Kartheeka Repalle

192372289

CSE-AI

26/07/2024

Stack:

A Stack is a linear data structure that follows the **LIFO (Last-In-First-Out)** principle so the last element inserted is the first to be popped out. It contains only one pointer **top pointer** pointing to the topmost element of the stack

Basic Operations on Stack:

In order to make manipulations in a stack, there are certain operations provided to us.

- **push()** to insert an element into the stack
- **pop()** to remove an element from the stack
- **top()** Returns the top element of the stack.
- **isEmpty()** returns true if stack is empty else false.
- **isFull()** returns true if the stack is full else false.

Implementation of stack:

1. Array:

```
#include <stdio.h>

#define MAX_SIZE 100

int stack[MAX_SIZE];
int top = -1;

// Check if the stack is empty
int isEmpty() {
   return top == -1;
}

// Check if the stack is full
int isFull() {
   return top == MAX_SIZE - 1;
}

// Push an element onto the stack
```

```
void push(int element) {
    if (isFull()) {
        printf("Stack is full. Cannot push %d onto the stack.\n", element);
        return;
    }
    stack[++top] = element;
    printf("%d pushed onto the stack.\n", element);
}

// Pop an element from the stack
int pop() {
    if (isEmpty()) {
        printf("Stack is empty. Cannot pop from the stack.\n");
        return -1; // Return a sentinel value to indicate failure
    }
    return stack[top--];
}

// Get the top element of the stack without removing it
int peek() {
    if (isEmpty()) {
        printf("Stack is empty. Cannot peek the stack.\n");
        return -1; // Return a sentinel value to indicate failure
    }
    return stack[top];
}

int main() {
    push(10);
    push(20);
    push(30);
    printf("Top element: %d\n", peek());
    printf("Popped element: %d\n", pop());
    printf("Top element: %d\n", peek());
    return 0;
}
```

OUTPUT:

```
10 pushed onto the stack.
20 pushed onto the stack.
30 pushed onto the stack.
Top element: 30
Popped element: 30
Top element: 20
```

2. Linked list:

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

// Define the structure for a linked list node
typedef struct Node {
    int data;
    struct Node* next;
} Node;

// Define the structure for the stack
typedef struct Stack {
    Node* top;
} Stack;

// Function to create a new node
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Function to initialize the stack
void initStack(Stack* stack) {
    stack->top = NULL;
}

// Check if the stack is empty
int isEmpty(Stack* stack) {
    return stack->top == NULL;
}

// Push an element onto the stack
void push(Stack* stack, int element) {
    Node* newNode = createNode(element);
    newNode->next = stack->top;
    stack->top = newNode;
}

// Pop an element from the stack
int pop(Stack* stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty. Cannot pop from the stack.\n");
        return -1; // Return a sentinel value to indicate failure
    }
    int data = stack->top->data;
    Node* temp = stack->top;
    stack->top = stack->top->next;
    free(temp);
    return data;
}

// Get the top element of the stack without removing it
int peek(Stack* stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty. Cannot peek the stack.\n");
        return -1; // Return a sentinel value to indicate failure
    }
    return stack->top->data;
}

int main() {
    Stack stack;
    initStack(&stack);
    push(&stack, 10);
    push(&stack, 20);
    push(&stack, 30);
    printf("Top element: %d\n", peek(&stack));
    printf("Popped element: %d\n", pop(&stack));
    printf("Top element: %d\n", peek(&stack));
    return 0;
}
```

OUTPUT:

```
Top element: 30
Popped element: 30
Top element: 20
```