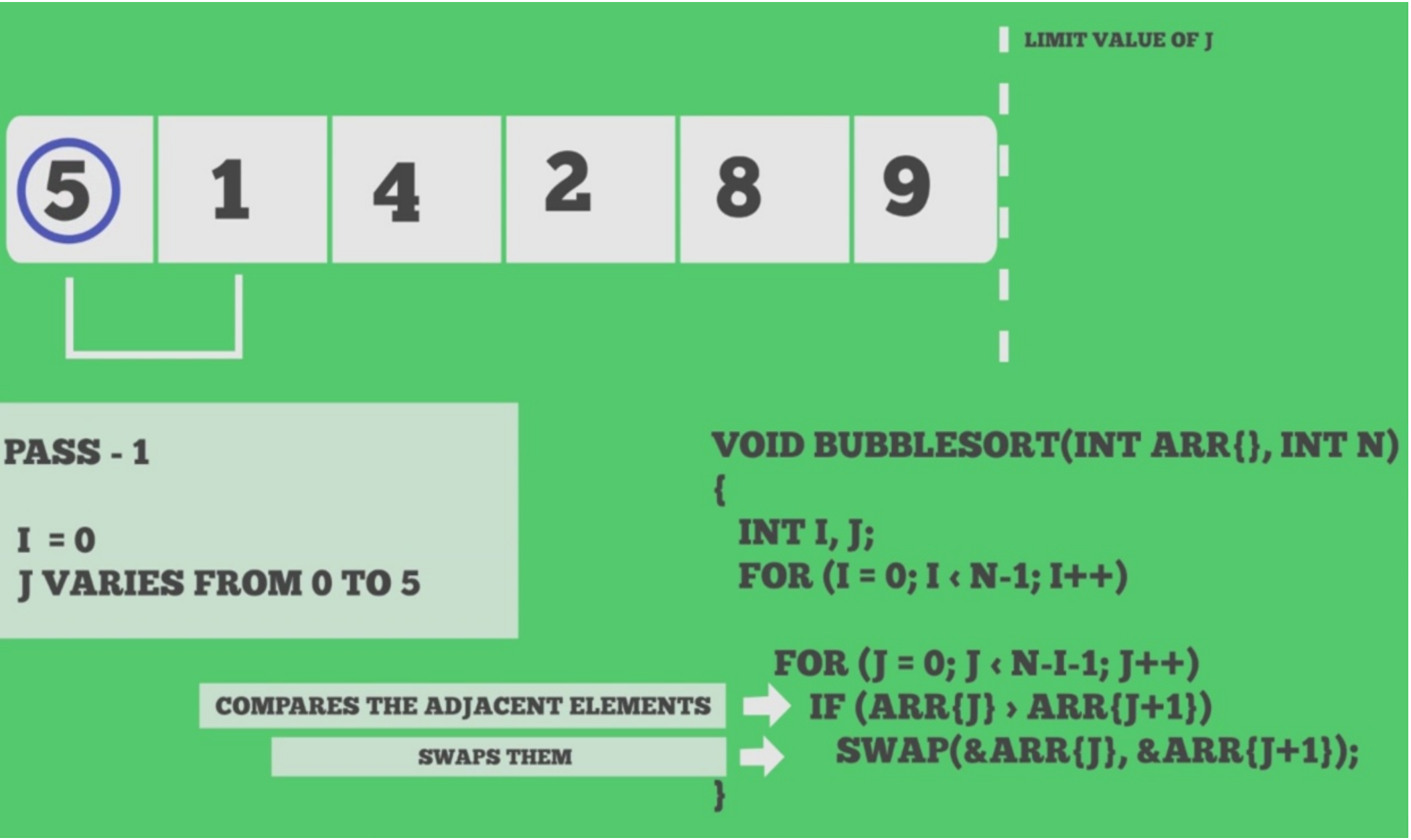
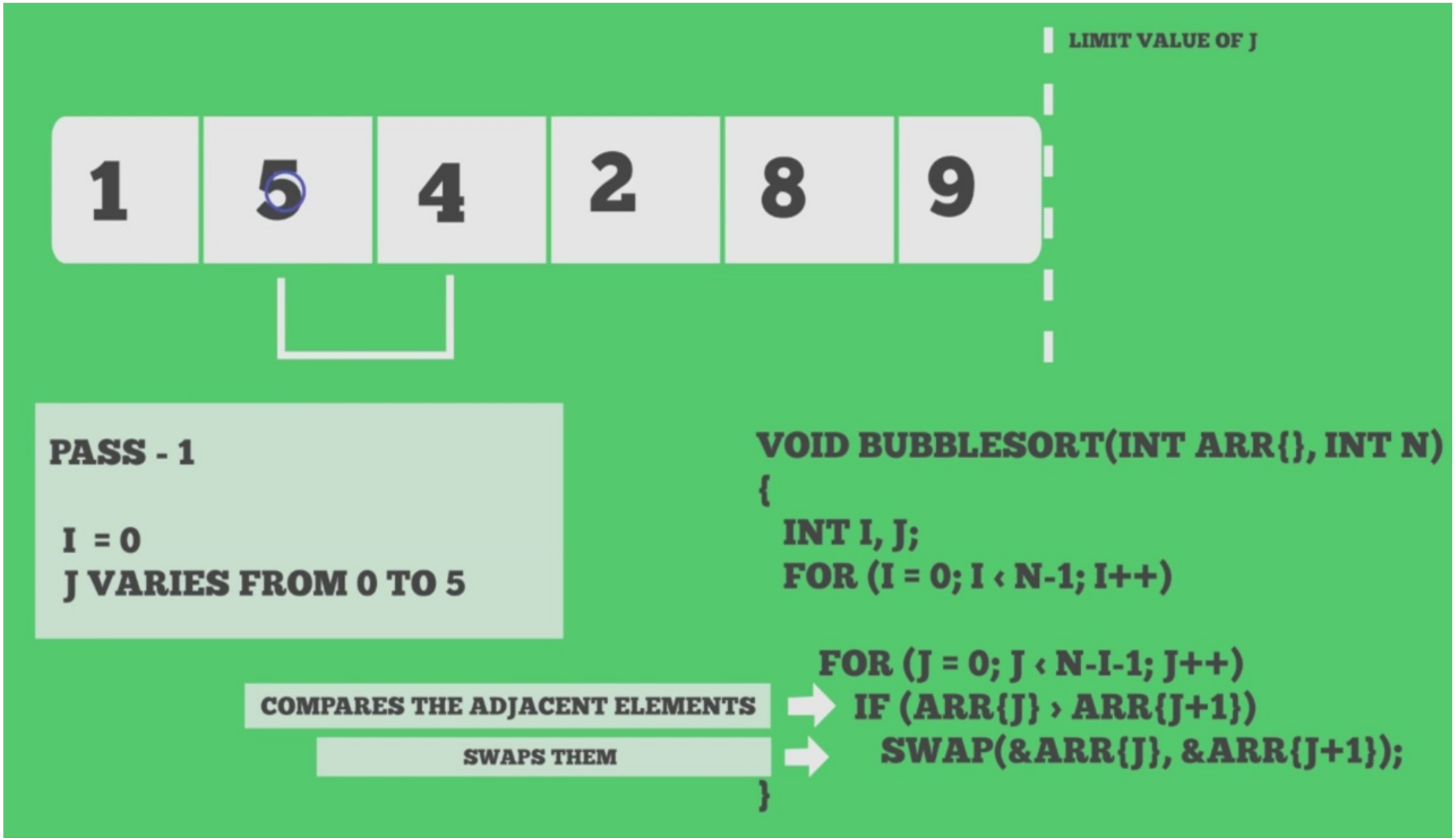
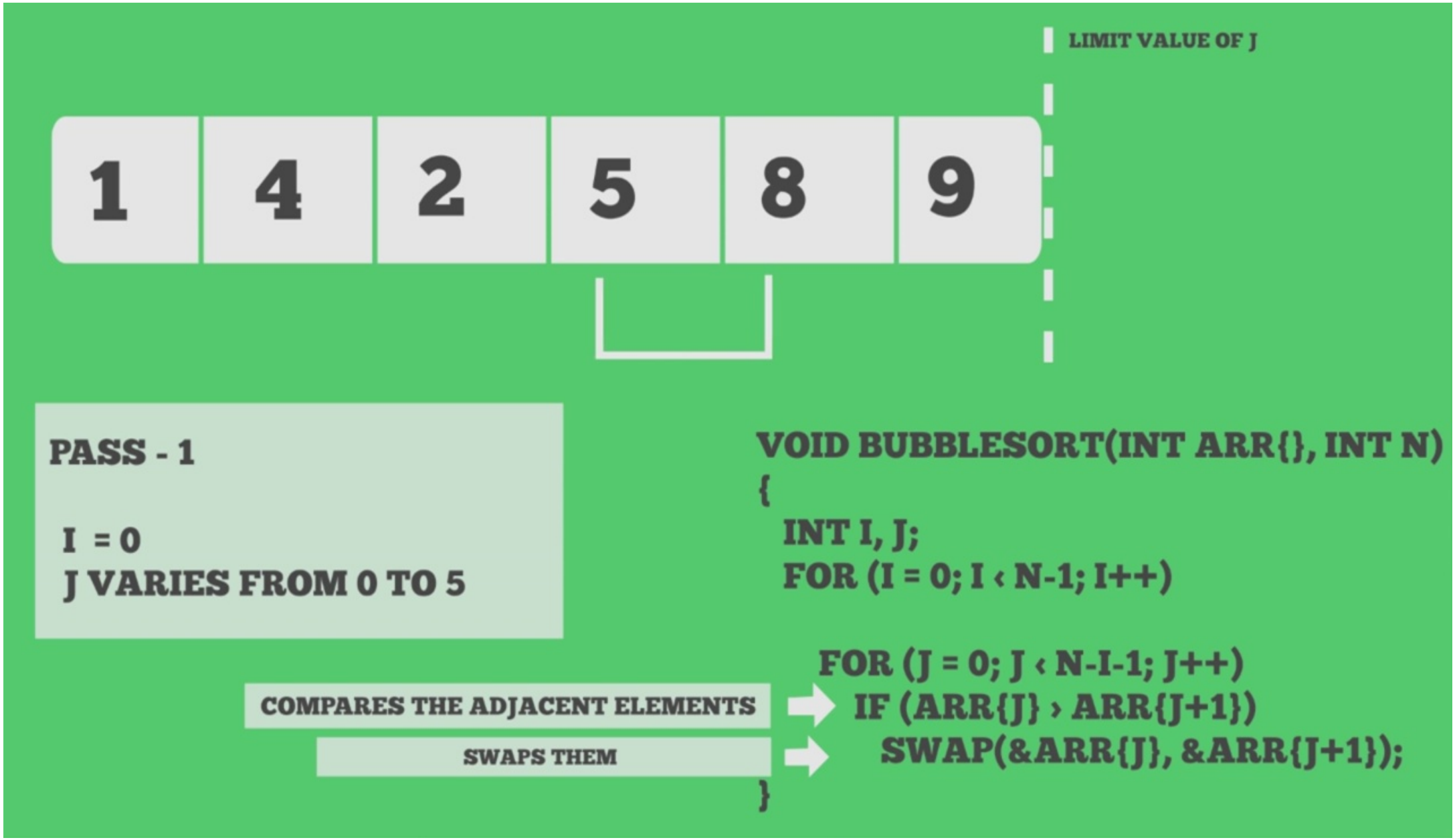

LIMIT VALUE OF J
5
1
4
2
8
9
PASS - 1
I = 0
J VARIES FROM 0 TO 5
VOID BUBBLESORT(INT ARR{}, INT N)
{
INT I, J;
FOR (I = 0; I ‹ N-1; I++)
FOR (J = 0; J ‹ N-I-1; J++)
COMPARES THE ADJACENT ELEMENTS
IF (ARR{J} › ARR{J+1})
SWAPS THEM
SWAP(&ARR{J}, &ARR{J+1});
}


LIMIT VALUE OF J
1
5
4
2
8
9
PASS - 1
I = 0
J VARIES FROM 0 TO 5
VOID BUBBLESORT(INT ARR{}, INT N)
{
INT I, J;
FOR (I = 0; I ‹ N-1; I++)
FOR (J = 0; J ‹ N-I-1; J++)
COMPARES THE ADJACENT ELEMENTS
IF (ARR{J} › ARR{J+1})
SWAPS THEM
SWAP(&ARR{J}, &ARR{J+1});
}


LIMIT VALUE OF J
1
4
2
5
8
9
PASS - 1
I = 0
J VARIES FROM 0 TO 5
VOID BUBBLESORT(INT ARR{}, INT N)
{
INT I, J;
FOR (I = 0; I ‹ N-1; I++)
FOR (J = 0; J ‹ N-I-1; J++)
COMPARES THE ADJACENT ELEMENTS
IF (ARR{J} › ARR{J+1})
SWAPS THEM
SWAP(&ARR{J}, &ARR{J+1});
}

PASS - 2

I = 1
J VARIES FROM 0 TO 4

COMPARES THE ADJACENT ELEMENTS

SWAPS THEM

```
VOID BUBBLESORT(INT ARR{}, INT N)
{
  INT I, J;
  FOR (I = 0; I < N-1; I++)

    FOR (J = 0; J < N-I-1; J++)
      IF (ARR{J} > ARR{J+1})
        SWAP(&ARR{J}, &ARR{J+1});
}
```

PASS - 3

I = 2
J VARIES FROM 0 TO 3

```
VOID BUBBLESORT(INT ARR{}, INT N)
{
  INT I, J;
  FOR (I = 0; I < N-1; I++)

    FOR (J = 0; J < N-I-1; J++)
      IF (ARR{J} > ARR{J+1})
        SWAP(&ARR{J}, &ARR{J+1});
}
```

COMPARES THE ADJACENT ELEMENTS

SWAPS THEM

PASS - 5

I = 4

J VARIES FROM 0 TO 1

COMPARES THE ADJACENT ELEMENTS

SWAPS THEM

```
VOID BUBBLESORT(INT ARR[], INT N)
{
  INT I, J;
  FOR (I = 0; I < N-1; I++)

    FOR (J = 0; J < N-I-1; J++)
      IF (ARR[J] > ARR[J+1])
        SWAP(&ARR[J], &ARR[J+1]);
}
```