

**Galgotias College of Engineering and Technology**

Master of Computer Application

II Semester

2023 - 2024

Data Structure and Analysis of Algorithm Lab

**Submitted by:** Akash Yadav 2300970140019

**Program: - 1**

**Objective:** Write a Program to implement addition and multiplication in Two Dimension Array.

**Code:**

#include <stdio.h>

#define ROWS 3

#define COLS 3

void addMatrices(int matrix1[ROWS][COLS], int matrix2[ROWS][COLS], int result[ROWS][COLS]) {

    for (int i = 0; i < ROWS; i++) {

        for (int j = 0; j < COLS; j++) {

            result[i][j] = matrix1[i][j] + matrix2[i][j];

        }

    }

}

void multiplyMatrices(int matrix1[ROWS][COLS], int matrix2[ROWS][COLS], int result[ROWS][COLS]) {

    for (int i = 0; i < ROWS; i++) {

        for (int j = 0; j < COLS; j++) {

            result[i][j] = 0;

            for (int k = 0; k < COLS; k++) {

                result[i][j] += matrix1[i][k] \* matrix2[k][j];

            }

        }

    }

}

void printMatrix(int matrix[ROWS][COLS]) {

    for (int i = 0; i < ROWS; i++) {

        for (int j = 0; j < COLS; j++) {

            printf("%d ", matrix[i][j]);

        }

        printf("\n");

    }

}

int main() {

    int matrix1[ROWS][COLS] = {

        {1, 2, 3},

        {4, 5, 6},

        {7, 8, 9}

    };

    int matrix2[ROWS][COLS] = {

        {9, 8, 7},

        {6, 5, 4},

        {3, 2, 1}

    };

    int result\_addition[ROWS][COLS];

    int result\_multiplication[ROWS][COLS];

    printf("Matrix 1:\n");

    printMatrix(matrix1);

    printf("\nMatrix 2:\n");

    printMatrix(matrix2);

*// Addition*

    printf("\nAddition:\n");

    addMatrices(matrix1, matrix2, result\_addition);

    printMatrix(result\_addition);

*// Multiplication*

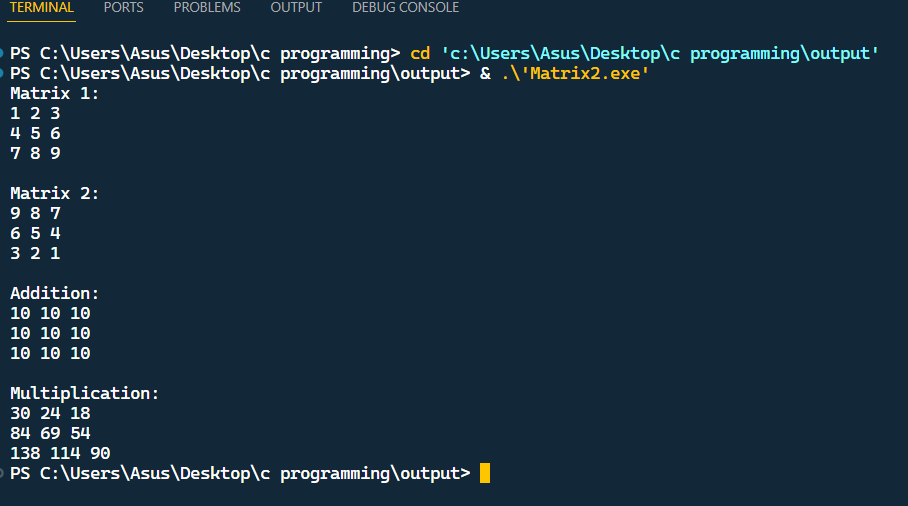
    printf("\nMultiplication:\n");

    multiplyMatrices(matrix1, matrix2, result\_multiplication);

    printMatrix(result\_multiplication);

    return 0;

}

****

**Output:-**

**Program: -2**

**Objective: -**Write a program to transpose a 2-d Array.

**Code: -**

#include <stdio.h>

#define ROWS 3

#define COLS 3

void transpose(int matrix[ROWS][COLS], int result[COLS][ROWS]) {

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

for (int j = 0; j < COLS; j++) {

result[j][i] = matrix[i][j];

}

}

}

void printMatrix(int matrix[ROWS][COLS]) {

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

for (int j = 0; j < COLS; j++) {

printf("%d ", matrix[i][j]);

}

printf("\n");

}

}

int main() {

int matrix[ROWS][COLS] = {

{1, 2, 3},

{4, 5, 6},

{7, 8, 9}

};

int result[COLS][ROWS];

printf("Original Matrix:\n");

printMatrix(matrix);

transpose(matrix, result);

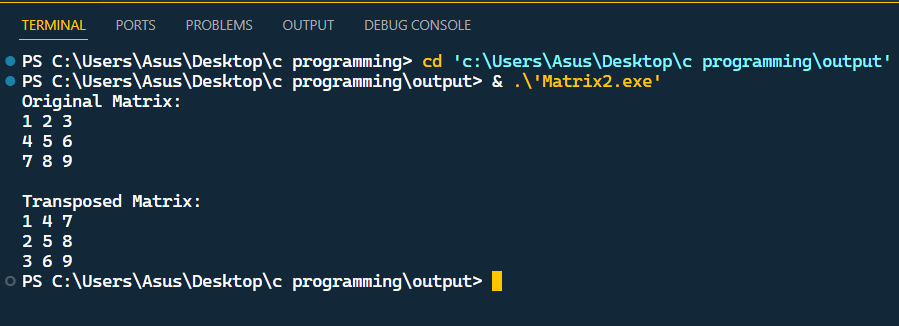
printf("\nTransposed Matrix:\n");

printMatrix(result);

return 0;

}

**Output: -**

****

**Program: -3**

**Objective: -** Write a program to implement a stack using array.

**Code: -**

#include <stdio.h>

#include <stdbool.h>

#define MAX\_SIZE 100

typedef struct {

    int arr[MAX\_SIZE];

    int top;

} *Stack*;

void initializeStack(*Stack* \*stack) {

    stack->top = -1; *// Set top to -1 to indicate an empty stack*

}

bool isEmpty(*Stack* \*stack) {

    return stack->top == -1;

}

bool isFull(*Stack* \*stack) {

    return stack->top == MAX\_SIZE - 1;

}

void push(*Stack* \*stack, int element) {

    if (isFull(stack)) {

        printf("Error: Stack is full, cannot push element.\n");

        return;

    }

    stack->top++;

    stack->arr[stack->top] = element;

}

int pop(*Stack* \*stack) {

    if (isEmpty(stack)) {

        printf("Error: Stack is empty, cannot pop element.\n");

        return -1; *// Return -1 to indicate an empty stack*

    }

    int poppedElement = stack->arr[stack->top];

    stack->top--;

    return poppedElement;

}

int peek(*Stack* \*stack) {

    if (isEmpty(stack)) {

        printf("Error: Stack is empty, cannot peek.\n");

        return -1; *// Return -1 to indicate an empty stack*

    }

    return stack->arr[stack->top];

}

int main() {

*Stack* stack;

    initializeStack(&stack);

    push(&stack, 10);

    push(&stack, 20);

    push(&stack, 30);

    printf("Top element of the stack: %d\n", peek(&stack));

    printf("Popped element: %d\n", pop(&stack));

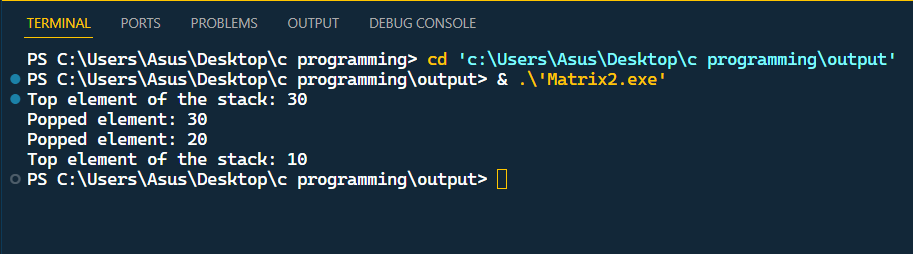
    printf("Popped element: %d\n", pop(&stack));

    printf("Top element of the stack: %d\n", peek(&stack));

    return 0;

}

**Output: -**

****

**Program:-4**

**Objective:-**Write a program to implement Queue using array.

**Code:-**

#include <stdio.h>

#include <stdlib.h>

#define MAX\_SIZE 100

typedef struct {

    int items[MAX\_SIZE];

    int front;

    int rear;

} *Queue*;

*// Function prototypes*

void initializeQueue(*Queue* \*q);

int isEmpty(*Queue* \*q);

int isFull(*Queue* \*q);

void enqueue(*Queue* \*q, int value);

int dequeue(*Queue* \*q);

void displayQueue(*Queue* \*q);

*// Initialize the queue*

void initializeQueue(*Queue* \*q) {

    q->front = -1;

    q->rear = -1;

}

*// Check if the queue is empty*

int isEmpty(*Queue* \*q) {

    return (q->front == -1 && q->rear == -1);

}

*// Check if the queue is full*

int isFull(*Queue* \*q) {

    return (q->rear + 1) % MAX\_SIZE == q->front ? 1 : 0;

}

*// Add an element to the queue*

void enqueue(*Queue* \*q, int value) {

    if (isFull(q)) {

        printf("Queue is full, cannot enqueue.\n");

        return;

    } else if (isEmpty(q)) {

        q->front = 0;

        q->rear = 0;

    } else {

        q->rear = (q->rear + 1) % MAX\_SIZE;

    }

    q->items[q->rear] = value;

}

*// Remove an element from the queue*

int dequeue(*Queue* \*q) {

    int value;

    if (isEmpty(q)) {

        printf("Queue is empty, cannot dequeue.\n");

        return -1;

    } else if (q->front == q->rear) {

        value = q->items[q->front];

        q->front = -1;

        q->rear = -1;

    } else {

        value = q->items[q->front];

        q->front = (q->front + 1) % MAX\_SIZE;

    }

    return value;

}

*// Display the elements of the queue*

void displayQueue(*Queue* \*q) {

    if (isEmpty(q)) {

        printf("Queue is empty.\n");

        return;

    }

    printf("Queue: ");

    int i = q->front;

    while (i != q->rear) {

        printf("%d ", q->items[i]);

        i = (i + 1) % MAX\_SIZE;

    }

    printf("%d\n", q->items[q->rear]);

}

*// Main function*

int main() {

*Queue* q;

    initializeQueue(&q);

*// Example usage*

    enqueue(&q, 10);

    enqueue(&q, 20);

    enqueue(&q, 30);

    displayQueue(&q);

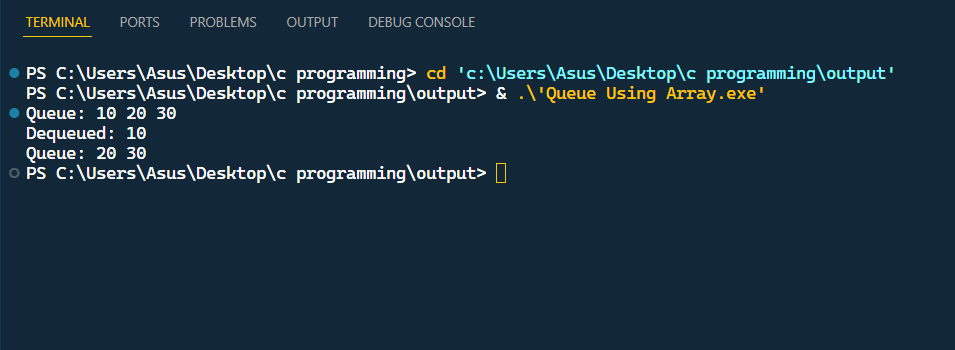
    printf("Dequeued: %d\n", dequeue(&q));

    displayQueue(&q);

    return 0;

}

**Output:-**

****