Algorithms and Data Structures

Queue

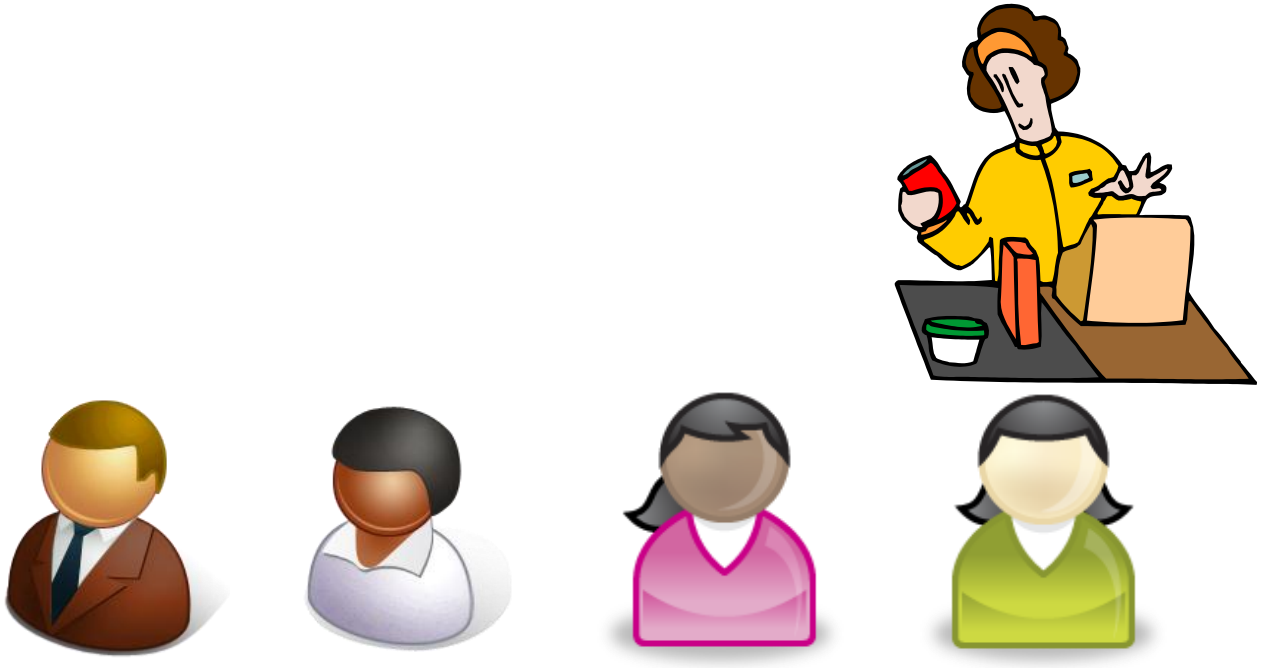
Robert Horvick

www.pluralsight.com

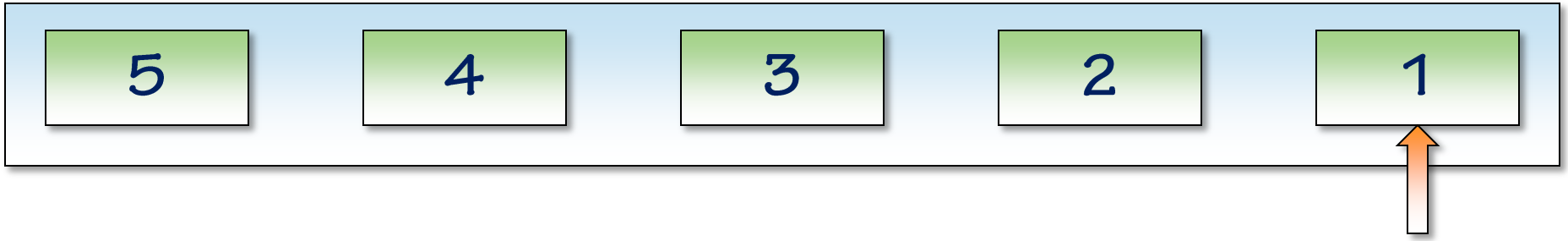
Outline

- First In, First Out (FIFO)
- Enqueue and Dequeue
- Linked List an Array implementations
- Priority Queue
- .NET and C++ Implementations

What is a Queue?

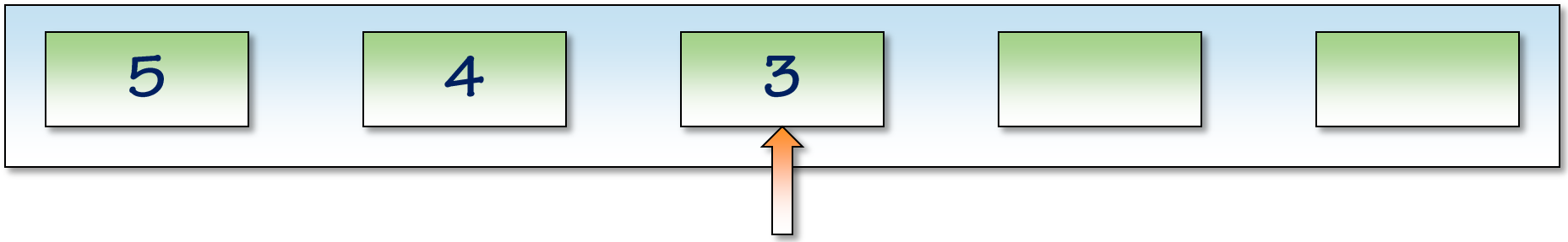


Enqueue




```
Queue<int> queue = new Queue<int>();
queue.Enqueue(1);
queue.Enqueue(2);
queue.Enqueue(3);
queue.Enqueue(4);
queue.Enqueue(5);
queue.Peek();
```

Dequeue

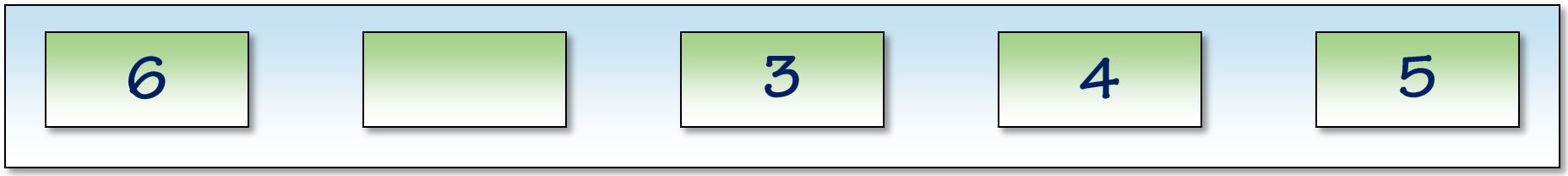



```
int one   = queue.Dequeue();
int two   = queue.Dequeue();
int three = queue.Dequeue();
int four  = queue.Dequeue();
int five  = queue.Dequeue();
```

Using a Linked List

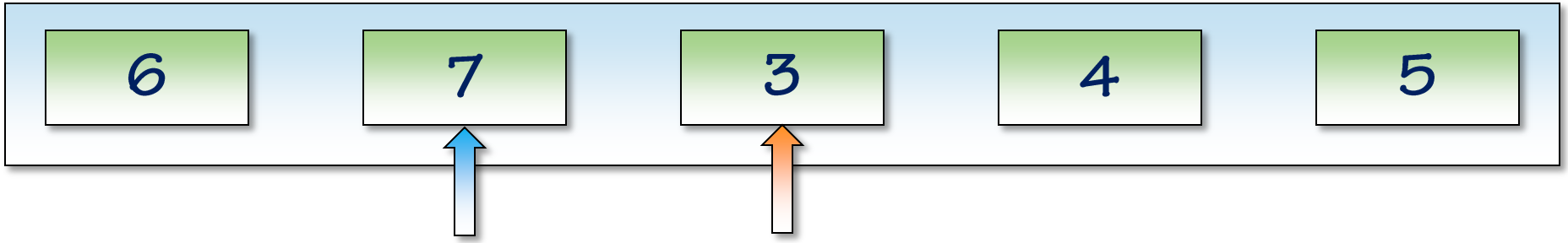
- Data items stored in LinkedList<T>
- Enqueue with AddLast
- LinkedList Head is Queue Head
- Dequeue with RemoveFirst
- AddLast/RemoveFirst allow list enumeration to “just work”

Using an Array (Enqueue/Dequeue)



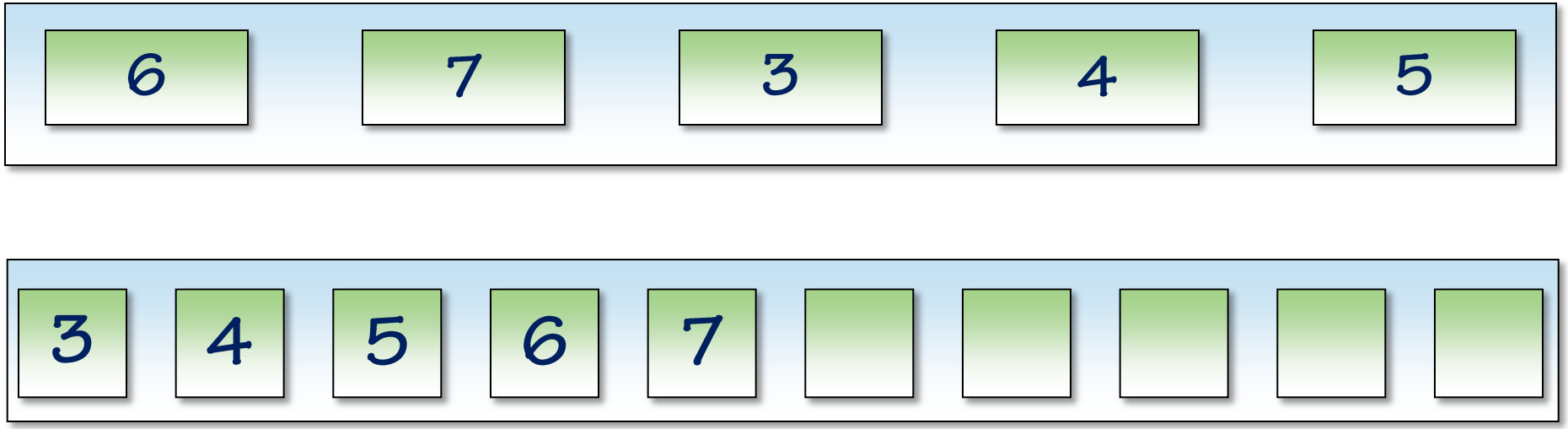

```
Queue<int> queue = new Queue<int>();
queue.Enqueue(1);
queue.Enqueue(2);
queue.Enqueue(3);
queue.Enqueue(4);
queue.Enqueue(5);
int one = queue.Dequeue();
int two = queue.Dequeue();
Queue.Enqueue(6);
```

Using an Array (Growth)



- Head is next item to Dequeue
- Tail is last item Enqueued
- Eventually the last open slot is filled
  - queue.Enqueue(7);
- What to do on next Enqueue?
  - Throw?
  - Grow?

Using an Array (Growth)



1. Allocate a new (longer) array
2. Copy Items from Head to End of array
3. If necessary, wrap and copy from 0 to Head-1
4. Use the new array
5. Update Head and Tail pointers

Priority Queue

- **Highest priority items Dequeued first**
  - Not First In, First Out
- **Call: Noise Complaint**
- **Call: Auto Accident**
- **Call: Theft**
- **Call: Noise Complaint**

Modern Implementations

- C#
  - Queue<T>
  - Enqueue, Dequeue
  - Implemented as Array

```
Queue<int> q = new Queue<int>();

for (int i = 0; i < 10; i++)
{
    q.Enqueue(i);
}

while (q.Count > 0)
{
    Console.WriteLine(q.Dequeue());
}
```

- C++
  - std::queue
  - push, pop, front
  - std::priority_queue

```
std::queue<int> q;

for(int i = 0; i < 10; i++)
{
    q.push(i);
}

while(!q.empty())
{
    std::cout << q.front() << std::endl;
    q.pop();
}
```

Summary

- **First In, First Out (FIFO)**
- **As Linked List**
  - Simple implementation
- **As Array**
  - Higher performance
- **Priority Queue**
  - Highest Priority Returned First
- **.NET and C++ Implementations**

References

- .NET Framework: Queue<T>
  - http://msdn.microsoft.com/en-us/library/7977ey2c.aspx
- C++ std::queue<T>
  - http://msdn.microsoft.com/en-us/library/s23s3de6.aspx