第 3 章　选择与循环

（视频讲解：1 小时）

裁缝做衣服需要尺子和剪刀，程序员控制程序执行逻辑运算需要选择和循环结构。本章主要讲解 C 语言中选择语句、循环语句的相关概念与知识。

学习目标：

- 选择结构程序设计
- 循环结构程序设计

3.1　选择结构程序设计

3.1.1　关系表达式与逻辑表达式

在介绍选择语句前，我们首先练习一下关系表达式与逻辑表达式。在第 2 章中，我们了解到算术运算符的优先级高于关系运算符、关系运算符的优先级高于逻辑与和逻辑或运算符、相同优先级的运算符从左至右进行结合等，那么表达式 5>3&&8<4-!0 的最终值是多少？其计算过程如图 3.1.1 所示。

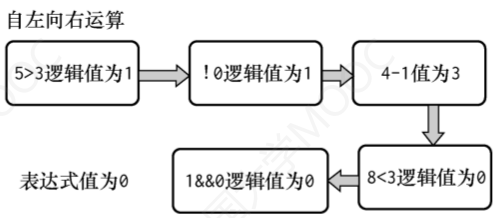


图 3.1.1　5>3&&8<4-!0 的计算过程

下面再来看一个判断年份是否为闰年的例子（闰年是能被 4 整除但不能被 100 整除，或者既能被 4 整除又能被 400 整除的年份）。具体的表达式为 year%4==0&&year%100!=0||year%400==0，也可以写成(year%4==0&&year%100!=0)||year%400==0，后者虽然可行，但括号显然是多余的。

3.1.2　if 语句

在你打开衣柜拿出最上面的一件衣服时，你会判断这件衣服是不是你想穿的。如果是，那么你就会穿上；如果不是，那么你就会去找其他衣服。在计算机中，我们用 if 判断语句来实现这样的效果：if 判断条件（表达式）为真，就执行某个语句，反之不执行这个语句。当然，也可以 if

判断条件（表达式）为真，就执行某个语句，反之用 else 分支执行另一个语句，具体流程如图 3.1.2 和图 3.1.3 所示。

下面来看一个判断输入值是否大于 0 的例子。

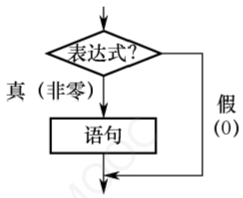


图 3.1.2　if 语句流程 1

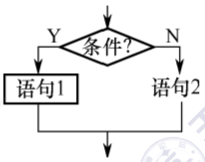


图 3.1.3　if 语句流程 2

当输入值大于 0 时，打印“i is bigger than 0”，当输入值小于等于 0 时，打印“i is not bigger than 0”，具体代码如图 3.1.4 所示。注意，在这个例子中，if 后面不能加分号，因为如果有 else 分支语句，那么加分号会导致编译不通过；如果没有 else 分支语句，那么加分号会导致 i 无论取何值，都会打印“i is bigger than 0”。

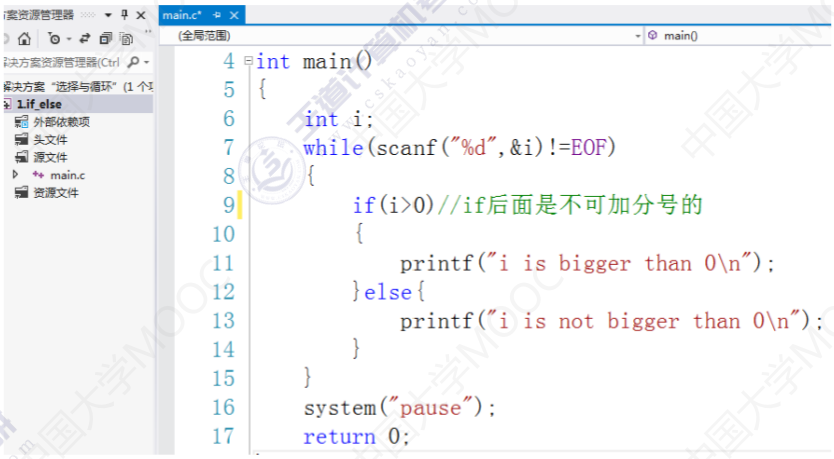


图 3.1.4　判断输入值是否大于 0

if 语句和 else 语句也可以多个同时使用（多分支语句），如图 3.1.5 所示。但是，无论有多少个 if 语句或 else if 语句，程序都只会执行其中的一个语句。下面是一个关于用电量的例子：用电量越高，电的单价越高，但最终 cost 只会被赋值一次。同时，if 语句也支持多层嵌套，在 if 语句中又包含一个或多个 if 语句称为 if 语句的嵌套，如图 3.1.6 所示。

```
if (表达式1) 语句1
else if (表达式2) 语句2
else if (表达式3) 语句3
     …
else if (表达式m) 语句m
else   语句n
```

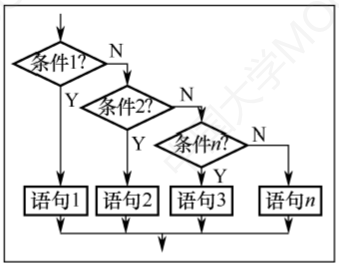


图 3.1.5 多分支语句

```
形式:
if()
     if()   语句1
     else   语句2
else                    内嵌if
     if()   语句3
     else   语句4
```

图 3.1.6 if 语句的嵌套

```
if (number>500)cost=0.15;
     else if(number>300)cost=0.10;
          else if(number>100)cost=0.075;
               else if(number>50)cost=0.05;
                    else   cost=0;
```

使用 if 嵌套语句时，要考虑“悬空的 else”问题。例如，在下面的例子中，else 子句从属于哪个 if 语句？

```
if(i>1)
     if(i<10)
          printf("i>1 and i<10\n");
     else
          printf("no,they are not\n");
```

和其他绝大多数语言一样，C 语言中的 else 子句从属于最靠近它的不完整的 if 语句。上例中的 else 子句从属于第二个 if 语句，如果想让它从属于第一个 if 语句，那么可以用一个花括号把第二个 if 语句包含在一个单独的代码块内，如下所示。

```
if(i>1){
          if(i<10)
            printf("i>1 and i<10\n");
}
     else
          printf("no,they are not\n");
```

在 if 语句中的语句列表前后加上花括号，可以防止不小心加了一句代码后，使实际未被包含的语句被包含在某个 if 语句中的错误。

3.2 循环结构程序设计

3.2.1 while 循环

while 语句用来实现“当型”循环结构，其一般形式为“while(表达式) 语句;”，当表达式的

值非 0 时，执行 while 语句中的内嵌语句。其特点是：先判断表达式，后执行语句，具体流程如图 3.2.1 所示。当表达式的值非 0 时，就会执行语句，从而实现语句多次执行的效果。为了避免程序进入死循环（不停地进行循环操作），在语句中需要有让表达式趋近于假的操作来使程序跳出循环。

下面来看使用 while 语句计算 1 到 100 之间所有整数之和的例子，如图 3.2.2 所示。注意，while 后面不能加分号，否则虽然编译可以通过，但是执行程序时会发生死循环。通常我们会将 while 语句用花括号括起来，就算 while 语句只有一句，也会用花括号括起来，这么做是因为一个程序往往会经过多次修改，使用花括号可以让程序更加清晰，避免向循环内添加语句时出错。

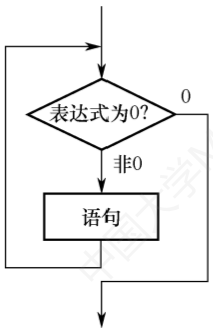


图 3.2.1　while 循环流程

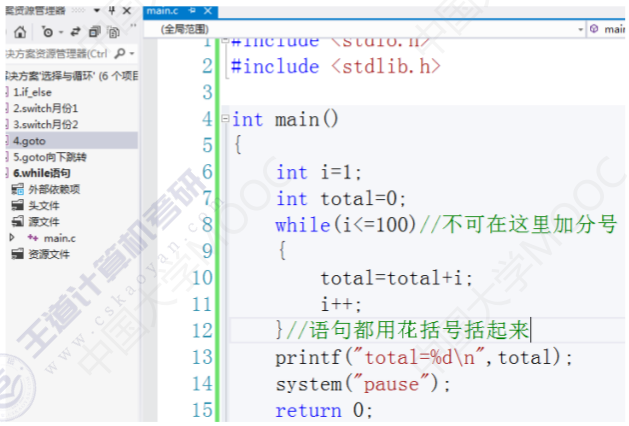


图 3.2.2　while 语句计算 1 到 100 之间的所有整数之和

在 Windows 操作系统下的 VS 集成开发环境中，我们可以用 fflush 或 rewind 清空标准输入缓冲区，但是这些函数在 Linux 操作系统中是无法使用的。那么如何自己写一个清空缓冲区的方法？可以用以下代码解决：

```
while((ch=getchar())!=EOF && ch!='\n');
```

3.2.2　for 循环

C 语言中的 for 循环语句使用最为灵活，不仅可以用于循环次数已经确定的情况，而且可以用于循环次数不确定而只给出循环结束条件的情况，它完全可以代替 while 循环语句。其一般形式为

```
for(表达式 1;表达式 2;表达式 3) 语句;
```

for 循环语句的执行过程如下，具体流程如图 3.2.3 所示。

（1）先求解表达式 1。

（2）求解表达式 2，若其值为真（值为非 0），则先执行 for 语句中指定的内嵌语句，后执行第（3）步。若其值为假（值为 0），则结束循环，转到第（5）步。

(3) 求解表达式3。

(4) 转回第(2)步继续执行。

(5) 循环结束，执行for语句下面的语句。

下面来看一个使用for循环语句计算1到100之间的所有整数之和的例子，for循环语句中必须且只能有两个分号，用于分割表达式1、表达式2和表达式3。表达式1、表达式2、表达式3也可省略，省略写法用得较少。如图3.2.4所示，“i=1,total=0”是表达式1，即表达式1可以使用逗号初始化多个变量，表达式3的作用是使表达式2趋近于假。

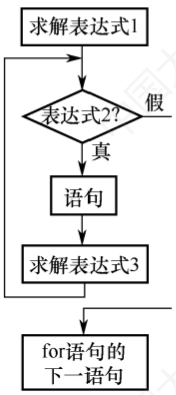
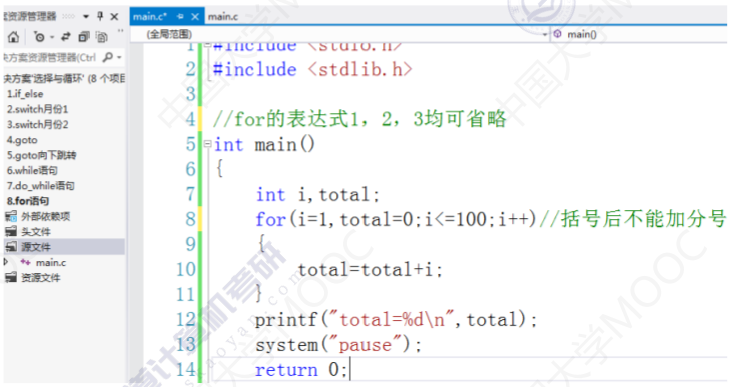


图3.2.3 for循环语句流程



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

//for的表达式1，2，3均可省略
int main()
{
    int i,total;
    for(i=1,total=0;i<=100;i++)//括号后不能加分号
    {
        total=total+i;
    }
    printf("total=%d\n",total);
    system("pause");
    return 0;
```

图3.2.4 for循环语句实现计算1到100之间的所有整数之和

for循环的可读性要比while循环的好，所以能使用for循环时不要强制改为while循环。

3.2.3 continue语句

continue语句的作用为结束本次循环，即跳过循环体中下面尚未执行的语句，接着进行是否执行下一次循环的判断。其一般形式为

```
continue;
```

下面来看一个对1到100之间的奇数进行求和的例子，如图3.2.5所示。这个例子是对图3.2.4中的for循环的改写，执行“continue;”语句后，执行的语句是“i++;”。当continue用于while和do while循环中时，注意不要跳过让循环趋近于假的语句。

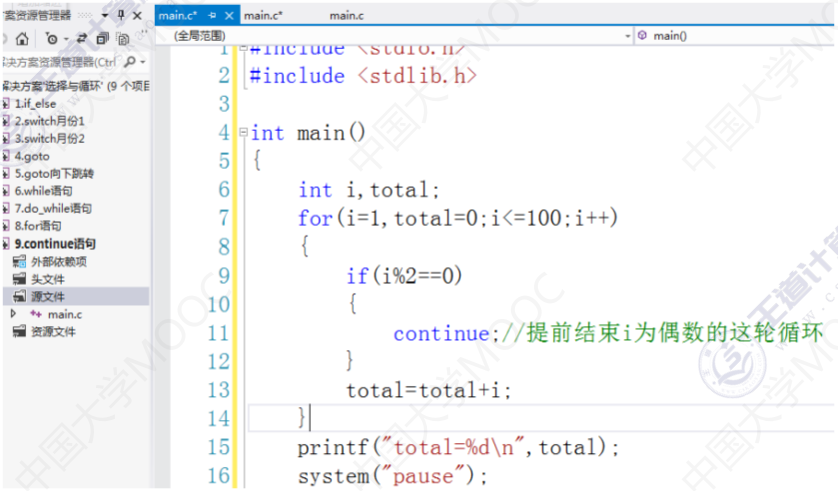


图 3.2.5　对 1 到 100 之间的奇数进行求和

3.2.4　break 语句

break 语句的作用是结束整个循环过程，不再判断执行循环的条件是否成立。图 3.2.6 是关于 break 语句的例子，例子从 1 开始累加，当累加的和大于 2000 时，结束 for 循环，同时打印此时 total 的值和 i 的值。一旦执行 break 语句，下一句要执行的就是“printf("total=%d,i=%d\n",total,i);”。Break 语句也可用在 while 循环和 do while 循环中，起结束对应循环的作用。

图 3.2.6　break 语句实例