第 3 章　程序流程控制

学习目标

(1) 掌握顺序结构的概念。

(2) 掌握分支和多分支语句的语法格式及用法。

(3) 理解循环的概念，掌握 while、do-while、for 3 种循环语句的语法格式、用法及区别。

(4) 掌握选择结构和循环结构嵌套的含义及用法。

3.1　顺序结构程序设计

用 C 编写程序时，实现顺序结构的方法非常简单，只需要将语句顺序排列即可。例如交换两个整数值的程序段：

```
t=x;
x=y;
y=t;
```

就是顺序结构。

3.2　选择结构程序设计

选择结构又称分支结构，它的作用是根据指定的条件，来决定某些操作是执行还是不执行，或者决定从给定的若干操作中选择部分代码执行。C 语言的选择结构通过 if 语句和 switch 语句来实现。

3.2.1　if 语句

if 语句用来判断给定的条件是否满足，根据判定结果的真或假来决定执行两组操作之一。C 语言中 if 语句有 3 种形式：单分支 if 语句、双分支 if 语句、多分支 if 语句。

1. 单分支 if 语句

基本格式为：

```
if(表达式)
    语句
```

执行这一结构时，首先对表达式的值进行判断。如果表达式的值为“真”，则执行其后的语句，否则不执行该语句。其执行流程如图 3.1 所示。

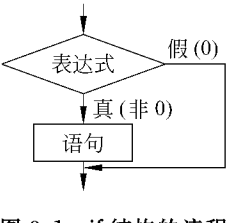

图 3.1　if 结构的流程

【例 3-1】　在两个数中取大数。

程序代码如下：

```
/*e3_1.c*/
```

```
#include <stdio.h>
main()
{
    int num1, num2, max;
    printf("\n input 2 numbers: ");
    scanf("%d%d", &num1, &num2);
    max=num1;
    if(max<num2) max=num2;
    printf("max=%d\n", max);
}
```

程序运行结果：

```
input 2 numbers:
5 3
max=5
```

程序说明：

本例中，输入两个整数 num1，num2。把 num1 先赋予变量 max，再用 if 语句判别 max 和 num2 的大小，如果 max 小于 num2，则把 num2 赋予 max。因此 max 总是两者中的大数，最后输出 max 的值。用 scanf 函数输入数据时，要注意输入格式。

2. 双分支 if 语句：if-else

if-else 结构构造了一种双分支结构。

基本格式为：

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

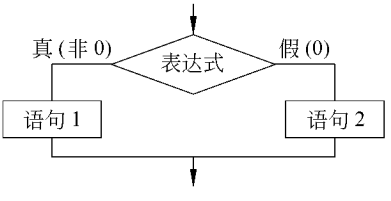

图 3.2 if-else 结构

执行该结构时，首先对表达式的值进行判断。如果表达式的值为“真”，则执行语句 1；否则执行语句 2。

【例 3-2】 在两个数中取大数。

程序代码如下：

```
/*e3_2.c*/
#include <stdio.h>
main()
{
    int num1, num2;
    printf("\n input 2 numbers: ");
    scanf("%d%d", &num1, &num2);
    if(num1>num2)
        printf("max=%d\n", num1);
    else
        printf("max=%d\n", num2);
}
```

程序运行结果：

```
input 2 numbers:
5 3
max=5
```

程序说明：

本例与例 3-1 功能相同，输入两个整数 num1 和 num2，判断两者的大小，若 num1 大，则输出 num1，否则输出 num2。

改用 if-else 语句实现，比 if 结构易于理解且格式清晰。

3. 多分支 if 语句：if-else if

if-else if 是一种多分支选择结构。其格式为：

```
if(表达式 1)
    语句 1
else if(表达式 2)
        语句 2
        …
      else if(表达式 n)
            语句 n
          else 语句 n+1
```

执行这一结构时，依次判断表达式的值，当出现某个表达式值为“真”时，则执行其对应的语句。然后跳到整个 if 语句之外继续执行。如果所有的表达式均为假，则执行语句 n＋1，然后继续执行后续程序。if-else if 结构的执行过程如图 3.3 所示。

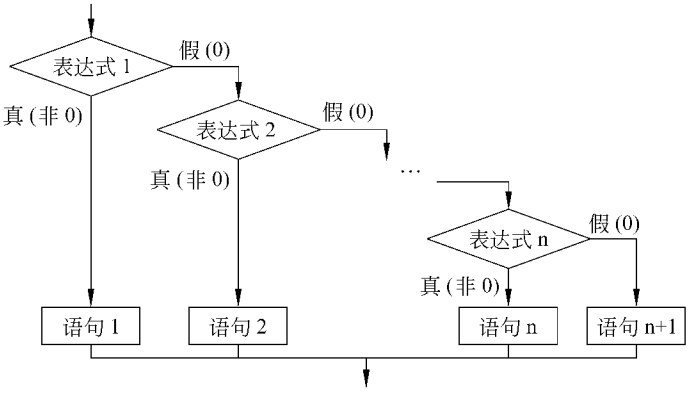

图 3.3　if-else if 结构的执行过程

【例 3-3】　将学生成绩由百分制转化为等级制。规则如下：

(1) 85 分(含)以上为 A 级。

(2) 70 分(含)以上且 85 分以下为 B 级。

(3) 60 分(含)以上且 70 分以下为 C 级。

(4) 60 分以下为 D 级。

程序代码如下：

```
/*e3_3.c*/
#include <stdio.h>
main()
```

```
{
    float score;
    printf("\n please input a score: ");
    scanf("%f", &score);
    if(score>=85)
        printf("the score %f is A \n", score);
    else if(score>=75)
                printf(" the score %f is B \n", score);
          else if(score>=60)
                      printf("the score %f is C \n", score);
                  else
                    printf("the score %f is D \n", score);
}
```

程序运行结果：

```
please input a score: 89
the score 89.000000 is A
```

程序说明：

这是一个多分支选择的问题，用 if-else if 语句实现，通过判断输入的成绩变量 score 所在的范围，分别给出不同的输出。

if-else if 结构列出了一系列的条件（表达式）和对应的操作，通过条件（表达式）成立与否，而执行相应的操作。执行该结构时，依次对各个条件（表达式）进行判断，一旦某一表达式值为“真”，就转去执行对应的操作，其他部分便不再执行；若所有表达式都为“假”，则执行最后一个 else 所对应的操作；若最后一个 else 不存在，同时所有表达式都为“假”，则此 if-else if 结构不作任何操作。

【例 3-4】 输入一个字符，如果是数字则输出 A；如果是大写字母则输出 B；如果是小写字母则输出 C；如果是空格则输出 D；如果是回车换行则输出 E，是其他符号时则输出 F。

程序代码如下：

```
/*e3_4.c*/
#include <stdio.h>
main()
{
    char c;
    scanf("%c", &c);
    if('0'<=c&&c<='9')
        printf("c=%c--->%c", c, 'A');
    else if('A'<=c&&c<='z')
             printf("c=%c--->%c", c, 'B');
    else if('a'<=c&&c<='z')
             printf("c=%c--->%c", c, 'C');
    else if(c==' ')
             printf("c=%c--->%c", c, 'D');
    else if(c=='\n')
             printf("c=%c--->%c", c, 'E');
    else
        printf("c=%c--->%c", c, 'F');
}
```

程序运行结果：

```
3
c=3--->A
M
c=M--->B
&
c=&--->F
```

程序说明：

程序中 4 个 if-else 语句结构中 else 后面的语句又是一个 if-else 语句，这种结构在 C 语言中称为多层嵌套的 if-else 语句。嵌套 if 和 else 总是成对出现的。当程序中存在 if-else 嵌套时，从后向前，else 总是与其前离它最近的一个 if 配对。若 else 之前又有一个未配对的 else，则先将其前(内层的)else 配对。依次下去，直至全部 else 用完。

请思考下面程序的运行结果是什么。

```
main()
{
    int x=100, y=10, a=30, b=20, k1=10, k2=6;
    if(a>b)
        if(b!=10)
            if(!k1)x=1;
            else if(k2) x=10;
                else x=20;
        printf("x=%d\n", x);
}
```

使用 if 语句还应注意下列几个问题。

(1) 3 种形式的 if 语句中，在关键字 if 之后均为“表达式”。该表达式通常是逻辑表达式或关系表达式。

例如：

```
if(x==y&&a==b) printf("x=y, a=b\n");
```

执行时 if 语句先对表达式求解，若表达式值为“真”，则执行后面的语句；

表达式也可以是其他表达式，如赋值表达式等，甚至也可以是一个变量。例如：

```
if(x=3) 语句;
if(a) 语句;
```

都是合法的，只要表达式的值为“真”，则执行其后的语句。

(2) 在 if 语句中，条件判断表达式必须用括号括起来，在语句之后必须加分号。空语句也是允许的，他表示什么也不做。如 if(b>0);。

(3) if 语句的 3 种形式中，所有语句均为单个语句，若需要在条件满足时执行多个语句，必须把多个语句用花括号括起来组成一个复合语句。注意在花括号后不要加分号。

例如：

```
if(x>1)
    {x++; y--;}
else
```

```
    {x--; y++;}
```

(4) 正确地使用缩进格式,有助于更好地理解程序,尤其是在使用 if 语句嵌套时。同时,在容易引起混淆的地方添加花括号保证逻辑关系的正确性。例如:

```
main()
{
    int x, y;
    printf("\n input x");
    scanf("%d", &x);
    if(x<0)
        y=-1;
    else
    {
        if(x==0)y=0;
        else y=1;
    }
    printf("x=%d, y=%d\n", x, y);
}
```

3.2.2 switch 语句

switch 是一种多分支选择结构,也称为标号分支结构。其格式为:

```
switch(表达式)
{
    case 常量表达式 1: 语句序列 1
    case 常量表达式 2: 语句序列 2
    ……
    case 常量表达式 n: 语句序列 n
    default: 语句序列 n+1
}
```

switch 结构在执行时,首先计算 switch 判断表达式的值,并按照计算结果依次寻找 case 子结构中与之相等的常量表达式,若找到,则执行该 case 子结构后的语句序列;若找不到与之相等的常量表达式,则执行 default 后的语句序列。default 子句不是必需的。若结构中无 default 子结构且没有相符的 case 子结构时,则什么也不做。执行流程如图 3.4 所示。

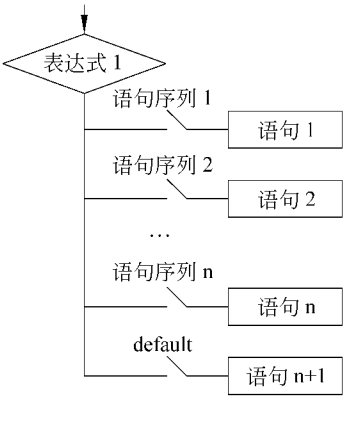

图 3.4 switch 结构的流程

在使用 switch 结构时,应注意以下几个问题。

(1) case 后的各常量表达式值不能相同。

(2) case 后允许有多个语句,可以不用花括号括起来。

(3) 程序执行至与 switch 表达式值匹配的 case 子结构后的语句序列 m 后,不是立即退出 switch 结构,而是继续执行语句序列 m+1,直至语句序列 n+1。若需要在执行语句序列 m 后,立即退出 switch 结构,则在每个语句序列后加一条 break 语句,break 语句用于跳出 switch 结构。

(4) switch 结构也可以嵌套。

【例 3-5】 输入 1～7 中的数字，将其转换成相应的星期英文单词。

程序代码如下：

```
/*e3_5.c*/
#include <stdio.h>
main()
{
    int num;
    scanf("%d", &num);
    switch(num)
    {
        case 1: printf("Monday\n"); break;
        case 2: printf("Tuesday\n"); break;
        case 3: printf("Wednesday\n"); break;
        case 4: printf("Tursday\n"); break;
        case 5: printf("Friday\n"); break;
        case 6: printf("Saturday\n"); break;
        case 7: printf("Sunday\n"); break;
        default: printf("error\n");
    }
}
```

程序运行结果：

```
1
Monday
```

若输入 1～7 之外的数字则显示：error。

程序说明：

在本例中，每一个 case 子结构最后都有一条 break 语句，用于跳出 switch 流程。程序中若无 break 语句，运行时，若输入数字 1，则运行结果为，

```
Monday
Tuesday
Wednesday
Tursday
Friday
Saturday
Sunday
Error
```

使用 switch 结构时，一定要注意 break 语句的位置。

【例 3-6】 编写程序测试输入的是数字、空白、还是其他字符。

程序代码如下：

```
/*e3_6.c*/
#include <stdio.h>
main()
{  char c;
   scanf ("%c",&c);
switch(c)
{
```

```
    case '0':
    case '1':
    case '2':
    case '3':
    case '4':
    case '5':
    case '6':
    case '7':
    case '8':
    case '9': printf("this is a digit\n"); break;
    case ' ':
    case '\n':
    case '\t': printf("this is a blank\n"); break;
    default: printf("this is a character\n"); break;
}
}
```

程序运行结果：

```
3
this is a digit
```

程序说明：

程序执行时，假设输入一个换行符(即 c='\n')，则从 case '0'寻找各 case 子结构后面的常数，从 case '\n'入口开始执行后面的语句，case '\n'后面没有语句，于是执行 case '\t'后面的 printf 函数语句，遇到 break 跳出 switch 流程。

3.3 循环结构程序设计

3.3.1 while 语句

while 用来实现“当型”循环结构。其格式为：

```
while(表达式)
    语句
```

在执行 while 语句时，先对表达式进行计算，若值为“真”(非 0)，则执行循环体语句；否则跳过循环体语句，执行 while 结构后面的语句。每执行完一次循环体语句，都对表达式进行一次计算和判断。若表达式值为 0，则立即跳出循环体。流程如图 3.5 所示。

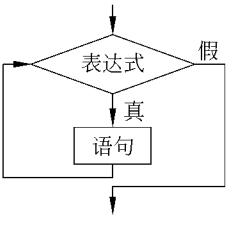

图 3.5 while 结构的流程

【例 3-7】 while 举例。

程序代码如下：

```
/*e3_7.c*/
#include <stdio.h>
main()
{
    int count=1;
    while(count<5)
    {
```

```
        printf("good morning!\n");
        count++;
    }
}
```

程序运行结果：

```
good morning!
good morning!
good morning!
good morning!
```

程序说明：

本例执行时，会输出 4 个令人愉快的“good morning!”，循环条件是 count＜5 ，count 的值控制循环执行的次数，称为循环变量。合理的循环中，其值应不断地修正。退出循环时 count 的值为 5。while 结构还可改为：

```
count=1;
while(count++<5)
    printf("good morning!\n");
```

【例 3-8】 计算 sum＝1＋2＋3＋4＋…＋100。

程序代码如下：

```
/*e3_8.c*/
#include <stdio.h>
main()
{
    int sum=0, i;
    i=1;
    while(i<=100)
    {
        sum=sum+i;
        i++;
    }
    printf("sum=%d\n", sum);
}
```

程序运行结果：

```
sum=5050
```

程序说明：

程序中，循环变量 i 初值为 1，循环体语句是复合语句{sum＝sum＋i;i＋＋;}，每累加一次，i 的值增 1，为下一次循环作准备。当 i＝101 时，跳出循环体，执行 while 循环后面的 printf 函数语句。

【例 3-9】 猴子吃桃问题。

猴子第一天摘下若干个桃子，当即吃了一半，不过瘾，又多吃了一个。第二天又将剩下的吃了一半，又多吃了一个，以后每天都吃剩下的一半多一个，到第 10 天时，只剩下一个桃子。求第一天共摘了多少个桃子。

设第 n 天的桃子数为 p_n，则算法可描述为：

$p_{10}=1$； ①

$p_n=p_{n-1}/2-1$； ②

要计算第一天的桃子数，采用逆推法。即从第10天开始推算，①式为赋初值，②式用C语言描述为：

```
p1=2 * (p2+1);
p2=p1;
```

这里的p1和p2分别代表每个“昨天”和每个“今天”桃子数。不断用“昨天”的桃数替代“今天”的桃数，这种不断地旧值递推得到新值的过程叫做迭代。迭代要有初值、迭代公式、迭代终止次数。

程序代码如下：

```
/ * e3_9.c * /
#include <stdio.h>
main()
{
    int p1, p2=1;
    int n=9;
    while(n>0)
    {
    p1=2 * (p2+1);
    p2=p1;
    n--;
    }
    printf("the total is %d\n", p1);
}
```

程序运行结果：

```
the total is 1534
```

程序说明：

迭代初值为1(p2=1)，即第10天剩下的桃子数；迭代公式为p1=2*(p2+1)，p2=p1。

使用while循环要注意以下几点。

(1) while是一个入口条件循环，是否执行循环体在进入循环之前决定，因此循环体有可能永远不被执行。

(2) 循环体部分可以是带有分号的简单语句，也可以是花括号中的一个复合语句。while语句在语法上是一个单独语句，即使使用的是复合语句。该语句从while开始，到第一个分号结束；若使用了复合语句，则执行至复合语句结束。

如程序片段：

```
int n=0;
while(n++<3 );
    printf("n is %d\n", n);
printf("it's over.\n");
```

其执行结果为：

```
n is 4
```