2023/2024

Mini Project Report

Parachute jump

Prepared by:
Amini Safa
Semaoui Maroua
Groupe: 06

SOMMAIRE

01 INTRODUCTION

02 PROBLEMATIC

03 ANALYSIS

04 ALGORITHM

05 PROGRAMMING

06 CONCLUSION

INTRODUCTION

This mini-project involves creating a program that simulates a parachutist's jump, taking into account various factors such as mass, fall speed, acceleration, altitude, and air friction. The simulation dynamically calculates and displays the parachutist's characteristics.

The project is structured into distinct phases: free fall, speed of sound, maximum speed, and parachute opening. The program's philosophy aims to accurately represent the physical principles governing the fall, adjusting parameters based on specific conditions.

The *development of the program posed challenges due to the dynamic nature of the parachutist's descent and the necessity to incorporate changing parameters.*

PROBLEMATIC

How can we develop an efficient C program to simulate the descent of a parachutist, accounting for key variables such as altitude, speed, acceleration, and air friction?

The challenge at hand involves the development of a program to model and simulate the descent of a parachutist, considering factors like altitude, speed, acceleration, and air friction, and detecting critical events such as surpassing the speed of sound and achieving maximum speed, and simulating the opening of the parachute when the altitude drops below 2500 m.

To do this simulation we have:

Key Data to be Manipulated:

Mass (m): *The parachutist's mass m=80 kg.*

Initial Altitude (h0): *The starting altitude of the parachutist h0=39,000 m.*

Speed (v): *The speed of the parachutist with v0=0 m/s.*

***Time* (*t*): With t0=0 s.**

***Acceleration* (*a*) *due to Gravity* (*g*):** *A constant with a value of 9.81 m/s^2.*

***Surface Area* (*s*):** *Represents the area of the parachutist exposed to air friction s=2 m^2.*

This is to obtain the desired objectives (results):

Desired Objectives (Results):

Free Fall Simulation: *Calculate and display the parachutist's altitude, speed, and acceleration at each second during the free fall until reaching the ground.*

Speed of Sound and Maximum Speed Notification: *Detect and notify when the parachutist's speed exceeds the speed of sound (343 m/s) and the maximum allowable speed (0.5 m/s^2 acceleration).*

Parachute Opening Simulation: *Simulate the opening of the parachute when the altitude drops below 2500 m, adjusting the surface area and updating the initial condition.*

ANALYSIS

The parachute jump simulation project involves modeling a parachutist's descent, considering inputs:

- Mass of the parachutist (m).
- Initial altitude (h0).
- Initial fall speed (v0).
- Gravitational constant ($g = 9.81\ m/s^2$).
- Surface area (initially set to $2.0\ m^2$).
- Opening altitude for parachute adjustment.
- Final surface area after parachute opening.

and Outputs:

- Time (t).
- Current altitude (h).
- Fall speed (v).
- Acceleration (a).

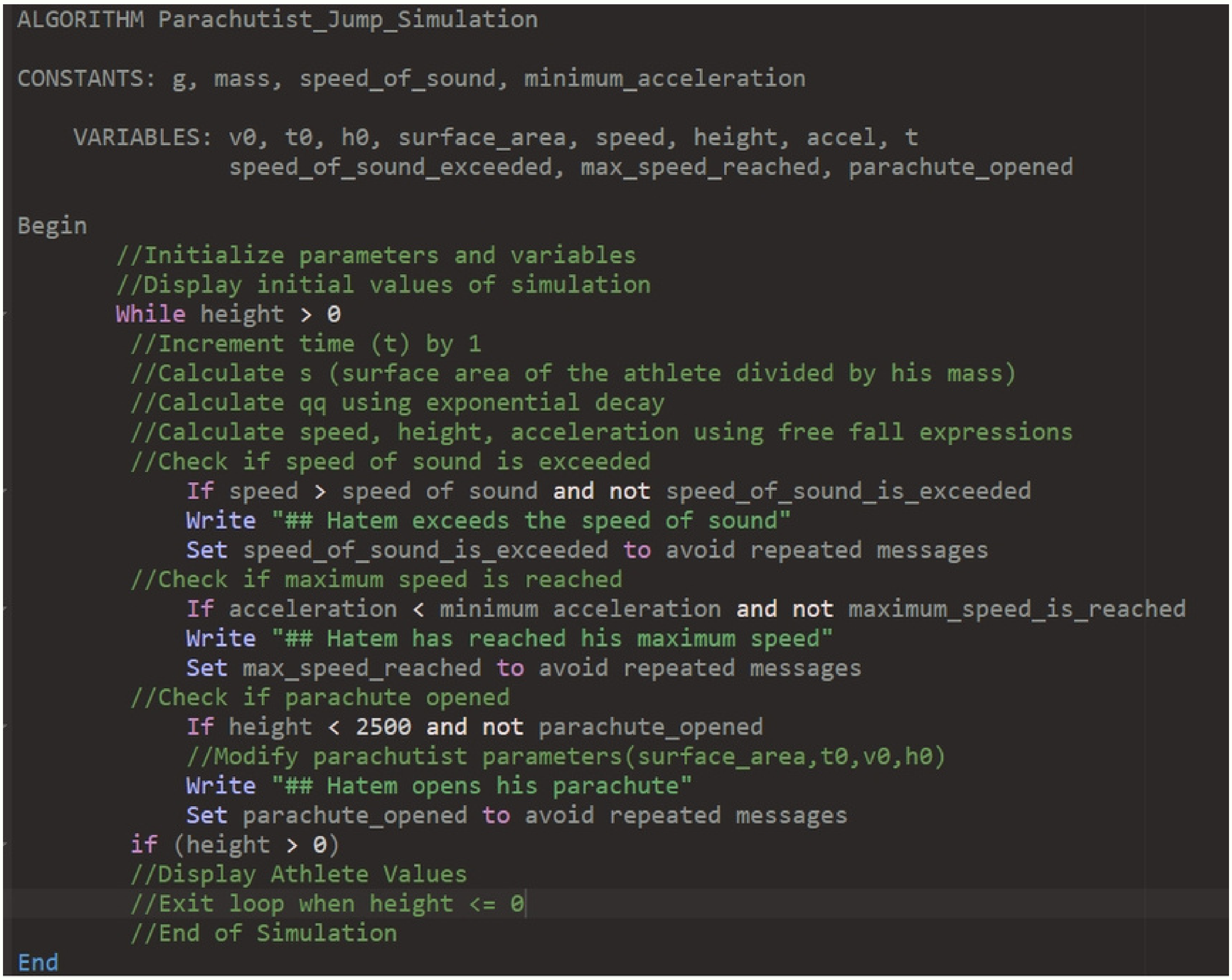
The top-down analysis outlines key steps:

- Initialize constants and variables.
- Display initial values.
- Simulate free fall with dynamic updates.
- Simulate parachute opening with surface area adjustments
- Alerts and message:

Display message: "## Hatem opens his parachute."
Display alerts if the speed of sound is exceeded and if the maximum speed is reached.

The program employs theoretical tools :
Kinematic equations for motion under constant acceleration.
Exponential decay (for calculating q in the free fall formula).
Basic principles of dynamics and aerodynamics.

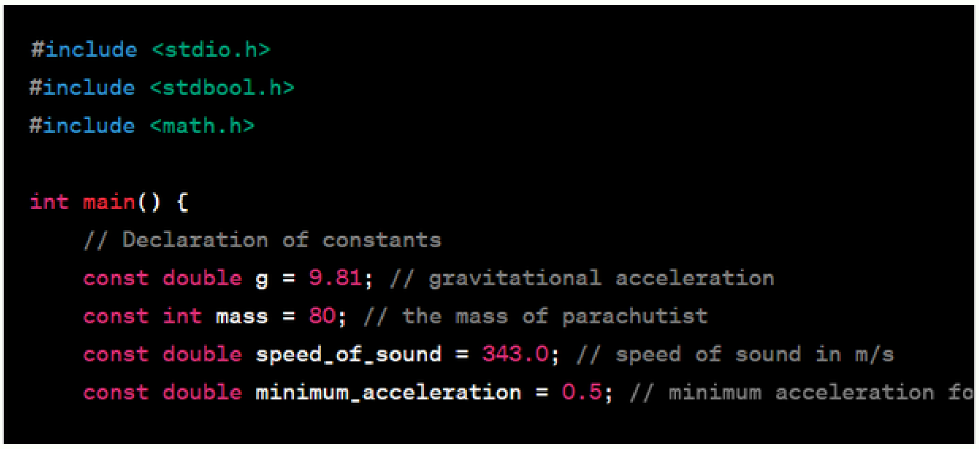
ALGORITHM



```
ALGORITHM Parachutist_Jump_Simulation

CONSTANTS: g, mass, speed_of_sound, minimum_acceleration

    VARIABLES: v0, t0, h0, surface_area, speed, height, accel, t
               speed_of_sound_exceeded, max_speed_reached, parachute_opened

Begin
       //Initialize parameters and variables
       //Display initial values of simulation
       While height > 0
        //Increment time (t) by 1
        //Calculate s (surface area of the athlete divided by his mass)
        //Calculate qq using exponential decay
        //Calculate speed, height, acceleration using free fall expressions
        //Check if speed of sound is exceeded
            If speed > speed of sound and not speed_of_sound_is_exceeded
            Write "## Hatem exceeds the speed of sound"
            Set speed_of_sound_is_exceeded to avoid repeated messages
        //Check if maximum speed is reached
            If acceleration < minimum acceleration and not maximum_speed_is_reached
            Write "## Hatem has reached his maximum speed"
            Set max_speed_reached to avoid repeated messages
        //Check if parachute opened
            If height < 2500 and not parachute_opened
            //Modify parachutist parameters(surface_area,t0,v0,h0)
            Write "## Hatem opens his parachute"
            Set parachute_opened to avoid repeated messages
        if (height > 0)
        //Display Athlete Values
        //Exit loop when height <= 0
        //End of Simulation
End
```

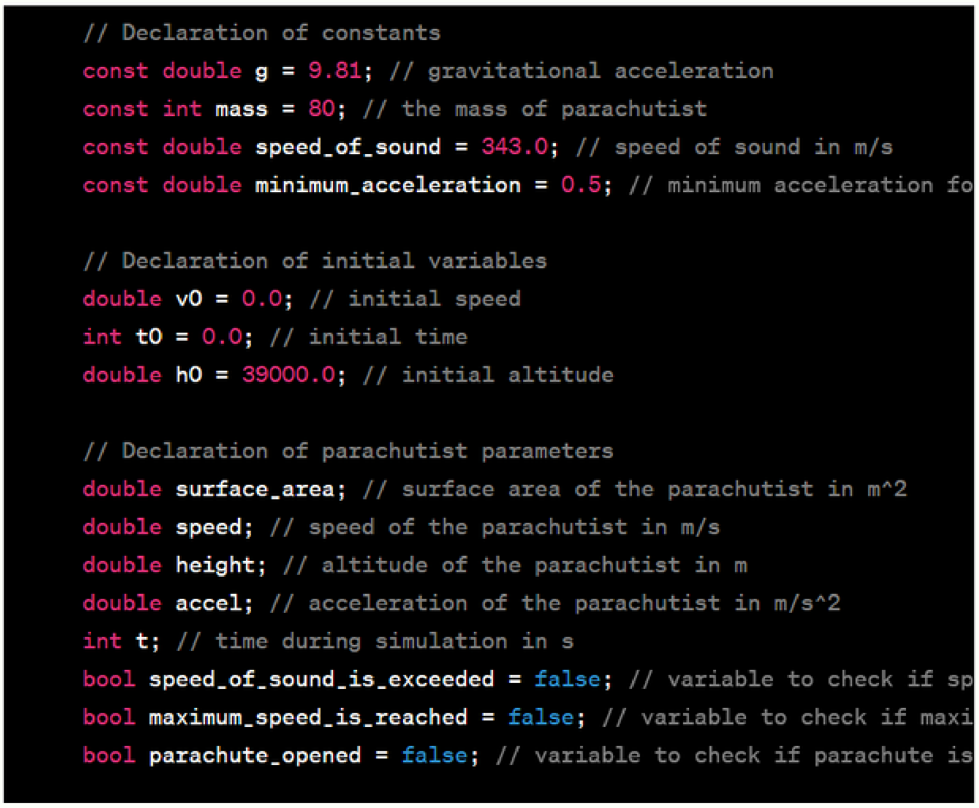
PROGRAMMING

Declaration of constants:



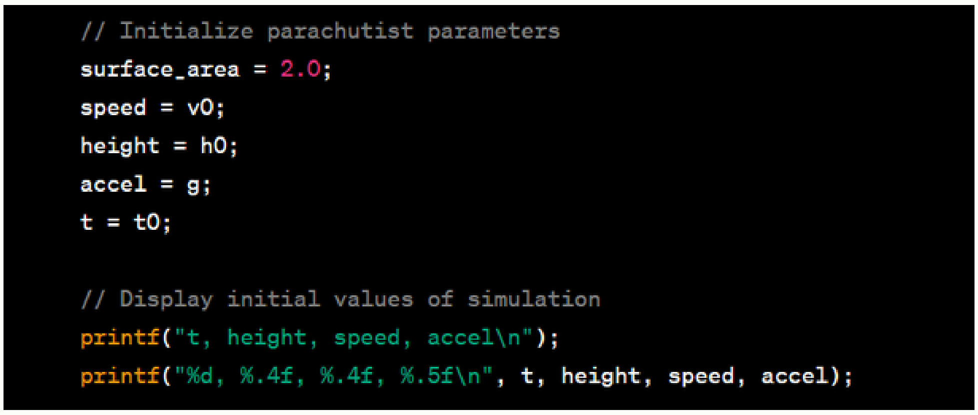
```
#include <stdio.h>
#include <stdbool.h>
#include <math.h>

int main() {
    // Declaration of constants
    const double g = 9.81; // gravitational acceleration
    const int mass = 80; // the mass of parachutist
    const double speed_of_sound = 343.0; // speed of sound in m/s
    const double minimum_acceleration = 0.5; // minimum acceleration fo
```

Declaration of initial variables and parachutist parameters:



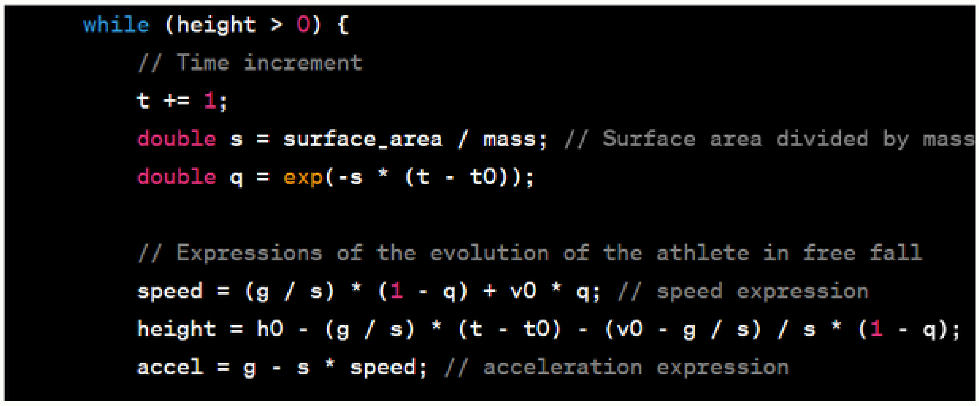
```
    // Declaration of constants
    const double g = 9.81; // gravitational acceleration
    const int mass = 80; // the mass of parachutist
    const double speed_of_sound = 343.0; // speed of sound in m/s
    const double minimum_acceleration = 0.5; // minimum acceleration fo

    // Declaration of initial variables
    double v0 = 0.0; // initial speed
    int t0 = 0.0; // initial time
    double h0 = 39000.0; // initial altitude

    // Declaration of parachutist parameters
    double surface_area; // surface area of the parachutist in m^2
    double speed; // speed of the parachutist in m/s
    double height; // altitude of the parachutist in m
    double accel; // acceleration of the parachutist in m/s^2
    int t; // time during simulation in s
    bool speed_of_sound_is_exceeded = false; // variable to check if sp
    bool maximum_speed_is_reached = false; // variable to check if maxi
    bool parachute_opened = false; // variable to check if parachute is
```

Initialize parachutist parameters and Display initial values of simulation:



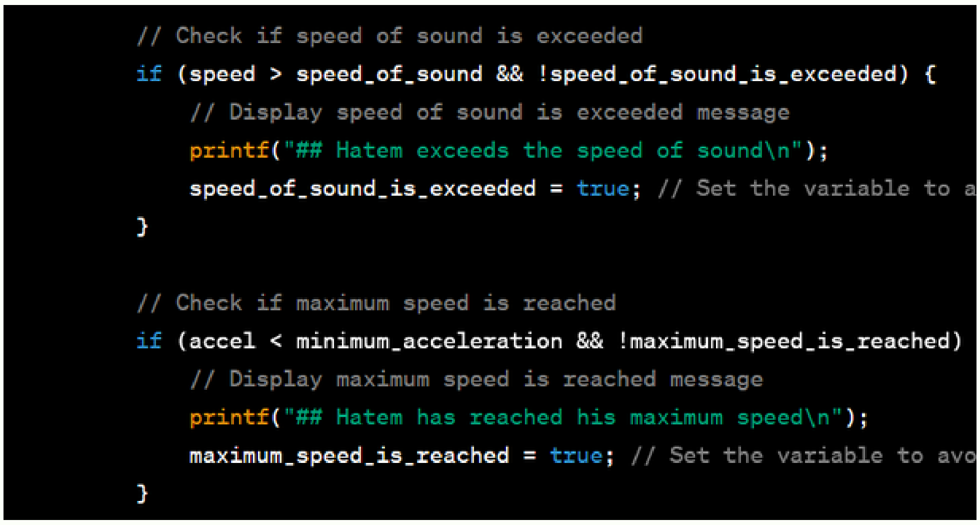
```c
// Initialize parachutist parameters
surface_area = 2.0;
speed = v0;
height = h0;
accel = g;
t = t0;

// Display initial values of simulation
printf("t, height, speed, accel\n");
printf("%d, %.4f, %.4f, %.5f\n", t, height, speed, accel);
```

Calculation of the evolution of the athlete:



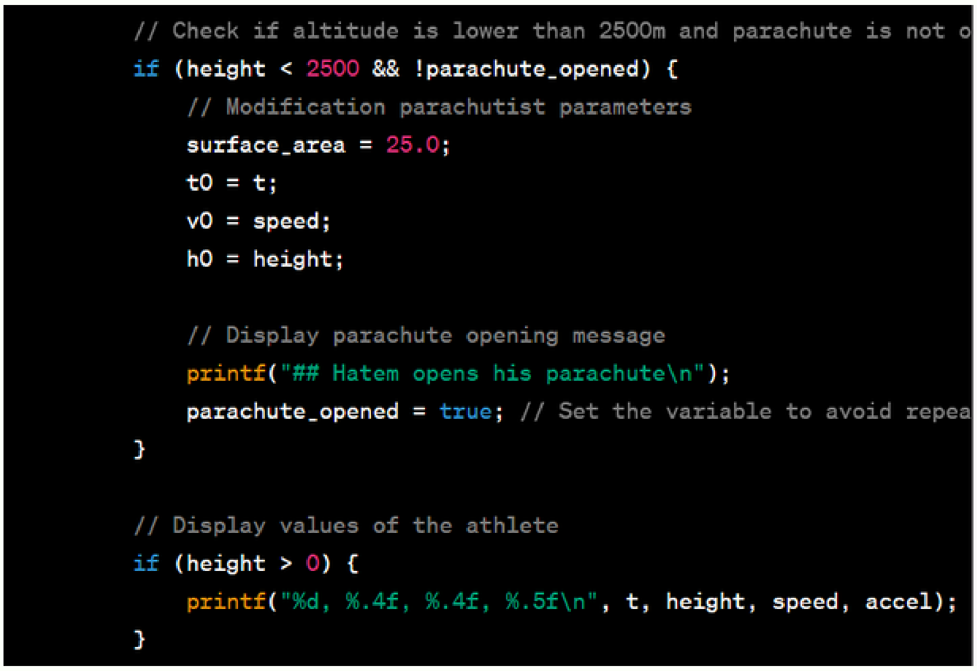
```c
while (height > 0) {
    // Time increment
    t += 1;
    double s = surface_area / mass; // Surface area divided by mass
    double q = exp(-s * (t - t0));

    // Expressions of the evolution of the athlete in free fall
    speed = (g / s) * (1 - q) + v0 * q; // speed expression
    height = h0 - (g / s) * (t - t0) - (v0 - g / s) / s * (1 - q);
    accel = g - s * speed; // acceleration expression
```

Check if speed of sound is exceeded and if maximum speed is reached:



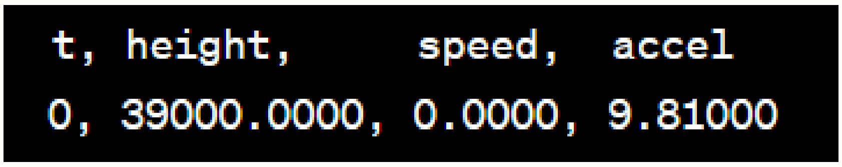
```c
    // Check if speed of sound is exceeded
    if (speed > speed_of_sound && !speed_of_sound_is_exceeded) {
        // Display speed of sound is exceeded message
        printf("## Hatem exceeds the speed of sound\n");
        speed_of_sound_is_exceeded = true; // Set the variable to a
    }

    // Check if maximum speed is reached
    if (accel < minimum_acceleration && !maximum_speed_is_reached)
        // Display maximum speed is reached message
        printf("## Hatem has reached his maximum speed\n");
        maximum_speed_is_reached = true; // Set the variable to avo
    }
```

Parachute opened and Display values of the athlete:



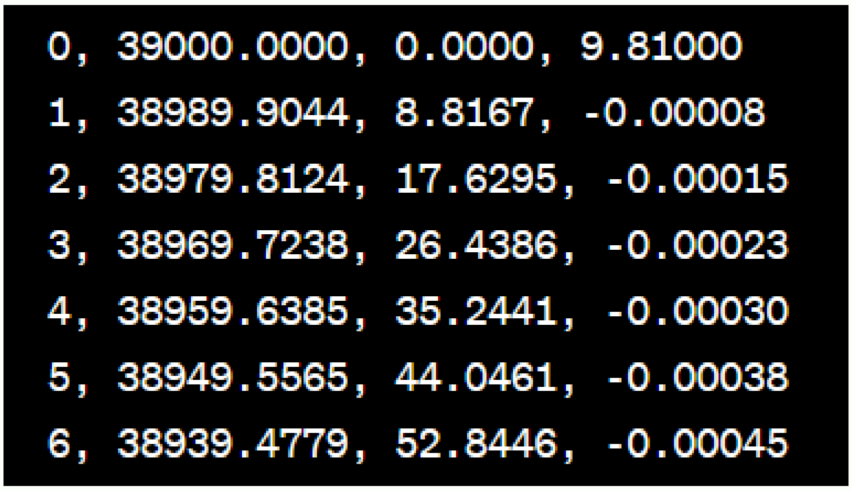
```c
// Check if altitude is lower than 2500m and parachute is not o
if (height < 2500 && !parachute_opened) {
    // Modification parachutist parameters
    surface_area = 25.0;
    t0 = t;
    v0 = speed;
    h0 = height;

    // Display parachute opening message
    printf("## Hatem opens his parachute\n");
    parachute_opened = true; // Set the variable to avoid repea
}

// Display values of the athlete
if (height > 0) {
    printf("%d, %.4f, %.4f, %.5f\n", t, height, speed, accel);
}
```

Display initial values of simulation:



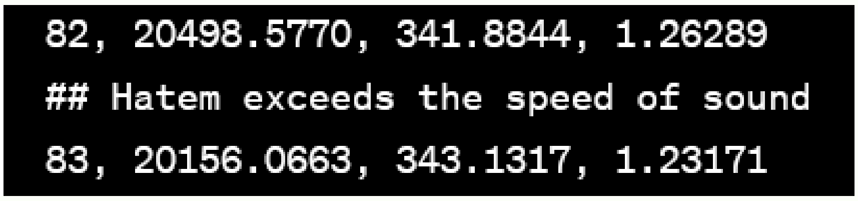
```
 t, height,     speed,   accel
0, 39000.0000, 0.0000, 9.81000
```

Display values of the athlete:



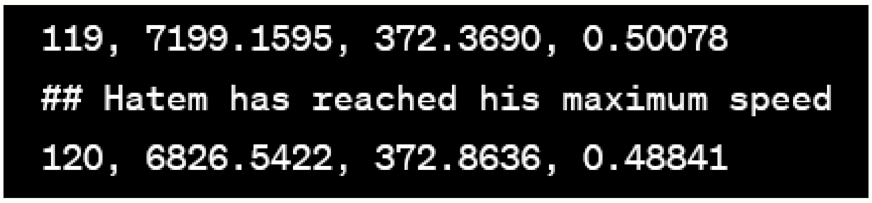
```
0, 39000.0000, 0.0000, 9.81000
1, 38989.9044, 8.8167, -0.00008
2, 38979.8124, 17.6295, -0.00015
3, 38969.7238, 26.4386, -0.00023
4, 38959.6385, 35.2441, -0.00030
5, 38949.5565, 44.0461, -0.00038
6, 38939.4779, 52.8446, -0.00045
```

Display speed of sound is exceeded message:



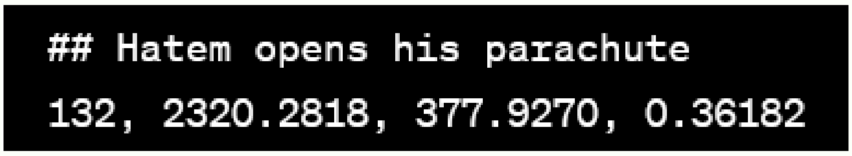
```
82, 20498.5770, 341.8844, 1.26289
## Hatem exceeds the speed of sound
83, 20156.0663, 343.1317, 1.23171
```

Display maximum speed is reached message:



```
119, 7199.1595, 372.3690, 0.50078
## Hatem has reached his maximum speed
120, 6826.5422, 372.8636, 0.48841
```

Display parachute opening message:



```
## Hatem opens his parachute
132, 2320.2818, 377.9270, 0.36182
```

CONCLUSION

The problem at hand involved simulating a parachutist's descent by considering various dynamic factors, such as mass, altitude, and the opening of the parachute affecting surface area.

In addressing this challenge, we developed a parachute jump simulation program with C implementations, implementing a comprehensive algorithm that captures the intricacies of free fall, parachute opening, and associated alerts.

While the simulation program provides a functional representation of a parachutist's fall, certain simplifications were made for the sake of clarity and ease of implementation. The program assumes a constant gravitational acceleration and neglects factors such as air density variations.

Looking ahead, there are opportunities to enhance the simulation by incorporating more sophisticated aerodynamic models and refining the numerical methods for calculating the parachutist's state. Real-world factors like wind resistance and variable gravitational forces could be considered for a more realistic simulation. Moreover, the program could be extended to include graphical visualization, facilitating a more immersive understanding of the parachutist's descent.