第 5 单元

循环结构——程序段重复执行

第 5 单元
知识导图

结构化程序设计有顺序、选择和循环三种结构。在实际应用中，许多问题都会涉及重复执行一些操作。例如，级数求和类问题、迭代递推类问题等。程序中的某段代码有被重复执行的需求，代码被重复执行就是循环结构。循环结构中某个代码段被重复执行，执行次数可由**某一条件来控制，这个条件称为循环条件，被重复执行的代码段**称为**循环体**。本单元主要介绍三种循环结构语句 while、do-while、for 和循环控制语句 break、continue 的使用。

第 5.1 关

第 5.1 关　认识循环

知识点

知识点 5.1.1　三种循环语句

理解提高

1. while 循环语句

C 语言中用 while 语句来实现“当型”循环结构，它的一般形式如下：

```
while(循环条件){
    循环体;
}
```

说明：

（1）首先求解循环条件，若为真，则执行循环体，否则结束循环。

（2）循环体执行完成，自动**跳转**到循环开始（while）处，再次求解循环条件，如果成立就开始下一次循环，如此往复。

（3）循环体只能是一个语句。如果有多个语句，则应该用花括号将其括起来使之成为一个复合语句；如果循环体只是一个语句，花括号也可以省略。

while 循环语句的流程如图 5-1 所示。

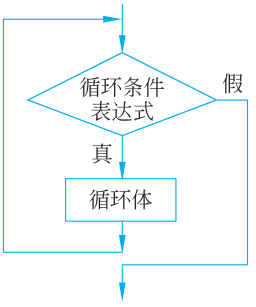

图 5-1　while 循环语句流程

示例代码 5.1.1.1　输出从 1 加到 10 的和

示例代码如下：

```
#include<stdio.h>
int main(){
    int s,i;
    s=0;
```

```
    i=1;
    while(i<=10){
        printf("%d,",i);
        s=s+i;
        i=i+1;
    }
    printf("\nS=%d",s);
    return 0;
}
```

2. do-while 循环语句

C 语言还提供了 do-while 语句用来实现“直到型”循环结构，它的一般形式如下：

```
do{
    循环体;
}while(循环条件);
```

说明：

(1) 在此结构中 do 相当于一个标号，标志循环结构开始。

(2) 首先无条件地执行一次循环体，然后求解循环条件，若为真，则跳转到 do 处再次执行循环体；若循环条件为假，则结束循环。

(3) 循环体只能是一个语句。如果有多个语句，则应该用花括号将其括起来使之成为一个复合语句；如果循环体只有一条语句，花括号可以省略。

(4) do-while 结构整体上是一条语句，所以 while 的括号后应加上分号。

do-while 循环语句的流程如图 5-2 所示。

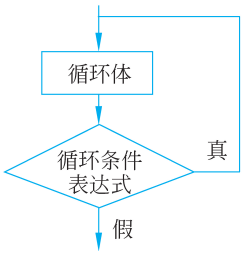

图 5-2　do-while 循环语句流程

以上两种循环比较，可以理解为：while 循环语句是在入口处判断条件，do-while 循环语句是在出口处判断条件。

示例代码 5.1.1.2　输出从 1 加到 10 的和

示例代码如下：

```
#include<stdio.h>
int main(){
    int s,i;
    s=0;
    i=1;
    do{
        printf("%d,",i);
        s=s+i;
        i=i+1;
    }while(i<=10);
    printf("\nS=%d",s);
    return 0;
}
```

3. for 循环语句

除了 while 循环语句和 do-while 循环语句,C 语言还提供了另一个使用更为广泛的循环语句——for 循环语句。for 循环语句的一般形式如下:

```
for(表达式 1;表达式 2;表达式 3){
    循环体;
}
```

说明:

(1) 首先求解表达式 1(该表达式只在这一步骤处被求解一次)。

(2) 求解表达式 2(循环条件),若为真则执行循环体,否则结束 for 语句。

(3) 循环体执行结束后,再求解表达式 3,然后转向步骤(2)。

(4) 循环体如果只是一条语句,花括号可以省略。

for 语句的流程如图 5-3 所示。

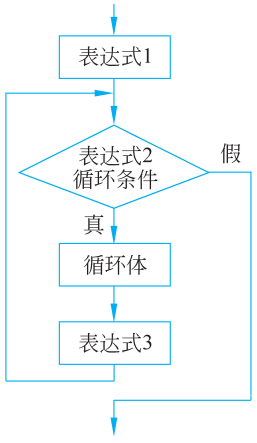

图 5-3 for 循环语句流程

示例代码 5.1.1.3 输出从 1 加到 10 的和

示例代码如下:

```
#include<stdio.h>
int main(){
    int s,i;
    s=0;
    for(i=1;i<=10;i++){
        printf("%d,",i);
        s=s+i;
    }
    printf("\nS=%d",s);
    return 0;
}
```

引导任务 5.1.1 从 1 加到 N

引导任务

认识基础

任务描述:

有数学王子之称的著名德国数学家高斯(1977—1855),和阿基米德、牛顿、欧拉并列称

为世界四大数学家，一生成就极为丰硕。

高斯 10 岁的时候，数学课上老师布置了一道题数学题：从 1 一直加到 100 等于多少？全班只有高斯算出了答案 5050，而且很快。起初老师并不相信高斯算出了正确答案，高斯就解释他是如何找到答案的，他发现：1＋100＝101，2＋99＝101，3＋98＝101，……，49＋52＝101，50＋51＝101，一共有 50 对和为 101 的数，所以答案是 101×50＝5050。

现在，老师给你布置一道编程题，输出从 1 加到 N 的和。你能做到吗？

输入格式：

一个整数 N(1≤N≤10000)。

输出格式：

一个整数，从 1 加到 N 的和。

输入样例：	输入样例：	输入样例：	输入样例：
1	10	100	1000
输出样例：	输出样例：	输出样例：	输出样例：
1	55	5050	500500

任务分析：

我们都知道，从 1 到 n 的自然数是一个等差数列，首项是 1，公差也是 1，共 n 项。所以根据等差数列的求和公式 $S=\frac{(a_1+a_n)n}{2}$，很容易得到从 1 加到 n 的和为 $\frac{n(n+1)}{2}$。

任务代码 5.1.1　从 1 加到 N_解法 1_根据等差数列求和公式直接求解

解法 1：根据等差数列求和公式求解。任务代码如下：

```
#include<stdio.h>
int main(){
    int n,s;
    scanf("%d",&n);                    //输入整数 n
    s=n*(n+1)/2;                       //公式直接计算从 1 加到 n 的和 s
    printf("%d",s);                    //输出结果
    return 0;
}
```

代码测试与分析：

在 Dev C++ 中执行以上代码：

输入：1　　输出：1　　（测试边界数据，可能的最小值）

输入：10　　输出：55

输入：100　　输出：5050

输入：1000　　输出：500500

输入：100000　　输出：5000050000

以上代码读者一定都能理解，现在讨论另一种解法，就是首先设置一个**和变量** s＝0，用来存储最终的和，初值为 0。然后设置**计数器变量** i，让 i 的值**从 1 到 n** 变化，**每一次都重复执行把 i 累加到变量 s 里**(s＝s＋i)。最后，变量 s 的值就是所求，输出即可。

“每一次都重复执行把 i 累加到变量 s 里”，这就是重复操作，**计数器 i 从 1 数到 n** 是重

复的次数。这里的**重复操作**就是循环的思想。我们可以用下面的算法来描述输出从1加到n的和的求解过程(算法流程见图5-4):

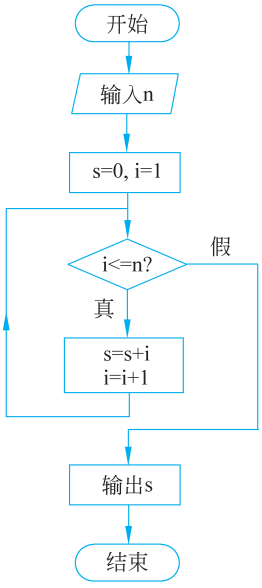

图5-4 算法流程图

(1) 输入变量n,定义和变量s=0(初值),计数器变量i=1(初值)。

(2) 如果i<=n成立就向下执行,如果不成立转到步骤(6)。

(3) s=s+i。

(4) i=i+1。

(5) 转到步骤(2)。

(6) 输出s。

以上就是"**求解从1加到n的和**"算法的形式化描述,其中的步骤(3)和步骤(4)就是被重复执行的部分,称为**循环体**。计数器变量i称为**循环变量**,i<=n称为**循环条件**。

任务代码5.1.1 从1加到N_解法2_使用while循环

解法2:使用while循环语句求解。任务代码如下:

```
#include<stdio.h>
int main(){
    int n,s,i;          //定义变量
    scanf("%d",&n);     //输入整数 n                    ---算法步骤(1)
    s=0; i=1;           //和变量 s,计数器变量 i 赋初值   ---算法步骤(1)
    while(i<=n){        //满足循环条件就进入循环         ---算法步骤(2)
        s=s+i;          //将变量 i 的值累加到和变量 s 中  ---算法步骤(3)
        i=i+1;          //计数器 i 向后计数              ---算法步骤(4)
    }                   //                              ---算法步骤(5)
    printf("%d",s);     //输出结果                       ---算法步骤(6)
    return 0;
}
```

代码测试与分析:

在 Dev C++ 中执行：

输入：1　　输出：1　　（测试边界数据，可能的最小值）

输入：10　　输出：55

输入：100　　输出：5050

输入：1000　　输出：500500

结合 while 语句的语法规则，结合上文中的算法，这个代码是不是很好理解呢？

这就是让循环变量 i 从 1 到 n，循环体执行 n 次的代码框架，可以称之为**计次循环（循环次数固定可数）**，请你一定要牢记这个结构：

```
i=1;                                    //循环变量赋初值
while(i<=n){                            //进入循环的条件
    循环体;
    i=i+1;                              //循环变量的增量
}
```

任务代码 5.1.1　从 1 加到 N_解法 3_使用 do-while 循环

解法 3：使用 do-while 循环语句求解。任务代码如下：

```
#include<stdio.h>
int main(){
    int n,s,i;
    scanf("%d",&n);                     //输入整数 n
    s=0; i=1;                           //和变量 s 赋初值 0,计数器变量 i 赋初值 1
    do{                                 //循环开始的标记
        s=s+i;                          //将变量 i 的值累加到和变量 s 中
        i=i+1;                          //计数器 i 向后计数
    }while(i<=n);                       //满足循环条件就再次进入循环
    printf("%d",s);                     //输出结果
    return 0;
}
```

代码测试与分析：

在 Dev C++ 中执行：

输入：1　　输出：1　　（测试边界数据，可能的最小值）

输入：10　　输出：55

输入：100　　输出：5050

输入：1000　　输出：500500

解法 3 的代码同样可以实现任务要求的功能，程序的原理和执行过程和解法 2 是完全一致的，只不过 do-while 循环是先执行一次循环体，再判断循环条件，对于输入的 n＞＝1 时，两种解法的输出结果都是一致的，都可以实现题目要求。

任务代码 5.1.1　从 1 加到 N_解法 4_使用 for 循环

解法 4：使用 for 循环语句求解。任务代码如下：

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

```
    int n,s,i;
    scanf("%d",&n);                    //输入整数 n
    s=0;                               //和变量 s 赋初值 0
    for(i=1;i<=n;i++){                 //典型的计次循环,i 从 1 开始到 n 结束,每次加 1
        s=s+i;                         //循环体
    }
    printf("%d",s);                    //输出结果
    return 0;
}
```

代码测试与分析:

在 Dev C++ 中执行:

输入:1　　输出:1　　(测试边界数据,可能的最小值)

输入:10　　输出:55

输入:100　　输出:5050

输入:1000　　输出:500500

从以上代码和测试结果可以看出,for 循环也可以方便地表达**计次循环**,"for (i=1; i<=n; i++)"清楚地说明了循环变量 i 从 1 开始,到 n 结束,每次加 1。

4. while 循环和 for 循环的转换

解法 4 代码从执行逻辑上看,先执行 i=1,然后进入循环"i<=n→s=s+i→i++",直到 i<=n 不成立。这跟 while 语句的逻辑是相同的,可见 while 循环和 for 循环是可以互相转换的,具体情形如表 5-1 所示。

表 5-1　while 循环和 for 循环的转换

while 循环	for 循环

表达式 1;
while(循环条件表达式 2){
　循环体;
　表达式 3(循环变量自加或自减);
}
	for(表达式 1;循环条件表达式 2;表达式 3){
	　循环体;
	}

s=0;
i=1;
while(i<=10){
　s=s+i;
　i++;
}
	s=0;
	for(i=1;i<=10;i++){
	　s=s+i;
	}

while 循环和 for 循环可以方便地互相转换,在设计程序时可以根据问题实际需要选择一种使用。

你要牢记,让循环变量 i 从 1 变到 n,循环体执行 n 次的**计次循环(循环次数固定可数)**代码框架如下:

```
for(i=1;i<=n;i++){
    循环体;
}
```

5. 穷举法

穷举法，又称枚举法、列举法，是一种在计算机科学和数学等领域广泛应用的基本算法策略。穷举法是指在一个有穷可能解的集合中，按照一定的顺序，逐个地对每一个可能的解进行考察检验，从中找出符合要求的解。

例如，输出从 1 加到 N 的和，可以将 1 到 N 的所有整数一一列举（穷举），每个数都累加到变量 S 中，最后 S 中的数就是所求。

例如，输出自然数 N 的阶乘，需要将 1 到 N 的所有整数一一列举（穷举），每个数都累乘到变量 F 中，最后 F 中的数就是所求。

例如，输出 1 到 N 的奇数，可以将 1 到 N 的所有整数一一列举（穷举），符合条件（奇数）就输出。

例如，输出 1 到 N 的素数，需要将 1 到 N 的所有整数一一列举（穷举），是素数的就输出。

例如，输出 N 的所有约数，需要将 1 到 N 的所有整数一一列举（穷举），如果是 N 的约数就输出。

穷举法简单直接，不需要复杂的算法设计和数学推导，易于理解和实现。穷举法结果准确，只要解空间是有限的，且枚举过程正确，就一定能得到问题的所有解或最优解。

穷举法当解空间规模较大时效率较低，但它是一种基础且重要的算法思想，在很多实际问题中仍然有着广泛的应用，尤其是在对效率要求不高、解空间较小的场景中，穷举法往往能发挥重要作用。

6. goto 语句*

C 语言中的 goto 语句可以实现无条件跳转，与标号语句一起可以构成循环。例如，以下代码也可以实现从 1 加到 N，请自行分析。

任务代码 5.1.1　从 1 加到 N_解法 5_使用 goto 语句求解

解法 5：使用 goto 语句求解。任务代码如下：

```
#include<stdio.h>
int main(){
    int n,s,i;
    scanf("%d",&n);
    s=0;
    loop:                                   //定义标号语句,标识位置
        s=s+i;
        i++;
        if(i<=n)
            goto loop;                      //跳转到标号 loop 处
    printf("%d",s);
    return 0;
}
```

goto 虽然可以方便地实现跳转，但过度使用会使程序的控制流程变得混乱，难以理解和维护，导致代码的可读性极差，并增加调试难度。所以虽然 C 语言保留了 goto 语句，现在几乎没有人使用，以更好地保持程序的可读性、可维护性和可扩展性。

知识点 5.1.2 循环控制语句

知识点 理解提高

C语言提供了两个控制循环进程的语句：**break语句**和**continue语句**。

1. break语句

break语句用于从循环体内跳出循环结构。当程序执行到break语句时，会立即跳出循环结构，结束循环。

2. continue语句

continue语句用于结束本次循环，开始下次循环。当执行到continue语句时，会立即结束本次循环(跳过循环体中后面的部分不执行)，直接跳转到循环开始处，执行下一次循环。

任务代码 5.1.1 从1加到N_解法6_使用循环控制语句

解法6：使用循环控制语句求解。任务代码如下：

```
#include<stdio.h>
int main(){
    int n,s,i;
    scanf("%d",&n);                    //输入整数n
    s=0; i=1;                          //和变量s赋初值0,计数器变量i赋初值1
    while(1){                          //进入循环时相当于无条件(1为真)
        s=s+i;                         //累加
        i=i+1;                         //计数器加1
        if(i>n) break;                 //i>n时跳出循环
        else    continue;              //删除这行语句,程序功能会有变化吗,为什么?
    }
    printf("%d",s);                    //输出结果
    return 0;
}
```

代码分析：

以上代码中进入循环时的条件为1(真)，表示永远允许进入循环，循环体内通过if语句判断，如果i>n成立就执行break语句跳出循环。

请思考：删除程序中if语句的else分支，程序功能会有变化吗？为什么？应用do-while语句、for语句和控循环制语句结合，你还有哪些不同的解法呢？

引导任务 5.1.2 找奇数

引导任务 认识基础

任务描述：

输入一个正整数N，输出从1到N所有的奇数。

输入格式：

一个正整数，不大于10000。

输出格式：

输出从1到N所有的奇数，用逗号分隔。

输入样例：

20

输出样例：

1,3,5,7,9,11,13,15,17,19

输入样例：

100

输出样例：

1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,51,53,55,57,59,61,63,65,67,69,71,73,75,77,79,81,83,85,87,89,91,93,95,97,99

任务分析：

输出从 1 到 N 的所有奇数，很明显问题的解在[1,N]内，可以使用穷举法一一列举从 1 到 N 的所有整数，如果是奇数，则输出。这也是典型的计次循环问题，循环的次数事先是已知的，是一一可数的。

任务代码 5.1.2　找奇数_解法 1_使用 for 语句穷举(有瑕疵的代码)

解法 1：使用 for 语句穷举(有瑕疵的代码)。任务代码如下：

```
#include<stdio.h>
int main(){
    int n,i;
    scanf("%d",&n);                          //输入整数 n
    for(i=1;i<=n;i++){                       //典型的计次循环
        if(i%2==1)                           //如果是奇数,则输出并加逗号
            printf("%d,",i);
    }
    return 0;
}
```

代码测试与分析：

在 Dev C++ 中执行：

输入：10　　输出：1,3,5,7,9,

输入：21　　输出：1,3,4,7,9,11,13,15,17,19,21,

输入：1　　输出：1,

从 1 到 n 的循环是典型的计次循环问题，用 for 语句很合适。for(i=1;i<=n;i++)表示循环变量 i 从 1 开始，到 n 结束，每次加 1，事先就可以明确得知这是一个计次循环问题，共循环 n 次。

在每次循环的循环体内，判断如果 i 是奇数就输出 i，根据题目中的要求，每个 i 的后面加一个逗号。

从测试输出结果可以看出，以上程序的输出实际和题目的要求相比，在最后一个奇数的后边多了一个逗号，所以这个代码在线上平台中提交是通不过的，显示：答案错误。

这个问题如何解决呢？方法就是对逗号的输出单独控制，因为第 1 个输出肯定是 1，是已知的。所以可以使用这样的逻辑：控制在 1 的前面不输出逗号，其他数据的前面输出逗号，这个问题就解决了。

任务代码 5.1.2　找奇数_解法 2_使用 for 语句穷举(正确的代码)

解法 2：使用 for 语句穷举(正确的代码)。任务代码如下：