

# 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> )
```

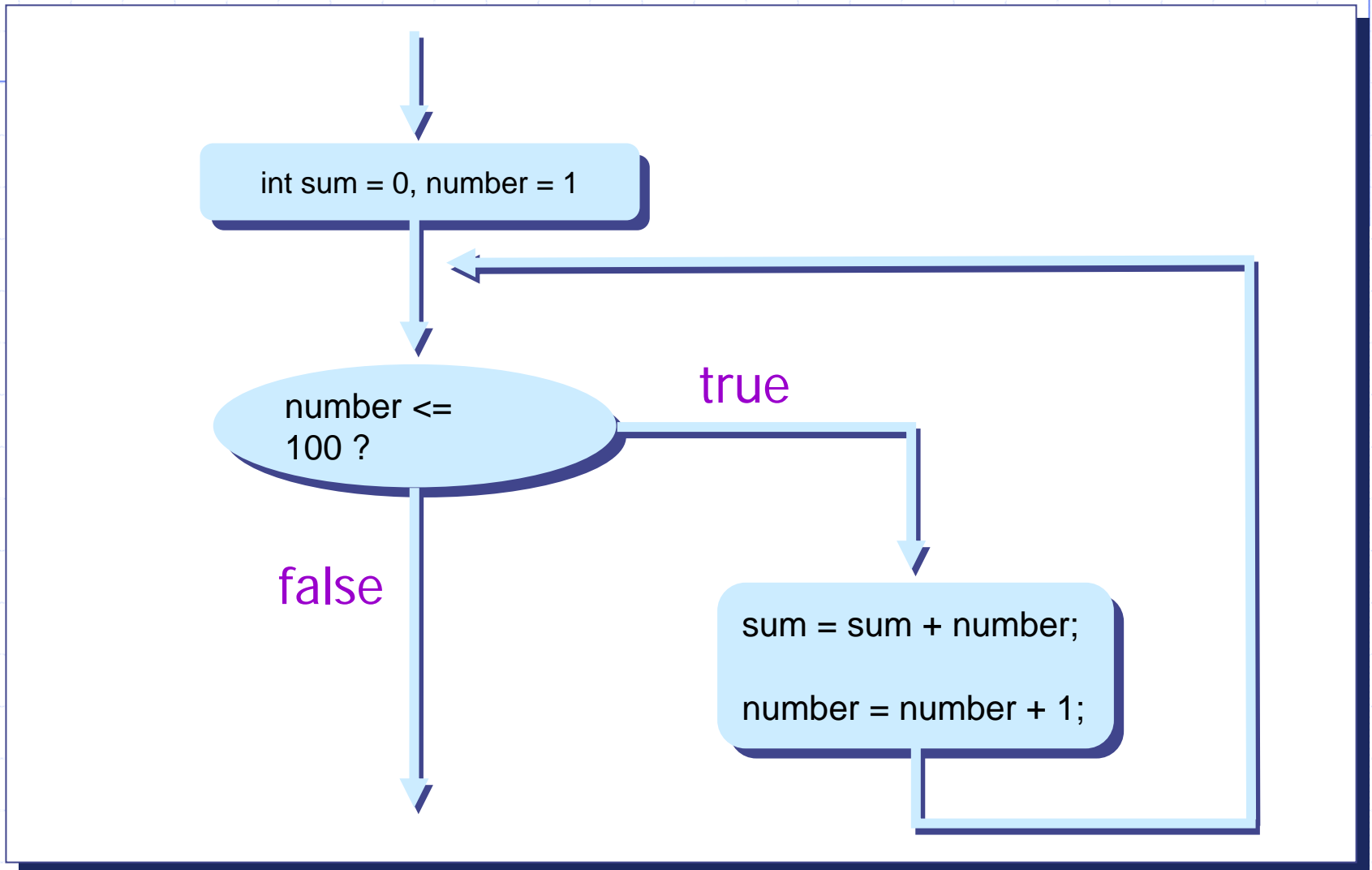
```
<loop body>
```

**Boolean Expression**

```
while ( number <= 100 ) {  
    sum = sum + number;  
    number = number + 1;  
}
```

**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>
```

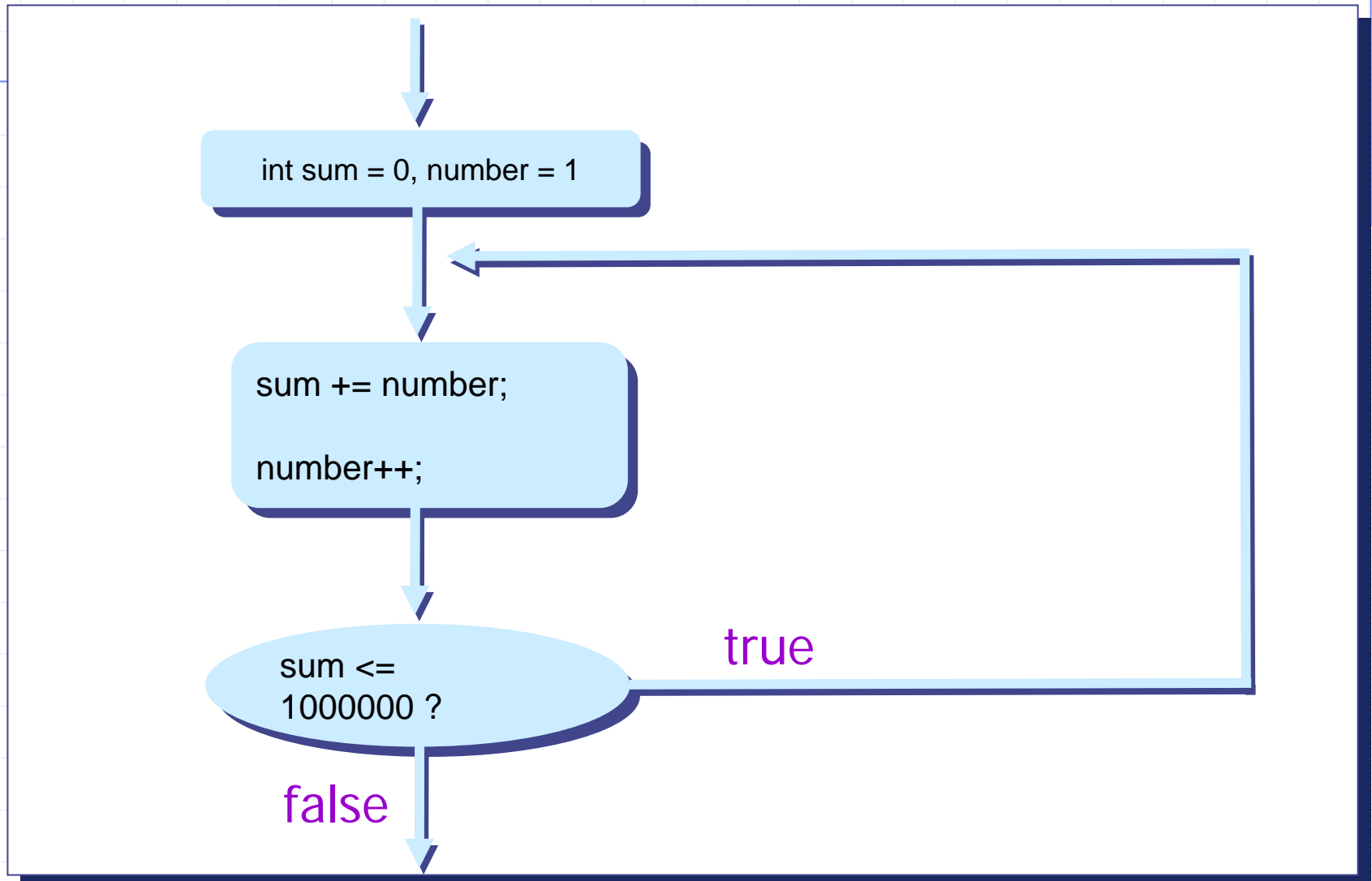
```
while ( <boolean expression> ) ;
```

```
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>
```

**Initialization**

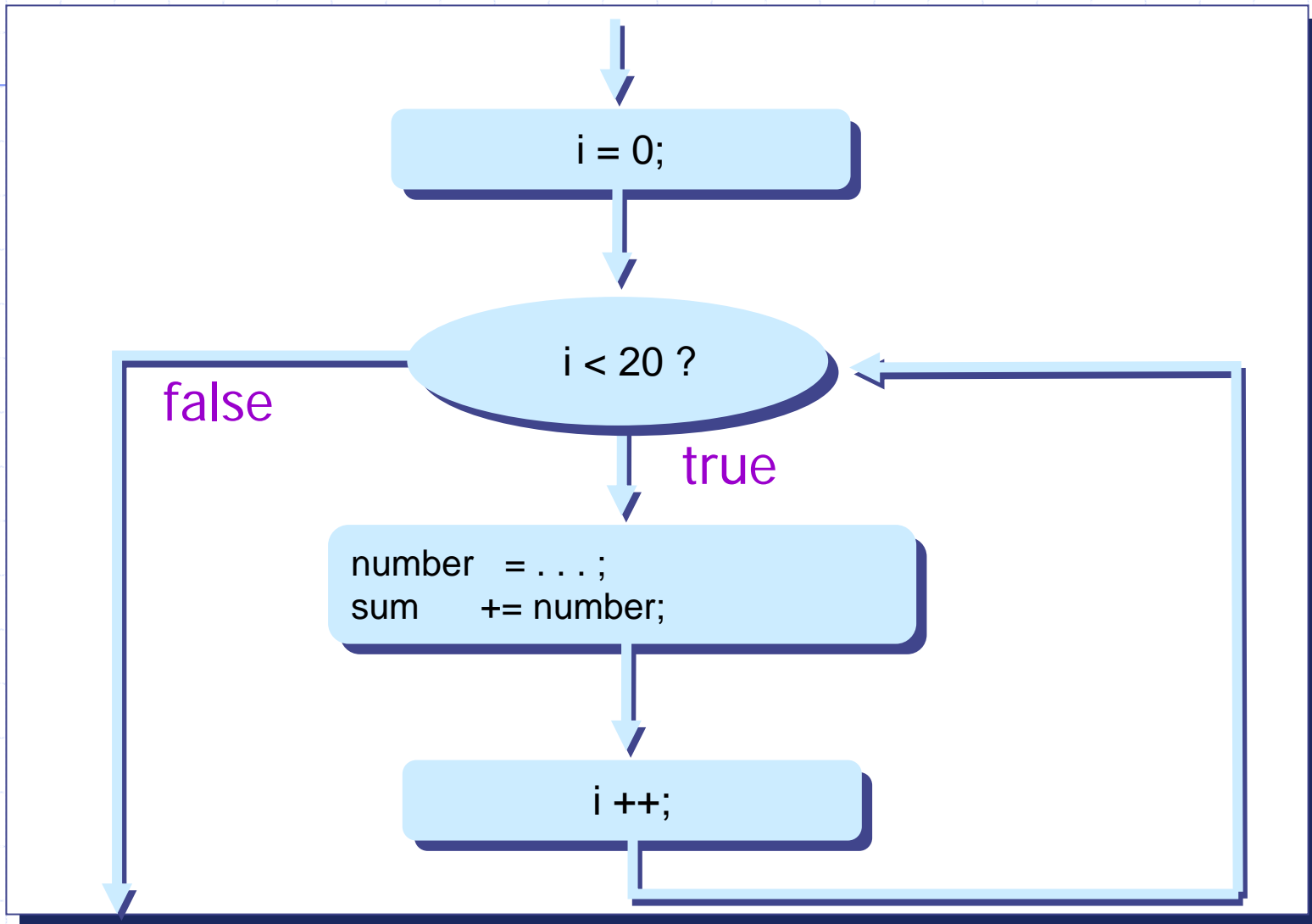
**Boolean Expression**

**Increment /  
Decrement**

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

**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`

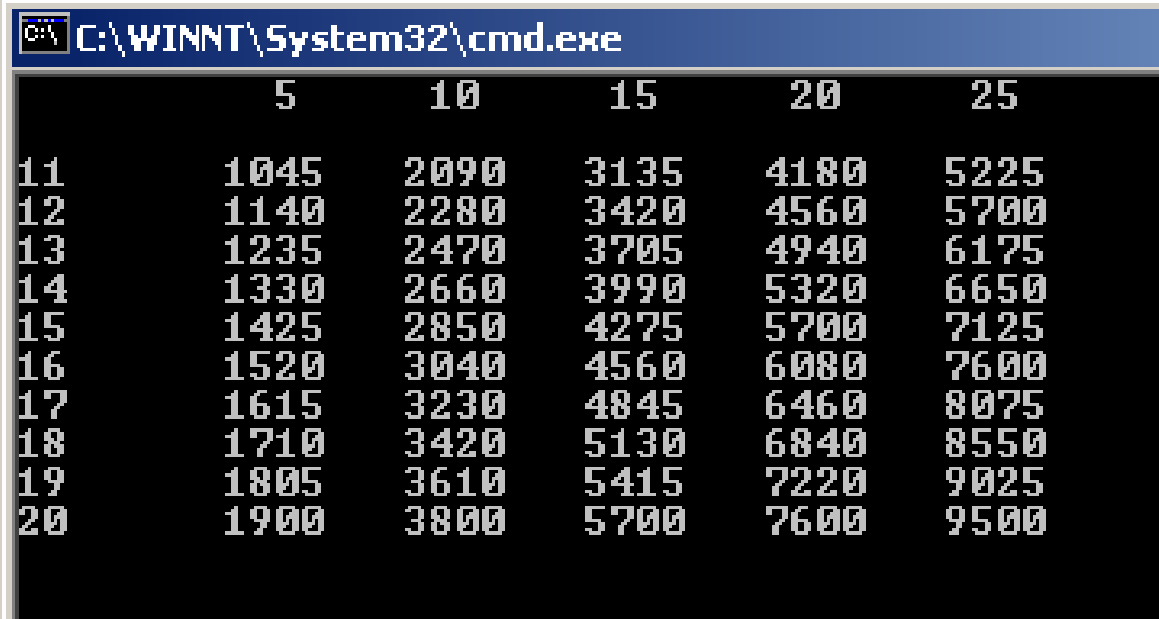
3

```
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.

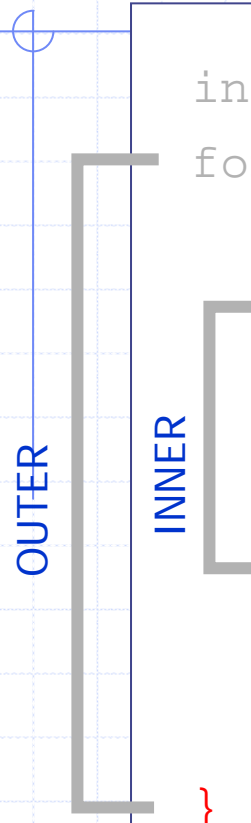


A screenshot of a Windows command prompt window titled "C:\WINNT\System32\cmd.exe". The window displays a table of numbers. The first row contains the numbers 5, 10, 15, 20, and 25. The subsequent rows contain numbers from 11 to 20, with each row having five columns of numbers. The numbers in each row increase by 10 from left to right, and by 10 from one row to the next.

	5	10	15	20	25
11	1045	2090	3135	4180	5225
12	1140	2280	3420	4560	5700
13	1235	2470	3705	4940	6175
14	1330	2660	3990	5320	6650
15	1425	2850	4275	5700	7125
16	1520	3040	4560	6080	7600
17	1615	3230	4845	6460	8075
18	1710	3420	5130	6840	8550
19	1805	3610	5415	7220	9025
20	1900	3800	5700	7600	9500



# 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("");

}
```