DSA SERIES

- Learn Coding

Topic to be Covered today

Stack

LETS START TODAY'S LECTURE

Stack

- Linear Data structure.
- It follows the principle of the LIFO.
- The element inserted last is the first to be removed.

- Stack internally behaves like a container where only the top element is accessible.

- Stack can be implanted using the array , linked list , stl ,,etc.

Implementation using the array

```
#include<iostream>
using namespace std;

class Stack{
    private:
    int *arr;
    int top ;
    int size;

    public:
    // Constructor
    Stack(int capacity){
        size = capacity;
        top=-1;
        arr = new int[size];
    }
```

```
void push(int val){
    if(top>=size-1){
        cout<<"Stack overflow\n";
        return ;
    }

    top++;
    arr[top] = val;
}

void pop(){
    if(top == -1){
        cout<<"Stack is empty  ";
    }
    top--;
}

int peek(){
    if(top == -1){
        cout<<"stack is empty !"<<endl;
        return -1;
    }
    return arr[top];
}
```

```
    bool isEmpty(){
        if(top==-1){
            return true;
        }
        return false;
    }

    void display(){
        if(top == -1){
            cout<<"the stack is empty.";
            return;
        }

        cout<<"Stack elements (top to bottom) :";
        for(int i =top;i>=0;i--){
            cout<<arr[i]<<" ";
        }
        cout<<endl;
    }


};
```

```
int main(){
    Stack s(5);

    s.push(10);
    s.push(20);
    s.push(30);
    s.push(40);
    s.display();

    cout<<"\ntop element : "<<s.peek()<<endl;

    s.pop();
    cout<<"\ntop element : "<<s.peek()<<endl;

    cout<<"Stack empty situation : "<<s.isEmpty();
}
```

Implementation using the Linked list

```
#include<iostream>
using namespace std;


struct Node {
    int data;
    Node* next;
};

Node* top = nullptr;

void push(int val){
    Node* newNode = new Node;
    newNode->data = val;
    newNode->next = top;
    top=newNode;
    cout<<val<<" is pushed into the stack."<<endl;
}
```

```
void pop(){
    if(top == NULL){
        cout<<"Stack is empty ";
        return ;
    }
    Node* temp = top;
    cout<<top->data <<" popped from the
stack.\n";
    top = top->next;
    delete temp;
}

int peek(){
    if(top == NULL){
        cout<<"Stack is empty ";
        return -1;
    }
    return top->data;
}
```

```
bool isEmpty(){
    if(top == NULL){
        cout<<"Stack is empty ";
        return true;
    } else{
        return false;
    }

    // return top==NULL;
}

void display(){
    if(top == NULL){
        cout<<"Stack is empty ";
        return ;
    }

    Node* temp = top;
    cout<<"Stack elements :"<<endl;
    while(temp!=NULL){
        cout<<temp->data<<" ";
        temp= temp->next;
```

```
    }
    cout<<endl;
}

int main(){
    push(10);
    push(20);
    push(30);
    push(40);
    display();
    cout<<"top element : " <<peek()<<endl;
    // cout<<"top element : " <<peek()<<endl;
    pop();
    pop();
    pop();
    pop();
    display();
    cout<<endl;
    cout<<isEmpty();
}
```

Implementation using the STL

```
#include<iostream>
#include<stack>

using namespace std;

void display(stack<int> s){
    while(!s.empty()){
        cout<<s.top()<<" ";
        s.pop();
    }
    cout<<endl;
}
int main(){
    stack<int> s;
    s.push(10);
    s.push(20);
    s.push(30);
    s.push(40);
    display(s);
    cout<<"Top element : "<<s.top()<<endl;
```

```
    s.pop();

    cout<<"Top element : "<<s.top()<<endl;
    display(s);

}
```

Learn coding

THANK YOU