Chapter 4: Control structures

Repetition

Loop Statements

After reading and studying this Section, student should be able to

- Implement repetition control in a program using while statements.
- Implement repetition control in a program using do-while statements.
- Implement a generic loop-and-a-half repetition control statement
- Implement repetition control in a program using for statements.
- Nest a loop repetition statement inside another repetition statement.
- Choose the appropriate repetition control statement for a given task

Definition

- Repetition statements control a block of code to be executed for a fixed number of times or until a certain condition is met.
- There are three types of repetition:
  - Count-controlled repetitions terminate the execution of the block after it is executed for a fixed number of times.
  - Sentinel-controlled repetitions terminate the execution of the block after one of the designated values called a *sentinel* is encountered.
  - Flag-controlled repetitions terminate the execution of the block after one of the designated values called a *sentinel* is encountered.
- Repetition statements are called loop statements also.

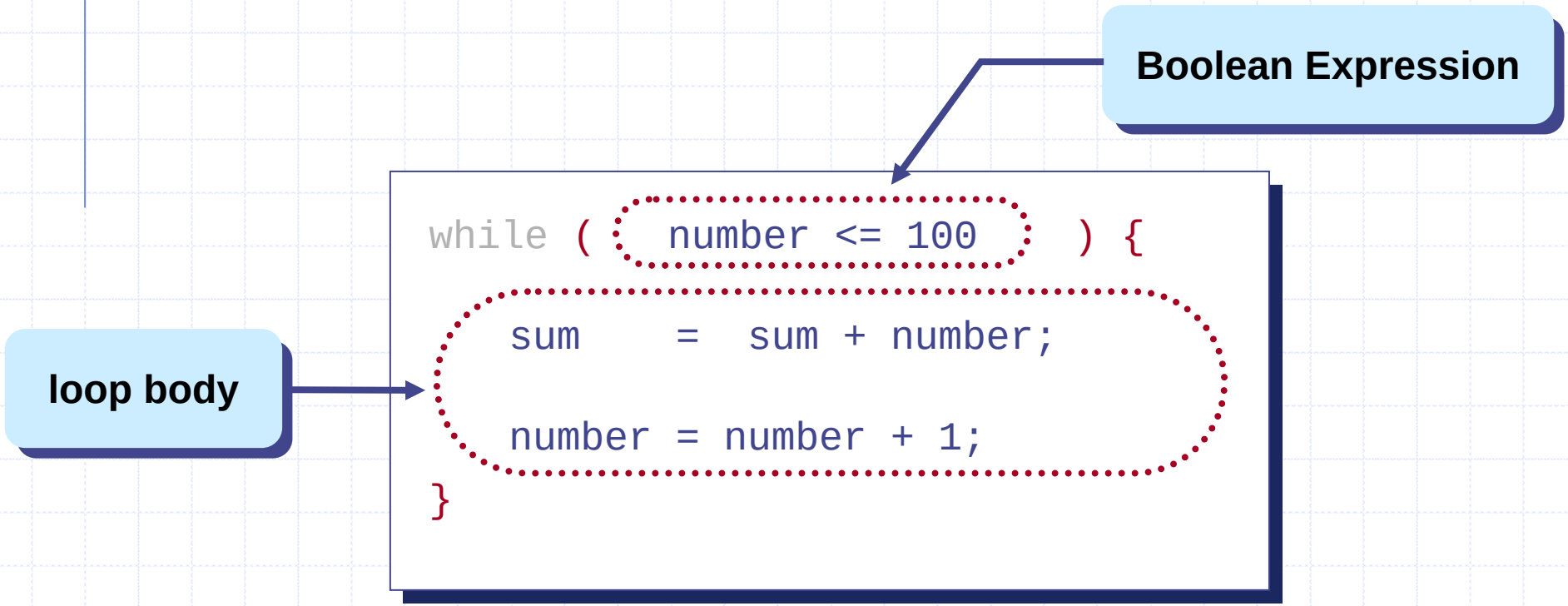
The while Statement

```
int sum = 0, number = 1;

while ( number <= 100 ) {
    sum    =  sum + number;

    number = number + 1;
}
```

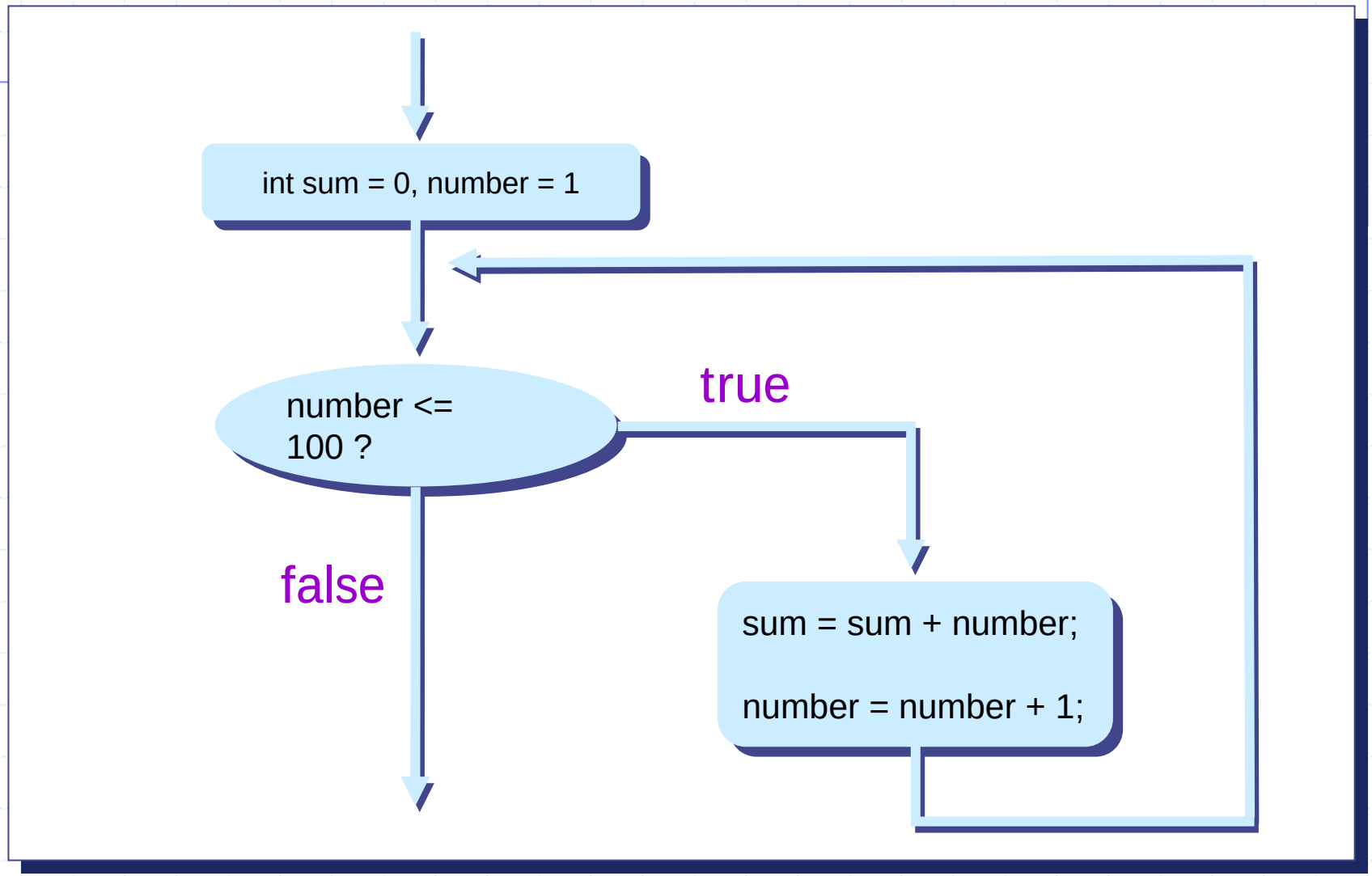
These statements are executed as long as number is less than or equal to 100.

Syntax for the while Statement

```
while ( <boolean expression> )

    <loop body>
```



Control Flow of while



More Examples

1

```
int sum = 0, number = 1;

while ( sum <= 1000000 ) {
     sum    =  sum + number;
     number = number + 1;
}
```

Keeps adding the numbers 1, 2, 3, … until the sum becomes larger than 1,000,000.

2

```
int product =  1, number = 1,
    count   = 20, lastNumber;


lastNumber = 2 * count - 1;


while (number <= lastNumber) {
     product = product * number;
     number  = number + 2;
}
```

Computes the product of the first 20 odd integers.

Loop Logical Errors

- Goal: Execute the loop body 10 times.

1
```
count = 1;
while ( count < 10 ){
    . . .
    count++;
}
```


2
```
count = 1;
while ( count <= 10 ){
    . . .
    count++;
}
```


3
```
count = 0;
while ( count <= 10 ){
    . . .
    count++;
}
```


4
```
count = 0;
while ( count < 10 ){
    . . .
    count++;
}
```


1 and 3 exhibit off-by-one error.

The do-while Statement

```
int sum = 0, number = 1;

do {
   sum += number;
   number++;
} while ( sum <= 1000000 );
```

These statements are executed as long as sum is less than or equal to 1,000,000.

Syntax for the do-while Statement

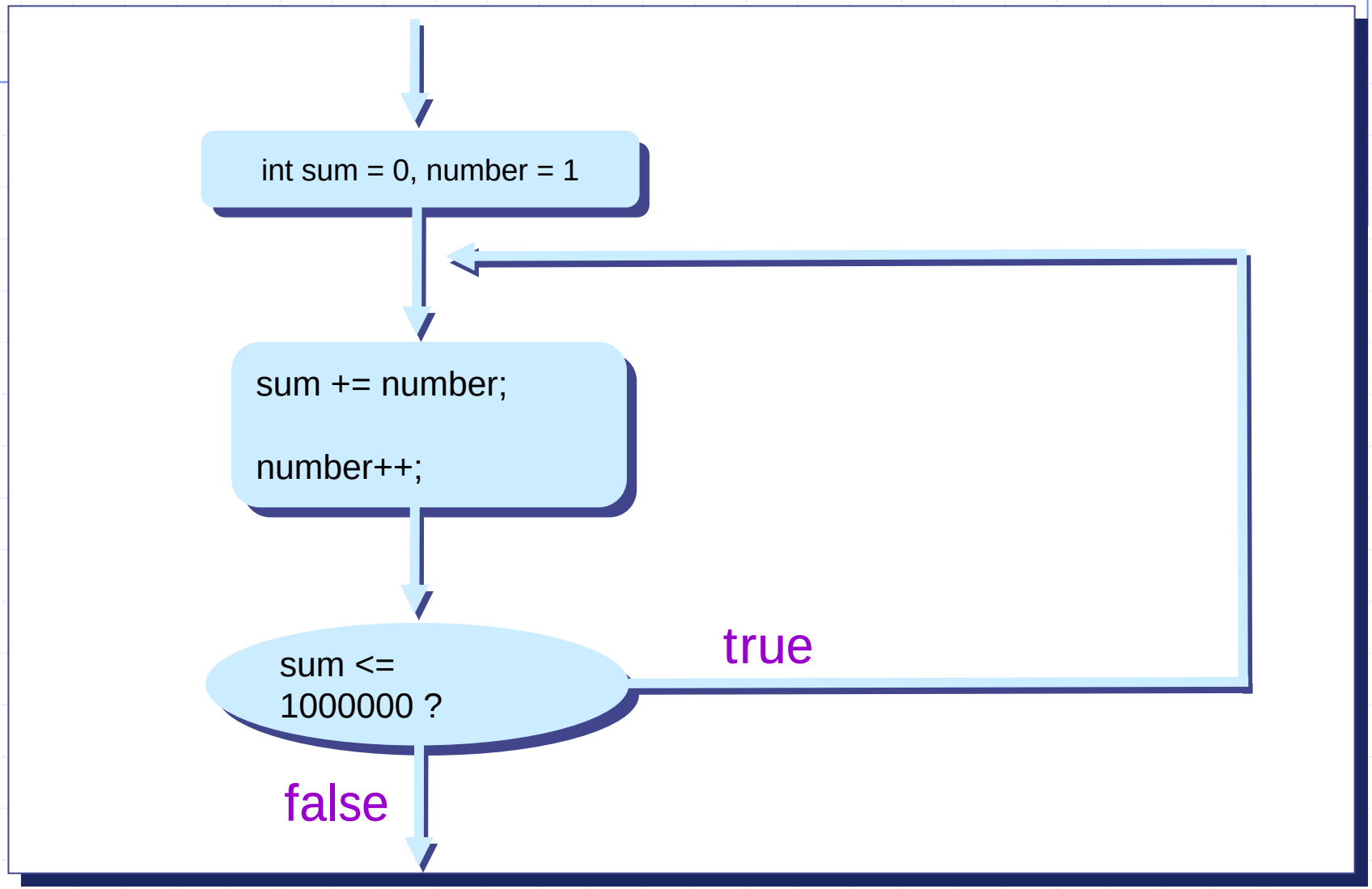
```
do
        <loop body>

while ( <boolean expression> ) ;
```

```
do   {
     sum += number;
     number++;
}    while (    sum <= 1000000    );
```

loop body

Boolean Expression

Control Flow of do-while



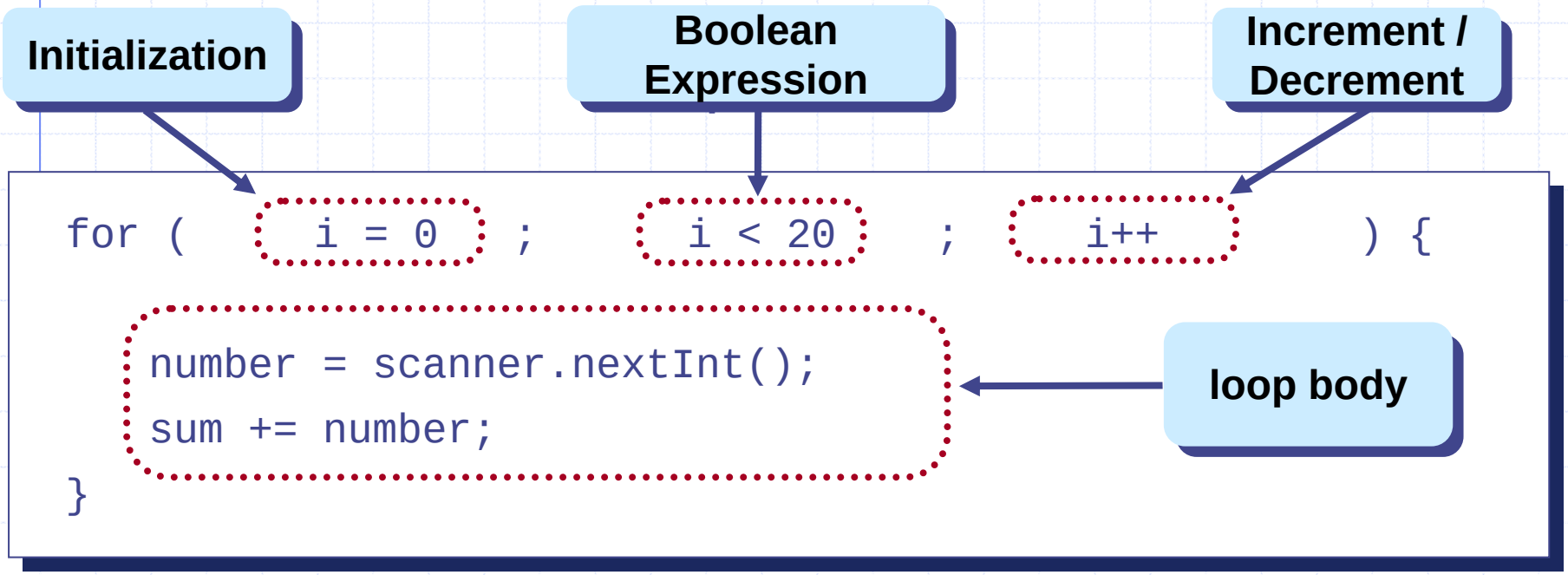
The for Statement

```
int i, sum = 0, number;

for (i = 0; i < 20; i++) {
    number = scanner.nextInt( );
    sum += number;
}
```

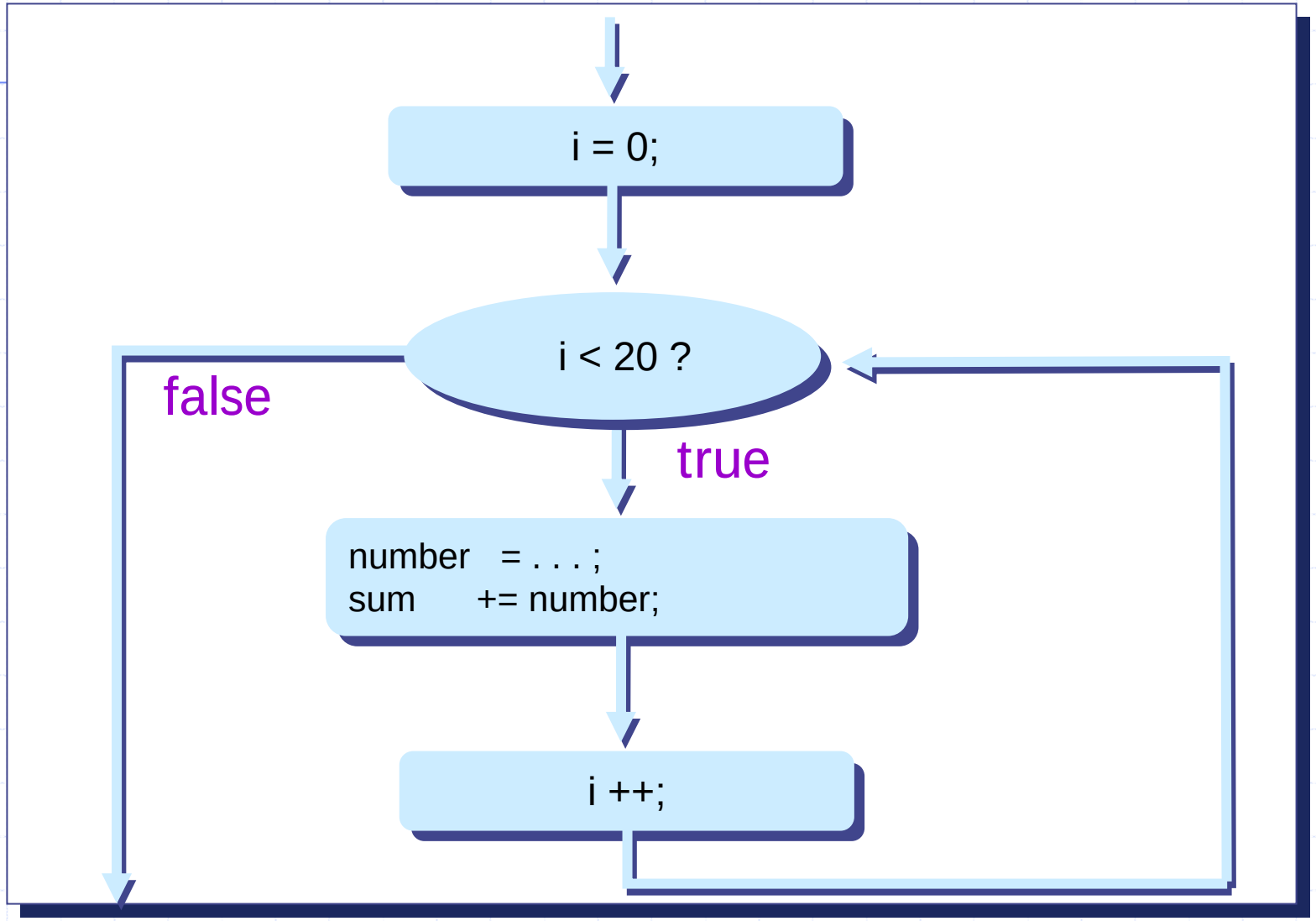
These statements are executed for 20 times (i = 0, 1, 2, … , 19).

Syntax for the for Statement

```
for ( <initialization>; <boolean expression>; <increment>  )

        <loop body>
```



Control Flow of for



More for Loop Examples

1

```
for (int i = 0; i < 100; i += 5)
```

i = 0, 5, 10, ... , 95

2

```
for (int j = 2; j < 40; j *= 2)
```

j = 2, 4, 8, 16, 32

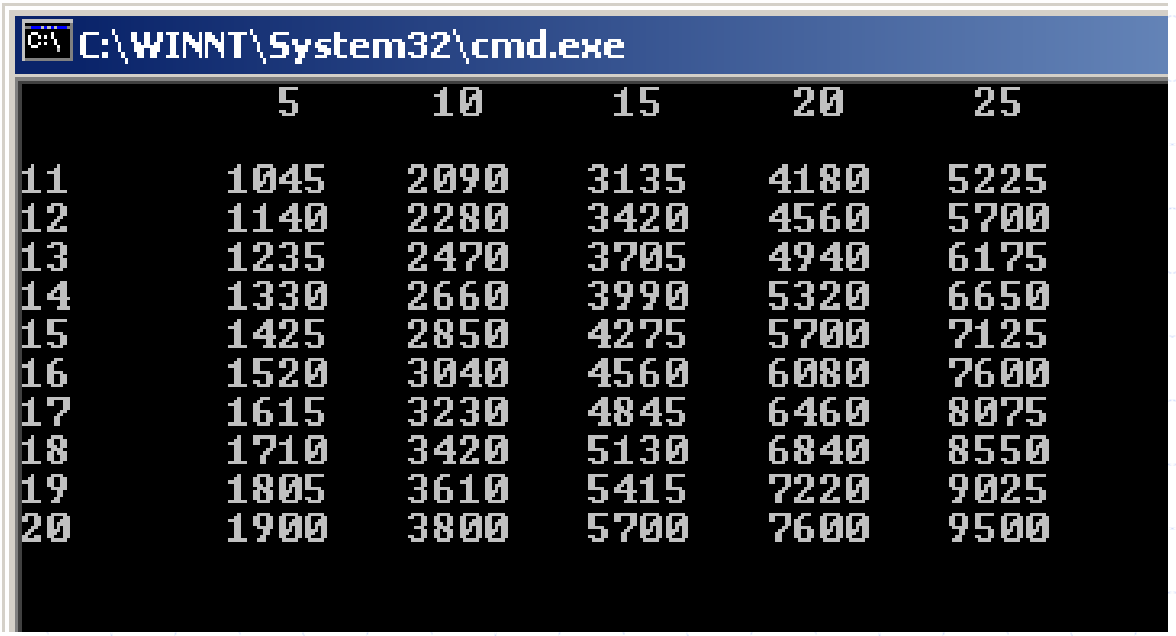
3

```
for (int k = 100; k > 0; k--) )
```

k = 100, 99, 98, 97, ..., 1

The Nested-for Statement

- Nesting a for statement inside another for statement is commonly used technique in programming.
- Let's generate the following table using nested-for statement.



Generating the Table

```
int price;
for (int width = 11; width <=20, width++){

    for (int length = 5, length <=25, length+=5){

      price = width * length * 19; //$19 per sq. ft.
      System.out.print ("  " + price);
    }
    //finished one row; move on to next row
    System.out.println("");
}
```

OUTER

INNER