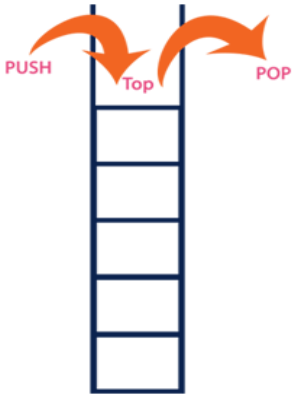
Practical # 07

Objective:

Define *Stack and* ***discuss*** *all the operations performed on Stack.*

Theory:

In this Lab, we discuss the stack data type. This data type supports Last In, First Out (LIFO) data access. Contiguous (array) and linked structures are used for implementations.



Basic Operations

Here are the minimal operations we'd need for an abstract stack (and their typical names):

- Push( ): Places a value/object on the Top of the stack.
- Pop( ): Removes a value/object from the Top of the stack and produces that value/object.
- IsEmpty( ): Reports whether the stack is empty or not.
- IsFull( ): Reports whether the stack is Full or not (for array implementation).
- Peak( ) : produces Top value/object of the stack without removing it.
- Traverse( ) : visit all elements from Top to Bottom without removing them.

Stack data structure can be implemented in two ways. They are as follows:

1. **Using Array**
2. **Using Linked List**

Lab Objectives:

- To be able to perform fundamental operations on stack.

Stack Operations using Array

A stack can be implemented using array as follows:

Before implementing actual operations: first follow the below steps to create an empty stack.

- **Step 1 -** Include all the **header files** which are used in the program and define a constant **'SIZE'** with specific value.
- **Step 2 -** Declare all the **functions** used in stack implementation.
- **Step 3 -** Create a one dimensional array with fixed size (**int stack[SIZE]**)
- **Step 4 -** Define a integer variable **'top'** and initialize with **'-1'**. (**int top = -1**)
- **Step 5 -** In main method, display menu with list of operations and make suitable function calls to perform operation selected by the user on the stack.

push(value) - Inserting value into the stack

In a stack, push() is a function used to insert an element into the stack. In a stack, the new element is always inserted at **top** position. Push function takes one integer value as parameter and inserts that value into the stack. We can use the following steps to push an element on to the stack.

- **Step 1 -** Check whether **stack** is **FULL**. (**top == SIZE-1**)
- **Step 2 -** If it is **FULL**, then display **"Stack is FULL!!! Insertion is not possible!!!"** and terminate the function.
- **Step 3 -** If it is **NOT FULL**, then increment **top** value by one (**top++**) and set stack[top] to value (**stack[top] = value**).

pop() - Delete a value from the Stack

In a stack, pop() is a function used to delete an element from the stack. In a stack, the element is always deleted from **top** position. Pop function does not take any value as parameter. We can use the following steps to pop an element from the stack...

- **Step 1 -** Check whether **stack** is **EMPTY**. (**top == -1**)
- **Step 2 -** If it is **EMPTY**, then display **"Stack is EMPTY!!! Deletion is not possible!!!"** and terminate the function.
- **Step 3 -** If it is **NOT EMPTY**, then delete **stack[top]** and decrement **top** value by one (**top--**).

display() - Displays the elements of a Stack

We can use the following steps to display the elements of a stack...

- **Step 1 -** Check whether **stack** is **EMPTY**. (**top == -1**)
- **Step 2 -** If it is **EMPTY**, then display **"Stack is EMPTY!!!"** and terminate the function.
- **Step 3 -** If it is **NOT EMPTY**, then define a variable '**i**' and initialize with top. Display **stack[i]** value and decrement **i** value by one (**i--**).
- **Step 3 -** Repeat above step until **i** value becomes '0'.

C++ program: Write C++ program to implement stack using Array.

```
#include <iostream>
using namespace std;
int stack[100], n=100, top=-1;
void push(int val) {
   if(top>=n-1)
   cout<<"Stack Overflow"<<endl;
   else {
      top++;
      stack[top]=val;
   }
}
void pop() {
   if(top<=-1)
   cout<<"Stack Underflow"<<endl;
   else {
      cout<<"The popped element is "<< stack[top] <<endl;
      top--;
   }
}
void display() {
   if(top>=0) {
      cout<<"Stack elements are:";
      for(int i=top; i>=0; i--)
      cout<<stack[i]<<" ";
      cout<<endl;
   } else
   cout<<"Stack is empty";
}
int main() {
   int ch, val;
   cout<<"1) Push in stack"<<endl;
   cout<<"2) Pop from stack"<<endl;
   cout<<"3) Display stack"<<endl;
   cout<<"4) Exit"<<endl;
   do {
      cout<<"Enter choice: "<<endl;
      cin>>ch;
      switch(ch) {
         case 1: {
            cout<<"Enter value to be pushed:"<<endl;
            cin>>val;
            push(val);
            break;
         }
         case 2: {
            pop();
            break;
         }
         case 3: {
            display();
            break;
         }
         case 4: {
            cout<<"Exit"<<endl;
            break;
         }
         default: {
            cout<<"Invalid Choice"<<endl;
         }
      }
   }while(ch!=4);
    return 0;
}
```

OUTPUT

```
1) Push in stack
2) Pop from stack
3) Display stack
4) Exit
Enter choice:
1
Enter value to be pushed:
10
Enter choice:
1
Enter value to be pushed:
20
Enter choice:
3
Stack elements are:20 10
Enter choice:
2
The popped element is 20
Enter choice:
3
Stack elements are:10
Enter choice:
4
Exit
```

Review Questions/ Exercise:

1. Explain the procedure of stack implementation using linked list.

   _____

2. Write C++ program to implement the stack using linked list

   _____

   _____

3. Write a C++ program that uses stack operations to convert a given infix expression into its postfix equivalent, Implement the stack using an array.

   _____

Name: _____

Roll #: _____

Date: _____

Subject Teacher

Remarks: