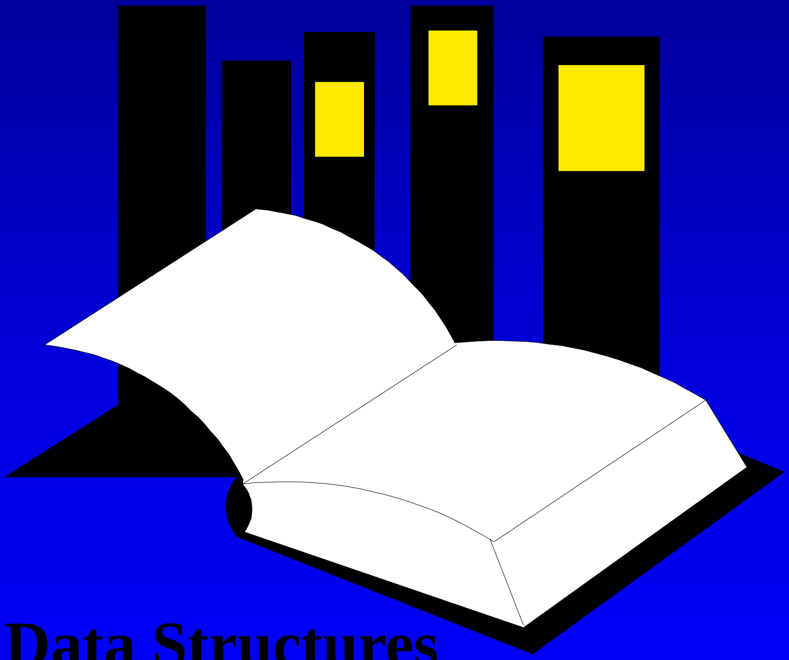
Using a Queue

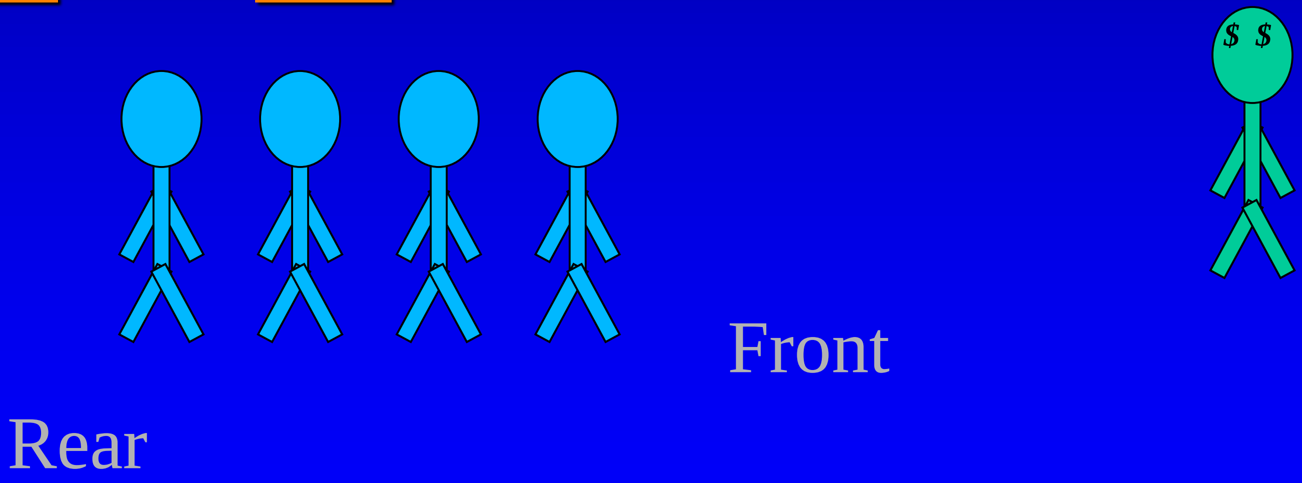


Data Structures and Other Objects Using C++

- Chapter 8 introduces the **queue** data type.
- Several example applications of queues are given in that chapter.
- This presentation describes the queue operations and two ways to implement a queue.

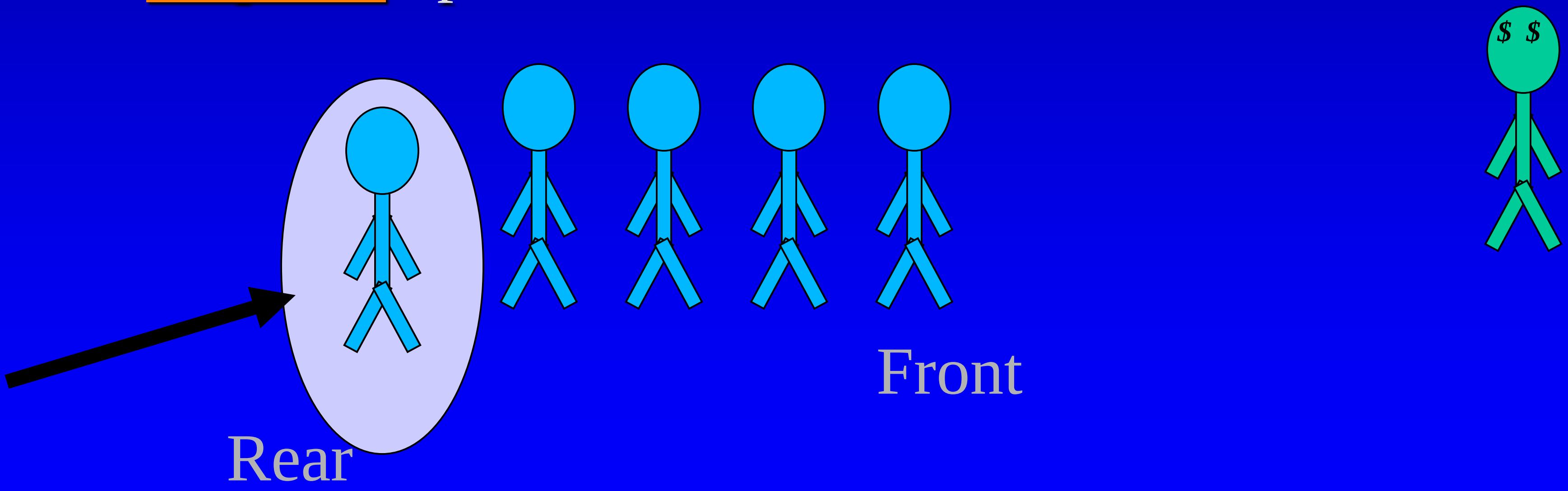
The Queue Operations

- A queue is like a line of people waiting for a bank teller. The queue has a **front** and a **rear**.



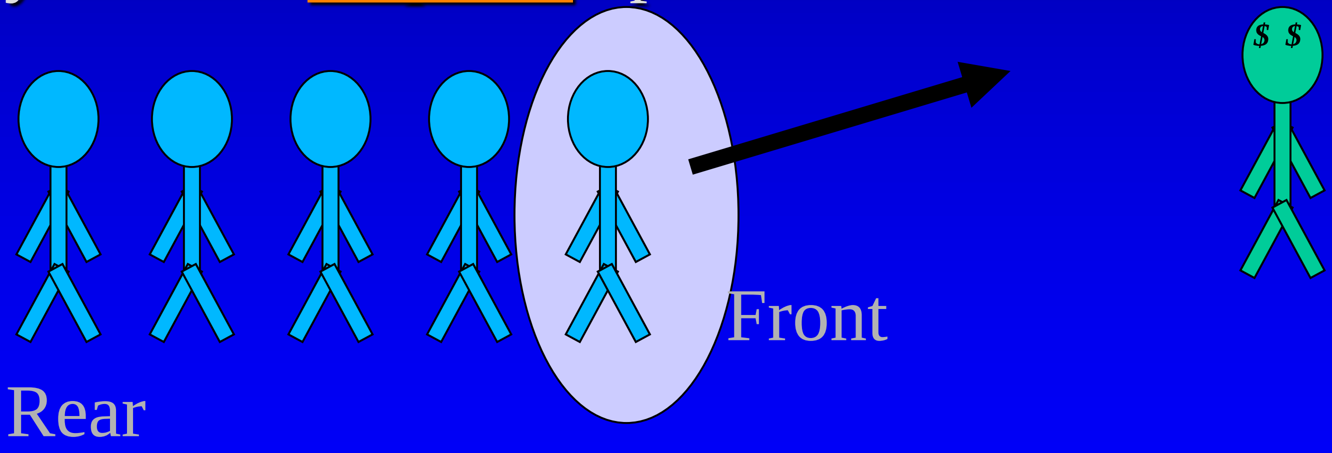
The Queue Operations

- New people must enter the queue at the rear. The C++ queue class calls this a **push**, although it is usually called an **enqueue** operation.



The Queue Operations

- When an item is taken from the queue, it always comes from the front. The C++ queue calls this a **pop**, although it is usually called a **dequeue** operation.



The Queue Class

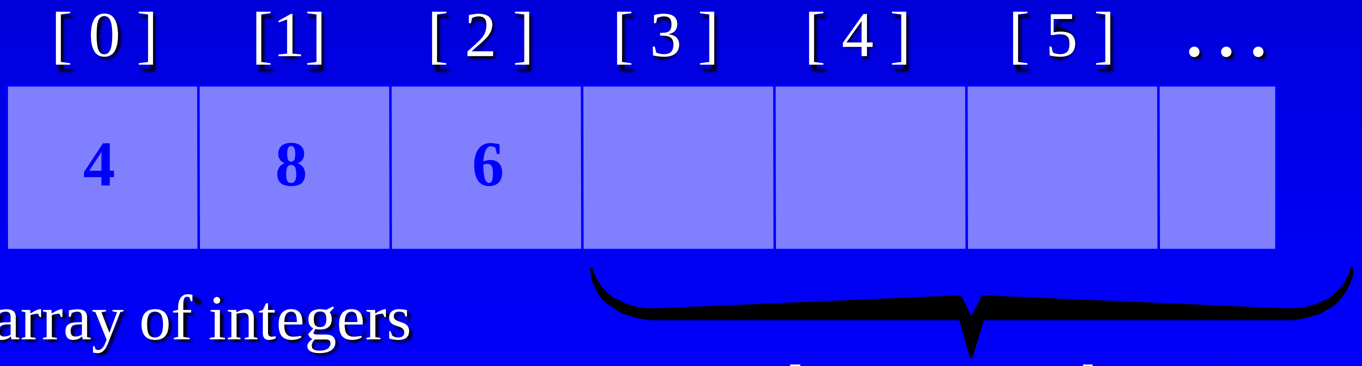
- The C++ standard template library has a queue template class.
- The template parameter is the type of the items that can be put in the queue.

```
template <class Item>
class queue<Item>
{
public:
        queue( );
        void push(const Item& entry);
        void pop( );
        bool empty( ) const;
        Item front( ) const;
        …

};
```

Array Implementation

- A queue can be implemented with an array, as shown here. For example, this queue contains the integers 4 (at the front), 8 and 6 (at the rear).

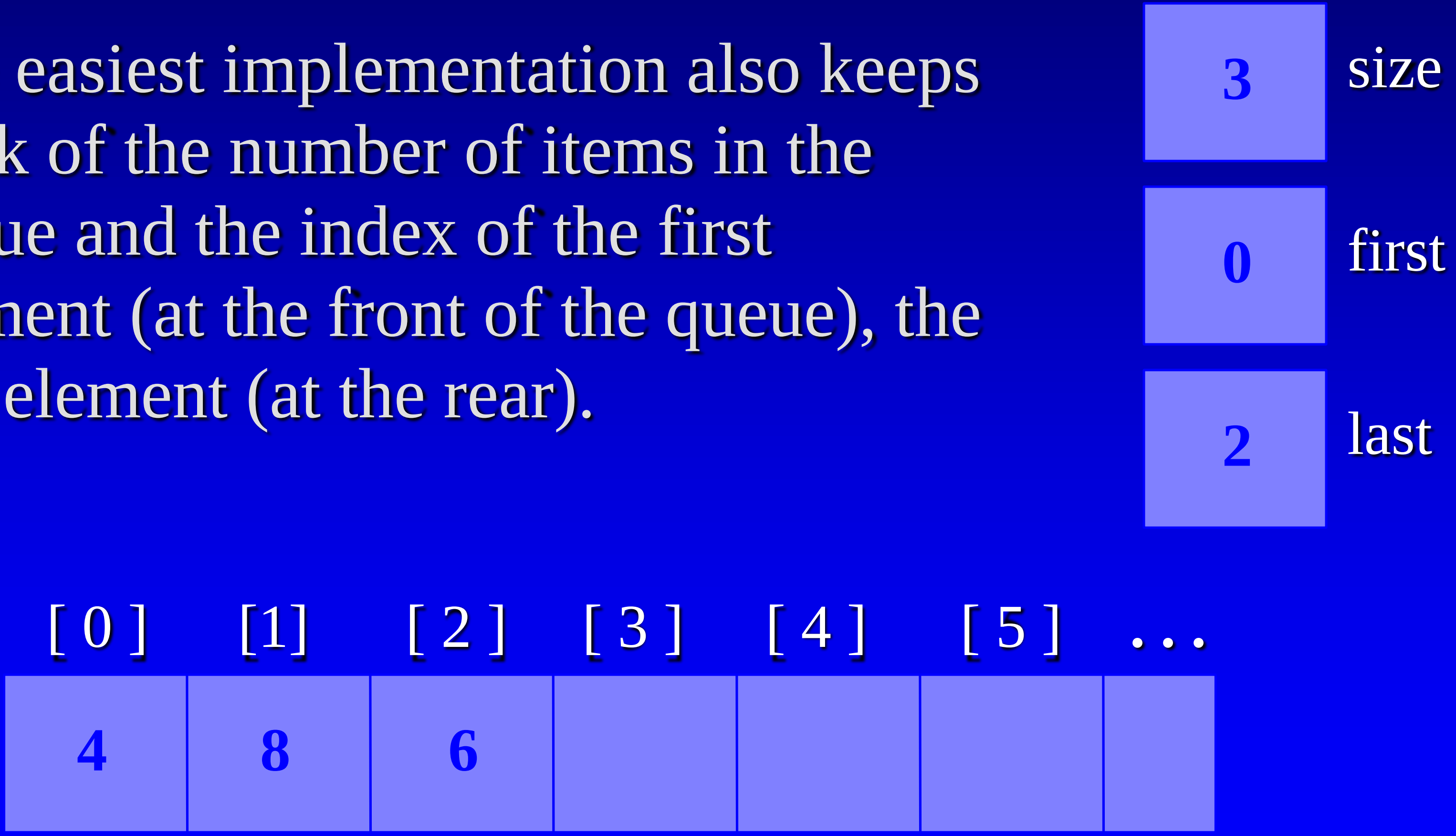


An array of integers to implement a queue of integers

We don't care what's in this part of the array.

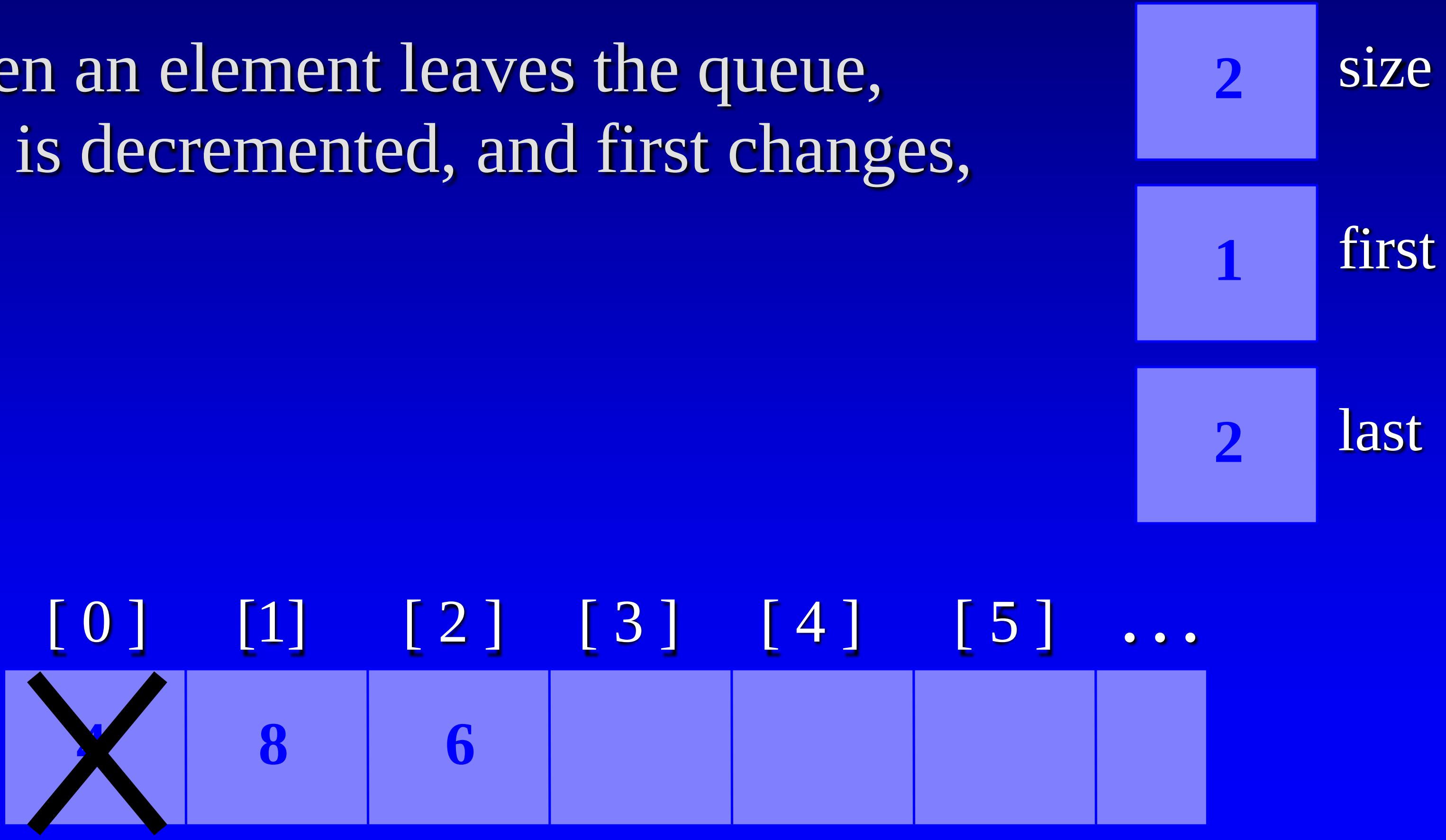
Array Implementation

- The easiest implementation also keeps track of the number of items in the queue and the index of the first element (at the front of the queue), the last element (at the rear).



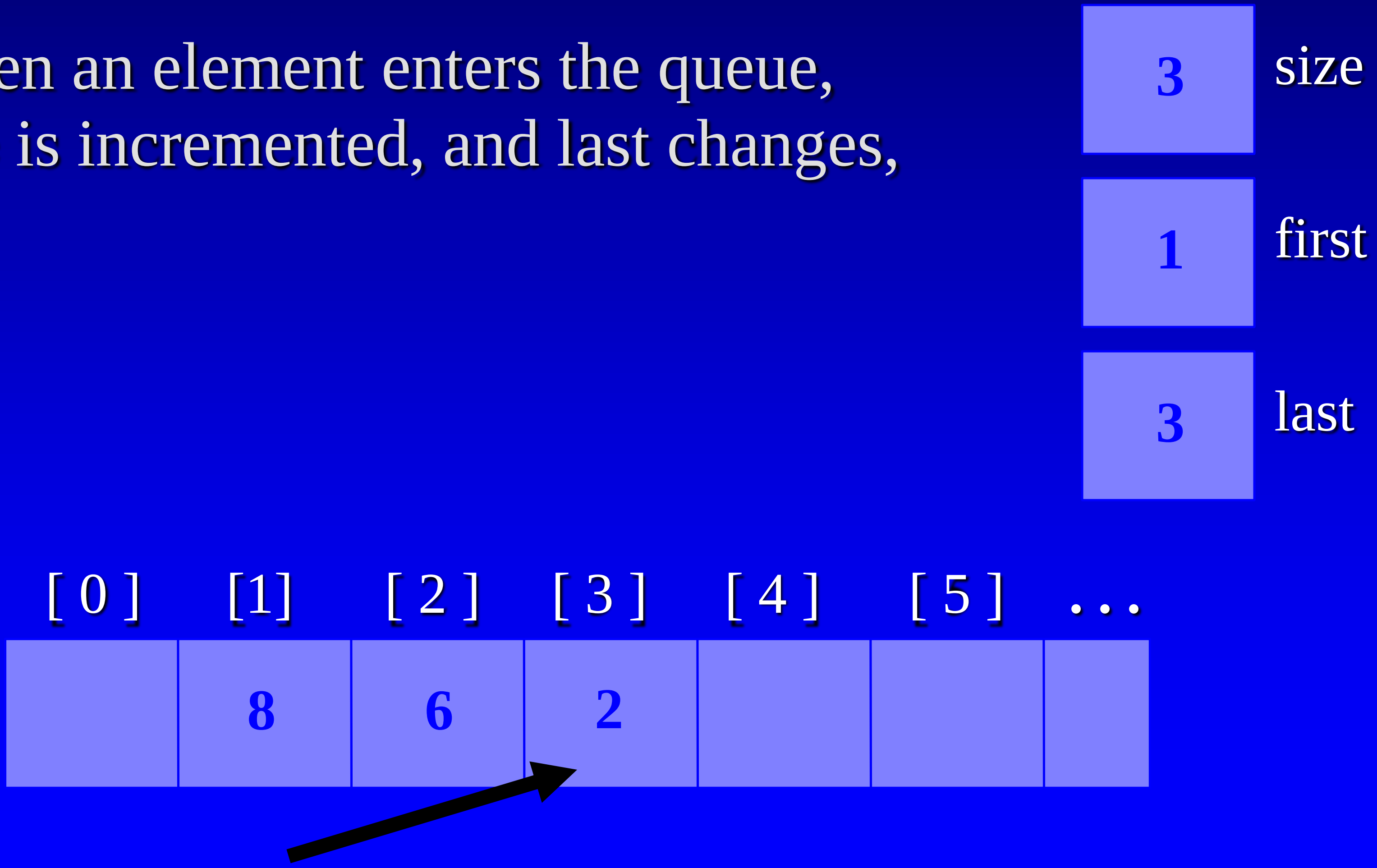
A Dequeue Operation

- When an element leaves the queue, size is decremented, and first changes, too.



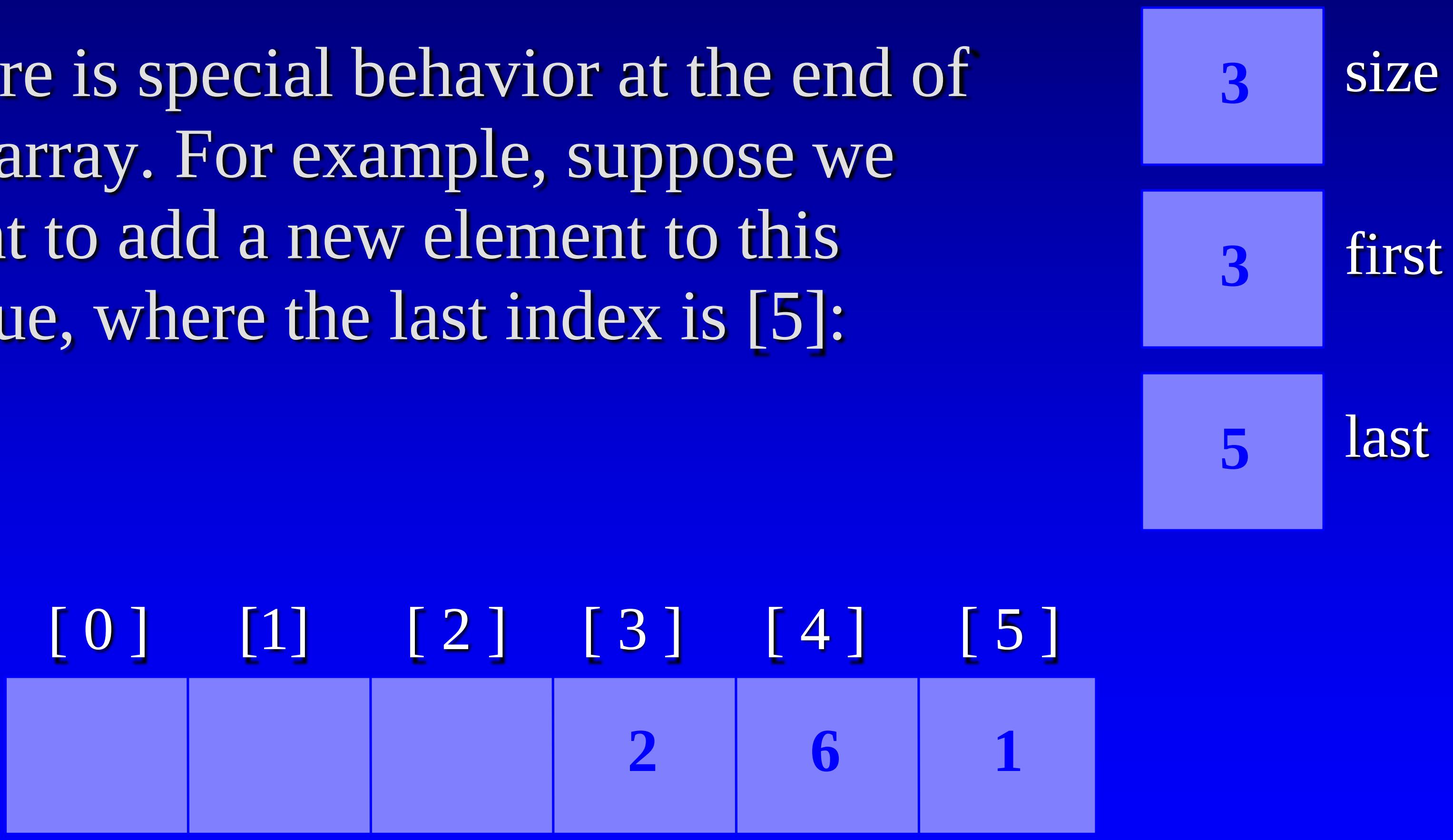
An Enqueue Operation

- When an element enters the queue, size is incremented, and last changes, too.



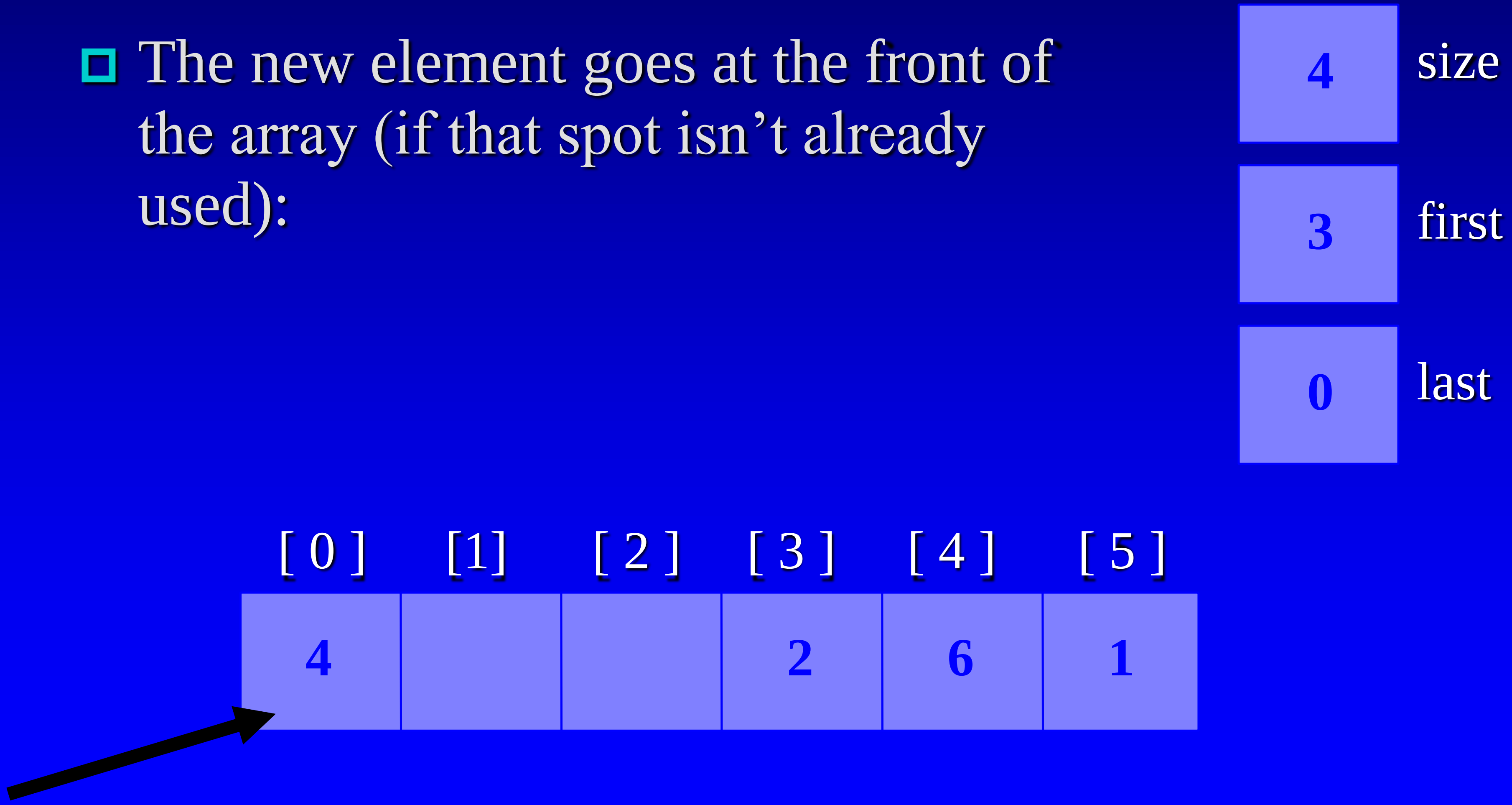
At the End of the Array

- There is special behavior at the end of the array. For example, suppose we want to add a new element to this queue, where the last index is [5]:



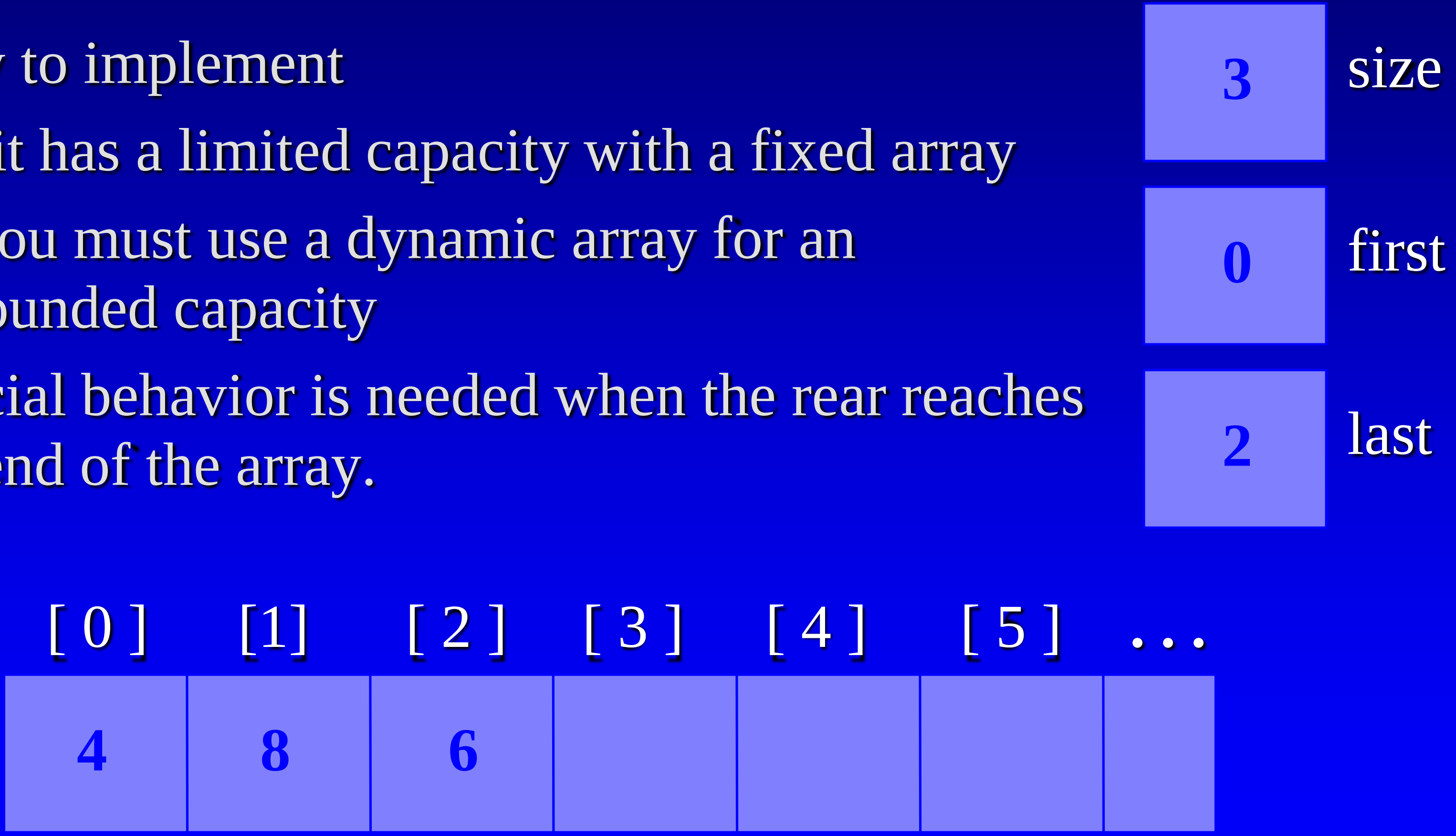
At the End of the Array

- The new element goes at the front of the array (if that spot isn't already used):



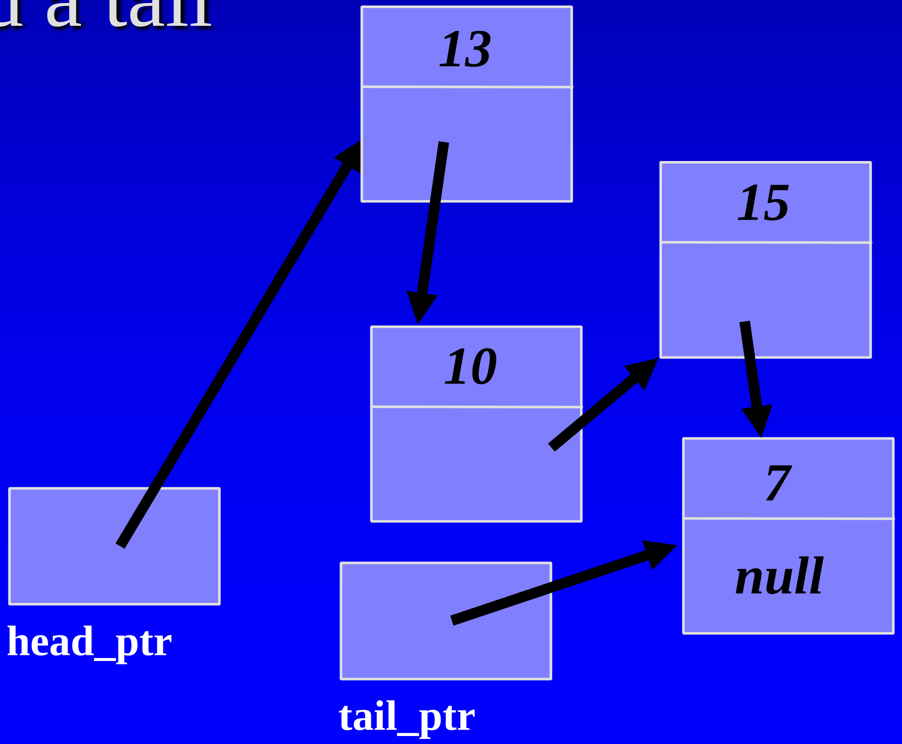
Array Implementation

- Easy to implement
- But it has a limited capacity with a fixed array
- Or you must use a dynamic array for an unbounded capacity
- Special behavior is needed when the rear reaches the end of the array.



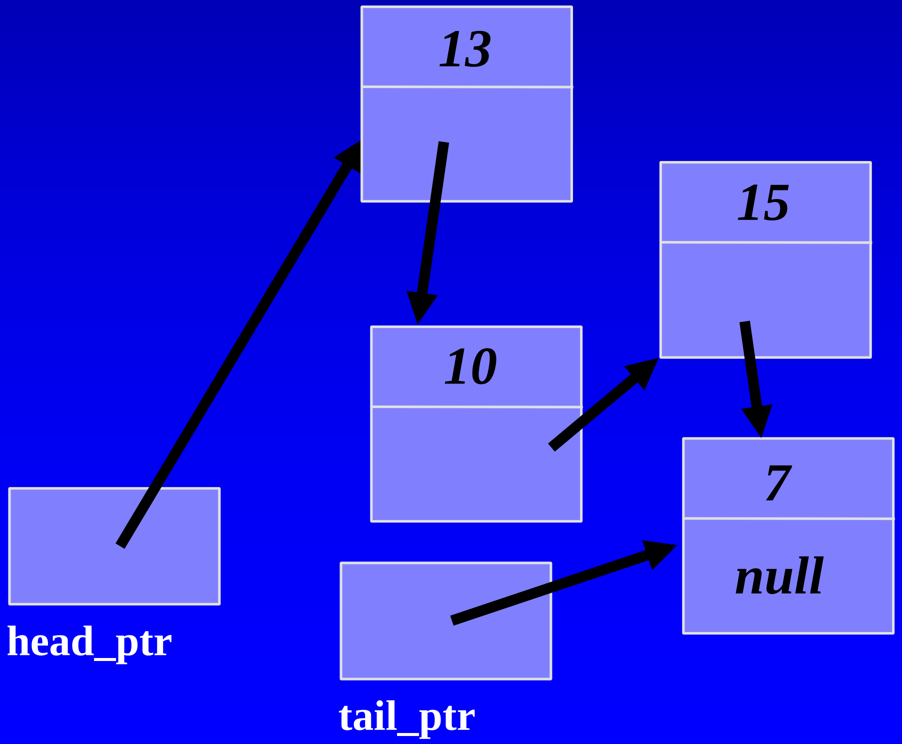
Linked List Implementation

- A queue can also be implemented with a linked list with both a head and a tail pointer.



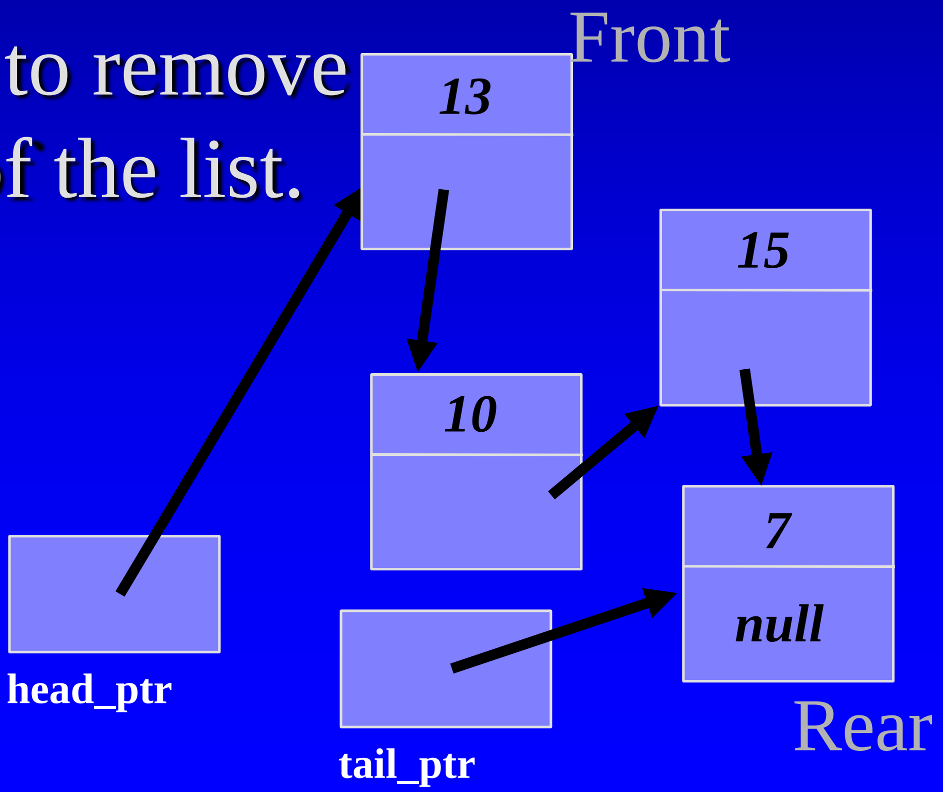
Linked List Implementation

- Which end do you think is the front of the queue? Why?



Linked List Implementation

- The head_ptr points to the front of the list.
- Because it is harder to remove items from the tail of the list.



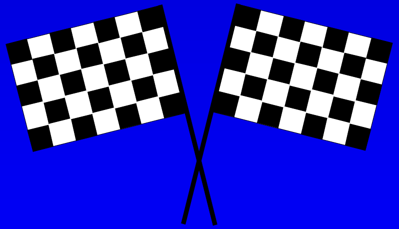
Summary

- Items enter a queue at the rear and leave a queue at the front.
- Queues can be implemented using an array or using a linked list.

Presentation copyright 2010, Addison Wesley Longman,
For use with *Data Structures and Other Objects Using C++*
by Michael Main and Walter Savitch.

Some artwork in the presentation is used with permission from Presentation Task Force (copyright New Vision Technologies Inc) and Corel Gallery Clipart Catalog (copyright Corel Corporation, 3G Graphics Inc, Archive Arts, Cartesia Software, Image Club Graphics Inc, One Mile Up Inc, TechPool Studios, Totem Graphics Inc).

Students and instructors who use *Data Structures and Other Objects* Using C++ are welcome to use this presentation however they see fit, so long as this copyright notice remains intact.



THE END