Let's go through another detailed example where we add two large numbers.

Example Numbers:

markdown Copy Edit

```
  92345
+  6789
---------
```

We'll represent these numbers as `largeIntegers` objects and store them **right-aligned** in a 100-character array.

Step 1: Right-Aligned Storage

Since `largeIntegers` stores numbers in a **fixed-size array of 100 characters**, they are **right-aligned**, meaning they are padded with `'0'` on the left.

Array Representation

Index	Number 1 (`92345`)	Number 2 (`6789`)
95	'9'	'0'
96	'2'	'6'
97	'3'	'7'
98	'4'	'8'
99	'5'	'9'

Step 2: Digit-Wise Addition

We add the numbers digit by digit from **right to left**, considering `carry`.

Index	`Array[i]` (92345)	`obj.Array[i]` (6789)	`carry`	`sum` = (digit1 + digit2 + carry)	`sum % 10` (Stored)	New `carry`
99	'5' (5)	'9' (9)	0	5 + 9 + 0 = **14**	4	1
98	'4' (4)	'8' (8)	1	4 + 8 + 1 = **13**	3	1
97	'3' (3)	'7' (7)	1	3 + 7 + 1 = **11**	1	1
96	'2' (2)	'6' (6)	1	2 + 6 + 1 = **9**	9	0
95	'9' (9)	'0' (0)	0	9 + 0 + 0 = **9**	9	0

Step 2: Iteration (Filling `Array` and `Array_Reversed`)

For each `i` from `0` to `length - 1` (i.e., `0` to `4`):

Iteration	`Array[MAX_SIZE - length + i]`	`Array_Reversed[i]`
`i = 0`	`Array[10 - 5 + 0] = Array[5] = 1`	`Array_Reversed[0] = input[4] = 5`
`i = 1`	`Array[10 - 5 + 1] = Array[6] = 2`	`Array_Reversed[1] = input[3] = 4`
`i = 2`	`Array[10 - 5 + 2] = Array[7] = 3`	`Array_Reversed[2] = input[2] = 3`
`i = 3`	`Array[10 - 5 + 3] = Array[8] = 4`	`Array_Reversed[3] = input[1] = 2`
`i = 4`	`Array[10 - 5 + 4] = Array[9] = 5`	`Array_Reversed[4] = input[0] = 1`

Final Result

After the loop:

cpp Copy Edit

```cpp
Array = {0, 0, 0, 0, 0, 1, 2, 3, 4, 5};  // Right-aligned
Array_Reversed = {5, 4, 3, 2, 1, 0, 0, 0, 0, 0};  // Reversed
```

Step 2: Iteration (Filling `Array` and `Array_Reversed`)

For each `i` from `0` to `length - 1` (i.e., `0` to `4`):

Iteration	`Array[MAX_SIZE - length + i]`	`Array_Reversed[i]`
`i = 0`	`Array[10 - 5 + 0] = Array[5] = 1`	`Array_Reversed[0] = input[4] = 5`
`i = 1`	`Array[10 - 5 + 1] = Array[6] = 2`	`Array_Reversed[1] = input[3] = 4`
`i = 2`	`Array[10 - 5 + 2] = Array[7] = 3`	`Array_Reversed[2] = input[2] = 3`
`i = 3`	`Array[10 - 5 + 3] = Array[8] = 4`	`Array_Reversed[3] = input[1] = 2`
`i = 4`	`Array[10 - 5 + 4] = Array[9] = 5`	`Array_Reversed[4] = input[0] = 1`

Final Result

After the loop:

```cpp
Array = {0, 0, 0, 0, 0, 1, 2, 3, 4, 5};  // Right-aligned
Array_Reversed = {5, 4, 3, 2, 1, 0, 0, 0, 0, 0};  // Reversed
```

markdown Copy Edit

```
  00123   (num1)
+ 00089   (num2)
---------
  00212   (result)
```

Iteration-wise Breakdown:

Index	num1.Array[i]	num2.Array[i]	Carry In	Sum	Result.Array[i]	Carry Out
4 (rightmost)	'3' (3)	'9' (9)	0	12	'2'	1
3	'2' (2)	'8' (8)	1	11	'1'	1
2	'1' (1)	'0' (0)	1	2	'2'	0
1	'0' (0)	'0' (0)	0	0	'0'	0
0 (leftmost)	'0' (0)	'0' (0)	0	0	'0'	0

Final `result.Array = {'0', '0', '2', '1', '2'}` → **Represents** `212`.