

程序设计语言与方法(C语言)

第六章 循环控制结构

问题：求给定整数的累加和

- n ?
- $1+2+\dots+(n-1)+n = ?$
- 回顾1到10的累加和的程序
- 还能一个加法一个加法的做么？

描述问题的求解方法与过程

- 1. 开始；
- 2. 设X的初值为0；
- 3. 按序从Y（1.....10）中取出一个数据；
- 4. 计算X 与所取得的数据的和，并将结果放于X中；
- 5. **重复**执行步骤3和4，直到Y中的数据用完；
- 6. X上的值为计算结果；
- 7. 结束。

重复执行代码：循环控制结构

- 需要解决的问题
 - 如何重复
 - 如何判定数据是否用完
- 循环控制语句
 - goto语句
 - while
 - do ... while
 - for

描述问题的求解方法与过程

- 1. 开始;
- 2. $\text{sum} = 0$;
- 3. $i = 1$;
- 4. $\text{sum} += i$; $i++$;
- 5. 如果 $i \leq 10$, 则转4继续;
- 6. 输出 sum ;
- 7. 结束。

转4继续?

这个怎么做?

求1..10的累加和（流程转移控制）

- 标记待转目的语句的位置

- 给语句加“标号”

- 跳转

- goto

- 如何跳转

- 条件判定

```
int main()
{
    int sum = 0;
    int i = 1;
    Goon: sum += i;
    i++;
    if(i <= 10) {
        goto Goon;
    }
    printf("%d\n", sum);
}
```

语句： WHILE、FOR、 DO...WHILE

```
int main()
{
    int sum = 0;
    int i = 1;
    while(i <= 10) {
        sum += i;
        i++;
    }
    printf("%d\n", sum);
}
```

```
int main()
{
    int sum = 0;
    int i;
    for (i = 1; i <= 10; i++) {
        sum += i;
    }
    printf("%d\n", sum);
}
```

```
int main()
{
    int sum = 0;
    int i = 1;
    do {
        sum += i;
        i++;
    } while(i <= 10);
    printf("%d\n", sum);
}
```

??

```
int main()
{
    int sum = 0;
    int i = 1;
    do {
        i++;
        sum += i;
    } while(i <= 10);
    printf("%d\n", sum);
}
```

求N的累加和

```
int main()
{
    int n, i;
    long sum=0;
    scanf("%d", &n);
    for (i = 1; i <= n; i++) {
        sum += i;
    }
    printf("%d\n", sum);
}
```


求N的累加和

```
int main()
{
    int n, i;
    long sum=0;
    scanf("%d", &n);
    i = 1;
    while(i <= n) {
        sum += i;
        i++;
    }
    printf("%ld\n", sum);
}
```

求N的累加和

```
int main()
{
    int n, i;
    long sum=0;
    scanf("%d", &n);
    i = 1;
    do {
        sum += i;
        i++;
    } while(i <= 10);
    printf("%d\n",
    sum);
}
```

嵌套循环

- 问题：求 $1! + 2! + 3! + 4! + \dots + n!$
- 本质上是求和
 - 1..n
 - 各个数据的阶乘

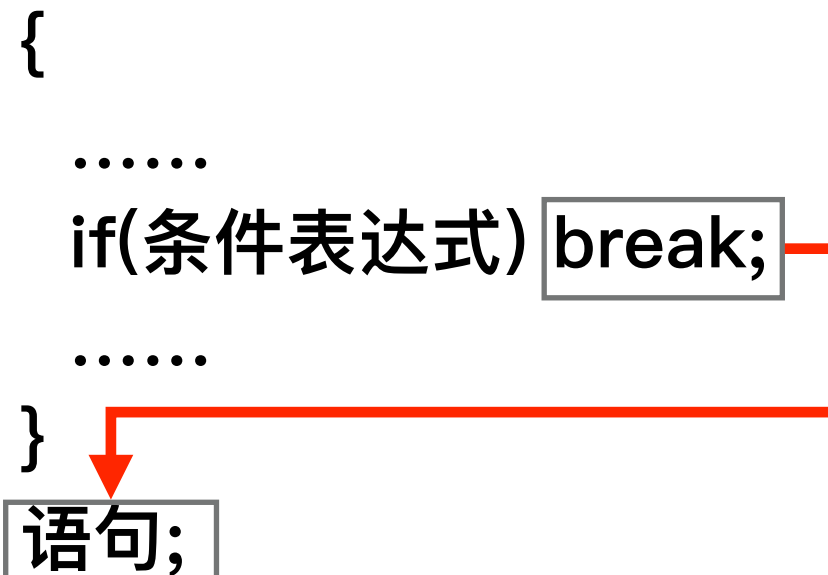
循环控制语句

- 循环控制表达式
 - 1. 计数循环
 - 2. 条件控制的循环
- 循环体
 - 处理
 - 变更循环控制表达式中的循环变量
- 初始化循环控制变量
 - 为循环体的执行做好准备
- 关键是控制条件表达式的值有序按需变化

流程的转移控制

➤ 终止循环体的执行

➤ break



➤ 中断循环体，执行下一次循环

➤ continue

