DATA STRUCTURES
USING "C"

(**Stack**)

For

BCA 2nd Year (Session 2018-21) Students

BY

ANANT KUMAR
MCA, M. Phil, M. Tech.
Faculty Member
Department of Computer Science
J. D. Women's College, Patna

Stack

A stack is a list of elements in which an element may be inserted or deleted only at one end, called the top of the stack. It is also known as LIFO (last in, first out) system.

The items can be added or removed only from the top i.e. the last item to be added to a stack is the first item to be removed.

The two basic operations associated with stacks are:

Push() - is the term used to insert an element into a stack.

Pop() is the term used to delete an element from a stack.

"Push" is the term used to insert an element into a stack. "Pop" is the term used to delete an element from the stack.

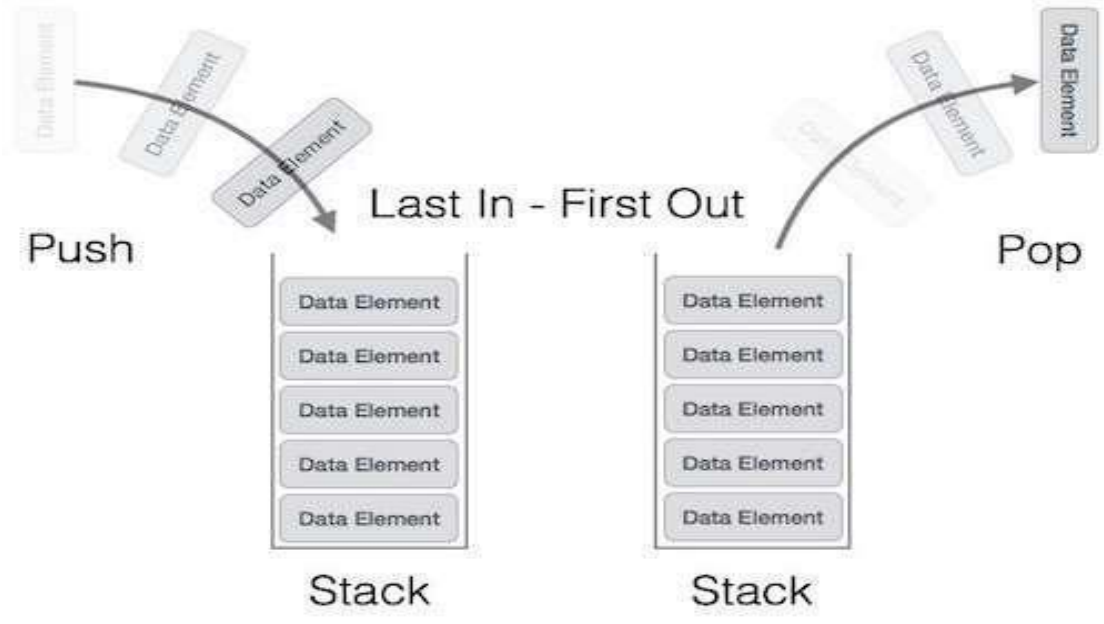
Some other operations associated with stacks are:

peek() − get the top data element of the stack, without removing it.

isFull() − check if stack is full.

isEmpty() − check if stack is empty.

Representation of Stack



Implementation of Stack

A stack can be implemented by:

i. An array
ii. Linked List.

Advantages of using Stack

- When a function is called the local variables are stored in a stack, and it is automatically destroyed once returned.
- A stack is used when a variable is not used outside that function.
- It allows you to control how memory is allocated and deallocated.
- Stack automatically cleans up the object.
- Variables cannot be resized.

Disadvantages of using Stack

- Stack memory is very limited.
- Creating too many objects on the stack can increase the risk of stack overflow.
- Random access is not possible.
- Variable storage will be overwritten, which sometimes leads to undefined behaviour of the function or program.
- The stack will fall outside of the memory area, which might lead to an abnormal termination.

Applications of Stack

Applications of stacks are:

1. Expression evaluation
2. Backtracking (game playing, finding paths, exhaustive searching)
3. Memory management, run-time environment for nested language features.
4. Tree Traversal
5. Graph Traversal
6. Recursion etc.

Stack operations using an array

```
# include <stdio.h>
# include <conio.h>
# include <stdlib.h>
# define MAX 6
int stack[MAX];
int top = -1;

void push();
void pop();
void display();
void peek();

void main()
{
        int ch;
        while(1)
        {
                clrscr();
                printf("\n … Stack operations using ARRAY... ");
                printf("\n----------**********  \n");
                printf("\n 1. Push ");
                printf("\n 2. Pop ");
                printf("\n 3. Display");
                printf("\n 4. Display Top Element");
                printf("\n 5. Quit ");
                printf("\n Enter your choice: ");
                scanf("%d", &ch);
                switch(ch)
                {
                        case1:
                                push();
                                break;
                        case2:
                                pop();
                                break;
                        case3:
                                display();
                                break;
```

```
                case4:
                        peek();
                        break;
                case 5:
                        exit(0);
                }
                getch();
        }
}


void pop()
{
        int val;
        if(top == -1)
                printf("\n\nStack Underflow..");
        else
        {
                printf("\n\npopped element is:" );
                val=stack[top]);
                top--;
                printf("%d", val);
        }
}


void push()
{
        int data;
        if(top == MAX)
                printf("\n\nStack Overflow..");
        else
        {
                printf("\n\nEnter data: ");
                scanf("%d", &data);
                top++;
                stack[top] = data;
                printf("\n\nData Pushed into the stack");
        }
}
```

```
void display()
{
	int i;
	if(top == -1)
		printf("\n\nStack empty..");
	else
	{
		printf("\n\nElements in stack:");
		for(i = 0; i < top; i++)
		printf("\t%d", stack[i]);
	}

}

void peek()
{
	if(top == -1)
		printf("\n\nStack Underflow..");
	else
		printf("\n\Top element is: %d", stack[top] );
}
```

Stack operations using linked list

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

typedef struct stack
{
        int data;
        struct stack *next;
}node;


node* getnode();
void push();
void pop();
void display();

node*start=NULL;
node *top =NULL;

void main()
{
        int ch;
        while(1)
        {
                clrscr();
                printf("\n … Stack operations using ARRAY... ");
                printf("\n----------**********  \n");
                printf("\n 1. Push ");
                printf("\n 2. Pop ");
                printf("\n 3. Display");
                printf("\n 4. Quit ");
                printf("\n Enter your choice: ");
                scanf("%d", &ch);
                switch(ch)
                {
                        case1:
                                newnode = getnode();
                                push(newnode);
                                break;
```

```
                case2:
                        pop();
                        break;
                case3:
                        display();
                        break;
                case 4:
                        exit(0);
                }
                getch();
        }
}

node* getnode()
{
        node *temp;
        temp=(node *) malloc( sizeof(node)) ;
        printf("\n Enter data ");
        scanf("%d", &temp -> data);
        temp -> next = NULL;
        return temp;
}

void push(node *newnode)
{
        node *temp;
        if( newnode == NULL )
                printf("\n Stack Overflow..");
        if(start == NULL)
        {
                start = newnode;
                top = newnode;
        }
        else
        {
                temp = start;
                while( temp -> next != NULL)
                        temp = temp -> next;
                temp -> next = newnode;
                top = newnode;
        }
```

```
        printf("\n\n\t Data pushed into stack");
}

void pop()
{
        node *temp;
        if(top == NULL)
                printf("\n\n\t Stack underflow");
        temp = start;
        if( start -> next == NULL)
        {
                printf("\n\n\t Popped element is %d ", top -> data);
                start = NULL;
                free(top);
                top = NULL;
        }
        else
        {
                while(temp -> next != top)
                        temp = temp -> next;
                temp -> next = NULL;
                printf("\n\n\t Popped element is %d ", top -> data);
                free(top);
                top = temp;
        }
}

void display()
{
        node *temp;
        if(top == NULL)
                printf("\n\n\t\t Stack is empty ");
        else
        {
                temp = start;
                printf("\n\n\n\t\t Elements in the stack: \n");
                printf("%5d ", temp -> data);
```

```
                while(temp != top)
                {
                        temp = temp -> next;
                        printf("%5d ", temp -> data);
                }
        }
}
```