Data Structure and Algorithm (MCA 271)

ESE 1 –

BY

Himanshu Heda (24225013)

SUBMITTED TO

Prof. Vandna Kansal

SCHOOL OF SCIENCES

2024-2025

Program Description:

Code of the program

Output: - Paste the o/p of the program.

F4

Page No

CHRIST (DEEMED TO BE UNIVERSITY) DELHI-NCR, INDIA

Name - Himanshu Heda Reg No. - 24225013
Class - 2 MCA Set No - 2
Sub - DSA ESE-1

Ans2 Stack :-

A stack is a ~~loan~~ linear data structure that follows LIFO concept (last in, first out). It means that the element will insert first will exit at the last. It will visual when the plate will add or remove the data.

-> It is having the time complexity :-

1) Insertion :-
We can insert the data with the help of O(n).
We will insert the data in the front and then it will pop out first.

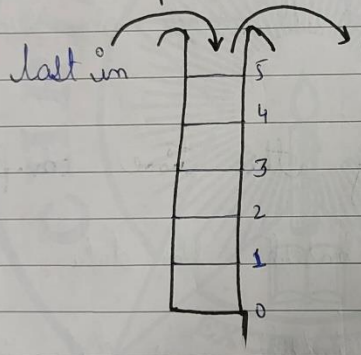
2) Deletion :-
We can delete the module by the help of O(1).
We can delete the element from the top to the bottom.

3) Searching :-
We can search the module by O(1) if its one we will add ~~find the~~ [illegible]

element weather it is present it ~~until~~ inside the node or not.

4) ~~Traversing~~ & - Upation :-
We will ~~update~~ update the element from the middle or from the beginning also basically we will update the element in the middle.
-> The time complexity for updation is O(1).

A basic representation of Stack is :-



Real life example of Stack is :-
-> Stack overf The best real life example for Stack is undo and redo.
-> The operation we insert the first will exits at the last and the operation we will perform in the first will exits at the first.

-> Another example is A bucket with a filled water.

-> Push ——— "
-> POP an element from the stack
-> PEEK (view the top element)
-> Display all the elements
-> Check weather if the stack is empty

Lets start the coding of all the above mentioned topics :-

```
# include < stdio.h>
# include < stdlib.h>
# include <
# Define MAX 100
```

```
int main(){
  void Push(){
      if (top == -1)
      {
          printf(" Enter the Insert the elements
```

```
Void int main(){ int top;
        printf(" Enter the choice :")
        printf(" 1. PUSH \n 2. POP \n 3. PEEK
          \n 4. DISPLAY \n 5. check if
            empty or not ");
}
  Void Push()
  {
      if (top == MAX - 1) {
```

```
    printf("Enter the element in the Stack:");
}
else{
    printf("Stack is empty");
}

void POP(){
    if(top == -1)
    {
        printf("the element is displayed.")
    }

void PEEK(){
    if (top == 1)
    {
        printf("The element of the stack is
    }

void display{
    if (top == 1)
    {
        printf("The All the elements are:");
    }
```

```
void main(){
    Push();
    Pop();
    PEEK();
    Display();
}
```

Practical Implementation : --

```c
#include <stdio.h>

#define MAX 100 // Maximum size of the stack

// Stack structure
struct Stack
{
    int items[MAX];
    int top;
};

// Function prototypes
void initStack(struct Stack *stack);
int isFull(struct Stack *stack);
int isEmpty(struct Stack *stack);
void push(struct Stack *stack, int value);
int pop(struct Stack *stack);
int peek(struct Stack *stack);
void display(struct Stack *stack);

int main()
{
    struct Stack stack;
    initStack(&stack);

    int ch, value;

    do
    {
        printf("\nEnter your choice:\n");
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Peek\n");
        printf("4. Display\n");
        printf("5. Exit\n");
        printf("Choice: ");
        scanf("%d", &ch);

        switch (ch)
```

```c
        {
        case 1:
            printf("Enter value to push: ");
            scanf("%d", &value);
            push(&stack, value);
            break;
        case 2:
            value = pop(&stack);
            if (value != -1)
            {
                printf("Popped value: %d\n", value);
            }
            break;
        case 3:
            value = peek(&stack);
            if (value != -1)
            {
                printf("Top value: %d\n", value);
            }
            break;
        case 4:
            display(&stack);
            break;
        case 5:
            printf("Exiting...\n");
            break;
        default:
            printf("Invalid choice, please try again.\n");
            break;
        }
    } while (ch != 5);

    return 0;
}

// Initialize the stack
void initStack(struct Stack *stack)
{
    stack->top = -1; // Stack is empty
}
```

```c
// Check if the stack is full
int isFull(struct Stack *stack)
{
    return stack->top == MAX - 1;
}

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

// Push an item onto the stack
void push(struct Stack *stack, int value)
{
    if (isFull(stack))
    {
        printf("Stack overflow! Cannot push %d\n", value);
    }
    else
    {
        stack->items[++stack->top] = value;
        printf("Pushed %d onto the stack.\n", value);
    }
}

// Pop an item from the stack
int pop(struct Stack *stack)
{
    if (isEmpty(stack))
    {
        printf("Stack underflow! Cannot pop from an empty stack.\n");
        return -1; // Return -1 to indicate an error
    }
    else
    {
        return stack->items[stack->top--];
    }
}

// Peek at the top item of the stack
```

```c
int peek(struct Stack *stack)
{
    if (isEmpty(stack))
    {
        printf("Stack is empty! Cannot peek.\n");
        return -1; // Return -1 to indicate an error
    }
    else
    {
        return stack->items[stack->top];
    }
}

// Display the elements of the stack
void display(struct Stack *stack)
{
    if (isEmpty(stack))
    {
        printf("Stack is empty!\n");
    }
    else
    {
        printf("Stack elements: ");
        for (int i = 0; i <= stack->top; i++)
        {
            printf("%d ", stack->items[i]);
        }
        printf("\n");
    }
}
```

OUTPUT : --

```
PS D:\2MCA\DSA> .\stack_op.exe

Enter your choice:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Choice: 1
Enter value to push: 3
Pushed 3 onto the stack.

Enter your choice:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Choice: 1
Enter value to push: 2
Pushed 2 onto the stack.

Enter your choice:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Choice: 1
Enter value to push: 5
Pushed 5 onto the stack.

Enter your choice:
1. Push
2. Pop
3. Peek
```

```
Enter your choice:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Choice: 2
Popped value: 5

Enter your choice:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Choice: 3
Top value: 2

Enter your choice:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Choice: 4
Stack elements: 3 2

Enter your choice:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Choice: 5
Exiting...
PS D:\2MCA\DSA>
```