**Manav Rachna International Institute of Research and Studies**

**Bachelor's in computer applications**

**Data Structures using C**



**Submitted By:** Vivek kumar

**Department:** School of Computer Applications

**Course:** Bachelor's in computer applications

**Roll No:** 24/SCA/BCA/054

**Semester:** 2nd

**Subject:** Data Structures using C

# 1. Insertion of array

#include <stdio.h> int main() { int arr[5]; int n,count=0 , loc, upd;

printf(" enter the elements of array \n");

for(int i=0; i<5; i++){

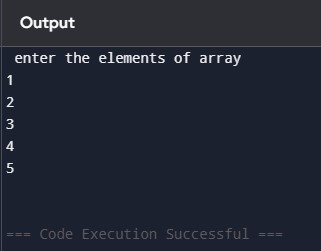
scanf("%d",&arr[i]);

}

return 0;

}

OUTPUT:



# 2.Searching on array

#include <stdio.h> int main() { int arr[5]; int n,count=0 , loc, upd;

printf(" enter the elements of array \n");

for(int i=0; i<5; i++){

scanf("%d",&arr[i]);

}

printf("enter the element you want to find\n"); scanf("%d",&n);

for(int i=0; i<5; i++){ if(arr[i] == n){

printf("%d found at location %d\n",n,i+1); count +=1;

}

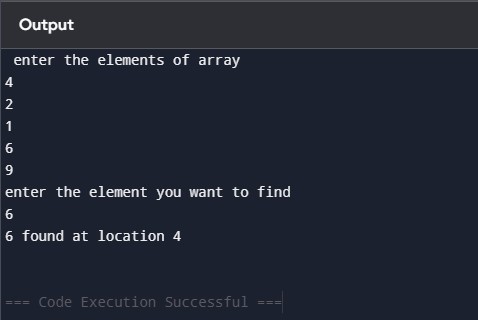
}

if(count == 0){

printf("%d not founded\n", n);

}

OUTPUT:



**3. Updating element on array:**

#include <stdio.h> int main() { int arr[5];

int n,count=0 , loc, upd;

printf(" enter the elements of array \n");

for(int i=0; i<5; i++){

scanf("%d",&arr[i]);

}

printf("enter the element you want to update\n"); scanf("%d",&upd); printf("enter the location\n"); scanf("%d",&loc);

arr[loc-1] = upd;

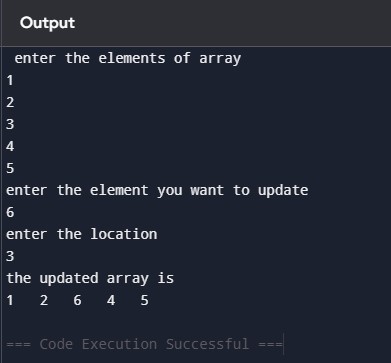
printf("the updated array is\n");

for(int i=0; i<5; i++){ printf("%d\t",arr[i]);

}

return 0;

}

OUTPUT:

## 4.Input a element and check wheather it is present in array or not if the element is present in array then print the position of element

#include <stdio.h>

int main() {

int arr[5] = {1,2,3,4,5}; int n,count=0;

printf("enter the elements to search\n"); scanf("%d",&n); for(int i=0; i<5; i++){ if(arr[i] == n){

printf("%d is present in the %d position", n,i+1); count +=1;

}

}

if (count ==0){

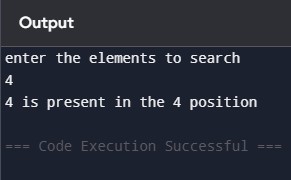
printf("%d is not present in the array\n",n);

}

return 0;

}

OUTPUT:



## 5.Sorting of element in array

#include <stdio.h>

int main() {

int arr[7];

printf(" enter seven elements \n");

for(int i=0; i<7; i++){

scanf("%d",&arr[i]);

}

int temp;

for(int i=0; i<7; i++){ for(int j=0; j<6-i; j++){ if(arr[j] > arr[j+1]){ temp = arr[j]; arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

printf("Ascending order\n"); for(int i=0; i<7; i++){ printf("%d\t",arr[i]);

}

for(int i=0; i<7; i++){ for(int j=0; j<6-i; j++){ if(arr[j] < arr[j+1]){ temp = arr[j]; arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

printf("\nDescending order\n"); for(int i=0; i<7; i++){

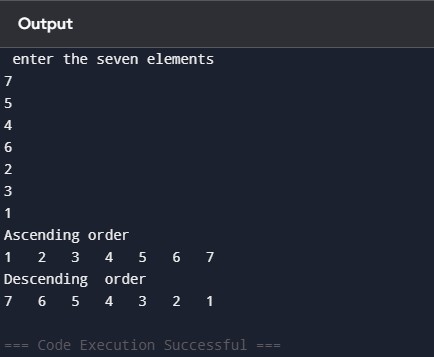
printf("%d\t",arr[i]);

}

return 0;

}

Output:



# 6.Deletion of array

#include <stdio.h> int main(){

int count = 0; int x;

int arr1[] = {1,2,3,4,5};

printf("Enter the element you want to delete: \n"); scanf("%d", &x); for(int i = 0; i<5; i++){ if(arr1[i] == x){ for(int j = i; j<5; j++){ arr1[j] = arr1[j + 1];

}

count = count + 1;

}

}

if(count == 0){

printf("Element is not found");

}

else{ for(int i = 0; i<4; i++){

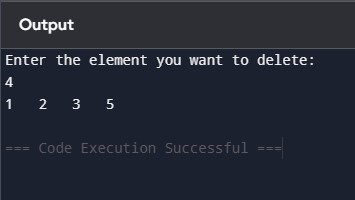
printf("%d\t", arr1[i]);

}

}

}

OUTPUT:



## 7. update any value and all the value of array

#include <stdio.h>

void main() { int n, value, index;

printf("Enter number of elements: "); scanf("%d", &n); int arr[n];

printf("Enter elements of the array:\n");

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

scanf("%d", &arr[i]);

}

printf("Enter index to update: "); scanf("%d", &index); printf("Enter new value: "); scanf("%d", &value); arr[index] = value;

// Update all elements with the same value for (int i = 0; i < n; i++) { arr[i] = value;

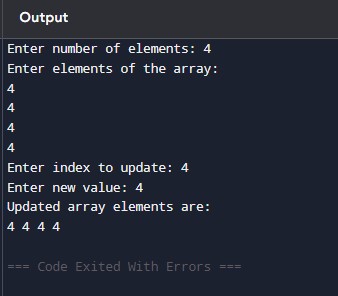
}

printf("Updated array elements are:\n"); for (int i = 0; i < n; i++) {

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

}

}



**8. Operations on Matrices: a) Addition of Matrices:**

#include <stdio.h>

void main() {

int row, col;

printf("Enter number of rows and columns: "); scanf("%d %d", &row, &col);

int matrix1[row][col], matrix2[row][col], sum[row][col];

printf("Enter elements of first matrix:\n");

for (int i = 0; i < row; i++) { for (int j = 0; j < col; j++) {

scanf("%d", &matrix1[i][j]);

}

}

printf("Enter elements of second matrix:\n");

for (int i = 0; i < row; i++) { for (int j = 0; j < col; j++) {

scanf("%d", &matrix2[i][j]);

}

}

// Adding matrices for (int i = 0; i < row; i++) { for (int j = 0; j < col; j++) {

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

}

}

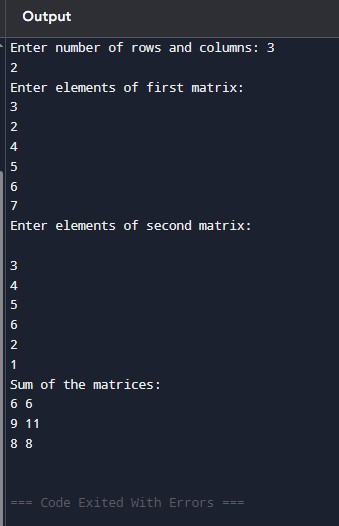
printf("Sum of the matrices:\n"); for (int i = 0; i < row; i++) { for (int j = 0; j < col; j++) { printf("%d ", sum[i][j]);

}

printf("\n");

}

}



## 9. b) Matrix Multiplication: c Copy

#include <stdio.h>

int main() { int row1, col1, row2, col2;

printf("Enter rows and columns for first matrix: "); scanf("%d %d", &row1, &col1);

printf("Enter rows and columns for second matrix: "); scanf("%d %d", &row2, &col2);

if (col1 != row2) {

printf("Matrix multiplication not possible.\n"); return 0;

}

int matrix1[row1][col1], matrix2[row2][col2], product[row1][col2];

printf("Enter elements of first matrix:\n");

for (int i = 0; i < row1; i++) { for (int j = 0; j < col1; j++) {

scanf("%d", &matrix1[i][j]);

}

}

printf("Enter elements of second matrix:\n");

for (int i = 0; i < row2; i++) { for (int j = 0; j < col2; j++) { scanf("%d", &matrix2[i][j]);

}

}

for (int i = 0; i < row1; i++) { for (int j = 0; j < col2; j++) { product[i][j] = 0; for (int k = 0; k < col1; k++) {

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

}

}

}

printf("Product of matrices:\n"); for (int i = 0; i < row1; i++) { for (int j = 0; j < col2; j++) {

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

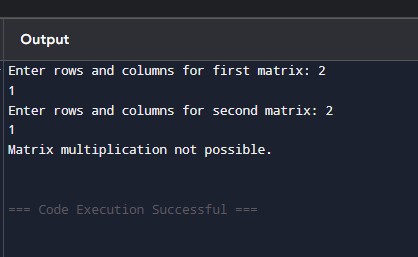
}

printf("\n");

}

return 0;

}



**10. c) Transpose of Matrix:**

#include <stdio.h>

int main() {

int row, col;

printf("Enter number of rows and columns: "); scanf("%d %d", &row, &col);

int matrix[row][col], transpose[col][row];

printf("Enter elements of the matrix:\n"); for (int i = 0; i < row; i++) { for (int j = 0; j < col; j++) {

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

}

}

// Transposing matrix for (int i = 0; i < row; i++) { for (int j = 0; j < col; j++) {

transpose[j][i] = matrix[i][j];

}

}

printf("Transpose of the matrix:\n"); for (int i = 0; i < col; i++) { for (int j = 0; j < row; j++) { printf("%d ", transpose[i][j]);

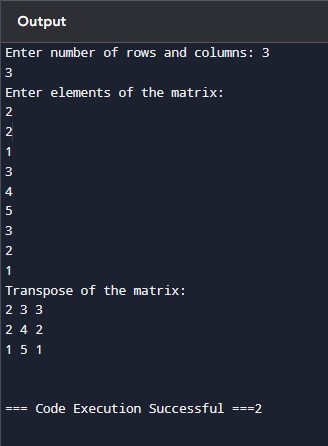
}

printf("\n");

}

return 0;

}



## 11. d) Convert into Unit Matrix (Identity Matrix)

#include <stdio.h> int main() { int row, col;

printf("Enter number of rows and columns: "); scanf("%d %d", &row, &col); if (row != col) {

printf("Identity matrix is possible only for square matrices.\n"); return 0;

}

int matrix[row][col];

// Creating identity matrix for (int i = 0; i < row; i++) { for (int j = 0; j < col; j++) {

if (i == j) matrix[i][j] = 1; else

matrix[i][j] = 0;

}

}

printf("Unit (Identity) matrix:\n");

for (int i = 0; i < row; i++) { for (int j = 0; j < col; j++) {

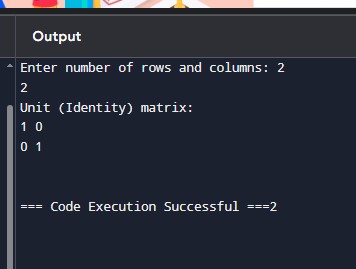
printf("%d ", matrix[i][j]);

}

printf("\n");

}

return 0;



**12. Show Sparse Matrix:**

#include <stdio.h>

void main() {

int row, col;

printf("Enter number of rows and columns: "); scanf("%d %d", &row, &col);

int matrix[row][col], count = 0;

printf("Enter elements of the matrix:\n");

for (int i = 0; i < row; i++) { for (int j = 0; j < col; j++) { scanf("%d", &matrix[i][j]); if (matrix[i][j] == 0)

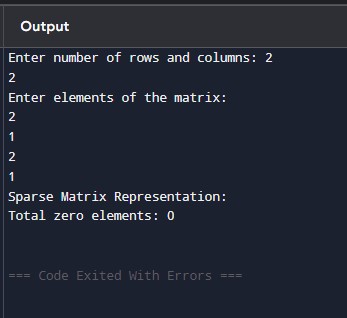
count++;

}

}

printf("Sparse Matrix Representation:\n"); printf("Total zero elements: %d\n", count);

}



## 13.Push operation in c

#include <stdio.h>

#include <stdlib.h>

#define MAX 5

struct Stack { int arr[MAX]; int top;

};

void initStack(struct Stack\* stack) {

stack->top = -1;

}

int isFull(struct Stack\* stack) {

return stack->top == MAX - 1;

}

void push(struct Stack\* stack, int value) {

if (isFull(stack)) {

printf("Stack Overflow! Cannot push %d\n", value);

} else {

stack->arr[++(stack->top)] = value; printf("%d pushed to stack\n", value);

}

}

void printStack(struct Stack\* stack) { if (stack->top == -1) { printf("Stack is empty\n");

} else {

printf("Stack elements: "); for (int i = 0; i <= stack->top; i++) {

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

}

printf("\n");

}

}

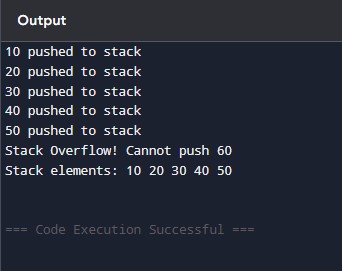
int main() { struct Stack stack; initStack(&stack); push(&stack, 10); push(&stack, 20); push(&stack, 30); push(&stack, 40); push(&stack, 50);

push(&stack, 60);

printStack(&stack);

return 0;

}



## 14.POP Operation

#include <stdio.h>

#include <stdlib.h>

#define MAX 5

struct Stack { int arr[MAX]; int top;

};

void initialize(struct Stack \*s) { s->top = -1;

}

int isFull(struct Stack \*s) {

return s->top == MAX - 1;

}

int isEmpty(struct Stack \*s) {

return s->top == -1;

}

void push(struct Stack \*s, int value) { if (isFull(s)) {

printf("Stack Overflow\n");

} else { s->arr[++(s->top)] = value;

printf("%d pushed to stack\n", value);

}

}

int pop(struct Stack \*s) { if (isEmpty(s)) {

printf("Stack Underflow\n"); return -1;

} else {

int poppedValue = s->arr[s->top--];

return poppedValue;

}

}

int peek(struct Stack \*s) { if (isEmpty(s)) { printf("Stack is empty\n"); return -1;

}

return s->arr[s->top];

}

int main() { struct Stack stack; initialize(&stack);

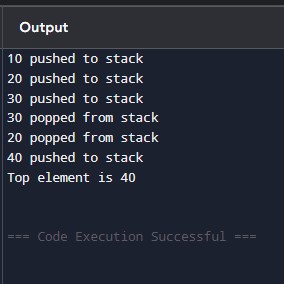
push(&stack, 10); push(&stack, 20); push(&stack, 30);

printf("%d popped from stack\n", pop(&stack)); printf("%d popped from stack\n", pop(&stack));

push(&stack, 40);

printf("Top element is %d\n", peek(&stack));

}



## 15.IMPLEMENTATION OF STACK BY USING ARRAY

#include <stdio.h>

#define MAX 5

int stack[MAX], top = -1;

void push(int x) { if (top == MAX - 1) {

printf("Stack Overflow!\n");

} else { top++; stack[top] = x;

printf("%d pushed to stack\n", x);

}

}

void pop() { if (top == -1) {

printf("Stack Underflow!\n");

} else {

printf("%d popped from stack\n", stack[top]); top--;

}

}

void display() { if (top == -1) {

printf("Stack is empty\n");

} else {

printf("Stack elements: "); for (int i = 0; i <= top; i++) {

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

}

printf("\n");

}

}

int main() { push(10); push(20); display(); pop(); display(); push(30); display(); return 0;

}

