Experiment No 1: To implement stack ADT using arrays

Aim: To implement stack ADT using arrays

Objective:

1) Understand the use of stack
2) Understand how to import an ADT in an application program
3) Understand the instantiation of stack ADT in an application program
4) Understand how the member function of an ADT are accessed in an application program

Theory: A stack is a list in which all insertions & deletions are made at one end, called top. It is collections of contiguous cells,stacked on top of each other. The last element to be inserted into the stack will be first to be removed. Thus stacks are sometimes reffered to as Last In First Out

Algorithm:

1 − Checks if the stack is full.

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

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

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

5 − Returns success.

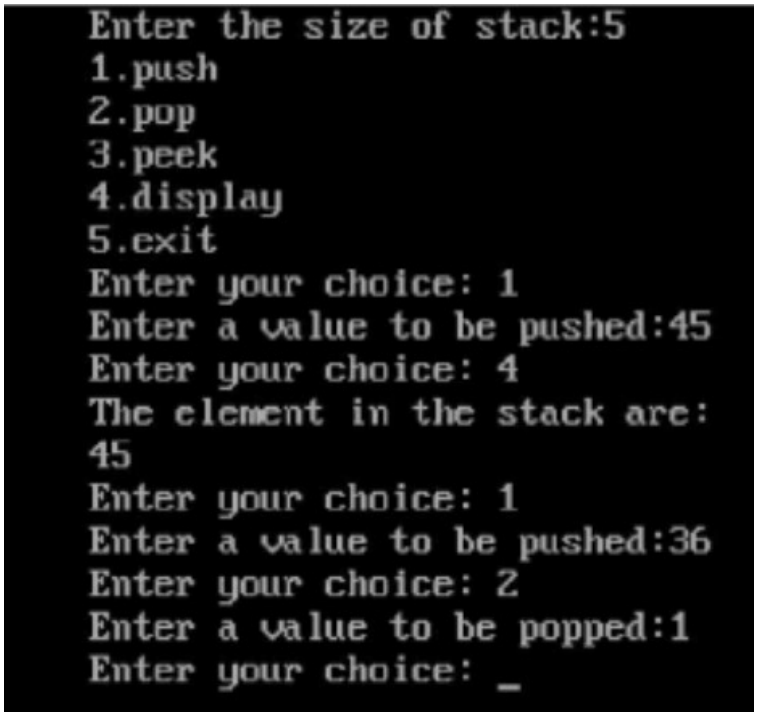
Code

```
#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()
```

```
{
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();
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;
}
}
}
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



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.