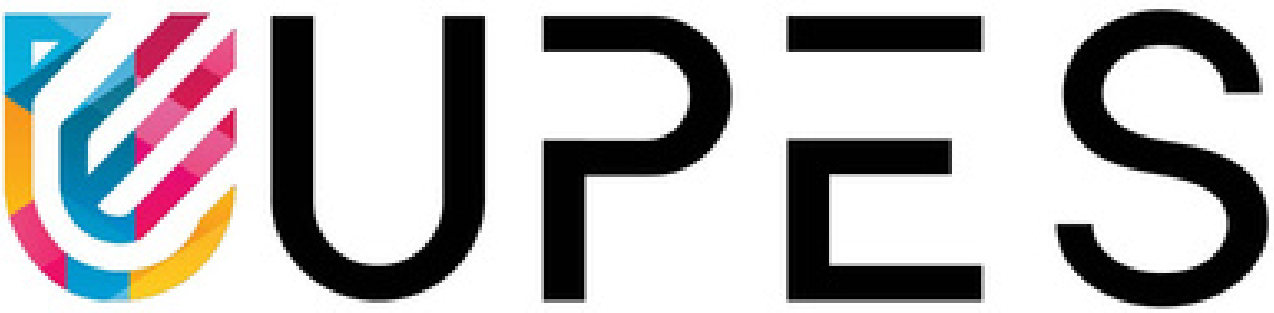


CLI HEALTH METRIC CALCULATOR

By - ARNAV MAAN

SAP ID - 590023637

ABSTRACT

The Health Metric Calculator is a Command Line Interface (CLI)–based software designed to compute essential health metrics such as Body Mass Index (BMI), Basal Metabolic Rate (BMR), Daily Calorie Requirements, and Ideal Body Weight using various standard health formulas. The project focuses on modular programming techniques using **header files**, **structures**, and **file handling** in C.

The objectives are to provide users with accurate health insights and demonstrate proper programming practices such as structured data handling, function modularity, input validation, and result logging using files. The system is efficient, user-friendly, and designed to showcase first-year programming skills effectively.

INTRODUCTION

Health awareness is increasing significantly, and individuals often want quick insights about their fitness and nutritional requirements. Metrics such as BMI, BMR, and Ideal Weight are commonly used to understand one’s body composition and calorie needs.

This project presents a command-line health metric calculator developed in C programming language using modular design principles. It accepts personal details (age, gender, height, weight, activity level) and computes multiple health indicators. The results are also saved in a text file for future reference.

This project emphasizes:

- Modular programming
- Use of structures
- Clean, efficient code
- Reusability using header files
- File I/O
- Real-world applicability

1. PROBLEM DEFINITION

1.1 OVERVIEW

Manually calculating health metrics can be time-consuming and prone to errors. Many users lack knowledge of health formulas like BMI, BMR, and calorie calculations. A simple automated system is needed to compute these values accurately.

1.2 OBJECTIVES

The main objectives of this project are:

1.To design a CLI tool for calculating important health metrics.

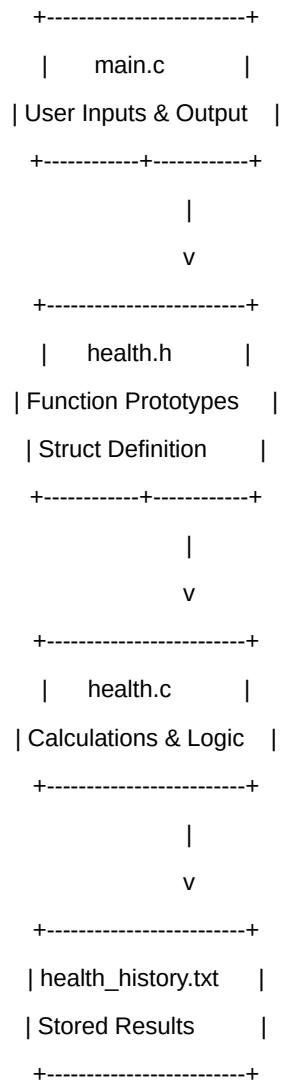
1. To implement modular programming using functions, header files, and structures.
2. To store calculated results in a file for later use.
3. To improve understanding of C programming concepts like:
    - Structures
    - File management
    - User input handling
    - Mathematical computations

4. To create an easy-to-use and efficient system suitable for personal use.

2. SYSTEM DESIGN

System design explains how the program is organized internally.

2.1 High-Level Architecture



2.2 Modules

- Input Module – collects user data
- Computation Module – BMI, BMR, Calorie, Ideal Weight
- File Module – logs results
- Output Module – displays results

3.3 Data Flow Diagram

User → Input → Processing → Output + File Storage

3. ALGORITHM

Step-by-step Algorithm

1. Start
2. Define a structure HealthProfile
3. Input: weight, height, age, gender, activity level
4. Call calculateBMI()
5. Call calculateBMR()
6. Call dailyCalories()
7. Call idealWeight()
8. Display all calculated results
9. Save results in file health_history.txt
10. End

4. **IMPORTANT CODE SNIPPETS**

- Structure definition

```c
typedef struct {
    float weight;
    float height;
    int age;
    char gender;
    int activityLevel;
} HealthProfile;
```

- BMI Calculation

```c
float calculateBMI(HealthProfile p) {
    return p.weight / (p.height * p.height);
}
```

- BMR Calculation

```c
float calculateBMR(HealthProfile p) {
    float h = p.height * 100;
    if (p.gender == 'M' || p.gender == 'm')
        return 88.362 + (13.397 * p.weight) + (4.799 * h) - (5.677 * p.age);
    else
        return 447.593 + (9.247 * p.weight) + (3.098 * h) - (4.330 * p.age);
}
```

- Ideal weight calculation

```c
float idealWeight(HealthProfile p) {
    float inch = p.height * 39.37;
    if (inch < 60) inch = 60;
    if (p.gender == 'M' || p.gender == 'm')
        return 50 + 2.3 * (inch - 60);
    return 45.5 + 2.3 * (inch - 60);
}
```

- File saving

```c
void saveToFile(float bmi, float bmr, float calories, float ideal) {
    FILE *fp = fopen("health_history.txt", "a");
    fprintf(fp, "BMI: %.2f | BMR: %.2f | Calories: %.2f | Ideal Wt: %.2f\n",
            bmi, bmr, calories, ideal);
    fclose(fp);
}
```

5. IMPLEMENTATION DETAILS

Programming Language:

C (ANSI C Standard)

Concepts Used:

- Structures
- Functions
- Header files
- File handling
- Mathematical formulas
- Modular design

File Handling:

health_history.txt stores historical results.

Each execution appends a new entry.

6. DATA STRUCTURES USED

Structure (struct)

Used to group all user input attributes:

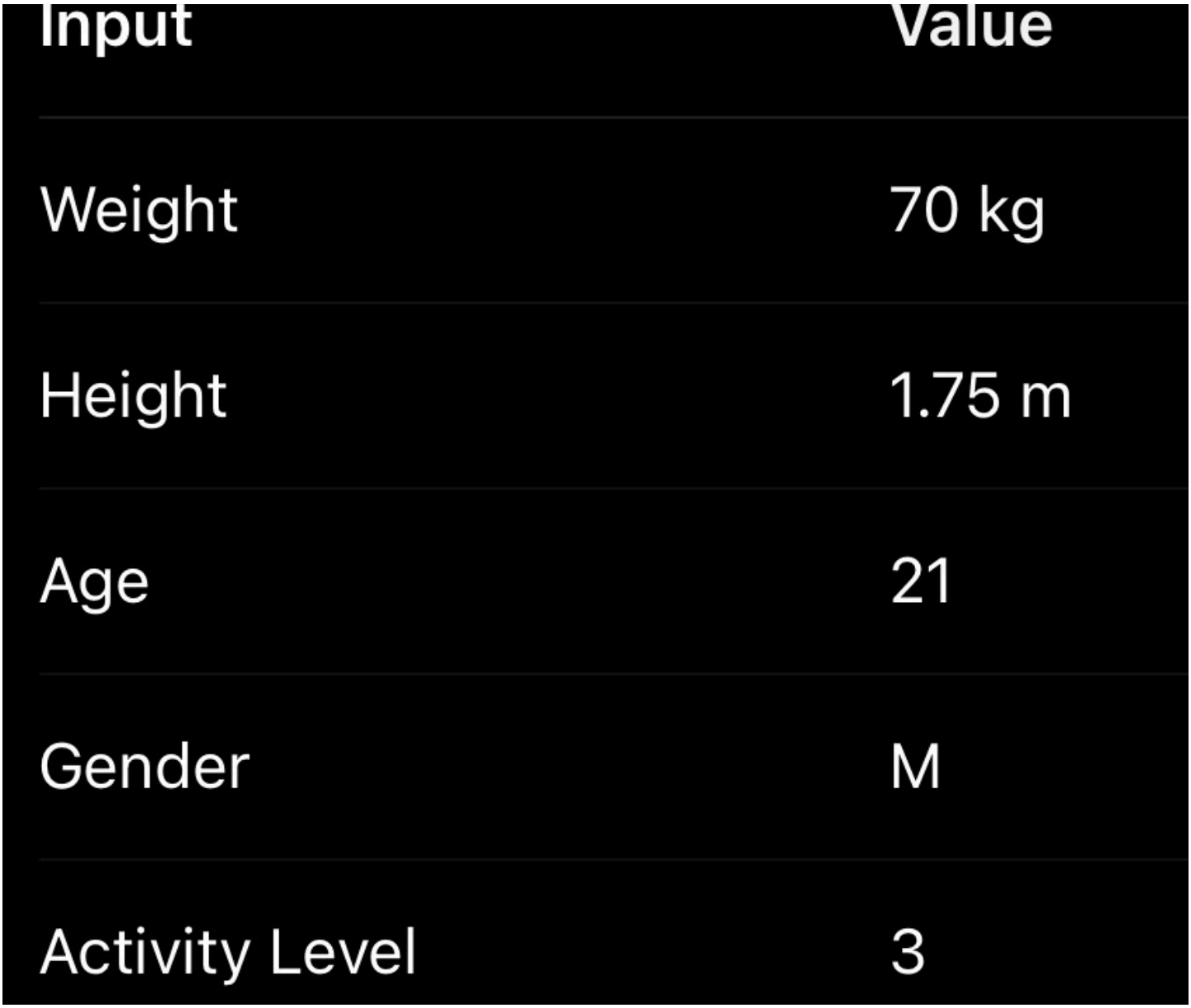
- weight
- height
- age
- gender
- activity level

Reason for using struct:

- Clean organization
- Easy to pass multiple values to functions
- Improves code readability and maintainability

7. TESTING AND RESULTS

- Test Case 1



Input	Value
Weight	70 kg
Height	1.75 m
Age	21
Gender	M
Activity Level	3

Expected Output:

- BMI ≈ 22.86
- BMR ≈ 1743.62
- Daily Calories ≈ 2702.61
- Ideal Weight ≈ 72.48 kg

Result:

Correct ✅

Output stored in file ✅

- Test Case 2

Input	Value
Weight	55 kg
Height	1.60 m
Age	19
Gender	F
Activity Level	1

Expected Output:

- BMI ≈ 21.48
- BMR ≈ 1350.36
- Daily Calories ≈ 1620.43
- Ideal Weight ≈ 52.38 kg

Result:

Correct ✅

File updated ✅

- Test Case 3 (Invalid activity level)

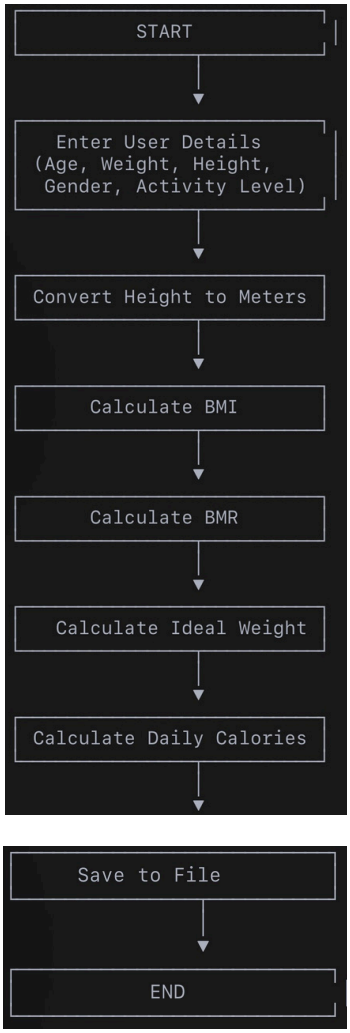
Input: Activity Level = 9

Program sets default = sedentary (1.2 multiplier)

Result:

Correct fallback ✅

FLOWCHART



9. OUTPUT SCREENSHOTS

```
===== HEALTH METRIC CALCULATOR =====
Enter weight (kg): 70
Enter height (meters): 1.75
Enter age: 21
Enter gender (M/F): M
Choose Activity Level: 3

===== RESULTS =====
BMI: 22.86
BMR: 1743.62 calories/day
Daily Calorie Needs: 2702.61 calories/day
Ideal Weight: 72.48 kg

Results saved to 'health_history.txt'
```

File output:

```
BMI: 22.86 | BMR: 1743.62 | Calories: 2702.61 | Ideal Wt: 72.48
```

CONCLUSION AND FUTURE USES:

The Health Metric Calculator is an efficient and practical C program that showcases modular programming using structures, functions, header files, and file handling. It successfully calculates major health indicators and stores them for future reference. The system is accurate, simple, and demonstrates effective use of C programming fundamentals suitable for first-year engineering or computer science coursework.

This project enhances understanding of:

- Modular software design
- Data structuring
- Mathematical processing
- Input validation

- File operations

FUTURE ENHANCEMENT -

•. Add GUI (Graphical User Interface)

• Integrate health suggestion system

• Add charts/graphs

• Connect to database instead of text file

• Support multiple users and profiles