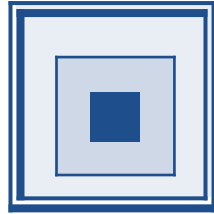


**YOUR INSTITUTE OF TECHNOLOGY, HYDERABAD**

Department of Computer Science & Engineering



*A Practical File on*

# **Data Structures Lab**

## **Practical File**

B.Tech CSE · Programs in C

Submitted in partial fulfilment of the practical examination requirements

---

SUBMITTED BY

Arjun Varma  
Roll No. 24CS1057

PROGRAMME

B.Tech III Sem, CSE-A

SUBMITTED TO

Ms. Sneha Reddy

**Session 2025–26**

# CERTIFICATE

This is to certify that Arjun Varma, a student of B.Tech III Sem, CSE-A, has satisfactorily completed the practical work in Data Structures Lab prescribed by the Department of Computer Science & Engineering during the academic session 2025–26.

The experiments recorded in this file were performed by the student in the laboratory under my supervision, and the observations and results are the student's own.

-----  
**Ms. Sneha Reddy**

Subject Teacher

-----  
**External Examiner**

Date: \_\_\_\_\_

-----  
**Dr. V. Srinivas**

Head of Department

---

# Acknowledgement

I would like to express my sincere gratitude to my subject teacher, Ms. Sneha Reddy, for guidance and encouragement throughout the practical sessions, and for patiently clearing my doubts in the laboratory.

I am also thankful to Dr. V. Srinivas, Head of Department, for providing a well-equipped laboratory, and to the laboratory staff for their help in arranging the apparatus for every experiment.

Finally, I thank my parents and classmates for their constant support while I completed this practical file.

**Arjun Varma**

# Index

| S. No. | Name of the Experiment        | Date        | Remarks | Teacher's Sign |
|--------|-------------------------------|-------------|---------|----------------|
| 1      | Linear and binary search      | 14 Jan 2026 |         |                |
| 2      | Bubble sort                   | 21 Jan 2026 |         |                |
| 3      | Stack using an array          | 28 Jan 2026 |         |                |
| 4      | Circular queue                | 04 Feb 2026 |         |                |
| 5      | Singly linked list operations | 11 Feb 2026 |         |                |

*Write the page number of each experiment in the remarks column once the file is complete.*

---

# Experiment 1: Linear and binary search

|                   |                 |                       |
|-------------------|-----------------|-----------------------|
| Date: 14 Jan 2026 | Page No.: _____ | Teacher's sign: _____ |
|-------------------|-----------------|-----------------------|

## Aim

To implement linear search and binary search on an array of integers in C.

## Software / Requirements

- GCC compiler
- Code::Blocks / VS Code

## Algorithm

1. Read n elements into an array (sorted for binary search).
2. Linear search compares the key with each element in turn.
3. Binary search compares with the middle element and halves the range.
4. Print the position or a not-found message.

## Program

C (gcc)

```
#include <stdio.h>
int binary(int a[], int n, int key) {
    int lo = 0, hi = n - 1;
    while (lo <= hi) {
        int mid = (lo + hi) / 2;
        if (a[mid] == key) return mid;
        if (a[mid] < key) lo = mid + 1; else hi = mid - 1;
    }
    return -1;
}
int main(void) {
    int a[] = {3, 8, 15, 21, 34, 47, 59}, n = 7, key = 34;
    int pos = binary(a, n, key);
    if (pos >= 0) printf("%d found at index %d\n", key, pos);
    else printf("%d not found\n", key);
    return 0;
}
```

## Output

Sample output

34 found at index 4

## Result

### Result

Both searches work correctly; binary search needs at most  $\log_2 n$  comparisons on sorted data.

## Viva Questions

**Q1.** What is the time complexity of binary search?

$O(\log n)$ .

**Q2.** What condition must the array satisfy for binary search?

It must be sorted.

## Experiment 2: Bubble sort

Date: 21 Jan 2026

Page No.: \_\_\_\_\_

Teacher's sign: \_\_\_\_\_

### Aim

To sort an array using bubble sort and count the number of swaps.

### Software / Requirements

- GCC compiler

### Algorithm

5. Read the array.
6. Repeatedly compare adjacent elements and swap them if they are out of order.
7. After each pass the largest remaining element settles at the end.
8. Print the sorted array and the swap count.

### Program

```
C (gcc)
#include <stdio.h>
void bubble(int a[], int n) {
    int swaps = 0;
    for (int i = 0; i < n - 1; i++)
        for (int j = 0; j < n - 1 - i; j++)
            if (a[j] > a[j + 1]) {
                int t = a[j]; a[j] = a[j + 1]; a[j + 1] = t; swaps++;
            }
    printf("Swaps: %d\n", swaps);
}
int main(void) {
    int a[] = {29, 10, 14, 37, 13}, n = 5;
    bubble(a, n);
    for (int i = 0; i < n; i++) printf("%d ", a[i]);
    return 0;
}
```

### Output

Sample output

```
Swaps: 5
10 13 14 29 37
```

## Result

### Result

The array is sorted in ascending order; bubble sort has  $O(n^2)$  time complexity.

---

## Experiment 3: Stack using an array

|                   |                 |                       |
|-------------------|-----------------|-----------------------|
| Date: 28 Jan 2026 | Page No.: _____ | Teacher's sign: _____ |
|-------------------|-----------------|-----------------------|

### Aim

To implement a stack of integers using an array with push, pop and peek operations.

### Software / Requirements

- GCC compiler

### Algorithm

9. Declare an array and a top index initialised to -1.
10. Push: check overflow, increment top and store.
11. Pop: check underflow, return the top element and decrement top.
12. Test the operations from main().

### Program

```
C (gcc)
#include <stdio.h>
#define MAX 5
int st[MAX], top = -1;
void push(int x) {
    if (top == MAX - 1) { printf("Overflow\n"); return; }
    st[++top] = x;
}
int pop(void) {
    if (top == -1) { printf("Underflow\n"); return -1; }
    return st[top--];
}
int main(void) {
    push(10); push(20); push(30);
    printf("Popped %d\n", pop());
    printf("Top is %d\n", st[top]);
    return 0;
}
```

### Output

Sample output

```
Popped 30
Top is 20
```

## Result

### Result

The stack follows LIFO order and handles overflow and underflow.

---

## Experiment 4: Circular queue

|                   |                 |                       |
|-------------------|-----------------|-----------------------|
| Date: 04 Feb 2026 | Page No.: _____ | Teacher's sign: _____ |
|-------------------|-----------------|-----------------------|

### Aim

To implement a circular queue using an array.

### Software / Requirements

- GCC compiler

### Algorithm

13. Use front and rear indices with modulo arithmetic.
14. Enqueue at rear if the queue is not full.
15. Dequeue from front if the queue is not empty.
16. Display elements from front to rear.

### Program

```
C (gcc)
#include <stdio.h>
#define N 4
int q[N], f = -1, r = -1;
void enq(int x) {
    if ((r + 1) % N == f) { printf("Full\n"); return; }
    if (f == -1) f = 0;
    r = (r + 1) % N; q[r] = x;
}
int deq(void) {
    int x = q[f];
    if (f == r) f = r = -1; else f = (f + 1) % N;
    return x;
}
int main(void) {
    enq(1); enq(2); enq(3); enq(4); enq(5);
    printf("Out: %d\n", deq());
    enq(5);
    printf("Out: %d\n", deq());
    return 0;
}
```

### Output

Sample output

```
Full
Out: 1
Out: 2
```

## Result

### Result

The circular queue reuses freed positions and detects the full condition correctly.

# Experiment 5: Singly linked list operations

Date: 11 Feb 2026

Page No.: \_\_\_\_\_

Teacher's sign: \_\_\_\_\_

## Aim

To create a singly linked list and perform insertion at the beginning, insertion at the end and display.

## Software / Requirements

- GCC compiler

## Algorithm

17. Define a node structure with data and next pointer.
18. Allocate nodes with malloc.
19. Insert at the beginning by updating head; at the end by traversing to the last node.
20. Traverse and print the list.

## Program

C (gcc)

```
#include <stdio.h>
#include <stdlib.h>
struct node { int data; struct node *next; };
struct node *head = NULL;
void push_front(int x) {
    struct node *n = malloc(sizeof *n);
    n->data = x; n->next = head; head = n;
}
void push_back(int x) {
    struct node *n = malloc(sizeof *n), *p = head;
    n->data = x; n->next = NULL;
    if (!head) { head = n; return; }
    while (p->next) p = p->next;
    p->next = n;
}
int main(void) {
    push_back(20); push_front(10); push_back(30);
    for (struct node *p = head; p; p = p->next) printf("%d -> ", p->data);
    printf("NULL\n");
    return 0;
}
```

## Output

Sample output

10 -> 20 -> 30 -> NULL

## Result

### Result

Nodes are inserted at both ends and the list is displayed correctly.

---

## Bibliography

- Laboratory manual issued by the department / school.
- Prescribed textbook for the subject (latest edition).
- Class notes and practical demonstrations by the subject teacher.