第 4 章　流程控制

学习目标：

- 了解选择与循环结构的算法流程图。
- 掌握选择结构的用法。
- 掌握循环结构的用法。
- 了解单分支和多分支选择结构的用法区别。
- 了解 while 和 do-while 循环的区别。
- 了解跳出循环语句 continue 和 break 的区别。

本章主要讲述 C 语言的流程控制，使读者掌握流程控制语句，了解这些语句是如何控制程序的执行顺序的。同时锻炼读者解决问题的思想，训练如何将一个抽象的问题转换成一个实际的算法，并最终将程序编写出来。

流程控制主要包括条件控制和循环控制，条件选择语句包括 if，if else，switch 等语句；循环控制包括 while，do while，for 等语句。学习时要掌握这些语句的用法和区别。最后还必须掌握选择结构和循环结构中的嵌套，因为单独使用选择结构和循环结构无法处理一些比较复杂的问题。

4.1　选择结构程序设计

在程序的三种基本结构中，第二种即为选择结构，其基本特点是：程序的流程由多路分支组成，在程序的一次执行过程中，根据不同的情况，只有一条支路被选中执行，而其他分支上的语句被直接跳过。

选择结构的语句分为单分支选择语句 if，双分支选择语句 if-else，多分支选择语句 switch。这些语句不仅可以单独使用，还可以嵌套使用，形成条件分支的嵌套。

4.1.1　单分支选择语句 if

在 C 语言中，实现单分支选择结构需要用 if 语句，单分支选择结构的流程图如图 4-1 所示。

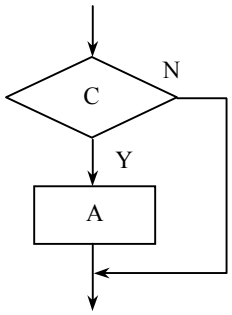

图 4-1　单分支选择结构

其一般形式如下：

if(表达式 C)语句 A;

其中 C 可以是任何表达式，常用的是关系和逻辑表达式。语句 A 可以是简单语句，也可以是复合语句。if 语句的功能是计算表达式 C 的值，如果值是非“0”，则执行语句 A；如果值是 0，则跳过语句 A。

【例 4-1】从键盘上输入三个整数值，显示其中的最大值。

```
#include<stdio.h>
void main()
{
   int a,b,c;
   scanf("%d,%d,%d",&a&b&c);
   if(a<b) a=b;
   if(a<c) a=c;
   printf("%d",a);
}
```

程序运行时，输入值 10，20，30。在运行到“if(a<b) a=b;”时可以发现，10<20，则执行 a=b，a 的值变为 20。继续执行“if(a<c) a=c;”，20<30，则执行 a=30。所以最后显示 a 的值为 30，是三个整数中的最大值。所以程序通过单分支选择结构达到了目的。

4.1.2　双分支选择语句 if-else

在 C 语言中，实现双分支选择结构需要用 if-else 语句，双分支选择结构的流程图如图 4-2 所示。

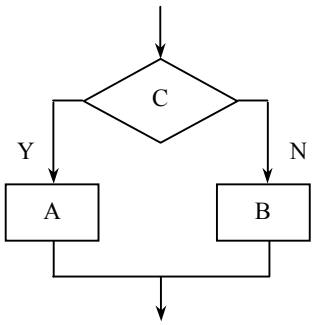

图 4-2　双分支选择结构

其一般形式如下：

if（表达式 C）

语句 A;

else

语句 B;

其中 C 可以是任何表达式，常用的是关系和逻辑表达式。语句 A、B 可以是简单语句，也可以是复合语句。if 语句的功能是计算表达式 C 的值，如果值是非“0”，则执行语句 A；如果值是 0，则执行语句 B。

【例 4-2】从键盘上输入三个整数值，显示其中的最小值。

```
#include<stdio.h>
main()
{
   int a,b,c,min;
   printf("input a,b,c:");
```

```
    scanf("%d %d %d",&a,&b,&c) ;
    if (a<b)
       min=a;
    else
       min=b;
    if (c<min)
       min=c;
    printf("The result is %d\n",min);
}
```

程序运行时，输入值 3，2，1。在运行到“if(a<b) min=a;”时可以发现，3>2，则跳到 else 执行 min=b，min 的值变为 2；继续执行“if(c<min) min=c;”，1<2，则执行 min=1；所以最后显示 min 的值为 1，是三个整数中的最小值。所以程序通过双分支选择和单分支选择结构的共同使用达到了目的。

这里顺便提一下程序书写的缩排问题。所谓缩排，即下一行与上一行相比，行首向右缩进若干字符，如上例的 min=a，min=b 等。适当的缩排能使程序的结构、层次清晰，一目了然，增加程序的易读性。应该养成好的书写习惯，包括必要的注释、适当的空行以及缩排。

4.1.3 多分支选择语句 switch

在 C 语言中，实现多分支结构的语句是 switch 语句。因为 if 语句只能处理从两者间选一，当要实现几种可能之一时，就要用 if…else if 甚至多重的嵌套 if 来实现。当分支较多时，程序变得复杂冗长，可读性降低。C 语言提供了 switch 语句专门处理多路分支的情形，使程序变得简洁。

其一般形式如下：

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

其中“表达式”可以是任何表达式，常用的是整型表达式。“常量表达式”是由常量运算符组成的表达式，常用的是整数或字符常量，所有常量表达式的值不能相同。语句可以是简单语句，也可以是复合语句。switch 语句的功能是计算表达式的值，依次和常量表达式的值作比较，找到与表达式值相同的常量表达式 i，然后从语句 i 开始向下依次执行，语句 i+1…语句 n+1。如果和每一个常量表达式都不相同，则执行语句 n+1。

【例 4-3】在给定的 5 个字母中显示输入字母以及它后面的所有字母。

```
#include<stdio.h>
main()
{
   char ch;
   scanf("%c",&ch);
   switch(ch)
   {
      case'a':printf("%c",'a');
      case'b':printf("%c",'b');
      case'c':printf("%c",'c');
      case'd':printf("%c",'d');
      case'e':printf("%c\n",'e');
      default:printf("those are all the letters.\n");
   }
}
```

本程序首先输入一个字符，例如输入'b'，那么首先检查表达式与哪个常量表达式值相等，发现与 case'b' 的常量表达式值相等。所以从“printf("%c\n",'b')”开始执行，直到“default:printf("those are all the letters.\n");”结束。所以输出结果为：

```
bcde
those are all the letters.
```

switch 语句还有两种变形的格式。

（1）能实现从若干分支中选择一条操作的形式。

```
switch(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：
  语句 n；break；
  default：语句 n+1；
}
```

该语句的功能是计算表达式的值，依次和常量表达式的值作比较，找到与表达式值相同的常量表达式 i，然后执行语句 i 后跳出。如果和每一个常量表达式都不相同，则执行语句 n+1。

【例 4-4】根据学生成绩打印出相应的分数段。

```
#include<stdio.h>
main()
{
   char grade;
   printf("input the grade(A,B,C,D,E):");
   scanf("%c",&grade);
   switch(grade)
```

```
    {
      case'A':printf("90-100\n");break;
      case'B':printf("80-90\n");break;
      case'C':printf("70-80\n");break;
      case'D':printf("60-70\n");break;
      case'E':printf("50-60\n");break;
      default:printf("error\n");
    }
}
```

假如输入 C，程序会寻找与'C'值相等的常量表达式 case'C'，然后执行完语句“printf("70-80\n");”后遇到“break;”跳出。所以达到输入与学生成绩对应分数段的目的。

（2）能实现从若干分支中合并某些分支的形式。

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

该语句的功能是计算表达式的值，依次和常量表达式的值作比较，找到与表达式值相同的常量表达式 i，然后依次执行语句 i，语句 i+1…，直到遇到“break;”后跳出。如果和每一个常量表达式都不相同，则执行语句 n+1。

【例 4-5】输入月份，打印 2005 年该月有几天。

```
#include<stdio.h>
main()
{
   int month;
   int day;
   printf("please input the month number:");
   scanf("%d",&month);
   switch(month)
   {
      case 1:
      case 3:
      case 5:
      case 7:
      case 8:
      case 10:
      case 12: day=31; break;
```

```
        case 4:
        case 6:
        case 9:
        case 11:day=30; break;
        case 2: day=28; break;
        default:day=-1;
    }
    if(day==-1)
        printf("Invalid month input!\n");
    else
        printf("2005.%d has %d days.\n",month,day);
}
```

假如输入月份为 6，那么首先检查表达式值与哪个常量表达式值相等。发现 case 6 与其值相等，所以从 case 6 开始往下依次执行，到“day=30; break;”时先将 day 值变为 30，然后跳出语句。经过双分支选择，执行“printf("2005.%d has %d days.\n",month,day);”，得到输出结果为：

```
2005.6 has 30 days.
```

4.1.4　条件分支的嵌套

C 语言提供的 switch 语句虽然可以实现多分支结构，但是要注意它的局限性很大。因为控制分支的表达式只有一个，无法同时实现多种选择关系；并且由于判断条件是常量表达式，也使很多其他数据类型的量不能使用，从而导致 switch 语句只能解决一部分问题。

因为 switch 语句在多分支选择结构中的局限性，在进行程序设计时，经常要用到条件分支嵌套。所谓条件分支嵌套，就是在一个分支中可以添加另一个分支，从而达到实现多个条件控制选择的目的。

1. 条件分支嵌套的一般结构

在条件分支结构中，当 if 语句中的执行语句又是 if 语句时，则构成了 if 语句嵌套的情形。其一般形式如下：

if(表达式 1)

　　if(表达式 2)语句 1；

　　else(表达式 3)语句 2；

else 语句 3；

这个结构就是条件分支嵌套的一般形式，其中 else 后面还可以继续嵌套选择分支。这样就实现了多个条件的选择。

if 语句嵌套的过程中会出现多个 if 和多个 else 重叠的情况，这时要特别注意 if 和 else 的配对问题。C 语言规定，else 总是与它前面最近的，且没有其他 else 与其配对的 if 配对。为了避免混淆，建议按照 C 语言的书写习惯书写程序，使不同层次的选择分支错开（如上面的条件分支结构的一般形式），这样会使结构一目了然，减少错误的发生。

【例 4-6】比较两数的大小关系。

```
#include<stdio.h>
main()
```

```
{
   int a,b;
   printf("please input A,B:");
   scanf("%d%d",&a,&b);
   if(a!=b)
      if(a>b) printf("A>B\n");
      else printf("A<B\n");
   else printf("A=B\n");
}
```

假如输入 1，20，首先会判断 a 是否与 b 相等。由于不相等，会转到嵌套分支，判断“a>b”，由于 1<20，所以会转到“else”，执行“printf("A<B\n");”，于是输出：

```
A<B
```

假如输入 1，1，首先会判断 a 是否与 b 相等。由于相等，会直接转到第二个“else”，执行“printf("A=B\n");”，于是输出：

```
A=B
```

2. if-else if 结构

if-else if 是条件分支嵌套最常用的一种形式，其形式如下：

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 n)语句 n;
else 语句 n+1;

该结构实际上还是 if 分支的多层嵌套。但因为如果嵌套层数过多，会使程序写得很靠右。因此用 if-else if 结构来代替这种多重嵌套。该结构有点像 switch 语句的结构，也可以进行多分支的选择控制。但由于它的判断条件是多个表达式，因此它比 switch 结构的适用范围要广。

该结构的功能是从表达式 1 开始依次往下检查表达式的值，直至满足表达式值为 1 的表达式 i 处，执行语句 i。如果 n 个表达式值都为 0，则执行语句 n+1。下面仍然以比较两个数的大小为例来介绍 if-else if 结构。

【例 4-7】比较两数的大小关系。

```
#include<stdio.h>
main()
{
   int a,b;
   printf("please input A,B:");
   scanf("%d%d",&a,&b);
   if(a==b) printf("A=B\n");
   else if(a>b) printf("A>B\n");
   else printf("A<B\n");
}
```

假如输入 10，20，首先判断是否满足 a==b，然后检查是否满足 20>10，由于 10<20，于是最后转到“printf("A<B\n");”。所以输出结果为：

```
A<B
```

4.1.5　选择结构设计实例

【例 4-8】输入三个整数 x，y，z，请把这三个数由小到大输出。

算法设计：想办法把最小的数放到 x 上，先将 x 与 y 进行比较，如果 x>y 则将 x 与 y 的值进行交换，然后再用 x 与 z 进行比较，如果 x>z 则将 x 与 z 的值进行交换，这样能使 x 最小。然后再用 y 与 z 作比较，如果 y>z 则将 y 与 z 值进行交换，这样就使 z 最大。这样就能满足从小到大排列的要求。

源程序：

```
#include<stdio.h>
main()
{
   int x,y,z,t;
   printf("please input three numbers:");
   scanf("%d%d%d",&x,&y,&z);
   if(x>y)
      {t=x;x=y;y=t;}
   if(x>z)
      {t=z;z=x;x=t;}
   if(y>z)
      {t=y;y=z;z=t;}
   printf("small to big: %d %d %d\n",x,y,z);
}
```

【例 4-9】输入星期几的数字，显示相应的英语单词。

算法设计：由于这道题涉及的判断量是整型量，可以利用 switch 进行多分支选择，执行完输出对应数字的英语单词就跳出选择结构。

源程序：

```
#include<stdio.h>
main()
{
   int a;
   printf("input integer number:");
   scanf("%d",&a);
   switch (a)
   {
      case 1:printf("Monday\n"); break;
      case 2:printf("Tuesday\n"); break;
      case 3:printf("Wednesday\n"); break;
      case 4:printf("Thursday\n"); break;
      case 5:printf("Friday\n"); break;
      case 6:printf("Saturday\n"); break;
      case 7:printf("Sunday\n"); break;
      default:printf("error\n");
   }
}
```

【例 4-10】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%；20 万到 40 万之间时，高于 20 万元的部分，可提成 5%；40 万到 60 万之间时高于 40 万元的部分，可提成 3%；60 万到 100 万之间时，高于 60 万元的部分，可提成 1.5%，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（精确到整数个位）。

算法设计：首先按照奖金的各个区间设计好选择条件，然后利用选择的嵌套结构。特别注意要把变量设成长整型，否则会发生数据溢出。

源程序：

```
#include<stdio.h>
main()
{
   long int i;
   int bonus1,bonus2,bonus4,bonus6,bonus10,bonus;
   printf("please input the profit:");
   scanf("%ld",&i);
   bonus1=100000*0.1;bonus2=bonus1+100000*0.75;
   bonus4=bonus2+200000*0.5;
   bonus6=bonus4+200000*0.3;
   bonus10=bonus6+400000*0.15;
   if(i<=100000)  bonus=i*0.1;
   else if(i<=200000)  bonus=bonus1+(i-100000)*0.075;
   else if(i<=400000)  bonus=bonus2+(i-200000)*0.05;
   else if(i<=600000)  bonus=bonus4+(i-400000)*0.03;
   else if(i<=1000000)  bonus=bonus6+(i-600000)*0.015;
   else  bonus=bonus10+(i-1000000)*0.01;
   printf("bonus=%d",bonus);
}
```

4.2 循环结构程序设计

循环控制结构（又称重复结构）是程序中的另一个基本结构。在实际问题中，常常需要进行大量的重复处理，循环结构可以使我们只写很少的语句，而让计算机反复执行，从而完成大量类同的计算。

循环结构的语句包括：当型循环语句 while，直到型循环语句 do-while，次数循环型语句 for、break 和 continue 以及 goto 语句。与条件结构一样，循环结构也有嵌套使用，以解决一些较复杂的问题。

4.2.1 当型循环语句 while

实现当型循环需要用 while 语句，当型循环结构的流程图如图 4-3 所示。

其一般形式是：

while(C)语句 A;

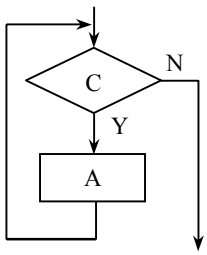

图 4-3　当型循环结构

其中 C 可以是任何表达式，常用的是关系和逻辑表达式。语句 A 可以是简单语句，也可以是复合语句，通常是复合语句。while 语句的功能是计算表达式 C 的值，如果值非“0”，则执行语句 A；反复执行语句 A，直到 C 的值为 0 为止，跳出循环。

【例 4-11】用 while 语句编程，计算 1+2+…+100 的值。

```
#include<stdio.h>
main()
{
   int i=1,sum=0;
   while(i<=100)
   {
      sum=sum+i;
      i++;
   }
   printf("the result is %d",sum);
}
```

在 while 的循环体中，i 从 1 开始，先判断 i 值是否小于等于 100，然后执行循环体。循环体中，先将 i 值加到 sum 上，然后自增一次，再返回判断表达式进行判断。若表达式值为“真”，则继续进行循环，直到 i 增到 101 时，不满足条件，跳出。这样就用简短的循环语句实现了累加的功能。

4.2.2　直到型循环语句 do-while

实现直到型循环需要用 do-while 语句，直到型循环结构的流程图如图 4-4 所示。

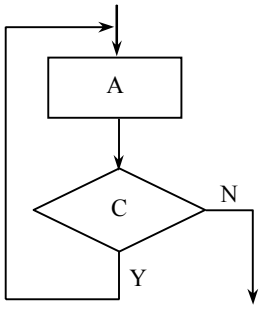

图 4-4　直到型循环结构

其一般形式是：

do 语句 A;

while(表达式 C)

其中 C 可以是任何表达式，常用的是关系和逻辑表达式。语句 A 可以是简单语句，也可以是复合语句，通常是复合语句。do-while 语句与 while 语句的区别是先执行语句 A，再计算表达式 C 的值。如果 C 值是非“0”，则继续执行语句 A；反复执行语句 A，直到 C 的值为 0 为止，跳出循环。

这里为了与 while 语句做个对比，仍然用 1 到 100 的累加为例来说明 do-while 语句的用法。

【例 4-12】用 do-while 语句编程，计算 1+2+…+100 的值。

```
#include<stdio.h>
main()
{
   int i=1,sum=0;
   do{
      sum=sum+i;
      i++;
   }while(i<=100)
   printf("the result is %d",sum);
}
```

在 do-while 的循环体中，i 从 1 开始，先执行循环，执行循环时先将 i 值加到 sum 上，然后自增一次，再计算表达式值判断是否继续进行循环，发现表达式值为“真”，继续执行循环体。直到 i 增到 101 时，不满足条件，跳出。由该例可以看出，do-while 语句与 while 语句的最大区别就是进行循环条件判断的位置。

4.2.3 次数循环型语句 for

在 C 语言中有一种循环结构能够限定循环的次数，实现这种循环的语句是 for 语句。它的一般形式如下：

for(表达式 1;表达式 2;表达式 3)

语句;

其中“表达式 1”、“表达式 2”和“表达式 3”原则上说可以是任意表达式。“表达式 2”常用关系表达式或逻辑表达式，用来控制循环。一般而言，“表达式 1”常常是一个在初次进入循环之前给某些变量赋初值的表达式；而“表达式 3”通常用来修改这些变量的值；“表达式 2”则用这些变量来指定循环条件。

在执行 for 语句时，首先求解表达式 1，然后求解表达式 2，若值为真，则执行 for 后面的“语句”，然后求解表达式 3；否则结束循环。求解表达式 3 后，转回表达式 2 继续重新执行。

次数型循环结构的流程图如图 4-5 所示。

下面是用 for 语句求 1+2+…+100 值的实例，请读者通过这个例子比较 for 语句与前面介绍的 while 语句和 do-while 语句的区别。

【例 4-13】用 for 语句编程，计算 1+2+…+100 的值。

```
#include<stdio.h>
main()
{
```

```
    int i,sum=0;
    for(i=1;i<=100;i++)
       sum=sum+i;
    printf("the result is %d",sum);
}
```

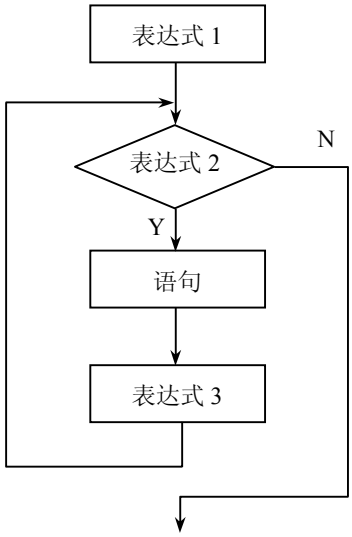

图 4-5　次数型循环结构

在 for 语句中，首先令 i=1，然后判断 i 的值是否超过 100，没有则执行“sum=sum+i;”，然后 i 自增；i 超过 100 则跳出循环。这样便得到了从 1 到 100 的累加和。

4.2.4　break 和 continue 语句

有时，我们需要在循环体中提前跳出循环，或者在满足某种条件下，不执行循环中剩下的语句而立即从头开始新的一轮循环，这时就要用到 break 和 continue 语句。

1．break 语句

在前面学习 switch 语句时，我们已经接触到 break 语句，在 case 子句执行完后，通过 break 语句控制立即跳出 switch 结构。在循环语句中，break 语句的作用是，在循环体中控制立即跳出循环结构，转而执行循环语句后的语句。

【例 4-14】检查输入的一行中有无相邻两字符相同。

```
#include<stdio.h>
main()
{
   char a,b;
   printf("please input the string:\n");
   b=getchar();
   while((a=getchar())!='\r')
   {
      if(a==b)
      {
```

```
            printf("same character\n");
            break;
        }
    b=a;}
}
```

本例程序中，输入一个字符串，把第一个读入的字符送入 b。然后进入循环，读入下一字符 a，比较 a，b 是否相等，若相等则输出提示串并中止循环；若不相等则把 a 中的字符赋予 b，输入下一次循环。要注意 getchar()的用法，在执行 getchar 函数时，虽然是读入一个字符，但不是从键盘按一个字符，该字符就被送给字符变量，而是等到输入完一行按回车键后，才将该行的字符输入缓冲区，然后 getchar 函数从缓冲区中一个一个地取字符给字符变量。

2. continue 语句

continue 语句只能用于循环结构中，一旦执行了 continue 语句，程序就跳过循环体中位于该语句后的所有语句，提前结束本次循环周期并开始新一轮循环。

【例 4-15】输出 100 以内能被 7 整除的数。

```
#include<stdio.h>
main()
{
   int n;
   for(n=7;n<=100;n++)
   {
      if (n%7!=0)
      continue;
      printf("%d",n);
   }
}
```

本例中，对 7～100 的每一个数进行测试，如该数不能被 7 整除，即求余运算不为 0，则由 continue 语句转去下一次循环。只有求余运算为 0 时，才能执行后面的 printf 语句，输出能被 7 整除的数。

3. break 语句和 continue 语句的比较

break 语句和 continue 语句虽然都是结束循环的语句，但要注意 continue 只结束本次循环，而 break 结束整个循环。

例如有下面两个循环：

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

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

它们的区别从流程图（如图 4-6 所示）上可以很容易看出来。从程序流程图中不难看出区别：当表达式 2 值为真时，break 语句直接跳出整个循环体，执行后面的语句。而 continue 语句只是跳过语句 2，接着进行循环判断，如条件满足，接着执行循环体，只有当循环判断条件不满足时，才跳出来，执行后面的语句。

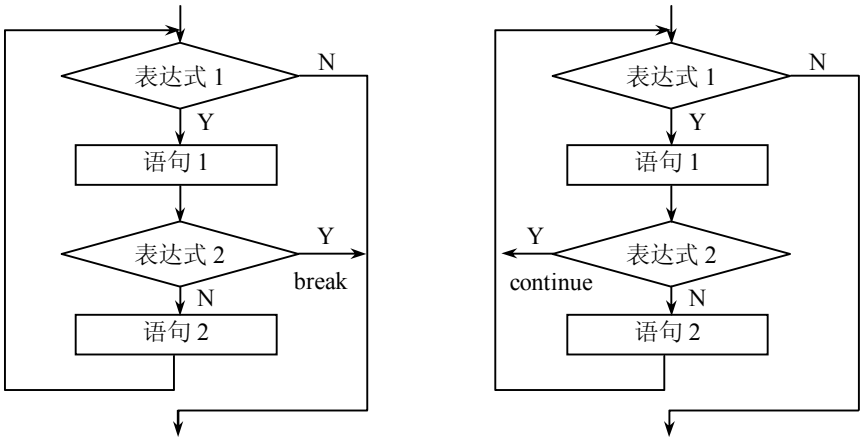

图 4-6　break 语句与 continue 语句的比较

4.2.5　语句标号和 goto 语句

1. 语句标号

语句标号是用户任意选取的标识符，其后跟一个“:”，可以放在程序中任意一条语句之前，作为该语句的一个代号。

例如 switch 语句结构中就有“default：语句 n+1”。其中“default”就是一个语句标号。

2. goto 语句

goto 语句是无条件转向语句，它使用的一般形式是：

goto 语句标号;

该语句的作用是把程序的执行转向语句标号所在的语句，这个语句标号必须和此 goto 语句处在同一函数中。

【例 4-16】统计从键盘输入一行字符的个数。

```
#include<stdio.h>
main()
{
   int n=0;
   printf("input a string:\n");
loop: if(getchar()!='\n')
   {
      n++;
      goto loop;
   }
   printf("The number is %d",n);
}
```

注意：goto 语句是一个特殊语句，在大部分高级语言中已经取消了。C 语言中虽然保留了 goto 语句，但是建议在程序中最好不用，因为 goto 语句会破坏程序设计的结构化，使得阅读程序变得困难。

4.2.6　循环的嵌套

一个循环结构内包含另一个循环结构，称为循环的嵌套。内嵌的循环中还可以嵌套循环，

即成为多重循环。例如：

```
while(表达式 1)
{语句 1
 ……
  do{语句 2
      ……
       for(表达式 3;表达式 4;表达式 5)
      {语句 3}
    }while(表达式 2)
}
```

【例 4-17】计算 $\sum_{i=1}^{100}\sum_{j=1}^{100} i \cdot j$ 的值。

```
#include<stdio.h>
main()
{
   long int i=1,j=1,sum=0;
   while(i<=100)
   {
      for(j=1;j<=100;j++)
      sum=sum+i*j;
      i++;
   }
   printf("the result is %ld",sum);
}
```

输出结果是 25502500。

在本例中用到了 while 语句和 for 语句的嵌套。在这里也可以直接用 for 语句之间的嵌套，那样程序会更简短。

4.2.7 循环程序设计实例

【例 4-18】有 1、2、3、4 四个数字，能组成多少个互不相同且无重复数字的三位数？都是多少？

算法设计：可填在百位、十位、个位的数字都是 1、2、3、4。利用循环结构组成所有的排列后再用选择结构去掉不满足条件的排列。

源程序：

```
#include<stdio.h>
void main()
{
   int i,j,k,m=0;
      for(i=1;i<=4;i++)
        {for(j=1;j<=4;j++)
           {for(k=1;k<=4;k++)
              {
```

```
            if(i!=k&&i!=j&&j!=k)
             {printf("%d%d%d\n",i,j,k);
              m++;
              }
            }
          }
        }
    printf("the number is %d",m);
}
```

【例 4-19】有一对兔子，从出生后第三个月起每个月都生一对兔子，小兔子长到第三个月后每个月又生一对兔子，假如兔子都不死，问每个月的兔子总数为 n 对？（50 个月内）

算法设计：首先找出兔子数量随时间变化的规律，兔子的数量满足 1，1，2，3，5，8，13，21…，很容易发现，某一个月的兔子数等于它相邻的前两个月的兔子数之和，这样就很容易得到源程序。

源程序：

```
#include<stdio.h>
main()
{
   long f1,f2;
   int i;
   f1=f2=1;
   for(i=1;i<=20;i++)
   {printf(" %ld %ld",f1,f2);
     if(i%2==0) printf("\n");
     f1=f1+f2;
     f2=f1+f2;
   }
}
```

【例 4-20】将一个正整数分解质因数。例如：输入 18，输出 18=2*3*3。

算法设计：对 n 进行分解质因数，应先找到一个最小的质数 k，然后按下述步骤完成：

（1）如果这个质数恰等于 n，则说明分解质因数的过程已经结束，打印出即可。

（2）如果 n>k，但 n 能被 k 整除，则应打印出 k 的值，并用 n 除以 k 的商作为新的正整数 n，重复执行第 1 步。

（3）如果 n 不能被 k 整除，则用 k+1 作为 k 的值，重复执行第 1 步。

源程序：

```
#include<stdio.h>
main()
{
   int n,i;
   printf("please input a number:\n");
   scanf("%d",&n);
   printf("the result is:%d=",n);
   for(i=2;i<=n;i++)
   {
```

```
        while(n!=i)
        {
           if(n%i==0)
           { printf("%d*",i);
             n=n/i;
           }
          else break;
        }
      }
      printf("%d",n);
  }
```

4.3 小结

C 语言提供了多种形式的条件语句以构成分支结构。

（1）if 语句主要用于单向选择。

（2）if-else 语句主要用于双向选择。

（3）if-else if 语句和 switch 语句用于多向选择。

这几种形式的条件语句一般而言是可以互相替代的。

C 语言提供了三种循环语句。以下是使用循环语句时的注意事项：

（1）for 语句主要用于控制循环次数的循环结构。

（2）循环次数及控制条件要在循环过程中才能确定的循环可用 while 或 do-while 语句。

（3）三种循环语句可以相互嵌套组成多重循环。循环之间可以并列但不能交叉。

（4）可用转移语句把流程转出循环体外，但不能从外面转向循环体内。

（5）在循环程序中应避免出现死循环，即应保证循环变量的值在运行过程中可以得到修改，并使循环条件逐步变为假，从而结束循环。

C 语言语句小结如下表。

名称	一般形式
简单语句	表达式;
空语句	;
复合语句	{语句;}

条件语句	if(表达式)语句;
	if(表达式)语句 1; else 语句 2;
	if(表达式 1)语句 1; else if(表达式 2)语句 2;

开关语句	switch(表达式){ case 常量表达式：语句…}

循环语句	while(表达式)语句;
	for(表达式 1;表达式 2;表达式 3)语句;
	break;
	goto;
	continue;
	return(表达式);

习题四

一、选择题

1．以下 for 循环的执行次数是（　）。

```
for(x=0,y=0;(y=123)&&(x<4);x++);
```

A）无限循环　　B）循环次数不定　　C）4 次　　D）3 次

2．若有如下语句：

```
int x=3;
do{ printf("%d\n",x-=2);}while(!(--x));
```

则上面程序段（　）。

A）输出的是 1　　B）输出的是 1 和-2

C）输出的是 3 和 0　　D）是死循环

3．若运行以下程序时，从键盘输入 ADescriptor<CR>（<CR>表示回车），则下面程序的运行结果是（　）。

```
#include <stdio.h>
main()
{char c;
 int v0=1,v1=0,v2=0;
 do{switch(c=getchar())
    {case 'a':case 'A':
     case 'e':case 'E':
     case 'i':case 'I':
     case 'o':case 'O':
     case 'u':case 'U':v1+=1;
     default:v0+=1;v2+=1;
    }
  }while(c!='\n');
 printf("v0=%d,v1=%d,v2=%d\n",v0,v1,v2);
}
```

A）v0=7,v1=4,v2=7　　B）v0=8,v1=4,v2=8

C）v0=11,v1=4,v2=11　　D）v0=12,v1=4,v2=12

4．下面有关 for 循环的正确描述是（　）。

A）for 循环只能用于循环次数已经确定的情况

B）for 循环是先执行循环体语句，后判断表达式

C）在 for 循环中，不能用 break 语句跳出循环体

D）for 循环的循环体语句中，可以包含多条语句，但必须用花括号括起来

5．C 语言中 while 和 do-while 循环的主要区别是（　）。

A）do-while 的循环体至少无条件执行一次

B）while 的循环控制条件比 do-while 的循环控制条件更严格

C）do-while 允许从外部转到循环体内

D）do-while 的循环体不能是复合语句

6．有如下程序：

```
main()
{  float x=2.0,y;
   if(x<0.0) y=0.0;
   else if(x<10.0) y=1.0/x;
   else y=1.0;
   printf("%f\n",y);
}
```

该程序的输出结果是（　）。

A）0.000000　　B）0.250000　　C）0.500000　　D）1.000000

7．有以下程序：

```
main()
{ int a=5,b=4,c=3,d=2;
  if(a>b>c)
     printf("%d\n",d);
  else if((c-1>=d)==1)
     printf("%d\n",d+1);
  else
     printf("%d\n",d+2)
}
```

执行后输出结果是（　）。

A）2　　B）3

C）4　　D）编译时有错，无结果

8．以下程序的输出结果是（　）。

```
main()
{ int a, b;
   for(a=1, b=1; a<=100; a++)
      {  if(b>=10) break;
         if(b%3==1)
         { b+=3; continue; }
      }
   printf("%d\n",a);
}
```

A）101　　B）6　　C）5　　D）4

二、填空题

1．以下程序运行后的输出结果是_____。

```
main()
{ int a=1,b=3,c=5;
  if(c=a+b) printf("yes\n");
  else printf("no\n");
}
```

2．以下程序运行后的输出结果是_____。

```
main()
{ int i,m=0,n=0,k=0;
  for(i=9; i<=11;i++)
      switch(i/10)
      { case 0: m++;n++;break;
        case 10: n++; break;
        default: k++;n++;
      }
  printf("%d %d %d\n",m,n,k);
}
```

3．执行以下程序后，输出'#'号的个数是_____。

```
#include <stdio.h>
main()
{int i,j;
  for(i=1; i<5; i++)
     for(j=2; j<=i; j++) putchar('#');
}
```

4．设有以下程序：

```
main()
{ int n1,n2;
  scanf("%d",&n2);
  while(n2!=0)
    { n1=n2%10;
      n2=n2/10;
      printf("%d",n1);
    }
}
```

程序运行后，如果从键盘上输入 1298，则输出结果为_____。

5．以下程序的输出结果是_____。

```
main()
{ int s,i;
  for(s=0,i=1;i<3;i++,s+=i);
  printf("%d\n",s);
}
```