210636R

Implementing stack using array

#include <iostream>

#include <cstdlib> // for rand() and srand()

#include <ctime> // for time()

using namespace std;

class Stack {

private:

int top;

int size;

int count;

int\* array;

public:

Stack(int size) {

this->size = size;

top = -1;

count=0;

array = new int[size];

}

//to push a value to the top of the stack

void push(int item) {

if (top == size - 1) {

cout << "Stack overflow" << endl;

return;

}

array[++top] = item;

count++;

}

//to pop a value

int pop() {

if (top == -1) {

cout << "Stack underflow" << endl;

return -1;

}

count--;

return array[top--];

}

//to check whether stack is empty

bool isEmpty() {

if(top==-1)

{

cout<<"Stack is empty"<<endl;

}

else cout<<"Stack is not empty"<<endl;

return top == -1;

}

bool isFull(){

if(top==size-1)

cout<<"Stack is overflow"<<endl;

else

cout<<"Stack is not full"<<endl;

return top==(size-1);

}

// to return the top element of the stack

int StackTop() {

if (top == -1) {

cout << "Stack is empty" << endl;

return -1;

}

return array[top];

}

//display whole stack

void display()

{

for(int j=count-1;j>=0;j--){

cout<<array[j]<<" ";

}

cout<<endl;

}

};

int main() {

srand(time(NULL)); // seed the random number generator with the current time

int size;

cout << "Enter the size of the array: ";

cin >> size;

Stack S(size);

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

int random\_value = rand() % 500;

// generate a random value between 0 and 500

S.push(random\_value);

}

S.push(8);

S.push(10);

S.push(5);

S.push(11);

S.push(15);

S.push(23);

S.push(6);

S.push(18);

S.push(20);

S.push(17);

S.display();

S.pop();

S.pop();

S.pop();

S.pop();

S.pop();

S.display();

S.push(4);

S.push(30);

S.push(3);

S.push(1);

S.display();

return 0;

}

Implementing stack using Linked list

#include <bits/stdc++.h>

using namespace std;

struct Node

{

int data;

Node\*next;

};

class Stack

{

Node\*head;

int stackSize;

public:

Stack()

{

head=NULL;

stackSize=0;

}

void push(int k)

{

Node\*temp= new Node();

temp-> data = k;

temp->next=head;

head=temp;

cout<<"Pushed "<<k<< " in to the stack"<<endl;

stackSize++;

}

void pop()

{

if(head==NULL){

cout<<"Stack is underflow"<<endl;

}

Node\*temp=head;

head= temp->next;

temp->next=NULL;

delete temp;

stackSize--;

}

int StackTop()

{

if(head==NULL)

{

cout<<"stack is empty"<<endl;

return -1;

}

cout<<"Top element is "<<head->data<<endl;

return head->data;

}

int size()

{

cout<<"size: "<<stackSize<<endl;

return stackSize;

}

bool isEmpty()

{

if(head==NULL)

{

cout<<"Stack is Empty"<<endl;

}

else cout<<"stack is not Empty"<<endl;

return head==NULL;

}

bool isFull() {

cout<<"Stack is not full"<<endl;

return false; // Linked list implementation of a stack cannot be full

}

void display() {

Node\* current = head;

while (current != NULL) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

};

int main()

{

Stack S;

S.push(8);

S.push(10);

S.push(5);

S.push(11);

S.push(15);

S.push(23);

S.push(6);

S.push(18);

S.push(20);

S.push(17);

S.display();

S.pop();

S.pop();

S.pop();

S.pop();

S.pop();

S.display();

S.push(4);

S.push(30);

S.push(3);

S.push(1);

S.display();

return 0;

}