15 C Programming Project Ideas with Instructions

Hello, World! Program

Objective: Print 'Hello, World!' to the console.

Concepts Covered: Basic syntax, printf function, and compilation process.

Instructions:

1. Set up your development environment.
2. Create a new file named hello_world.c.
3. Write code to print 'Hello, World!' to the console.
4. Compile and run your program.

Simple Calculator

Objective: Create a program that can perform basic arithmetic operations (addition, subtraction, multiplication, division).

Concepts Covered: Variables, data types, basic I/O, conditional statements, and loops.

Instructions:

1. Create a new file named calculator.c.
2. Prompt the user for two numbers and an operation (+, -, *, /).
3. Perform the chosen operation and display the result.
4. Implement error handling for invalid inputs.

Temperature Converter

Objective: Convert temperatures between Celsius, Fahrenheit, and Kelvin.

Concepts Covered: Functions, user input, mathematical operations.

Instructions:

1. Create a new file named temp_converter.c.
2. Implement functions to convert between Celsius, Fahrenheit, and Kelvin.

3. Prompt the user for the input temperature and the target unit.

4. Perform the conversion and display the result.

Guessing Game

Objective: Implement a number guessing game where the program randomly selects a number, and the user has to guess it.

Concepts Covered: Random number generation, loops, conditional statements.

Instructions:

1. Create a new file named guessing_game.c.

2. Generate a random number between 1 and 100.

3. Prompt the user to guess the number and provide feedback (higher/lower).

4. Loop until the user guesses the correct number.

Basic File I/O

Objective: Read from and write to a text file.

Concepts Covered: File handling functions (fopen, fclose, fgets, fprintf), error handling.

Instructions:

1. Create a new file named file_io.c.

2. Implement functionality to write a message to a file.

3. Implement functionality to read the contents of the file and display them.

4. Add error handling for file operations.

Array Operations

Objective: Perform operations like finding the maximum, minimum, average, and sorting on an array of integers.

Concepts Covered: Arrays, loops, functions, basic algorithms (sorting).

Instructions:

1. Create a new file named array_operations.c.

2. Implement functions to find the maximum, minimum, and average of an array.

3. Implement a sorting function (e.g., bubble sort).

4. Test the functions with sample arrays.

Simple Text Editor

Objective: Create a program that allows basic text editing functionalities like adding, deleting, and saving text.

Concepts Covered: String manipulation, file handling, user interface basics.

Instructions:

1. Create a new file named text_editor.c.

2. Implement functionality to add, delete, and display text.

3. Implement functionality to save and load text from a file.

4. Add a simple user interface for text editing.

Tic-Tac-Toe Game

Objective: Implement a simple tic-tac-toe game for two players.

Concepts Covered: 2D arrays, game logic, user input, loops, conditional statements.

Instructions:

1. Create a new file named tic_tac_toe.c.

2. Implement a 2D array to represent the game board.

3. Implement game logic to check for wins and draws.

4. Prompt players for their moves and update the game board.

5. Display the game board after each move.

Linked List Implementation

Objective: Implement a singly linked list with basic operations like insertion, deletion, and traversal.

Concepts Covered: Dynamic memory allocation, pointers, structs.

Instructions:

1. Create a new file named linked_list.c.

2. Define a struct for the linked list node.

3. Implement functions for insertion, deletion, and traversal.

4. Test the functions by creating and manipulating a linked list.

Basic Banking System

Objective: Simulate a simple banking system where users can create accounts, deposit, withdraw, and check balance.

Concepts Covered: Structs, file handling, functions, loops, conditional statements.

Instructions:

1. Create a new file named banking_system.c.

2. Define a struct for user accounts.

3. Implement functions for creating accounts, depositing, withdrawing, and checking balance.

4. Implement file handling to save and load account information.

Simple Shell

Objective: Create a basic command-line shell that can execute simple commands like ls, pwd, and cd.

Concepts Covered: System calls, process management, string manipulation, loops.

Instructions:

1. Create a new file named simple_shell.c.

2. Implement functionality to read and parse user input.

3. Implement functionality to execute basic commands using system calls.

4. Add error handling for invalid commands.

Hangman Game

Objective: Implement the hangman game where the player guesses letters of a hidden word.

Concepts Covered: Arrays, strings, loops, conditional statements, user input.

Instructions:

1. Create a new file named hangman_game.c.

2. Define an array to store the hidden word and the player's guesses.

3. Implement game logic to check guesses and update the game state.

4. Prompt the player for guesses and display the current state of the word.

Basic Data Structures

Objective: Implement basic data structures like stacks and queues with their respective operations.

Concepts Covered: Structs, dynamic memory allocation, pointers, functions.

Instructions:

1. Create a new file named data_structures.c.

2. Define structs for stacks and queues.

3. Implement functions for stack and queue operations (push, pop, enqueue, dequeue).

4. Test the functions by creating and manipulating stacks and queues.

Student Record System

Objective: Develop a system to manage student records with functionalities like adding, deleting, and displaying records.

Concepts Covered: Structs, file handling, arrays, functions.

Instructions:

1. Create a new file named student_records.c.

2. Define a struct for student records.

3. Implement functions to add, delete, and display student records.

4. Implement file handling to save and load student records.

Mini-Music Player

Objective: Create a simple music player that can play, pause, and stop songs (using external libraries).

Concepts Covered: External libraries, basic user interface, file handling.

Instructions:

1. Create a new file named mini_music_player.c.

2. Integrate an external library for audio playback.

3. Implement functions to play, pause, and stop songs.

4. Implement a simple user interface for controlling the music player.