嵌入式C语言之-

for循环语句

讲师：叶大鹏

助力你成为优秀的电子工程师！

PN学堂

应用案例

- 需求：打印输出5个星型(*)
- 当然，我们可以使用printf("*****\n")，但是为了说明循环语句的用法，我们使用最原始，最笨的办法：

```
printf("*");
printf("*");
printf("*");
printf("*");
printf("*");
printf("\n");
```

上面的代码冗余度(重复代码)过高，而且不利于后期的代码维护(如果后期我想输出其他内容，需要修改5行代码)，所以我们可以使用循环结构语句进行简化。

C语言的程序结构

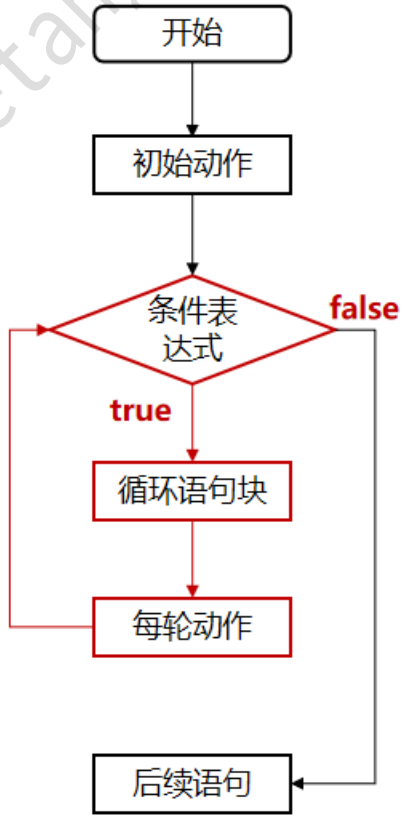
- 顺序结构，按部就班，依次执行。
- 选择结构，包括if条件语句和switch开关语句，if语句类似日常生活中的“如果...那么...否则”。
- 循环结构，包括for、while、do-while循环语句，在编写代码时，常常希望反复执行同一段代码，此时可以使用循环来完成这个功能，这样就用不着重复地写若干行相同的代码。

for语句整体介绍

- **for循环语句的语法格式：**

```
for (初始动作; 条件; 每轮的动作)
{
    循环语句块
}
后续语句
```

三个表达式之间用；隔开

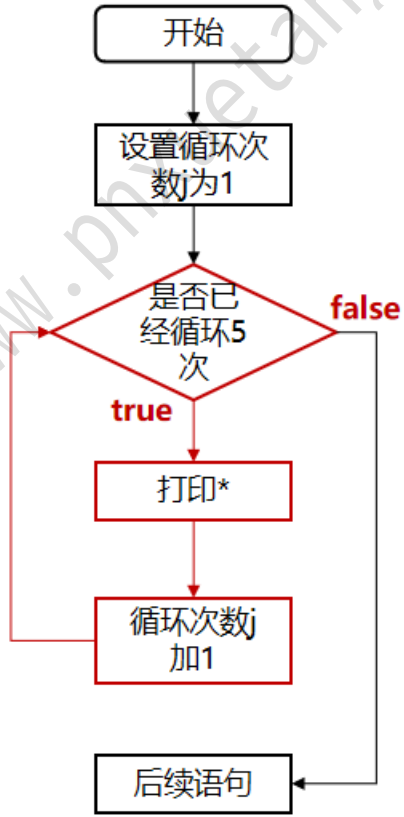


应用案例

- 需求：打印输出5个星型(*)
- 我们使用for循环语句来实现：

```
for (uint8_t j = 1; j < 6; j++)
{
        printf("*");
}
printf("\n");
```

在这个案例里，j并没有参与到循环语句块中，
它只是起到记录循环次数的作用。



应用案例

- 需求：求1到100的累加和
- 我们使用for循环语句来实现：

```
uint32_t sum = 0;
for (uint32_t i = 1; i <= 100; i++)
{
    sum +=i;
}
printf("%d\n", sum);
```

在这个案例里，i不仅起到记录循环次数的作用，
还作为累加因子参与到循环语句块中。

编程规范

1. **三个表达式之间隔1个空格；**
2. **{}要新起一行书写。**

```
for (uint32_t i = 1; i <= 100; i++)
{
    sum += i;
}
```

应用案例

- 需求：打印输出4行5个星型(*)
- 我们使用for循环语句来实现：

```
for (uint8_t i = 1; i < 6; i++)
{
        printf("*");
}
printf("\n");
for (uint8_t i = 1; i < 6; i++)
{
        printf("*");
}
printf("\n");
for (uint8_t i = 1; i < 6; i++)
{
        printf("*");
}
printf("\n");
for (uint8_t i = 1; i < 6; i++)
{
        printf("*");
}
printf("\n");
```

- 代码里，包含了4段重复代码，所以我们可以再使用for循环语句对它进行优化。

应用案例

- 需求：打印输出4行5个星型(*)
- 我们可以将

```
for (uint8_t j = 1; j < 6; j++)
{
        printf("*");
}
printf("\n");
```

看作是一个可执行语句块，接下来要做的

就是使用for循环重复执行4次，对应打印4行：

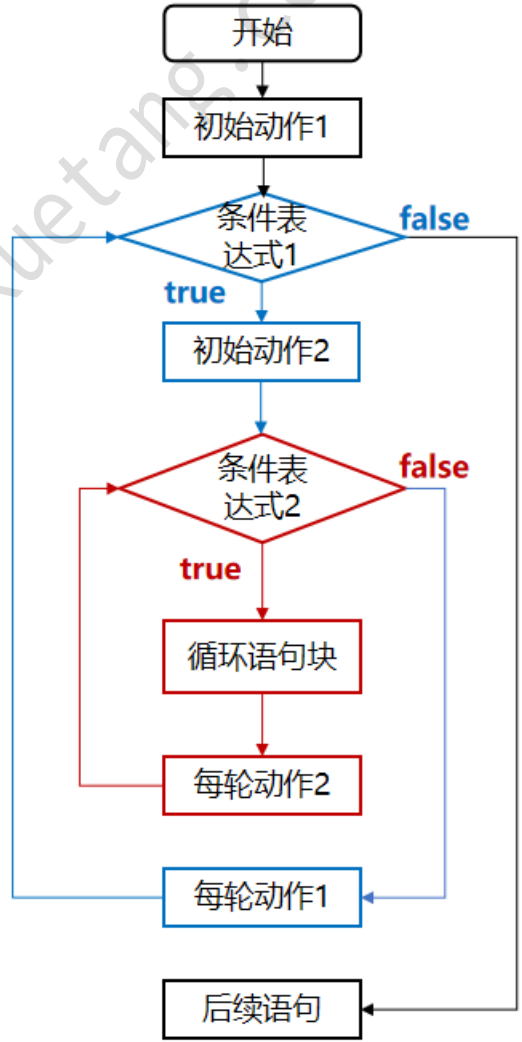
```
for (uint8_t i = 1; i < 5; i++)
{
    xxx;
}
```

```
for (uint8_t i = 1; i < 5; i++)
{
    for (uint8_t j = 1; j < 6; j++)
    {
        printf("*");
    }
    printf("\n");
}
```

for语句嵌套循环

- **语法格式：**

```
for (初始动作1; 条件1; 每轮的动作1)
{
    ...
    for (初始动作2; 条件2; 每轮的动作2)
    {
        循环语句块
    }
    ...
}
后续语句
```



应用案例

- 需求：打印输出下面格式的星型(*)

应用案例

- 需求：打印输出下面格式的星型(*)

```
*
**
***
****
```

➢ 我们来找下规律：

- 第一行：

```
for (uint8_t j = 1; j < 2; j++)
{
        printf("*");
}
printf("\n");
```

- 第二行：

```
for (uint8_t j = 1; j < 3; j++)
{
        printf("*");
}
printf("\n");
```

- 第三行：

```
for (uint8_t j = 1; j < 4; j++)
{
        printf("*");
}
printf("\n");
```

- 第四行：

```
for (uint8_t j = 1; j < 5; j++)
{
        printf("*");
}
printf("\n");
```

j < 行数值 + 1

应用案例

- 需求：打印输出下面格式的星型(*)

```
*
**
***
****
```

```
for (uint8_t i = 1; i < 5; i++)
{
    for (uint8_t j = 1; j < 行数值 + 1; j++)
    {
        printf("*");
    }
    printf("\n");
}
```

i代表行数

```
for (uint8_t i = 1; i < 5; i++)
{
    for (uint8_t j = 1; j < i + 1; j++)
    {
        printf("*");
    }
    printf("\n");
}
```

应用案例

- 讲解数组课程时，会分享更多的for循环应用案例。

for循环语句扩展

- **for循环语句的语法格式：**

```
for (初始动作; 条件; 每轮的动作)
{
    循环语句块
}
后续语句
```

三个表达式都可以省略不写，但是；不能省略

```
uint32_t sum = 0;
for (uint32_t j = 1; j <= 100; j++)
{
    sum += j;
}
printf("%d\n", sum);
```

```
uint32_t sum = 0;
uint32_t j = 1;
for (; j <= 100; )
{
    sum += j;
    j++;
}
printf("%d\n", sum);
```

for循环语句扩展

for(;;) //死循环，在rtos的任务线程里可以看到

{

}

```
static portTASK_FUNCTION( vComRxTask, pvParameters )
{
    signed char cExpectedByte, cByteRxed;
    BaseType_t xResyncRequired = pdFALSE, xErrorOccurred = pdFALSE;

    /* Just to stop compiler warnings. */
    ( void ) pvParameters;

    for( ; ; )
    {
```

for循环语句扩展

- 需求：求1到100的累加和

```
uint32_t sum = 0;
for (uint32_t i = 1; i <= 100; i++)
{
    sum +=i;
}
printf("%d\n", sum);
```

```
uint32_t sum = 0;
for (uint32_t i = 1, j = 100; i <= j; i++, j--)
{
    sum += i + j;
}
printf("%d\n", sum);
```

- **for循环的三个表达式可以使任何合法的C语言表达式：**

1. 当需要为多个变量赋初值时，表达式1可以用逗号表达式顺序的执行为多个变量赋初值的操作；

2.当每轮动作需要有多个变量发生变化时，也可以使用逗号表达式。

编程规范

1. **即使可执行循环语句块只有一条语句，也不要省略{}，造成代码结构不清晰，不推荐下面的风格：**

```
for (uint32_t i = 1; i <= 100; i++)
        sum +=i;
printf("%d\n", sum);
```

THANK YOU!