Data Structures Lab

Experiment 8: Multiple Stacks

Project By:

Mohammed Rabeeh
Roll No: 35
TVE18CS036

Contents

1 Aim

Write a program to implement multiple stacks using array.

2 Problem Description

The array is divided in to the required number of stacks each having a top and bottom pointer Insertion and deletion is made through the top END. Every stack can grow from boundary [stackno]+1 to boundary[stackno + 1] .During insertion if no free space is available in the required stack, a search is made for free space in any of the stack on the right side. If found the elements are shifted one position to the right side to make space for insertion and the pointers are updated. Else a similar search and updating is performed on the left side of the given stack.

3 Algorithm

3.1 Push

1. BEGIN
2. IF (top1+1!=top2 and top1!=size-1 and top2!=0)
3. PRINT overflow
4. ELSE
5. READ stack choice
6. READ item
7. IF (ch==1)
8. stack[++top]=item
9. ELSE
10. stack[–top2]=item
11. END IF

12. END IF
13. END PUSH

3.2 Pop

1. BEGIN
2. READ stack choice
3. IF (ch==1)
4. IF (top1==-1)
5. PRINT underflow
6. ELSE
7. top1–
8. END IF
9. ELSE
10. IF (top2==-1)
11. PRINT underflow
12. ELSE IF (top2==size-1)
13. top2=-1
14. ELSE
15. top2++
16. END IF
17. END IF
18. END POP

3.3 Display

1. BEGIN
2. READ stack choice
3. IF (ch==1)
4. i=top1
5. WHILE(i¿=0)
6. PRINT stack[i]
7. i–
8. END WHILE
9. ELSE
10. i=top2
11. WHILE(i¡size)
12. PRINT stack[i]
13. i++
14. END WHILE
15. END IF
16. END DISPLAY

4 Program Code

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
    int n,top1,top2,ch=1,a,i,arr[100];
    printf("Enter size of array\n");
```

```
scanf("%d",&n);
top1=-1;
top2=n;
while(ch!=0)
{
printf("Enter choice:\n");
printf("1.Push element in stack 1\n");
printf("2.Push element in stack 2\n");
printf("3.Pop element from stack 1\n");
printf("4.Pop element from stack 2\n");
printf("5.Display stack 1\n");
printf("6.Display stack 2\n");
printf("7.EXIT\n");
scanf("%d",&ch);
switch(ch)
{
    case 1:
    {
        printf("Enter the element\n");
        scanf("%d",&a);
        if(top1!=(top2-1))
            arr[++top1]=a;
        else
            printf("Overflow\n");
        break;
    }
    case 2:
    {
        printf("Enter the element\n");
        scanf("%d",&a);
        if(top2!=(top1+1))
            arr[--top2]=a;
        else
            printf("Overflow\n");
        break;
    }
    case 3:
    {
        if(top1==-1)
            printf("Stack1 is empty\n");
```

```
        else
        {
            a=arr[top1--];
            printf("%d\n",a);
        }
        break;
    }
    case 4:
    {
        if(top2==n)
            printf("Stack2 is empty\n");
        else
        {
            a=arr[top2++];
            printf("%d\n",a);
        }
        break;
    }
    case 5:
    {
        if(top1==-1)
            printf("Stack1 is empty\n");
        else
        {
            printf("Stack1 is \n");
            for(i=0;i<=top1;i++)
                printf("%d ",arr[i]);
            printf("\n");
        }
        break;
    }
case 6:
{
    if(top2==n)
        printf("Stack2 is empty\n");
    else
    {
        printf("Stack2 is \n");
        for(i=(n-1);i>=top2;i--)
        printf("%d ",arr[i]);
```

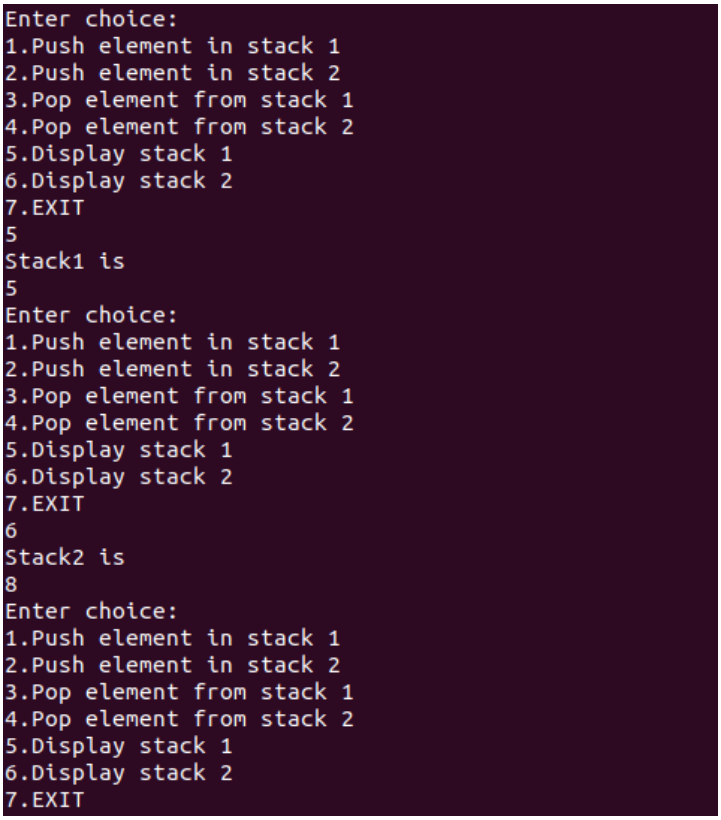
```
            printf("\n");
        }
        break;
}
case 7:
return;
}
}
}
```

5 Output

```
Enter size of array
4
Enter choice:
1.Push element in stack 1
2.Push element in stack 2
3.Pop element from stack 1
4.Pop element from stack 2
5.Display stack 1
6.Display stack 2
7.EXIT
1
Enter the element
5
Enter choice:
1.Push element in stack 1
2.Push element in stack 2
3.Pop element from stack 1
4.Pop element from stack 2
5.Display stack 1
6.Display stack 2
7.EXIT
1
Enter the element
6
Enter choice:
1.Push element in stack 1
2.Push element in stack 2
3.Pop element from stack 1
4.Pop element from stack 2
5.Display stack 1
6.Display stack 2
7.EXIT
5
Stack1 is
5 6
```

```
Enter choice:
1.Push element in stack 1
2.Push element in stack 2
3.Pop element from stack 1
4.Pop element from stack 2
5.Display stack 1
6.Display stack 2
7.EXIT
6
Stack2 is empty
Enter choice:
1.Push element in stack 1
2.Push element in stack 2
3.Pop element from stack 1
4.Pop element from stack 2
5.Display stack 1
6.Display stack 2
7.EXIT
2
Enter the element
8
Enter choice:
1.Push element in stack 1
2.Push element in stack 2
3.Pop element from stack 1
4.Pop element from stack 2
5.Display stack 1
6.Display stack 2
```

```
Enter choice:
1.Push element in stack 1
2.Push element in stack 2
3.Pop element from stack 1
4.Pop element from stack 2
5.Display stack 1
6.Display stack 2
7.EXIT
5
Stack1 is
5 6
Enter choice:
1.Push element in stack 1
2.Push element in stack 2
3.Pop element from stack 1
4.Pop element from stack 2
5.Display stack 1
6.Display stack 2
7.EXIT
6
Stack2 is
8 5
Enter choice:
1.Push element in stack 1
2.Push element in stack 2
3.Pop element from stack 1
4.Pop element from stack 2
5.Display stack 1
6.Display stack 2
7.EXIT
3
6
Enter choice:
1.Push element in stack 1
2.Push element in stack 2
3.Pop element from stack 1
4.Pop element from stack 2
5.Display stack 1
6.Display stack 2
7.EXIT
4
5
```



6 Result

Multiple stacks using array was implemented using C and the output was verified,