**Question No-1 Write a Menu Driven C program to implement Array Insertion , Array Deletion , displaying the array & Exit from the menu.**

#include <stdio.h>

#define MAX\_SIZE 100

int array[MAX\_SIZE] ;

int main() {

   int n, position, element, i ;

  // Get the size of the array

  printf("Enter the size of the array: ");

  scanf("%d", &n);

  // Initialize the array

  for (i = 0; i < n; i++) {

    array[i] = 0;

  }

  // Display the menu

  printf("\n\n1. Insert element\n2. Delete element\n3. Display array\n4. Exit\n");

  // Keep looping until the user wants to exit

  while (1) {

    // Get the users choice

    printf("\nEnter your choice: ");

    scanf("%d", &i);

    // Switch on the user's choice

    switch (i) {

      case 1:

        // Insert element

        printf("\nEnter the element to be inserted: ");

        scanf("%d", &element);

        printf("\nEnter the position where the element should be inserted: ");

        scanf("%d", &position);

        insertElement(n, position, element);

        break;

      case 2:

        // Delete element

        printf("\nEnter the position of the element to be deleted: ");

        scanf("%d", &position);

        deleteElement(n, position);

        break;

      case 3:

        // Display array

        displayArray( n );

        break;

      case 4:

        // Exit

        printf("\nExiting...\n");

        return 0;

      default:

        // Invalid choice

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

    }

  }

  return 0;

}

// Function to insert element at any position in the array

void insertElement(int n, int position, int element) {

  // Check if the position is valid

  if (position < 0 || position > n) {

    printf("\nInvalid position.\n");

    return;

  }

  // Shift all the elements after the position by one

  for (int i = n - 1; i >= position; i--) {

    array[i + 1] = array[i];

  }

  // Insert the element at the specified position

  array[position] = element;

  // Increment the size of the array

  n++;

}

// Function to delete element from any position in the array

void deleteElement(int n, int position) {

  // Check if the position is valid

  if (position < 0 || position >= n) {

    printf("\nInvalid position.\n");

    return;

  }

  // Shift all the elements after the position by one

  for (int i = position; i < n - 1; i++) {

    array[i] = array[i + 1];

  }

  // Decrement the size of the array

  n--;

}

// Function to display the array

void displayArray(int n) {

  // Check if the array is empty or not

  if (n == 0) {

    printf("\nArray is empty.\n");

    return;

  }

  // Print all the elements in the array

  for (int i = 0; i < n; i++) {

    printf("%d ", array[i]);

  }

  printf("\n");

}

**OUTPUT :**

Enter the size of the array: 5

1. Insert element

2. Delete element

3. Display array

4. Exit

Enter your choice: 1

Enter the element to be inserted: 2

Enter the position where the element should be inserted: 2

Enter your choice: 1

Enter the element to be inserted: 3

Enter the position where the element should be inserted: 3

Enter your choice: 3

0 0 2 3 0

Enter your choice: 2

Enter the position of the element to be deleted: 2

Enter your choice: 3

0 0 3 0

Enter your choice: 4

Exiting...

**Question 2 : Write a Menu Driven C program to implement Stack Insertion , Deletion , Display & Exit from the Menu .**

#include <stdio.h>

#include <stdbool.h>

#define MAX\_SIZE 100

// Function prototypes

void push(int value);

int pop();

void display();

bool isFull();

bool isEmpty();

// Global stack variables

int stack[MAX\_SIZE];

int top = -1;

int main() {

    int choice, value;

    while (1) {

        // Display the menu

        printf("\nStack Menu:\n");

        printf("1. Push (Insertion)\n");

        printf("2. Pop (Deletion)\n");

        printf("3. Display\n");

        printf("4. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                // Push (Insertion) operation

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

                scanf("%d", &value);

                push(value);

                break;

            case 2:

                // Pop (Deletion) operation

                if (!isEmpty()) {

                    int deletedValue = pop();

                    printf("Popped value: %d\n", deletedValue);

                } else {

                    printf("Stack is empty. Cannot pop.\n");

                }

                break;

            case 3:

                // Display operation

                if (!isEmpty()) {

                    printf("Stack elements: ");

                    display();

                } else {

                    printf("Stack is empty.\n");

                }

                break;

            case 4:

                // Exit the program

                printf("Exiting...\n");

                return 0;

            default:

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

        }

    }

    return 0;

}

// Function to push (insert) an element onto the stack

void push(int value) {

    if (!isFull()) {

        stack[++top] = value;

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

    } else {

        printf("Stack is full. Cannot push.\n");

    }

}

// Function to pop (delete) an element from the stack

int pop() {

    return stack[top--];

}

// Function to display the elements of the stack

void display() {

    for (int i = 0; i <= top; i++) {

        printf("%d ", stack[i]);

    }

    printf("\n");

}

// Function to check if the stack is full

bool isFull() {

    return top == MAX\_SIZE - 1;

}

// Function to check if the stack is empty

bool isEmpty() {

    return top == -1;

}

**OUTPUT :**

Stack Menu:

1. Push (Insertion)

2. Pop (Deletion)

3. Display

4. Exit

Enter your choice: 1

Enter the value to push: 11

11 pushed into the stack.

Stack Menu:

1. Push (Insertion)

2. Pop (Deletion)

3. Display

4. Exit

Enter your choice: 1

Enter the value to push: 22

22 pushed into the stack.

Stack Menu:

1. Push (Insertion)

2. Pop (Deletion)

3. Display

4. Exit

Enter your choice: 1

Enter the value to push: 33

33 pushed into the stack.

Stack Menu:

1. Push (Insertion)

2. Pop (Deletion)

3. Display

4. Exit

Enter your choice: 3

Stack elements: 11 22 33

Stack Menu:

1. Push (Insertion)

2. Pop (Deletion)

3. Display

4. Exit

Enter your choice: 2

Popped value: 33

Stack Menu:

1. Push (Insertion)

2. Pop (Deletion)

3. Display

4. Exit

Enter your choice: 3

Stack elements: 11 22

Stack Menu:

1. Push (Insertion)

2. Pop (Deletion)

3. Display

4. Exit

Enter your choice: 1

Enter the value to push: 60

60 pushed into the stack.

Stack Menu:

1. Push (Insertion)

2. Pop (Deletion)

3. Display

4. Exit

Enter your choice: 3

Stack elements: 11 22 60

Stack Menu:

1. Push (Insertion)

2. Pop (Deletion)

3. Display

4. Exit

Enter your choice: 4

Exiting...

**Question 3 : Write a C Program to perform Binary Search Operation Using Array**

#include <stdio.h>

// Function to perform binary search

int binarySearch(int arr[], int n, int target) {

    int left = 0;          // Leftmost index of the array

    int right = n - 1;     // Rightmost index of the array

    while (left <= right) {

        int mid = left + (right - left) / 2; // Calculate the middle index

        if (arr[mid] == target) {

            return mid; // Element found at the middle index

        }

        if (arr[mid] < target) {

            left = mid + 1; // If target is greater, search the right half

        } else {

            right = mid - 1; // If target is smaller, search the left half

        }

    }

    return -1; // Element not found in the array

}

int main() {

    int n, target;

    printf("Enter the number of elements in the array: ");

    scanf("%d", &n);

    int arr[n];

    printf("Enter the elements in sorted order: ");

    for (int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);

    }

    printf("Enter the element to search for: ");

    scanf("%d", &target);

    int result = binarySearch(arr, n, target);

    if (result != -1) {

        printf("Element found at index %d.\n", result);

    } else {

        printf("Element not found in the array.\n");

    }

    return 0;

}

**OUTPUT :**

Enter the number of elements in the array: 9

Enter the elements in sorted order: 11 22 33 44 55 66 77 88 99

Enter the element to search for: 21

Element not found in the array.

**QUESTION NO : 4 TOWER OF HANOI**

//TOWER OF HANOI

#include<stdio.h>

void TowerOfHanoi( int n , char source , char aux , char dest){

    //If Number of disk  Zero and also Base Condition

    if(n==0){

        return ;

    }

    // Recursive Function

    TowerOfHanoi(n-1, source , aux , dest) ;

    printf("Move Disk From %d from %c to %c \n", n , source , dest ) ;

    TowerOfHanoi( n-1 , source , aux , dest ) ;

}

int main() {

    int n;

    printf("Enter the number of disks: ");

    scanf("%d", &n);

    TowerOfHanoi(n, 'A', 'B', 'C');

return 0;

}

OUTPUT :

Enter the number of disks: 3

Move Disk From 1 from A to C

Move Disk From 2 from A to C

Move Disk From 1 from A to C

Move Disk From 3 from A to C

Move Disk From 1 from A to C

Move Disk From 2 from A to C

Move Disk From 1 from A to C

Enter the number of disks: 2

Move Disk From 1 from A to C

Move Disk From 2 from A to C

Move Disk From 1 from A to C

**Question No 5: Write a C program to perform Quene Insertion , Deletion , Display and Exit from the menu .**

#include <stdio.h>

#define MAX 5

int queue[MAX];

int front = -1;

int rear = -1;

//Function for Insertion

void enqueue(int data) {

  if (rear == MAX - 1) {

    printf("Queue is full\n");

    return;

  }

  rear++;

  queue[rear] = data;

}

//Function for deletion

int dequeue() {

  if (front == rear) {

    printf("Queue is empty\n");

    return -1;

  }

  front++;

  return queue[front];

}

//Funtion to Display Quence

void display() {

  if (front == rear) {

    printf("Queue is empty\n");

    return;

  }

  int i;

  for (i = front + 1; i <= rear; i++) {

    printf("%d ", queue[i]);

  }

  printf("\n");

}

int main() {

  int choice;

  int data;

  while (1) {

    printf("\n \n") ;

    printf("1. Enqueue\n");

    printf("2. Dequeue\n");

    printf("3. Display\n");

    printf("4. Exit\n");

    printf("Enter your choice: ");

    scanf("%d", &choice);

    switch (choice) {

      case 1:

        printf("Enter data to enqueue: ");

        scanf("%d", &data);

        enqueue(data);

        break;

      case 2:

        data = dequeue();

        if (data == -1) {

          printf("Queue is empty\n");

        } else {

          printf("Data dequeued: %d\n", data);

        }

        break;

      case 3:

        display();

        break;

      case 4:

        return 0;

      default:

        printf("Invalid choice\n");

    }

  }

}

**OUTPUT :**

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 1

Enter data to enqueue: 32

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 1

Enter data to enqueue: 42

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 1

Enter data to enqueue: 90

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 1

Enter data to enqueue: 40

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 3

32 42 90 40

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 2

Data dequeued: 32

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 3

42 90 40

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 2

Data dequeued: 42

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 3

90 40

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 1

Enter data to enqueue: 12

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 3

90 40 12

1. Enqueue

2. Dequeue

3. Display

4. Exit

Enter your choice: 4

**Question 6 : Write a Program to perform Linear Search Operation.**

#include <stdio.h>

int arr[100], N;

int linear\_search(int val)

{

    for (int i = 0; i < N; i++)

    {

        if (arr[i] == val)

        {

            return i + 1;

        }

    }

    return 0;

}

int main()

{

    int arr[100];

    int val, c = 0;

    printf("Enter the total number of elements");

    scanf("%d", &N);

    printf("Enter the elements\n");

    for (int i = 0; i < N; i++)

        scanf("%d", &arr[i]);

    printf("\n Enter the value to be searched\n");

    scanf("%d", &val);

    printf("Element found in %d th position", linear\_search(val));

    return 0;

}

**OUTPUT :**

Enter Your X elemnt:8

ENter the size of the array5

Enter The elements of array: 2 4 3 8 9

Present