

# Lab Experiment- C Language

## Objective:

- To gain practical experience with advanced pointer concepts in C, including pointer arithmetic, pointers and arrays, and function pointers. Master usage of gcc compiler and gdb.

## Materials needed:

- Computer with a C compiler (e.g., GCC)
- Debugger (GDB)
- Text editor or IDE

## Part 1: Pointer Basics and Arithmetic

**Use this template code for this part onwards:** [template code Part1.c](#)

### Task 1.1

Create a program that demonstrates basic pointer usage:

- Declare an integer variable and a pointer to it
- Print the value of the variable using both direct access and the pointer
- Modify the value using the pointer and print the new value

### Task 1.2

Implement a function that swaps two integers using pointers

### Task 1.3

Create an array of integers and use pointer arithmetic to:

- Print all elements of the array
- Calculate the sum of all elements
- Reverse the array in-place

## Part 2: Pointers and Arrays

### Task 2.1

Create a 2D array (matrix) of integers and write functions to:

- Initialize the matrix with random values
- Print the matrix
- Find the maximum element in the matrix

### Task 2.2

Implement a function that takes a 2D array as a parameter and calculates the sum of each row

## Part 3: Function Pointers

### Task 3.1

Create an array of integers and implement the following sorting algorithms:

- Bubble sort
- Selection sort

### Task 3.2

Create a function pointer for the sorting algorithm and use it to sort the array

### Task 3.3

Implement a simple calculator program that uses function pointers to perform basic arithmetic operations (addition, subtraction, multiplication, division)

## Part 4: Advanced Challenge

### Task 4.1

Implement a simple linked list with the following operations:

- Insert a node at the beginning
- Delete a node by value
- Print the list

## Task 4.2

Use function pointers to create a generic linked list that can store different data types.

# Part 5: Dynamic Memory Allocation

## Task 5.1

Create a program that:

- Dynamically allocates an array of integers
- Allows the user to input the size of the array and its elements
- Calculates and prints the sum and average of the elements

## Task 5.2

Implement a function that uses `realloc()` to extend an existing dynamically allocated array

## Task 5.3

Create a simple memory leak detector:

- Write functions to allocate and free memory
- Keep track of allocated memory addresses
- Print a warning if the program ends with unfreed memory

# Part 6: Structures and Unions

## Task 6.1

Create a structure to represent a student with fields for name, ID, and grades in three subjects

## Task 6.2

Implement functions to:

- Input student data

## Setting Up the Environment

- Calculate the average grade
- Print student information

### Task 6.3

Create a nested structure to represent a university with departments and students

### Task 6.4

Implement a union to store different types of data (int, float, char) and demonstrate its usage

## Part 7: File I/O

### Task 7.1

Create a program that:

- Writes the student data (from Part 6) to a text file
- Reads the data back from the file and prints it

### Task 7.2

Modify the program to write and read the student data in binary format

### Task 7.3

Implement a simple log file system:

- Create functions to log messages with timestamps
- Allow appending to an existing log file
- Implement a function to read and display the log

## Helping Material:

- Vim:
  - [MIT OpenCourseware](#)

- C Language:
  - The C Programming Language by Kernighan & Ritchie ([available here](#))
  - C Tutorial – Tutorialspoint ([available here](#))
- GCC:
  - Opensource: [click here](#)
  - IOFlood: [click here](#)
  - GNU GCC Guide: [click here](#)
- GDB:
  - Geeks for Geeks: [click here](#)
  - Medium: [click here](#)
  - Baeldung: [click here](#)
  - How to forge: [click here](#)