COMP 322
Winter Semester 2024

INSTRUCTOR: DR. CHAD ZAMMAR
chad.zammar@mcgill.ca

Assignment 1: Exploring Functions, Files and arrays.

Due date: 10 February 2024, 10:00 AM.

Before you start:

- Research for similar problems on the internet is recommended. However, your submission should reflect individual work and personal effort.
- Some of the topics may not be covered in class due to our limited time. You are encouraged to find answers online. You can also reach out to the TAs for guidance.
- Please submit your assignment before the due date to avoid penalties or worse risking your assignment being rejected.
- Submit one file called **assignment1.cpp** containing all the functions together with their implementations. It will also contain the main() function that runs everything.

Make sure your code is clear and readable. **Readability of your code as well as the quality of your comments will be graded**.

- No submission by email. Submit your work to mycourse.
- If your code does not compile it will not be graded.
- For any assignment related question, feel free to ask in the appropriate Ed discussion thread that is created for this purpose. Please don't reveal any solution or long code snippet in the discussion thread.

Objective:

In this assignment, you will design and implement a Health Assistant Program using C++. The program will interact with users to calculate their body fat percentage and propose a personalized calorie breakdown intake, aiding them in their journey to improve their health.

Tasks:

Q1: Input gathering Function [10 pts]

The Health Assistant Program aims to gather essential information for accurate calculations. Write a C++ function that will prompt the user to input the following details:

1. Gender: Please specify your gender as either male or female.
2. Age: Enter your age.
3. Weight: Enter your body weight in kilograms.
4. Waist: Input your waist measurement using centimeters.
5. Neck: Provide your neck measurement in centimeters.
6. Height: Input height measurement in centimeters.
7. Lifestyle: Provide information about your current lifestyle: sedentary, moderate (moderately active) or active.

For Female Users Only:

If you identify as female, the program will additionally request:

1. Hip Measurement: Enter your hip measurement in centimeters.

The function should have the following signature:

void getUserDetails();

All inputs will be stored in global variables created for this purpose.

Q2: Body Fat Percentage [17 pts]

The calculation of body fat percentage using waist and neck measurements typically involves the use of the U.S. Navy formula. The formula is different for males and females. Here are the formulas:

For Males:

BF_percentage = 495 / (1.0324 - 0.19077 * log10(waist - neck) + 0.15456 * log10(height)) - 450

For Females:

BF_percentage = 495 / (1.29579 - 0.35004 * log10(waist + hip - neck) + 0.22100 * log10(height)) - 450

Implement a C++ function called get_bfp() that implements the previous formulas.

The function will have the following signature:

std::pair<int, string> get_bfp(double waist, double neck, double height, double hip, string gender, int age);

The function returns back the body fat percentage value together with the associated group (low, normal, high, very high) according to the following chart:

BODY FAT PERCENTAGE GROUPS

Sex	Age	Low	Normal	High	Very High
Female	20 – 39	< 21	21.0 – 32.9	33.0 – 38.9	≥ 39
	40 – 59	< 23	23.0 – 33.9	34.0 – 39.9	≥ 40
	60 – 79	< 24	24.0 – 35.9	36.0 – 41.9	≥ 42
Male	20 – 39	< 8	8.0 – 19.9	20.0 – 24.9	≥ 25
	40 – 59	< 11	11.0 – 21.9	22.0 – 27.9	≥ 28
	60 – 79	< 13	13.0 – 24.9	25.0 – 29.9	≥ 30

Adapted from NIH/WHO Guidelines for BMI; Gallagher et al, American Journal of Clinical Nutrition, Vol. 72, September 2000

Note: there are multiple techniques in C++ to return more than one value. We will use std::pair for this purpose. You are invited to research how std::pair works.

Q3: How many calories per day? [15 pts]

Now we need to create a function that determines the needed daily calorie intake based on lifestyle, gender and age. We'll consider three lifestyles: sedentary, moderate (moderately active) and active. Use the following chart to determine the required calories:

Gender	Age	Calorie needed for each activity level		
		Sedentary	Moderately Active	Active
Male	19-30	2400	2600-2800	3000
	31-50	2200	2400-2600	2800-3000
	51+	2000	2200-2400	2400-2800
Female	19-30	2000	2000-2200	2400
	31-50	1800	2000	2200
	51+	1600	1800	2000-2200

The function’s signature is:

int get_daily_calories(double age, string gender, string lifestyle);

Function’s output is the number of needed daily calories

Q4: Macronutrient breakdown [15 pts]

Having identified the target calorie intake for an individual, the next step involves breaking down the meals to ensure the right proportions of protein, fat, and carbohydrates. A commonly recommended breakdown is 50% carbohydrates, 30% protein, and 20% fat. Let's customize the meals, keeping in mind that 1 gram of carbohydrates equals 4 calories, 1 gram of protein equals 4 calories, and 1 gram of fat equals 9 calories.

Now, let's code up a C++ function named **meal_prep()** that will determine the required grams of protein, fat, and carbs based on the individual's daily caloric needs. The function's signature is:

void meal_prep(int calories_input, double& carbs_output, double& protein_output, double& fat_output);

The function will return three values corresponding for the amount in grams of carbs, amount in grams of protein and amount in grams of fat. This time we will use an old technique to return back multiple values in C++ out of a function using references. You are invited to research this method in order to understand how it works.

Q5: Display [10 pts]

Create a user-friendly interface that displays the input details about the user such as age, weight etc, body fat percentage, suggested daily caloric intake, and macronutrient breakdown. Ensure that the information is presented in a clear and readable format.

Function's signature is: **void display();**

Q6: Persistence [16 pts]

When rerunning the program, it's essential to preserve the previously input data for continuity. To achieve this, let's create a function named serialize(). This function is designed to save all user data to a CSV-formatted file, where each user's information corresponds to a line in the file.

The function's signature is: **void serialize(string filename);**

The serialize() function will take the filename as a parameter and ensure that the user data is persistently stored for seamless access across multiple program executions.

If you run your program multiple times, the csv file will have as many rows as runs. Each run and therefore each line in the csv file will correspond to a user.

The csv file should have the following format:

Gender, Age, Weight, Waist, Neck, Hip, Height, Lifestyle

Example:

male, 28, 72, 91, 43,, 172, sedentary

Female, 23, 61, 68, 36, 70, 170, moderate

Note that the hip measurement is kept empty for males.

Note: you are not allowed to use any csv utility library that does the job. You need to code the logic yourself.

Q7: Read from file [17 pts]

To enhance the program, let's modify the main() function to support an optional argument. If a filename is provided following the program's name during execution, the program will load existing user data from the specified CSV file.

To efficiently manage the data, we'll employ C++ vectors, with each vector corresponding to a row in the CSV file. For simplicity, we assume that the file will reside in the same local folder as the program.

The function that you must create for this purpose is called **readFromFile** and has the following signature:

void readFromFile(string filename);

Note: you are not allowed to use any csv utility library that does the job. You need to code the logic yourself. You also need to research the use of vectors in C++.

Sample test main function:

```
int main(int argc, char* argv[]) {
    // Check if a filename is provided as an argument
    if (argc > 1) {
        // Load existing user data from the specified CSV file
        readFromFile(argv[1]);
    } else {
        // Gather user details
        getUserDetails();

        // Calculate body fat percentage
        auto bfpResult = get_bfp(g_waist, g_neck, g_height, g_hip,
g_gender, g_age);
        std::cout << "Body Fat Percentage: " << bfpResult.first << "% (" <<
bfpResult.second << std::endl;

        // Calculate daily calorie intake
        int dailyCalories = get_daily_calories(g_age, g_gender,
g_lifestyle);
        std::cout << "Daily Caloric Intake: " << dailyCalories << "
```

```
calories" << std::endl;

        // Calculate macronutrient breakdown
        double carbs, protein, fat;
        meal_prep(dailyCalories, carbs, protein, fat);
        std::cout << "Macronutrient Breakdown:\n";
        std::cout << "Carbs: " << carbs << "g, Protein: " << protein << "g,
Fat: " << fat << "g" << std::endl;

        // Display user information
        display();

        // Save user data to a CSV file
        serialize("user_data.csv");
    }

    return 0;
}
```