EX.NO:1A ARRAY IMPLEMENTATION OF STACK

ADT **AIM:**

Aim is to write a program in C to implement the stack ADT using array concept that performs all the operations of stack.

DESCRIPTION:

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.

ALGORITHM:

STEP 1: Define an array to store the element.

STEP 2: Get the users' choice.

STEP 3: If the option is 1 perform creation operation and goto step4.

If the option is 2 perform insertion operation and goto step5.

If the option is 3 perform deletion operation and goto step6.

If the option is 4 perform display operation and goto step7.

STEP 4: Create the stack. Initially get the limit of stack and the get the items. If the limit of stack is exceeds print the message unable to create the stack.

STEP 5: Get the element to be pushed. If top pointer exceeds stack capacity. Print Error message that the stack overflow. If not, increment the top pointer by one and store the element in the position which is denoted by top pointer.

STEP 6: If the stack is empty, then print error message that stack is empty. If not fetch the element from the position which is denoted by top pointer and decrement the top pointer by one

STEP 7: If the top value is not less than the 0 the stack is display otherwise print the message "stack is empty".

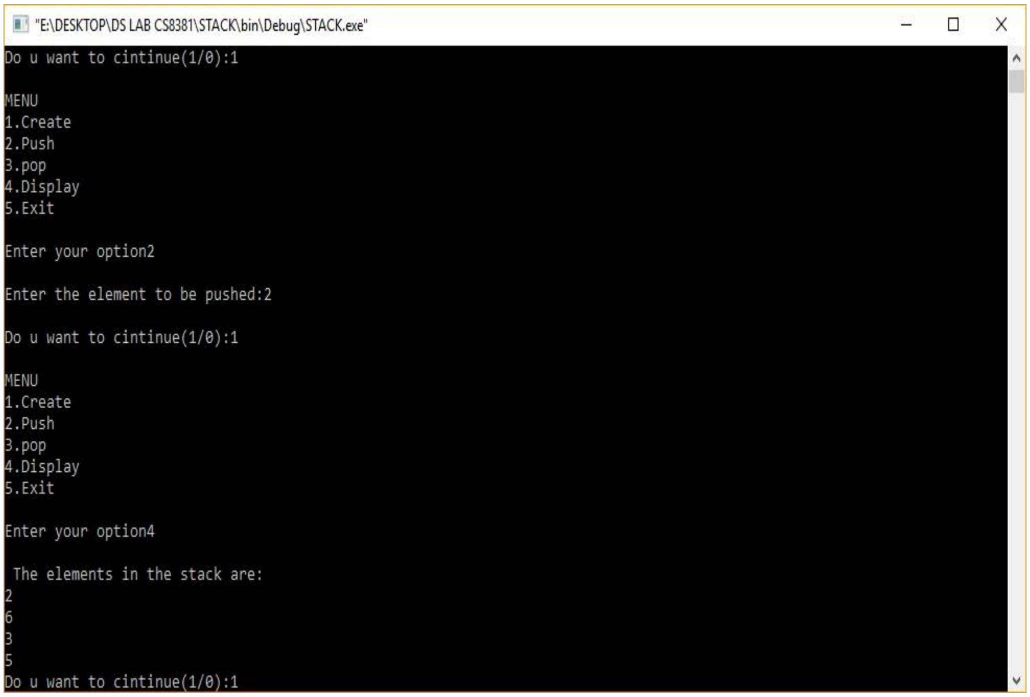
STEP 8: Stop the execution.

PROGRAM:

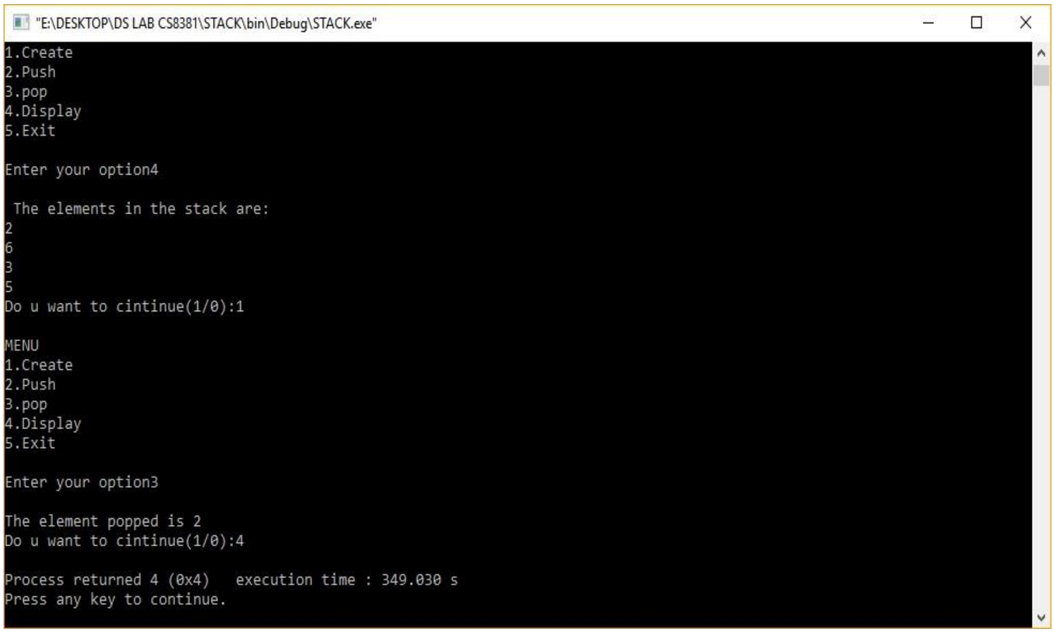
```
#include<stdio.h>
#include<conio.h> #define max 20
int opt, a[20],i,top=0,n; void
main() {  void
create(),push(),pop(),disp();
int wish;
     do
     {
     clrscr();      printf("\nMENU");
printf("\n1.Create\n2.Push\n3.pop\n4.Display\n5.Exit\n");
printf("\nEnter your option");     scanf("%d",&opt);
switch(opt)
     {
          case
1:create();break;        case
2:push();break;          case
3:pop();break;           case
4:disp();break;          case
5:exit(0);
     }
     printf("\nDo u want to
cintinue(1/0):");   scanf("%d",&wish);
}while(wish==1);} void create()
{    printf("\n Enter the limit of
stack");  scanf("%d",&n);if(n<max)


{         printf("\nEnter the
items");       for(i=0;i<n;i++)
     scanf("%d",&a[i]);
top=n-1;
     }
     else
     printf("\nUnable to create the stack");
} void
push()
{
     int x;
if(top<max){
               printf("\nEnter the element to be pushed:");

scanf("%d",&x);
top=top+1;
a[top]=x;
n=top;
     }
     else
     printf("\n Stack is full");
} void
pop()
{
     if(top<0)
printf("\n Stack is empty");
     else
     {
          printf("\nThe element popped is %d",a[top]);
          top=top-1;
     n=top;
```

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

OUTPUT:



```
Do u want to cintinue(1/0):1

MENU
1.Create
2.Push
3.pop
4.Display
5.Exit

Enter your option2

Enter the element to be pushed:2

Do u want to cintinue(1/0):1

MENU
1.Create
2.Push
3.pop
4.Display
5.Exit

Enter your option4

 The elements in the stack are:
2
6
3
5
Do u want to cintinue(1/0):1
```



```
1.Create
2.Push
3.pop
4.Display
5.Exit

Enter your option4

 The elements in the stack are:
2
6
3
5
Do u want to cintinue(1/0):1

MENU
1.Create
2.Push
3.pop
4.Display
5.Exit

Enter your option3

The element popped is 2
Do u want to cintinue(1/0):4

Process returned 4 (0x4)   execution time : 349.030 s
Press any key to continue.
```

RESULT:

Thus a C program for Stack using ADT was implemented successfully**.**