Experiment No.1
Implement Stack ADT using array.
Date of Performance:25/07/2023
Date of Submission:13/08/2023

Experiment No. 1: To implement stack ADT using arrays

Aim: To implement stack ADT using arrays.

Objective:

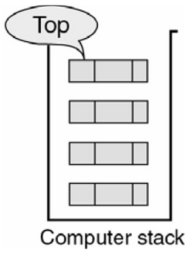
1) Understand the Stack Data Structure and its basic operators.

2) Understand the method of defining stack ADT and implement the basic operators.

3) Learn how to create objects from an ADT and invoke member functions.

Theory:

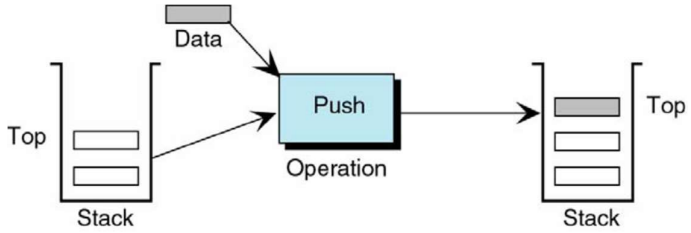
A stack is a list in which all insertions and deletions are made at one end, called the top. It is a collection of contiguous cells, stacked on top of each other. The last element to be inserted into the stack will be the first to be removed. Thus stacks are sometimes referred to as Last In First Out (LIFO) lists.



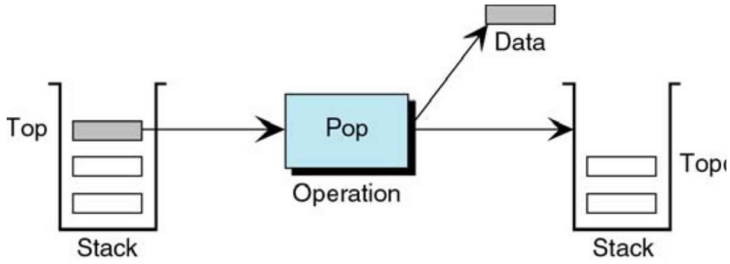
The operations that can be performed on a stack are push, pop which are main operations while auxiliary operations are peek, isEmpty and isFull. Push is to insert an element at the top of the stack. Pop is deleting an element that is at the top most position in the stack. Peek simply examines and returns the top most value in the stack without deleting it.

Push on an already filled stack and pop on an empty stack results in serious errors so isEmpty and isFull function checks for stack empty and stack full respectively. Before any insertion, the value of the variable top is initialized to -1.

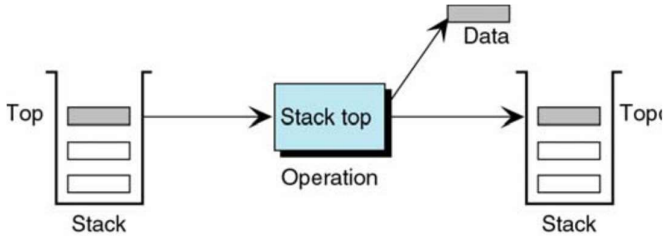
Push Operation



Pop Operation



Peek Operation



Algorithm:

PUSH(item)

1. If (stack is full)

   Print "overflow"

2. top = top + 1
3. stack[top] = item

   Return

POP()

1. If (stack is empty)

   Print "underflow"

2. Item = stack[top]
3. top = top – 1
4. Return item

PEEK()

1. If (stack is empty)

   Print "underflow"

2. Item = stack[top]

3. Return item

ISEMPTY()

1. If(top = -1)then

return 1

2. return 0

ISFULL()

1. If(top = max)then

return 1

2. return 0

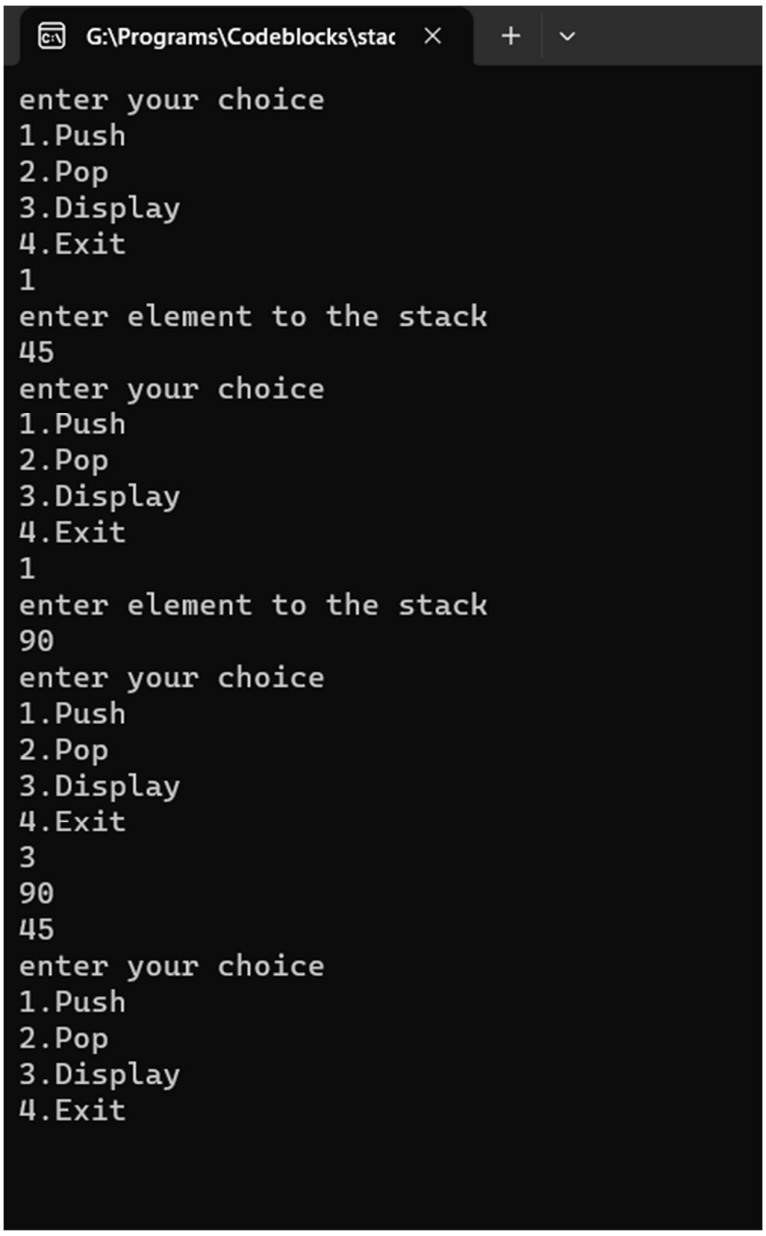
Code:

```
#include <stdio.h>
#include <stdlib.h>
#define max 15
int top = -1, a[max];
void push()
{
int x;
if (top == max - 1)
{ printf("stack overflow condition");
} else
{ printf("enter element to the stack\n"); scanf("%d", &x); top++; a[top] = x;
}
```

```
 }
void display()
{ int i;
for (i = top; i >= 0; i--)
{ printf("%d\n",a[i]);
}
}
 void pop()
{ if (top == -1)
{ printf("stack underflow condition\n");
}
else
{ printf("pop element:%d\n", a[top]);
top--;
}
}
int main()
{
int ch;
while (1)
{
printf("enter your choice\n1.Push\n2.Pop\n3.Display\n4.Exit\n");
scanf("%d", &ch);
```

```
switch (ch)
{ case 1: push(); break;
case 2: pop(); break;
 case 3:
display(); break;
case 4:
exit(0); break;
default:
    printf("invlaid choice\n"); break;
}
} return 0;
}
```

Output:



Conclusion:

Stack is one of the simplest data structures to implement and execute. It used LIFO operation i.e (last in first out) method to carry out push and pop operations. Time complexity of the

program was O(1).Here Arrays are used to stored data instead of linked list so it was easier to display. Also we can use stack as function ton carry out another tasks.

Applications:

1) Well formed-ness of Parenthesis
2) Infix to postfix conversation
3) Postfix evaluation
4) Prefix evaluation