**Practical - 4**

**Aim:** implementation of stack using array and linked list.

**Description:**

Stack is a linear data structure that 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).

There are many real-life examples of a stack. Consider an example of plates stacked over one another in the canteen. The plate which is at the top is the first one to be removed, i.e. the plate which has been placed at the bottommost position remains in the stack for the longest period of time. So, it can be simply seen to follow LIFO(Last In First Out)/FILO(First In Last Out) order.

Primary Stack Operations:

void push(int data): an element is inserted into the stack.

int pop(): an element is removed from the top of the stack and is returned.

What is meant by Top of the Stack?

The pointer through which the elements are accessed, inserted, and deleted in the stack is called the top of the stack. It is the pointer to the topmost element of the stack.

* Application of Stack Data Structure:

Stack is used for evaluating expressions with operands and operations.

Matching tags in HTML and XML

Undo function in any text editor.

Infix to Postfix conversion.

Stacks are used for backtracking and parenthesis matching.

* Advantages of Stack:

Stack helps in managing data that follows the LIFO technique.

Stacks are used for systematic Memory Management.

It is used in many virtual machines like JVM.

* Disadvantages of Stack:

Stack memory is of limited size.

The total size of the stack must be defined before.

If too many objects are created then it can lead to stack overflow.

Random accessing is not possible in stack.

If the stack falls outside the memory it can lead to abnormal termination.

1. **Stack using array**

Code:

#include <iostream>

#include <stdlib.h>

using namespace std;

template<typename T>

class Stack{

T \*arr;

int size, top = -1;

public:

Stack(int s){

size = s;

arr = new T[size];

}

bool isEmpty(){

return top == -1? true: false;

}

bool isFull(){

return top == size - 1? true: false;

}

void push(){

if(!isFull()){

T data;

cout << "enter element : ";

cin >> data;

arr[++top] = data;

cout << arr[top] <<" inserted into Stack"<<endl;

}else{

cout << "Overflow!!"<<endl;

}

}

void pop(){

if(!isEmpty()){

cout << arr[top] <<" removed from Stack"<<endl;

top--;

}else{

cout << "Underflow!!"<<endl;

}

}

string toString(){

if(this->isEmpty()){

return "Stack is empty!\n";

}

string str = "stack [";

for(int i = this->top; i >= 0; i--){

str.append(arr[i]);

if(i != 0)

str.append(", ");

}

return str.append("]\n");

}

};

void utilStack(){

int choice = 1, size;

cout << "enter stack size : ";

cin >> size ;

Stack<string> \*s = new Stack<string>(size);

while(choice == 1 || choice == 2 || choice == 3 ){

cout << "Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}\nchoice : ";

cin >> choice;

switch(choice){

case 1: s->push(); break;

case 2: s->pop(); break;

case 3: cout << s->toString(); break;

}

}

cout << "closing the application..." << endl;

}

int main(){

utilStack();

return 0;

}

Output:

| enter stack size : 3  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 1  enter element : first  first inserted into Stack  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 1  enter element : second  second inserted into Stack  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 1  enter element : third  third inserted into Stack  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 3  stack [third, second, first]  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 2  third removed from Stack  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 2  second removed from Stack  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 2  first removed from Stack  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 2  Underflow!!  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit} |
| --- |

1. **Stack using linked list**

#include <iostream>

using namespace std;

template<typename T>

class Stack{

class Node{

public:

T data;

Node \*next;

Node(int data){

this->data = data;

this->next = NULL;

}

};

public:

Node \*head = NULL;

int cnt = 0;

bool isEmpty(){

return this->head == NULL?true:false;

}

int size(){

return this->cnt;

}

void push(T data){

Node \*n = new Node(data);

if(this->isEmpty()){

this->head = n;

}else{

n->next = this->head;

this->head = n;

}

cnt++;

}

bool pop(){

if(this->isEmpty())

return false;

cout << head->data <<" removed from the stack" << endl;

Node \*temp = head;

head = head->next;

delete temp;

return true;

}

string toString(){

if(this->isEmpty()){

return "Stack is empty!\n";

}

Node \*temp = head;

string str = "stack [";

while(temp != NULL){

str = str.append(to\_string(temp->data));

str = str.append(", ");

temp = temp -> next;

}

str = str.substr(0, str.length()-2);

str.append("]\n");

return str;

}

};

void utilStack(){

int choice = 1;

Stack<int> \*s = new Stack<int>();

while(choice == 1 || choice == 2 || choice == 3 ){

cout << "Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}\nchoice : ";

cin >> choice;

switch(choice){

case 1: {

int data;

cout << "enter element : ";

cin >> data;

s-> push(data);

}; break;

case 2: if(!s->pop())

cout <<"underflow!" << endl;

break;

case 3: cout << s->toString(); break;

}

}

cout << "closing the application..." << endl;

}

int main(int argc, char const \*argv[]){

utilStack();

return 0;

}

Output:

| Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 1  enter element : 23  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 1  enter element : 43  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 1  enter element : 54  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 1  enter element : 33  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 3  stack [33, 54, 43, 23]  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 2  33 removed from the stack  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 2  54 removed from the stack  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 3  stack [43, 23]  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 2  43 removed from the stack  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 2  23 removed from the stack  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 2  underflow!  Stack Operations {1 : push, 2 : pop, 3 : display, other : exit}  choice : 4  closing the application... |
| --- |

**The complexity of Stack:**

**Time complexity -**

|  | Access | Search | Insert | Delete |
| --- | --- | --- | --- | --- |
| Average | O(n) | O(n) | O(1) | O(1) |
| Worst | O(n) | O(n) | O(1) | O(1) |

**Space Complexity -** O(n)

**Conclusion:** stack implemented using array and list successfully.