DOCUMENTATION

DEFINITION

A linked list is a fundamental linear data structure in computer science, consisting of a sequence of elements called nodes, where each node stores data and a reference (link) to the next node in the sequence.

TYPES:

- Single linked list
- Double linked list
- Circular linked list

IMPORTANCE | BENEFITS

Efficient insertion and deletions

- Unlike arrays or List<T>, Linked list allows insertions and deletions from the beginning, middle or end (once you have a reference to the node).
- It makes ideal for scenarios where the collection size changes frequently.

Doubly link nodes

- Each node contains references to both the next and previous nodes.

No shifting needed (no resizing)

- Useful when working with large datasets or in real-time systems where performance predictability matters.

Supports non-contiguous memory (Flexible memory use)

- Nodes are allocated independently in memory, which can be useful in systems where contiguous memory allocation is a constraint.

USEFUL SCENARIO

Browser History (Going Back and Forward on Web Pages)

- When you use a web browser (like Chrome or Edge), you can click Back to go to the previous page, and forward to go to the next one.

Undo and Redo Functionality in Editors (Like MS Word or VSCode)

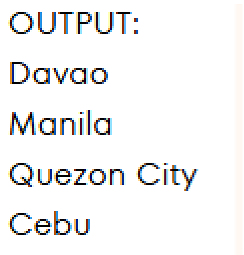
- When you type something and press Undo (Ctrl + Z), it brings back your last change. If you press Redo (Ctrl + Y), it puts it back again.

Music and Video Playlist (Like Spotify or YouTube)

- In a music or video streaming app, you can play songs or videos one after another, skip, or go back.

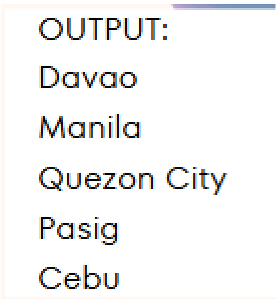
SAMPLE SYNTAX

```
class Program

{

  static void Main()

  {

    // Create a linked list of strings

    LinkedList<string> cities = new LinkedList<string>();


    // Add elements to the list

    cities.AddLast("Manila");

    cities.AddLast("Quezon City");

    cities.AddFirst("Davao");

    cities.AddLast("Cebu");


    // Display the list

    Console.WriteLine("Cities:");

    foreach (var city in cities)

    {

      Console.WriteLine(city);

    }
```



```
OUTPUT:
Davao
Manila
Quezon City
Cebu
```

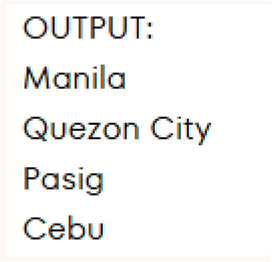
```
    // Insert after a specific node

    LinkedListNode<string> node = cities.Find("Quezon City");

    if (node != null)

    {

      cities.AddAfter(node, "Pasig");

    }
```



```
OUTPUT:
Davao
Manila
Quezon City
Pasig
Cebu
```

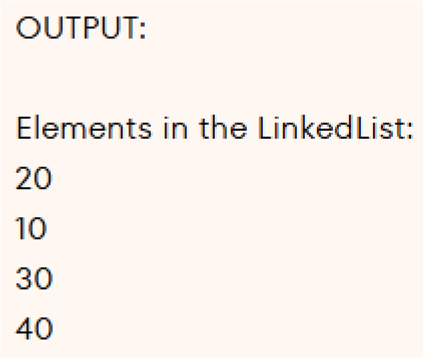
```
    // Remove an element

    cities.Remove("Davao");
```

```
        Console.WriteLine("\nUpdated Cities:");
        foreach (var city in cities)
        {
            Console.WriteLine(city);
        }
    }
}
```



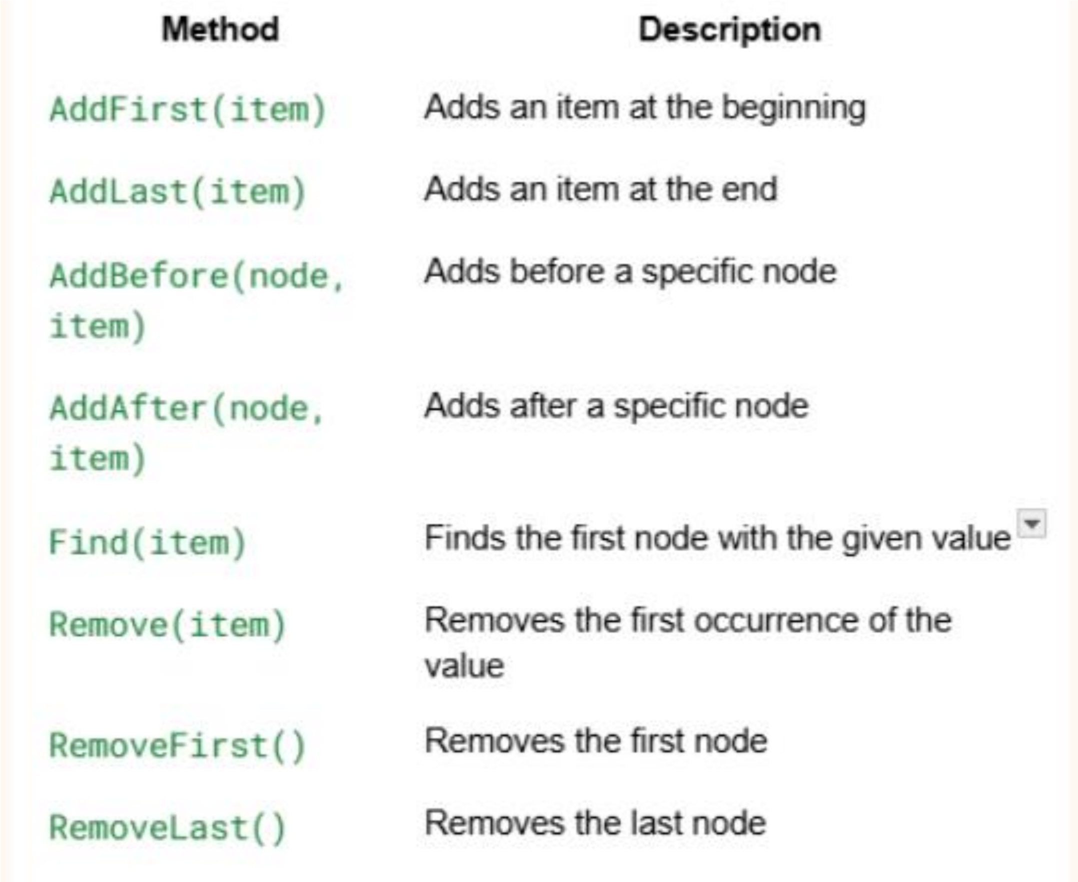
```
OUTPUT:
Manila
Quezon City
Pasig
Cebu
```

```
class Geeks {
    static void Main()
    {
        // Create a new LinkedList of strings
        LinkedList<int> l = new LinkedList<int>();
        // Adds at the end
        l.AddLast(10);
        // Adds at the beginning
        l.AddFirst(20);
        // Adds at the end
        l.AddLast(30);
        // Adds at the end
        l.AddLast(40);

        // Display the elements in the LinkedList
        Console.WriteLine("Elements in the LinkedList:");
        foreach(var i in l) {
          Console.WriteLine(i);
        }
    }
}
```



```
OUTPUT:

Elements in the LinkedList:
20
10
30
40
```

ADDITIONAL METHODS



Method	Description
`AddFirst(item)`	Adds an item at the beginning
`AddLast(item)`	Adds an item at the end
`AddBefore(node, item)`	Adds before a specific node
`AddAfter(node, item)`	Adds after a specific node
`Find(item)`	Finds the first node with the given value
`Remove(item)`	Removes the first occurrence of the value
`RemoveFirst()`	Removes the first node
`RemoveLast()`	Removes the last node

```
LinkedList<int> list = new LinkedList<int>();

list.AddLast(10);

list.AddLast(20);

list.AddLast(30);


var node = list.Find(20);

if (node != null)

{

   node.Value = 99; // Update the value of the node

}
```

REFERENCES

LinkedList<T> Class- Microsoft

GeeksforGeeks. (2025, July 11). C# LinkedList. GeeksforGeeks. https://www.geeksforgeeks.org/c-sharp/linked-list-implementation-in-c-sharp/

Dotnet-Bot. (n.d.). LinkedList Class (System.Collections.Generic). Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/api/system.collections.generic.linkedlist-1?view=net-9.0

MEMBERS & CONTRIBUTIONS

MARMITO – Provided information & references

NICOLAS – PPT, Presenter, & provided information

PAMPILON - Presenter, & provided information

RABARA - Presenter, & provided information

SILDORA - Presenter, & provided information