Embedded C Puzzles – Summer Project Report

Name: Soumya Jain

Roll Number: MT2024154

Course: M.Tech CSE

Institute: IIIT Bangalore

Professor/Mentor: Prof. Kurian Polachan

Date of Submission: 27th July, 2025

Abstract

This project involved designing and compiling a comprehensive lab manual for the C Programming language. It includes concept-based examples based on basic data types, operators, control flow, macro usage, structure manipulation, pointer operations, logical operator precedence, storage classes. All programs are compiled and managed using a portable Makefile to enable ease of execution on any Linux system.

The notion page has chapter wise each concept explained, code example and line by line explanation of the output.

Link to Notion page: https://www.notion.so/kurianpolachan/Soumya-Embedded-C-Puzzles-22bd051c359e80cc877aeb5147008ec8?source=copy_link

Objective

- To reinforce core C programming concepts through code.
- To create reusable and organized code examples.
- To automate compilation via a system-independent Makefile.

Tools and Technologies Used

- GCC Compiler via WSL Ubuntu
- Bash / Makefile
- Linux File System
- Online compilers (for verification)

Project Description

The project includes a set of organized C programs demonstrating core topics of the C language. Programs are separated by concept such as data types, arrays, loops, structures, pointers, macros, and operators. A Makefile is provided to compile all programs independently into executables using GCC. Folder structure includes:

- `src/`: Contains all `.c` and `.h` files
- `bin/`: Stores compiled executables
- `Makefile`: Automates compilation
- `README.txt`: Contains compilation and execution instructions.

The zip folder containing makefile, /src, /bin is uploaded on Github: https://github.com/sj1412/C_Puzzles.git

How to run code examples?

Step 1: **Install Required Tools**

Open a terminal (on **Ubuntu/Linux/WSL**) and install tools if not already available:

Run the bash command: sudo apt update

Run the bash command: sudo apt install build-essential

This installs gcc and make.

Step 2: **Move and Extract the Folder**

If the .zip is in **Windows**, move it to your Linux home:

Run the bash commands:

- cp /mnt/c/Users/<YourName>/Desktop/C_Lab_Manual.zip ~/
- cd ~
- unzip C_Lab_Manual.zip
- cd C_Lab_Manual

Step 3: Build All Executables

In the C_Lab_Manual/ folder, run the bash command:

- make

This will compile all .c files in the src/ folder and generate executables in the bin/ folder.

Step 4: Run Any Program

To run a program (e.g., 1.2code-example.c), run the bash command:

- ./bin/1.2code-example.c

Conclusion

This project provided hands-on experience in writing, managing, and compiling multiple C programs. It strengthened understanding of Makefile automation, program modularity, and core programming concepts & helped me understand the nuances of C programming better.

Future Scope

- Add problem statements or questions for each code.
- Include error explanation and handling in comments.
- Add test scripts for verifying output correctness.

Appendix

- 1.1code-example.c –Basic Arithmetic Operator
- 1.2code-example.c –Assignment Operator
- 1.3code-example.c- Logic & Increment operator
- 1.4code-example.c-Bitwise Operator
- 1.5code-example.c-Relational & Conditional Operators
- 1.6code-example.c-Operator Precedence and Evaluation
- 2.1code-example.c- Character, String & integer
- 2.2code-example.c- Integer and Floating Point Casts
- 2.3code-example.c- More Casts
- 3.1code-example.c- If statement
- 3.2code-example.c- While and For Statements
- 3.3code-example.c- Statement Nesting
- 3.4code-example.c -Switch, Break, and Continue Statements
- 4.1code-example.c- Simple Pointer & Array
- 4.2code-example.c- Array Of Pointers
- 4.3code-example.c- Multi-dimensional Array
- 4.4code-example.c- Pointer Stew
- 5.1code-example.c- Simple Structures & Nested Structures
- 5.2code-example.c- Array of Structures
- 5.3code-example.c- Array of Pointers to Structures
- 6.1code-example.c-Storage Classes 1: Blocks
- 6.2code-example.c-Storage Classes 2: Functions
- 6.3code-example.c-Storage Classes 3: More Functions
- 6.4code-example.c-Storage Classes 4: Files