

**Chapter 04: Implementation of Data Structures**

HND CSD

BCAS Campus, Kalmunai

 By: Eng. A.L. Jubailah Begum

DATA STRUCTURES & ALGORITHMS

C# Stack with Examples

Let’s see how to **create** a stack using Stack() constructor:

**Step 1:**Include *System.Collections* namespace in your program with the help of using keyword.

using System.Collections;

**Step 2:**Create a stack using Stack class as shown below:

Stack stack\_name = new Stack();

**Step 3:**If you want to add elements in your stack, then use *Push()* method to add elements in your stack. As shown in the below example.

// C# program to illustrate how to

// create a stack

using System;

using System.Collections;

class DSA {

    // Main Method

    static public void Main()

    {

        // Create a stack

        // Using Stack class

        Stack my\_stack = new Stack();

        // Adding elements in the Stack

        // Using Push method

        my\_stack.Push("CSA");

        my\_stack.Push("DM");

        my\_stack.Push('DDD');

        my\_stack.Push(null);

        my\_stack.Push(1234);

        my\_stack.Push(490.98);

        // Accessing the elements

        // of my\_stack Stack

        // Using foreach loop

        foreach(var elem in my\_stack)

        {

            Console.WriteLine(elem);

        }

    }

}

Let’s see how to **remove** elements from a stack:

// C# program to illustrate how to

// remove elements from the stack

using System;

using System.Collections;

class DSA {

    // Main Method

    static public void Main()

    {

        // Create a stack

        // Using Stack class

        Stack my\_stack = new Stack();

        // Adding elements in the Stack

        // Using Push method

        my\_stack.Push("CSA");

        my\_stack.Push("DM");

        my\_stack.Push("DDD");

        my\_stack.Push("QSIT");

        Console.WriteLine("Total elements present in"+

                    " my\_stack: {0}", my\_stack.Count);

        my\_stack.Pop();

        // After Pop method

        Console.WriteLine("Total elements present in "+

                      "my\_stack: {0}", my\_stack.Count);

        // Remove all the elements

        // from the stack

        my\_stack.Clear();

        // After Pop method

        Console.WriteLine("Total elements present in "+

                      "my\_stack: {0}", my\_stack.Count);

    }

}

Let’s see how to get the **topmost** element of a stack:

// C# program to illustrate how to

// get topmost elements of the stack

using System;

using System.Collections;

class DSA {

    // Main Method

    static public void Main()

    {

        // Create a stack

        // Using Stack class

        Stack my\_stack = new Stack();

        // Adding elements in the Stack

        // Using Push method

        my\_stack.Push("CSA");

        my\_stack.Push("DM");

        my\_stack.Push("DDD");

        my\_stack.Push("QSIT");

        Console.WriteLine("Total elements present in"+

                     " my\_stack: {0}",my\_stack.Count);

        // Obtain the topmost element

        // of my\_stack Using Pop method

        Console.WriteLine("Topmost element of my\_stack"

                          + " is: {0}",my\_stack.Pop());

        Console.WriteLine("Total elements present in"+

                    " my\_stack: {0}", my\_stack.Count);

        // Obtain the topmost element

        // of my\_stack Using Peek method

        Console.WriteLine("Topmost element of my\_stack "+

                              "is: {0}",my\_stack.Peek());

        Console.WriteLine("Total elements present "+

                 "in my\_stack: {0}",my\_stack.Count);

    }

}

Let’s see how to check the **availability** of elements in a stack:

// C# program to illustrate how

// to check element present in

// the stack or not

using System;

using System.Collections;

class DSA {

    // Main  Method

    static public void Main()

    {

        // Create a stack

        // Using Stack class

        Stack my\_stack = new Stack();

        // Adding elements in the Stack

        // Using Push method

        my\_stack.Push("CSA");

        my\_stack.Push("DM");

        my\_stack.Push("DDD");

        my\_stack.Push("QSIT");

        // Checking if the element is

        // present in the Stack or not

        if (my\_stack.Contains("QSIT") == true)

        {

            Console.WriteLine("Element is found...!!");

        }

        else

        {

            Console.WriteLine("Element is not found...!!");

        }

    }

}

# C# Queue with Examples

Let’s see how to **create** a Queue using Queue() constructor:

**Step 1:**Include *System.Collections* namespace in your program with the help of using keyword.

**Syntax:**

using System.Collections;

**Step 2:**Create an queue using Queue class as shown below:

Queue queue\_name = new Queue();

**Step 3:**If you want to add elements in your queue then use *[Enqueue()](https://www.geeksforgeeks.org/queue-enqueue-method-in-c-sharp/" \t "_blank)* method to add elements in your queue. As shown in the below example.

// C# program to illustrate queue

using System;

using System.Collections;

public class DSA {

    static public void Main()

    {

        // Create a queue

        // Using Queue class

        Queue my\_queue = new Queue();

        // Adding elements in Queue

        // Using Enqueue() method

        my\_queue.Enqueue("DSA");

        my\_queue.Enqueue(1);

        my\_queue.Enqueue(100);

        my\_queue.Enqueue(null);

        my\_queue.Enqueue(2.4);

        my\_queue.Enqueue("DDD");

        // Accessing the elements

        // of my\_queue Queue

        // Using foreach loop

        foreach(var ele in my\_queue)

        {

            Console.WriteLine(ele);

        }

    }

}

Let’s see how to **remove** elements from a Queue:

// C# program to illustrate how

// to remove elements from queue

using System;

using System.Collections;

public class DSA {

    static public void Main()

    {

        // Create a queue

        // Using Queue class

        Queue my\_queue = new Queue();

        // Adding elements in Queue

        // Using Enqueue() method

        my\_queue.Enqueue("DSA");

        my\_queue.Enqueue(1);

        my\_queue.Enqueue(100);

        my\_queue.Enqueue(2.4);

        my\_queue.Enqueue("DDD");

        Console.WriteLine("Total elements present in my\_queue: {0}",

                                                my\_queue.Count);

        my\_queue.Dequeue();

        // After Dequeue method

        Console.WriteLine("Total elements present in my\_queue: {0}",

                                                my\_queue.Count);

        // Remove all the elements from the queue

        my\_queue.Clear();

        // After Clear method

        Console.WriteLine("Total elements present in my\_queue: {0}",

                                                my\_queue.Count);

    }

}

Let’s see how to **get topmost** element from a Queue:

// C# program to illustrate how

// get topmost elements of the queue

using System;

using System.Collections;

public class DSA {

    static public void Main()

    {

        // Create a queue

        // Using Queue class

        Queue my\_queue = new Queue();

        // Adding elements in Queue

        // Using Enqueue() method

        my\_queue.Enqueue("CSA");

        my\_queue.Enqueue("DM");

        my\_queue.Enqueue("DF");

        my\_queue.Enqueue("DDD");

        my\_queue.Enqueue("QSIT");

        Console.WriteLine("Total elements present in my\_queue: {0}",

                                                my\_queue.Count);

        // Obtain the topmost element of my\_queue

        // Using Dequeue method

        Console.WriteLine("Topmost element of my\_queue"

                     + " is: {0}", my\_queue.Dequeue());

        Console.WriteLine("Total elements present in my\_queue: {0}",

                                                my\_queue.Count);

        // Obtain the topmost element of my\_queue

        // Using Peek method

        Console.WriteLine("Topmost element of my\_queue is: {0}",

                                               my\_queue.Peek());

        Console.WriteLine("Total elements present in my\_queue: {0}",

                                                my\_queue.Count);

    }}

Let’s see how to check the **availability** of an element in a Queue:

// C# program to illustrate how

// to check element present in

// the queue or not

using System;

using System.Collections;

class DSA {

    static public void Main()

    {

        // Create a queue

        // Using Queue class

        Queue my\_queue = new Queue();

        // Adding elements in Queue

        // Using Enqueue() method

        my\_queue.Enqueue("CSA");

        my\_queue.Enqueue("DM");

        my\_queue.Enqueue("DF");

        my\_queue.Enqueue("DDD");

        my\_queue.Enqueue("QSIT");

        // Checking if the element is

        // present in the Queue or not

        if (my\_queue.Contains("DF") == true) {

            Console.WriteLine("Element available...!!");

        }

        else {

            Console.WriteLine("Element not available...!!");

        }

    }

}

# C# Linked List with Examples

**First Simple Linked List in C#**

Let us **create** a simple linked list with 3 nodes.

// A simple C# program to introduce a linked list

using System;

public class LinkedList {

    Node head; // head of list

    /\* Linked list Node. This inner class is made static so that

    main() can access it \*/

    public class Node {

        public int data;

        public Node next;

        public Node(int d)

        {

            data = d;

            next = null;

        } // Constructor

    }

    /\* method to create a simple linked list with 3 nodes\*/

    public static void Main(String[] args)

    {

        /\* Start with the empty list. \*/

        LinkedList llist = new LinkedList();

        llist.head = new Node(1);

        Node second = new Node(2);

        Node third = new Node(3);

        /\* Three nodes have been allocated dynamically.

        We have references to these three blocks as head,

        second and third

        llist.head     second             third

            |             |                 |

            |             |                 |

        +----+------+     +----+------+     +----+------+

        | 1 | null |     | 2 | null |     | 3 | null |

        +----+------+     +----+------+     +----+------+ \*/

        llist.head.next = second; // Link first node with the second node

        /\* Now next of first Node refers to second. So they

            both are linked.

        llist.head     second             third

            |             |                 |

            |             |                 |

        +----+------+     +----+------+     +----+------+

        | 1 | o-------->| 2 | null |     | 3 | null |

        +----+------+     +----+------+     +----+------+ \*/

        second.next = third; // Link second node with the third node

        /\* Now next of the second Node refers to third. So all three

            nodes are linked.

        llist.head     second             third

            |             |                 |

            |             |                 |

        +----+------+     +----+------+     +----+------+

        | 1 | o-------->| 2 | o-------->| 3 | null |

        +----+------+     +----+------+     +----+------+ \*/

    }

}

**Linked List Traversal**

// A simple C# program for traversal of a linked list

using System;

public class LinkedList {

    Node head; // head of list

    /\* Linked list Node. This inner

    class is made static so that

    main() can access it \*/

public class Node {

        public int data;

        public Node next;

        public Node(int d)

        {

            data = d;

            next = null;

        } // Constructor

    }

    /\* This function prints contents of

    linked list starting from head \*/

    public void printList()

    {

        Node n = head;

        while (n != null) {

            Console.Write(n.data + " ");

            n = n.next;

        }

    }

    /\* method to create a simple linked list with 3 nodes\*/

    public static void Main(String[] args)

    {

        /\* Start with the empty list. \*/

        LinkedList llist = new LinkedList();

        llist.head = new Node(1);

        Node second = new Node(2);

        Node third = new Node(3);

        llist.head.next = second; // Link first node with the second node

        second.next = third; // Link first node with the second node

        llist.printList();

    }

}

**Inserting a node in a linked list**

In this part, methods to insert a new node in linked list are discussed. A node can be added in three ways  
**1)** At the front of the linked list  
**2)**After a given node.  
**3)** At the end of the linked list.

// A complete working C# program to demonstrate

// all insertion methods on linked list

using System;

class DSA

{

    public Node head; // head of list

    /\* Linked list Node\*/

    public class Node

    {

        public int data;

        public Node next;

        public Node(int d) {data = d; next = null;}

    }

    /\* Inserts a new Node at front of the list. \*/

    public void push(int new\_data)

    {

        /\* 1 & 2: Allocate the Node &

                Put in the data\*/

        Node new\_node = new Node(new\_data);

        /\* 3. Make next of new Node as head \*/

        new\_node.next = head;

        /\* 4. Move the head to point to new Node \*/

        head = new\_node;

    }

    /\* Inserts a new node after the given prev\_node. \*/

    public void insertAfter(Node prev\_node, int new\_data)

    {

        /\* 1. Check if the given Node is null \*/

        if (prev\_node == null)

        {

            Console.WriteLine("The given previous" +

                              " node cannot be null");

            return;

        }

        /\* 2 & 3: Allocate the Node &

                Put in the data\*/

        Node new\_node = new Node(new\_data);

        /\* 4. Make next of new Node as

              next of prev\_node \*/

        new\_node.next = prev\_node.next;

        /\* 5. make next of prev\_node as new\_node \*/

        prev\_node.next = new\_node;

    }

    /\* Appends a new node at the end. This method is

    defined inside LinkedList class shown above \*/

    public void append(int new\_data)

    {

        /\*

1. Allocate the Node &

        2. Put in the data

        3. Set next as null \*/

        Node new\_node = new Node(new\_data);

        /\* 4. If the Linked List is empty,

        then make the new node as head \*/

        if (head == null)

        {

            head = new Node(new\_data);

            return;

        }

        /\* 4. This new node is going to be the last node,

            so make next of it as null \*/

new\_node.next = null;

        /\* 5. Else traverse till the last node \*/

        Node last = head;

        while (last.next != null)

            last = last.next;

        /\* 6. Change the next of last node \*/

        last.next = new\_node;

        return;

    }

    /\* This function prints contents of linked list

    starting from the given node \*/

    public void printList()

    {

        Node tnode = head;

        while (tnode != null)

        {

            Console.Write(tnode.data + " ");

            tnode = tnode.next;

        }

    }

    // Driver Code

    public static void Main(String[] args)

    {

        /\* Start with the empty list \*/

        DSA llist = new DSA();

        // Insert 6. So linked list becomes 6->NUllist

        llist.append(6);

        // Insert 7 at the beginning.

        // So linked list becomes 7->6->NUllist

        llist.push(7);

        // Insert 1 at the beginning.

        // So linked list becomes 1->7->6->NUllist

        llist.push(1);

        // Insert 4 at the end. So linked list becomes

        // 1->7->6->4->NUllist

        llist.append(4);

        // Insert 8, after 7. So linked list becomes

        // 1->7->8->6->4->NUllist

        llist.insertAfter(llist.head.next, 8);

        Console.Write("Created Linked list is: ");

        llist.printList();

    }

}

# C# Binary Tree with Examples

// A C# program to introduce Binary Tree

using System;

/\* Class containing left and right child

of current node and key value\*/

public class Node

{

    public int key;

    public Node left, right;

    public Node(int item)

    {

        key = item;

        left = right = null;

    }

}

public class BinaryTree

{

    // Root of Binary Tree

    Node root;

    // Constructors

    BinaryTree(int key)

    {

        root = new Node(key);

    }

    BinaryTree()

    {

        root = null;

    }

    // Driver Code

    public static void Main(String[] args)

    {

        BinaryTree tree = new BinaryTree();

        /\*create root\*/

        tree.root = new Node(1);

        /\* following is the tree after above statement

             1

            / \

         null null     \*/

        tree.root.left = new Node(2);

        tree.root.right = new Node(3);

        /\* 2 and 3 become left and right children of 1

             1

            / \

         2    3

         / \ / \

        null null null null \*/

        tree.root.left.left = new Node(4);

        /\* 4 becomes left child of 2

                     1

                 /  \

             2  3

             / \   / \

             4 null null null

         / \

         null null

        \*/

    }

}

# Insertion in a Binary Tree in level order

// C# program to insert element in binary tree

using System;

using System.Collections.Generic;

class DSA

{

    /\* A binary tree node has key, pointer to

    left child and a pointer to right child \*/

    public class Node

    {

        public int key;

        public Node left, right;

        // constructor

        public Node(int key)

        {

            this.key = key;

            left = null;

            right = null;

        }

    }

    static Node root;

    static Node temp = root;

    /\* Inorder traversal of a binary tree\*/

    static void inorder(Node temp)

    {

        if (temp == null)

            return;

        inorder(temp.left);

        Console.Write(temp.key + " ");

        inorder(temp.right);

    }

    /\*function to insert element in binary tree \*/

    static void insert(Node temp, int key)

    {

        Queue<Node> q = new Queue<Node>();

        q.Enqueue(temp);

        // Do level order traversal until we find

        // an empty place.

        while (q.Count != 0)

        {

            temp = q.Peek();

            q.Dequeue();

            if (temp.left == null)

            {

                temp.left = new Node(key);

                break;

            } else

                q.Enqueue(temp.left);

            if (temp.right == null)

            {

                temp.right = new Node(key);

                break;

            } else

                q.Enqueue(temp.right);

        }

    }

    // Driver code

    public static void Main(String []args)

    {

        root = new Node(10);

        root.left = new Node(11);

        root.left.left = new Node(7);

        root.right = new Node(9);

        root.right.left = new Node(15);

        root.right.right = new Node(8);

        Console.Write("Inorder traversal before insertion:");

        inorder(root);

        int key = 12;

        insert(root, key);

        Console.Write("\nInorder traversal after insertion:");

        inorder(root);

    }

}

# Tree Traversals (Inorder, Preorder and Postorder)

// C# program for different

// tree traversals

using System;

/\* Class containing left and

right child of current

node and key value\*/

class Node

{

    public int key;

    public Node left, right;

    public Node(int item)

    {

        key = item;

        left = right = null;

    }

}

class BinaryTree

{

// Root of Binary Tree

Node root;

BinaryTree()

{

    root = null;

}

/\* Given a binary tree, print

   its nodes according to the

   "bottom-up" postorder traversal. \*/

void printPostorder(Node node)

{

    if (node == null)

        return;

    // first recur on left subtree

    printPostorder(node.left);

    // then recur on right subtree

    printPostorder(node.right);

    // now deal with the node

    Console.Write(node.key + " ");

}

/\* Given a binary tree, print

   its nodes in inorder\*/

void printInorder(Node node)

{

    if (node == null)

        return;

    /\* first recur on left child \*/

    printInorder(node.left);

    /\* then print the data of node \*/

    Console.Write(node.key + " ");

    /\* now recur on right child \*/

    printInorder(node.right);

}

/\* Given a binary tree, print

   its nodes in preorder\*/

void printPreorder(Node node)

{

    if (node == null)

        return;

    /\* first print data of node \*/

    Console.Write(node.key + " ");

    /\* then recur on left sutree \*/

    printPreorder(node.left);

    /\* now recur on right subtree \*/

    printPreorder(node.right);

}

// Wrappers over above recursive functions

void printPostorder() {printPostorder(root);}

void printInorder()  {printInorder(root);}

void printPreorder() {printPreorder(root);}

// Driver Code

static public void Main(String []args)

{

    BinaryTree tree = new BinaryTree();

    tree.root = new Node(1);

    tree.root.left = new Node(2);

    tree.root.right = new Node(3);

    tree.root.left.left = new Node(4);

    tree.root.left.right = new Node(5);

    Console.WriteLine("Preorder traversal " +

                       "of binary tree is ");

    tree.printPreorder();

    Console.WriteLine("\nInorder traversal " +

                        "of binary tree is ");

    tree.printInorder();

    Console.WriteLine("\nPostorder traversal " +

                          "of binary tree is ");

    tree.printPostorder();

}

}

**Output:**

Preorder traversal of binary tree is

1 2 4 5 3

Inorder traversal of binary tree is

4 2 5 1 3

Postorder traversal of binary tree is

4 5 2 3 1