|  |  |
| --- | --- |
| **Ex No** | **Name of Experiment** |
| 1 A | Simple Algorithm |
| 1 B | Insertion sort |
| 2 | Bubble Sort |
| 3 A | Recurrence Type-Merge sort |
| 3 B | Recurrence Type- Linear search |
| 4 A | Quicksort |
| 4 B | Binary search |
| 5 | Strassen Matrix multiplication |
| 6 A | Finding Maximum and Minimum in an array |
| 6 B | Convex Hull problem |
| 7 A | Huffman coding Using Greedy |
| 7 B | knapsack using greedy |
| 8 A | Tree Traversal |
| 8 B | Krukshall’s MST |
| 8 C | Prims MST |
| 9 | Longest common subsequence |
| 10 | N queen’s problem |
| 11 | Travelling salesman problem |
| 12 | BFS and DFS implementation with array |
| 13 | Randomized quick sort |
| 14 | String matching algorithms |
| 15 | Discussion over analyzing a real time problem |

|  |  |
| --- | --- |
| Ex No: 1 A | FACTORIAL OF N NUMBERS |
| Date: |

**AIM**

To Write a C Program for finding Factorial of N numbers

**ALGORITHM**

step 1. Start

step 2. Read the number n

step 3. [Initialize]

i=1, fact=1

step 4. Repeat step 4 through 6 until i=n

step 5. fact=fact\*i

step 6. i=i+1

step 7. Print fact

step 8. Stop

**PROGRAM**

#include<stdio.h>

#include<conio.h>

int main()

{

int n,i,fact=1;

clrscr();

printf("Enter any number : ");

scanf("%d", &n);

for(i=1; i<=n; i++)

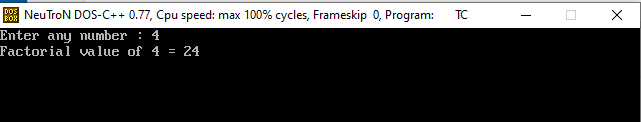
fact = fact \* i;

printf("Factorial value of %d = %d",n,fact);

getch();

}

**INPUT & OUTPUT**

****

**RESULT**

Thus the C Program for Finding factorial of N numbers has been executed and the output has been verified

|  |  |
| --- | --- |
| Ex No: 1 B | INSERTION SORT |
| Date: |

**AIM**

To Write a C Program for Sorting of N numbers using Insertion Sort

**ALGORITHM**

Step 1 − If the element is the first one, it is already sorted.

Step 2 – Move to next element

Step 3 − Compare the current element with all elements in the sorted array

Step 4 – If the element in the sorted array is smaller than the current element, iterate to the next element. Otherwise, shift the entire greater element in the array by one position towards the right

Step 5 − Insert the value at the correct position

Step 6 − Repeat until the complete list is sorted

**PROGRAM**

#include<stdio.h>

#include<conio.h>

int main(){

int i, j, count, temp, number[25];

clrscr();

printf("How many numbers u are going to enter?: ");

scanf("%d",&count);

printf("Enter %d elements: ", count);

// This loop would store the input numbers in array

for(i=0;i<count;i++)

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

// Implementation of insertion sort algorithm

for(i=1;i<count;i++){

temp=number[i];

j=i-1;

while((temp<number[j])&&(j>=0)){

number[j+1]=number[j];

j=j-1;

}

number[j+1]=temp;

}

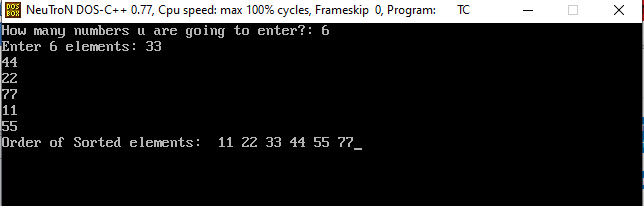
printf("Order of Sorted elements: ");

for(i=0;i<count;i++)

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

getch();

}

**INPUT & OUTPUT**

**RESULT**

Thus the C Program for Sorting of N numbers using Insertion Sort has been executed and the output has been verified

|  |  |
| --- | --- |
| Ex No: 2 | Bubble Sort |
| Date: |

**AIM**

To Write a C Program for Sorting of N numbers using Bubble Sort

**ALGORITHM**

Step 1: Repeat Steps 2 and 3 for i=1 to 10

Step 2: Set j=1

Step 3: Repeat while j<=n

if a[i] < a[j]

Then interchange a[i] and a[j]

[End of if]

Set j = j+1

[End of Inner Loop]

[End of Step 1 Outer Loop]

Step 4: Exit

**PROGRAM**

#include<stdio.h>

#include<conio.h>

void main()

{

int i,n,temp,j,arr[25];

clrscr();

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

scanf("%d",&n);

printf("\nEnter the elements:\n\n");

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

{

printf(" Array[%d] = ",i);

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

}

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

{

for(j=0 ; j<n-i-1 ; j++)

{

if(arr[j]>arr[j+1]) //Swapping Condition is Checked

{

temp=arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

printf("\nThe Sorted Array is:\n\n");

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

{

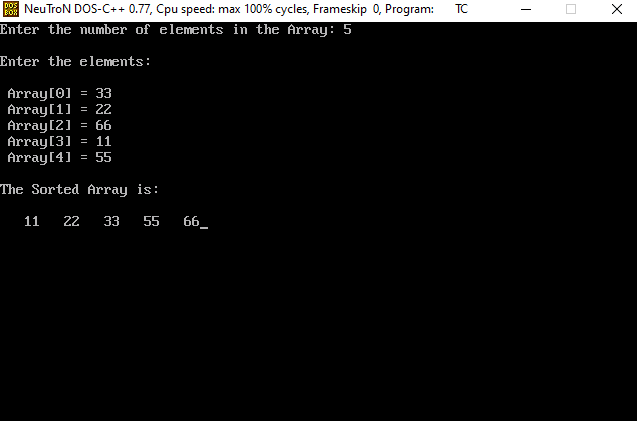
printf(" %4d",arr[i]);

}

getch();

}

**INPUT& OUTPUT**

****

**RESULT**

Thus the C Program for Sorting of N numbers using Bubble Sort has been executed and the output has been verified

|  |  |
| --- | --- |
| Ex No: 3 A | Recurrence Type-Merge sort |
| Date: |

**AIM**

To Write a C Program for Sorting of N numbers using merge sort

**ALGORITHM**

Step 1 : Declare left and right var which will mark the extreme indices of the array

Step 2: Left will be assigned to 0 and right will be assigned to n-1

Step 3: Find mid = (left+right)/2

Step 4: Call mergeSort on (left,mid) and (mid+1,rear)

MergeSort(arr, left, right):

if left > right

return

mid = (left+right)/2

mergeSort(arr, left, mid)

mergeSort(arr, mid+1, right)

merge(arr, left, mid, right)

end

Step 5: continue till left<right

Step 6: merge on the 2 sub problems

**PROGRAM**

#include <stdio.h>

#define max 10

int a[11] = { 10, 14, 19, 26, 27, 31, 33, 35, 42, 44, 0 };

int b[10];

void merging(int low, int mid, int high) {

int l1, l2, i;

for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high; i++) {

if(a[l1] <= a[l2])

b[i] = a[l1++];

else

b[i] = a[l2++];

}

while(l1 <= mid)

b[i++] = a[l1++];

while(l2 <= high)

b[i++] = a[l2++];

for(i = low; i <= high; i++)

a[i] = b[i];

}

void sort(int low, int high) {

int mid;

if(low < high) {

mid = (low + high) / 2;

sort(low, mid);

sort(mid+1, high);

merging(low, mid, high);

} else

{

return;

}

}

int main()

{

int i;

clrscr();

printf("List before sorting\n");

for(i = 0; i <= max; i++)

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

sort(0, max);

printf("\nList after sorting\n");

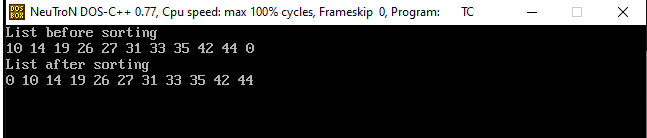
for(i = 0; i <= max; i++)

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

getch();

}

**INPUT& OUTPUT**



**RESULT**

Thus the C Program for Sorting of N numbers using Merge Sort has been executed and the output has been verified

|  |  |
| --- | --- |
| Ex No: 3 B | Recurrence Type-Linear Search |
| Date: |

**AIM**

To Write a C Program for searching an element using Linear Search

**ALGORITHM**

Step 1 - Read the search element from the user.

Step 2 - Compare the search element with the first element in the list.

Step 3 - If both are matched, then display "Given element is found!!!" and terminate the function

Step 4 - If both are not matched, then compare search element with the next element in the list.

Step 5 - Repeat steps 3 and 4 until search element is compared with last element in the list.

Step 6 - If last element in the list also doesn't match, then display "Element is not found!!!" and terminate the function.

**PROGRAM**

#include<stdio.h>

#include<conio.h>

void main(){

int list[20],size,i,sElement;

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

scanf("%d",&size);

printf("Enter any %d integer values: ",size);

for(i = 0; i < size; i++)

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

printf("Enter the element to be Search: ");

scanf("%d",&sElement);

// Linear Search Logic

for(i = 0; i < size; i++)

{

if(sElement == list[i])

{

printf("Element is found at %d index", i);

break;

}

}

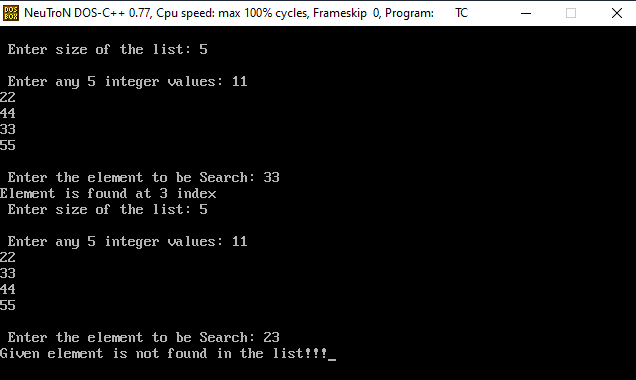
if(i == size)

printf("Given element is not found in the list!!!");

getch();

}

**INPUT& OUTPUT**



**RESULT**

Thus the C Program for searching an element by using linear Search has been executed and the output has been verified

|  |  |
| --- | --- |
| Ex No: 4 A | Quick sort |
| Date: |

**AIM**

To Write a C Program for Sorting of N numbers using Quick Sort

**ALGORITHM**

Step 1 - Consider the first element of the list as pivot (i.e., Element at first position in the list).

Step 2 - Define two variables i and j. Set i and j to first and last elements of the list respectively.

Step 3 - Increment i until list[i] > pivot then stop.

Step 4 - Decrement j until list[j] < pivot then stop.

Step 5 - If i < j then exchange list[i] and list[j].

Step 6 - Repeat steps 3,4 & 5 until i > j.

Step 7 - Exchange the pivot element with list[j] element.

**PROGRAM**

#include<stdio.h>

#include<conio.h>

void quickSort(int [10],int,int);

void main(){

int list[20],size,i;

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

scanf("%d",&size);

printf("Enter %d integer values: ",size);

for(i = 0; i < size; i++)

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

quickSort(list,0,size-1);

printf("List after sorting is: ");

for(i = 0; i < size; i++)

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

getch();

}

void quickSort(int list[10],int first,int last){

int pivot,i,j,temp;

if(first < last)

{

pivot = first;

i = first;

j = last;

while(i < j){

while(list[i] <= list[pivot] && i < last)

i++;

while(list[j] > list[pivot])

j--;

if(i <j)

{

temp = list[i];

list[i] = list[j];

list[j] = temp;

}

}

temp = list[pivot];

list[pivot] = list[j];

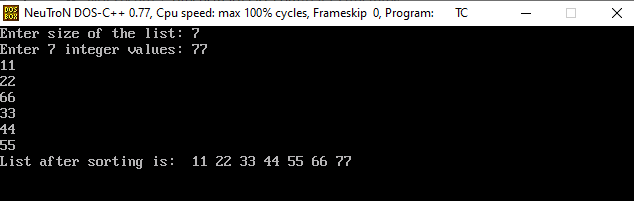
list[j] = temp;

quickSort(list,first,j-1);

quickSort(list,j+1,last);

}

}

**INPUT & OUTPUT**

**RESULT**

Thus the C Program for Sorting of N numbers using Quick Sort has been executed and the output has been verified

|  |  |
| --- | --- |
| Ex No: 4 B | **Binary Search** |
| Date: |

**AIM**

To Write a C Program for searching an element using Binary Search

**ALGORITHM**

Step 1 - Read the search element from the user.

Step 2 - Find the middle element in the sorted list.

Step 3 - Compare the search element with the middle element in the sorted list.

Step 4 - If both are matched, then display "Given element is found!!!" and terminate the function.

Step 5 - If both are not matched, then check whether the search element is smaller or larger than the middle element.

Step 6 - If the search element is smaller than middle element, repeat steps 2, 3, 4 and 5 for the left sublist of the middle element.

Step 7 - If the search element is larger than middle element, repeat steps 2, 3, 4 and 5 for the right sublist of the middle element.

Step 8 - Repeat the same process until we find the search element in the list or until sublist contains only one element.

Step 9 - If that element also doesn't match with the search element, then display "Element is not found in the list!!!" and terminate the function.

**PROGRAM**

#include<stdio.h>

#include<conio.h>

void main()

{

int first, last, middle, size, i, sElement, list[100];

clrscr();

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

scanf("%d",&size);

printf("Enter %d integer values in Assending order\n", size);

for (i = 0; i < size; i++)

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

printf("Enter value to be search: ");

scanf("%d", &sElement);

first = 0;

last = size - 1;

middle = (first+last)/2;

while (first <= last) {

if (list[middle] < sElement)

first = middle + 1;

else if (list[middle] == sElement) {

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

break;

}

else

last = middle - 1;

middle = (first + last)/2;

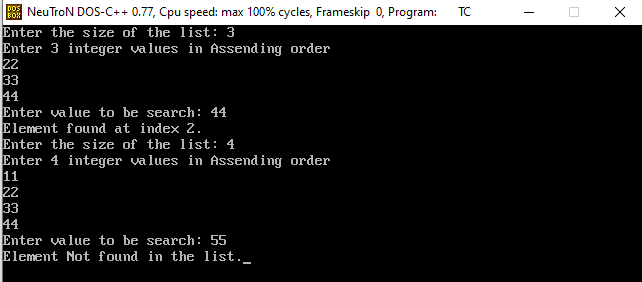
}

if (first > last)

printf("Element Not found in the list.");

getch();

}

**INPUT & OUTPUT**

**RESULT**

Thus the C Program for searching an element by using Binary Search has been executed and the output has been verified

|  |  |
| --- | --- |
| Ex No: 5 | **Strassen Matrix multiplication** |
| Date: |

**AIM**

To Write a C Program for performing matrix Multiplication using divide and Conquer

**ALGORITHM**

Step 1: Algorithm Strassen(n, a, b, d)

begin

Step 2: If n = threshold then compute

C = a \* b is a conventional matrix.

Else

Step 3: Partition a into four sub matrices a11, a12, a21, a22.

Partition b into four sub matrices b11, b12, b21, b22.

Strassen ( n/2, a11 + a22, b11 + b22, d1)

Strassen ( n/2, a21 + a22, b11, d2)

Strassen ( n/2, a11, b12 – b22, d3)

Strassen ( n/2, a22, b21 – b11, d4)

Strassen ( n/2, a11 + a12, b22, d5)

Strassen (n/2, a21 – a11, b11 + b22, d6)

Strassen (n/2, a12 – a22, b21 + b22, d7)

Step 4: C = d1+d4-d5+d7 d3+d5

d2+d4 d1+d3-d2-d6

end if

return (C)

end.

**PROGRAM**

#include<stdio.h>

int main(){

int a[2][2],b[2][2],c[2][2],i,j;

int m1,m2,m3,m4,m5,m6,m7;

printf("Enter the 4 elements of first matrix: ");

for(i=0;i<2;i++)

for(j=0;j<2;j++)

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

printf("Enter the 4 elements of second matrix: ");

for(i=0;i<2;i++)

for(j=0;j<2;j++)

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

printf("\nThe first matrix is\n");

for(i=0;i<2;i++){

printf("\n");

for(j=0;j<2;j++)

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

}

printf("\nThe second matrix is\n");

for(i=0;i<2;i++){

printf("\n");

for(j=0;j<2;j++)

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

}

m1= (a[0][0] + a[1][1])\*(b[0][0]+b[1][1]);

m2= (a[1][0]+a[1][1])\*b[0][0];

m3= a[0][0]\*(b[0][1]-b[1][1]);

m4= a[1][1]\*(b[1][0]-b[0][0]);

m5= (a[0][0]+a[0][1])\*b[1][1];

m6= (a[1][0]-a[0][0])\*(b[0][0]+b[0][1]);

m7= (a[0][1]-a[1][1])\*(b[1][0]+b[1][1]);

c[0][0]=m1+m4-m5+m7;

c[0][1]=m3+m5;

c[1][0]=m2+m4;

c[1][1]=m1-m2+m3+m6;

printf("\nAfter multiplication using \n");

for(i=0;i<2;i++){

printf("\n");

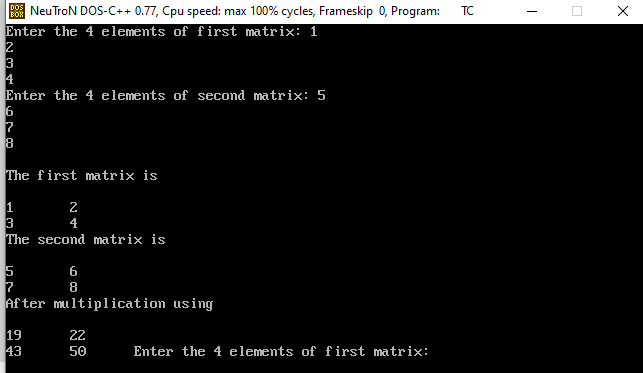
for(j=0;j<2;j++)

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

}

return 0;

}

**INPUT & OUTPUT**

**RESULT**

Thus the C Program for multiplication of 2\*2 elements Using Strassen Matrix multiplication has been executed and the output has been verified