if语句外,C语言还提供了另一条语句——switch语句。switch语句又称为开关语句。

switch语句用来实现多分支结构,它在使用时的一般格式为:

```
switch(表达式)
{
     case  值1:语句组1; break;
     case  值2:语句组2; break;
           …
```

```
    case  值 n:语句组 n; break;
    default:语句组;
}
```

其执行过程如图 5-6 所示。

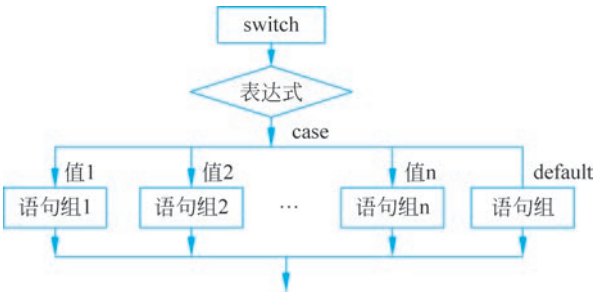

图 5-6　switch 语句的执行过程

首先计算表达式的值，然后用此值与各个 case 后面的值进行比较，若相等则执行该 case 后面的语句组；若与 case 后面的值都不相等，则执行 default 后面的语句组。

实际上，正确理解 switch 语句的执行，应该找到 switch 语句的入口和出口。通常情况下，当表达式与 case 后的值相等的地方称为 switch 语句的入口，否则执行 default 后的语句；如遇 break 语句，则跳出 switch 语句，即为出口，否则将依次执行后面的语句，直至完全执行完结束。

【例 5-9】　输入一个百分制成绩，输出其等级。等级划分及条件满足如下：

A(90～100)、B(80～89)、C(70～79)、D(60～69)和 E(60 以下)。

算法分析：

前面已经分析过，设百分制成绩用变量 score(0＜＝score＜＝100)表示，等级用变量 grade 表示，最终输出变量 grade 的值。变量 grade 的值是'A'、'B'、'C'、'D'或'E'，到底是哪个值，取决于变量 score 符合哪种情况。从这里可以看出，情况有多种，本题用 switch 语句来实现多分支结构。

switch 语句中 case 后面的值就是每一种情况的反映，也就是分数的最高位。要得到分数的最高位可以用表达式 score/10 得到。因此，用 switch 语句实现此分支结构的算法模型为：

```
switch(score/10)                    /* 开关语句 switch */
{
    case 10:grade = 'A';break;      /*  score/10 == 10 时,等级(grade)为'A' */
    case 9:grade = 'A';break;       /*  score/10 == 9 时,等级(grade)为'A' */
    case 8:grade = 'B';break;       /*  score/10 == 8 时,等级(grade)为'B' */
    case 7:grade = 'C';break;       /*  score/10 == 7 时,等级(grade)为'C' */
    case 6:grade = 'D';break;       /*  score/10 == 6 时,等级(grade)为'D' */
    default:grade = 'E';            /*  score/10 为其他值时,等级(grade)为'E' */
}
```

源程序的编写步骤：

(1) 输入成绩 score——用 scanf 函数。

(2) 用 switch 实现成绩判断，得出对应等级。代码段见算法分析。

(3) 输出等级 grade——用 printf 函数或 putchar 函数。

源程序：

```
/*5-9.c*/
#include<stdio.h>
int main(void)
{
 int score;
 char grade;
 scanf("%d",&score);                    /*输入成绩*/
 if(score>=0 && score<=100)
 {
   switch(score/10)                     /*开关语句 switch*/
   {
     case 10:grade='A';break;
     case 9:grade='A';break;
     case 8:grade='B';break;
     case 7:grade='C';break;
     case 6:grade='D';break;
     default:grade='E';
   }
    printf("%c\n",grade);               /*输出等级*/
  }
    else printf("成绩输入有误!\n");
  return 0;
}
```

程序运行结果如下。

```
92↙
A
```

在使用 switch 语句时，要注意以下几点。

(1) 每个 case 后的值为常量，该值的类型应与 switch 后面圆括号内表达式的类型一致，表达式的值只能是整型、字符型或枚举型。

(2) 关键字 case 与常量中间至少有一个空格，常量后有一个冒号。

(3) 每个“case 常量”后面可以跟任意数量的语句，无须用大括号标识这些语句。

(4) 每个 case 后面的值必须互不相同。

(5) case 和 default 的位置是任意的，default 子句可以省略。

(6) case 值只起语句标号作用，不进行条件判断，在执行完某个 case 后面的语句后，将自动执行该语句后面的语句。因此，在执行一个 case 分支后，要使流程跳出 switch 结构，即终止 switch 语句的执行，可以在语句组后面加上一条 break 语句。

(7) 当若干 case 后执行的语句相同时，可以将这若干 case 连续写在一起，保留最后一个 case 后执行的语句。此时，case 后面执行的语句被省略时，冒号不能省略。

例如：

```
switch(s)
{
    case 10:
    case 9:grade='A';break;
   …
}
```

5.3　选择结构程序设计的典型应用


视频讲解

本节主要讨论以下问题。

(1) 选择结构程序设计的算法分析如何进行?

(2) 如何编写选择结构程序?

选择结构的程序执行是按照条件的真假选择执行,实现选择结构的语句有 if 语句和 switch 语句等,能够实现单分支结构、双分支结构和多分支结构。一般来说,只要涉及条件处理问题就需要使用选择结构。下面主要通过数的最值、方程根、奖金和运算器等问题介绍如何进行选择结构程序设计。

5.3.1　数的最值问题

数的最值问题是计算机程序设计中经常会遇到的问题,如求两个整数中的最大值或最小值,求 3 个整数中的最大值或最小值以及求 n 个整数中的最大值或最小值。当然,数不仅仅可以是整数,也可以是其他类型数据,如若干字符串、结构体类型的数据等。

1. 问题描述

求 3 个整数中的最大值。

2. 算法分析

根据问题的描述可以得知,此题中涉及 4 个数据,即 3 个整数和最大值。设 3 个整数分别用变量 a、b、c 表示,最大值用变量 max 表示。求最大值 max 的算法模型如下。

首先,置 max 初值为 a。也就是说,当只有一个数据 a 时,最大值 max 就是 a。

然后,将 b 与 max 进行比较。若 b>max,则 max 的值为 b。至此,得到两个数 a 和 b 的最大值。

最后,将 c 与 max 进行比较。若 c>max,则 max 的值为 c。至此,得到 3 个数 a、b 和 c 的最大值。

3. 源程序的编写步骤

(1) 输入 3 个整数 a、b、c——用 scanf 函数。

(2) 分别用 if 语句的单分支结构实现 b 与 max、c 与 max 的比较。

(3) 输出最大值 max——用 printf 函数。

4. 源程序

```
1  /* 数的最值问题.c */
2  #include <stdio.h>
3  int main(void)
4  {
5      int a,b,c,max;
6      scanf("%d%d%d",&a,&b,&c);
7      max=a;
8      if (max<b) max=b;
9      if (max<c) max=c;
10     printf("%d\n",max);
11     return 0;
12 }
```

程序运行结果如下。

```
35 20 60↙
60
```

【举一反三】

(1) 求3个整数中的最小值。要求用多种方法完成。

(2) 求4个整数中的最小值。要求用三条if-else语句完成。

(3) 将4个整数按照从小到大的顺序输出。

5.3.2 方程根问题

1. 问题描述

设方程为$ax^2+bx+c=0$,其中a、b和c为方程的系数。要求输入该方程的3个系数并计算输出该方程的根。

2. 算法分析

根据问题的描述,了解该问题中涉及的数据主要有3个系数,分别用变量a、b和c表示,一个判别式用变量beta表示,两个根用变量x1和x2表示。方程的根是哪一种情况,取决于系数a和b以及根的判别式。若是一元二次方程,则根有3种情况:两个相等实根、两个不相等实根和两个虚根。若不是一元二次方程,则根的情况又由系数b来确定,若$b\neq0$则有一个根,否则无解。

根据以上分析,建立的算法模型如下。

根据输入的系数a是否为0来确定是否为一元二次方程。

(1) 若是一元二次方程,则根据beta与0的关系确定方程根的情况,共有3种情况,它们分别是:当beta=0时,有两个相等实根,这两个实根为x1=x2=-b/2a;当beta>0时,有两个不相等实根,这两个实根分别是为$x1=\frac{-b+\sqrt{b^2-4ac}}{2a}$,$x2=\frac{-b-\sqrt{b^2-4ac}}{2a}$;当beta<0时,有两个不相等的虚根,这两个虚根分别是为$x1=\frac{-b+\sqrt{|b^2-4ac|}i}{2a}$,$x2=\frac{-b-\sqrt{|b^2-4ac|}i}{2a}$。通过此分析,实际上也可以分为两种情况,beta≥0和beta<0。第一种情况的根是实根,直接求出和输出;第二种情况的根是虚根,虚根由实部和虚部两部分组成。因此,在求虚根时,要分别求出实部和虚部,实部用m表示,$m=\frac{-b}{2a}$,虚部用n表示,$n=\frac{\sqrt{|b^2-4ac|}}{2a}$,这两个虚根用x1和x2表示,分别是x1=m+ni,x2=m-ni。在输出虚根时要分别输出实部和虚部,这两部分之间需要用符号“+”或“-”连接,且虚部后面带一个字母i。

(2) 若不是一元二次方程,方程根的情况又有两种:若$b\neq0$则有一个根,否则无解。

3. 源程序的编写步骤

(1) 输入3个系数a、b、c——用scanf函数。

(2) a!=0时,该方程为一元二次方程,求根的判别式beta,$beta=b^2-4ac$,其根的情况

如下所述。

① 求实根并输出。代码为：

```
if(beta>=0)
  if(beta>0)
 {
   x1=(-b+sqrt(beta))/(2*a);
   x2=(-b-sqrt(beta))/(2*a);
   printf("x1=%.2f,x2=%.2f\n",x1,x2);
  }
 else
 {
   x=-b/(2*a);
   printf("x1=x2=%.2f\n",x2);
  }
```

② 求虚根并输出。代码为：

```
if(beta<0)
{
  m=-b/(2*a);
  n=sqrt(fabs(beta))/(2*a);
  printf("x1=%.2f+%.2fi,x2=%.2f-%.2fi\n",m,n,m,n);
}
```

(3) 当 a==0 时，该方程为一元一次方程，其根的情况如下所述。

```
if(b!=0) printf("%.2f\n",-c/b);
    else  printf("无解\n");
```

4. 源程序

```
1  /*方程根问题.c*/
2  #include<stdio.h>
3  #include<math.h>
4  int main()
5  {
6   int a,b,c;
7   float beta,x1,x2,m,n;
8   scanf("%d%d%d",&a,&b,&c);          /*输入3个系数a、b、c */
9   beta= b*b-4*a*c;                   /*求根的判别式 beta */
10  if(a!=0)                           /* a!=0, 一元二次方程 */
11    if(beta>=0)                      /* beta>=0 */
12       if(beta>0)                    /* beta>0,有两个不等的实根 x1、x2 */
13        {
14         x1=(-b+sqrt(beta))/(2*a);
15         x2=(-b-sqrt(beta))/(2*a);
16         printf("x1=%.2f,x2=%.2f\n",x1,x2);
17        }
18         else                        /*beta=0,有两个相等的实根 x1、x2 */
19          {
20           x1=x2=-b/(2*a);
21           printf("x1=x2=%.2f\n",x2);
22           }
23     else                            /*beta<0,有两个虚根 x1、x2*/
24      {
25        m=-b/(2*a);
26        n=sqrt(fabs(beta))/(2*a);
```

```
        printf("x1 = %.2f + %.2fi,x2 = %.2f - %.2fi\n",m, n,m,n);
      }
else                  /*a==0,一元一次方程*/
   if(b!=0) printf("%.2f\n", -(float)c/b);
       else printf("无解\n");
return 0;
}
```

程序在不同情况下的运行结果如下(共执行5次)。

```
3 4 5↙
x1 = 0.00 + 1.11i,x2 = 0.00 - 1.11i
5 7 2↙
x1 = -0.40,x2 = -1.00
2 8 8↙
x1 = x2 = -2.00
0 0 2↙
无解
0 1 3↙
-3.00
```

【举一反三】

(1) 输入三条线段的长度,判断是否能组成一个三角形,如能则求该三角形的面积。

(2) 输入三条线段的长度,判定它们能否构成一个三角形。如果能构成三角形,则打印它们所构成三角形的名称,包括等边、等腰、直角或任意三角形。

5.3.3 奖金问题

1. 问题描述

某车间按工人加工零件的数量发放奖金,奖金分为5个等级:每月加工零件数n<1000者奖金为100元;1000≤n<1100者奖金为300元;1100≤n<1200者奖金为500元;1200≤n<1300者奖金为700元;n≥1300者奖金为900元,如表5-3所示。从键盘输入2人加工零件数量,显示应发奖金数。

表5-3 问题描述列表

加工零件数n	奖金/元
n<1000	100
1000≤n<1100	300
1100≤n<1200	500
1200≤n<1300	700
n≥1300	900

2. 算法分析

根据问题描述,该问题中涉及的数据主要有:加工零件数用变量n表示,奖金数用变量p表示,问题程序的实现需要使用选择结构语句。若用if语句,就需要把条件表达式用关系运算符或逻辑运算符表达出来,但通过观察条件发现,其蕴含着一定的规律。因此,像这种选择结构问题的实现往往使用switch语句。

为了使用switch语句必须将加工零件数n与奖金的关系转换成某些整数与奖金的关

系。分析本题可知，加工零件数都是 100 的整数倍（1000、1100、1200…），因此，将加工零件数 n 整除 100，即缩小为原来的百分之一，则：

n＜1000	对应 0、1、2、3、4、5、6、7、8、9
1000≤n＜1100	对应 10
1100≤n＜1200	对应 11
1200≤n＜13000	对应 12
1300≤n	对应 13、14、15…

根据以上分析，建立的算法模型如下：

```
switch(n/100)                                   /* 开关语句 switch */
{
 case 0: case 1: case 2: case 3: case 4: case 5:
 case 6: case 7: case 8: case 9: p = 100;break;  /* 当 n < 1000 时，奖金 p = 100 */
 case 10: p = 300;break;                         /* 当 1000≤n < 1100 时，奖金 p = 300 */
 case 11: p = 500;break;                         /* 当 1100≤n < 1200 时，奖金 p = 500 */
 case 12: p = 700;break;                         /* 当 1200≤n < 13000 时，奖金 p = 700 */
 default: p = 900;                               /* 当 1300≤n 时，奖金 p = 900 */
 }
```

3. 源程序的编写步骤

（1）输入工人加工零件的数量 n——用 scanf 函数。

（2）将加工零件数 n 除以 100，得到一个值为 t，t＝n/100。

（3）根据算法分析的规则，用 switch 语句，根据 t 值确定工人得到的资金数 p，代码段如下：

```
switch(t)                                       /* 开关语句 switch */
{
 case 0:
 case 1:
 case 2:
 case 3:
 case 4:
 case 5:
 case 6:
 case 7:
 case 8:
 case 9: p = 100;break;                         /* 当 t < 10 时，奖金 p = 100 */
 case 10: p = 300;break;                        /* 当 t = 10 时，奖金 p = 300 */
 case 11: p = 500;break;                        /* 当 t = 11 时，奖金 p = 500 */
 case 12: p = 700;break;                        /* 当 t = 12 时，奖金 p = 700 */
 default: p = 900;                              /* 当 t >= 13 时，奖金 p = 900 */
 }
```

（4）输出资金 p——用 printf 函数。

4. 源程序

```
1 /* 奖金问题.c */
2 #include <stdio.h>
3 int main(void)
4 {
5    int n,p,t;
6    scanf("%d",&n);                    /* 输入加工零件数量 */
7    if(n >= 0)
```

```
8   {
9     t = n/100;                          /* 将数量缩小为原来的百分之一 */
10    switch(t)                           /* 开关语句 switch */
11    {
12      case 0:
13      case 1:
14      case 2:
15      case 3:
16      case 4:
17      case 5:
18      case 6:
19      case 7:
20      case 8:
21      case 9: p = 100;break;            /* 当 t < 10 时,奖金 p = 100 */
22      case 10: p = 300;break;           /* 当 t = 10 时,奖金 p = 300 */
23      case 11: p = 500;break;           /* 当 t = 11 时,奖金 p = 500 */
24      case 12: p = 700;break;           /* 当 t = 12 时,奖金 p = 700 */
25      default: p = 900;                 /* 当 t >= 13 时,奖金 p = 900 */
26    }
27    printf("应发奖金:%d\n",p);          /* 输出奖金 */
28  }
29     else printf("输入有误!\n");
30   return 0;
31 }
```

程序在不同情况下的运行结果如下(共执行 5 次)。

```
5000↙
应发奖金:900
1050↙
应发奖金:300
1105↙
应发奖金:500
1210↙
应发奖金:700
700↙
应发奖金:100
```

【举一反三】

(1) 编写一个程序,输入年份和月份,判断该年是否为闰年,并根据给出的月份判断是什么季节。

(2) 选择输入长方形、圆形以及三角形中的一种,并输入长方形的长宽或圆形的半径或三角形的三条边长,输出其面积。

5.3.4 运算器问题

1. 问题描述

从键盘上输入任意两个数和一个运算符(+、-、*、/),计算其运算的结果并输出。

2. 算法分析

根据问题描述,该问题中涉及的数据主要有:两个操作数用变量 a 和 b 表示,运算符用变量 op 表示,运算结果用变量 result 表示。从问题的描述可以看出,该问题需要根据输入的不同运算符来决定计算结果,因此程序中考虑使用选择结构的 switch 语句来实现这个过

程，不同的 case 中对应不同的运算符来进行计算。

根据以上分析，建立的算法模型如下：

```
switch ( op )
 {
  case  '+':  result = a + b;   break;
  case  '-':   result = a - b;    break;
  case  '*':  result = a * b;   break;
  case  '/':    result = a / b;    break;
  default:   printf ("illegal arithmetic lable.\n");
 }
```

其中，在除法运算中，为了保证除数为 0 时不出现错误结果，还要作进一步判断。

另外，如果需要计算器可以重复被使用，则可以定义一个字符变量作为是否结束使用计算器的标志变量，此时需要使用循环结构来实现这个过程。

3. 源程序的编写步骤

(1) 输入两个操作数 a、b 和运算符 op——用 scanf 函数。

(2) 根据运算符 op 进行相应的运算，在进行除法运算时，判别除数是否为 0，若为 0，运算非法，给出相关提示信息。若运算符号不是＋、－、＊、/则同样是非法的，也给出相应提示信息。此问题实现时，用 tag 作为标志位，判断是否合法。tag＝0 时为合法，tag＝1 时为非法。代码段如下：

```
switch ( ch )
 {
  case  '+':  result = a + b;   break;
  case  '-':   result = a - b;    break;
  case  '*':  result = a * b;   break;
  case  '/':    if (!b)
                  {
                     printf ("divisor is zero!\n");
                     tag = 1;
                  }
                else
                   result = a / b;
                break;
 default:   printf ("illegal arithmetic lable.\n");
          tag = 1;
}
```

(3) 合法情况时，输出运算结果 result，用 if 语句和 printf 函数。

```
if (!tag)
   printf ("%.2f\n", result);
```

4. 源程序

```
1  /*运算器问题.c*/
2  #include <stdio.h>
3  int main ( )
4  {
5    float a, b;
6    int tag = 0;
7    char op;
8    float result;
```

```
    scanf ("%f%f", &a, &b);
    getchar();
    scanf ("%c", &op);
    switch ( op )
    {
     case '+': result = a + b; break;
     case '-': result = a - b; break;
     case '*': result = a * b; break;
     case '/': if (!b)
               {
                 printf ("divisor is zero!\n");
                 tag = 1;
                 }
                else
                  result = a / b;
               break;
    default:  printf ("illegal arithmetic lable.\n");
              tag = 1;
    }
    if (!tag)
       printf ("%.2f\n",result);
    return 0;
}
```

程序在不同情况下的运行结果如下(共执行 6 次)。

```
6.5 7.8↙
-↙
-1.30
6.5 7↙
+↙
13.50
7 8↙
*↙
56.00
7 0↙
/↙
divisor is zero!
6.89 9↙
/↙
0.77
3.5 2.9↙
)↙
illegal arithmetic lable.
```

【举一反三】

有一个函数,定义如下:

$$f(x)=\begin{cases}x^2+x-6 & x<0,x\neq-3\\ x^2-5x+6 & 0\leqslant x<10,x\neq 2,x\neq 3\\ x^2-x-1 & \text{其他}\end{cases}$$

编写一个程序,输入 x,输出 y。要求分别用 if 和 switch 语句完成。

5.4　本章小结

5.4.1　知识梳理

C 语言程序的执行部分是由语句组成的。程序的功能也是由执行语句实现的。C 语言中的语句分为表达式语句、函数调用语句、复合语句、空语句及控制语句五类。

关系表达式和逻辑表达式是两种重要的表达式，主要用于条件执行的判断和循环执行的判断。

C 语言提供了多种形式的条件语句以构成选择结构：if 语句主要用于单分支选择；if-else 语句主要用于双分支选择；if-else-if 语句和 switch 语句用于多分支选择。

任何一种选择结构都可以用 if 语句来实现，但并非所有的 if 语句都有等价的 switch 语句。switch 语句只能用来实现以相等关系作为判断条件的选择结构。

在调试包含选择结构的程序时，为了保证程序的完备性，必须保证选择结构的各个分支情况都要能正确执行。

本章知识导图如图 5-7 所示。

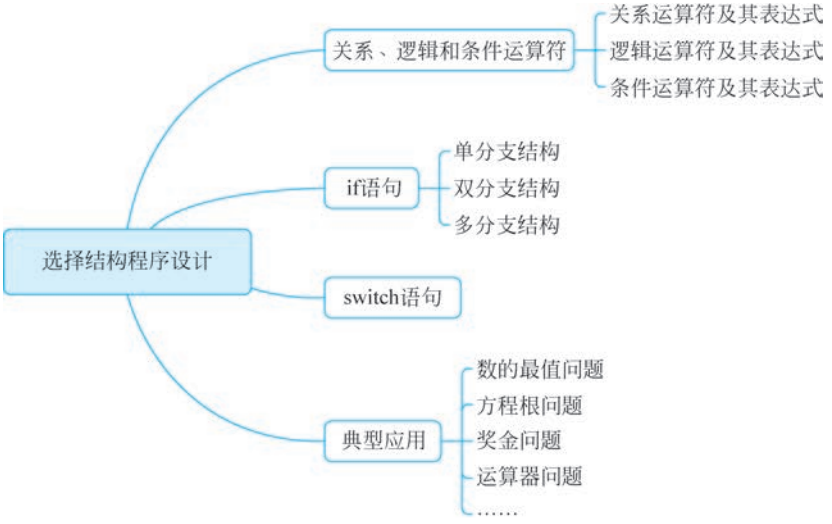

图 5-7　本章知识导图

5.4.2　常见上机问题及解决方法

1. 在关系表达式中误用“=”来代替“==”

可能由于代数中“=”表示相等关系，或者在输入时粗心所致，很多程序员使用“=”号来代替“==”运算符，特别是在 if、while、for 语句中更是常见。例如：

```
if(i = 2)  printf("i is 2");
k = 1;
while(!(k = 10))
{
```

```
  printf("%d",k);
  k++;
  }
```

2. case 语句漏掉 break

初学者在使用 switch 语句时经常会忘记在 case 语句后面增加一条 break 语句。由于没有 break 语句,switch 语句执行后与预想的结果有很大的不同。例如:

```
#include"stdio.h"
int main()
{
   char c;
   c=getch();
   switch(c)
   {
     case 'a': printf("A");
     case 'b': printf("B");
     default: printf("OK");
   }
   return 0;
}
```

上面的程序运行时,由于 switch 语句中没有 break 语句,因而如果输入'a',会输出 ABOK; 如果输入'b',会输出 BOK,与程序员的设计意图不符。

3. if 语句后多了“;”

由于 C 语言的每条语句都以“;”结尾,因此有些初学者在输入 if 语句时不经意间也会在 if 语句的()后面增加一个分号,例如:

```
if(k>10);                                    /*多了一个;*/
printf("k>10");
```

程序员本来的意图是当 k 大于 10 时输出 k>10,但由于 if()后面多了一个分号,这时,程序没有错误但表示满足条件时不执行任何操作,运行结果产生异常,表示不管是否满足条件都会输出 k>10。

4. 复合语句漏掉了“{ }”

如果 if、while、for 语句包含多条语句,那么就需要使用复合语句。例如,下面的 if 语句的功能是如果变量 k 的值大于 10,则输出 k 的值并将 k 值减 5。

```
scanf("%d",&k);
if(k>10)
{
  printf("%d",k);
  k-=5;
}
```

如果 if 语句漏掉了“{}”,则程序段的功能便不同了。

```
scanf("%d",&k);
if(k>10)
   printf("%d",k);
k-=5;
```

上面的程序段中,无论 k 的值是多少,k 的值都会被减 5。

5. 表达式中“()”不配对,复合语句中“{}”不配对

“()”或“{}”的不配对有时很隐蔽,有时引起的编译错误很奇怪。例如:

```
while((c = getch()!= 27)                          /* 漏掉了) */
   putchar(c);
while((c = getch())!= 27)
{
   c += 26;
   printf("%c",c);
/* 漏掉了} */
```

6. case 后面跟着变量表达式

switch 语句中 case 后面必须是常量表达式，不能是包含有变量的表达式。下面的用法是错误的。

```
char c1,c2;
scanf("%c,%c",&c1,&c2);
switch(c2)
{
 case c1 - 1:                                     /* case 后面跟着变量表达式 */
      printf("c2 = c1 - 1");
      break;
 case c1:                                         /* case 后面跟着变量表达式 */
      printf("c2 = c1");
      break;
case c1 + 1:                                      /* case 后面跟着变量表达式 */
      printf("c2 = c1 + 1");
      break;
}
```

上面程序段的意图是判断输入的两个字符是否相邻或相等，因此不能直接使用 switch 语句，可以使用 if-else 语句，或者利用 switch 语句判断 c2-c1 表达式的值。

7. 两个关系表达式连用

代数中可以这样来表达一种关系：10<x<100。但这种表达在 C 语言中便失去了原来的意义。C 语言中两个关系表达式不能连用，只能用“&&”进行连接。即表达 10<x<100 的关系，只能这样来表达：10<x&&x<100。

但 10<x<100 这个表达式并没有语法错误，编译时并不会出现错误。但含义已经变了，10<x 的值是 1 或 0，再比较 1 或 0 与 100 的大小。

8. 将“==”“&&”“||”误输入为“=”“&”“|”

如果要比较 a 和 b 是否相等，应该使用关系表达式 a==b，而不能使用 a=b。a=b 表示将 b 的值赋值给 a。

表示 a>b 并且 c>d，应该使用逻辑表达式 a>b&&c>d，而不能使用 a>b&c>d。

表达式 a>b&c>d 的值是这样计算的：先计算 a>b 的值，然后计算 c>d 的值，最后将两个值进行按位与操作。

要表示 a>b 或者 c>d，应使用逻辑表达式 a>b||c>d，而不能使用 a>b|c>d。

表达式 a>b||c>d 的值是这样计算的：先计算 a>b 的值，然后计算 c>d 的值，最后将两个值进行按位或操作。

9. 用!>=表示不大于或等于

在 C 语言中“!”表示逻辑非，它是单目运算符，即“!”后面只能跟一个表达式。例如，!(a>b)等价于 a<=b，而!0 的值是真，!3 的值是假。

因此,不能这样来表达a不大于或等于b:a!>=b。因为!后面没有跟表达式。a不大于或等于b可以这样表示:!(a>=b)。

10. “==”“!=”“<=”“>=”运算符中多了空格

C语言中“==”“!=”“<=”“>=”运算符由两部分组成的,是一个整体,输入时中间不能有空格,否则就会出现编译错误。例如,这些表达式是非法的:A= =B,A! =B,A> =B。

扩展阅读:程序调试方法和技巧

从这一章开始,编写的程序越来越复杂了,如何保证程序能正确地解决问题呢?当然首先要保证程序正确,然后就要通过测试问题的各种情况以保证能解决问题。经过这两方面的验证,才是符合问题要求的正确程序。

1. 调试程序技巧

程序编写完成后就要通过编译程序验证是否正确。判断一个程序是否正确可以从两个方面进行:一是看程序是否能够得到结果;二是看程序运行后的结果是否符合用户要求。若一个程序能够正常运行且能够得到用户要求的结果,可以说这个程序基本上完成了某项要求的功能,否则就需要对程序进行调试和修改。通过调试,找到程序中出现错误的地方,进行修改。如此反复,直至程序完全正确为止。

程序调试主要有两种方法,即静态调试和动态调试。程序的静态调试就是在程序编写完以后,由人工“代替”或“模拟”计算机,对程序进行仔细检查,主要检查程序中的语法规则和逻辑结构的正确性。实践表明,有很大一部分错误可以通过静态检查来发现。通过静态调试,可以大大缩短上机调试的时间,提高上机的效率。程序的动态调试就是实际上机调试,它贯穿在编译、连接和运行的整个过程中。根据程序编译、连接和运行时计算机给出的错误信息进行程序调试,这是程序调试中最常用的方法,也是最初步的动态调试。在此基础上,通过“分段隔离”“设置断点”“跟踪打印”进行程序的调试。实践表明,对于查找某些类型的错误来说,静态调试比动态调试更有效,对于其他类型的错误来说则刚好相反。因此静态调试和动态调试互相补充、相辅相成,缺少其中任何一种方法都会使查找错误的效率降低。

1) 静态调试法

(1) 对程序语法规则进行检查。

① 语句正确性检查。保证程序中每条语句的正确性是编写程序的基本要求。由于程序中包含大量的语句,书写过程中由于疏忽或笔误,语句写错在所难免。对程序语句的检查应注意以下几点。

- 检查每条语句的书写是否有字符遗漏,包括必要的空格符是否都有。
- 检查形体相近的字符是否书写正确。例如字母o和数字0,书写时要有明显的分别。
- 检查函数调用时形参和实参的类型、个数是否相同。

② 语法正确性检查。每种计算机语言都有自己的语法规则,书写程序时必须遵守一定的语法规则,否则编译时程序将给出错误信息。

- 语句的配对检查。许多语句都配对出现,不能只写半条语句。另外,语句有多重括

号时，每个括号也都应成对出现，不能缺左少右。

- 注意检查语句顺序。有些语句不仅句法本身要正确，而且语句在程序中的位置也必须正确。例如，变量定义要放在所有可执行语句之前。

（2）检查程序的逻辑结构。

① 检查程序中各变量的初值和初值的位置是否正确。我们经常遇到的“累加”和“累乘”，其初值和位置都非常重要。用于累加的变量应取 0 初值或给定的初值，用于累乘的变量应赋初值 1 或给定的值。因为累加或累乘都通过循环结构来实现，因此这些变量赋初值语句应在循环体之外。对于多重循环结构，内循环体中的变量赋初值语句应在内循环之外；外循环体中的变量赋初值语句应在外循环之外。如果赋初值的位置放错了，那么将得不到预想的结果。

② 检查程序中分支结构是否正确。程序中的分支结构都是根据给定的条件来决定执行不同的路径，因此在设置各条路径的条件时一定要谨慎。在设置“大于”和“小于”这些条件时，一定要仔细考虑是否应该包括“等于”这个条件，更不能把条件写反。尤其要注意的是，实型数据在运算过程中会产生误差，如果用“等于”或“不等于”对实数的运算结果进行比较，则会因为误差而产生误判断，路径选择也就错了。因此在遇到要将判断实数 a 与 b 相等与否作为条件来选择路径时，应该把条件写成 if(fabs(a－b)＜＝1e－6)，而不应该写成 if(a＝＝b)。要特别注意条件语句嵌套时，if 和 else 的配对关系。

③ 检查程序中循环结构的循环次数和循环嵌套的正确性。C 语言中可用 for 循环、while 循环、do-while 循环。在给定循环条件时，不仅要考虑循环变量的初始条件，还要考虑循环变量的变化规律、循环变量变化的时间，任何一条变化都会引起循环次数的变化。

④ 检查表达式的合理与否。程序中不仅要保证表达式的正确性，而且还要保证表达式的合理性。尤其要注意表达式运算中的溢出问题，运算数值可能超出整数范围就不应该采用整型运算，否则必然导致运算结果的错误。两个相近的数不能相减，以免产生“下溢”。更要避免在一个分式的分母运算中发生“下溢”，因为编译系统常把下溢做零处理。因此分母中出现下溢时要产生“被零除”的错误。由于表达式不合理而引起的程序运行错误往往很难查找，会增加程序调试的难度。因此，认真检查表达式的合理性，可以减少程序运行的错误，提高程序的动态调试效率。

2）动态调试法

在静态调试中可以发现和改正很多错误，但由于静态调试的特点，有一些比较隐蔽的错误还不能检查出来。只有上机进行动态调试，才能够找到这些错误并改正它们。

（1）编译过程中的调试。

编译过程除了将源程序翻译成目标程序外，还要对源程序进行语法检查。如果发现源程序有语法错误，系统将显示错误信息。用户可以根据这些提示信息查找出错误性质，并在程序中对出错之处进行相应的修改。有时我们会发现编译时有几行的错误信息都是一样的，检查这些行本身时并没有发现错误，这时要仔细检查与这些行有关的名字、表达式是否有问题。例如，因为程序中的数组说明语句有错，这时那些与该数组有关的程序行都会被编译系统检查出错。这种情况下，用户只要仔细分析一下，修改了数组说明语句的错误，许多错误就没有了。对于编译阶段的调试，要充分利用给出的错误信息，对它们进行仔细地分析判断。只要不断积累并总结经验，使程序通过编译是不难做到的。

(2) 连接过程的调试。

编译通过后要进行连接。连接的过程也有查错的功能,它将指出外部调用、函数之间的联系及存储区设置等方面的错误。如果连接时有这类错误,编译系统也会给出错误信息,用户要对这些信息仔细判断,从而找出程序中的问题并改正。连接时较常见的错误有以下几类。

① 某个外部调用有错。通常系统明确提示了外部调用的名字,只要仔细检查各模块中与该名有关的语句,就不难发现错误。

② 找不到某个库函数或某个库文件。这类错误是由于库函数名写错、疏忽了某个库文件的连接等引起。

③ 某些模块的参数超过系统的限制。如模块的大小、库文件的个数超出要求等。

引起连接错误的原因很多,而且很隐蔽,给出的错误信息也不如编译时给出的直接、具体。因此,连接时的错误要比编译错误更难查找,需要仔细分析判断,而且对系统的限制和要求要有所了解。

(3) 运行过程中的调试。

运行过程中的调试是动态调试的最后一个阶段。这一阶段的错误大体可分为以下两类。

① 运行程序时给出出错信息。运行时出错多与数据的输入、输出格式有关,以及与文件的操作有关。如果给出的数据格式有错,这时要对有关的输入输出数据格式进行检查,一般较容易发现错误。如果程序中的输入输出函数较多,则可以在中间插入调试语句,采取分段隔离的方法,很快就可以确定错误的位置了。如果文件操作有误,也可以针对程序中的有关文件的操作采取类似的方法进行检查。

② 运行结果不正常或不正确。这种错误可能与数据的输出格式或算法有关。

2. 调试分支程序技巧

(1) 验证分支结构程序的正确性,在调试程序时必须保证使条件为真和条件为假时的数据都要被输入一次,包括边界条件。

(2) 在 if 语句的定义格式中,不管在哪种情况中,要执行的语句末尾都有一个分号。C语言规定,每一条语句都以分号结束,不能省略。这一点与其他高级语言有所区别。

习题 5

1. 选择题

(1) 有以下程序:

```
#include"stdio.h"
int main(void)
{
   int a=3,b=4,c=5,d=2;
   if(a>b)
     if(b>c)
          printf("%d",d++ +1);
       else
          printf("%d",++d +1);
   printf("%d\n",d);
```

```
return 0;
}
```

程序运行后的输出结果是(　　)。

A. 2　　B. 3　　C. 43　　D. 44

(2)

```
#include "stdio.h"
int main()
{
  int x,y;
  scanf("%d",&x);
  y=0;
  if (x>=0)
    {if (x>0) y=1;}
   else y=-1;
  printf ("%d",y);
  return 0;}
```

当从键盘输入 32 时,程序的输出结果为(　　)。

A. 0　　B. −1　　C. 1　　D. 不确定值

(3) 下列条件语句中,功能与其他语句不同的是(　　)。

A. if(a) printf("%d\n",x); else printf("%d\n",y);

B. if(a==0) printf("%d\n",y); else printf("%d\n",x);

C. if (a!=0) printf("%d\n",x); else printf("%d\n",y);

D. if(a==0) printf("%d\n",x); else printf("%d\n",y);

(4) 以下程序段中与语句"k=a>b?(b>c?1:0):0;"功能等价的是(　　)。

A.
```
if((a>b) &&(b>c) ) k=1
    else k=0;
```

B.
```
if((a>b) ||(b>c) ) k=1;
    else  k=0;
```

C.
```
if(a<=b)  k=0;
    else if(b<=c)  k=1;
```

D.
```
if(a>b)  k=1;
    else if(b>c)  k=1;
      else k=0;
```

(5) 有定义语句"int a=1,b=2,c=3,x;",则以下选项中各程序段执行后,x 的值不为 3 的是(　　)。

A.
```
if (c<a) x=1;
    else if (b<a) x=2;
      else x=3;
```

B.
```
if(a<3) x=3;
      else if (a<2) x=2;
            else x=1;
```

C.
```
if (a<3) x=3;
if (a<2) x=2;
if (a<1) x=1;
```

D.
```
if(a<b) x=b;
if  (b<c) x=c;
if  (c<a) x=a;
```

(6) 有以下程序:

```
#include"stdio.h"
int main(void)
{  int i=1,j=1,k=2;
   if((j++ || k++)&&i++) printf("%d,%d,%d\n",i,j,k);
   return 0;
}
```

执行后输出的结果是(　　)。

A. 1,1,2　　B. 2,2,1　　C. 2,2,2　　D. 2,2,3

(7) 已有定义“int x=3,y=4,z=5;”,则表达式“!(x+y)+z-1 && y+z/2”的值是(　　)。

A. 6　　B. 0　　C. 2　　D. 1

(8) 有以下程序:

```
#include"stdio.h"
int main(void)
{   int a=15,b=21,m=0;
    switch(a%3)
    { case 0:m++;break;
      case 1:m++;
     switch(b%2)
       { default:m++;
         case 0:m++;break;
       }
     }
    printf("%d\n",m);
    return 0;
}
```

程序运行后的输出结果是(　　)。

A. 1　　B. 2　　C. 3　　D. 4

(9) 设a、b、c、d、m、n均为int型变量,且a=5、b=6、c=7、d=8、m=2、n=2,则逻辑表达式(m=a>b)&&(n=c>d)运算后,n的值位为(　　)。

A. 0　　B. 1　　C. 2　　D. 3

(10) 能正确表示逻辑关系“a≥10或a≤0”的C语言表达式是(　　)。

A. a>=10 or a<=0　　B. a>=0|a<=10

C. a>=10 && a<=0　　D. a>=10 || a<=0

(11) 两次运行下面的程序,如果从键盘上分别输入6和4,则输出结果是(　　)。

```
#include<stdio.h>
int  main()
{
  int x;
  scanf("%d",&x);
  if(x++>5) printf("%d",x);
   else printf("%d\n",x--);
  return 0;
}
```

A. 7　5　　B. 6　3　　C. 7　4　　D. 6　4

(12) 以下程序的输出结果是(　　)。

```
#include<stdio.h>
int main()
{   int a=-1,b=4,k;
    k=(++a<0)&&(b--<=0);
    printf("%d%d%d\n",k,a,b);
    return 0;
}
```

A. 104　　B. 003　　C. 103　　D. 004

(13) 能正确表示 a≥10 并且 a≤0 的关系表达式是(　　)。

A. a>=10 or a<=0　　B. a>=10 | a<=0

C. a>=10 && a<=0　　D. a>=10 || a<=0

(14) 假定所有变量均已正确说明,下列程序段运行后 x 的值是(　　)。

```
a = b = c = 0;x = 35;
if(!a)x -- ;
  else if(b);
if(c)x = 3;
  else x = 4;
```

A. 34　　B. 4　　C. 35　　D. 3

(15) 数学表达式 X≤Y≤Z 所对应的 C 语言表达式为(　　)。

A. (X<=Y)&&(Y<=Z)　　B. (X<=Y)and(Y<=Z)

C. (X<=Y<=Z)　　D. (X<=Y)&(Y<=Z)

(16) 如下程序的输出结果为(　　)。

```
#include <stdio.h>
int main()
{ int a,b,c = 246;
  a = c/100 % 9;
  b = ( - 1)&&( - 1);
  printf(" % d, % d\n",a,b);
  return 0;
}
```

A. 2,1　　B. 3,2　　C. 4,3　　D. 2,-1

(17) 以下程序的输出结果是(　　)。

```
#include <stdio.h>
int main()
{ int a = - 1,b = 1,k;
   if( (++a < 0)&&!(b -- <= 0))
       printf(" % d % d\n",a,b);
    else
       printf(" % d % d\n",b,a);
   return 0;
}
```

A. -1　1　　B. 0　1　　C. 1　0　　D. 0　0

(18) 下列关于 switch 语句和 break 语句的结论中,正确的是(　　)。

A. break 语句是 switch 语句中的一部分

B. 在 switch 语句中必须使用 break 语句

C. 在 switch 语句中可根据需要使用或不使用 break 语句

D. break 语句只能用于 switch 语句中

(19) 为避免在嵌套的条件语句 if-else 中产生二义性,C 语言规定:else 子句总是与(　　)相配对。

A. 缩排位相同的 if　　B. 其之前最近的 if

C. 其之后最近的 if　　D. 同一行上的 if

(20) 有说明语句“int a=1,b=0;”,则执行下列语句后,输出为(　　)。

```
switch(a)
{   case 1:
        switch(b)
        {
         case 0: printf("**0**");break;
         case 1: printf("**1**");break;
         }
   case 2: printf("**2**"); break; }
```

A. **0**　　B. **0****2**

C. **0****1****2**　　D. 有语法错误

2. 阅读程序题,写出以下程序的执行结果。

(1)

```
#include "stdio.h"
int main()
{
   int  score;
   score=6;
   switch(score)
   {      case    1:    printf("Monday!");
          case    2:    printf("Tuesday!");
          case    3:    printf("Wednesday!");
          case    4:    printf("Thursday!");
          case    5:    printf("Friday!");
          case    6:    printf("Saturday!");
          case    7:    printf("Sunday!");
          default   :    printf("data error!");
   }
   return 0;
}
```

(2)

```
#include"stdio.h"
int main()
{
   int   m=5;
   if(m++>5)  printf("%d\n",m);
          else  printf("%d\n",m--);
   return 0;
}
```

(3)

```
#include"stdio.h"
int main()
{   int   x=1,a=0,b=0;
    switch(x){
      case 0:   b++;
      case 1:   a++
      case 2:   a++;b++
     }
     printf("a=%d,b=%d\n",a,b);
     return 0;
}
```

(4) 输入值分别为 90、80、70、60、50。

```
#include <stdio.h>
int main()
{
   int score,s;
   char grade;
   printf("please input score(0 - 100)\n");
   scanf("%d",&score);
   s = score/10;
   switch(s)
   {
     case 10:
     case 9:grade = 'A';
     case 8:grade = 'B';break;
     case 7:grade = 'C';break;
     case 6:grade = 'C';
     default:grade = 'D';break;
   }
   printf("%c",grade);
   return 0;
  }
```

(5) 输入值分别为'm','n','k'。

```
#include <stdio.h>
int  main()
{  char ch;
   printf("Enter  ch:  ");
   scanf("%c",&ch);
   switch(ch)
   {
      case 'm' : printf("Good morning !\n");       break;
      case 'n' : printf("Good night!\n ");         break;
      default : printf("I can not understand!\n");break;
   }
   printf("All right!\n");
   return  0;
}
```

(6)

```
#include <stdio.h>
int  main()
{
    int a,b;
    a = b = 5;
    if(a == 1)
     if(b == 5)
       {a += b;
        printf("a = %d\n ",a);
       }
     else
       {a -= b;
        printf("a = %d\n",a);
       }
  printf("a + b = %d",a + b);
  return 0;
}
```

3. 程序设计题

(1) **求最小值**。输入两个整型数据,输出最小数。

(2) **数的排序**。输入3个整数x、y、z,把这3个数由小到大输出。

(3) **天数统计**。输入某年某月某日,判断这一天是这一年的第几天。

(4) **奖金计算**。企业发放的奖金根据利润提成。利润(i)低于或等于10万元时,奖金可提10%;利润高于10万元、低于20万元时,低于10万元的部分按10%提成,高于10万元的部分,可提成7.5%;利润在20万到40万元之间时,高于20万元的部分,可提成5%;利润在40万到60万元之间时,高于40万元的部分,可提成3%;利润在60万到100万元之间时,高于60万元的部分,可提成1.5%;利润高于100万元时,超过100万元的部分按1%提成。从键盘输入当月利润i,求应发放奖金总数。

(5) **函数值的计算**。分段函数值如下,输入x的值,输出y的值。

$$y=\begin{cases}0 & (x<0)\\ x & (0\leqslant x<5)\\ 5 & (5\leqslant x\leqslant 10)\\ 2x-5 & (x>10)\end{cases}$$

(6) **计算器**。输入两个整型数据和一个运算符,根据输入的运算符求这两个数的运算结果。如输入“+”,则求这两个数的和。

(7) **解方程**。编写程序,解一元一次方程ax+b=0。

(8) **闰年判断**。编写程序,判断2000年、2008年、2100年是否为闰年。

(9) **整数位数统计与拆分**。有一个不多于5位的正整数,求它的位数和每位数字。

(10) **数字转换成英文名称**。编写程序,将输入的数字(0~6)转换成对应的星期英文名称输出。