

## 大数据与自动化学院 2020 年公招计算机类教师试讲内容

### C 语言循环结构

1. 为什么要用循环结构
2. 循环结构流程图
3. 循环语句
4. 循环案例

试讲教材资料考生自己准备

附：C 语言循环参考资料

(4) 条件运算符的结合方向为“自右至左”。例如：  
 $x > 0 ? 1 : x < 0 ? -1 : 0$  等效于  $x > 0 ? 1 : (x < 0 ? -1 : 0)$

(5) 表达式 1、表达式 2 和表达式 3 的类型可以不同，此时条件表达式的值的类型为它们中较高的类型。

**【例 3-12】** 从键盘输入一个字符，如果是大写字母，将它转换成小写字母并输出，如果是小写字母直接输出。

算法设计：

- (1) 定义字符型变量  $a, b$ ;
- (2) 判断输入字符的大小写，若是大写字母转换为小写字母，小写字母的 ASCII 码值比相应大写字母的 ASCII 码值大 32;
- (3) 输出转换后的字符。

程序代码：

```
#include <stdio.h>
void main()
{
    char ch;
    printf("Please enter a character: \n");
    scanf("%c", &ch);
    ch = (ch >= 'A' && ch <= 'Z') ? (ch + 32) : ch;
    printf("%c\n", ch);
}
```

程序结果：

Please enter a character:

A  
a

### 3.4 循环结构程序设计

循环结构是程序中一种很重要的结构，其特点是在给定条件成立时反复执行某程序段，直到条件不成立为止。给定的条件称为循环条件，反复执行的程序段称为循环体。C 语言提供了 3 种实现循环结构的循环语句，即 while 循环语句、do...while 循环语句、for 循环语句。

#### 3.4.1 while 语句

格式：

```
while (条件表达式)
{ 循环体 }
```

功能:先计算条件表达式的值,当值为真(非0)时执行循环体语句,当条件表达式立(值为0)时结束循环,执行循环体外的语句,其执行过程如图3-3(a)所示。

**【例3-13】** 求  $\text{sum} = \sum_{n=1}^{100} n$ 。

算法设计:

累加运算是一个表达式( $\text{sum} = \text{sum} + i$ )多次执行,执行100次,所以在累加运算中开始至100依次累加,在程序设计中将变量*i*作为循环控制变量。

- (1) 设累加变量  $\text{sum} = 0$ ;
- (2) 循环控制变量*i*赋初值1,设定循环条件为  $i \leq 100$  (即循环控制变量终值为100);
- (3) 输出  $\text{sum}$  的值。

程序代码:

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

/\* 为循环控制变量*i*赋初值1 \*/  
/\* 循环控制变量的终值为100 \*/

/\* 每循环一次,循环控制变量增加1 \*/

在使用 while 语句时应注意以下几点:

- (1) while 语句中的表达式一般是关系表达式或逻辑表达式,只要表达式的值为真(非0)即可继续循环。
- (2) 循环体如包括一个以上的语句,则必须用{}括起来,组成复合语句。如果不包括括号,则 while 语句的范围只到 while 后面的第一个分号处。
- (3) 在循环体中应有使循环趋向于结束的语句,否则循环就是死循环,如上例中“ $i++$ ”语句。

**【例3-14】** 求  $p = n!$  (从键盘输入一个整数存入变量*n*)。

算法设计:

累乘运算是一个表达式( $p = p * i$ )多次执行,执行*n*次,所以在累乘运算中从1开始依次累乘,在程序设计中一般用变量*i*作为循环控制变量。

- (1) 设定变量  $p = 1, i = 1$ ;
- (2) 设定循环条件为  $i \leq n$ , *n* 由键盘输入,循环语句为“ $p = p * i;$ ”;
- (3) 输出  $\text{sum}$  的值。

程序代码:

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

```
{
    long p = 1;
    int i, n;
    scanf("%d", &n);
    i = 1;
    while(i <= n)
    {
        p = p * i;
        i++;
    }
    printf("%ld\n", p);
}
```

**【例3-15】** 计算

算法设计:

- 对从2到*n*-1
- (1) 设定变量“*s*”
- (2) 设定循环“*k*”
- (3) 输出*s*的

程序代码:

```
#include <stdio.h>
void main()
{
    int n, k, s;
    scanf("%d", &n);
    k = 2;
    while(k < n)
    {
        if(n % k == 0)
            s++;
        k++;
    }
    printf("%d\n", s);
}
```

**【例3-16】**

算法设计:

- (1) 输入
- (2) 当
- (3) 再
- (4) 输出

程序代

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

```
scanf("%d", &n);
while(n != 0)
{
    if(n % 2 != 0) js++;
    else os++;
    scanf("%d", &n);
}
printf("js = %d, os = %d\n", js, os);
}
```

/\* 从键盘获取下一个循环操作数据 \*/

### 3.4.2 do...while 语句

格式:

```
do
{
    循环体
}while(条件表达式);
```

do 循环与 while 循环的不同在于它先执行循环中的语句,然后判断表达式是否为“真”,如果为“真”则继续循环,如果为“假”则终止循环,因此 do...while 循环至少要求执行一次循环语句。其执行过程如图 3-3(b)所示。

**【例 3-17】** 用 do...while 语句求 100~200 之间的奇数和。

```
#include <stdio.h>
void main()
{
    int i, sum = 0;
    i = 100;
    do
    {
        if(i % 2 == 1)
            sum = sum + i;
        i++;
    } while(i <= 200);
    printf("%d\n", sum);
}
```

注:当循环体内有多条语句时要用“{}”把它们括起来。

**【例 3-18】** while 和 do...while 循环比较。

下面两种编写方式有什么区别,请读者上机运行,运行时输入数据为 15,注意检查结果。

(1)

```
#include <stdio.h>
void main()
{
    int sum = 0, i;
    scanf("%d", &i);
    while(i <= 10)
```

```
{
    sum = sum + i;
    i++;
}
printf("sum = %d\n", sum);
}
```

(2)

```
#include <stdio.h>
void main()
{
    int sum = 0;
    scanf("%d", &i);
    do
    {
        sum = sum + i;
        i++;
    } while(i <= 10);
    printf("sum = %d\n", sum);
}
```

### 3.4.3

在 C 语言  
用 while 语句

格式:

```
for(表达式1;
    表达式2;
    表达式3)
```

执行过程

(1) 先

(2) 再

循环体,然后

到第(5)步

(3)

(4)

(5)

其

for

变量

```

    sum = sum + i;
    i++;
}
printf("sum = %d", sum);
}

```

(2)

```

#include <stdio.h>
void main()
{
    int sum = 0, i;
    scanf("%d", &i);
    do
    {
        sum = sum + i;
        i++;
    } while(i <= 10);
    printf("sum = %d", sum);
}

```

### 3.4.3 for 语句

在C语言中 for 语句的使用最为灵活,一般用于计数型循环,也可以用于条件型循环,用 while 语句和 do...while 语句所能解决的问题大部分可以用 for 语句来解决。

格式:

```

for(表达式1; 表达式2; 表达式3)
{ 循环体; }

```

执行过程如下:

- (1) 先求解表达式1。
- (2) 求解表达式2,若其值为真(非0),则执行 for 语句中的循环体,然后执行下面的第(3)步;若其值为假(0),则结束循环,转到第(5)步。

(3) 求解表达式3。

(4) 转回上面第(2)步继续执行。

(5) 循环结束,执行 for 语句下面的一个语句。

其执行过程可以用图 3-5 表示。

for 语句也可以用下面的形式表示:

```

for(循环控制变量赋初值; 循环条件; 循环控制变量增量)
{ 循环体 }

```

循环控制变量赋初值总是一个赋值语句,它用来给循环控制变量赋初值;循环条件是一个关系表达式,它决定什么时候退出

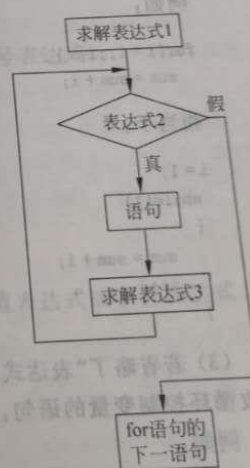


图 3-5 执行过程

循环: 循环变量增量定义循环控制变量每循环一次后按什么方式变化, 这3个部分之间用“;”分开。

例如:

```
for(i=1; i<=100; i++)
    sum = sum + i;
```

先给循环控制变量*i*赋初值1, 判断*i*是否小于等于100, 若为真, 则执行语句“sum+i”, 之后*i*值增加1。再重新判断*i*是否小于等于100, 重复执行语句, 直到条件*i*>100时结束循环。相当于:

```
i=1;
while(i<=100)
{
    sum = sum + i;
    i++;
}
```

for 循环语句的一般形式对应以下 while 循环形式:

```
表达式1;
while(表达式2)
{
    语句;
    表达式3;
}
```

注意:

- (1) for 循环语句中的“表达式1(循环控制变量赋初值)”、“表达式2(循环条件)”、“表达式3(循环控制变量增量)”都是可选项, 即可以省略, 但“;”不能省略。
- (2) 省略“表达式2(循环条件)”, 若不做其他处理, 便成为死循环。

例如:

```
for(i=1;;i++)
    sum = sum + i;
```

相当于:

```
i=1;
while(1)
{
    sum = sum + i;
    i++;
}
```

- (3) 若省略了“表达式3”, 则不对循环控制变量进行修改操作, 这时可在语句体内修改循环控制变量的语句。

例如:

```
for(i=1; i<=100;)
{
    sum = sum + i;
```

i++;

}

(4) 省略“表达式2”

例如:

```
for(; i<=100; i++)
{
    sum = sum + i;
}
```

相当于:

```
while(i<=100)
{
    sum = sum + i;
}
```

(5) 3个表达式都省略

例如:

```
for(;;)
```

相当于:

```
while(1)
```

(6) 表达式2为0

例如:

```
for(sum=0; ; i++)
```

(7) 表达式1为0

例如:

```
for(i=0; i<=100; i++)
```

或:

```
for(i=0; ; i++)
```

k=

(8)

要其值

例

for

Y

f