|  |  |
| --- | --- |
|  | **DEPARTMENT OF COMPUTER ENGINEERING** |

**Experiment No. 01**

|  |  |
| --- | --- |
| Semester | S.E. Semester III – Computer Engineering |
| Subject | Data Structures Lab (CSL301) |
| Subject Professor In-charge | Prof. Swapnil S. Sonawane |
| Assisting Teachers | Prof. Swapnil S. Sonawane |

|  |  |
| --- | --- |
| Student Name | Trisha Shah |
| Roll Number | 20102A0004 |

**Title:**

Implement Stack ADT using array.

**Objective:**

Students will be able to implement Linear and Non-Linear data structures.

**Explanation:**

**Stack:** A stack is a linear data structure that follows the LIFO (Last in First Out) principle. It is an Abstract Data Type (ADT), commonly used in most programming languages. It contains only one pointer TOP pointing to the topmost element of the stack. Whenever an element is added to the stack, it is added to the top of the stack and only the top element can be deleted from the stack.

Stack operations may involve initializing the stack, using it and then de-initializing it. Apart from these basic stuffs, a stack is used for the following operations −

**Push Operation:** It is used to insert a new element into stack. Push operation involves a series of steps −

Step 1 − Checks if the stack is full.

Step 2 − If the stack is full, produces an error and exit.

Step 3 − If the stack is not full, increments top to point next empty space.

Step 4 − Adds data element to the stack location, where top is pointing.

Step 5 − Returns success.

**Pop Operation:** Accessing the content while removing it from the stack, is known as a Pop Operation. In an array implementation of pop() operation, the data element is not actually removed, instead top is decremented to a lower position in the stack to point to the next value.

**Peek Operation:** It is used to retrieve topmost element of the stack.

**Stack Overflow:** While pushing element into stack we need to check “Stack Overflow” condition.

**Stack Underflow:** While popping element we need to check the “Stack Underflow” condition.

**Program Code:**

#include<stdio.h>

#define N 20

typedef struct stack

{

int a[N];

int top;

}stack;

void push(stack \*s,int x)

{

if(s->top==N-1)

printf("\nStack Overflow...");

else

{

s->top=s->top+1;

s->a[s->top]=x;

}

}

int isempty(stack \*s)

{

if(s->top==-1)

return 1;

else

return 0;

}

int pop(stack \*s)

{

int x;

if(isempty(s))

{

return -1;

}

else

{

x=s->a[s->top];

s->top=s->top-1;

return x;

}

}

int peek(stack \*s)

{

if(isempty(s))

return -1;

else

return s->a[s->top];

}

void display(stack \*s)

{

int i;

if(isempty(s))

{

printf("\nStack is empty...");

}

else

{

for(i=s->top;i>=0;i--)

{

printf("\n%d",s->a[i]);

}

}

}

int main()

{

int ch,x;

stack s;

s.top=-1;

while(1)

{

**printf("\n1:Push\n2:Pop\n3:Peek\n4:Display\n5:Exit\nEnter choice=");**

**scanf("%d",&ch);**

**if(ch==5)**

**break;**

**switch(ch)**

**{**

**case 1:**

**{**

**printf("\nEnter element to be pushed=");**

**scanf("%d",&x);**

**push(&s,x);**

**}**

**break;**

**case 2:**

**{**

**x=pop(&s);**

**if(x==-1)**

**printf("\nStack Underflow...");**

**else**

**printf("\nPopped Element=%d",x);**

**}**

**break;**

**case 3:**

**{**

**x=peek(&s);**

**if(x==-1)**

**printf("\nStack is empty...");**

**else**

**printf("\nStack top element=%d",x);**

**}**

**break;**

**case 4:**

**{**

**display(&s);**

**}**

**break;**

**default:**

**{**

**printf("\nInvalid Choice...");**

**}**

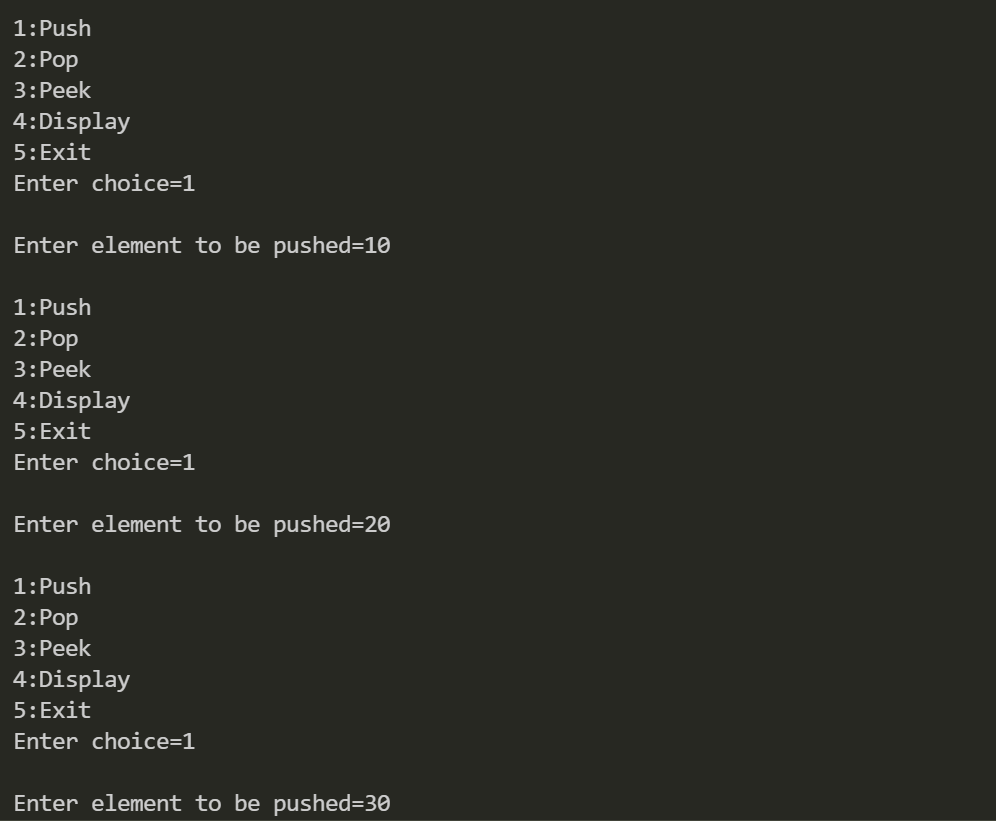
**}**

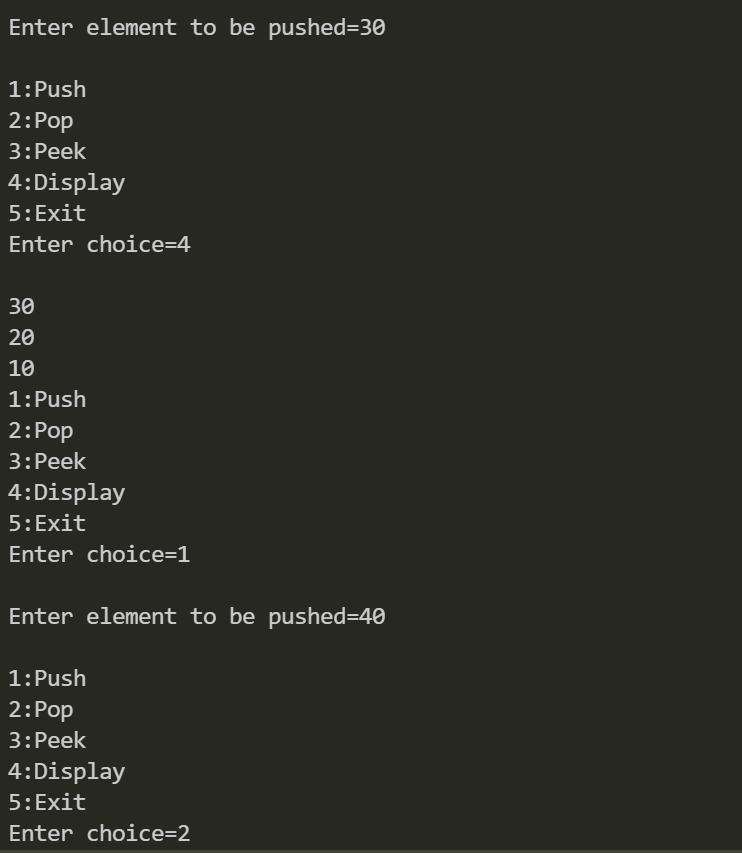
**}**

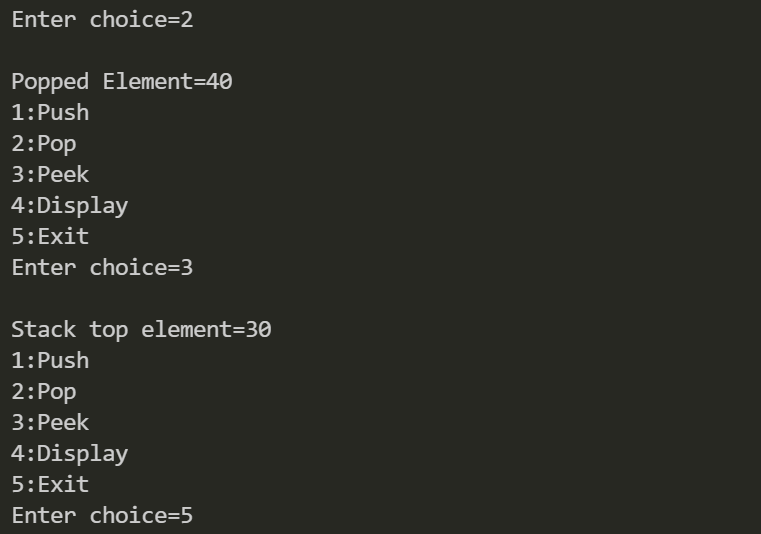
**return 0;**

**}**

**Output:**

****

****

****

**Conclusion:**

Through this experiment, students learned how to create a stack using an array and learned to implement various operations such as, pop, push peek and display.