Resizing Arrays

STACK allocated arrays cannot change size!

```
int main() {
  // Fixed size array on the stack
  int values[5] = {1, 2, 3, 4, 5};

  for (size_t i = 0; i < 5; ++i)
    printf("%i\n", values[i]);

  return 0;
}
```

Only HEAP arrays can resize

```
int main() {
  // Dynamic array on the heap
  int *values = calloc(5, sizeof(int));

  for (size_t i = 0; i < 5; ++i)
    printf("%i\n", values[i]);

  free(values);

  return 0;
}
```

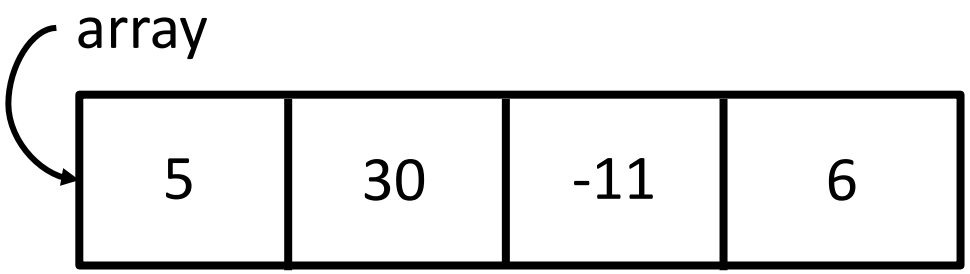
Refresher: Array Storage

- Arrays are stored in CONTIGUOUS areas of memory
  - Starting at the beginning of the array, each element in the array will follow the previous element consecutively in memory

- This is required due to the way we access array elements
  - The array variable is a pointer to the first element of the array (offset of 0)
  - Subsequent elements are found at the starting location + an offset
  - This is what happens when we request array[n] where 0 <= n < the array size.

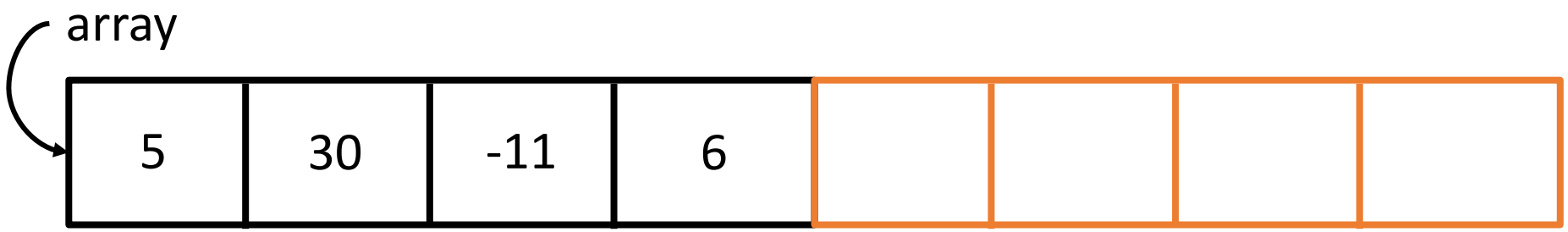
Resizing an Array

We can acquire more memory for our heap arrays in one of two ways depending on our memory layout.



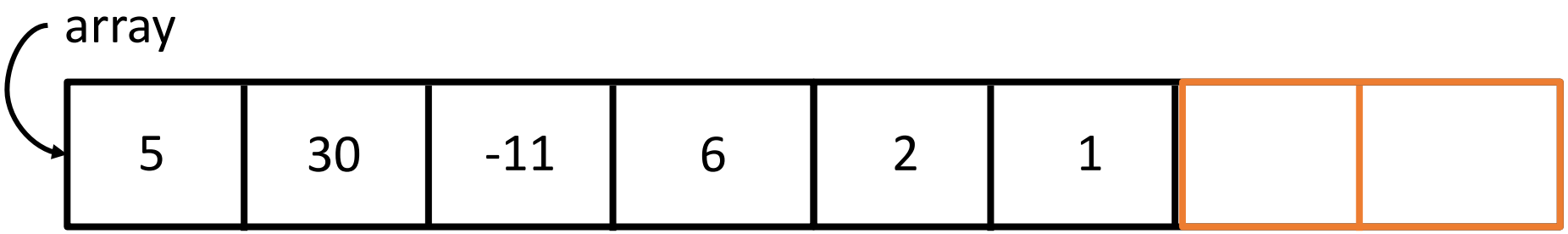
Resizing an Array: Condition #1

There is adequate free space next to the current array



Resizing an Array: Condition #1

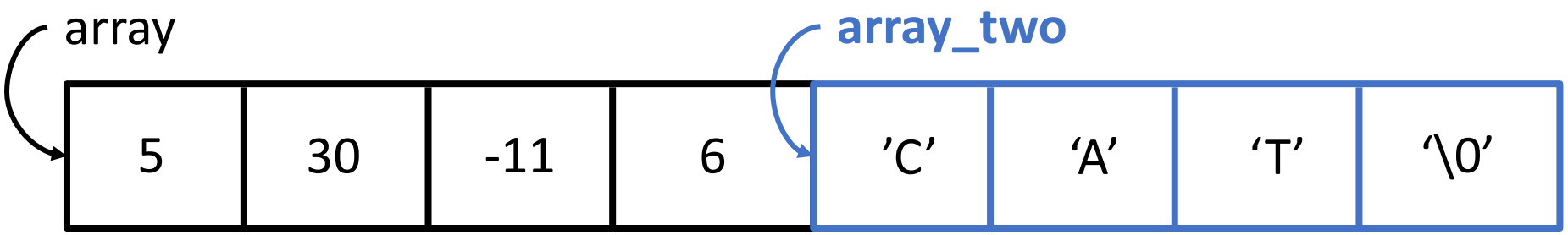
There is adequate free space next to the current array



We can grow our array in place and use the spare memory

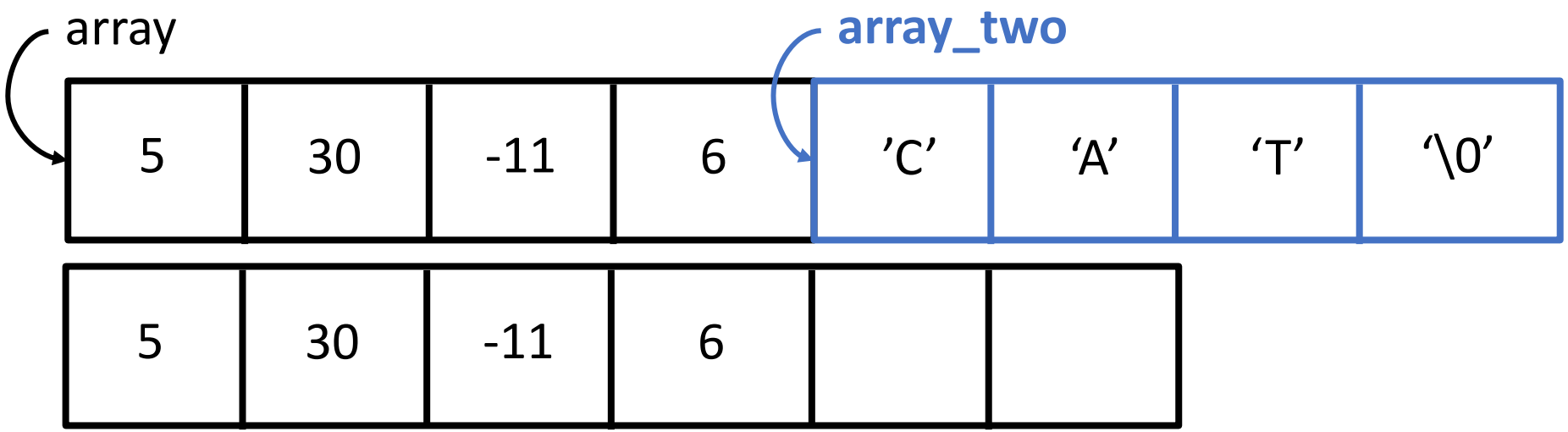
Resizing an Array: Condition #2

There is NOT enough free space next to the current array



Resizing an Array: Condition #2

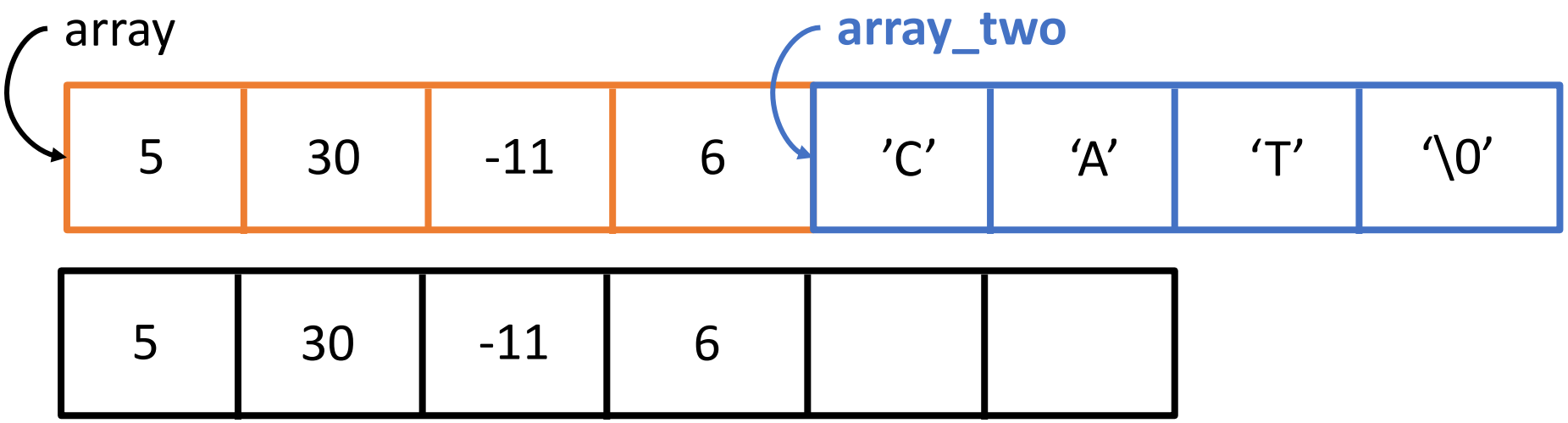
There is NOT enough free space next to the current array



1) Copy the old array to a larger memory location

Resizing an Array: Condition #2

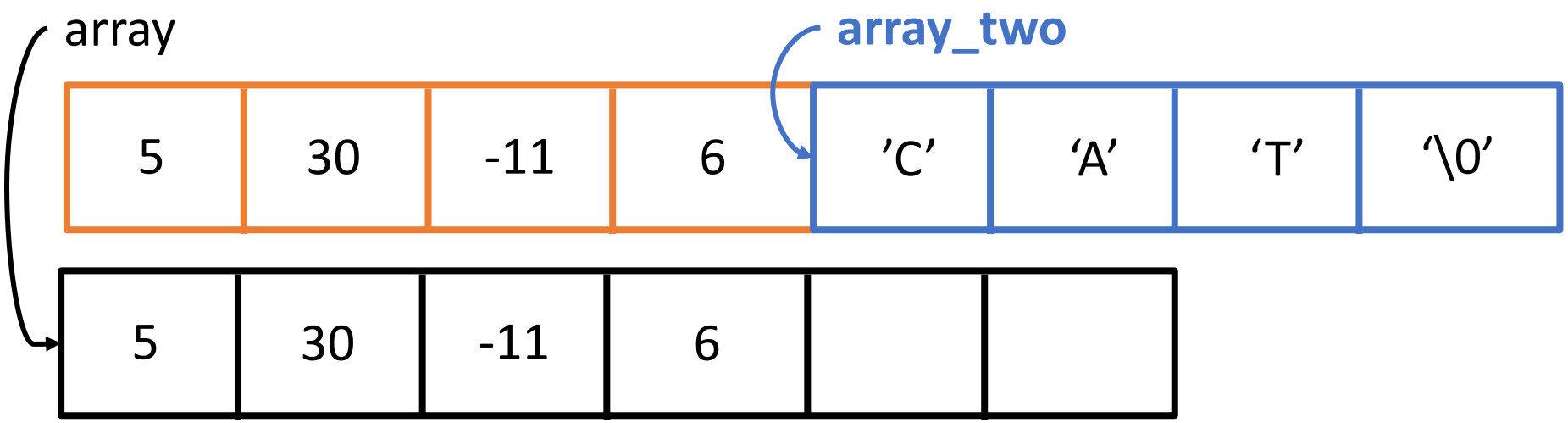
There is NOT enough free space next to the current array



1) Copy the old array to a larger memory location
2) Free the old array

Resizing an Array: Condition #2

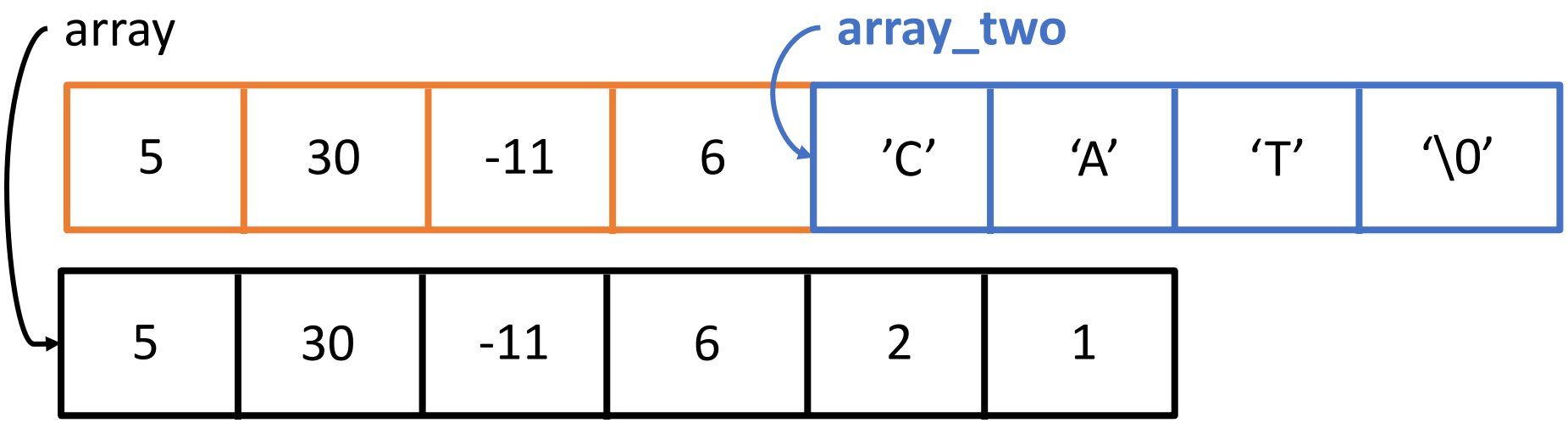
There is NOT enough free space next to the current array



1) Copy the old array to a larger memory location
2) Free the old array
3) Update the pointer

Resizing an Array: Condition #2

There is NOT enough free space next to the current array



1) Copy the old array to a larger memory location
2) Free the old array
3) Update the pointer

The realloc Function

- Good news, the realloc function will take care of this
  - `void *realloc(void *ptr, size_t size)`
  - Returns: a void pointer
  - Parameters: pointer to the original array, new size in bytes
- If there is consecutive free space next to the array realloc…
  - acquires extra space and returns the original pointer
- If there is insufficient space realloc…
  - creates a new array, copies the old data to the new array, frees old memory, and returns a pointer to the new array location