第5章

选 择 结 构

5.1 概　　述

在问题求解中语句的执行需要根据条件进行判断，并选择符合逻辑的语句或者语句块来执行，具体表现形式为分支选择结构。分支选择结构的主要作用是在顺序结构的基础上，根据条件判断结果改变代码执行顺序，使得程序能够根据不同的情况做不同的响应。

常见的选择结构有单分支选择结构、双分支选择结构、多分支选择结构以及嵌套分支选择结构，如图 5.1 所示。

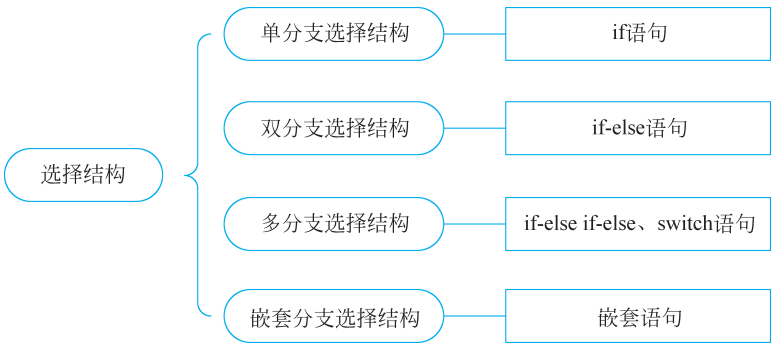

图 5.1　选择结构

5.2 单分支选择结构

单分支选择结构语法：

```
if(表达式){
    语句;
}
```

如果表达式为真(非 0)，则执行语句，否则不执行。

说明：

(1) 表达式可以是任何类型，常用的是关系或逻辑表达式。

(2) 语句可以是一条语句或者语句块，也可以是另一个 if 语句。

执行流程如图 5.2 所示。

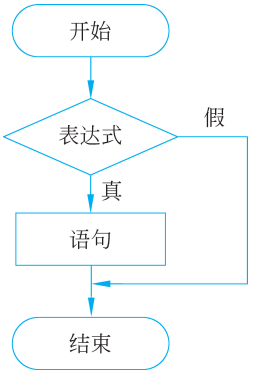

图 5.2 单分支选择结构

例 5.1 输入一个整数,如果是偶数,则输出 even。

判断一个整数 n 是否为偶数的方法是测试该整数是否为 2 的倍数,即 n%2 的结果是否为 0。根据这个结果进行相应输出,对不同的 n 进行不同的响应。

代码如下:

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

int main(){
    int num;
    scanf("%d",&num);
    if(num %2 == 0){
        printf("even");
    }
    return 0;
}
```

当 if 中表达式成立时,说明 num 是偶数,此时输出结果。本例中仅执行一条输出语句,可以不用{},但为提高代码的可读性,培养良好的编程规范,建议加上{}。

5.3 双分支选择结构

当判断为假的分支也需要处理相应的逻辑时,可以使用双分支选择结构,相应的语句为 if-else 语句,语法如下。

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

如果表达式为真(非 0),则执行语句 1,否则执行语句 2。

说明:

(1) 表达式可以是任何类型的,常用的是关系或逻辑表达式。

(2) 语句 1 和语句 2 可以是任何可执行语句,也可以是另一个 if-else 语句。

执行流程如图 5.3 所示。

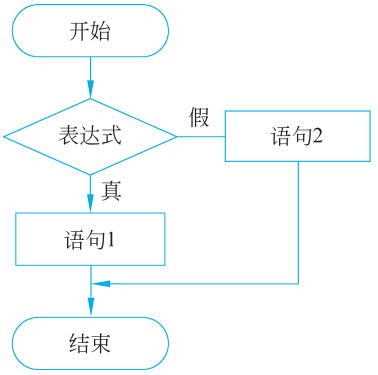

图 5.3　双分支选择结构

例 5.2　输入一个字符,如果是数字字符则输出"digital",否则,输出"non digital"。

对于一个字符 ch,测试其 ASCII 是否为'0'～'9',如果是则为数字字符,输出"digital",否则输出"non digital"。因为本例对于是或不是数字字符这两种情况都需要有响应,所以需要使用 if-else 结构。

代码如下:

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

int main(){
    char char;
    scanf("%c",&ch);
    if(c >= '0' && c <= '9'){
        printf("digital");
    }else{
        printf("non digital");
    }
    return 0;
}
```

例 5.3　输入年份,判断该年份是否为闰年。

判断年份 year 是否为闰年的表达式为:

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

因为表达式中逻辑与 && 的优先级高于逻辑或||的优先级,所以先计算 year % 4==0 && year % 100 !=0 子式,如果该子式成立,则不计算第二个子式 year % 400==0,否则计算第二个子式。

为了提高程序的可读性,也可以通过()来明确子式的计算顺序:

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

完整代码如下:

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

```
int main(){
    int year;
    scanf("%d",&year);
    if((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)){
        printf("%d is leap year.");
    }else{
        printf("%d is not leap year");
    }
    return 0;
}
```

例 5.4 再论鸡兔同笼。

鸡兔同笼需要求解方程组:

$$\begin{cases} x = \dfrac{4n - m}{2} \\ y = \dfrac{m - 2n}{2} \end{cases}$$

其中,n 为头的数量,m 为腿的数量。考虑到问题的特殊性,求解结果 x、y 必须是整数解,因此,首先要判断 $4n-m$、$m-2n$ 是否能够被 2 整除,如果能够整除,则存在合法的解,否则,问题不存在解。

改进后的代码如下:

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

int main(){
    int n,m,x,y;
    scanf("%d %d",&n, &m);
    if((4 * n - m) % 2 == 0 && (m - 2 * n) % 2 == 0){
        x = (4 * n - m) / 2;
        y = (m - 2 * n) / 2;
        printf("chickens:%d rabbits:%d\n",x,y);
    }else{
        printf("No solution.");
    }
    return 0;
}
```

测试 1:

```
35 94
chickens:23 rabbits:12
```

测试 2:

```
35 93
No solution.
```

5.4　多分支选择结构

5.4.1　else if 多分支选择结构

多分支选择结构通过 else if 多分支选择语句实现。

语法：

```
if(表达式 1){
   语句 1;
}else if(表达式 2){
   语句 2;
}else if(表达式 3){
   …
}else if(表达式 n - 1){
   语句 n - 1;
}else{
   语句 n;
}
```

else if 多分支选择结构

这种有规则的多层嵌套，又称为 if 语句的 else if 结构。最后一个 else 及其下面的语句 n 也可以不存在。

注意：

（1）选择结构是从上到下匹配的，一旦匹配上某个条件后，整个条件语句就结束了，即使后面也能匹配上条件也不会再执行了。

（2）使用 if-else if 后可以不写 else。

执行流程如图 5.4 所示。

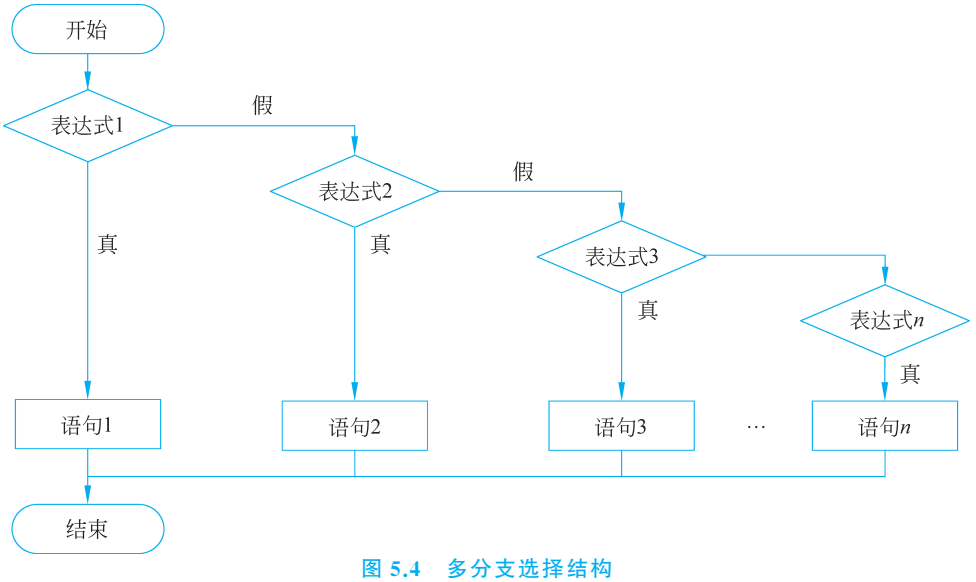

图 5.4　多分支选择结构

例 5.5 输入学生成绩,输出其等级。

代码如下:

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

int main(){
    int grade;
    scanf("%d",&grade);
    if(grade >= 90){
        printf("A");
    }else if(grade >= 80){
        printf("B");
    }else if(grade >= 70){
        printf("C");
    }else if(grade >= 60){
        printf("D");
    }else{
        printf("E");
    }
    return 0;
}
```

如果第一个判断成绩成立,则不执行后续的判断,否则,执行第二个判断。如果第二个判断不成立,则执行第三个判断,直到执行最后的 else。

如果成绩为 95,则第一个判断成立,输出 A,后续的判断都不执行;如果输入的成绩为 65,则直到 else if(grade ＞＝60)时判断才成立,输出 D。


switch 多分支选择结构

5.4.2 switch 多分支选择结构

当分支较多时,使用 else if 语句的形式比较复杂,可以使用 switch 语句来实现,具体语法为:

```
switch(整型表达式){
    case 常量表达式 1:
         语句 1;
         [break;]
    case 常量表达式 2:
         语句 2;
         [break;]
         …
  case 常量表达式 n-1:
         语句 n-1
         [break;]
      default:
         语句 n
         [break;]
}
```

在计算整型表达式的值后，将得到的值与每个 case 后的常量表达式进行比较，当表达式的值与某个常量表达式的值相等时，执行后面的语句，直到遇到 break 语句为止。若表达式的值与所有 case 后的常量表达式均不相同时，则执行 default 对应的语句。

例 5.6　输入学生成绩，输出其等级（用 switch 语句实现）。

switch 语句中用于判断的 case 语句判断的是一个常量，而成绩等级是一个范围，因此，在使用 switch 时要将一个成绩范围表示为一个常量。

对于一个给定的 grade，执行 grade＝grade/10，将一个 10 分的区间压缩为一个常量。如将 60～69 的成绩映射到 6，70～79 的成绩映射到 7，80～89 的成绩映射到 8……

具体代码如下：

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

int main(){
    int grade;
    scanf("%d",&grade);
    grade /= 10;
    switch(grade){
        case 10:
        case 9:  printf("A"); break;
        case 8:  printf("B"); break;
        case 7:  printf("C"); break;
        case 6:  printf("D"); break;
        case 5:
        case 4:
        case 3:
        case 2:
        case 1:
        case 0:  printf("E"); break;
    }
    return 0;
}
```

注意：

（1）switch 语句与 if 语句不同，switch 仅能判断表达式的值是否等于指定的常量，而 if 可以计算并判断各种表达式。

（2）switch 语句后必须为整型表达式。

（3）switch 中可以有任意多的 case 语句，case 后必须为常量。

（4）default 可以省略。

（5）case 和 default 顺序可以颠倒。

例 5.7　输入一个日期（含年、月、日），计算该日期是该年度中的第几天。

根据月份计算天数，其中 1、3、5、7、8、10、12 月的天数为 31，4、6、9、11 月的天数为 30 天，2 月份的天数需要根据年份判断，如果为闰年则为 29 天，否则为 28 天。

将指定月份之前所有月份的天数相加，然后加上本月的天数，如果月份大于 2 且为闰年，天数加 1。

完整代码如下：

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

int main(){
    int year, month, day;
    scanf("%d-%d-%d",&year,&month,&day);
    //判断闰年
    int isLeapYear = (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
    int days = 0;
    switch(month){
        case 1: days = day; break;
        case 2: days = 31 + day; break;
        case 3: days = 31 + 28 + day; break;
        case 4: days = 31 + 28 + 31 + day; break;
        case 5: days = 31 + 28 + 31 + 30 + day; break;
        case 6: days = 31 + 28 + 31 + 30 + 31 + day; break;
        case 7: days = 31 + 28 + 31 + 30 + 31 + 30 + day; break;
        case 8: days = 31 + 28 + 31 + 30 + 31 + 30 + 31 + day; break;
        case 9: days = 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + day; break;
        case 10: days = 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + day; break;
        case 11: days = 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + day; break;
        case 12: days = 31 + 28 + 31 + 30 + 31 + 30 + 31 + 31 + 30 + 31 + 30 + day;
break;
    }
    if(month > 2 && isLeapYear){
        days ++;
    }
    printf("%d-%d-%d is the %dth day of year",year,month,day,days);
    return 0;
}
```

运行结果：

```
2020-12-31
2020-12-31 is the 366th day of year
```

5.5 嵌套分支选择结构

选择结构可以嵌套使用，即在判断正确的分支中执行的是一条选择语句。

例 5.8 输入 3 个整数，输出最大的整数。

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

int main(){
    int x, y, z, max;
```

```
    scanf("%d,%d,%d",&x, &y, &z);
    if(x > y){
        if(x > z) {
            max = x;
        }else{
            max = z;
        }
    }else{
        if(y > z){
            max = y;
        }else{
            max = z;
        }
    }
    printf("the max is : %d\n",max);
    return 0;
}
```

注：C 语言规定了 if 和 else 的就近匹配原则，即 else 和最近的没有配对的 if 配对，与书写格式无关。

在执行语句后面都加上{}，匹配的 if else 保持相同的缩进，可以使逻辑更加清晰，且不容易出错。

查看如下代码：

```
int main(){
    if(…)
        if(…) printf("…");
    else printf("…");
    return 0;
}
```

代码虽然主观上将 else 与第一个 if 配对，但按照就近原则，else 与第二个 if 配对，由此造成了程序逻辑错误。因此，在编写代码时，尽量用{}界定范围，防止出现语法错误。

例 5.6 中虽然可以将百分制输出为等级制，但在边界上存在问题，即当输入－1～－9、101～109 时，程序仍然可以输出等级，因此，对于边界的成绩需要进行判断，只对合法的输入进行等级输出，其他的成绩输出为“输入错误”，代码修改如下：

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

int main(){
    int grade;
    scanf("%d",&grade);
    if(grade >= 0 && grade <= 100){
       grade /= 10;
       switch(grade){
          case 10:
          case 9:  printf("A"); break;
```

```
            case 8:  printf("B"); break;
            case 7:  printf("C"); break;
            case 6:  printf("D"); break;
            case 5:
            case 4:
            case 3:
            case 2:
            case 1:
            case 0:  printf("E"); break;
        }
    }else{
        printf("输入错误");
    }
    return 0;
}
```

5.6 条件表达式

条件运算符是 C 语言中唯一的三元运算符,需要 3 个运算对象,每个运算对象都是一个表达式,如下所示:

```
表达式 1 ? 表达式 2: 表达式 3
```

计算方法:如果表达式 1 为真,整个条件表达式的值是表达式 2 的值,否则,是表达式 3 的值。因此,三目运算符实际上就是一个 if-else 结构。

例 5.9 求 3 个整数中的最大值。

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

int main(){
    int x, y, z;
    scanf("%d,%d,%d",&x,&y,&z);
    int max = x > y ? ( x > z ? x : z) : (y > z ? y : z);
    printf("the max element is %d\n", max);
    return 0;
}
```

使用嵌套的三目运算符虽然可以用少量的代码实现比较复杂的功能,但这将导致程序的可读性大大降低,因此,不建议将三目运算符进行嵌套。

5.7 能 力 拓 展

一元二次方程求根

5.7.1 一元二次方程求根

给定一元二次方程 $f(x)=ax^2+bx+c$,其中 a、b、c 为实数,求方程的根(精确到小数点后 6 位)。