

**Fitness Tracker Application**

**Mini Project**



**Name :** BIBIN BENNY

**Roll No :** 23

**Course Name:** C programming

**INTRODUCTION**

Project overview:

The purpose of this project is to develop a system for managing and tracking personal fitness activities. The system is designed to enable users to record and view daily fitness data such as steps taken, calories burned, and progress towards fitness goals. Additionally, it allows users to update existing records, ensuring accurate and up-to-date information. Future enhancements include user authentication, a graphical user interface, advanced data analytics, integration with wearable devices, notifications and reminders, and data export/import capabilities.

Problem statement:

Develop an application to efficiently track fitness activities, including steps taken, calories burned, and progress towards fitness goals.

Objective:

Create an application to manage and track fitness activities, ensuring users can easily monitor their steps taken, calories burned, and progress towards fitness goals.

**System Requirements**

**Minimum Requirements for C Programming Code to Run:**

**➢Hardware Requirement:**

* A computer with at least 4GB RAM
* 500MB of free disk space

**➢Software Requirement:**

* **Operating System:** Windows/Linux/MacOS
* **Compiler:** GCC or any C compiler
* **IDE**: Code: Blocks, Dev-C++, or any C IDE

**Design and Development**

**❖ Program Logic:**

The fitness tracker application is designed to help users manage and monitor their daily fitness activities, such as steps taken, calories burned, and progress towards fitness goals. It provides functionalities to add new data, view existing records in a structured table format, and update specific entries based on the date. Here’s a brief explanation of the program logic:

#### 1. Data Structures:

* **FitnessData**:
  + A structure that holds the following attributes:
    - steps: Number of steps taken.
    - calories: Calories burned.
    - progress: Progress towards fitness goals (as a percentage).
    - date: The date of the recorded data (formatted as YYYY-MM-DD).

#### 2. Functions:

* **viewData()**:
  + Purpose: Display all recorded fitness data in a table format.
  + Logic: Reads data from a text file and prints it in a structured manner using printf for better readability.
* **addData()**:
  + Purpose: Add a new fitness record to the application.
  + Logic: Prompts the user to input steps, calories, progress, and date. This data is then saved to the text file.
* **updateData()**:
  + Purpose: Update an existing fitness record for a specified date.
  + Logic: Asks the user for the date of the record they wish to update. Checks if data for that date exists. If found, allows the user to update the steps, calories, and progress. Saves the updated data back to the text file.
* **saveData(FitnessData data)**:
  + Purpose: Save new fitness data to the text file.
  + Logic: Appends the new fitness data to the text file to ensure data persistence.
* **loadData(FitnessData \*data, const char \*date)**:
  + Purpose: Load data for a specific date to facilitate updates.
  + Logic: Reads the text file to find and load data for the specified date. Returns 1 if data is found, 0 otherwise.
* **displayMenu()**:
  + Purpose: Display the main menu of the application.
  + Logic: Prints menu options for viewing, adding, and updating data, and for exiting the application.

**❖ Pseudocode:**

START

// Define Structure for FitnessData

DEFINE STRUCTURE FitnessData

INTEGER steps

INTEGER calories

FLOAT progress

STRING date (format: YYYY-MM-DD)

END STRUCTURE

// Function to display the main menu

FUNCTION displayMenu()

PRINT "1. View Data"

PRINT "2. Add Data"

PRINT "3. Update Data"

PRINT "4. Exit"

END FUNCTION

// Function to view all fitness data

FUNCTION viewData()

OPEN file "fitness\_data.txt" FOR reading

IF file IS NULL

PRINT "No data available."

RETURN

END IF

PRINT "----------------------------------------"

PRINT "| Date | Steps | Calories | Progress |"

PRINT "----------------------------------------"

WHILE NOT EOF(file)

READ data FROM file INTO steps, calories, progress, date

PRINT "|", date, "|", steps, "|", calories, "|", progress, "|"

END WHILE

CLOSE file

END FUNCTION

// Function to add new fitness data

FUNCTION addData()

DECLARE FitnessData data

PROMPT "Enter date (YYYY-MM-DD): " INPUT data.date

PROMPT "Enter steps: " INPUT data.steps

PROMPT "Enter calories: " INPUT data.calories

PROMPT "Enter progress: " INPUT data.progress

CALL saveData(data)

END FUNCTION

// Function to update existing fitness data

FUNCTION updateData()

DECLARE STRING date

DECLARE FitnessData data

PROMPT "Enter date (YYYY-MM-DD) to update: " INPUT date

IF NOT loadData(data, date)

PRINT "Data for the given date not available."

RETURN

END IF

PROMPT "Enter new steps: " INPUT data.steps

PROMPT "Enter new calories: " INPUT data.calories

PROMPT "Enter new progress: " INPUT data.progress

CALL saveData(data)

END FUNCTION

// Function to save fitness data to file

FUNCTION saveData(FitnessData data)

OPEN file "fitness\_data.txt" FOR appending

IF file IS NOT NULL

WRITE data.steps, data.calories, data.progress, data.date TO file

CLOSE file

ELSE

PRINT "Error saving data."

END IF

END FUNCTION

// Function to load fitness data for a specific date

FUNCTION loadData(OUTPUT FitnessData data, STRING date)

OPEN file "fitness\_data.txt" FOR reading

IF file IS NULL

RETURN 0

END IF

WHILE NOT EOF(file)

READ data FROM file INTO steps, calories, progress, data.date

IF data.date IS EQUAL TO date

CLOSE file

RETURN 1

END IF

END WHILE

CLOSE file

RETURN 0

END FUNCTION

// Main program logic

DECLARE INTEGER choice

DO

CALL displayMenu()

PROMPT "Enter your choice: " INPUT choice

SWITCH choice

CASE 1:

CALL viewData()

CASE 2:

CALL addData()

CASE 3:

CALL updateData()

CASE 4:

PRINT "Exiting application."

DEFAULT:

PRINT "Invalid choice. Please try again."

END SWITCH

WHILE choice IS NOT EQUAL TO 4

END

**Testing and Results**

**❖ Test cases:**

To validate the fitness tracker application, test cases include adding new fitness data to verify accurate recording and updating existing records to ensure changes reflect correctly, ensuring proper data handling and persistence across different operations.

1. **Add new fitness data:**

* **Input:** 2024-07-18, 8000 steps, 350 calories, 65.2 progress
* **Expected Output:** "Fitness data added successfully."

1. **View all fitness data:**

* **Input:** Select "View Data" option
* **Expected Output:** Display of all recorded fitness data in a table format.

1. **Update existing fitness data:**

* **Input:** 2024-07-18, 10000 steps, 400 calories, 70.5 progress
* **Expected Output:** "Fitness data updated successfully."

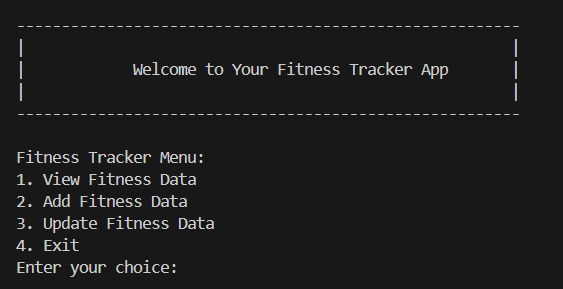
1. **View updated fitness data:**

* **Input:** Select "View Data" option
* **Expected Output:** Display of all recorded fitness data in a table format including the updated one.

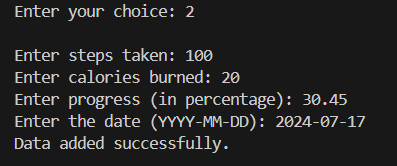
1. **Exit the application:**

* **Input:** Select "Exit" option
* **Expected Output:** "Exiting the fitness tracker application."

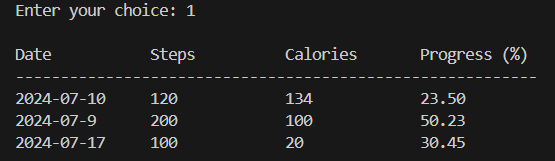
**Output:**

****

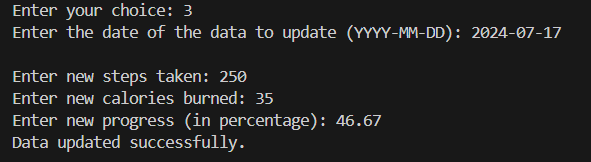
**Test case 1:**

****

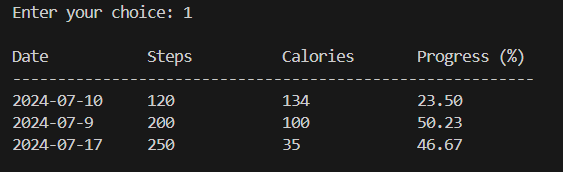
**Test case 2:**

****

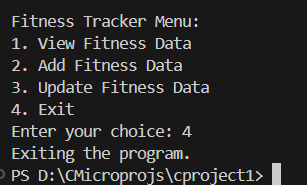
**Test case 3:**

****

**Test case 4:**

****

**Test case 5:**

****

**Discussion of results**

Managing fitness data and updates proves effective. Users can seamlessly add new fitness records, update existing data, and view comprehensive summaries, ensuring straightforward management of their fitness progress.

**Conclusion**

**➢Summary of the Project:**

The Fitness Tracker Application provides users with a streamlined platform to monitor and track their daily fitness activities, including steps taken, calories burned, and progress towards fitness goals. It facilitates easy recording of new data, updating existing records, and viewing comprehensive summaries, enabling users to maintain an organized approach to achieving their fitness objectives.

**➢Future Enhancements:**

Features that might appear in forthcoming versions of the system include integration with veterinary clinics, mobile app support, and advanced notification systems.

**References**

[Stack Overflow - Where Developers Learn, Share, & Build Careers](https://stackoverflow.com/)

[freecodecamp](https://capstoneguide.com/pet-care-management-system-capstone-project-document/)

[Phind : AI Search Engine](https://capstoneguide.com/pet-care-management-system-capstone-project-document/)

### **Appendices**

### **Source code:**

#include <stdio.h>

#include <string.h>

#define FILE\_NAME "fitness\_data.txt"

#define DATE\_LENGTH 11

typedef struct {

    int steps;

    int calories;

    float progress;

    char date[DATE\_LENGTH];

} FitnessData;

void viewData();

void addData();

void updateData();

void saveData(FitnessData data);

void displayMenu();

int loadData(FitnessData \*data, const char \*date);

int main() {

    int choice;

    printf("\n");

    printf("--------------------------------------------------------\n");

    printf("|                                                      |\n");

    printf("|            Welcome to Your Fitness Tracker App       |\n");

    printf("|                                                      |\n");

    printf("--------------------------------------------------------\n");

    printf("\n");

    do {

        displayMenu();

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                viewData();

                break;

            case 2:

                addData();

                break;

            case 3:

                updateData();

                break;

            case 4:

                printf("Exiting the program.\n");

                break;

            default:

                printf("Invalid choice. Please try again.\n");

        }

    } while (choice != 4);

    return 0;

}

void viewData() {

    FILE \*file = fopen(FILE\_NAME, "r");

    FitnessData data;

    if (file == NULL) {

        printf("No data found.\n");

        return;

    }

    printf("\n%-15s%-15s%-15s%-15s\n", "Date", "Steps", "Calories", "Progress (%)");

    printf("----------------------------------------------------------\n");

    while (fscanf(file, "%d %d %f %s", &data.steps, &data.calories, &data.progress, data.date) != EOF) {

        printf("%-15s%-15d%-15d%-15.2f\n", data.date, data.steps, data.calories, data.progress);

    }

    fclose(file);

    printf("\n");

}

void addData() {

    FitnessData data;

    printf("\nEnter steps taken: ");

    scanf("%d", &data.steps);

    printf("Enter calories burned: ");

    scanf("%d", &data.calories);

    printf("Enter progress (in percentage): ");

    scanf("%f", &data.progress);

    printf("Enter the date (YYYY-MM-DD): ");

    scanf("%s", data.date);

    saveData(data);

    printf("Data added successfully.\n\n");

}

void updateData() {

    FitnessData data;

    char date[DATE\_LENGTH];

    printf("Enter the date of the data to update (YYYY-MM-DD): ");

    scanf("%s", date);

    if (loadData(&data, date)) {

        printf("\nEnter new steps taken: ");

        scanf("%d", &data.steps);

        printf("Enter new calories burned: ");

        scanf("%d", &data.calories);

        printf("Enter new progress (in percentage): ");

        scanf("%f", &data.progress);

        strcpy(data.date, date);

        FILE \*file = fopen(FILE\_NAME, "r+");

        if (file == NULL) {

            printf("Error opening data file.\n");

            return;

        }

        FitnessData temp;

        long pos;

        while (fscanf(file, "%d %d %f %s", &temp.steps, &temp.calories, &temp.progress, temp.date) != EOF) {

            if (strcmp(temp.date, date) == 0) {

                pos = ftell(file);

                fseek(file, pos - sizeof(temp), SEEK\_SET);

                fprintf(file, "%d %d %.2f %s\n", data.steps, data.calories, data.progress, data.date);

                break;

            }

        }

        fclose(file);

        printf("Data updated successfully.\n\n");

    } else {

        printf("Data for the given date is not available.\n");

    }

}

void saveData(FitnessData data) {

    FILE \*file = fopen(FILE\_NAME, "a");

    if (file == NULL) {

        printf("Error saving data.\n");

        return;

    }

    fprintf(file, "%d %d %.2f %s\n", data.steps, data.calories, data.progress, data.date);

    fclose(file);

}

void displayMenu() {

    printf("Fitness Tracker Menu:\n");

    printf("1. View Fitness Data\n");

    printf("2. Add Fitness Data\n");

    printf("3. Update Fitness Data\n");

    printf("4. Exit\n");

}

int loadData(FitnessData \*data, const char \*date) {

    FILE \*file = fopen(FILE\_NAME, "r");

    if (file == NULL) {

        return 0;

    }

    while (fscanf(file, "%d %d %f %s", &((\*data).steps), &((\*data).calories), &((\*data).progress), (\*data).date) != EOF) {

        if (strcmp((\*data).date, date) == 0) {

            fclose(file);

            return 1;

        }

    }

    fclose(file);

    return 0;

}

### 