Algorithms and Data Structures

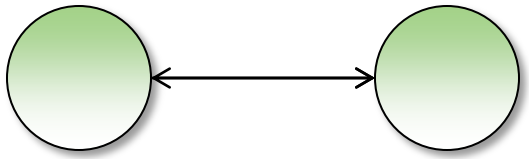
Linked Lists

Robert Horvick

www.pluralsight.com

Outline

- Node
- Node chains
- Linked List
- Doubly Linked List
- Modern Implementations



The Node

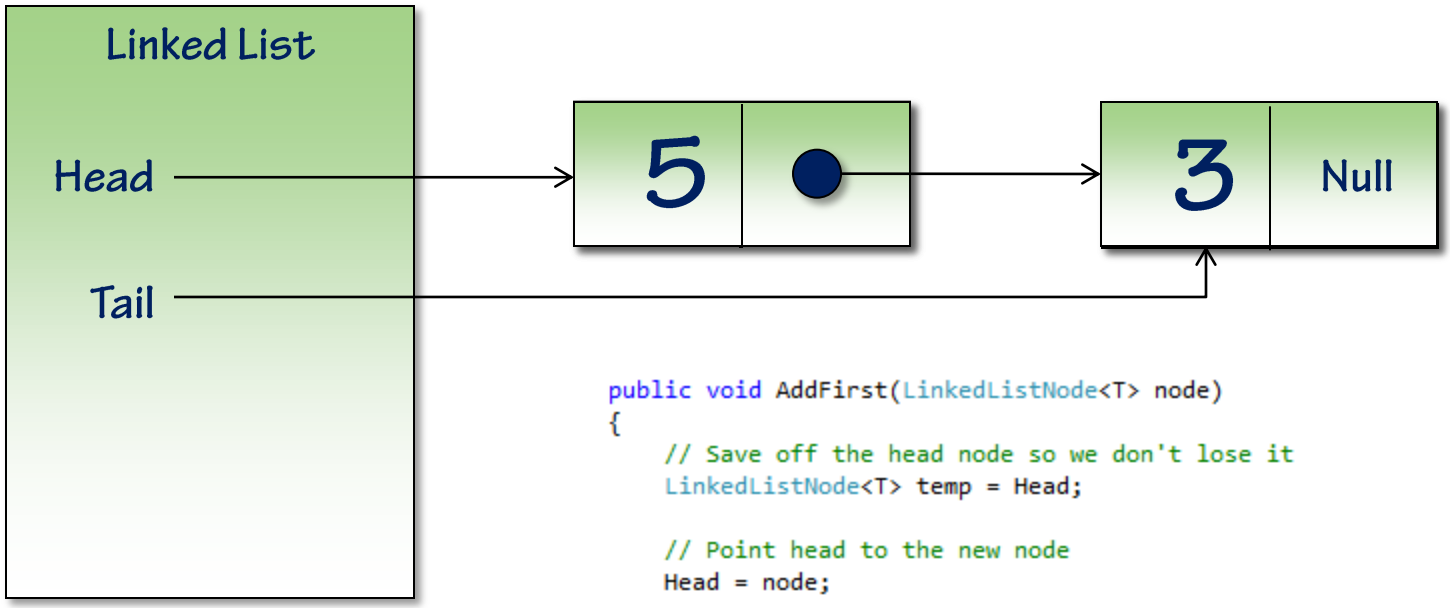
Node Chains

```
public class Node
{
    public int Value { get; set; }
    public Node Next { get; set; }
}

Node first = new Node { Value = 3 };
Node middle = new Node { Value = 5 };
first.Next = middle;

Node last = new Node { Value = 7 };
middle.Next = last;
```

Linked List

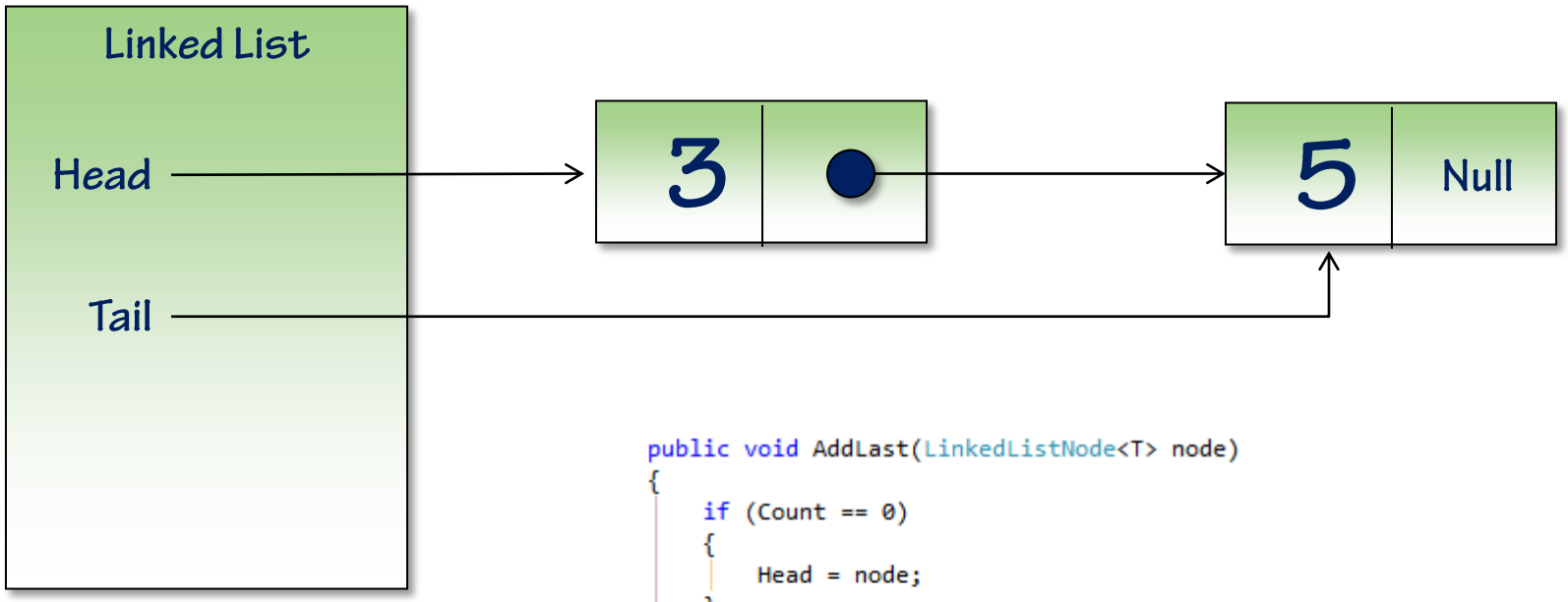
- Single chain of nodes
- Head Pointer
- Tail Pointer
- Operations
  - Add
  - Remove
  - Find
  - Enumerate

Add to Front



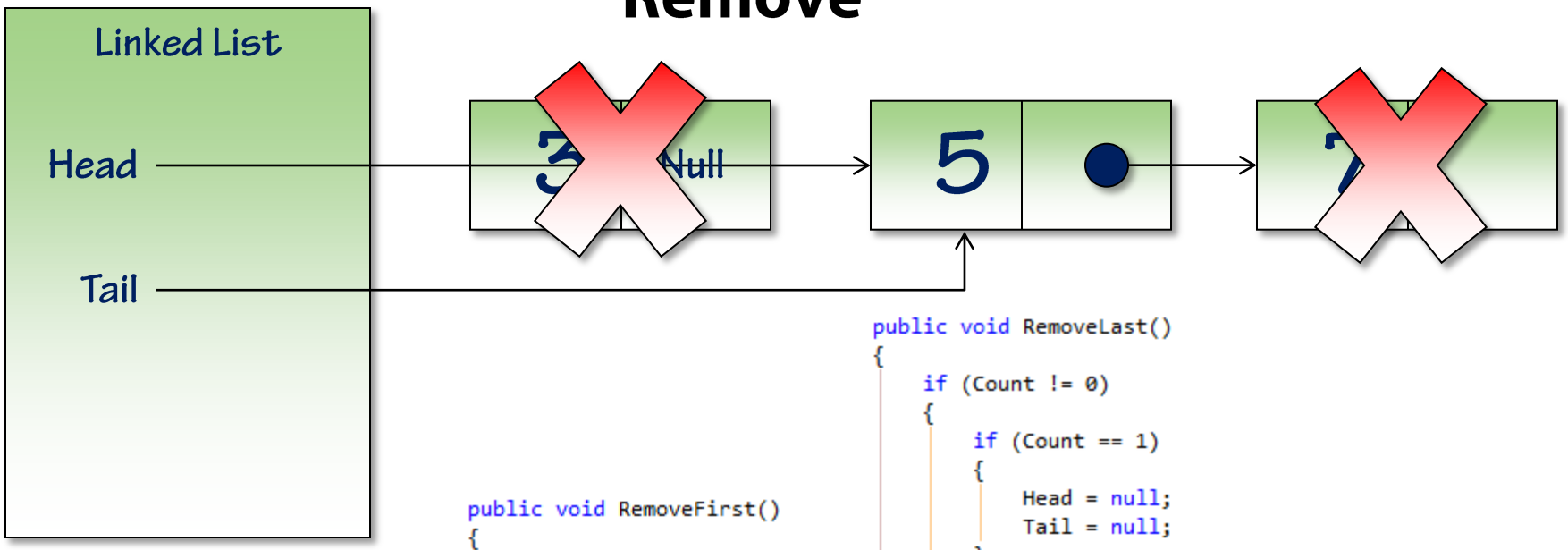

```
public void AddFirst(LinkedListNode<T> node)
{
    // Save off the head node so we don't lose it
    LinkedListNode<T> temp = Head;

    // Point head to the new node
    Head = node;

    // Insert the rest of the list behind the head
    Head.Next = temp;

    Count++;

    if (Count == 1)
    {
        // if the list was empty then Head and Tail should
        // both point to the new node.
        Tail = Head;
    }
}
```

Add to End




```
public void AddLast(LinkedListNode<T> node)
{
    if (Count == 0)
    {
        Head = node;
    }
    else
    {
        Tail.Next = node;
    }

    Tail = node;
    Count++;
}
```

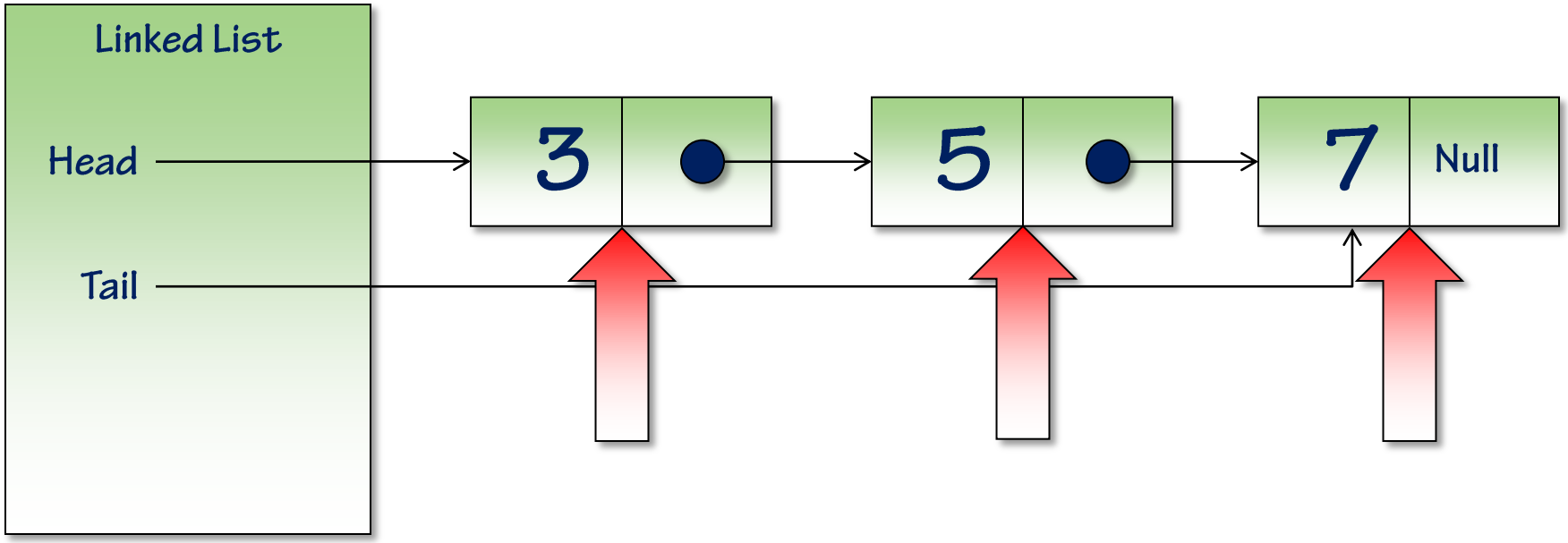
Remove




```
public void RemoveFirst()
{
    if (Count != 0)
    {
        Head = Head.Next;
        Count--;

        if (Count == 0)
        {
            Tail = null;
        }
    }
}
```

```
public void RemoveLast()
{
    if (Count != 0)
    {
        if (Count == 1)
        {
            Head = null;
            Tail = null;
        }
        else
        {
            LinkedListNode<T> current = Head;
            while (current.Next != Tail)
            {
                current = current.Next;
            }

            current.Next = null;
            Tail = current;
        }

        Count--;
    }
}
```

Enumerate




```
System.Collections.Generic.IEnumerator<T> System.Collections.Generic.IEnumerable<T>.GetEnumerator()
{
    LinkedListNode<T> current = Head;
    while (current != null)
    {
        yield return current.Value;
        current = current.Next;
    }
}
```

Doubly Linked List

Modern Implementations

- **.NET Framework**
  - LinkedList<T>
- **C++**
  - std::list<T>

.NET Framework

```
using System;
using System.Collections.Generic;

namespace NetFxLinkedList
{
    class Program
    {
        static void Main(string[] args)
        {
            LinkedList<int> list = new LinkedList<int>();
            list.AddLast(3);
            list.AddLast(5);
            list.AddLast(7);

            foreach (int value in list)
            {
                Console.WriteLine(value);
            }
        }
    }
}
```

- System.Collections.Generic
- Doubly linked list
- Common Operations
  - AddFirst, AddLast
  - RemoveFirst, RemoveLast
  - Find, FindLast

C++

```cpp
#include <iostream>
#include <algorithm>
#include <list>

int main()
{
    std::list<int> list;
    list.push_back(3);
    list.push_back(5);
    list.push_back(7);

    std::for_each(list.begin(), list.end(), [](int value) {
        std::cout << value << std::endl;
    });
}
```

- #include <list>
- Doubly linked list
- Common Operations
  - push_front, push_back
  - pop_front, pop_back
  - iterators

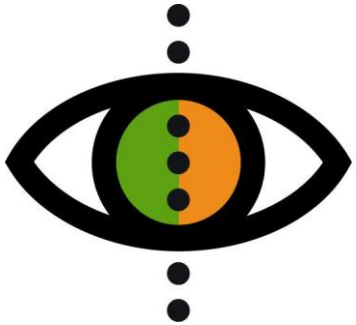
Summary

- **Nodes and node chaining**
- **Singly and doubly linked lists**
- **Operations**
  - Add
  - Remove
  - Enumerate
  - Find
- **Modern Implementations**
  - LinkedList<T>
  - std::list<T>

References

- LinkedList<T> on MSDN
  - http://msdn.microsoft.com/en-us/library/he2s3bh7.aspx
- std::list<T> on MSDN
  - http://msdn.microsoft.com/en-us/library/802d66bt(v=VS.100).aspx
- NUnit
  - http://www.nunit.org/
- LinkedList on Wikipedia
  - http://en.wikipedia.org/wiki/Linked_list

For more in-depth online developer training visit



pluralsight

see what you can learn

on-demand content from authors you trust