# High-performance Modifications with Linked Lists

Simon Robinson
SOFTWARE DEVELOPER

@techiesimon www.simonrobinson.com

# Overview

**Linked lists**

- Efficient for adding and removing elements
- Complex
- No direct element lookup

# Demo

**TourBooker app:**

- Input tours into the app:
    - Create and edit a list of countries
    - Using a linked list

# Linked Lists vs. Lists

## LinkedList<T>

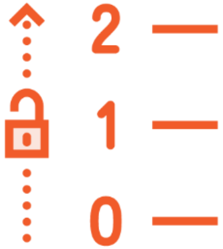
Definite order

Optimized for fast changes

## List<T> (and T[ ])

Definite order

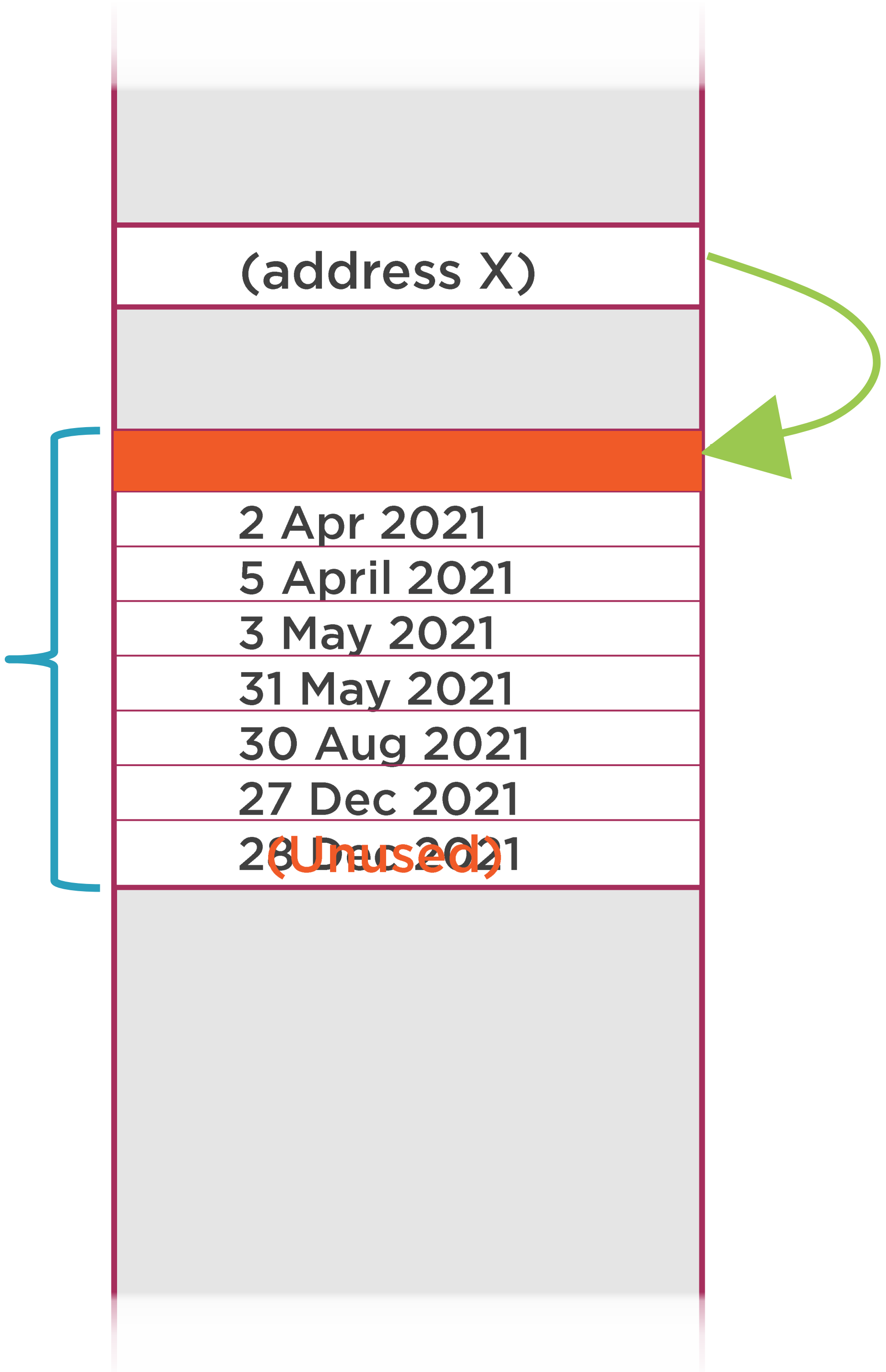
Optimized for fast lookup



# Structure of a List

```
bankHolsLst.RemoveAt(0);
```

Everything stored sequentially in memory



(address X)

2 Apr 2021

5 April 2021

3 May 2021

31 May 2021

30 Aug 2021

27 Dec 2021

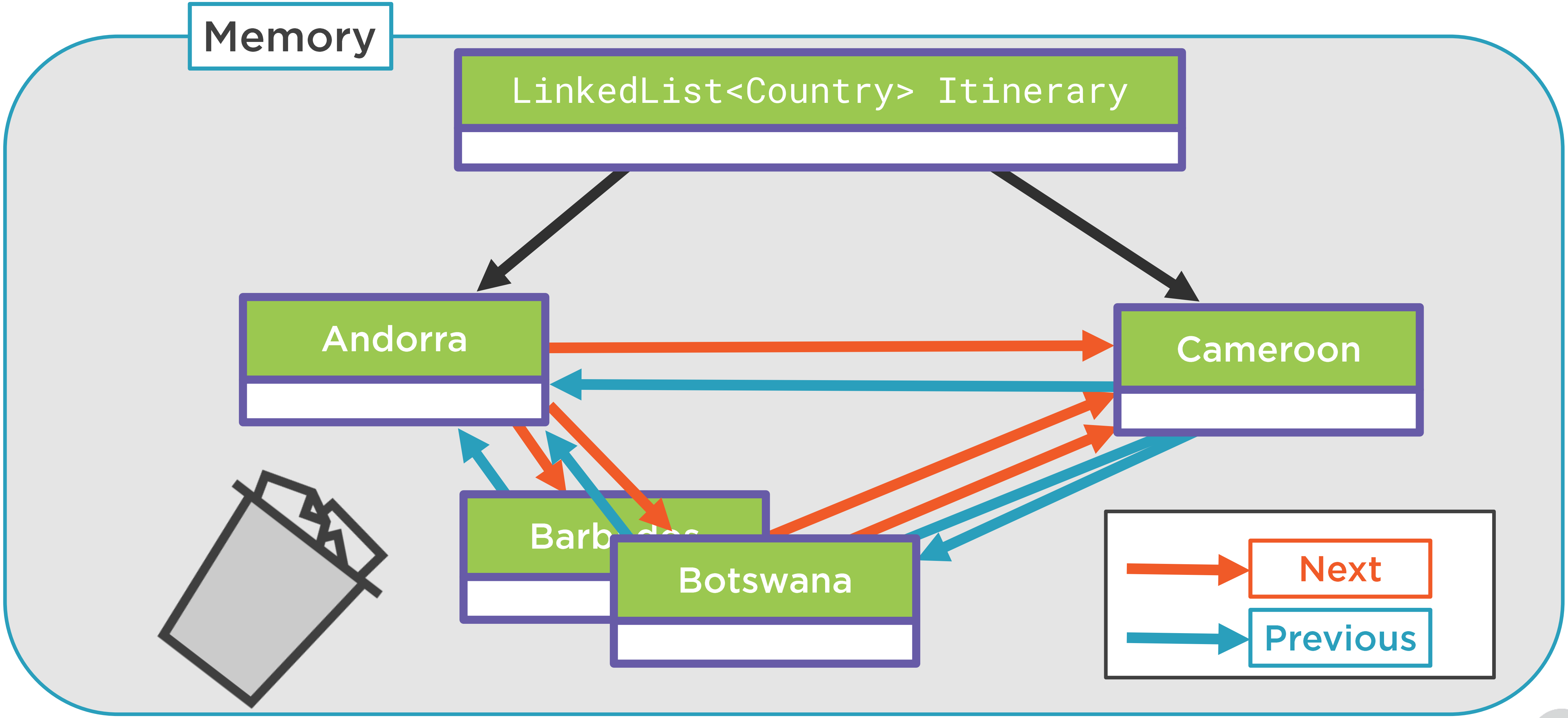
28 Dec 2021

(Unused)

# Linked Lists

No requirement to store items sequentially in memory

# Linked Lists



# Linked Lists

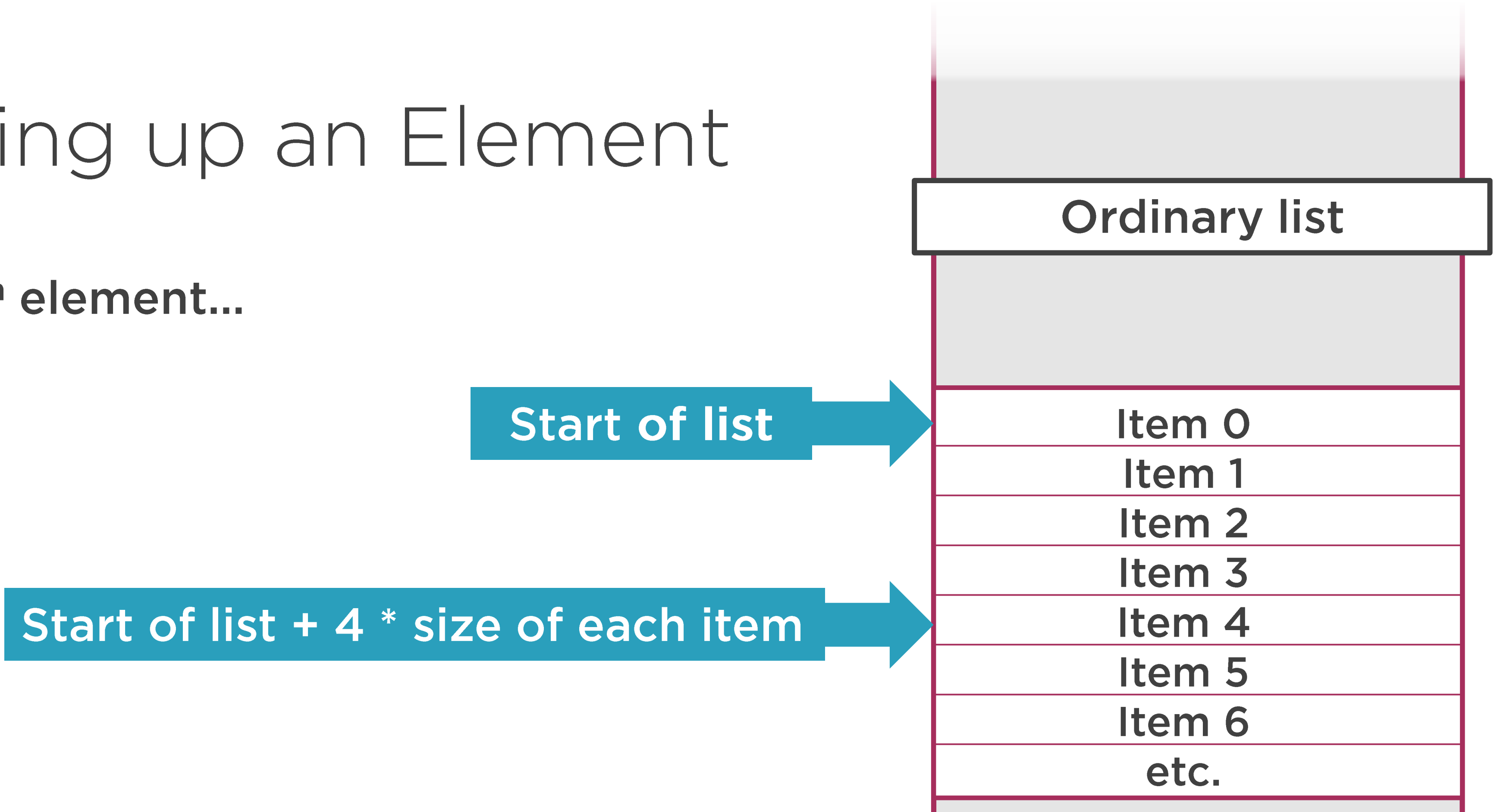
Adding and removing is fast

Never have to move stuff in memory

No fast lookup

# Looking up an Element

To get 5th element…



Ordinary list

Start of list

Start of list + 4 * size of each item

Item 0

Item 1

Item 2

Item 3

Item 4

Item 5

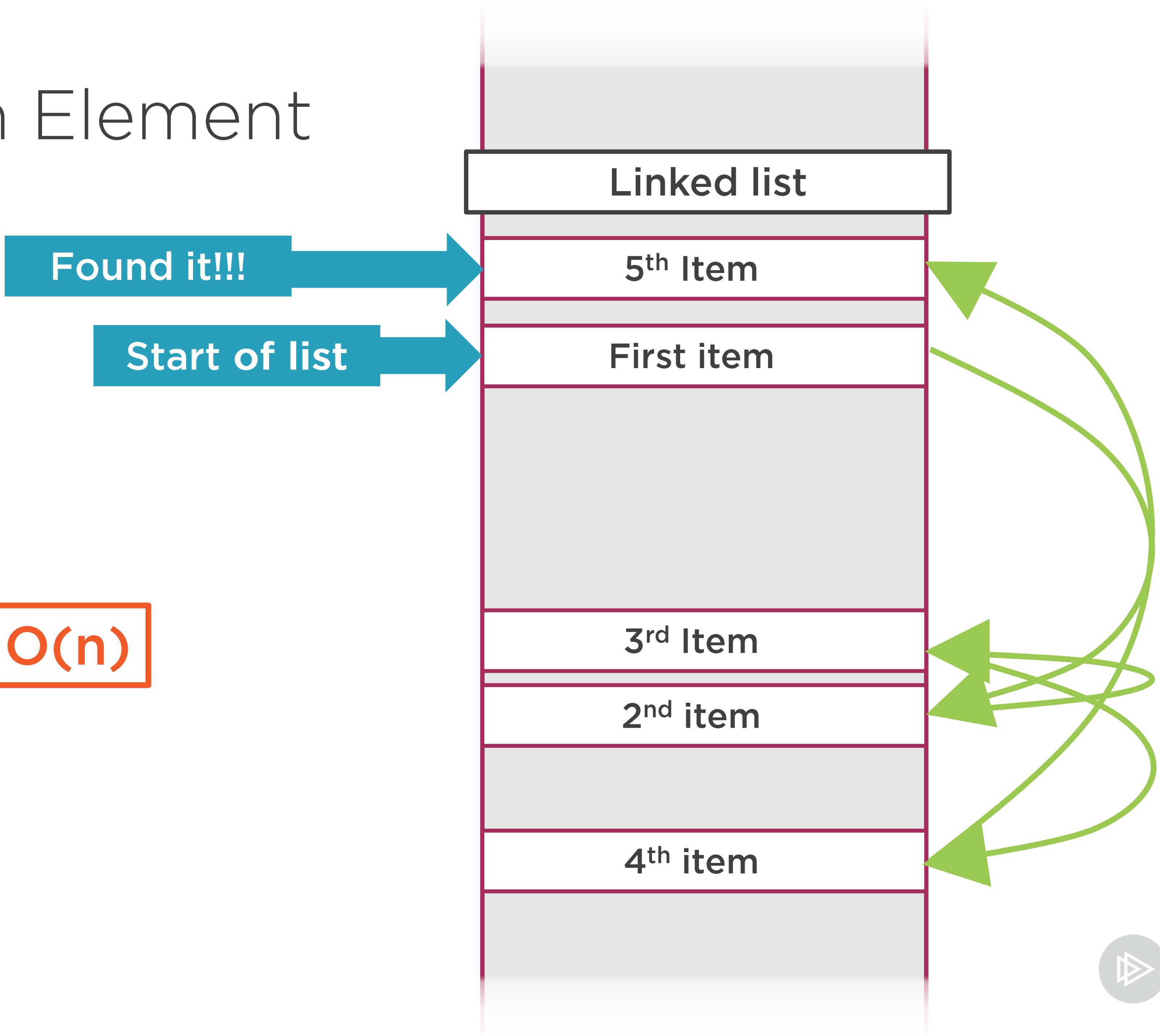
Item 6

etc.

Computer can get to any element with a single calculation

# Looking up an Element

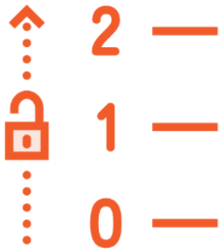
To get 5th element...



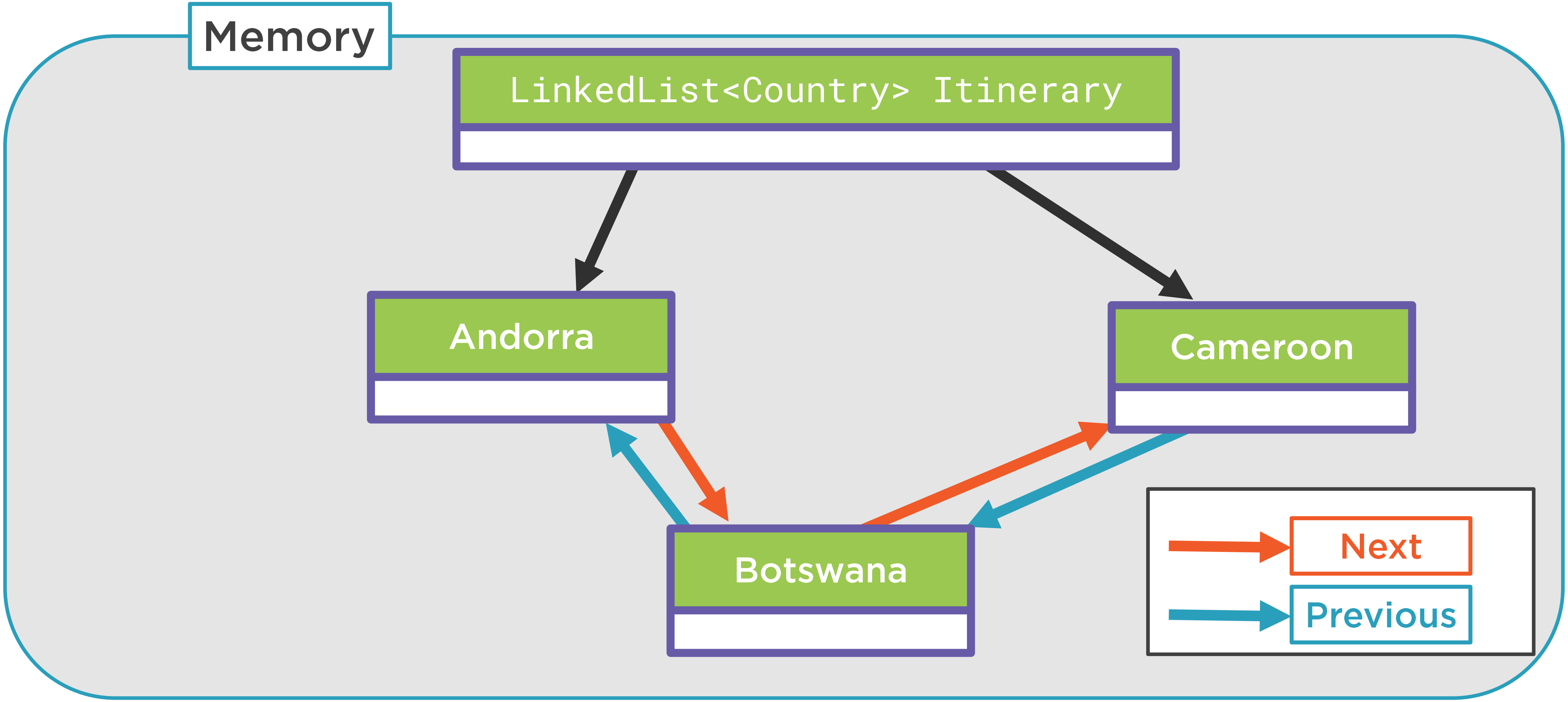
This will scale as O(n)

# Linked Lists vs Lists

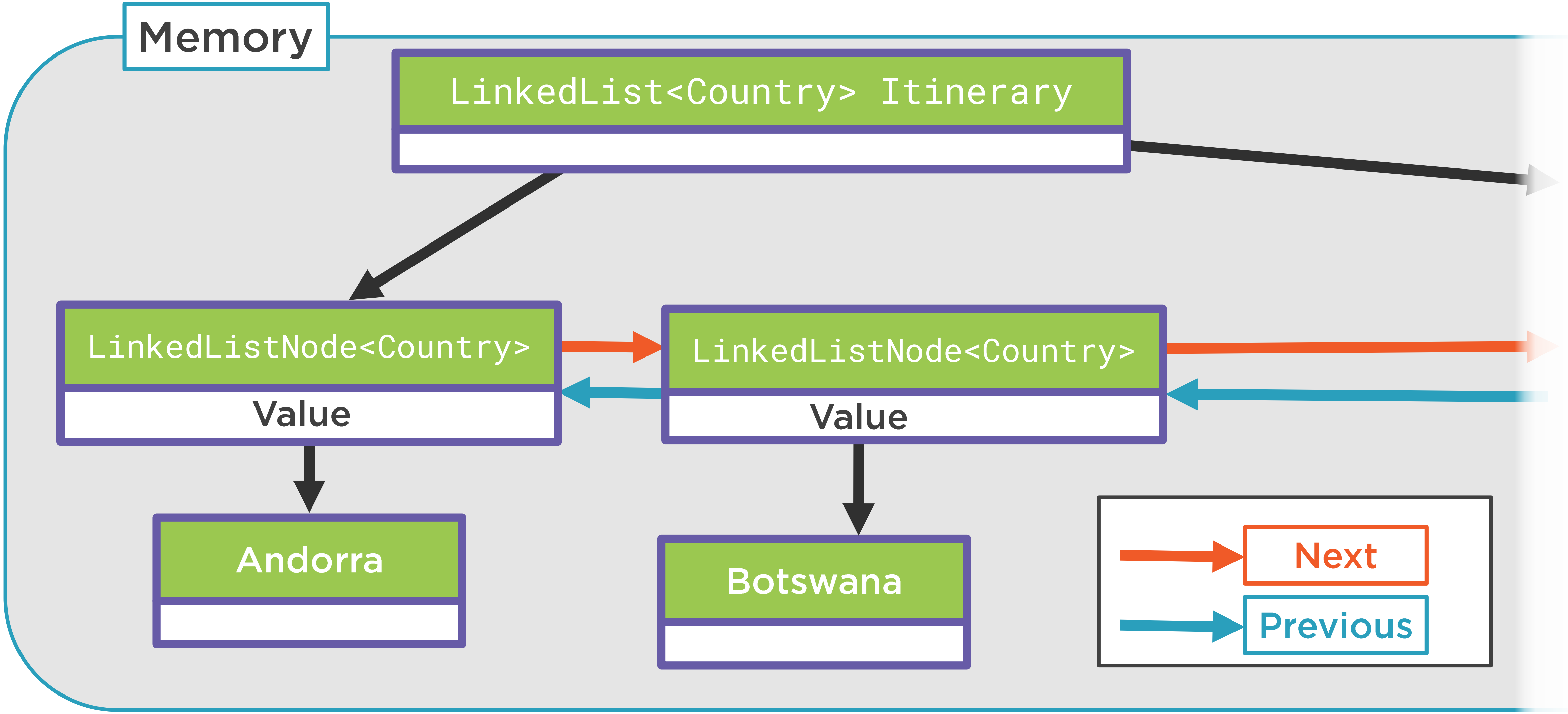
| LinkedList<T> | List<T> |
|---|---|
| ✗ Slow lookup | Fast lookup ✓ |
| ✓ Fast changes | Slow changes ✗ |

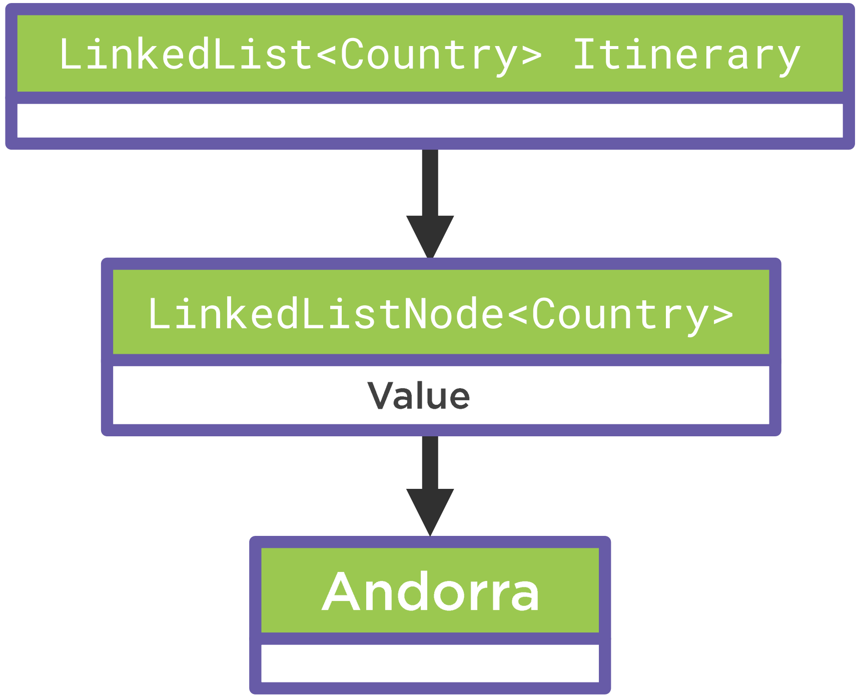


# Linked Lists



# Linked Lists





LinkedList<Country> Itinerary

LinkedListNode<Country>

Value

Andorra

# Linked Lists

LinkedListNode<T>: Wrapper to let you put stuff in linked lists

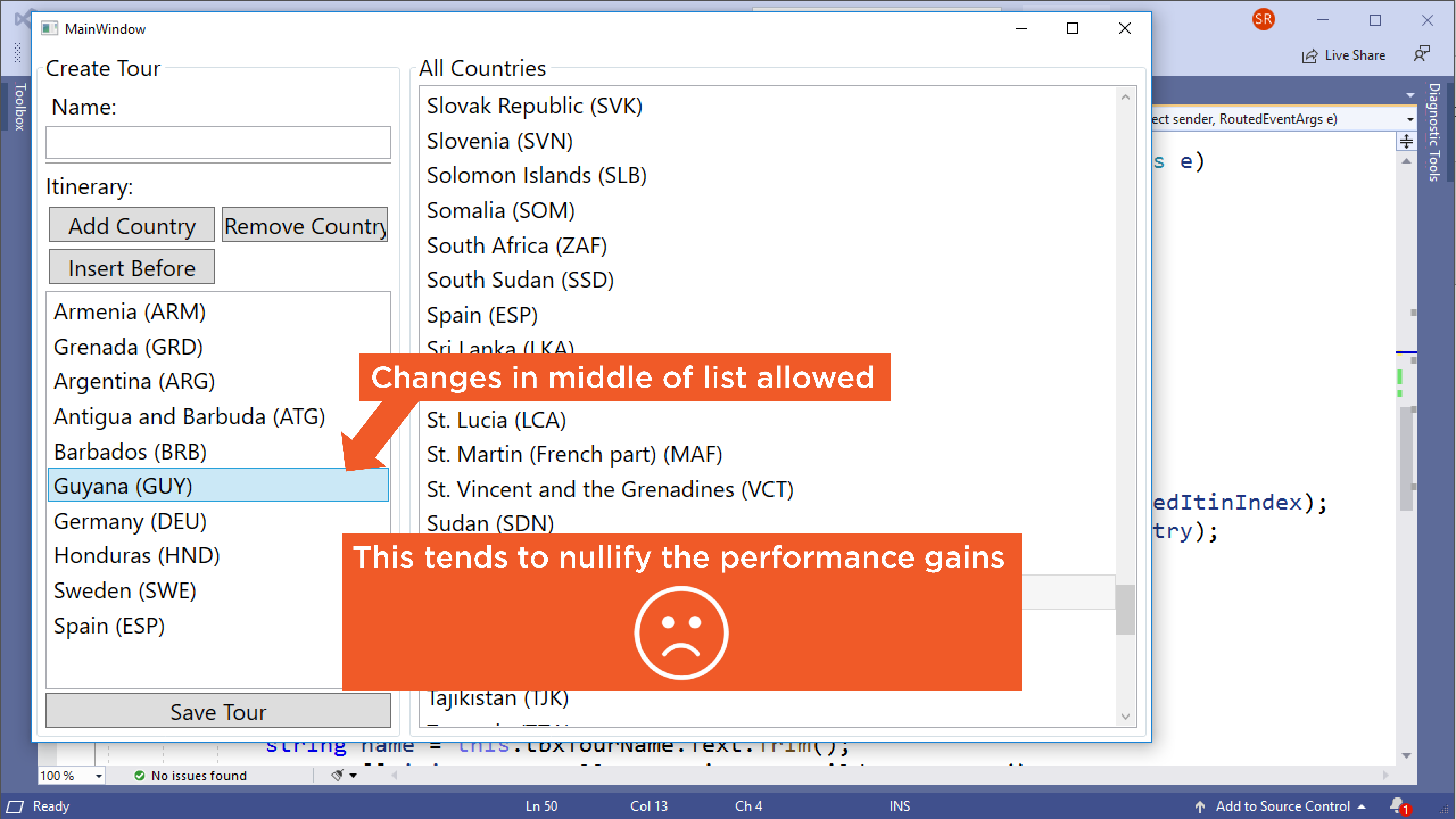
Can’t put items directly in a linked list

Must use LinkedListNode<T> as an intermediary

# Demo

**Coding with `LinkedList<T>`**

- Create a tour itinerary
- Add and remove countries



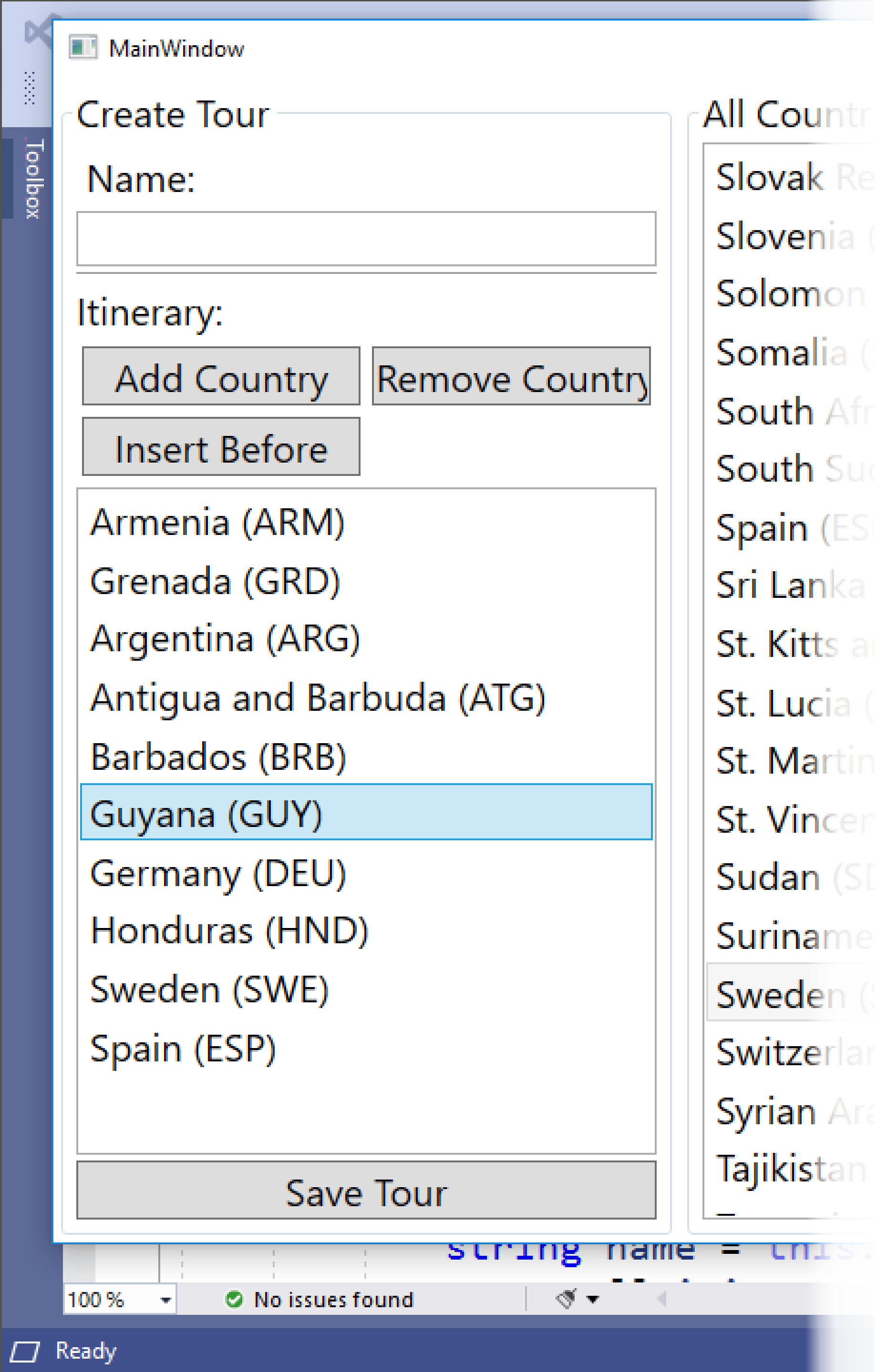
Armenia (ARM)
Grenada (GRD)
Argentina (ARG)
Antigua and Barbuda (ATG)
Barbados (BRB)
Guyana (GUY)
Germany (DEU)
Honduras (HND)
Sweden (SWE)
Spain (ESP)

Slovak Republic (SVK)
Slovenia (SVN)
Solomon Islands (SLB)
Somalia (SOM)
South Africa (ZAF)
South Sudan (SSD)
Spain (ESP)
Sri Lanka (LKA)
St. Lucia (LCA)
St. Martin (French part) (MAF)
St. Vincent and the Grenadines (VCT)
Sudan (SDN)
Tajikistan (TJK)

```
ect sender, RoutedEventArgs e)
s e)
edItinIndex);
try);
string name = this.tbxTourName.Text.Trim();
```

Changes in middle of list allowed

This tends to nullify the performance gains ☹



Armenia (ARM)
Grenada (GRD)
Argentina (ARG)
Antigua and Barbuda (ATG)
Barbados (BRB)
Guyana (GUY)
Germany (DEU)
Honduras (HND)
Sweden (SWE)
Spain (ESP)

Slovak Re
Slovenia (
Solomon
Somalia (
South Afr
South Su
Spain (ES
Sri Lanka
St. Kitts a
St. Lucia (
St. Martin
St. Vincen
Sudan (SI
Suriname
Sweden (
Switzerlar
Syrian Ara
Tajikistan

```
string name = this.
```

# Poor Use for LinkedList<T>

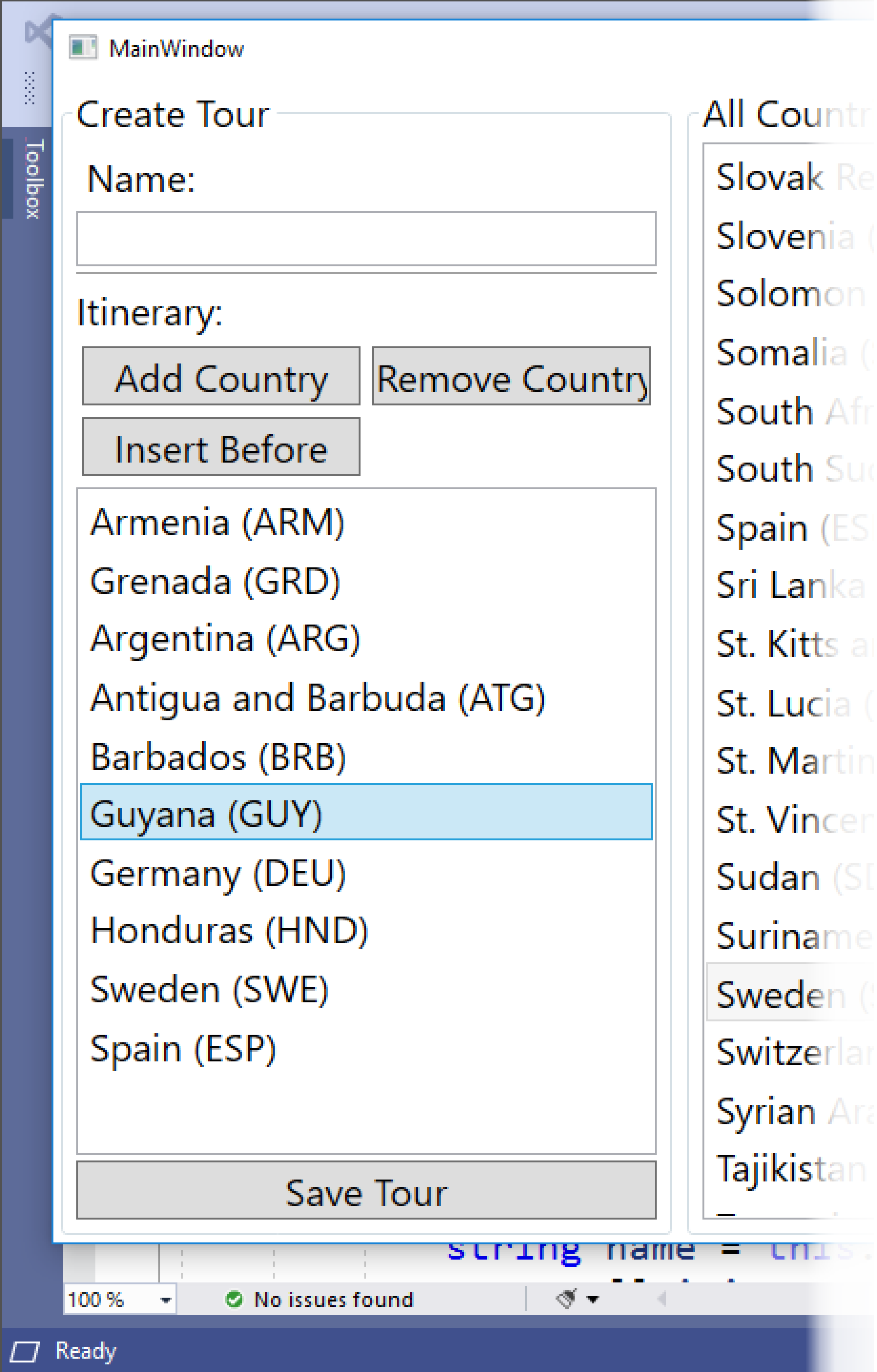
**Changes in middle of list**

**The list is too small for scalability to be important**

**Using in UI context**

- Performance dominated by UI

**So ordinary list may be simpler here**



Armenia (ARM)
Grenada (GRD)
Argentina (ARG)
Antigua and Barbuda (ATG)
Barbados (BRB)
Guyana (GUY)
Germany (DEU)
Honduras (HND)
Sweden (SWE)
Spain (ESP)

Slovak Re
Slovenia (
Solomon
Somalia (
South Afr
South Su
Spain (ES
Sri Lanka
St. Kitts a
St. Lucia (
St. Martin
St. Vincer
Sudan (SI
Suriname
Sweden (
Switzerlar
Syrian Ara
Tajikistan

# WPF Disclaimer

**In a real WPF app:**

- You’d separate UI and backend data
- WPF has `ObservableCollection<T>` to help display lists
- Listboxes often connect to `ObservableCollection<T>`

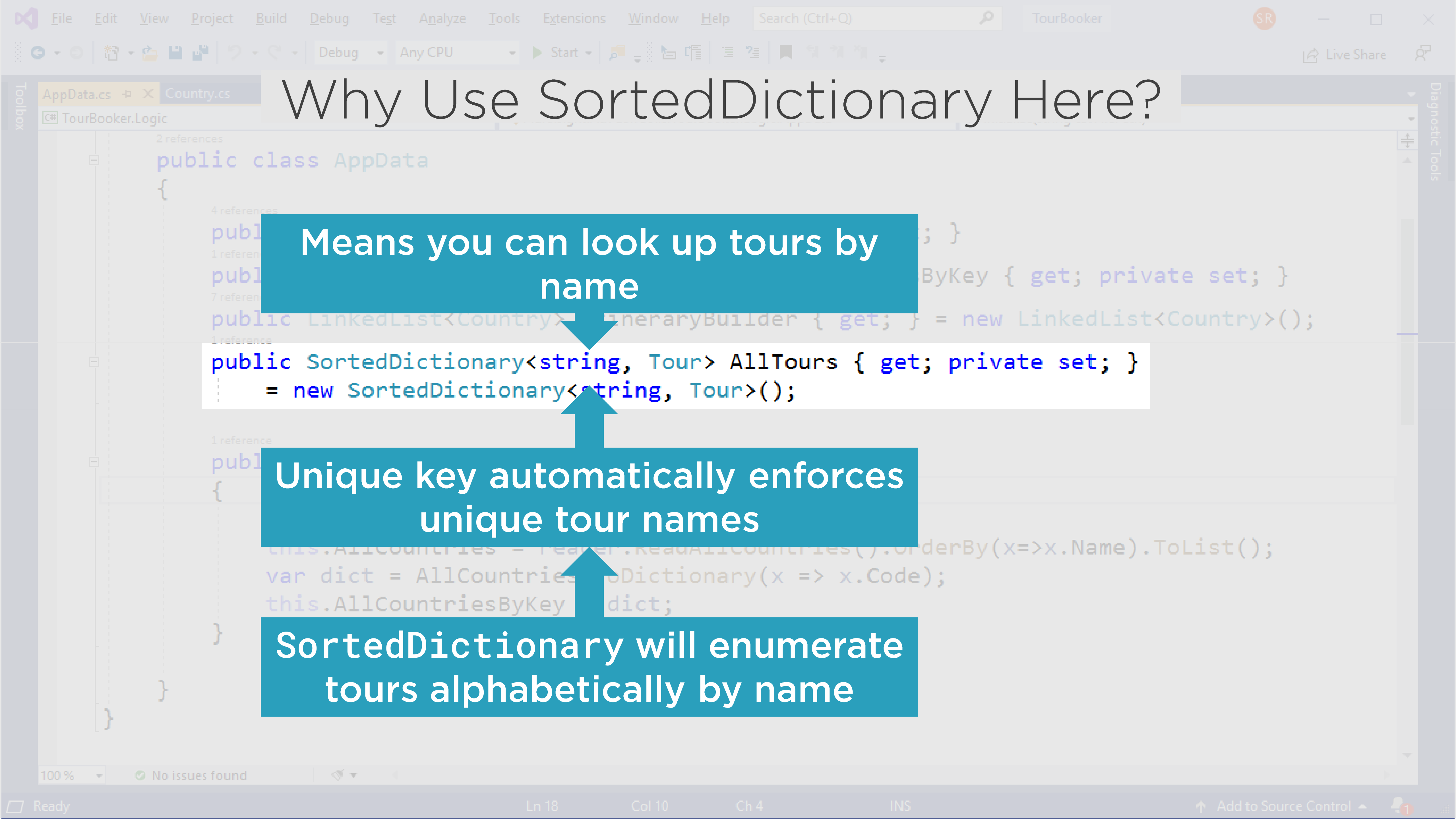
**But that’s out of scope for this course**

# Demo

**Saving a tour**

- Here, saving means, writing to another property
- No ability to persist data

# Why Use SortedDictionary Here?



```
public SortedDictionary<string, Tour> AllTours { get; private set; }
    = new SortedDictionary<string, Tour>();
```

Means you can look up tours by name

Unique key automatically enforces unique tour names

SortedDictionary will enumerate tours alphabetically by name

# Summary

**`LinkedList<T>`**

- Similar to a `List<T>` but optimised for modifications
- No direct indexed access to elements
- Have to wrap each element in `LinkedListNode<T>`
- Usually copy to another collection when done editing

**`SortedDictionary<TKey, TValue>`**

- Preserves uniqueness of keys

**Next up: Stacks**