CALORIE CALCULATOR IN C PROGRAMMING

****

Yogini. Shalini. Sudhakar. Babu

Advisor: Mr. Sahabzada Betab Badar

Department of Computer Science and Information Technology

Jain(Deemed-to-be) University

This report is submitted for the course of

Programming in C (24BCA1C04)

Declaration

I, Yogini. Shalini. Sudhakar. Babu, a student of 1st year G sec of batch 2024 from Jain (deemed-to -be) University, hereby declare that the project titled “Calorie Calculator in C Programming” submitted to the department of Computer Science and Information Technology, is an original work completed by me under the guidance of Mr. Sahabzada Betab Bada sir.

This project is solely for academic purpose as a part of the internal assessment for the subject Programming in C. I have ensure that the code and logic used in this project have been researched, analyzed and written entirely by me, except for libraries and modules that are explicitly mentioned in the project report.

I further declare that this work is the collective work of me and my team members of team 7.

Yogini. Shalini. Sudhakar. Babu

USN No: JUUG24BCAS28776

Department of Computer Science and Information Technology,

Jain (Deemed-to-be) University,

Bengaluru.

Acknowledgments

I would like to express my sincere gratitude to our Professor Mr.Sahabzada Betab Badar sir, for the valuable guidance and support and encouragement throughout the preparation of this report. This insightful feedback and constructive suggestions have been instrumental in the successful completion of this group activity.

We are also thankful to our fellow group members for their collaboration, dedication, and teamwork, which played a vital role in achieving the objectives of this assessment. Each members efforts and contributions are deeply appreciated.

Lastly, I extend my gratitude to Jain University and to the department of Computer Science and Information Technology for providing the resources and opportunities that facilitated this learning experience.

Abstract

This report details the development of a Calorie Calculator in C Programming, created as part of a collaborative group activity. The primary objective of the project was to design a user-friendly program that calculates daily caloric needs based on user inputs such as age, gender, weight, height and activity level. The project involved utilizing fundamental C programming concepts, including functions, conditi-onal statements and loops, to ensure accurate and efficient calculati-ons.

The group collaboration to divide tasks, ensuring smooth integration of components like input validation, formula, implementation and an interactive user interface. The report also highlights the challenges encountered during development, such as debugging and optimizing code for accuracy and how teamwork facilitated effective problem solving.

The final output successfully provides users with their daily caloric requirements, promoting awareness of dietary needs. This project not only enhanced the group’s technical skills in C programming but also emphasized the importance of teamwork and communication in software development.

Contents

[1.Introduction 6](#_Toc184584208)

[1.1 Objective 7](#_Toc184584209)

[1.2 Features 8](#_Toc184584210)

[1.2.1 User-Friendly Input System 8](#_Toc184584211)

[1.2.2 Basal Metabolic Rate (BMR) Calculation 8](#_Toc184584212)

[1.3.3. Daily Calorie Needs Calculation 8](#_Toc184584213)

[1.2.4.Support for Multiple Users 8](#_Toc184584214)

[1.2.5.Error Handling 8](#_Toc184584215)

[1.2.6.Detailed Results 9](#_Toc184584216)

[1.2.7. Efficient Loop Design 9](#_Toc184584217)

[1.3 Contribution 10](#_Toc184584218)

[2. Why Calorie Calculator in C 13](#_Toc184584219)

[2.2 Techniques 14](#_Toc184584220)

[3.Code Components 15](#_Toc184584221)

[3.1Header Section 16](#_Toc184584222)

[3.2 Calculate BMR 16](#_Toc184584223)

[3.3 Calculate Calories Function 17](#_Toc184584224)

[3.4 Validation Function 18](#_Toc184584225)

[3.5 Validate Activity Level Function 19](#_Toc184584226)

[3.6 Main Function 19](#_Toc184584227)

[4. Summary 22](#_Toc184584228)

[5.Data Description of the Calorie Calculator Program 23](#_Toc184584229)

[5.1 Output data 24](#_Toc184584230)

[6. Flowchart 26](#_Toc184584231)

[7.Result of the code 27](#_Toc184584232)

[8.Conclusion 33](#_Toc184584233)

[9.Glossary 35](#_Toc184584234)

[10.Referance 38](#_Toc184584235)

# 1.Introduction

A calorie calculator in C is a simple yet effective program designed to estimate the number of calories an individual needs daily based on various personal and lifestyle factors. It uses inputs such as age, gender, weight, height, and activity level to compute the Basal Metabolic Rate (BMR), which represents the number of calories the body requires to perform basic functions like breathing and maintaining body temperature while at rest. Once the BMR is calculated, it is adjusted according to the individual's physical activity level to determine the Total Daily Energy Expenditure (TDEE), which reflects the total calories needed to maintain their current weight. By providing precise calculations, the program helps users gain insights into their energy needs, making it an essential tool for those seeking to maintain, lose, or gain weight. Implementing the calculator in C involves using conditional statements to handle input variations, functions to segregate calculations, and simple mathematical formulas to process data efficiently. The program typically interacts with users through a command-line interface, ensuring simplicity and accessibility. Beyond just a utility, creating such a calculator is an excellent project for coding enthusiasts, as it combines problem-solving skills with real-world applications. Furthermore, it emphasizes user interaction, numerical computations, and logic building, making it a comprehensive learning experience for beginners. With modifications, the program can include additional features like calculating macronutrient requirements or suggesting dietary plans, enhancing its utility for fitness and health enthusiasts.do

Introduction

## Objective

The objective of a calorie calculator in C is to create a functional and user-friendly program that estimates an individual’s daily caloric needs based on scientific principles and user-specific data. This tool aims to provide an accurate calculation of the Basal Metabolic Rate (BMR) and Total Daily Energy Expenditure (TDEE) by utilizing user inputs such as age, gender, weight, height, and activity level. The purpose of the program is to assist users in understanding their daily energy requirements, which is crucial for maintaining, losing, or gaining weight. By offering a clear and precise way to evaluate caloric needs, the program empowers users to make informed decisions about their dietary and fitness goals.

The program is designed to be simple, efficient, and interactive, making it accessible for users with minimal technical knowledge. It focuses on accuracy and reliability by implementing mathematical formulas, such as the Harris-Benedict equation, to ensure scientifically grounded results. For developers, the calorie calculator provides an opportunity to apply core programming concepts like input handling, conditional logic, functions, and mathematical computations.

Furthermore, the objective extends beyond providing calorie calculations by fostering awareness about energy balance and healthy lifestyle practices. The program's structure allows for scalability, enabling the addition of advanced features such as meal planning or macronutrient distribution analysis. Ultimately, the calorie calculator in C serves as both a practical utility for health-conscious individuals and an educational project for developers to enhance their programming skills while addressing real-world problems.

## 

Introduction

## 1.2Features

## 1.2.1 User-Friendly Input System

The program prompts users step-by-step to input essential details like gender, weight, height, age, and activity level.

Validates each input and provides error messages for incorrect entries

## 1.2.2 Basal Metabolic Rate (BMR) Calculation

Accurately computes the BMR based on scientifically validated formulas:

Separate formulas for males and females.

## 1.3.3. Daily Calorie Needs Calculation

Adjusts BMR using activity multipliers based on the user’s lifestyle:

Sedentary, Lightly Active, Moderately Active, Very Active, Extra Active.

## 1.2.4.Support for Multiple Users

Allows multiple users to calculate their calorie needs in one program session.

Displays individual results for each user sequentially.

## 1.2.5.Error Handling

Includes robust input validation to prevent invalid data entry:

Ensures gender input is either ‘M’ or ‘F’.

Checks that weight, height, and age are positive numbers.

Ensures the activity level matches predefined options.

## 1.2.6.Detailed Results

Displays two key outputs for each user:

Basal Metabolic Rate (BMR).

Daily calorie needs based on their activity level.

Introduction

## 1.2.7. Efficient Loop Design

Uses loops to process multiple users, ensuring smooth execution without restarting the program for every new user.

1.2.8. Clear Exit Path

Allows users to exit the program once all calculations are complete, with a final thank-you message.

1.2.9. Scalable Design

Easily extendable to add new features, such as more activity levels or advanced calorie breakdowns.

1.2.10.Cross-Platform Compatibility

Designed in C programming language, ensuring the code can run on various platforms with minimal modifications.

These features make the program practical, easy to use, and accurate for estimating caloric requirements for diverse users.

Introduction

## 1.3 Contribution

As a member of team of the topic Calorie calculator project, my primary contribution was designing and implementing the methodology that forms the backbone of the program. The goal was to create a reliable and efficient system to estimate daily caloric needs based on user input and scientific formulas.

Researched and implemented the Harris-Benedict Equation and Mifflin-St Jeror formula for calculating Basal Metabolic Rate(BMR), which serves as the foundation for the calorie calculator. Designed the program logic to account for factors such as age, weight, height and activity levels.

Ensure the methodology aligns with standard nutritional guidenlines and best practices.

Created a robust function in C to calculate caloric requirements dynamically based on input parameters.

Implemented an different activity multiplier to adjust BMR calculations for different activity levels (eg: sedentary, moderate, active).

Debugged and optimized the program to ensure accuracy and efficiency in calculations.

I worked closely with the development team to integrate the methodology into the main program and tested it rigorously to validate results.

Introduction

One of the main challenges was ensure the program could handle edge cases, such as invalid or extreme input values. I addressed this incorporating input validation checks and error-handling mechanisms.

Through my efforts, the calorie calculator methodology was established as a reliable tool for estimating caloric needs. This experience allowed me to enhance my skills in algorithm designs, research, and practical application of scientific concepts in programming

# 2. Why Calorie Calculator in C

The justification for developing a calorie calculator in C lies in its ability to address practical needs while serving as a foundational project for learning programming concepts. With the growing emphasis on health and fitness, understanding caloric needs is crucial for individuals aiming to maintain, lose, or gain weight effectively. A calorie calculator simplifies this process by providing accurate estimations of Basal Metabolic Rate (BMR) and Total Daily Energy Expenditure (TDEE), making it an invaluable tool for personal health management. Implementing this calculator in C offers a dual benefit: users gain a helpful resource for their fitness journey, and developers gain hands-on experience with programming fundamentals.

C is a widely-used programming language known for its speed and simplicity, making it an excellent choice for building efficient and lightweight applications. Through the development of a calorie calculator, developers can practice critical skills such as handling user input, implementing conditional logic, utilizing functions for modular design, and performing mathematical computations. Moreover, the project encourages problem-solving and algorithmic thinking, as developers must design a logical flow to process user inputs and deliver accurate results.

Additionally, building a calorie calculator promotes awareness about energy balance and the science behind caloric needs, encouraging users to adopt healthier habits. Its scalable design also allows for future enhancements, such as integrating features for dietary planning or tracking macronutrient consumption. Overall, the calorie calculator justifies its development by addressing a relevant real-world need while providing a robust learning experience for aspiring programmers.

# Why Calorie Calculator in C

## 2.2 Techniques

**Input Handling**: The program collects user inputs such as age, gender, weight, height, and activity level. This is typically achieved using standard input/output functions like scanf() and printf() to ensure interactive communication with the user.

**Conditional Logic**: Decision-making structures like if-else or switch statements are used to handle different user-specific cases, such as gender-specific BMR calculations and varying activity levels.

**Functions for Modularity**: Functions are employed to separate logic into manageable units. For instance, one function could calculate the BMR, while another adjusts it based on activity levels to compute the Total Daily Energy Expenditure (TDEE). This improves code readability and reusability.

**Mathematical Operations**: The program incorporates mathematical formulas, such as the Harris-Benedict equation, to compute caloric needs accurately. Operators like +, -, \*, and / are utilized for these calculations.

**Error Handling**: Input validation techniques ensure that user inputs are within realistic ranges (e.g., positive numbers for weight or height), enhancing program robustness.

**Loop Structures**: Loops like while or for may be used to allow users to perform multiple calculations without restarting the program .

# 3.Code Components

The code for a calorie calculator in C consists of several essential components working together to achieve accurate results. At its core, the program begins with a main() function, which serves as the entry point for execution and coordinates the overall logic. It includes input handling, where the program uses scanf() to gather user-provided details such as age, gender, weight, height, and activity level. These inputs are stored in appropriately typed variables like int and float to facilitate calculations.

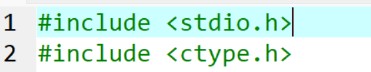
The program utilizes functions to modularize the code. For example, one function might calculate the Basal Metabolic Rate (BMR) based on gender and physical attributes, while another function adjusts the BMR using activity multipliers to compute the Total Daily Energy Expenditure (TDEE). These functions are called from the main() function, ensuring clarity and reusability.

Conditional statements such as if-else or switch are employed to implement gender-specific BMR formulas and activity level adjustments, ensuring the calculations align with user data. Error-handling mechanisms validate inputs, preventing invalid data like negative values for weight or height.

The program may also include loops, like a while loop, to allow users to perform multiple calculations without restarting the application. Finally, output handling through printf() is used to display results clearly, providing users with insights into their daily caloric needs. These components together form a robust and efficient structure for implementing the calorie calculator in C.

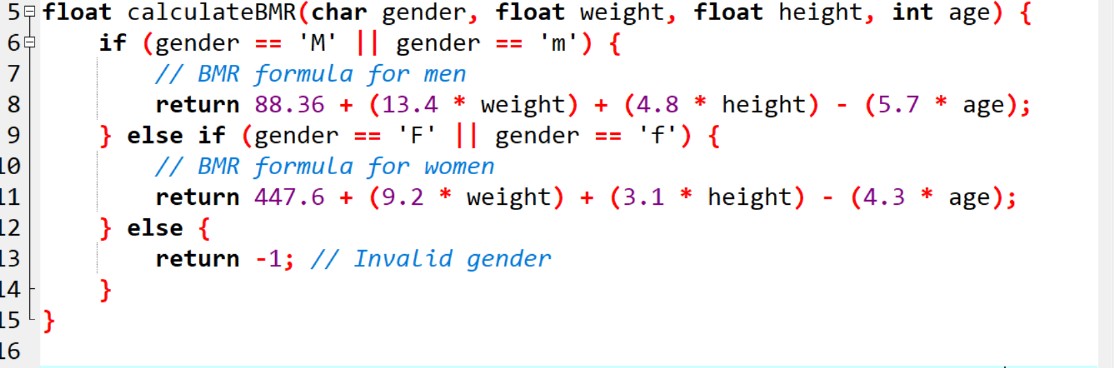
Code Components

## 3.1Header Section



* **#include <stdio.h>:** This includes the standard input/output library in C, which provides functions like printf and scanf for printing output and taking input, respectively.
* **#include <ctype.h>**: This includes the character handling library, which provides functions like toupper() to convert characters to uppercase.

## 3.2 Calculate BMR



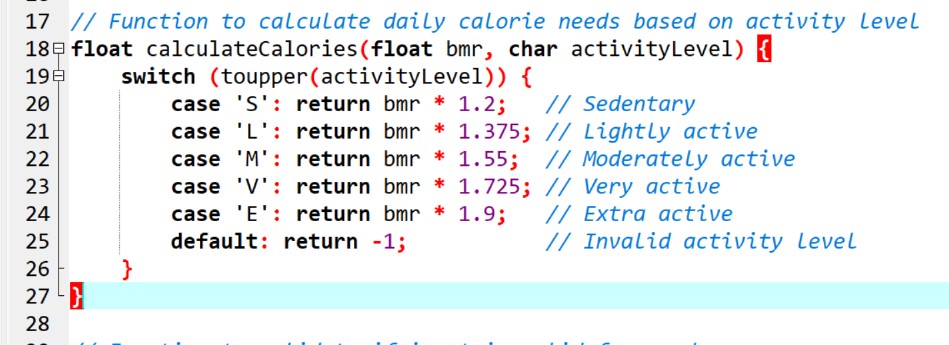
# 

* **float calculateBMR(char gender, float weight, float height, int age)**: This function calculates the Basal Metabolic Rate (BMR), which is the number of calories a person needs to maintain basic bodily functions like breathing and digestion.
* It takes four parameters: gender, weight, height, and age.
* **if (gender == 'M' || gender == 'm')**: If the gender is male (either uppercase 'M' or lowercase 'm'), it calculates the BMR using the male-specific formula.

Code Components

* **else if (gender == 'F' || gender == 'f')**: If the gender is female (either uppercase 'F' or lowercase 'f'), it uses the female-specific formula for BMR.
* **else { return -1; }**: If the gender is neither 'M' nor 'F', it returns -1 to indicate an invalid input.
* The formulas used for BMR are based on the Harris-Benedict equation, adjusted for each gender.

## 3.3 Calculate Calories Function

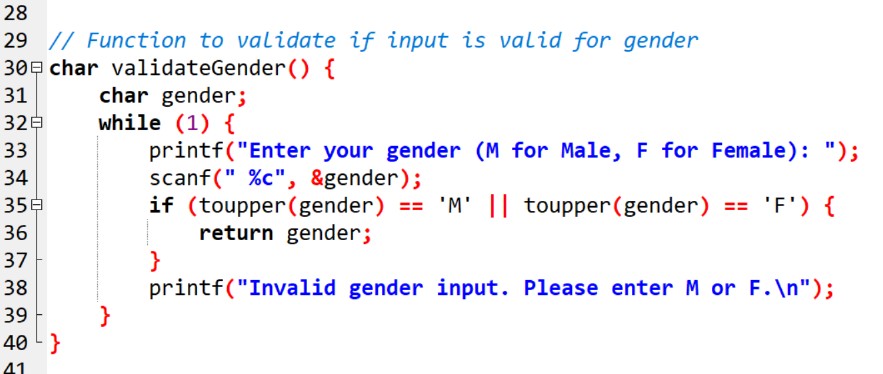


* **float calculateCalories(float bmr, char activityLevel)**: This function calculates the daily calorie needs based on the BMR and the activity level.
* **switch (toupper(activityLevel))**: Converts the activity level character to uppercase for consistent comparison.
* It uses a switch statement to match the user's activity level ('S', 'L', 'M', 'V', 'E'), and calculates the calorie needs by multiplying the BMR by an activity multiplier.
* **case 'S'**: Sedentary activity level (little or no exercise), BMR \* 1.2.
* **case 'L'**: Lightly active (light exercise 1-3 days/week), BMR \* 1.375.

Code Components

* **case 'M'**: Moderately active (moderate exercise 3-5 days/week), BMR \* 1.55.
* **case 'V'**: Very active (hard exercise 6-7 days/week), BMR \* 1.725.
* **case 'E'**: Extra active (very hard exercise or physical job), BMR \* 1.9.
* If the activity level is invalid, it returns -1.

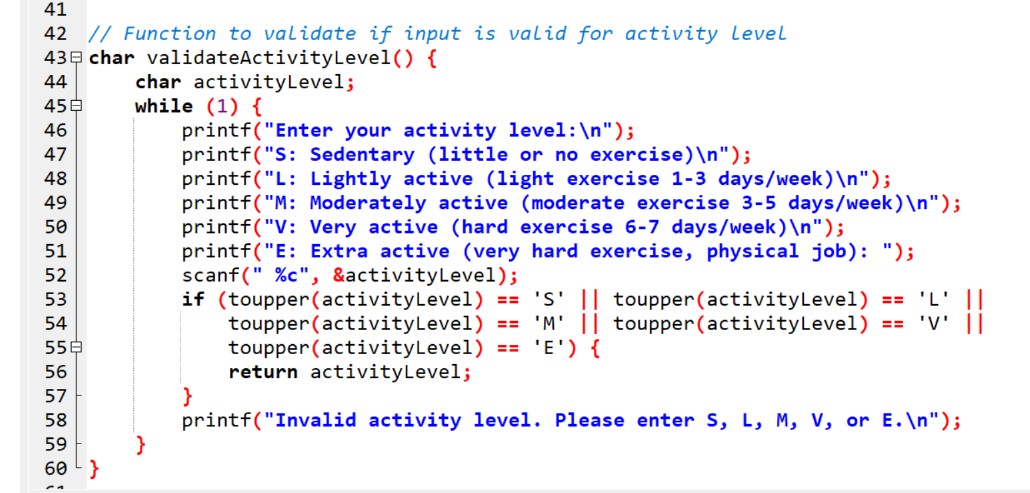
## 3.4 Validation Function



* **char validateGender()**: This function prompts the user to input their gender and ensures it is valid ('M' or 'F').
* It uses a while (1) loop to repeatedly ask the user for input until a valid gender is entered.
* **scanf(" %c", &gender)**: Takes a single character as input for gender. The space before %c ensures that any leftover whitespace characters are ignored.
* **if (toupper(gender) == 'M' || toupper(gender) == 'F')**: Checks if the input is either 'M' or 'F', regardless of case.
* If the gender is valid, it returns the entered gender.
* If the input is invalid, it displays an error message and asks for the gender again.

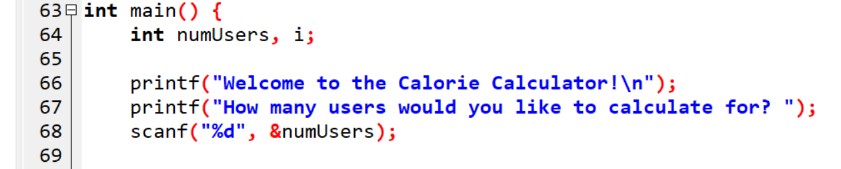
Code Components

3.5 Validate Activity Level Function



* **char validateActivityLevel()**: This function prompts the user to input their activity level and ensures the input is valid.
* The function provides a list of valid activity levels (S, L, M, V, E) and keeps prompting the user until a valid level is entered.
* **scanf(" %c", &activityLevel)**: Reads a single character input for activity level.
* The function then checks if the input is one of the valid characters, and if valid, returns the activity level. If invalid, it prompts the user again.

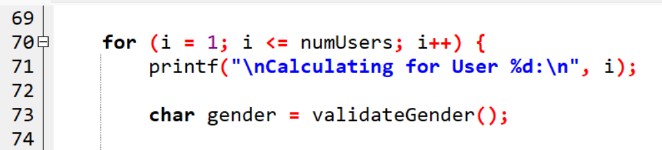
## 3.6 Main Function



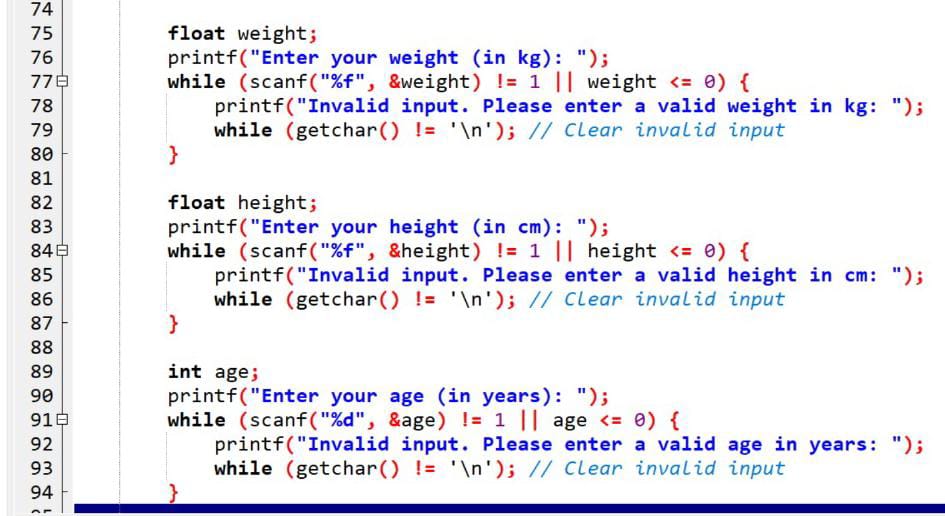
* **int main()**: The entry point of the program.
* **int numUsers, i;**: Declares two variables: numUsers for how many users' calorie calculations are needed, and i for looping through each user.

Code Components

* **scanf("%d", &numUsers)**: Reads an integer input for the number of users.

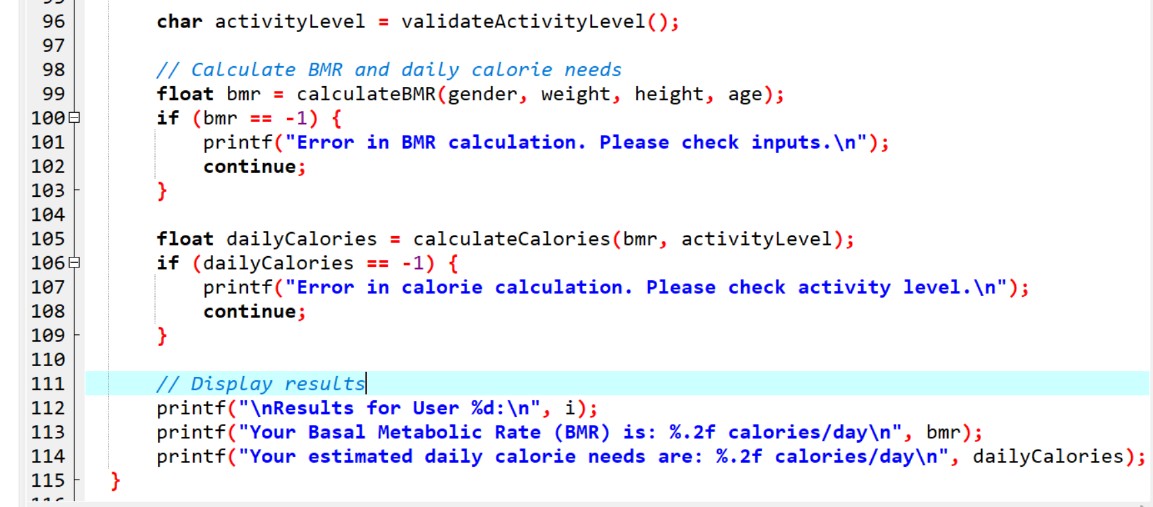


* **for (i = 1; i <= numUsers; i++)**: Loops through the number of users specified by numUsers.
* **char gender = validateGender();**: Calls the validateGender() function to get a valid gender input from the user.

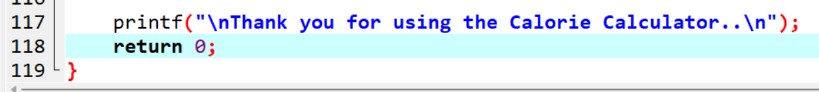


* **float weight;**: Declares a variable to store the user's weight.
* The while loop ensures that the weight input is valid (a positive number) and prompts for re-entry if it’s not.
* Similarly, this part validates the user’s height input.
* Similarly, this part validates the user’s age input.

Code Components



* Calls validate activity level() to get a valid activity level input from the user.
* Calls the calculateBMR() function to compute the user’s BMR.
* If the BMR is invalid (-1), an error message is displayed, and the program continues to the next user.
* Calls calculateCalories() to compute the user’s daily calorie needs.
* If the calorie calculation returns an error (-1), an error message is displayed.
* Displays the BMR and daily calorie needs for the user.



* After calculating for all users, the program ends with a thank-you message and returns 0 to indicate successful execution.

## 4. Summary

The Calorie Calculator program in C calculates a person’s Basal Metabolic Rate (BMR) and daily calorie needs based on factors such as gender, weight, height, age, and activity level. The program begins by prompting the user to input their gender, weight, height, and age. It validates the gender input to ensure that it is either "M" for male or "F" for female, and checks that the weight, height, and age are positive numbers. The program then asks for the user’s activity level, which can be one of five options: Sedentary, Lightly active, Moderately active, Very active, or Extra active. This input is validated to ensure correctness. The BMR is then calculated using gender-specific formulas. For men, the formula is 88.36 + (13.4 \* weight) + (4.8 \* height) - (5.7 \* age), while for women, it is 447.6 + (9.2 \* weight) + (3.1 \* height) - (4.3 \* age). The daily calorie requirement is then determined by multiplying the BMR by an activity factor based on the chosen activity level. The program repeats this process for multiple users, allowing the user to enter data for several individuals in one session. If any invalid input is detected during data entry, the program prompts the user to correct it. At the end of each user’s data entry, the program outputs the calculated BMR and daily calorie needs. The program concludes with a thank-you message, completing the calorie calculation process for all users.

# 5.Data Description of the Calorie Calculator Program

The Calorie Calculator program processes several types of input data to calculate the Basal Metabolic Rate (BMR) and daily caloric needs

for users. This section outlines the types of data used, their purpose, and the calculations performed.

1. Input Data

The program collects the following inputs for each user:

Parameter Data Type Description Validation

Gender char Specifies the user’s gender as either M (male) or F (female).Must be M or F. Case insensitive.

Weight floatThe user’s weight in kilograms.Must be a positive number.

Height float The user’s height in centimeters. Must be a positive number.

Age int The user’s age in years. Must be a positive integer.

Activity Level char Indicates the user’s level of physical activity (S, L, M, V, E). Must match predefined activity categories.

2. Intermediate Data

The program uses the collected inputs to calculate intermediate values:

Parameter Data Type Description

Basal Metabolic Rate (BMR) float The amount of energy (calories) needed to sustain the body’s basic functions at rest, calculated using the gender-based formula.

Gender-Based BMR Formulas:

* Male
* Female

Data Description of the Calorie Calculator Program

## 5.1 Output data

The program generates the following outputs for each user:

Output Data Type Description

BMR float The calories required by the user’s body at rest, based on weight, height, age, and gender.

Daily Caloric Needs float The user’s total daily caloric requirement, calculated by multiplying the BMR by an activity factor.

Activity Level Multipliers:

Activity Level Multiplier

Sedentary (S)-1.2

Lightly Active (L) -1.375

Moderately Active (M)-1.55

Very Active (V)-1.725

Extra Active (E)-1.9

Error Handling

The program handles invalid data inputs and generates error messages to prompt the user to enter correct values:

•Invalid gender input (X): Displays “Invalid gender input. Please enter M or F.”

•Invalid activity level input: Displays “Invalid activity level. Please enter S, L, M, V, or E.”

•Invalid numeric inputs (e.g., negative or zero values): Displays “Invalid input. Please enter a valid value.”

Sample Data and Results

Data Description of the Calorie Calculator Program

Input Output

Gender: M BMR: 1676.60 calories/day

Weight: 70 kg Daily Calorie Needs: 2599.73

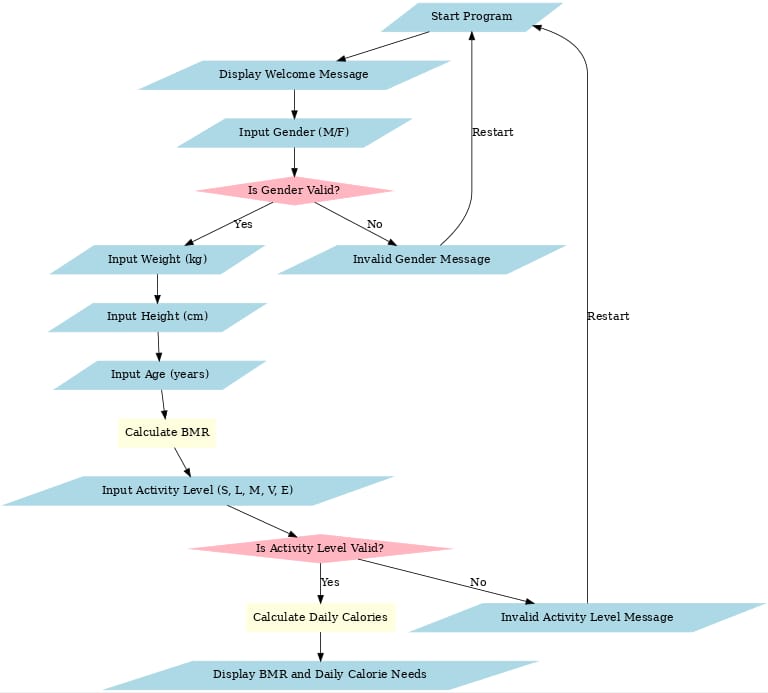
Height: 175 cm

Age: 25 years

Activity Level: M

This structured data flow ensures the program produces accurate, actionable results for the user.

# 6. Flowchart



7.Result of the code

The calorie calculator program is designed to calculate and display important health metrics based on the user’s personal data. It calculates the Basal Metabolic Rate (BMR) and Daily Caloric Needs based on gender, weight, height, age, and activity level. Here’s a more detailed explanation of the different results and how they are displayed:

1. Basal Metabolic Rate (BMR)

Definition:

BMR represents the number of calories your body needs to perform essential functions (like breathing, digestion, and circulation) while at rest. It is the baseline caloric requirement before any physical activity is taken into account.

How It’s Calculated:

The BMR formula differs for men and women. It factors in gender, weight, height, and age:

For men:

BMR = 88.36 + (13.4 \* weight) + (4.8 \* height) - (5.7 \* age)

For women:

BMR = 447.6 + (9.2 \* weight) + (3.1 \* height) - (4.3 \* age)

Example:

•Inputs: Male, 70 kg, 175 cm, 25 years

•Output:

BMR = 1676.60 calories/day

This means that a male of 70 kg weight, 175 cm height, and 25 years old will need 1676.60 calories per day just for basic bodily functions, with no activity.

Result of the code

2. Daily Caloric Needs

Definition:

After calculating the BMR, the program adjusts this value based on the user’s activity level. This is called the Daily Caloric Needs, which represents how many calories the user should consume to maintain their current weight, considering physical activity.

•Activity Levels and Multipliers:

The program uses the following multipliers based on the user’s activity level:

•Sedentary (little or no exercise): Multiply BMR by 1.2

•Lightly Active (light exercise 1-3 days/week): Multiply BMR by 1.375

•Moderately Active (moderate exercise 3-5 days/week): Multiply BMR by 1.55

•Very Active (hard exercise 6-7 days/week): Multiply BMR by 1.725

•Extra Active (very hard exercise or physical job): Multiply BMR by 1.9

Example:

•Inputs: Activity Level: Moderately Active

•Output:

Daily Calorie Needs = 2599.73 calories/day

This means that, with a moderately active lifestyle, the user needs 2599.73 calories per day to maintain their current weight.

3. Error Messages for Invalid Inputs

Definition:

The program checks whether the inputs entered by the user are valid. If an invalid input is entered (such as an incorrect gender or activity

Result of the code

level), the program will display an error message and ask the user to provide the correct input.

•Examples of Invalid Inputs:

•Invalid Gender Input: If the user enters anything other than ‘M’ (Male) or ‘F’ (Female), the program will display an error.

•Example Input: Gender = ‘X’

•Output:

"Invalid gender input. Please restart the program."

•Invalid Activity Level Input: If the user enters an activity level outside the predefined options (‘S’, ‘L’, ‘M’, ‘V’, or ‘E’), the program will prompt them to provide a valid input.

Purpose:

This validation ensures that only the correct and expected data is used for calculation, making the results accurate and preventing the program from breaking due to unexpected inputs.

4. Multiple User Support

Definition:

The program is designed to handle multiple users in a single session. It can calculate the calorie needs for several people in a row, one at a time.

How It Works:

The program asks the user how many people they would like to calculate for.

Result of the code

It then loops through the calculation process for each person, collecting and validating their input, calculating their BMR and daily calorie needs, and displaying the results.

Example:

Case 1 (User 1):

•Inputs: Gender = Male, Weight = 70 kg, Height = 175 cm, Age = 25 years, Activity Level = Moderately Active

•Output:

BMR = 1676.60 calories/day

Daily Caloric Needs = 2599.73 calories/day

•Case 2 (User 2):

•Inputs: Gender = Female, Weight = 60 kg, Height = 160 cm, Age = 30 years, Activity Level = Lightly Active

•Output:

BMR = 1365.60 calories/day

Daily Caloric Needs = 1875.90 calories/day

•The program will continue to ask for user inputs until all requested users have had their results displayed.

Example Output:

Case 1: Valid Input

•Inputs:

•Gender: Male

•Weight: 70 kg

•Height: 175 cm

Result of the code

•Age: 25 years

•Activity Level: Moderately Active

•Output:

•Your Basal Metabolic Rate (BMR) is: 1676.60 calories/day

•Your estimated daily calorie needs are: 2599.73 calories/day

This means that the male user with these inputs needs 1676.60 calories per day at rest, and 2599.73 calories per day when considering their activity level.

Case 2: Invalid Input

•Inputs:

•Gender: X (Invalid input)

•Output:

•"Invalid gender input. Please restart the program."

This error message is displayed when the user enters something other than ‘M’ or ‘F’ for the gender input.

# 8.Conclusion

The development of the Calorie Calculator in C was a productive group activity that provided valuable insights into both programming and teamwork. This project aimed to create a user-friendly application

that calculates the calories of food items based on user inputs. The following points summarize our experience and outcomes:

Technical Accomplishments:

Successfully implemented core functionalities using C programming concepts such as loops, arrays, and functions.

Ensured accurate calorie calculations by integrating predefined data and user inputs.

Collaborative Efforts:

Task distribution and regular discussions allowed us to leverage each member's strengths effectively.

Communication and coordination were key in resolving challenges and ensuring timely completion.

Learning Outcomes:

Strengthened our understanding of programming logic, debugging, and testing.

Gained practical experience in writing modular and reusable code.

Learned the importance of documentation for better code readability and maintenance.

Challenges:

Handling edge cases and ensuring input validation was a critical part of the project.

Conclusion

Integrating contributions from multiple team members required careful version control and coordination.

Overall, the Calorie Calculator project was a successful endeavor that combined technical learning with practical teamwork skills. It highlighted the importance of collaboration in software development and the relevance of programming in solving real-world problems. This experience will serve as a foundation for future group projects and coding challenges.

# 9.Glossary

Glossary for Calorie Calculator

1.Basal Metabolic Rate (BMR):

The number of calories the body needs to perform basic functions such as breathing, circulation, and maintaining body temperature while at rest.

2.Activity Level:

A measure of how active a person is on a daily basis, used to adjust the BMR to calculate daily caloric needs. Examples include sedentary, lightly active, moderately active, very active, and extra active.

3.Calories:

A unit of energy that measures the amount of energy provided by food and drinks. It is also the energy the body uses for physical activity and basic bodily functions.

4.Gender:

Used to determine which formula to apply for BMR calculation, as males and females have different metabolic rates.

5.Weight:

The mass of a person, typically measured in kilograms, which is a key input in calculating the BMR.

6.Height:

The measure of a person’s stature from head to toe, typically measured in centimeters, and used in BMR calculation.

7.Age:

The number of years a person has lived, which affects their metabolic rate and is used in BMR calculations.

8.Sedentary:

Glossary

A category of activity level for individuals who do little to no physical activity during the day. It corresponds to a multiplier of 1.2 in the calorie calculation.

9.Lightly Active:

A category for individuals who engage in light exercise 1–3 days per week. It corresponds to a multiplier of 1.375 in the calorie calculation.

10.Moderately Active:

A category for individuals who engage in moderate exercise 3–5 days per week. It corresponds to a multiplier of 1.55 in the calorie calculation.

11.Very Active:

A category for individuals who engage in hard exercise 6–7 days per week. It corresponds to a multiplier of 1.725 in the calorie calculation.

12.Extra Active:

A category for individuals with very hard physical activity or a physically demanding job. It corresponds to a multiplier of 1.9 in the calorie calculation.

13.Error Handling:

The process of identifying and managing invalid inputs (e.g., incorrect gender or activity level) to ensure the program runs smoothly.

14.Input Validation:

A feature of the program that ensures all user-provided data (e.g., weight, height, gender) meets the required format and criteria.

15.Daily Caloric Needs:

Glossary

The total number of calories a person requires in a day, calculated by multiplying the BMR by an activity level multiplier.

This glossary helps clarify the key terms and concepts used in the calorie calculator program.

# 10.Referance

The following resources and tools were used to develop the Calorie Calculator Program and related report:

1.Programming Concepts:

•Basic principles of C programming: Input handling, conditional statements, loops, and functions.

•Reference: Kernighan, B. W., & Ritchie, D. M. (1988). The C Programming Language. Prentice Hall.

2.Calorie Calculation Formulas:

•BMR formulas for males and females:

•Harris-Benedict Equation (Revised):

•Mifflin, M. D., & St Jeor, S. T. (1990). “A new predictive equation for resting energy expenditure in healthy individuals.” The American Journal of Clinical Nutrition.

3.Activity Level Multipliers:

•Derived from general fitness and dietary research based on daily energy expenditure guidelines.

•Source: Mayo Clinic Staff (2023). “Calories: How to Know if You Go Overboard.”

4.C Programming Resources:

•Input validation techniques and error handling practices:

•GeeksforGeeks. “Input Validation in C Programming.”

•TutorialsPoint. “C Programming Basics and Functions.”

5.Documentation Support:

•Structuring reports and presentations:

•Purdue Online Writing Lab (OWL). “Technical Writing Guidelines.”

Referance

•Canva. “Effective PowerPoint Presentation Techniques.”

6.Team Collaboration:

•Contributions from team members in the areas of coding, presentation creation, and report drafting.

•Group brainstorming sessions and code reviews to ensure program accuracy and clarity.

These references were critical in ensuring the program’s accuracy and the clarity of the report and presentation.