Stack and Queue

Course: Introduction to Programming and Data Structures

Dr. Laltu Sardar

Institute for Advancing Intelligence (IAI),
TCG Centres for Research and Education in Science and Technology (TCG Crest)

1. Stack and Queue
    - Introduction
    - Stack
    - Queue
    - Stack and Queue using linked list
    - Stack Implementation Using Linked List
    - Queue Implementation Using Linked List

Stack and Queue

Introduction to Stack and Queue

- Stack and Queue are fundamental data structures in computer science.
- Both are used to store and retrieve data efficiently.
- They follow different methods of insertion and deletion.

What is a Stack?

- A Stack is a linear data structure that follows the Last In, First Out (LIFO) principle.
- The element added last is the one to be removed first.
- Examples of stack operations:
    - **Push:** Add an element to the top of the stack.
    - **Pop:** Remove the element from the top of the stack.
    - **Peek/Top:** View the element on the top without removing it.

Important Stack Operations

- **Push:** Add an element to the top of the stack.
  - Example: If the stack is [3, 7], after pushing 10, it becomes [3, 7, 10].
- **Pop:** Remove the element from the top of the stack.
  - Example: If the stack is [3, 7, 10], after popping, it becomes [3, 7].
- **Peek/Top:** View the element on the top of the stack without removing it.
  - Example: If the stack is [3, 7, 10], the top element is 10.
- **isEmpty:** Check whether the stack is empty.
  - Example: If the stack is [], then it is empty.
- **Size:** Return the number of elements in the stack.
  - Example: If the stack is [3, 7, 10], the size is 3.
- **Clear:** Remove all elements from the stack.
  - Example: If the stack is [3, 7, 10], after clearing, it becomes [].

Applications of Stack

- Function Call Management
- Undo Mechanism in Text Editors
- Parsing Expressions (e.g., in compilers)

What is a Queue?

- A Queue is a linear data structure that follows the First In, First Out (FIFO) principle.
- The element added first is the one to be removed first.
- Examples of queue operations:
  - **Enqueue:** Add an element to the end of the queue.
  - **Dequeue:** Remove the element from the front of the queue.
  - **Front:** View the element at the front without removing it.

Important Queue Operations

- **Enqueue:** Add an element to the end of the queue.
  - Example: If the queue is [3, 7], after enqueuing 10, it becomes [3, 7, 10].
- **Dequeue:** Remove the element from the front of the queue.
  - Example: If the queue is [3, 7, 10], after dequeuing, it becomes [7, 10].
- **Front:** View the element at the front of the queue without removing it.
  - Example: If the queue is [3, 7, 10], the front element is 3.
- **isEmpty:** Check whether the queue is empty.
  - Example: If the queue is [], then it is empty.
- **Size:** Return the number of elements in the queue.
  - Example: If the queue is [3, 7, 10], the size is 3.
- **Clear:** Remove all elements from the queue.
  - Example: If the queue is [3, 7, 10], after clearing, it becomes [].

Applications of Queue

- Scheduling Processes in Operating Systems
- Handling Requests in Web Servers
- Breadth-First Search in Graphs

Stack and Queue using linked list

Introduction to Linked Lists

- A Linked List is a linear data structure where elements are stored in nodes.
- Each node contains:
    - Data
    - A pointer to the next node in the list
- Types of Linked Lists:
    - Singly Linked List
    - Doubly Linked List
    - Circular Linked List

Stack Implementation Using Linked List

```
struct Node* top = NULL;
void push(int data) {
    struct Node* newNode=(struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = top;
    top = newNode;
}
int pop() {
    if (top == NULL) {  printf("Stack is empty\n"); return -1;  }
    int popped = top->data;
    struct Node* temp = top;
    top = top->next;
    free(temp);
    return popped;
}
int peek() {
    if (top == NULL) {  printf("Stack is empty\n");    return -1; }
    return top->data;
}
```

Queue Implementation Using Linked List

```c
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

struct Node *front = NULL, *rear = NULL;

void enqueue(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node
        ));
    newNode->data = data;
    newNode->next = NULL;
    if (rear == NULL) {
        front = rear = newNode;
        return;
    }
    rear->next = newNode;
    rear = newNode;
}
```

Homework

Complete the following tasks

1. Implement Stack using Linked list
2. Implement Stack using Array
3. Implement Queue using Linked list
4. Implement Queue using Array

tcg crest
Inventing Harmonious Future

Dr. Laltu Sardar
laltu.sardar@tcgcrest.org
https://laltu-sardar.github.io.