**STACKS**

Stacks are a type of container adaptors with LIFO(Last in First Out) type of working, where a new element is added at one end and (top) an element is removed from that end only. stack <int> s;



Inserting element:

s.push(10);

s.push(30);

s.push(20);

s.push(5);

s.push(1)

The stack is : 1 5 20 30 10

//element is not sorted. Top most element is 1 because it was taken last . and the last element is 10 because it was taken first.

s.size() : 5

s.top() : 1

//removing the top element of stack

s.pop() : 5 20 30 10

**Working of Stack Data Structure**

The operations work as follows:

1. A pointer called TOP is used to keep track of the top element in the stack.
2. When initializing the stack, we set its value to -1 so that we can check if the stack is empty by comparing TOP == -1.
3. On pushing an element, we increase the value of TOP and place the new element in the position pointed to by TOP.
4. On popping an element, we return the element pointed to by TOP and reduce its value.
5. Before pushing, we check if the stack is already full
6. Before popping, we check if the stack is already empty



#include <stdlib.h>

#define MAX 10

int size = 0;

// Creating a stack

struct stack {

int items[MAX];

int top;

};

typedef struct stack st;

void createEmptyStack(st \*s) {

s->top = -1;

}

// Check if the stack is full

int isfull(st \*s) {

if (s->top == MAX - 1)

return 1;

else

return 0;

}

// Check if the stack is empty

int isempty(st \*s) {

if (s->top == -1)

return 1;

else

return 0;

}

// Add elements into stack

void push(st \*s, int newitem) {

if (isfull(s)) {

printf("STACK FULL");

} else {

s->top++;

s->items[s->top] = newitem;

}

size++;

}

// Remove element from stack

void pop(st \*s) {

if (isempty(s)) {

printf("\n STACK EMPTY \n");

} else {

printf("Item popped= %d", s->items[s->top]);

s->top--;

}

size--;

cout << endl;

}

// Print elements of stack

void printStack(st \*s) {

printf("Stack: ");

for (int i = 0; i < size; i++) {

cout << s->items[i] << " ";

}

cout << endl;

}

// Driver code

int main() {

int ch;

st \*s = (st \*)malloc(sizeof(st));

createEmptyStack(s);

push(s, 1);

push(s, 2);

push(s, 3);

push(s, 4);

printStack(s);

pop(s);

cout << "\nAfter popping out\n";

printStack(s);

}

#include <iostream>

#include <stack>

using namespace std;

// Function to print all elements which are greater than all

// elements present to its right

void find(int arr[], int n)

{

// create an empty stack

stack<int> stk;

// do for each element

for (int i = 0; i < n; i++)

{

// pop all the elements that are less than the current element

while (!stk.empty() && stk.top() < arr[i])

stk.pop();

// push current element into the stack

stk.push(arr[i]);

}

// print all elements in stack

while (!stk.empty())

{

cout << stk.top() << " ";

stk.pop();

}

}

int main()

{

int arr[] = { 10, 4, 6, 3, 5 };

int n = sizeof(arr)/sizeof(arr[0]);

find(arr, n);

return 0;

}