**ASSIGNMENT**

**SUBMITTED TO:** Mr Sajid Maqbool

**SUBMITTED BY:** Abdur-Rehman Munir

**ROLL NO:** 2K23-BSCS-524

**SECTION:** M

**DEPARTMENT:** COMPUTER SCIENCE

**Session:**  2K23 – 2K27

**Semester:** 3rd

**Subject:** Data Structures



**NFC Institute of Engineering and Technology**

### ****Stack Implementation Using Array****

**Code:1**

#include<stdio.h>

//Implementing Stack using Array

//if the value of top is -1 then stack is empty

//if the value of top is capacity-1 then stack is full

//Global Variables

int stack[10], n = 10, top = -1;

//Push: Adding element to stack

void push(int val)

{

if (top >= n - 1)

{

printf("Stack is Full");

}

//top=-1, Top+1=

//top=0

//stack[0]=val

else

{

top++;

stack[top] = val;

}

}

//Pop, Removing Element from the stack

void pop()

{

if (top <= -1)

{

printf("Stack is Empty\n");

}

else

{

printf("The popped element in the stack is %d\n", stack[top]);//stack[0]

top--;//0-1=-1

}

}

//Print the all elements of stack

void display() {

if (top >= 0)

{

printf("Stack elements are: ");

//Loop for printing array elements

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

{

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

}

printf("\n");

}

else

{

printf("Stack is empty\n");

}

}

//Main Function

//if value of f is

//1-Push

//2-pop

//3-Display

//4-exit

//5-retry

int main()

{

int f, value;

do

{

scanf("%d", &f);

if (f == 1)

{

printf("Enter the value to push: ");

scanf("%d", &value);

push(value);

}

else if (f == 2)

{

pop();

}

else if (f == 3)

{

display();

}

else if (f == 4)

{

printf("Exiting program.\n");

}

else

{

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

}

}

while (f != 4);

return 0;

}

### **stack2**Code output:****

### ****Explanation of the Code:****

**Push Function (**push(int val)**)**:

* 1. This function adds a value to the stack. It checks whether the stack is full (i.e., top >= n - 1). If it is full, it prints a message indicating the stack cannot accept more elements. Otherwise, it increments the top index and stores the value in the stack array.

**Pop Function (**pop()**)**:

* 1. This function removes and prints the top value from the stack. It first checks if the stack is empty (i.e., top <= -1). If the stack is empty, it prints an error message. Otherwise, it prints the element at the top of the stack and then decrements the top index to remove it from the stack.

**Display Function (**display()**)**:

* 1. This function prints all the elements in the stack from the top to the bottom. If the stack is empty, it displays a message saying "Stack is empty."

**Main Function (**main()**)**:

* 1. The program presents a simple text-based menu where the user can choose to push a value, pop a value, display the stack, or exit the program. The program continues to run in a loop until the user chooses to exit (option 4).

This program demonstrates how to implement basic stack operations using a fixed-size array, with clear separation of functions for each stack operation. The user interface is interactive and user-friendly, making it easy to test and understand how stack operations work in C.

**C Program to Reverse a String Using a Stack Implemented with a Character Array**

This program demonstrates how to use a stack, implemented as a character array, to reverse the string of a name. It includes functions to push and pop characters from the stack, and a main function to read user input, push characters onto the stack, and display the reversed string by popping characters. The implementation avoids predefined functions for simplicity.

### *****Code:2*****

#include<stdio.h>

// Global Variables

char stack[100]; // Stack implemented as a character array

int top = -1; // Top of the stack

// Function to push a character onto the stack

void push(char val) {

if (top >= 99) { // Check if stack is full

printf("Stack is Full\n");

} else {

top++;

stack[top] = val;

}

}

// Function to pop a character from the stack

char pop() {

if (top <= -1) { // Check if stack is empty

printf("Stack is Empty\n");

return '\0'; // Return null character if empty

} else {

char val = stack[top];

top--;

return val;

}

}

// Function to reverse and display the string

void displayReverse(char str[]) {

int i = 0;

// Push all characters of the string onto the stack

while (str[i] != '\0') {

push(str[i]);

i++;

}

// Pop and print all characters from the stack to reverse the string

printf("Reversed string: ");

while (top >= 0) {

printf("%c", pop());

}

printf("\n");

}

// Main function

int main() {

char name[100];

int i = 0;

// Reading the name manually

printf("Enter your name: ");

while (1) {

char ch = getchar(); // Read one character at a time

if (ch == '\n') { // Stop reading when newline is encountered

break;

}

name[i] = ch; // Store character in name array

i++;

}

name[i] = '\0'; // Null-terminate the string

// Call function to reverse and display the string

displayReverse(name);

return 0;

}

### ****rev1****

### Explanation of the Program:

**Stack Implementation**:

* 1. The stack is implemented as a character array (stack[100]).
  2. top is used to track the index of the topmost element in the stack.

**Functions**:

* 1. push(char val): Adds a character to the stack if it's not full.
  2. pop(): Removes and returns the top character from the stack if it's not empty.
  3. displayReverse(char str[]): Pushes all characters of the input string onto the stack and then pops them to display the reversed string.

**Input Handling**:

* 1. The name is read character by character using getchar() to avoid using predefined functions like fgets() or strcspn.

**Output**:

* 1. The reversed string is displayed after all characters are popped from the stack.

### ****Enter your name: abdurrehman munir****

### ****Reversed string: rinum namherrudba****