K.R. MANGALAM UNIVERSITY
THE COMPLETE WORLD OF EDUCATION

Course: Data Structure

(Course Code : ENCS205)

UNIT-1: Foundations of Data Structures

School of Engineering & Technology
K.R. Mangalam University

SESSION 2:
Abstract Data Types

Recapitulation (Previous Session)

Quiz

Q1: Which data structure follows a FIFO approach?

Q2: Is an array a primitive data structure or non-primitive data structure?

Q3: Give two examples of non-primitive data structures.

Q4: What are the different applications where we use graphs?

Abstract Data Types

Abstract Data Type

- An ADT is composed of – a collection of data and a set of operations on that data.

ADT = Data + Operations

- Specifications of an ADT indicate – What the ADT operations do, not how to implement them.
- Implementation of an ADT- includes choosing a particular data structure.

Need of Abstract Data Types (ADT)

Typical operations on data:

- Add data to a data collection
- Remove data from a data collection
- Ask questions about the data in a data collection

Data Abstraction:

- Allows you to develop each data structure in relative isolation from the rest of the solution.

Abstract Data Types (Cont..)

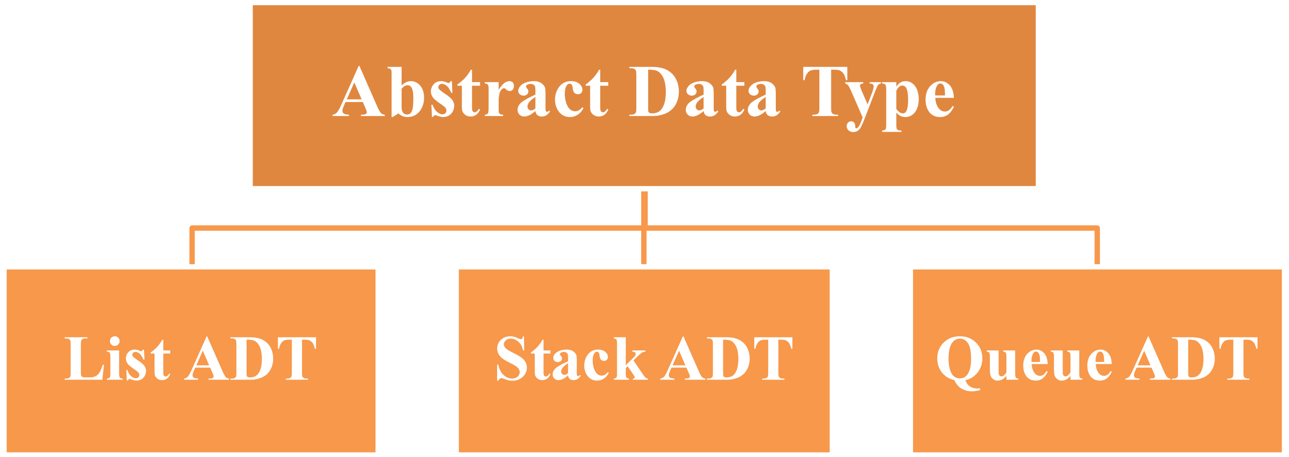

Fig.1 : Types of ADT

List ADT

- Sequential Access, Dynamic size, ordered collection
- Random Access, support for insertion and deletion,
- Easy Traversal

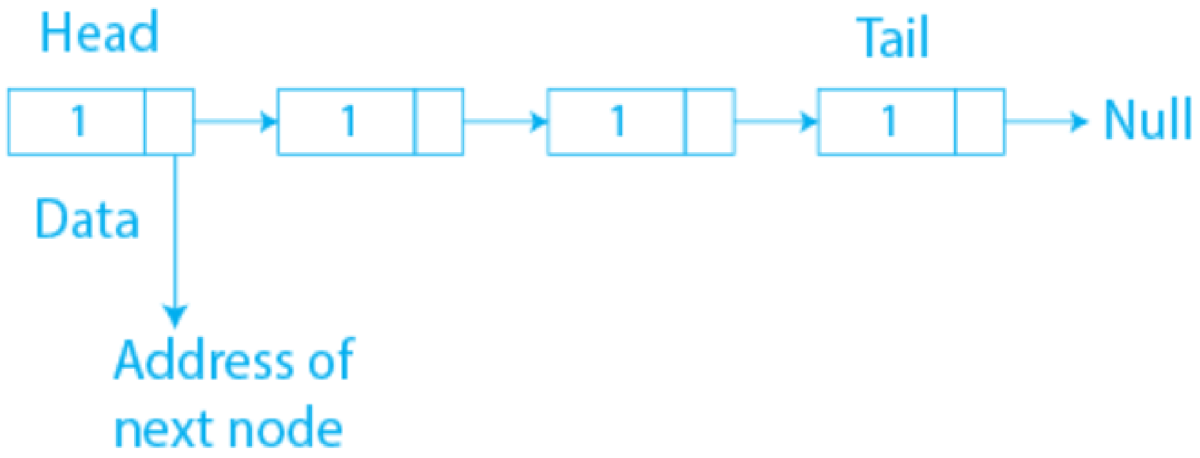

Fig.2 : Representation of List

List ADT (Cont..)

Operations defined on List ADT:

- **front():** returns the value of the node present at the front of the list.
- **back():** returns the value of the node present at the back of the list.
- **pop_front():** removes the front node from the list.
- **pop_back():** removes the last node from the list
- **empty():** returns true if the list is empty, otherwise returns false.
- **size():** returns the number of nodes that are present in the list.

Stack ADT

- Follows LIFO Principle

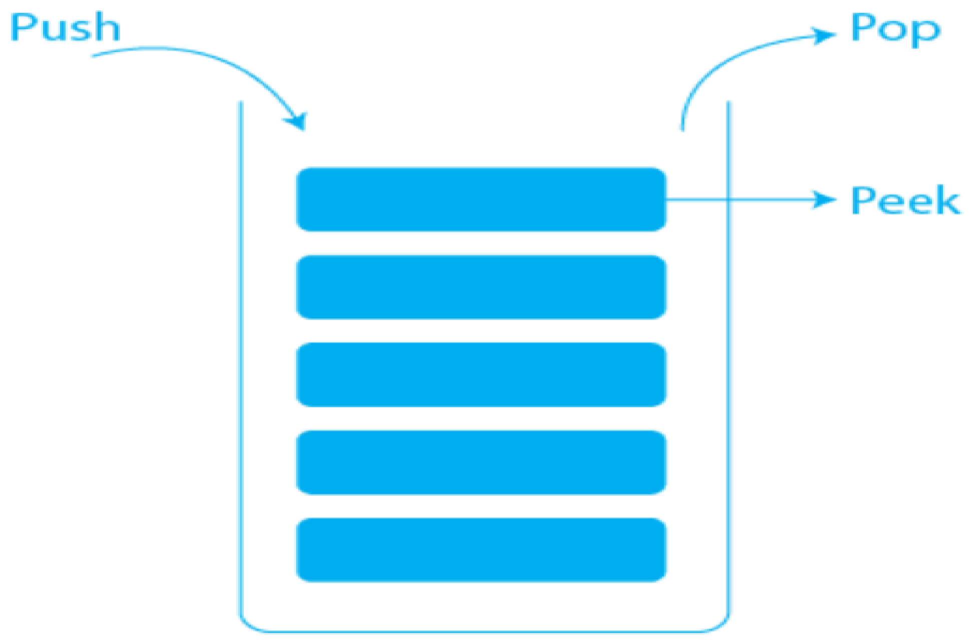

Fig.3: Representation of Stack

Stack ADT (Cont..)

Operations defined on Stack ADT:

- **top():** returns the value of the node present at the top of the stack.
- **push(int val):** creates a node with value = val and puts it at the stack top.
- **pop():** removes the node from the top of the stack.
- **empty():** returns true if the stack is empty, otherwise returns false.
- **size():** returns the number of nodes that are present in the stack.

Queue ADT

- Follows FIFO Principle

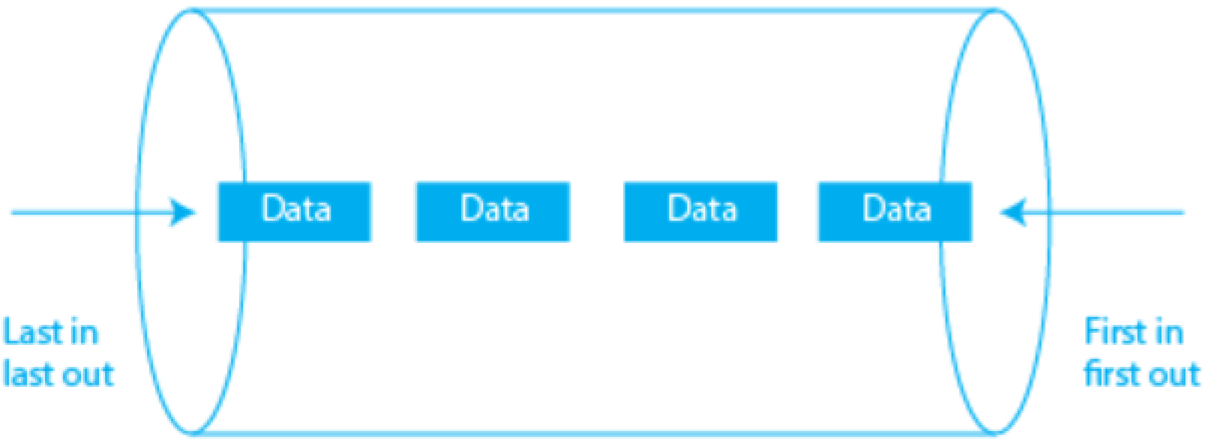

Fig.4: Representation of Queue

Queue ADT (Cont..)

Operations defined on Queue ADT:

- **front():** returns the value of the node present at the front of the queue.
- **back():** returns the value of the node present at the back of the queue.
- **push(int val):** creates a node with value = val and puts it at the front of the queue.
- **pop():** removes the node from the rear of the queue.
- **empty():** returns true if the queue is empty, otherwise returns false.
- **size():** returns the number of nodes that are present in the queue.

Advantages of ADT in Data Structures

- Provides abstraction
- Enhances program modularity
- Enables code reusability
- Promotes concept of data hiding
- Supports polymorphism

Disadvantages of ADT in Data Structures

- **Overhead**- additional overhead due to abstraction and encapsulation.

- **Limited Control-** limited control of programmer on data structure.

- **Performance Impact-** performance lower than custom data structure.

Situation- Based Questions

Q1 You are designing a music playlist application. How would you use a List ADT to manage the songs in a playlist? What operations would you need to support?

Q2 You are developing a text editor application. Explain how you would use a Stack ADT to implement the undo functionality for text edits. What operations would be involved?

Situation- Based Questions

Using List ADT for a Music Playlist:

- **Order Maintenance:** A list maintains the order of songs as added by the user, which is crucial for playlist functionality.
- **Dynamic Sizing:** Lists can expand or shrink dynamically, accommodating any number of songs as the user wants to add or remove.

Key Operations:

- **Add Song (Insert):** Add a song to the end of the playlist or at a specific position.
- **Remove Song (Delete):** Remove a song from the playlist. This could be based on the song name or index.
- **Play Next (Access):** Access the next song to play. This is typically the first song in the list if playing sequentially.
- **Search for Song:** Find a song in the playlist to check if it already exists or to find its index.
- **Shuffle:** Randomize the order of songs in the playlist. This can be implemented by generating random indices and swapping elements.
- **Sort:** Sort the songs in the playlist, perhaps by title, artist, or date added.

Situation- Based Questions

Q2 You are developing a text editor application. Explain how you would use a Stack ADT to implement the undo functionality for text edits. What operations would be involved?

Situation- Based Questions

In a text editor application, a Stack Abstract Data Type (ADT) is ideal for implementing the undo functionality due to its Last In, First Out (LIFO) nature.

Action Recording: Each action a user takes that modifies the text (like inserting, deleting, or formatting text) is recorded as an "edit action" and pushed onto the stack.
Reversibility: When the undo command is issued, the stack is popped to remove the most recent action, and the action is reversed in the text editor.

Key Operations:
1.Push (Record Action): Whenever a user makes a change to the text, that change is encapsulated in an action object with all necessary details to reverse the change (like the type of change, the text involved, and its position) and pushed onto the stack.
2.Pop (Undo Action): When the user wants to undo the last action, the stack is popped to retrieve the most recent action. The action details are then used to reverse the change in the document.
3.Peek (Check Last Action): Optionally, you can peek at the top of the stack to display or log what the last action was without removing it, which can be useful for showing undo previews.

Situation- Based Questions-do it yourself

Q3 In a manufacturing plant, items are produced and need to be stored temporarily before packaging. Describe how you would use a Queue ADT to manage the items waiting for packaging. What operations would be crucial in this scenario?

Q4 You are building a web server to handle incoming requests. Explain how you would use a Queue ADT to manage the request queue. What operations would be essential for handling requests efficiently?

THANK YOU