Name: Bhupendra Yadav ,   Div: C , Roll no:65

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 ADT and invoke member functions.

Theory:

PUSH function in the code is used to insert an element to the top of stack, POP function used to remove the element from the top of stack.

Finally, the display function in the code is used to print the values. All stack functions are implemented in C Code.

The same implementation of stack using c is written using pointers: Stack operations using pointers in c

Algorithm:

Step 1: Define the stack structure and variables

Step 2: Initialize the stack

Step 3: Check if the stack is empty

Step 4: Check if the stack is full
Step 5: Push an element onto the stack

Step 6: Pop an element from the stack

Step 7: Peek at the top element of the stack without removing it

Step 8: Main program to test the stack

Code:

```c
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
int stack[100],top,n,choice,x,i;
void push();
void pop();
void peek();
void display();
int main()
{
clrscr ();
top=-1;
printf("Enter the size of stack:");
scanf("%d",&n);
printf("1.push\n2.pop\n3.peek\n4.display\n5.exit\n");
while(choice!=5)
{
printf("Enter your choice: ");
scanf("%d",&choice);
switch (choice)
{
case 1:
{
push();
```

```c
break;
}
case 2:
{
pop();
break;
}
case 3:
{
peek();
break;
}
case 4:
{
display();
break;
}
case 5:
{
printf("exit point");
break;
}
default:
{
printf("Enter a valid choice");
}
getch();
return 0;
}
}
}
```
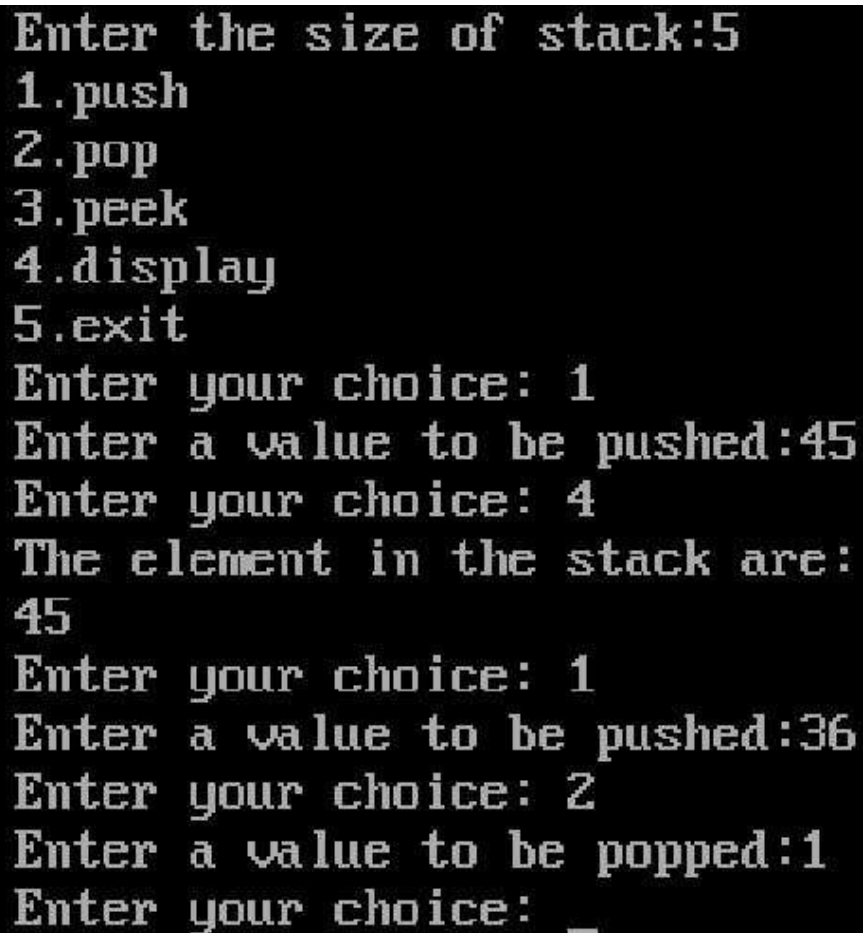
```c
void push()
{
if (top==n-1)
{
printf("Stack is overflow\n");
}
else
{
printf("Enter a value to be pushed:");
scanf("%d",&x);
top++;
stack[top]=x;
}
}
void pop()
{
if (top==n-1)
{
printf("Stack is underflow\n");
}
else
{
printf("Enter a value to be popped:");
scanf("%d",&x);
top--;
stack[top]=x;
}
}
void peek()
{
printf("The top element of the stack is %d \n",x);
```

```
}
void display()
{
if (top==0)
{
printf("The element in the stack are:\n");
for(i=top;i>=0;i--)
printf("%d\n",stack[i]);
}
else
{
printf("The stack is empty\n");
}
}
```

Output:


```
Enter the size of stack:5
1.push
2.pop
3.peek
4.display
5.exit
Enter your choice: 1
Enter a value to be pushed:45
Enter your choice: 4
The element in the stack are:
45
Enter your choice: 1
Enter a value to be pushed:36
Enter your choice: 2
Enter a value to be popped:1
Enter your choice: _
```

Conclusion:

In conclusion, the implementation of a stack Abstract Data Type (ADT) using arrays in C provides a straightforward and efficient way to manage Last-In-First-Out (LIFO) data structures. By utilizing an array for storage and tracking the top index, we achieve constant-time complexity for push and pop operations. However, care must be taken to handle potential overflow and underflow scenarios to ensure the integrity of the stack. Overall, this implementation offers a fundamental foundation for various applications that require stack-based data manipulation.