



**MVPS"s
RAJARSHI SHAHU MAHARAJ POLYTECHNIC,
NASIK**

Subject

Data Structure Using "C" (22317)

**MICRO-PROJECT
ON**

"Develop C program to perform all sorting method by giving choice to sort with help of switch statement"

Submitted By

| SR. NO | ENROLLMENT NO | EXAM SEAT NO | STUDENT NAME |
|---------------|----------------------|---------------------|-----------------------|
| 1 | 1910020362 | 240630 | Savant Omkar Vitthal |
| 2 | 1910020360 | 240628 | Raut Atharva Satish |
| 3 | 1610020163 | 240586 | Wani Pushpak Shrikant |

Guided By

Prof. G.N.Handge

COMPUTER TECHNOLOGY DEPARTMENT

ACADEMIC YEAR 2020-21



MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION

CERTIFICATE

This is to certify that Ms./Mr. Savant Omkar Vitthal
Roll No. 45 of 3rd semester of Diploma in **Computer Technology** of Institute
MVPS's Rajarshi Shahu Maharaj Polytechnic, Nasik (Code: 1002) has
successfully completed micro-project in **Data Structure Using „C“ (22317)** for
academic year 2020-21 as prescribed in curriculum of MSBTE, Mumbai.

Place: Nashik

Enrollment no.: 1910020362

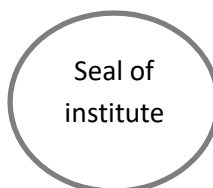
Date:.....

Exam seat no: 240630

Prof.G.N.Handge
Course Teacher/Guide

Prof.P.D.Boraste
H.O.D

Dr.D.B.Uphade
Principal





MVPS's RAJARSHI SHAHU MAHARAJ POLYTECHNIC, NASIK
Institute Code: 1002
COMPUTER TECHNOLOGY DEPARTMENT

Log Book for Micro Project

Academic Year :- 2020-21

Semester :- III

Name of Course :- Data Structure Using „C“

Scheme :- I

Class:- SYCM

Course Code:- (22317)

Title of the project: Develop C program to perform all sorting method by giving choice to sort with help of switch statement

Group Members:-

| Sr. No. | Roll No. | Enrollment Number | Exam Seat No. | Name of the Student | Signature of student |
|---------|----------|-------------------|---------------|-----------------------|----------------------|
| 1 | 45 | 1910020362 | 240630 | Savant Omkar Vitthal | |
| 2 | 43 | 1910020360 | 240628 | Raut Atharva Satish | |
| 3 | 1 | 1610020163 | 240586 | Wani Pushpak Shrikant | |

Project Reporting:

| Sr. No. | Date | Discussion & details | Group members present | Teacher's comment/remark | Signature of teacher |
|---------|------|--|-----------------------|--------------------------|----------------------|
| 1 | | Formation of groups | | | |
| 2 | | Discussion on concept of Micro Project | | | |
| 3 | | Topic selection for the Micro Project | | | |
| 4 | | Preliminary discussion with guide | | | |
| 5 | | Submission of Micro Project proposal | | | |
| 6 | | Information Gathered | | | |
| 7 | | Literature survey (Introduction) | | | |
| 8 | | Discussion with guide | | | |
| 9 | | Information Sorting | | | |
| 10 | | Chapter Sequencing and Designing. | | | |
| 11 | | Reporting to guide about Chapters. | | | |
| 12 | | Merging of all individual chapters into one single project | | | |
| 13 | | Draft copy of report. | | | |
| 14 | | Final report | | | |
| 15 | | Presentation & oral | | | |
| 16 | | Final submission | | | |

Prof.G.N.Handge
Name & Signature of Course Teacher/Guide



**MVPS"s RAJARSHI SHAHU MAHARAJ
POLYTECHNIC,NASIK**

Institute Code: 1002

COMPUTER TECHNOLOGY DEPARTMENT

Rubrics for Evaluation of Micro Project

Academic Year :- 2020-21

Scheme :- I

Semester :- III

Class:- SYCM

Name of Course :- Data Structure Using „C“

Course Code:- (22317)

Group Members:-

| Sr. No. | Roll No. | Enrollment Number | Exam Seat No. | Name of the Student | Signature of student |
|---------|----------|-------------------|---------------|-----------------------|----------------------|
| 1 | 45 | 1910020362 | 240630 | Savant Omkar Vitthal | |
| 2 | 43 | 1910020360 | 240628 | Raut Atharva Satish | |
| 3 | 1 | 1610020163 | 240586 | Wani Pushpak Shrikant | |

| Sr.No. | Criteria | Indicators of different levels of performance | | | | | Marks obtained |
|-----------------------|---|---|-------------------|-----------|----------------|----------------|----------------|
| | | Poor (01) | Satisfactory (02) | Good (03) | Very Good (04) | Excellent (05) | |
| 1 | Selection Of Topic | | | | | | |
| 2 | Concept/ Content/ Function Descriptions | | | | | | |
| 3 | Report Designing and Understanding | | | | | | |
| 4 | Questions and Answers | | | | | | |
| 5 | Timely Submission | | | | | | |
| Total marks Out of 25 | | | | | | | |
| Marks Out of 6 | | | | | | | |

Prof.G.N.Handge
Name & Signature of Course Teacher/Guide



**MVPS's RAJARSHI SHAHU MAHARAJ
POLYTECHNIC, NASIK**

Institute Code: 1002

COMPUTER TECHNOLOGY DEPARTMENT

ANNEXTURE II

Academic Year: 2020-2021

Name of Faculty: Prof.G.N.Handge

Course: Data Structure Using „C“

Course code: (22317) Semester: III

Title of the project: Develop C program to perform all sorting method by giving choice to sort with help of switch statement

COs addressed by the Micro Project:

CO-317.1 Perform basic operation on array.

CO-317.2 Apply different searching and sorting techniques.

CO-317.3 Implement basic operations on stack and queue using array representation.

CO-317.4 Implement basic operations on Linked List.

CO-317.5 Implement program to create and traverse tree to solve problems.

Major learning Outcomes achieved by students by doing the Project:

a) Practical Outcomes

.....
.....

b) Unit outcomes in cognitive domain

.....
.....

c) Outcomes in affective domain

.....
.....

Comment/Suggestions about team work/leadership/inter-personal communication (if any)

.....
.....

| Roll. No. | Enrollment no | Exam seat no | Student Name | Marks out of 6 for performance in group activity | Marks out of 4 for performance oral/ presentation | Total out of 10 |
|-----------|---------------|--------------|-----------------------|--|---|-----------------|
| 45 | 1910020362 | 240630 | Savant Omkar Vitthal | | | |
| 43 | 1910020360 | 240628 | Raut Atharva Satish | | | |
| 1 | 1610020163 | 240586 | Wani Pushpak Shrikant | | | |

Prof. G.N.Handge
Course Teacher/Guide

Prof.P.D.Boraste
H.O.D

ABSTRACT

This paper is scrutinizes the use of different Data Structures in C programming language, enabling viewer to get the complete concept of different aspects of C programming. Sorting is one of the most fundamental problems in computer science, as it is used in most software applications To satisfy this we created a simple menu-driven program displaying various sorting programs. We'll look at two searching algorithms and four sorting algorithms here in detail, and go through examples of each algorithm and determine the performance of each algorithm, in terms of how “quickly” each algorithm completes its task.

Table of Content

| | |
|---|-------|
| 1. Introduction..... | 3-12 |
| a) Sorting..... | 3 |
| b) Bubble Sort..... | 4 |
| c) Selection Sort..... | 6 |
| d) Quick Sort..... | 8 |
| e) Insertion Sort..... | 10 |
| 2. Design / Implementation..... | 12-17 |
| a) Algorithm..... | 12 |
| b) Flowchart..... | 13 |
| c) Code..... | 14 |
| 3. Output And Analysis..... | 18 |
| a) Output..... | 18 |
| b) Analysis..... | 18 |
| 4. Conclusion And Future Enhancement..... | 19 |
| a) Conclusion..... | 19 |
| b) Future Enhancement..... | 19 |
| 5. References..... | 20 |

Introduction

1.Sorting:

The process of Sorting can be explained as a technique of rearranging the elements in any particular order, which can be set ready for further processing by the program logic. In C programming language, there are multiple sorting algorithms available, which can be incorporated inside the code. The various types of sorting methods possible in the C language are Bubble sort, Selection sort, Quick sort, Merge sort, Heap sort and Insertion sort.

2. How Sorting is Performed in C?

Sorting can be performed in various ways based on the sorting algorithm. In C programming language we do have several approaches to sort the list. The term sorting states arranging of data in a particular manner usually in ascending order. Though the way to sort the data is different in all of the sorting algorithms, the outcome of all of them is the same.

Usually, in sorting, the program searches for the minimum number and shifted that number to the beginning of the list and repeat the same searches. Again once the other small number is encountered, it is shifted to the next space in the list right after the first index and this process keeps on repeating until the sort list is obtained. This is the way sorting is done in the C programming language.

In all the approaches to sort the list, the array plays a very vital role in the C programming language. In every algorithm, the array has been used to store the list of the elements that have to be sorted. For instance, in bubble sort, the elements are stored in the single array and the values in the array have been processed to convert them into a list of sorted data.

In the selection sort, the same array has been treated as two arrays where the first array is considered to be vacant in order to tell the sorted values while the second array holds the unsorted list. To serve the purpose of sorting the array is used very often instead of holding the values in individual variables. Among all of the algorithms, quick sort works very quick and hence named quick sort. It takes much less time as compared to the other sorting algorithms.

Types of Sorting in C:

1. Bubble Sort:

Bubble sort may be defined as the sorting algorithm that follows the approach of replacing the value in the first index with the smallest value in the array and keep it repeating until the list is sorted. It is a very simple way of performing sorting. In this way to sort the array, the value has to be assigned to the array in the beginning before starting the sorting.

Below is the program to sort the array using bubble sort where the values have been taken from the user. Once the program is compiled and run, it will ask the user for the number of elements that they want to sort. Once the number is provided, the program will ask the user to provide values equivalent to the count that they have provided. The values will be stored in the array and will be processed further using nested for loop together with decision making using “if” in order to sort the array.

The first smallest value found in the array has been moved to the first index of the array and then the search begins again to find the other smallest number. Once the next smallest number is found, it replaces the value in the second index and the process keeps on repeating until the array consists of a sorted list of values.

Output:

```
How many number you want to input?
7
Please enter 7 integers that has to be sorted
5
3
66
14
1
2
645
Below is the list of elements sorted in ascending order:
1
2
3
5
14
66
645
```

Code:

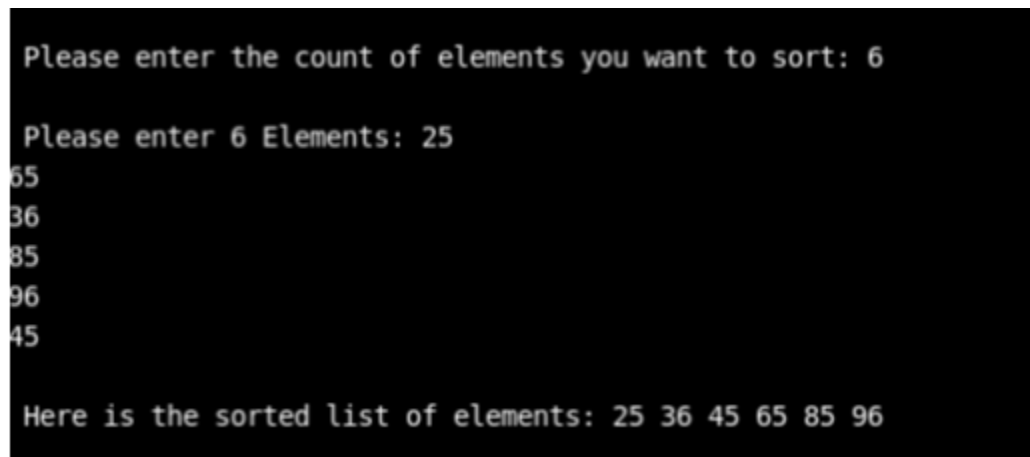
```
#include <stdio.h>
int main()
{
int total_count, counter, counter1, swap_var;
int array[20];
printf("How many number you want to input?\n");
scanf("%d", &total_count);
printf("Please enter %d integers that has to be
sorted\n", total_count);
for (counter = 0; counter < total_count; counter++)
scanf("%d", &array[counter]);
for (counter = 0 ; counter < total_count - 1;
counter++)
{
for (counter1 = 0 ; counter1 < total_count - counter -
1; counter1++)
{
if (array[counter1] > array[counter1+1]) /* For
decreasing order use < */
{
swap_var      = array[counter1];
array[counter1]  = array[counter1+1];
array[counter1+1] = swap_var;
}
}
}
printf("Below is the list of elements sorted in
ascending order:\n");
for (counter = 0; counter < total_count; counter++)
printf("%d\n", array[counter]);
return 0;
}
```

2. Selection Sort:

Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list. The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.

This algorithm is not suitable for large data sets as its average and worst case complexities are of $O(n^2)$, where n is the number of items.

Output:



```
Please enter the count of elements you want to sort: 6
Please enter 6 Elements: 25
65
36
86
96
45

Here is the sorted list of elements: 25 36 45 65 86 96
```

On asking for the count of elements that has to be sorted, the user has provided 6 in the below output. Later the values that have been input are 25 65 36 86 96 45. These values are stored in the array which is expected to be bifurcated into two arrays where one will be empty to store the sorted list and the other will be having the unsorted list. After processing the input, the outcome was 25 36 45 65 86 96. This list has been sorted using the selection sort. Once all the six values have been moved to the first array in the sorted form, the second array will become empty and the algorithm will be terminated.

Code:

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int total_count,counter1,counter2,minimum,temp_value;
    int a[20];
    printf("\n Enter the Number of Elements: ");
    scanf("%d",&total_count);
    printf("\n Enter %d Elements: ",total_count);
    for(counter1=0;counter1<total_count;counter1++)
    {
        scanf("%d",&a[counter1]);
    }
    for(counter1=0;counter1<total_count-1;counter1++)
    {
        minimum=counter1;
        for(counter2=counter1+1;counter2<total_count;counter2++)
        {
            if(a[minimum]>a[counter2])
            minimum=counter2;
        }
        if(minimum!=counter1)
        {
            temp_value=a[counter1];
            a[counter1]=a[minimum];
            a[minimum]=temp_value;
        }
    }
    printf("\n The Sorted array in ascending order: ");
    for(counter1=0;counter1<total_count;counter1++)
    {
        printf("%d ",a[counter1]);
    }
    getch();
}
```

3. Quick Sort:

Quicksort can be defined as the other algorithm for sorting the list in which the approach is to divide the array in terms of greater than and less than values until the entire values are divided into individual forms. In this algorithm, the value of the last index of the array has been selected as a pivot and all the values smaller than pivot have been shifted to the array that is expected to occur in the left of the value and the elements having a higher value than the pivot are shifted to the right array. Again one pivot is selected from the newly formed array that had the values less than the last pivot value. Similarly, the values smaller than the new pivot will be shifted to the array that will be left and the values more than the new pivot will be shifted in the right array.

The below program is the quicksort implementation using the C programming language. Once the program runs, it will ask the user for the number of elements that they want to sort. Based on the count, the for loop will iterate estimated times to take the input from the user. The input will be processed using the if conditions together with the for loop in order to generate a sorted list. The array will keep on arranging the values using the pivot value until all the values have been checked for the smallest value.

The sorting done using this algorithm is way too faster as compared to the other sorting algorithms and that's why it has been named quick sort. Quicksort is the only algorithm that leads to dividing the array until all the values are separated into the individual arrays. They will be then added or aggregated in a single array which is considered as the sorted list.

Output:

```
Please enter the total count of the elements that you want to sort: 6
Please input the elements that has to be sorted:
56
35
24
86
98
2
Output generated after using quick sort
2 24 35 56 86 98
```

Code:

```
#include <stdio.h>
void quicksort_method (int [], int, int);
int main()
{
    int element_list[50], count, counter;
    printf("Please enter the total count of the elements that you want to sort: ");
    scanf("%d", &count);
    printf("Please input the elements that has to be sorted:\n");
    for (counter = 0; counter < count; counter++)
    {
        scanf("%d", &element_list[counter]);
    }
    quicksort_method(element_list, 0, count - 1);
    printf("Output generated after using quick sort\n");
    for (counter = 0; counter < count; counter++)
    {
        printf("%d ", element_list[counter]);
    }
    printf("\n");
    return 0;
}

void quicksort_method(int element_list[], int low, int high)
{
    int pivot, value1, value2, temp;
    if (low < high)
    {
        pivot = low;
        value1 = low;
        value2 = high;
        while (value1 < value2)
        {
            while (element_list[value1] <= element_list[pivot] && value1 <= high)
            {
                value1++;
            }
            while (element_list[value2] > element_list[pivot] && value2 >= low)
            {
                value2--;
            }
            if (value1 < value2)
            {
                temp = element_list[value1];
                element_list[value1] = element_list[value2];
                element_list[value2] = temp;
            }
        }
        temp = element_list[value2];
        element_list[value2] = element_list[pivot];
        element_list[pivot] = temp;
        quicksort_method(element_list, low, value2 - 1);
        quicksort_method(element_list, value2 + 1, high);
    }
}
```

4. Insertion Sort

Insertion sort may be defined as the sorting algorithm that works by moving the minimum value at the beginning of the list one at a time. This is a very less efficient sorting algorithm and not found suitable to deal with the large list.

This approach of sorting the algorithm works very slowly and usually not preferred in any of the applications. It can work good with the list that has pretty few numbers of elements. For the applications, that have the requirement to process a few numbers of values can leverage this algorithm.

Output:

```
Please enter the total count of the elements that you want to sort:
6
Please input the elements that has to be sorted:
35
24
68
95
62
3

Output generated after using insertion sort
3 24 35 62 68 95
```

When the program runs, the user will have to input the number of values that they need to sort. Afterward, the values entered by the user will be stored into the array. They will then go under processing and by the use of for loop and condition checking, the minimum value will be moved to the beginning in every recursion and end up by generating a sorted array. The values will be displayed to the user at the end of the program.

Code:

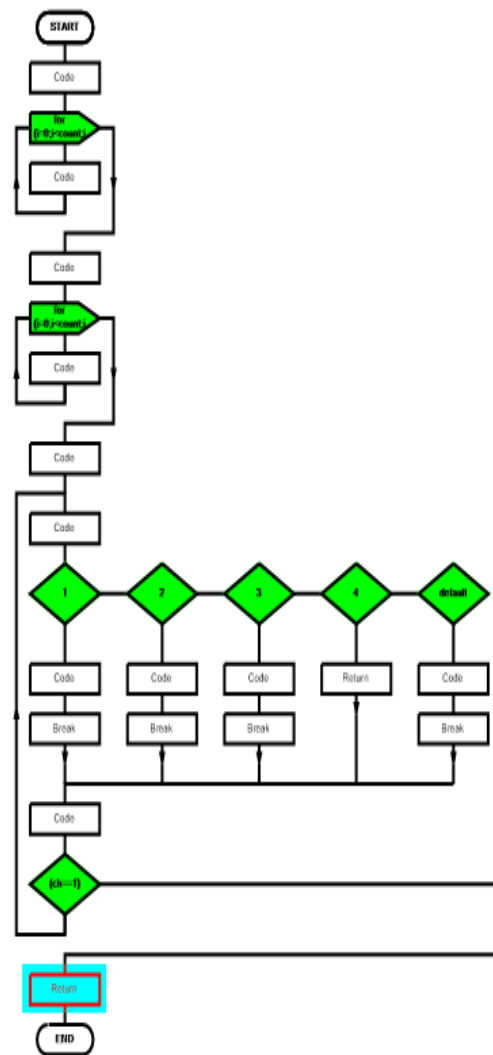
```
#include<stdio.h>
int main()
{
int counter1,counter2,chk,temp_val,val[100];
printf("Please enter the total count of the elements
that you want to sort: \n");
scanf("%d",&chk);
printf("Please input the elements that has to be
sorted:\n");
for(counter1=0;counter1<chk;counter1++)
{
scanf("%d",&val[counter1]);
}
for(counter1=1;counter1<=chk-1;counter1++)
{
temp_val=val[counter1];
counter2=counter1-1;
while((temp_val<val[counter2])&&(counter2>=0))
{
val[counter2+1]=val[counter2];
counter2=counter2-1;
}
val[counter2+1]=temp_val;
}
printf("\n Output generated after using insertion sort
\n");
for(counter1=0;counter1<chk;counter1++)
{
printf("%d ",val[counter1]);
}
return 0;
}
```

Design/Implementation:

Algorithm:

- 1.Start
- 2.Declare ch
- 3.Switch Statement
 1. If i=1 then go to step 4
 2. If i=2 then go to step 5
 3. If i=3 then go to step 6
 4. If i=4 then return 0
- 4.Initiate Bubblesort()
- 5.Initiate Selectionsort()
- 6.Inititate Insertionsort()
- 7.Stop

Flowchart:



Code:

```
#include<stdio.h>
#include<stdlib.h>
void bubble(int *,int);
void selection(int *,int);
void insertion(int *,int);
int main()
{
    int count=0;           //size of array
    int choice=0,ch=0;      //variables used to store user choice
    int check=0;           //used to check status
    int i=0;               //loop variable
    printf("\nEnter the size of the list: ");
    scanf("%d",&count);
    //creating array of appropriate size
    int list[count];
    //filling in the array
    for(i=0;i<count;i++)
    {
        printf("\nEnter element %d: ",i+1);
        scanf("%d",&list[i]);
    }
    //prints the list
    printf("\tNumbers entered: ");
    for(i=0;i<count;i++)
        printf("%d,",list[i]);
    printf("\n");
    printf("\n");
    // Menu
    do{
        printf("\tMenu:\n");
        printf("\t1.Bubble sort\n\t2.Selection Sort\n\t3.Insertion sort\n\t4.Exit\n\tYour choice: ");
        scanf("%d",&choice);
        switch(choice)
```

```

{
    case 1:
        bubble(list,count);
        break;
    case 2:
        selection(list,count);
        break;
    case 3:
        insertion(list,count);
        break;
    case 4: return 0;
    default: printf("Invalid option\nRetry: ");
        break; }
    printf("\tDo you want to continue(press 1 to continue):");
    scanf("%d",&ch);
    printf("\n");
} while(ch==1);
return 0;
}

void bubble(int *list,int n) //bubble
{
    int i,j;
    int c;
    for(i=0;i<n;i++)
    {
        for (j=0;j<n-i-1;j++)
        {
            if (list[j] > list[j+1])
            {
                c=list[j];
                list[j]=list[j+1];
                list[j+1]=c;
            }
        }
    }
    printf("\tSorted list in ascending order:");

```

```

    for ( i = 0 ; i < n ; i++ )
        printf("%d,",list[i]);
    printf("\n");
}

void selection(int *list,int n) //selection{
int i;
    int j,min;
    int k;
    for(j=0;j<n-1;j++)
    {
        min=list[j];
        k=j;
        for(i=j+1;i<n;i++)
        {
            if(list[i]<min)
            {
                min=list[i];
                k=i;
            }
        }
        list[k]=list[j];
        list[j]=min;
    }printf("\tSorted list is:");
    for(i=0;i<n;i++)
    {
        printf("%d,",list[i]);
    }
    printf("\n");
}

void insertion(int *list,int n) //insertion
{
    int temp;
    int i=0,j=0;

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

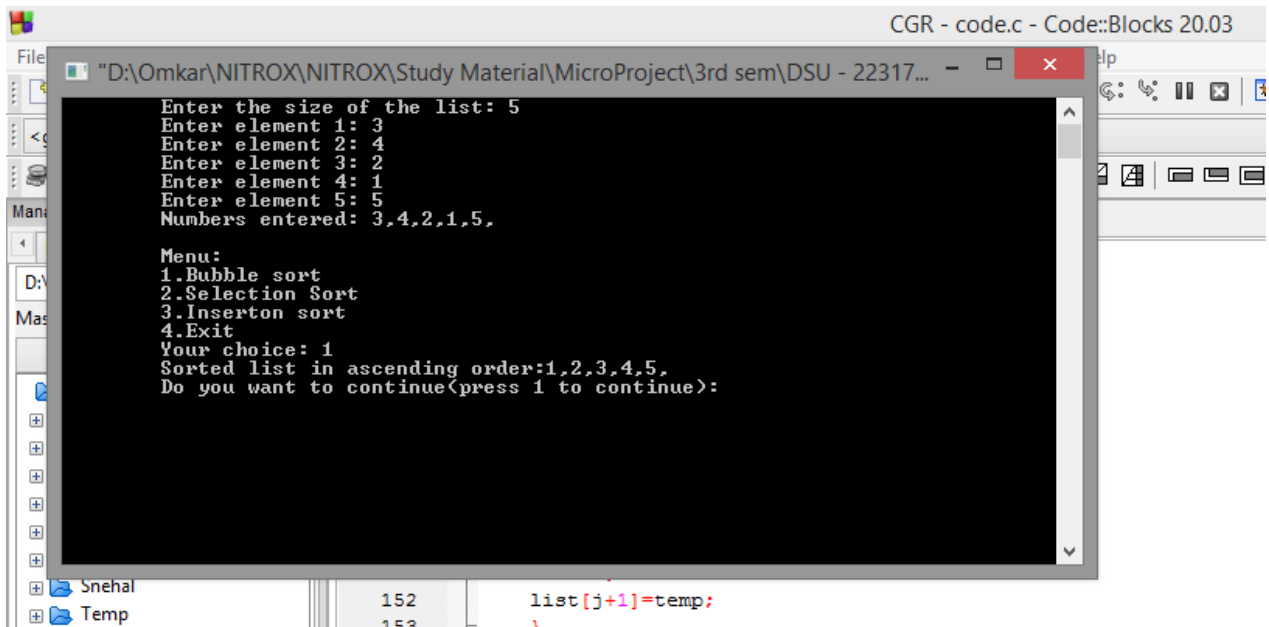
```

```
        temp=list[i];
        j=i-1;
        while ((j>=0) && (list[j]>temp))
        {
            list[j+1]=list[j];
            j--;
        }
        list[j+1]=temp;
    }

    printf("\tSorted list is: ");
    for (i=0;i<n;i++) {
        printf("%d,",list[i]);
    }
    printf("\n");
}
```

Output and Analysis

Output:



```
Enter the size of the list: 5
Enter element 1: 3
Enter element 2: 4
Enter element 3: 2
Enter element 4: 1
Enter element 5: 5
Numbers entered: 3,4,2,1,5,

Menu:
1.Bubble sort
2.Selection Sort
3.Insertion sort
4.Exit
Your choice: 1
Sorted list in ascending order:1,2,3,4,5,
Do you want to continue<press 1 to continue>:
```

Analysis:

C language was created by Dennis Ritchie. C language has a great aspect in its own field. It has various syntaxes and its own special structure, which makes it a unique. With use of different loops and statements the following output is created as requested. We created first an algorithm to first understand the concept of it. After creating the algorithm we understood the concept of microproject.

After Algorithm, we created the Flowchart which enabled us to get a graphical representation of the program. Next we created the program which gives us the following output.

Switch Statement makes the program user friendly a lot. It makes user to select it's suitable option of choice without any mistake.

Last but not least all the compilers used while writing the code were of great use.

Conclusion and Future Enhancement

Conclusion:

The sorting algorithm is used to generate a sorted list which is a normal list where all the values are sorted in a particular manner. The list has been used very often in the actual application to bring some functionalities. In this article we have covered bubble sort, selection sort, and quicksort while there are several other algorithms like merge sort are also there that can be leveraged to generate a sorted list. Among all of the sorting algorithms, quicksort works very fast and helps to sort the list very quickly. The programs written here are basically to implement these sorting algorithms using the C programming language. If you are willing to implement the same in other programming languages, you can use the same logic and the only thing that may vary can be the syntax and keywords.

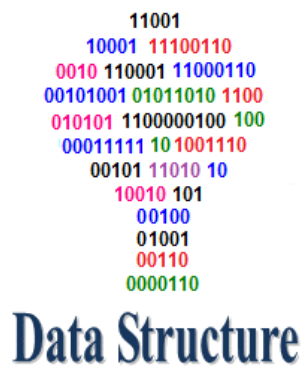
Future Enhancement:

Data Structures in C are used to store data in an organised and efficient manner. The C Programming language has many data structures like an array, stack, queue, linked list, tree, etc. A programmer selects an appropriate data structure and uses it according to their convenience.

The data structure is not any programming language like C, C++, java, etc. It is a set of algorithms that we can use in any programming language to structure the data in the memory.

The data structure name indicates itself that organizing the data in memory. There are many ways of organizing the data in the memory as we have already seen one of the data structures, i.e., array in C language. Array is a collection of memory elements in which data is stored sequentially, i.e., one after another. In other words, we can say that array stores the elements in a continuous manner. This organization of data is done with the help of an array of data structures. There are also other ways to organize the data in memory.

To structure the data in memory, 'n' number of algorithms were proposed, and all these algorithms are known as Abstract data types. These abstract data types are the set of rules.



References

- [1] Cederman D, Tsigas P. GPU-quicksort: A practical quicksort algorithm for graphics processors. *J. Exp. Algorithmics* Jan 2010; 14:4:1.4–4:1.24, doi:10.1145/1498698.1564500. [2] Satish N, Harris M, Garland M. Designing efficient sorting algorithms for manycore GPUs. *Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing, IPDPS '09*, IEEE Computer Society: Washington, DC, USA, 2009; 1–10, doi:10.1109/IPDPS.2009.5161005.
- [3] Knuth, Donald E. [1998]. *The Art of Computer Programming, Volume 3, Sorting and Searching*. Addison-Wesley, Reading, Massachusetts.
- [4] Batcher KE. Sorting networks and their applications. *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference, AFIPS '68 (Spring)*, ACM: New York, NY, USA, 1968; 307–314, doi:10.1145/1468075 [5] Singleton R C. Algorithm 347: An efficient algorithm for sorting with minimal storage [m1]. *Commun. ACM* Mar 1969; 12(3):185–186, doi:10.1145/362875.362901