PROLOG ACADEMY

DATA STRUCTURE

By- MASEERA ALI

 Book followed - Data structures by Seymour Lipschutz (Schaum Series)

LET’S START!

STACK

- Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO(Last In First Out) or FILO(First In Last Out).
- Basic operations are -
- Push: Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
- Pop: Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.
- Peek: Get the topmost item.
- Implementation:
- There are two ways to implement a stack:
- Using array
- Using linked list

Applications of stack

- Balancing of symbols
- Infix to Postfix/Prefix conversion
- Redo-undo features at many places like editors, photoshop.
- Forward and backward feature in web browsers
- Used in many algorithms like Tower of Hanoi, tree traversals, stock span problem, histogram problem.

Other applications can be Backtracking, Knight tour problem, rat in a maze, N queen problem and sudoku solver

IMPLEMENTATION IN C

```
#include <stdio.h>
#include <stdlib.h>
#define maxsize 30
int stack [maxsize],top=-1;
void display (){
int i;
printf("\nElements in stack are:\n");
for(i=0;i<=top;i++){
printf("%d\t",stack[i]);
}
}
```

```
void push (int no)
{
if (top==maxsize-1)
{
printf ("\n stack overflow");
return;
}
top++;
stack[top]=no;
display();
}
```

```
int pop ()
{
int no;
if (top==-1)
{
printf ("\n stack underflow");
return -1;
}
no=stack[top];
top--;
display();
return no;
}
```

```
void main ( )
{
        int n,no;
        while(n)
        {
                printf ("\nEnter your choice\n1.Push\n2.Pop\n3.Exit\n");
                fflush(stdin);
                scanf ("%d", &n);
                switch(n)
                {
                        case 1: printf ("Enter number: \n");
                                        scanf ("%d", &no);
                                        push (no);
                                break;
                        case 2: no=pop ();
                                break;
                        default:exit (0);
                }
    }
}
```

- Infix expression: The expression of the form a op b. When an operator is in-between every pair of operands.
- Postfix expression: The expression of the form a b op. When an operator is followed for the pair of operands.
- Prefix expression: The expression of the form op a b . When an operator is present before the pair of operands.

Why postfix representation of the expression?

The compiler scans the expression either from left to right or from right to left.
Consider the below expression: a + b* c + d
The compiler first scans the expression to evaluate the expression b * c, then again scan the expression to add a to it. The result is then added to d after another scan.
The repeated scanning makes it very in-efficient. It is better to convert the expression to postfix(or prefix) form before evaluation.