

# 嵌入式C语言之- while循环语句

讲师：叶大鹏

助力你成为优秀的电子工程师！



# while语句整体介绍

- while循环语句的语法格式:

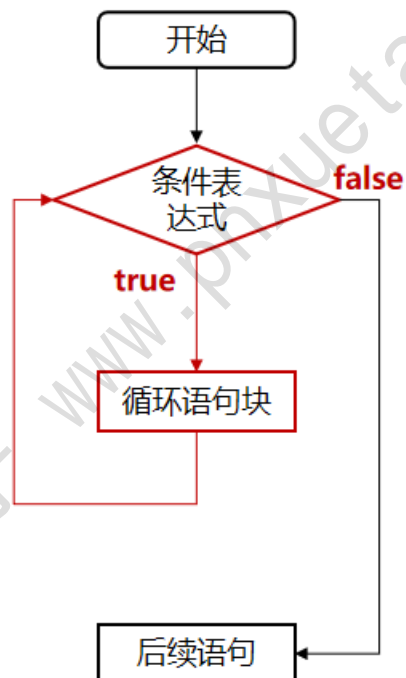
```
while (条件表达式)
```

```
{
```

```
    循环语句块
```

```
}
```

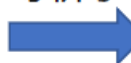
```
后续语句
```



# 应用案例

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

等价于



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

# 循环的选择

- for循环可以实现while循环的功能，while循环也可以实现for循环的功能，没有哪个更好的说法，要看应用场景；
- for循环通常用在事先知道总循环次数的场景；
- while循环通常用在事先不知道循环次数的场景，而是以达到某个目标为目的。

```
static void sgp30_thread(void *arg)
{
    int16_t status = 0;

    /* 对传感器进行初始化，循环等待，直到成功为止 */
    while (status != 1)
    {
        status = sgp_probe();
        vTaskDelay(1000 / portTICK_RATE_MS);
    }
    HK_LOG_I(TAG, "SGP sensor probing successful!");

    ...
}
```

# 循环语句的特殊用法：死循环

1. 在裸机程序的main函数里，完成初始化以后，需要在死循环内反复的执行业务逻辑；
2. 在RTOS的线程里，同样是完成初始化以后，需要在死循环内反复的执行业务逻辑；

- 方法1，使用while语句

```
while(1)
```

```
{
```

```
    //code
```

```
}
```

- 方法2，使用for语句

```
for(;;)
```

```
{
```

```
    //code
```

```
}
```

## 循环语句的特殊用法：死循环

```
int main(void)
{
    gd_eval_led_init(LED);
    systick_config();

    while(1)
    {
        /* turn on LED1 */
        gd_eval_led_on(LED);
        /* insert 200 ms delay */
        delay_1ms(200);

        /* turn off LEDs */
        gd_eval_led_off(LED);
        /* insert 200 ms delay */
        delay_1ms(200);
    }
}
```

```
void buzzer_thread(void * arg)
{
    for(;;)
    {
        vTaskDelay(50 / portTICK_RATE_MS);
        if(g_set_state == 0)continue;
        uint32_t system_ms = xTaskGetTickCount();
        buzzer_end();
        g_set_state = 0;
    }
}
```

# 嵌入式C语言之- do-while循环语句

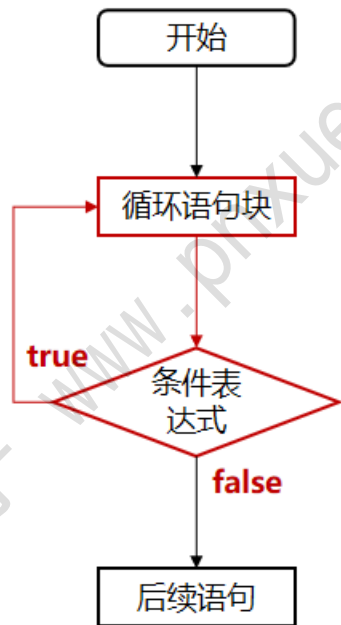
讲师：叶大鹏

助力你成为优秀的电子工程师！

# do-while语句整体介绍

- do-while循环语句的语法格式:

```
do  
{  
    循环语句块  
} while (条件表达式);  
后续语句
```





# do-while和#define配合使用

```
#define DISK_LOCK(mux) do {  
    if (pthread_mutex_lock(mux) != 0) {  
        PRINT_ERR("%s %d, mutex lock failed\n", __FUNCTION__, __LINE__);  
    }  
} while (0)
```

```
#define TRACE_HIT_CACHE(pc) do { pc->hit++; g_totalPathCacheHit++; } while (0)
```

**THANK YOU!**