# Array-based list

# Source code

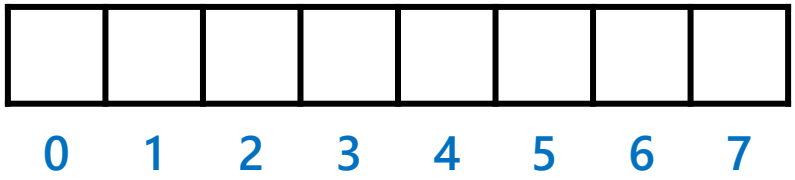
- available on Lectures page on onq
- see **`list.h`** and **`list.c`**
  - implements a dynamically resizable list of **`int`** elements
- new functions used:
  - **`memcpy`**
    - https://en.cppreference.com/w/c/string/byte/memccpy
  - **`memmove`**
    - https://en.cppreference.com/w/c/string/byte/memmove
  - **`snprintf`**
    - https://en.cppreference.com/w/c/io/fprintf

# Lists

- a list is a sequence of elements of a single type
  - elements have positions 0, 1, 2, …
- the size of a list is the number of elements in the list
  - the empty list has no elements
- the capacity of a list is the maximum number of elements that can be stored in the list
- adding an element to the list adds the element to the end of the list
  - increases the size by 1
- often convenient if the capacity automatically increases when the list becomes full

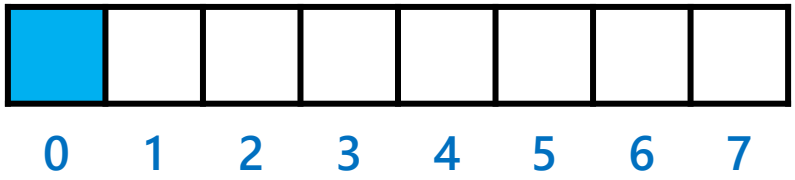
# Empty list

- capacity of 8, size of 0
- valid indexes: none



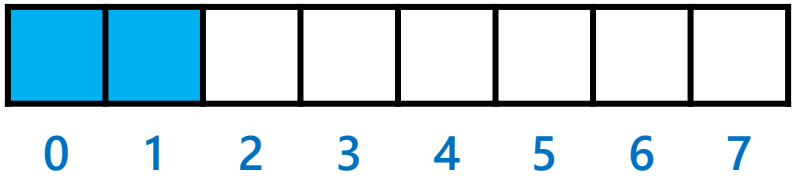
# List of size 1

- adding an element to an empty list
- capacity of 8, size of 1
- valid indexes: 0



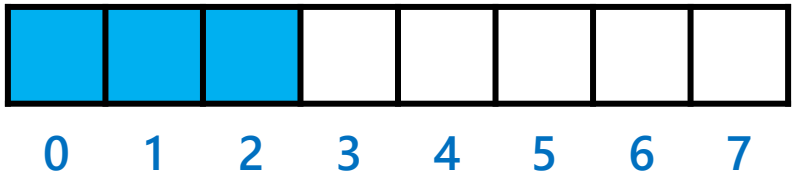
# List of size 2

- adding an element to the previous list
- capacity of 8, size of 2
- valid indexes: 0, 1



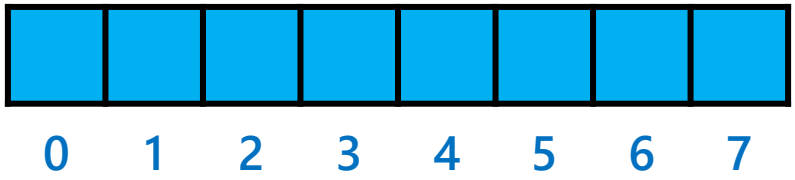
# List of size 3

- adding an element to the previous list
- capacity of 8, size of 3
- valid indexes: 0, 1, 2



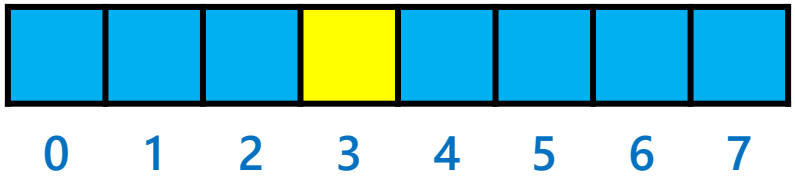
# List of size 8

- adding 5 more elements to the previous list
- capacity of 8, size of 8
- valid indexes: 0 to (size - 1), inclusive



# Getting/setting an element

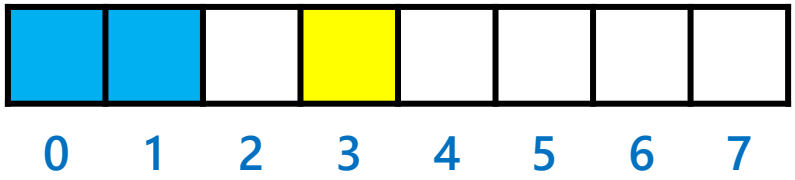
- an element can be retrieved/set using a zero-based index
  - e.g., getting the element at index 3 returns the value of the element shown in yellow




  - e.g. setting the element at index 3 changes the value of the element shown in yellow

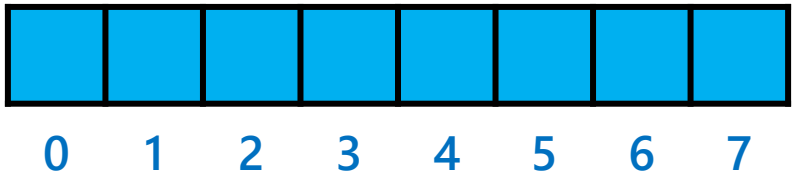
# Getting/setting an element

- undefined behavior if the index is invalid
  - e.g., getting/setting the element at index 3 can cause anything to happen



# Adding to a full list

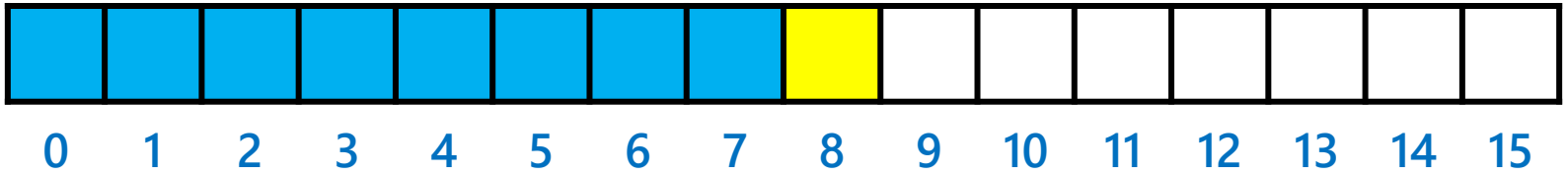
- in a resizable list, adding an element to a full list causes the list to resize its storage array and copy the existing elements into the newly allocated array




- the capacity of the new array should be a multiple (greater than 1) of the current capacity
  - e.g., double the current capacity

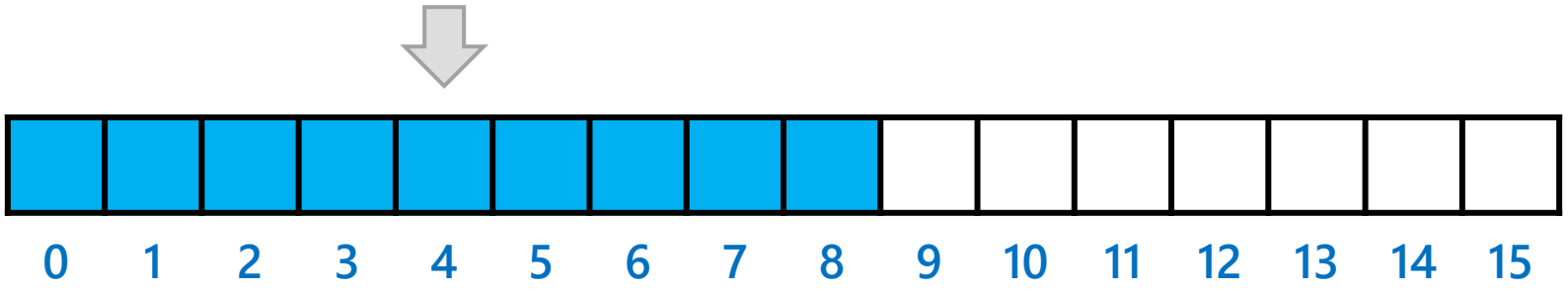
# Adding to a full list

- after increasing the capacity of the array, the element is added to the end of the unoccupied part of the array (shown in yellow)



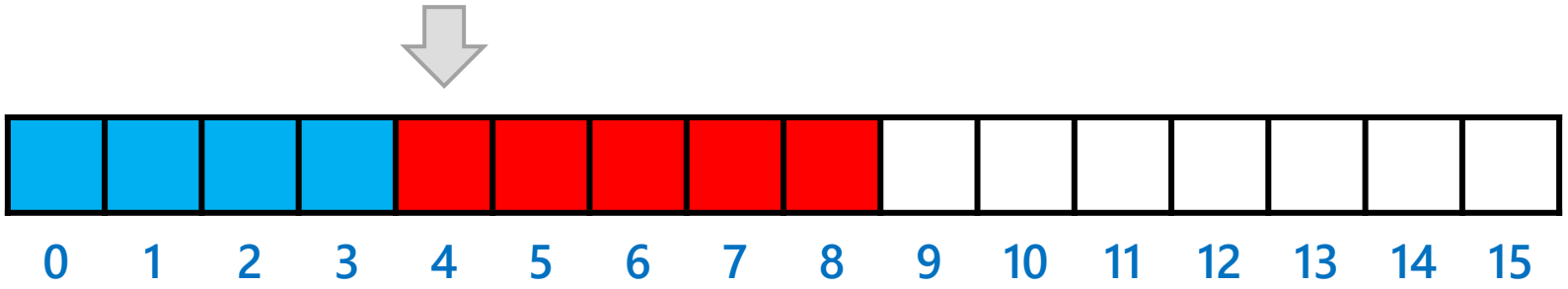
# Insertion using an index

- inserting an element into a list at a specified index requires moving the current elements starting at the specified index one position to the right
  - e.g., insert at index 4



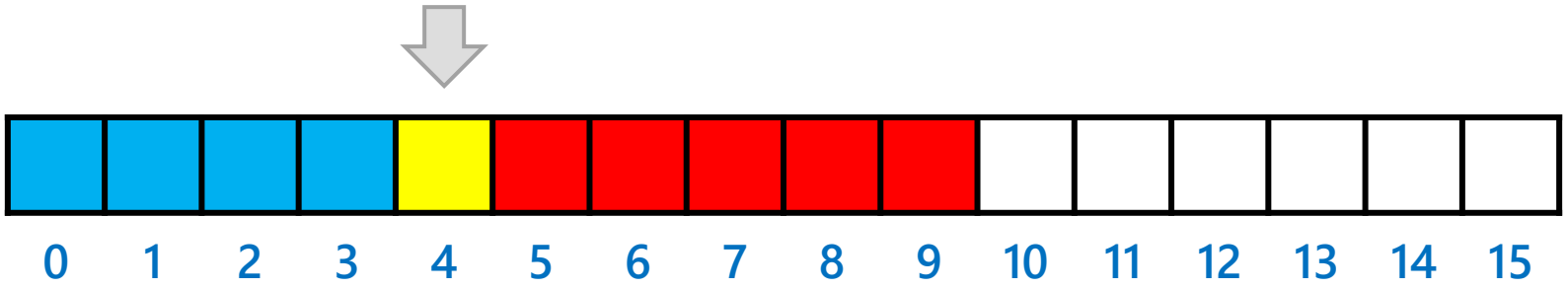
# Insertion using an index

- shifting the elements in red one position to the right…



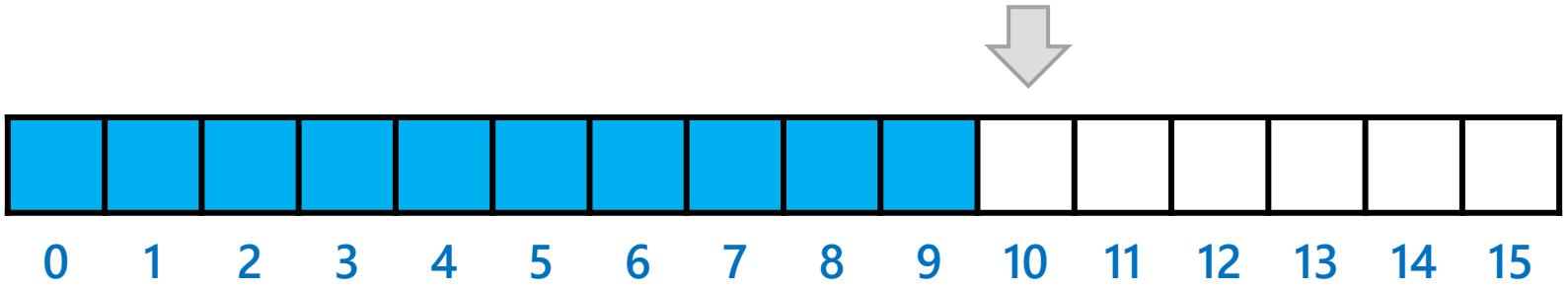
# Insertion using an index

- … makes room to insert the new element (shown in yellow)



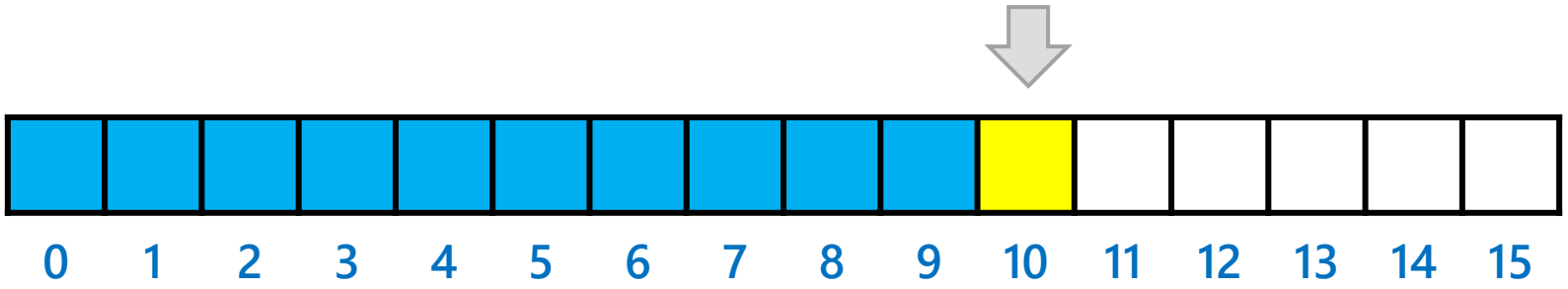
# Insertion using an index

- inserting at the end of the list is usually allowed
  - slightly unusual because the index is not technically valid



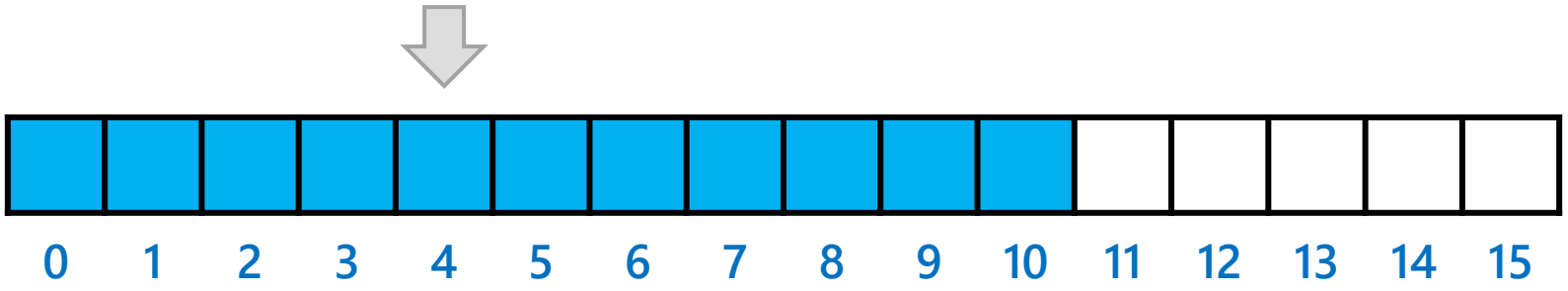
# Insertion using an index

- can be implemented simply by calling the add function



# Removing using an index

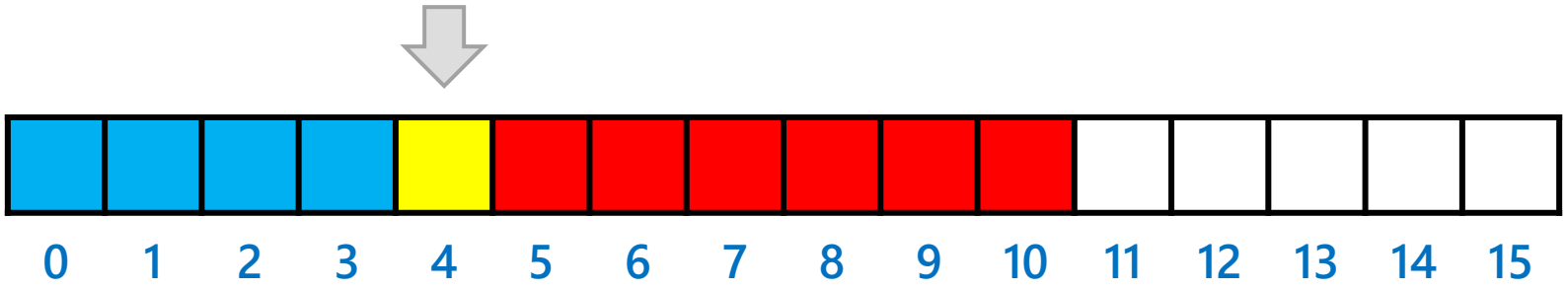
- removing an element at a specified index $i$ requires moving the current elements starting at $(i + 1)$ one position to the left




- the removed element is usually returned to the caller

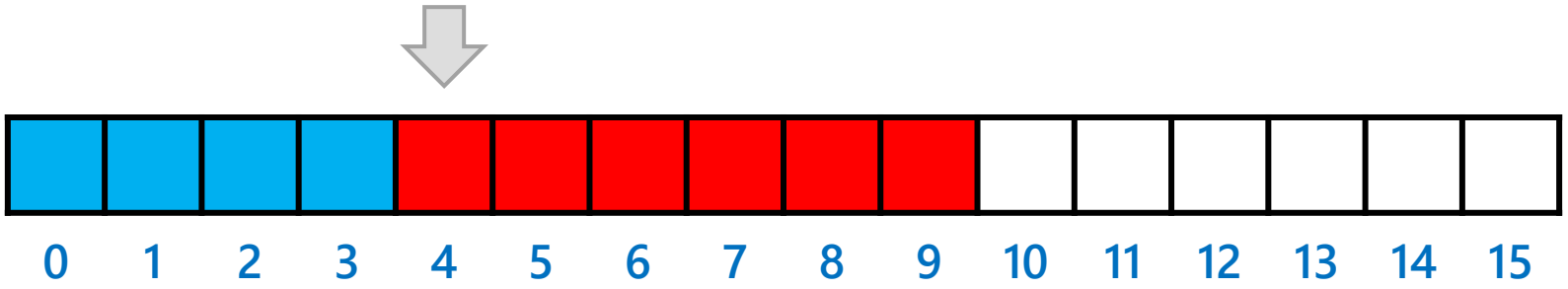
# Removing using an index

- first copy the element shown in yellow so that it can be returned to the caller, then
- shifting the elements in red one position to the left…



# Removing using an index

- … removes the element at the specified index



# assert

- including the header file **<assert.h>** allows the programmer to use the **assert** macro
  - a macro is a fragment of code that has been given a name
    - the preprocessor replaces all occurrences of the macro name with the macro code

```
assert(condition);
```

causes a running program to terminate if the condition is false

- assertions can be disabled via a command-line option to the compiler