

HIGH-LEVEL PROGRAMMING I

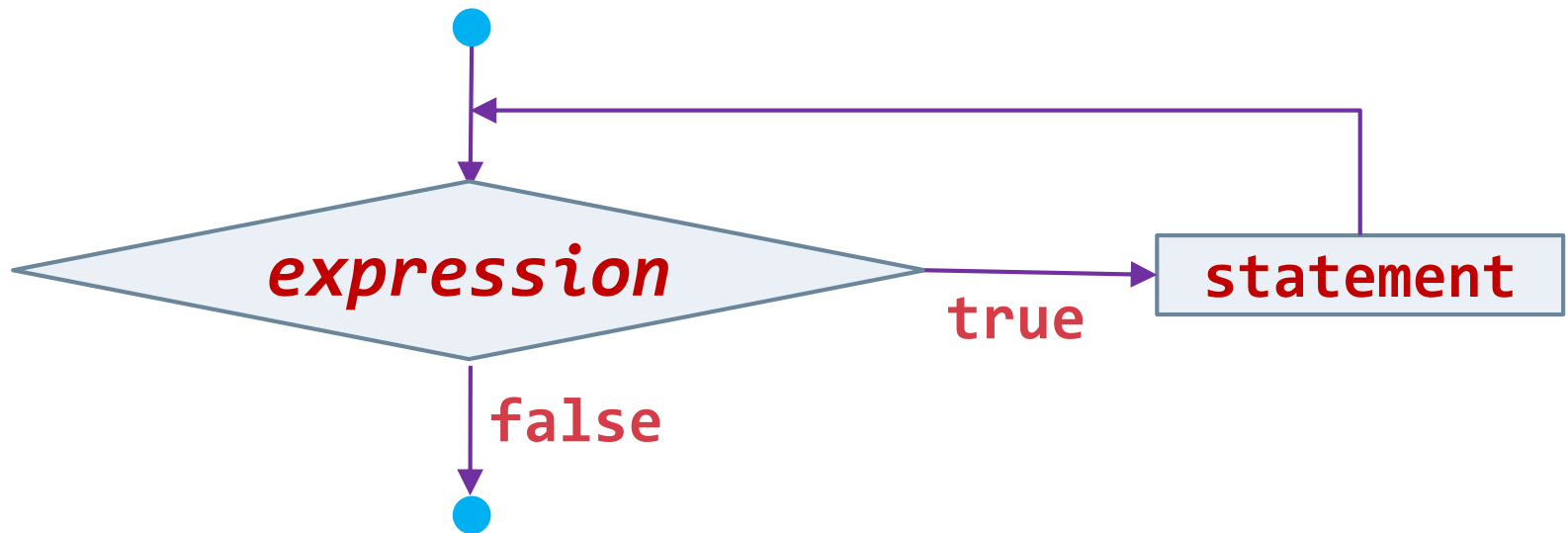
for Looping Structure

by Prasanna Ghali

while Loop Structure

2

- Repeat actions *while* a condition remains true

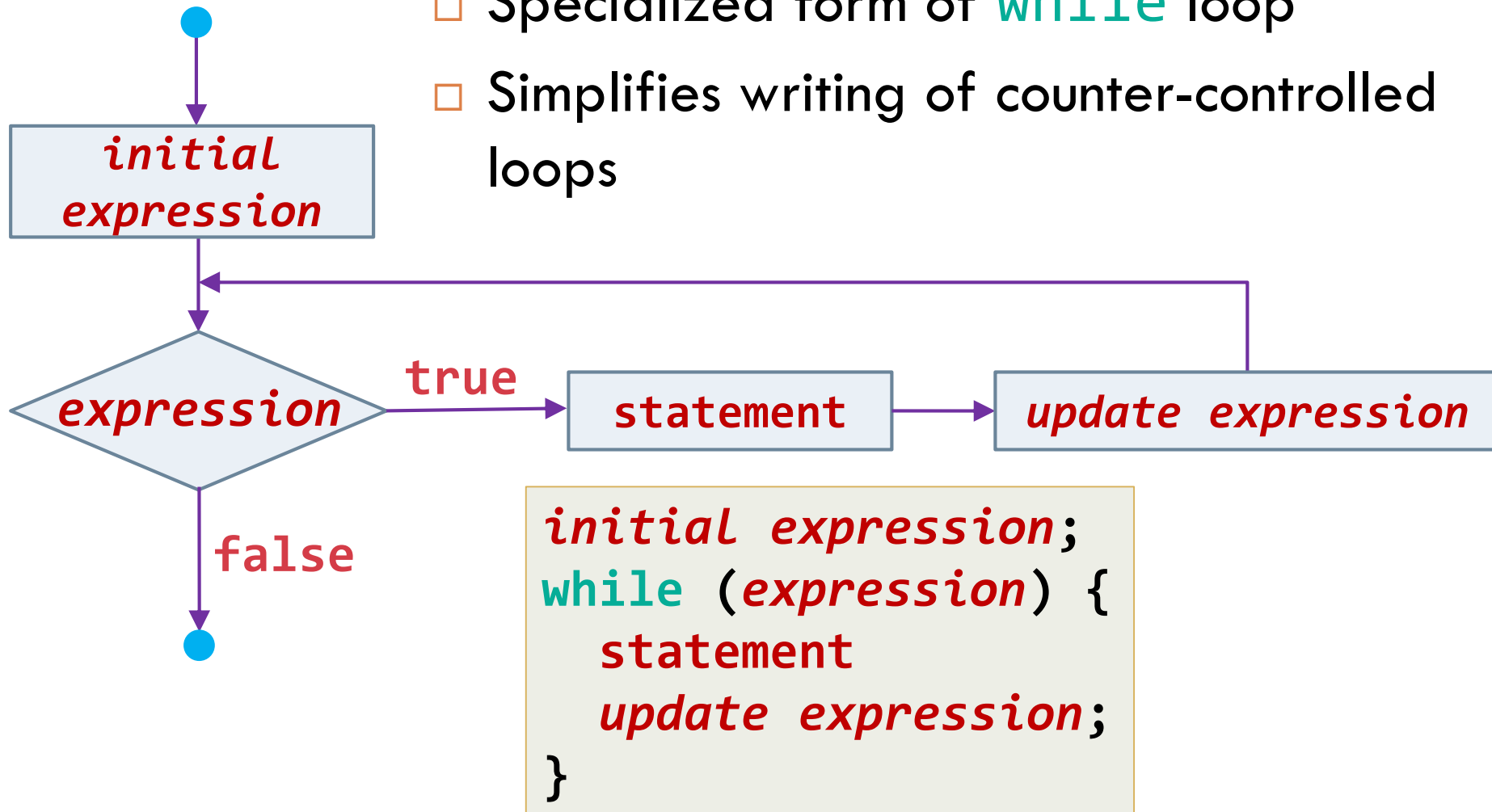


```
while (expression)  
    statement
```

for Loop Structure (1 / 4)

3

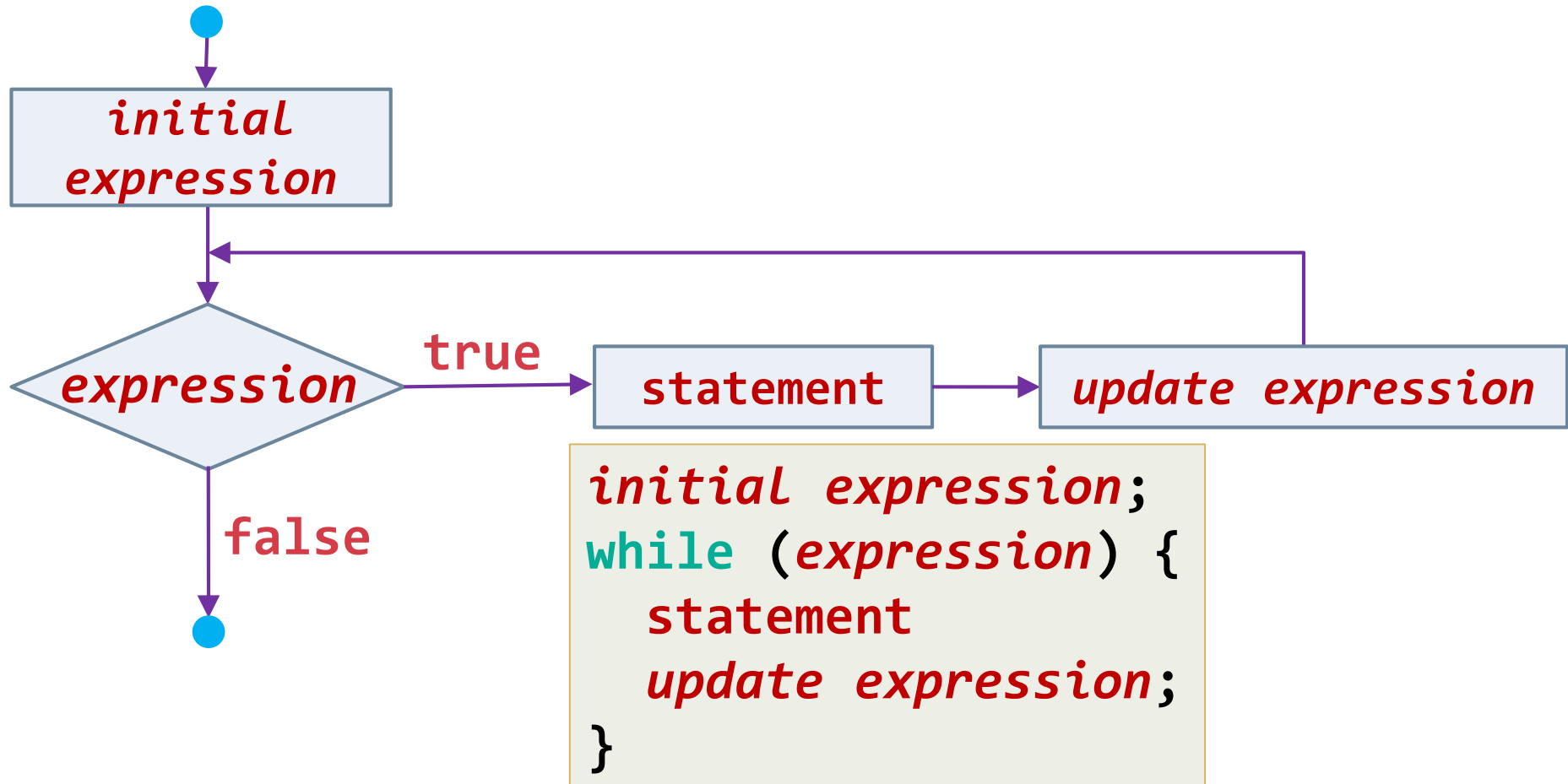
- Specialized form of **while** loop
- Simplifies writing of counter-controlled loops



for Loop Structure (2/4)

4

```
for (initial expression; expression; update expression)  
statement
```



for Loop Structure (3/4)

5

evaluate *initial expression* once and only once

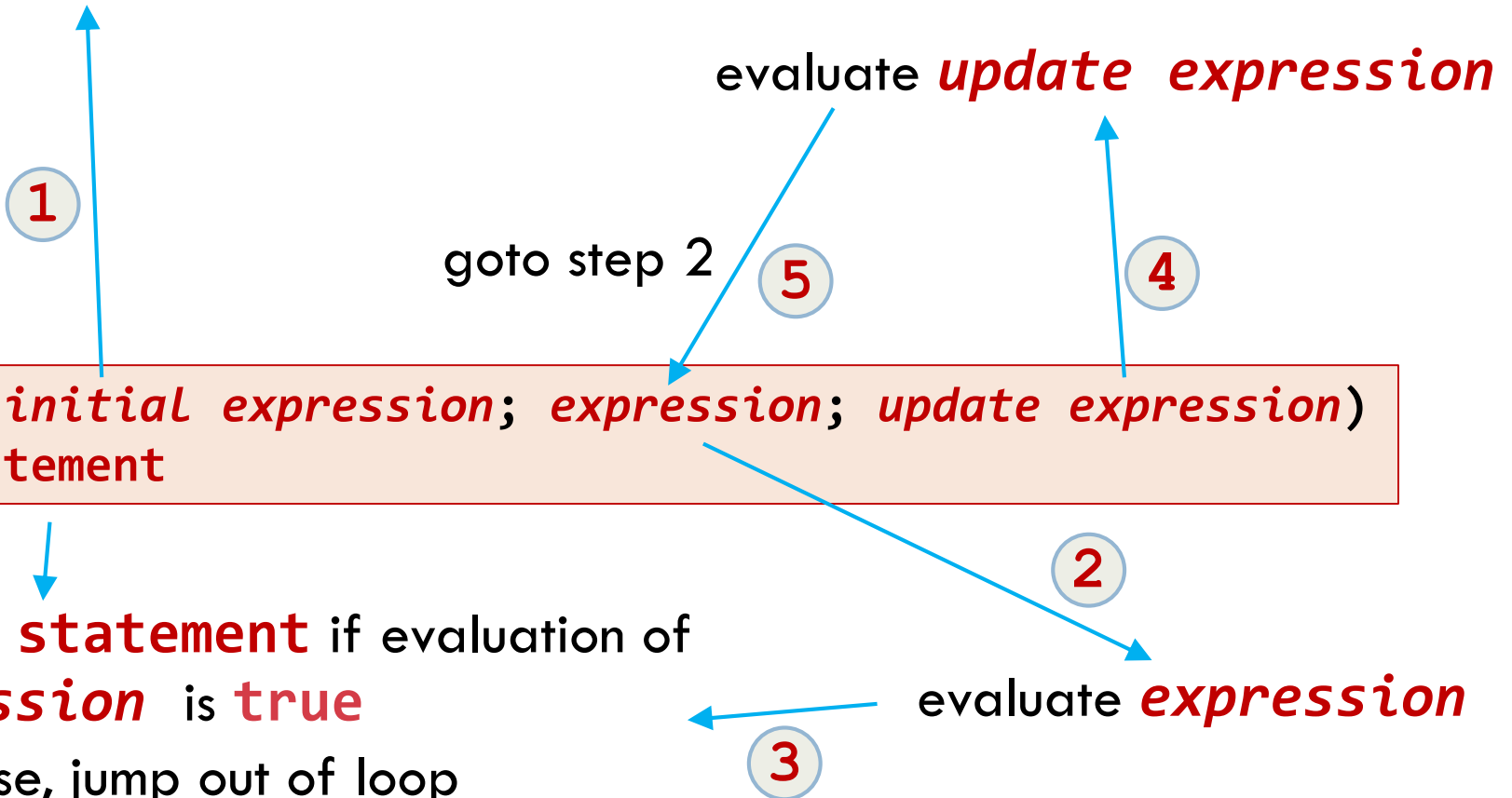
evaluate *update expression*

goto step 2

```
for (initial expression; expression; update expression)  
    statement
```

execute *statement* if evaluation of
expression is *true*
otherwise, jump out of loop

evaluate *expression*



for Loop Structure (4/4)

6

```
int i = 0;
while (i <= 20) {
    printf("%d ", i);
    i += 5;
}
```

0 5 10 15 20

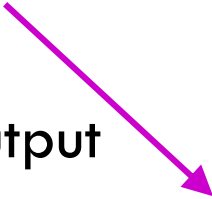
```
for (int i = 0; i <= 20; i += 5) {
    printf("%d ", i);
}
```

for Loop: Example 1

7

```
for (int i = 0; i <= 7; i += 1) {  
    printf("%d squared is %2d\n", i, i*i);  
}
```

program output



**0 squared is 0
1 squared is 1
2 squared is 4
3 squared is 9
4 squared is 16
5 squared is 25
6 squared is 36
7 squared is 49**

Example 2: Find Sum of 1st N Numbers

8

```
printf("Enter number of +ve integers to be added: ");
int N;
scanf("%d", &N);
int sum = 0;
for (int counter = 0; counter < N; counter += 1) {
    sum += counter;
}
printf("Sum of 1st %d +ve integers is %d\n", N, sum);
```


Missing Expressions (1 / 2)

9

- Any or all of expressions in **for** statement can be omitted

```
int i = 1;
for (; i <= 5;) {
    printf("%d ", i);
    i += 1;
}
```

- Equivalent to:

```
int i = 1;
while (i <= 5) {
    printf("%d ", i);
    i += 1;
}
```

Missing Expressions (2/2)

10

- This is an infinite loop

```
for (; ;) {  
    printf("This is infinite loop ... \n");  
}
```

- Default value of "no expression" in *update expression* in **for** statement is **true**