Techno International New Town
(Formerly known as Techno India College of Technology)

Introduction And Implementations Stacks in C

Name : PRIYADEEP MITRA
Roll : 18730322002
Registration no : 221870110042
Stream : ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING
Paper Name : Economics for Engineers (Humanities-II)
Paper Code : HSMC-301
Semester : 3RD
Year : 2023-24

CONTENT PAGE

INTRODUCTION

A stack is a fundamental data structure in computer science that follows the Last-In-First-Out (LIFO) principle. It is commonly used to manage and manipulate data in various programming scenarios. In C programming, stacks play a crucial role in efficiently handling function calls, managing expressions, and implementing undo/redo functionality . At its core, a stack can be visualized as a collection of elements stacked on top of each other, much like a stack of plates. The last item added to the stack is the first one to be removed, mirroring the way we handle items in real-world scenarios.

BASIC OPERATIONS ON STACK

In the context of stacks, several fundamental operations are used to interact with and manipulate the data stored within the stack. These operations are the building blocks that allow us to effectively manage the LIFO behavior of stacks. Let's explore these operations in more detail:

- **Push Operation**

The push operation involves adding an element onto the top of the stack.This operation increments the stack's "top" pointer to point to the newly added element.

- **Pop Operation**

The pop operation removes the element from the top of the stack.This operation decrements the stack's "top" pointer to the previous element.It is performed using a function like int pop(struct Stack stack);*

- **Peek (Top) Operation**

The peek operation allows you to view the element at the top of the stack without removing it.It helps you access the top element's value without altering the stack's structure.

- **IsEmpty Operation**

The isEmpty operation is used to check whether the stack is empty or not.It returns a boolean value indicating whether the stack contains any elements.

Implementation of Stack Using Arrays

- Array-based stack structure Defining the stack structure Code snippet:

struct Stack { int maxSize; int top; int array; };*

- Initializing the stack Code snippet:

struct Stack create Stack(int maxSize);*

- Push and Pop operations Code snippets:

void push(struct Stack stack, int item);, int pop(struct Stack* stack);*

- Memory management and resizing Code snippet:

void resize(struct Stack stack);*

- Implementation of Stack Using Linked Lists Linked list-based stack structure Defining the stack node structure Code snippet:

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

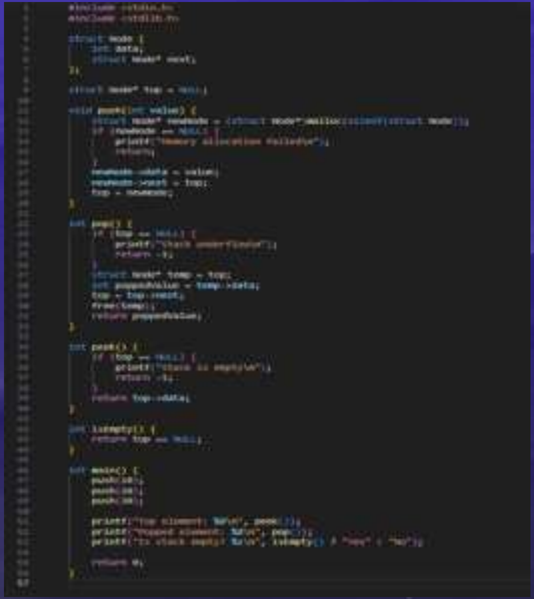
- Initializing the stack Code snippet:

struct StackNode createStack();*

- Push and Pop operations Code snippets:

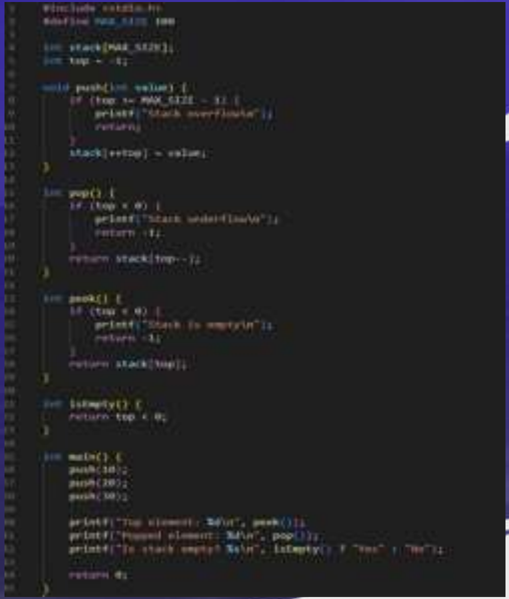
*void push(struct StackNode** root, int item);*
*int pop(struct StackNode** root);*

Stack Implementation in C (Array, linked list)



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

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

struct Node* top = NULL;

void push(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        return;
    }
    newNode->data = value;
    newNode->next = top;
    top = newNode;
}

int pop() {
    if (top == NULL) {
        printf("Stack underflow\n");
        return -1;
    }
    struct Node* temp = top;
    int poppedValue = temp->data;
    top = top->next;
    free(temp);
    return poppedValue;
}

int peek() {
    if (top == NULL) {
        printf("Stack is empty\n");
        return -1;
    }
    return top->data;
}

int isEmpty() {
    return top == NULL;
}

int main() {
    push(10);
    push(20);
    push(30);

    printf("Top element: %d\n", peek());
    printf("Popped element: %d\n", pop());
    printf("Is stack empty? %s\n", isEmpty() ? "Yes" : "No");

    return 0;
}
```

Stack in Linked list



```c
#include <stdio.h>
#define MAX_SIZE 100

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

void push(int value) {
    if (top >= MAX_SIZE - 1) {
        printf("Stack overflow\n");
        return;
    }
    stack[++top] = value;
}

int pop() {
    if (top < 0) {
        printf("Stack underflow\n");
        return -1;
    }
    return stack[top--];
}

int peek() {
    if (top < 0) {
        printf("Stack is empty\n");
        return -1;
    }
    return stack[top];
}

int isEmpty() {
    return top < 0;
}

int main() {
    push(10);
    push(20);
    push(30);

    printf("Top element: %d\n", peek());
    printf("Popped element: %d\n", pop());
    printf("Is stack empty? %s\n", isEmpty() ? "Yes" : "No");

    return 0;
}
```

Stack in Array

THANK YOU