NAME. : SUSHMITHA

ROLLNO: 25A31A0123.

BRANCH: CIVIL

1-3 / 11

Student Health Tracker

A C Programming Mini-Project

Project Overview & Purpose

The Student Health Tracker is a simple yet powerful C programming mini-project designed to calculate key health metrics. This tool helps students understand and monitor their basic health parameters.

Hands-On Learning

Excellent for learning C language fundamentals: functions, conditional logic, and arithmetic operations.

Practical Application

Provides immediate health insights, such as BMI and corresponding health categories.

Modular Design

Showcases good programming practices through a structured, modular approach.

Essential Features Included

This project covers a range of health calculations, providing a comprehensive report for each student.

1

Take Student Details

Collects name, age, height, and weight for personalized analysis.

2

Height Conversion

Automatically converts height from feet & inches to meters for accurate calculations.

3

Calculate BMI

Computes Body Mass Index using standard formulas.

4

Display BMI Category

Categorizes health status as Underweight, Normal, Overweight, or Obese.

5

Generate Health Report

Presents a full, easy-to-understand health summary.

Hardware & Software Requirements

To successfully develop and run the Student Health Tracker, ensure you have the following setup:

Hardware

- Computer / Laptop
- Minimum 2 GB RAM
- Processor: Intel i3 / AMD equivalent
- Storage: 50 MB free space
- Keyboard for input

Software

- Operating System: Windows / Linux
- C Compiler (e.g., GCC, Turbo C, CodeBlocks, Dev-C++, VS Code with C extension)
- Online compilers (e.g., Programiz, Replit)
- Text Editor / IDE for writing code

Core Programming Concepts

This project reinforces fundamental C programming concepts, crucial for beginners.

Variables

Storing different types of data.

Functions

Modularizing code for reusability.

Conditional Statements

Controlling program flow based on conditions (if/else).

Arithmetic Operators

Performing mathematical calculations.

Input/Output Operations

Interacting with the user (scanf, printf).

C Standard Library

Utilizing essential header files like `<stdio.h>` and `<math.h>`.

```
Algorithm: Student Health Tracker System with Health Calculations

1. START

2. Define Structure for Student Health Record
   Structure StudentHealth:
       - roll_number (integer)
       - name (string)
       - age (integer)
       - gender (char: M/F)
       - height (float, in cm)
       - weight (float, in kg)
       - systolic_bp (integer)
       - diastolic_bp (integer)
       - heart_rate (integer)
       - bmi (float)
       - bmi_category (string)
       - health_status (string)

3. Main Function
   REPEAT
       Display Menu:
           1. Add Student Health Record
           2. Display All Records
           3. Search Student by Roll Number
           4. Calculate and Update Health Metrics
           5. Display Health Statistics
           6. Check Health Alerts
           7. Exit

       Input choice

       SWITCH (choice)
           CASE 1: Call AddStudentRecord()
           CASE 2: Call DisplayAllRecords()
           CASE 3: Call SearchStudent()
           CASE 4: Call UpdateHealthMetrics()
           CASE 5: Call DisplayStatistics()
           CASE 6: Call CheckHealthAlerts()
           CASE 7: EXIT
           DEFAULT: Display "Invalid Choice"
   UNTIL choice == 7

4. Function: AddStudentRecord()
   Input roll_number

   // Check if student already exists
   IF student exists THEN
       Display "Student already registered"
       RETURN
```

```
5. Function: CalculateBMI(weight, height)
   // BMI Formula: weight(kg) / (height(m))^2
   height_in_meters = height / 100.0
   bmi = weight / (height_in_meters * height_in_meters)
   RETURN bmi

6. Function: ClassifyBMI(bmi)
   IF bmi < 18.5 THEN
       RETURN "Underweight"
   ELSE IF bmi >= 18.5 AND bmi < 25.0 THEN
       RETURN "Normal weight"
   ELSE IF bmi >= 25.0 AND bmi < 30.0 THEN
       RETURN "Overweight"
   ELSE
       RETURN "Obese"

7. Function: CalculateBMR(weight, height, age, gender)
   // Basal Metabolic Rate (Harris-Benedict Equation)
   IF gender == 'M' THEN
       bmr = 88.362 + (13.397 * weight) + (4.799 * height) - (5.677 * age)
   ELSE
       bmr = 447.593 + (9.247 * weight) + (3.098 * height) - (4.330 * age)
   RETURN bmr

8. Function: ClassifyBloodPressure(systolic, diastolic)
   IF systolic < 120 AND diastolic < 80 THEN
       RETURN "Normal"
   ELSE IF systolic >= 120 AND systolic < 130 AND diastolic < 80 THEN
       RETURN "Elevated"
   ELSE IF systolic >= 130 AND systolic < 140 OR diastolic >= 80 AND diastolic <
90 THEN
       RETURN "Hypertension Stage 1"
   ELSE IF systolic >= 140 OR diastolic >= 90 THEN
       RETURN "Hypertension Stage 2"
   ELSE
       RETURN "Hypertensive Crisis"

9. Function: ClassifyHeartRate(heart_rate, age)
   // Normal resting heart rate: 60-100 bpm for adults
   IF age >= 18 THEN
       IF heart_rate < 60 THEN
           RETURN "Low (Bradycardia)"
       ELSE IF heart_rate >= 60 AND heart_rate <= 100 THEN
           RETURN "Normal"
       ELSE
           RETURN "High (Tachycardia)"
   ELSE
       // For children/teens, higher rates are normal
       IF heart_rate < 70 THEN
           RETURN "Low"
```

```
11. Function: DetermineHealthStatus(bmi, systolic_bp, diastolic_bp, heart_rate,
age)
    health_score = 0

    // BMI check
    IF bmi >= 18.5 AND bmi < 25.0 THEN
        health_score = health_score + 25
    ELSE IF bmi >= 25.0 AND bmi < 30.0 THEN
        health_score = health_score + 15
    ELSE
        health_score = health_score + 5

    // Blood pressure check
    IF systolic_bp < 120 AND diastolic_bp < 80 THEN
        health_score = health_score + 25
    ELSE IF systolic_bp < 140 AND diastolic_bp < 90 THEN
        health_score = health_score + 15
    ELSE
        health_score = health_score + 5

    // Heart rate check
    heart_rate_status = ClassifyHeartRate(heart_rate, age)
    IF heart_rate_status == "Normal" THEN
        health_score = health_score + 25
    ELSE
        health_score = health_score + 10

    // Determine overall status
    IF health_score >= 70 THEN
        RETURN "Excellent Health"
    ELSE IF health_score >= 50 THEN
        RETURN "Good Health"
    ELSE IF health_score >= 30 THEN
        RETURN "Fair Health - Needs Attention"
    ELSE
        RETURN "Poor Health - Immediate Attention Required"

12. Function: SearchStudent()
    Input roll_number

    Search in records array/file

    IF found THEN
        Display complete health record:
            - Personal details
            - BMI and category
            - Blood pressure status
            - Heart rate status
            - BMR (calculate)
            - Ideal weight (calculate)
            - Overall health status
```

14. Function: DisplayStatistics()
    Calculate and display:
        - Total number of students
        - Average BMI
        - Percentage in each BMI category
        - Students with normal BP vs high BP
        - Average heart rate
        - Overall health distribution

15. Function: CheckHealthAlerts()
    FOR each student record DO
        alert_count = 0

        IF bmi < 18.5 OR bmi >= 30.0 THEN
            Display "Alert: Roll " + roll_number + " - BMI out of healthy range"
            alert_count++

        IF systolic_bp >= 140 OR diastolic_bp >= 90 THEN
            Display "Alert: Roll " + roll_number + " - High blood pressure"
            alert_count++

        IF heart_rate < 60 OR heart_rate > 100 THEN
            Display "Alert: Roll " + roll_number + " - Abnormal heart rate"
            alert_count++
    END FOR

    IF alert_count == 0 THEN
        Display "All students have normal health parameters"

16. Function: UpdateHealthMetrics()
    Input roll_number

    IF student not found THEN
        Display "Student not found"
        RETURN

    Display current metrics

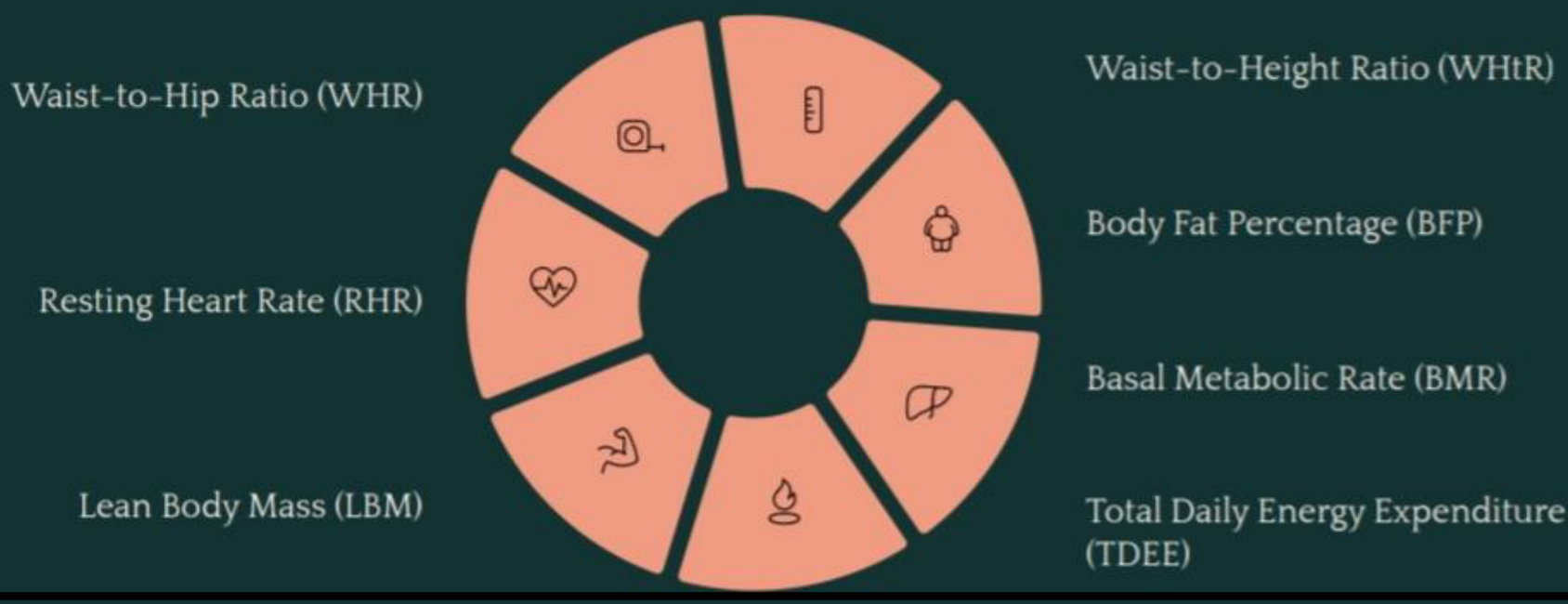
    Input new values (height, weight, BP, heart rate)

    Recalculate:
        - BMI
        - BMI category
        - BP status
        - Heart rate status
        - Overall health status

    Update record
    Display updated information

Advanced Health Metrics Calculation

Beyond BMI, the tracker computes several other critical health indicators, offering a more detailed personal health assessment.



Modular Design: A Structured Approach

The project's architecture is divided into distinct modules, promoting clarity, maintainability, and testability.

Input Module

Handles reading and validating all user inputs for health parameters.

Calculation Module

Contains sub-modules for each specific health metric calculation.

Utilities Module

Provides helper functions, such as activity factor mapping.

Output Module

Formats and displays the final health report and interpretations.

Main / Integration Module

Orchestrates the program flow, calling other modules in sequence.

Project Flow: From Input to Insight

The program follows a clear, logical sequence to process data and generate health reports.

01

Start Program

02

Read Inputs

Collect user data through the Input Module.

03

Validate Inputs

Ensure data is within acceptable numeric ranges.

04

Compute Metrics

Utilize the Calculation Module for all health metrics.

05

Display Results

Present a comprehensive report using the Output Module.

06

End Program

Sample Output: A Health Report in Action

Observe a typical interaction and the detailed health report generated by the program.

Input Example

Gender: 1 (Male)
Age: 20
Height: 170 (cm)
Weight: 65 (kg)
Waist: 80 (cm)
Hip: 90 (cm)
Neck: 38 (cm)
RHR: 72 (bpm)
Activity: 3 (Moderately Active)

Generated Report

--- Student Health Report ---
WHR: 0.89
WHtR: 0.47
Body Fat %: 16.82
BMR: 1615.00 kcal/day
TDEE: 2504.00 kcal/day
Lean Body Mass: 53.36 kg
Body Surface Area: 1.75 m²
Resting Heart Rate: 72.00 bpm

***CODE OF THE PROJECT*::**

```c
#include <stdio.h>
#include <string.h>

struct Student
{
int roll, age, sys, dia, hr;
float height, weight, bmi;
char name[20], status[30];
} s[50];

int n = 0;

const char* health(float bmi, int
sys, int dia, int hr) {
int score = 0;
if (bmi >= 18.5 && bmi < 25) score
+= 25;
else if (bmi >= 25 && bmi < 30)
score += 15;
else score += 5;
if (sys < 120 && dia < 80) score
+= 25;
else if (sys < 140 && dia < 90)
score += 15;
else score += 5;
if (hr >= 60 && hr <= 100) score
+= 25;
else score += 10;
if (score >= 70) return
```

```
ƒ

void addStudent() {
printf("\nRoll: "); scanf("%d",
&s[n].roll);
printf("Name: "); scanf("%s",
s[n].name);
printf("Age: "); scanf("%d",
&s[n].age);
printf("Height(cm): ");
scanf("%f", &s[n].height);
printf("Weight(kg): ");
scanf("%f", &s[n].weight);
printf("Sys BP: "); scanf("%d",
&s[n].sys);
printf("Dia BP: "); scanf("%d",
&s[n].dia);
printf("Heart Rate: ");
scanf("%d", &s[n].hr);
float h = s[n].height / 100.0;
s[n].bmi = s[n].weight / (h * h);
strcpy(s[n].status,
health(s[n].bmi, s[n].sys,
s[n].dia, s[n].hr));
printf("\nAdded! BMI = %.2f |
Status = %s\n", s[n].bmi,
s[n].status);
n++;
}
```

Output of program::

Result output:

1.Add 2.Display 3.Search 4.Exit

Choice: 1

Roll: 101

Name: riya

Age: 19

Height(cm): 160

Weight(kg): 52

Sys BP: 115

Dia BP: 75

Heart Rate: 78

Added! BMI = 20.31 | Status = Excellent

1.Add 2.Display 3.Search 4.Exit

Choice: 1

Roll: 102

Name: arjun

Dia BP: 85

Heart Rate: 95

Added! BMI = 26.99 | Status = Good

1.Add 2.Display 3.Search 4.Exit

Choice: 2

ROLL NAME BMI STATUS

101 riya 20.3 Excellent

102 arjun 27.0 Good

1.Add 2.Display 3.Search 4.Exit

Choice: 3

Enter roll: 102

Found!

Name: arjun

BMI: 26.99

Status: Good

1.Add 2.Display 3.Search 4.Exit

Conclusion

The Student Health Tracker project successfully calculates basic health indicators like WHR and BMI using C programming.

Its menu-driven structure makes the system simple and user-friendly.

Modular coding improves clarity, maintainability, and future expansion.

The project helps users understand their health status through quick calculations.

Overall, it meets its objective of providing an efficient and easy-to-use health monitoring tool.

THANK YOU 👍