THE UNIVERSITY OF BUEA

FACULTY OF ENGINEERING AND

TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

COURSE CODE AND COURSE TITLE: CEF 341/ALGORITHMS AND DATA STRUCTURE .

ACADEMIC YEAR: 2024/2025

REPORT TITLE: DATA STRUCTURE

NAME: TEBOH MANOIELLAA EFON

MATRICLE: FE23A167

**DATA STRUCTURE**

Data structure is a way of storing and organizing data efficiently such that the required operation on them can be perform efficiently with respect to time as well as memory.

* Stack

A stack is a dynamic endless list of items in which you can only insert or remove an element at one end called the **head** or **top** of the stack

* **Operation on a stack.**

1. **Push:** This is the act of inserting an element in the stack.

**Function to push**

void push(int value) {

if (top == MAX - 1) {

printf("Stack overflow. Cannot push %d onto the stack.\n", value);

} else {

top++;

stack[top] = value;

printf("%d pushed onto the stack.\n", value);

}

}

1. **Pop:** This is the act of removing an element from the stack.

**Function to pop**

void pop() {

if (top == -1) {

printf("Stack underflow. No elements to pop.\n");

} else {

int poppedValue = stack[top];

top--;

printf("%d popped from the stack.\n", poppedValue);

}

}

1. **StackisEmpty:** checking if the stack is empty.

**Function stackisEmpty**

int isEmpty(){

if(top==-1){

printf(" %d\n");

return 0;}

}

1. **StackisFull:** checking is stack is full.

int isFull(){

if(top==N-1){

printf("stack is full!\n");

return 0;

}}

Function to display all elements of the stack

void display() {

if (top == -1) {

printf("The stack is empty.\n");

} else {

printf("Stack elements are: ");

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

printf("%d ", stack[i]);

}

printf("\n");

}

}

**Function to get the current size of the stack**

void sizeOfStack() {

printf("Current size of the stack is %d.\n", top + 1);

}

**Function to get the top element of the stack**

void topOfStack() {

if (top == -1) {

printf("The stack is empty.\n");

} else {

printf("Top element of the stack is %d.\n", stack[top]);

}

}

The entire program is as seen bellow.

#include <stdio.h>

#include <stdlib.h>

#define MAX 100 // Maximum size of the stack

int stack[MAX];

int top = -1;

// Function to create a stack (resetting it)

void create() {

top = -1;

printf("Stack created/reset successfully.\n");

}

// Function to push an element onto the stack

void push(int value) {

if (top == MAX - 1) {

printf("Stack overflow. Cannot push %d onto the stack.\n", value);

} else {

top++;

stack[top] = value;

printf("%d pushed onto the stack.\n", value);

}

}

// Function to pop an element from the stack

void pop() {

if (top == -1) {

printf("Stack underflow. No elements to pop.\n");

} else {

int poppedValue = stack[top];

top--;

printf("%d popped from the stack.\n", poppedValue);

}

}

// Function to display all elements of the stack

void display() {

if (top == -1) {

printf("The stack is empty.\n");

} else {

printf("Stack elements are: ");

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

printf("%d ", stack[i]);

}

printf("\n");

}

}

// Function to get the current size of the stack

void sizeOfStack() {

printf("Current size of the stack is %d.\n", top + 1);

}

// Function to get the top element of the stack

void topOfStack() {

if (top == -1) {

printf("The stack is empty.\n");

} else {

printf("Top element of the stack is %d.\n", stack[top]);

}

}

int main() {

int choice, value;

create(); // Initialize the stack at the start

do {

printf("\nChoose an operation:\n");

printf("1: Push\n");

printf("2: Pop\n");

printf("3: Display\n");

printf("4: Size of Stack\n");

printf("5: Top of Stack\n");

printf("0: Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter value to push: ");

scanf("%d", &value);

push(value);

break;

case 2:

pop();

break;

case 3:

display();

break;

case 4:

sizeOfStack();

break;

case 5:

topOfStack();

break;

case 0:

printf("Exiting program.\n");

break;

default:

printf("Invalid choice. Please try again.\n");

}

} while (choice != 0);

return 0;

}

