INDEX

Ex. no.: 2

Date :

ARRAY IMPLEMENTATION OF STACK

Aim

To write a C-program to implement stack using array data structure.

And perform the following stack operations

1. PUSH
2. POP
3. Display

Algorithm for PUSH and POP operations on Stack

Let Stack[MAXSIZE] be an array to implement the stack. The variable top denotes the top of the stack.

i) Algorithm for PUSH (inserting an item into the stack) operation:
This algorithm adds or inserts an item at the top of the stack

1. [Check for stack overflow?]
if top=MAXSIZE-1 then
print "Stack Overflow" and Exit
else
Set top=top+1 [Increase top by 1]
Set Stack[top]:= item [Inserts item in new top position]

2. Exit

ii) Algorithm for POP (removing an item from the stack) operation
This algorithm deletes the top element of the stack and assign it to a variable *item*

1.[Check for the stack Underflow]
If top<0 then
print "Stack Underflow" and Exit
else
[Remove the top element]
Set item=Stack [top]
[Decrement top by 1]
Set top=top-1
Return the deleted item from the stack
2. Exit

Program:

```
/* Array Implementation of Stack */
#include<stdio.h>
#include<conio.h>
#define MAX 100

struct stack
{
	int items[MAX];	//Declaring an array to store items
	int top;		//Top of a stack
};

typedef struct stack st;
void create_empty_stack(st *s);  //function prototype
void push(st *s, int);
void pop(st *s);
void display(st *s);

//Main Function
void main()
{
	int element, choice;
	st *s;
	int flag=1;
	clrscr();
	create_empty_stack(s);	/* s->top=-1; indicates empty stack */
	do
	{
		printf("\n\n Enter your choice");
		printf(" \n\n\t 1:Push the elements");
		printf(" \n\n\t 2: To display the elements");
		printf(" \n\n\t 3: Pop the element");
		printf(" \n\n\t 4: Exit");
		printf("\n\n\n Enter of your choice:\t");
		scanf("%d",&choice);
		clrscr();
		switch(choice)
		{
		case 1:
			printf("\n Enter the number:");
			scanf("%d", &element); /*Read an element from keyboard*/
			push(s,element);
			break;
		case 2:
			display(s);
			break;
		case 3:
			clrscr();
			pop(s);
			break;
		case 4:
			flag=0;
			break;
			default:
			printf("\n Invalid Choice");
		}
```

```
	}while(flag);
getch();
}

/*Function to create an empty stack*/
void create_empty_stack(st *s)
{
	s->top=-1;
}

/*Function to check whether the stack is empty or not */
int isempty(st *s)
{
	if(s->top==-1)
		return 1;
	else
		return 0;
}

/*function to check whether the stack is full or not*/
int isfull(st *s)
{
	if(s->top==MAX-1)
		return 1;
	else
		return 0;
}


/* push() function definition */
void push(st *s, int element)
{
	if(isfull(s))		/* Checking Overflow condition */
		printf("\n \nThe stack is overflow: Stack Full!!\n");
	else
		s->items[++(s->top)]=element;
}

/* Function for displaying elements of a stack*/
void display(st *s)
{
	int i;
	if(isempty(s))
		printf("\nThe Stack does not contain any Elements");
	else
	{
		printf("\nThe elements in the stack is/are:\n");
		for(i=s->top;i>=0;i--)
		printf("%d\n",s->items[i]);
	}
}

/* the POP function definition*/
void pop(st *s)
{
	if(isempty(s))
		printf("\n\nstack Underflow: Empty Stack!!!");
```

```
        else
                printf("\n\nthe deleted item is %d:\t",s->items[s->top--]);
}
```

Output:

Result Analysis: