**Stack Using Array**

[http://btechsmartclass.com/DS/images/Next.png](http://btechsmartclass.com/DS/U2_T3.html)

A stack data structure can be implemented using one dimensional array. But stack implemented using array, can store only fixed number of data values. This implementation is very simple, just define a one dimensional array of specific size and insert or delete the values into that array by using **LIFO principle** with the help of a variable **'top'**. Initially top is set to -1. Whenever we want to insert a value into the stack, increment the top value by one and then insert. Whenever we want to delete a value from the stack, then delete the top value and decrement the top value by one.

**Stack Operations using Array**

A stack can be implemented using array as follows...  
  
Before implementing actual operations, first follow the below steps to create an empty stack.

* **Step 1:** Include all the **header files** which are used in the program and define a constant **'SIZE'** with specific value.
* **Step 2:** Declare all the **functions** used in stack implementation.
* **Step 3:** Create a one dimensional array with fixed size (**int stack[SIZE]**)
* **Step 4:** Define a integer variable **'top'** and initialize with **'-1'**. (**int top = -1**)
* **Step 5:** In main method display menu with list of operations and make suitable function calls to perform operation selected by the user on the stack.

**push(value) - Inserting value into the stack**

In a stack, push() is a function used to insert an element into the stack. In a stack, the new element is always inserted at **top** position. Push function takes one integer value as parameter and inserts that value into the stack. We can use the following steps to push an element on to the stack...

* **Step 1:** Check whether **stack** is **FULL**. (**top == SIZE-1**)
* **Step 2:** If it is **FULL**, then display **"Stack is FULL!!! Insertion is not possible!!!"** and terminate the function.
* **Step 3:** If it is **NOT FULL**, then increment **top** value by one (**top++**) and set stack[top] to value (**stack[top] = value**).

**pop() - Delete a value from the Stack**

In a stack, pop() is a function used to delete an element from the stack. In a stack, the element is always deleted from **top** position. Pop function does not take any value as parameter. We can use the following steps to pop an element from the stack...

* **Step 1:** Check whether **stack** is **EMPTY**. (**top == -1**)
* **Step 2:** If it is **EMPTY**, then display **"Stack is EMPTY!!! Deletion is not possible!!!"** and terminate the function.
* **Step 3:** If it is **NOT EMPTY**, then delete **stack[top]** and decrement **top** value by one (**top--**).

**display() - Displays the elements of a Stack**

We can use the following steps to display the elements of a stack...

* **Step 1:** Check whether **stack** is **EMPTY**. (**top == -1**)
* **Step 2:** If it is **EMPTY**, then display **"Stack is EMPTY!!!"** and terminate the function.
* **Step 3:** If it is **NOT EMPTY**, then define a variable '**i**' and initialize with top. Display **stack[i]** value and decrement **i** value by one (**i--**).
* **Step 3:** Repeat above step until **i** value becomes '0'.

**Complete Program in C Programming Language**#include<stdio.h>

#include<conio.h>

struct Node

{

int data;

struct Node \*next;

}\*top = NULL;

void push(int);

void pop();

void display();

void main()

{

int choice, value;

clrscr();

printf("\n:: Stack using Linked List ::\n");

while(1){

printf("\n\*\*\*\*\*\* MENU \*\*\*\*\*\*\n");

printf("1. Push\n2. Pop\n3. Display\n4. Exit\n");

printf("Enter your choice: ");

scanf("%d",&choice);

switch(choice){

case 1: printf("Enter the value to be insert: ");

scanf("%d", &value);

push(value);

break;

case 2: pop(); break;

case 3: display(); break;

case 4: exit(0);

default: printf("\nWrong selection!!! Please try again!!!\n");

}

}

}

void push(int value)

{

struct Node \*newNode;

newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = value;

if(top == NULL)

newNode->next = NULL;

else

newNode->next = top;

top = newNode;

printf("\nInsertion is Success!!!\n");

}

void pop()

{

if(top == NULL)

printf("\nStack is Empty!!!\n");

else{

struct Node \*temp = top;

printf("\nDeleted element: %d", temp->data);

top = temp->next;

free(temp);

}

}

void display()

{

if(top == NULL)

printf("\nStack is Empty!!!\n");

else{

struct Node \*temp = top;

while(temp->next != NULL){

printf("%d--->",temp->data);

temp = temp -> next;

}

printf("%d--->NULL",temp->data);

}

}