Data Structures - Lab Manual

Prepared By : Vasudevan T V

List of Programs

1 - Array Operations

Write a C program to perform the following operations on a one- dimensional array: (i) Traverse and print all elements (ii) Insert an element at a specific position (iii) Delete an element from a specific position (iv) Search for an element in the array and return its index

```
Program 1(i)
------------
#include <stdio.h>
#include <stdlib.h>
int main()
{
int i, n, arr[20];
system("cls");
printf("\n Enter the number of elements in the array : ");
scanf("%d", &n);
for(i=0;i<n;i++)
{
printf("\n arr[%d] = ", i);
scanf("%d",&arr[i]);
}
printf("\n The array elements are ");
for(i=0;i<n;i++)
printf("\t %d", arr[i]);
return(0);
}

Program 1(ii)
-------------
#include <stdio.h>
#include <stdlib.h>
int main()
{
int i, n, num, arr[10];
system("cls");
printf("\n Enter the number of elements in the array : ");
scanf("%d", &n);
for(i=0;i<n;i++)
{
printf("\n arr[%d] = ", i);
scanf("%d", &arr[i]);
}
printf("\n Enter the number to be inserted : ");
scanf("%d", &num);
int pos;
printf("\n Enter the position at which the number has to be added : ");
scanf("%d", &pos);
for(i=n-1;i>=pos;i--)
arr[i+1] = arr[i];
arr[pos] = num;
n = n+1;
printf("\n The array after insertion of %d is : ", num);
for(i=0;i<n;i++)
printf("\n arr[%d] = %d", i, arr[i]);
return 0;
}

Program 1(iii)
--------------
#include <stdio.h>
#include <stdlib.h>
int main()
{
int i, n, pos, arr[10];
system("cls");
printf("\n Enter the number of elements in the array : ");
scanf("%d", &n);
for(i=0;i<n;i++)
{
printf("\n arr[%d] = ", i);
scanf("%d", &arr[i]);
}
printf("\nEnter the position from which the number has to be deleted : ");
scanf("%d", &pos);
for(i=pos; i<n-1;i++)
arr[i] = arr[i+1];
n--;
printf("\n The array after deletion is : ");
for(i=0;i<n;i++)
printf("\n arr[%d] = %d", i, arr[i]);
return 0;
}
```

```
Program 1 (iv)
-------------
#include <stdio.h>
#define size 20 // Added to alter size of the array
int main( )
{
 int arr[size], num, i, n, found = 0, pos = -1;
 printf("\n Enter the number of elements in the array : ");
 scanf("%d", &n);
 printf("\n Enter the elements: ");
 for(i=0;i<n;i++)
 {
  scanf("%d", &arr[i]);
 }
 printf("\n Enter the number that has to be searched : ");
 scanf("%d", &num);
 for(i=0;i<n;i++)
 {
  if(arr[i] == num)
  {
   found =1;
   pos=i;
   printf("\n %d is found in the array at position= %d", num,i+1);
   /* +1 added in line 23 so that it would display the number in
   the first place in the array as in position 1 instead of 0 */
   break;
  }
 }
 if (found == 0)
     printf("\n %d does not exist in the array", num);
 return 0;
}
```

2 - Multi-Dimensional Arrays

Implement a C program to perform matrix addition and matrix multiplication using two-dimensional arrays.

```
Program 2a - Matrix Addition
---------------------------


#include <stdio.h>
```

```
#include <stdlib.h>
int main()
{
 int i, j;
 int rows1, cols1, rows2, cols2, rows_sum, cols_sum;
 int mat1[5][5], mat2[5][5], sum[5][5];
 printf("\n Enter the number of rows in the first matrix : ");
 scanf("%d",&rows1);
 printf("\n Enter the number of columns in the first matrix : ");
 scanf("%d",&cols1);
 printf("\n Enter the number of rows in the second matrix : ");
 scanf("%d",&rows2);
 printf("\n Enter the number of columns in the second matrix : ");
 scanf("%d",&cols2);
 if(rows1 != rows2 || cols1 != cols2)
 {
  printf("\n Number of rows and columns of both matrices must be equal");
  exit(0);
 }
 rows_sum = rows1;
 cols_sum = cols1;
 printf("\n Enter the elements of the first matrix ");
 for(i=0;i<rows1;i++)
 {
  for(j=0;j<cols1;j++)
  {
   scanf("%d",&mat1[i][j]);
  }
 }
 printf("\n Enter the elements of the second matrix ");
 for(i=0;i<rows2;i++)
 {
  for(j=0;j<cols2;j++)
  {
   scanf("%d",&mat2[i][j]);
  }
 }
 for(i=0;i<rows_sum;i++)
 {
  for(j=0;j<cols_sum;j++)
   sum[i][j] = mat1[i][j] + mat2[i][j];
 }
 printf("\n The elements of the resultant matrix are ");
 for(i=0;i<rows_sum;i++)
 {
  printf("\n");
  for(j=0;j<cols_sum;j++)
   printf("\t %d", sum[i][j]);
 }
 return 0;
}

Program 2b - Matrix Multiplication
----------------------------------

#include <stdio.h>
#include <stdlib.h>
int main()
{
 int i, j, k;
 int rows1, cols1, rows2, cols2, res_rows, res_cols;
 int mat1[5][5], mat2[5][5], res[5][5];
 printf("\n Enter the number of rows in the first matrix : ");
 scanf("%d",&rows1);
 printf("\n Enter the number of columns in the first matrix : ");
 scanf("%d",&cols1);
 printf("\n Enter the number of rows in the second matrix : ");
 scanf("%d",&rows2);
 printf("\n Enter the number of columns in the second matrix : ");
 scanf("%d",&cols2);
 if(cols1 != rows2)
 {
  printf("\n The number of columns in the first matrix must be equal
   to the number of rows in the second matrix");
  exit(0);
 }
 res_rows = rows1;
 res_cols = cols2;
 printf("\n Enter the elements of the first matrix ");
 for(i=0;i<rows1;i++)
 {
  for(j=0;j<cols1;j++)
  {
   scanf("%d",&mat1[i][j]);
  }
 }
 printf("\n Enter the elements of the second matrix ");
 for(i=0;i<rows2;i++)
 {
  for(j=0;j<cols2;j++)
  {
   scanf("%d",&mat2[i][j]);
  }
 }
 for(i=0;i<res_rows;i++)
 {
  for(j=0;j<res_cols;j++)
  {
   res[i][j]=0;
   for(k=0; k<res_cols;k++)
    res[i][j] += mat1[i][k] * mat2[k][j];
  }
 }
 printf("\n The elements of the product matrix are ");
 for(i=0;i<res_rows;i++)
 {
  printf("\n");
  for(j=0;j<res_cols;j++)
   printf("\t %d",res[i][j]);
 }
 return 0;
}
```

3 - String Manipulation

Write a C program to perform various string operations without using built-in string functions: (i) Find the length of a string (ii) Compare two strings (iii) Concatenate two strings (iv) Reverse a string (v) Check if a string is a palindrome.

```
Program 3(i)
------------

#include <stdio.h>
int main()
{
 char str[100], i = 0, length;
 printf("\n Enter the string : ");
 gets(str);
 while(str[i] != ’\0’)
  i++;
 length = i;
 printf("\n The length of the string is : %d", length);
 return 0;
}

Program 3(ii)
-------------

#include <stdio.h>
int compareStrings(char str1[], char str2[])
{
    int i = 0;

    // Compare each character of both strings
    while (str1[i] != ’\0’ && str2[i] != ’\0’)
    {
        if (str1[i] != str2[i])
        {
            // Return the difference of the first non-matching characters
            return str1[i] - str2[i];
        }
        i++;
    }
    // If we reach the end of one string, return the difference
    return str1[i] - str2[i];
}

int main()
{
 char str1[50], str2[50];
 printf("\n Enter the first string : ");
 gets(str1);
 printf("\n Enter the second string : ");
 gets(str2);
 // Compare the two strings
     int result = compareStrings(str1, str2);

     if (result == 0)
     {
         printf("The strings are equal.\n");
     }
     else if (result < 0)
     {
         printf("The first string is less than the second string.\n");
     }
     else
     {
         printf("The first string is greater than the second string.\n");
     }

     return 0;
}

Program 3(iii)
--------------

#include <stdio.h>
int main()
{
 char string1[25], string2[25], string3[50];
 int i=0, j=0, k=0;
 printf("\n Enter the first string : ");
 gets(string1);
 printf("\n Enter the second string : ");
 gets(string2);
 while(string1[i] != ’\0’)
 {
   string3[k] = string1[i];
   i++;
   k++;
 }
 while(string2[j] != ’\0’)
 {
   string3[k] = string2[j];
  j++;
   k++;
 }
 string3[k] = ’\0’;
 printf("\n The concatenated string is : ");
 puts(string3);
 return 0;
}

Program 3(iv)
-------------
#include <stdio.h>
int main()
{
 char string[25], reverse_string[25];
 int i=0, j=0;
 printf("\n Enter the string : ");
 gets(string);
 while(string[i] != ’\0’)
 {
   i++;
 }
 i--;
 while(i != -1)
 {
   reverse_string[j] = string[i];
   j++;
   i--;
 }
 reverse_string[j] = ’\0’;
 printf("\n The reverse string is : ");
 puts(reverse_string);
 return 0;
}

Program 3(v)
------------
#include <stdio.h>
int compareStrings(char str1[], char str2[])
{
    int i = 0;

    // Compare each character of both strings
    while (str1[i] != ’\0’ && str2[i] != ’\0’)
    {
        if (str1[i] != str2[i])
 {
            return str1[i] - str2[i]; // Return the difference of the
```

```
                                      // first non-matching characters
        }
        i++;
    }
    // If we reach the end of one string, return the difference
    return str1[i] - str2[i];
}

int main()
{
 char string1[25], string2[25], reverse_string[25];
 int i=0, j=0, k=0, result;
 printf("\n Enter the string : ");
 gets(string1);
 while(string1[i] != ’\0’)
 {
         if(string1[i] != ’ ’ && string1[i] != ’\’’)
           {
            string2[j] = string1[i];
            j++;
           }
       i++;
 }
    string2[j] = ’\0’;

 j--;
 while(j != -1)
 {
        reverse_string[k] = string2[j];
        k++;
        j--;
 }
 reverse_string[k] = ’\0’;
 result = compareStrings(string2, reverse_string);
 if (result == 0)
        printf("The string is a palindrome.\n");

    else
        printf("The string is not a palindrome.\n");
 return 0;
}
```

4 - Recursion - Factorial Calculation

Implement a recursive function in C to calculate the factorial of a given number.

```
Program 4
---------
#include <stdio.h>
int factorial(int n)
{
    if(n==0 || n==1)
        return 1;
    else
        return (n * factorial(n-1));
}
int main()
{
    int number, value;
    printf("\n Enter the number: ");
    scanf("%i", &number);
    value = factorial(number);
    printf("\n Factorial of %i = %i\n", number, value);
    return 0;
}
```

5 - Recursion - Fibonacci Series

Write a C program to generate the Fibonacci series up to a given number using both recursive and iterative methods.

```
Program 5 - Recursive Method
----------------------------
#include <stdio.h>
int Fibonacci(int n)
{
    if ( n == 0 )
        return 0;
    else if ( n == 1 )
  return 1;
    else
        return ( Fibonacci(n-1) + Fibonacci(n-2) );
}
```

```
int main()
{
    int number, i = 0, result;
    printf("Enter the number of terms\n");
    scanf("%i",&number);
    printf("Fibonacci series\n");
    for(i = 0; i < number; i++ )
    {
        result = Fibonacci(i);
        printf("%i\t",result);
    }
    printf("\n");
    return 0;
}

Program 5 - Iterative Method
----------------------------
#include <stdio.h>
int main()
{
    int number, i = 0, fibonacci[50];
    printf("Enter the number of terms\n");
    scanf("%i",&number);
    printf("Fibonacci series\n");
    fibonacci[0] = 0;
    fibonacci[1] = 1;
    for(i = 2; i < number; i++)
        fibonacci[i] = fibonacci[i-1] + fibonacci[i-2];
    for(i = 0; i < number; i++ )
        printf("%i\t", fibonacci[i]);
    printf("\n");
    return(0);
}
```

6 - Recursion - Towers of Hanoi

Implement the Towers of Hanoi problem using recursion in C. The program should print the steps involved in moving disks from the source to the destination peg.

```
Program 6
---------
#include <stdio.h>
```

```
void move(int number, char source, char destination, char spare)
{
 if (number==1)
    printf("\n Move disk 1 from Tower %c to Tower %c",source,destination);
 else
    {
     move(number-1,source,spare,destination);
     printf("\n Move disk %i from Tower %c to Tower %c",
           number,source,destination);
     move(number-1,spare,destination,source);
    }
}
int main()
{
    int number;
    printf("\n Enter the number of disks: ");
    scanf("%i", &number);
    printf("\n Source = Tower A");
    printf("\n Destination = Tower C");
    move(number,'A', 'C', 'B');
    printf("\n");
    return 0;
}
```

7 - Dynamic Memory Allocation

Write a C program to dynamically allocate memory for a one-dimensional array using malloc(). Perform insertion and deletion operations on the array. Deallocate the memory once operations are complete.

```
Program 7
---------
#include <stdio.h>
#include <stdlib.h>
int size, *array, number;
void insert()
{
    int element,position, i;
    printf("\n Enter the element to be inserted : ");
    scanf("%i", &element);
    printf("\n Enter the position at which the element has to be added: ");
    scanf("%i", &position);
    if(position > number)
      printf(" Invalid position\n");
    else
    {
        for(i=number-1;i>=position;i--)
            array[i+1] = array[i];
        array[position] = element;
        number = number+1;
        printf("\n The array after insertion of %i is\n", element);
        for(i=0;i<number;i++)
            printf(" %i ", array[i]);
        printf("\n");
    }
}
void delete()
{
    int position, i;
    printf("\n Enter the position from which the element is to be
    deleted : ");
    scanf("%i", &position);
    if(position >= number)
      printf(" Invalid position\n");
    else
    {
        for(i=position; i<number-1;i++)
            array[i] = array[i+1];
        number--;
        printf("\n The array after deletion is\n ");
        for(i=0;i<number;i++)
            printf(" %i ", array[i]);
        printf("\n");
    }
}
int main()
{
    int i;
    printf("\n Enter the size of the array:");
    scanf("%i", &size);
    array = (int *)malloc(size * sizeof(int));
    if(array == NULL)
    {
        printf("\n Memory Allocation Failed");
        exit(0);
    }
    printf("\n Enter the number of elements of the array:");
    scanf("%i", &number);
    if(number==0)
    {
        printf("\n Array is empty\n");
        exit(0);
    }
    else if(number > size)
    {
        printf("\n Number of elements should not be greater than size\n");
        exit(0);
    }
    for(i = 0;i < number;i++)
    {
        printf("\n Enter the element[%i] of the array: ", i);
        scanf("%i", &array[i]);
    }

    printf("\n The array contains \n");
    for(i = 0;i < number;i++)
    printf(" %i ", array[i]);
 printf("\n");

 if(size==number)
 {
     printf(" Array is Full\n");
     delete();
 }
 else if(size > number)
 {
     insert();
     delete();
 }
 free(array);
 return 0;
}
```

8 - Structures and Unions

Define a structure in C to store information about a student (Name, Roll Number, Marks in three subjects). Write a program to input data for multiple students and calculate the total and average marks for each student. Also, demonstrate the use of unions in storing the same information.

```
Program 8_i (Structure)
-----------------------
#include <stdio.h>
int main()
{
    struct student
    {
  char name[30];
        int roll_number;
  int mark1;
  int mark2;
  int mark3;
    };
    struct student stud[10];
    int i, number, total_marks;
    float average_marks;
    printf("\n Enter the number of students : ");
    scanf("%i", &number);
    for(i=0;i<number;i++)
    {
        printf("\n Enter the name : ");
        fgets(stud[i].name, 30, stdin);
        fgets(stud[i].name, 30, stdin);
        printf("\n Enter the roll number : ");
        scanf("%i", &stud[i].roll_number);
        printf("\n Enter the mark in subject 1 : ");
        scanf("%i",&stud[i].mark1);
        printf("\n Enter the mark in subject 2 : ");
        scanf("%i",&stud[i].mark2);
        printf("\n Enter the mark in subject 3 : ");
        scanf("%i",&stud[i].mark3);
    }
    for(i=0;i<number;i++)
    {
        printf("\n ********DETAILS OF STUDENT %i*******", i+1);
        printf("\n ROLL No. = %i", stud[i].roll_number);
        printf("\n NAME = %s", stud[i].name);
        total_marks = stud[i].mark1 + stud[i].mark2 + stud[i].mark3;
        average_marks = total_marks / 3;
        printf("\n TOTAL MARKS = %i", total_marks);
        printf("\n AVERAGE MARKS = %.2f", average_marks);
    }
    printf("\n");
    return(0);
}

Program 8_ii (Union)
--------------------
#include <stdio.h>
int main()
{
    union student
    {
  char name[30];
        int roll_number;
  int mark1;
  int mark2;
  int mark3;
    };
    union student stud[10];
    int i, number, total_marks;
    float average_marks;
    printf("\n Enter the number of students : ");
    scanf("%i", &number);
    for(i=0;i<number;i++)
    {
        printf("\n Enter the name : ");
        fgets(stud[i].name, 30, stdin);
        fgets(stud[i].name, 30, stdin);
        printf("\n Enter the roll number : ");
        scanf("%i", &stud[i].roll_number);
        printf("\n Enter the mark in subject 1 : ");
        scanf("%i",&stud[i].mark1);
        printf("\n Enter the mark in subject 2 : ");
        scanf("%i",&stud[i].mark2);
        printf("\n Enter the mark in subject 3 : ");
        scanf("%i",&stud[i].mark3);
    }
    for(i=0;i<number;i++)
    {
        printf("\n ********DETAILS OF STUDENT %i*******", i+1);
        printf("\n ROLL No. = %i", stud[i].roll_number);
        printf("\n NAME = %s", stud[i].name);
        total_marks = stud[i].mark1 + stud[i].mark2 + stud[i].mark3;
        average_marks = total_marks / 3;
        printf("\n TOTAL MARKS = %i", total_marks);
        printf("\n AVERAGE MARKS = %.2f", average_marks);
    }
    printf("\n");
    return(0);
}
```

16 - Stack Implementation Using Arrays

Write a C program to implement a stack using arrays. Perform the following stack operations: (i) Push an element onto the stack (ii) Pop an element from the stack (iii) Display the top element of the stack (iv) Check if the stack is empty or full.

```
Program 16.c
------------
#include <stdio.h>
#define MAX 3 // Altering this value changes size of stack created
int stack[MAX], top=-1;
void push(int stack[], int value);
int pop(int stack[]);
int gettop(int stack[]);
void display(int stack[]);
int main()
{
 int value, option;
 do
 {
  printf("\n *****MAIN MENU*****");
  printf("\n 1. PUSH");
  printf("\n 2. POP");
  printf("\n 3. TOP");
  printf("\n 4. DISPLAY");
  printf("\n 5. EXIT");
  printf("\n Enter your option: ");
  scanf("%i", &option);
  switch(option)
  {
   case 1:
```

```
   printf("\n Enter the number to be pushed on stack: ");
   scanf("%i", &value);
   push(stack, value);
   break;
   case 2:
   value = pop(stack);
   if(value != -1)
    printf("\n The value deleted from stack is: %i", value);
   break;
   case 3:
   value = gettop(stack);
   if(value != -1)
    printf("\n The value stored at top of stack is: %i", value);
   break;
   case 4:
   display(stack);
   break;
  }
 }while(option != 5);
 return 0;
}
void push(int stack[], int value)
{
 if(top == MAX-1)
  printf("\n STACK IS FULL");
 else
 {
  top++;
  stack[top] = value;
 }
}
int pop(int stack[])
{
 int value;
 if(top == -1)
 {
  printf("\n STACK IS EMPTY");
  return -1;
 }
 else
 {
  value = stack[top];
  top--;
  return value;
 }
}

void display(int stack[])
{
 int i;
 if(top == -1)
  printf("\n STACK IS EMPTY");
 else
 {
 for(i=top;i>=0;i--)
  printf("\n %i",stack[i]);
 printf("\n"); // Added for formatting purposes
 }
}
int gettop(int stack[])
{
 if(top == -1)
 {
  printf("\n STACK IS EMPTY");
  return -1;
 }
 else
  return (stack[top]);
}
```

20 - Queue Implementation Using Arrays

Write a C program to implement a simple queue using arrays. Implement the following queue operations: (i) Enqueue an element (ii) Dequeue an element (iii) Display the elements of the queue (iv) Check if the queue is empty or full.

```
Program 20
----------

#include <stdio.h>
#define MAX 10 // Changing this value will change length of array
int queue[MAX];
int front = -1, rear = -1;
void insert(void);
```

```
int delete_element(void);
void display(void);
int main()
{
 int option, value;
 do
 {
  printf("\n *****Main Menu******");
  printf("\n 1. Insert an element");
  printf("\n 2. Delete an element");
  printf("\n 3. Display the queue");
  printf("\n 4. EXIT");
  printf("\n Enter your option : ");
  scanf("%i", &option);
  switch(option)
  {
   case 1:
    insert();
    break;
   case 2:
    value = delete_element();
    if (value != -1)
     printf("\n The number deleted is : %i", value);
    break;
   case 3:
    display();
    break;
  }
 }while(option != 4);
 return 0;
}
void insert()
{
 int number;
 printf("\n Enter the number to be inserted in the queue : ");
 scanf("%i", &number);
 if(rear == MAX-1)
 {
  printf("\n Queue is Full");
  return;
 }
 else if(front == -1 && rear == -1)
```

```
  front = rear = 0;
 else
  rear++;
 queue[rear] = number;
}
int delete_element()
{
 int value;
 if(front == -1 || front>rear)
 {
  printf("\n Queue is Empty");
  return -1;
 }
 else
 {
  value = queue[front];
  front++;
  if(front > rear)
  front = rear = -1;
  return value;
 }
}

void display()
{
 int i;
 printf("\n");
 if(front == -1 || front > rear)
  printf("\n QUEUE IS EMPTY");
 else
 {
  for(i = front;i <= rear;i++)
  printf("\t %i", queue[i]);
 }
}
```

26 - Linear Search Implementation

Write a C program to implement the linear search algorithm. The program should: (i) Search for a key in an unsorted array (ii) Return the index of the key if found, otherwise return -1 (iii) Count the number of comparisons made during the search.

```
Program 26
----------
#include <stdio.h>
int main()
{
    int array[10], key, i, n, found = -1, index = -1, count = 0;
    printf("\n Enter the number of keys in the array : ");
    scanf("%i", &n);
    printf("\n Enter the keys: ");
    for(i=0;i<n;i++)
    {
        scanf("%i", &array[i]);
    }
    printf("\n Enter the key that has to be searched : ");
    scanf("%i", &key);
    for(i=0;i<n;i++)
    {
        count++;
        if(array[i] == key)
        {
           found = 1;
           index = i;
           printf("\n %i is found in the array at index = %i", key, index);
           break;
        }
    }
    if (found == -1)
        printf("\n  %i does not exist in the array : %i", key, index);
    printf("\n \n Number of Comparisons: %i \n", count);
    return 0;
}
```

27 - Binary Search Implementation

Implement the binary search algorithm in C. The program should: (i) Search for a key in a sorted array (ii) Return the index of the key if found, otherwise return -1 (iii) Count the number of comparisons made during the search.

```
Program 27
----------
#include <stdio.h>
int main( )
```

```
{
 int array[10] = {10,20,30,40,50,60,70,80,90,100};
 int  key, i, n = 10, found = -1, index = -1, count = 0;
 int lower = 0, upper = 9, mid ;
 printf("\n The keys in the sorted array are: ");
 for(i=0;i<n;i++)
    printf("%i ",array[i]);
 printf ( "\n Enter the key to be searched: " ) ;
 scanf ( "%i", &key) ;

 while ( lower <= upper )
 {
  mid = ( lower + upper ) / 2 ;
  count++;
  if ( key == array[mid] )
  {
   found = 1;
   printf("\n %i is found in the array at index = %i", key, mid);
   break;
  }
  if ( key > array[mid] )
   lower = mid + 1 ;
  if ( key < array[mid] )
   upper = mid - 1 ;
 }
 if (found == -1)
        printf("\n  %i does not exist in the array : %i", key, index);
 printf("\n \n Number of Comparisons: %i \n", count);
 return 0 ;
}
```

28 - Bubble Sort Implementation

Write a C program to implement the bubble sort algorithm. The program should: (i) Sort an array of integers in ascending order (ii) Display the number of swaps and comparisons made during the sorting process.

```
Program 28
----------
#include <stdio.h>
int main()
{
    int i, n, temp, j, array[10], number_of_comparisons = 0,
    number_of_swaps = 0;
    printf("\n Enter the number of elements in the array : ");
    scanf("%i", &n);
    printf("\n Enter the elements: ");
    for(i=0;i<n;i++)
        scanf("%i", &array[i]);
    for(i=0;i<n;i++)
    {
        for(j=0;j<n-i-1;j++)
        {
            number_of_comparisons++;
            if(array[j] > array[j+1])
            {
                temp = array[j];
                array[j] = array[j+1];
                array[j+1] = temp;
                number_of_swaps++;
            }
        }
    }
    printf("\n The array sorted in ascending order is: ");
    for(i=0;i<n;i++)
        printf("%i ", array[i]);
    printf("\n Number of Comparisons : %i", number_of_comparisons);
    printf("\n Number of Swaps : %i\n", number_of_swaps);
    return 0;
}
```

29 - Insertion Sort Implementation

Implement the insertion sort algorithm in C. The program should: (i) Sort an array of integers in ascending order (ii) Display the number of comparisons and shifts made during the sorting process.

```
Program 29
----------
#include <stdio.h>
int main()
{
    int i, n, temp, j, array[10];
    int number_of_comparisons = 0, number_of_shifts = 0;
    printf("\n Enter the number of elements in the array : ");
    scanf("%i", &n);
    printf("\n Enter the elements: ");
    for(i=0;i<n;i++)
        scanf("%i", &array[i]);
    for(i=1;i<n;i++)
    {
        temp = array[i];
        j = i-1;
        while((temp < array[j]) && (j>=0))
        {
            array[j+1] = array[j];
            number_of_shifts++;
            j--;
            number_of_comparisons++;
        }
        if(j>=0)
            number_of_comparisons++;
        array[j+1] = temp;
    }
    printf("\n The array sorted in ascending order is: ");
    for(i=0;i<n;i++)
        printf("%i ", array[i]);
    printf("\n Number of Comparisons : %i", number_of_comparisons);
    printf("\n Number of Shifts : %i\n", number_of_shifts);
    return 0;
}
```

30 - Selection Sort Implementation

Write a C program to implement the selection sort algorithm. Your program should: (i) Sort an array of integers in ascending order (ii) Display the number of comparisons and swaps made during the sorting process.

```
Program 30
----------
#include <stdio.h>
int main()
{
    int i, n, temp, j, array[10];
    int number_of_comparisons = 0, number_of_swaps = 0;
    printf("\n Enter the number of elements in the array : ");
    scanf("%i", &n);
    printf("\n Enter the elements: ");
    for(i=0;i<n;i++)
        scanf("%i", &array[i]);
    for ( i = 0 ; i < n - 1 ; i++ )
    {
        for ( j = i + 1 ; j < n ; j++ )
        {
            number_of_comparisons++;
            if ( array[i] > array[j] )
            {
                temp = array[i] ;
                array[i] = array[j] ;
                array[j] = temp ;
                number_of_swaps++;
            }
        }
    }
    printf("\n The array sorted in ascending order is: ");
    for(i=0;i<n;i++)
        printf("%i ", array[i]);
    printf("\n Number of Comparisons : %i", number_of_comparisons);
    printf("\n Number of Swaps : %i\n", number_of_swaps);
    return 0;
}
```

31 - Quick Sort Implementation

Implement the quick sort algorithm in C. Your program should: (i) Sort an array of integers in ascending order (ii) Display the array after each partition step.

```
Program 31
----------
#include <stdio.h>
int split ( int array[ ], int lower, int upper )
{
    int p, q, number, temp ;
    p = lower + 1 ;
    q = upper ;
    number = array[ lower ] ;
    while ( q >= p )
    {
        while ( array[p] < number )
            p++ ;
        while ( array[q] > number )
            q-- ;
        if ( q > p )
        {
            temp = array[p] ;
            array[p] = array[q] ;
            array[q] = temp ;
        }
    }
    temp = array[lower] ;
    array[lower] = array[q] ;
    array[q] = temp ;
    return q ;
}
void quicksort ( int array[ ], int lower, int upper )
{
    int i, k ;
    if ( upper > lower )
    {
        i = split ( array, lower, upper ) ;
        // Display the array after partitioning
        printf("Array after partitioning: ");
        for(k = 0; k <= upper; k++)
            printf("%i ", array[k]);
        printf("\n");
        quicksort ( array, lower, i - 1 ) ;
        quicksort ( array, i + 1, upper ) ;
    }
}

int main()
{
    int i, n, temp, j, array[10];
    printf("\n Enter the number of elements in the array : ");
    scanf("%i", &n);
    printf("\n Enter the elements: ");
    for(i=0;i<n;i++)
        scanf("%i", &array[i]);
    quicksort ( array, 0, n-1 ) ;
    printf("\n The array sorted in ascending order is: ");
    for(i=0;i<n;i++)
        printf("%i ", array[i]);
    printf("\n");
    return 0;
}
```

32 - Merge Sort Implementation

Write a C program to implement the merge sort algorithm. Your program should: (i) Sort an array of integers in ascending order (ii) Display the array after each merge operation.

```
Program 32
----------
#include <stdio.h>
void merge(int array[], int beginning, int mid, int end)
{
    int i=beginning, j=mid+1, index=beginning, temp[10], k;
    while((i<=mid) && (j<=end))
    {

        if(array[i] < array[j])
        {

            temp[index] = array[i];

            i++;
        }
        else
        {

            temp[index] = array[j];

            j++;
        }

        index++;
    }
    if(i>mid)
    {
        while(j<=end)
        {
            temp[index] = array[j];
            j++;
            index++;
        }
    }
    else
    {
        while(i<=mid)
        {
            temp[index] = array[i];
            i++;
            index++;
        }
    }
    for(k=beginning;k<index;k++)
        array[k] = temp[k];
    printf("\n The merged array is: ");
    for(k=beginning;k<index;k++)
        printf(" %i ", array[k]);
    printf("\n");
}
void merge_sort(int array[], int beginning, int end)
{
    int mid;
    if(beginning<end)
    {
        mid = (beginning+end)/2;
        merge_sort(array, beginning, mid);
        merge_sort(array, mid+1, end);
        merge(array, beginning, mid, end);
    }
}
void main()
{
    int array[10], i, n;
    printf("\n Enter the number of elements in the array : ");
    scanf("%i", &n);
    printf("\n Enter the elements of the array: ");
    for(i=0;i<n;i++)
    {
        scanf("%i", &array[i]);
    }
    merge_sort(array, 0, n-1);
    printf("\n The sorted array is: ");
    for(i=0;i<n;i++)
        printf(" %i ", array[i]);
    printf("\n");
}
```