# 自增、自减运算符

自增、自减运算符用来对一个变量进行加 1 或减 1 运算，其结果仍然赋予该变量。

自增运算符：x++表示在使用 x 之后，使 x 的值加 1；++x 表示在使用 x 之前，使 x 的值加 1。

自减运算符：x--表示在使用 x 之后，使 x 的值减 1；--x 表示在使用 x 之前，使 x 的值减 1。

# 自增运算

分析如下程序实例:

```
#include <iostream>
using namespace std;
int main()
{
    int i = 1;
    int j = 1;
    int a = i++;
    int b = ++j;
    cout << "i= " << i << endl;
    cout << "j= " << j << endl;
    cout << "a= " << a << endl;
    cout << "b= " << b << endl;
    return 0;
}
```

上面的程序与如下程序等价:

```
#include <iostream>
using namespace std;
int main()
{
    int i = 1;
    int j = 1;
    int a = i;
    i = i + 1;
    j = j + 1;
    int b = j;
    cout << "i= " << i << endl;
    cout << "j= " << j << endl;
    cout << "a= " << a << endl;
    cout << "b= " << b << endl;
    return 0;
}
```

所以程序的输出结果是:

```
i = 2
j = 2
a = 1
```

```
b = 2
```

# 变量自加运算

```
1.  #include<iostream>
2.  using namespace std;
3.  int main()
4.  {
5.      int x = 7;
6.      int y = 8;
7.      int z1 = y - (x++);
8.      int z2 = y - (++x);
9.      cout << "z1=" << z1 << endl << "z2=" << z2;
10.     return 0;
11. }
```

上面的程序与下面的程序等价。

```
1.  #include<iostream>
2.  using namespace std;
3.  int main()
4.  {
5.      int x = 7;
6.      int y = 8;
7.      int z1 = y - x; // y=8, x=7
8.      x = x + 1;  // x=8
9.      x = x + 1;  // x=9
10.     int z2 = y - x; // y=8, x=9
11.     cout << "z1=" << z1 << endl << "z2=" << z2;
12.     return 0;
13. }
```

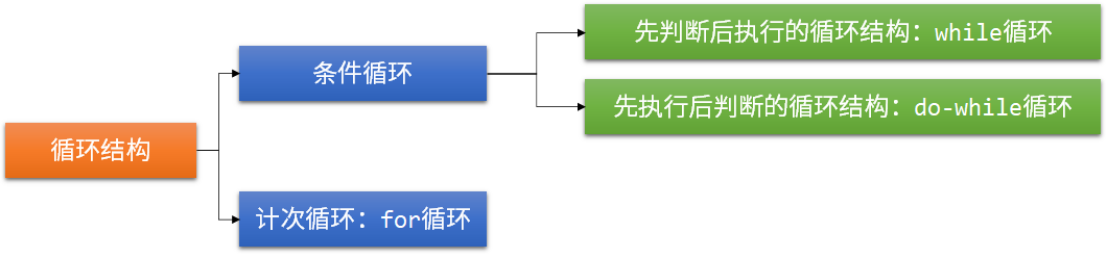
所以最后的输出结果为：

```
z1=1
z2=-1
```

# 循环结构

循环：在程序中是指某些代码会被重复执行。

循环结构：指在程序中需要反复执行某个功能而设置的一种程序结构。



# 第一个 for 循环程序

这是一个 for 循环程序，你能想出它的输出结果吗？

```
1.  #include<iostream>
2.  using namespace std;
3.  int main()
4.  {
5.      int i = 0;
6.      cout << i << " before loop" << endl;
7.      for(i=1; i<=10; i++)
8.      {
9.          cout << i << endl;
10.     }
11.     cout << i << " after loop" << endl;
12.     return 0;
13. }
```

下面就是程序的输出结果，循环前输出的 i 值是 0，10 次循环输出从 1 到 10，循环后输出 11。

```
0 before loop
1
2
3
4
5
6
7
8
9
10
11 after loop
```

下面我们来正式介绍 C++的标准 for 循环结构。

# for 循环语法结构及执行过程

C++标准 for 循环的语法结构如下所示：

```
1. for (循环控制变量初始化表达式; 循环条件表达式; 循环控制变量更新表达式)
2. {
3.     ……
4. }
```

其中循环体部分由多个语句构成，被一对花括号括起来，构成一个语句块的形式。在书写时，循环体的语句要相对于 for 缩进。

当循环体内的语句只有一条时，可以简化为如下形式：

```
1. for (循环控制变量初始化表达式; 循环条件表达式; 循环控制变量更新表达式)
2.     语句;
```

这是一段最简单的 for 循环程序代码：

```
1. for(i=0; i<10; i++)
2. {
3.     cout << i << endl;
4. }
```

可以简化书写为：

```
1. for(i=0; i<10; i++)
2.     cout << i << endl;
```

或：

```
1. for(i = 0; i < 10; i ++) cout << i << endl;
```

把这段代码和 C++标准 for 循环的语法结构对比，可以得到如下对应关系：

| | |
|---|---|
| 循环控制变量初始化表达式 | i=0 |
| 循环条件表达式 | i<10 |
| 循环控制变量更新表达式 | i++ |

这就表示循环从 i=0 开始，i=10 结束，每循环一次，i 自动增加 1。每次循环过程中，向屏幕输出当前 i 的取值。因此在屏幕上会输出 0~9 十行数字。

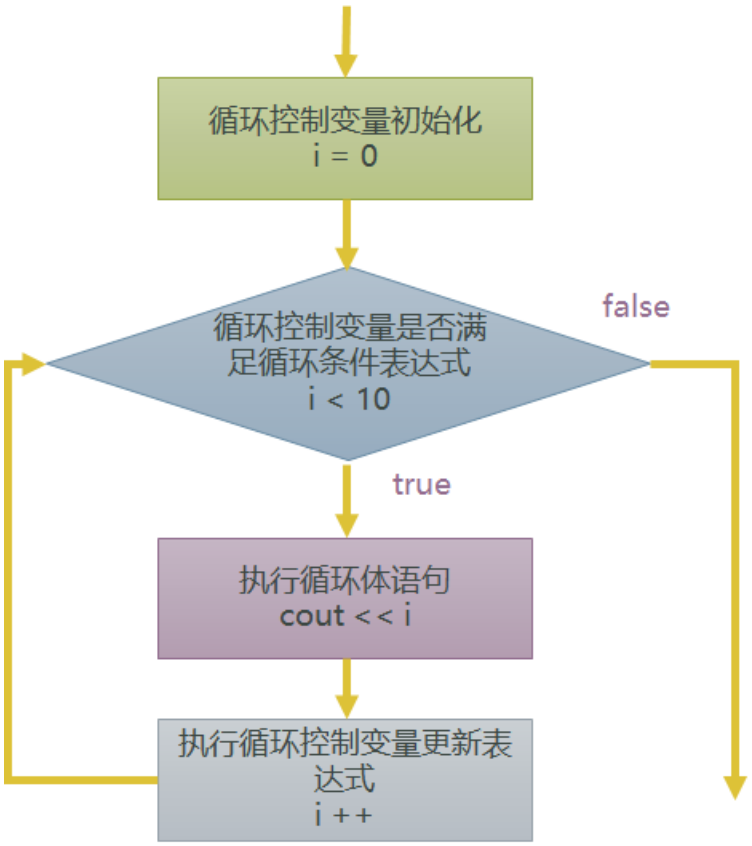



图1. for 循环执行过程

C++的 for 循环结构具体执行过程如下所述：

（1）计算“循环控制变量初始化表达式”，给循环控制变量赋初值。

（2）判断循环控制变量是否满足“循环条件表达式”，若满足条件则执行一遍循环体，否则结束整个 for 语句，继续执行 for 循环后面的语句。

（3）根据“循环控制变量更新表达式”，计算控制变量的新值。

（4）自动转到第二步。

# for 语句举例

让我们来看一些 for 语句的例子：

| | |
|---|---|
| for(i=1; i<=100; i++) | 控制变量从 1 变到 100，增量为 1 |
| for(i=100; i>=1; i--) | 控制变量从 100 变到 1，增量为-1 |
| for(i=5; i<=50; i+=5) | 控制变量从 5 变到 50，增量为 5 |

也可以在 for 循环“控制变量初始化表达式”中声明变量：

| | |
|---|---|
| for(int i=20; i>=2; i-=2) | 控制变量从 20 变到 2，增量为-2 |
| for(int j=99; j>=0; j-=11) | 控制变量值：99、88、77、66、55、44、33、22、11、0，增量为-11 |
| for(int i=1, j=2; i<=99 && j<=100; i+=2, j+=2) | 控制变量 i 和 j 共同进行循环控制，i 从 1 变到 99，j 从 2 变到 100，增量均为 2 |

在 for 循环“循环控制变量初始化表达式”中声明的变量只在 for 循环结构中有效，离开了该 for 结构，变量就无效了。

这个程序中的 i 是在循环语句以外定义的，因此循环结束后，仍然可以输出 i 的值。

```
int i;
for(i = 0; i < 10; i ++)
{
    cout << i << endl;
}
// 输出 i 是 10
cout << i << endl;
```

这个程序中的 i 是在循环语句内定义的，因此在循环语句外部是不可见的。

```
for(int i = 0; i < 10; i ++)
{
    cout << i << endl;
}
// 这里编译不通过
cout << i << endl;
```