TASK 1

3 august 2024
CGPA Calculator

Lana Akoum
Code Alpha

INTRODUCTION

A Cumulative Grade Point Average (CGPA) calculator is an essential tool for both students and educational institutions. It provides a standardized way to measure a student's academic performance over an entire course or program.

The CGPA is calculated by averaging the grade points obtained in all courses, weighted by the number of credits each course carries. This helps in evaluating a student's overall academic progress and standing.

The CGPA calculator is important because it gives a clear pictures of a student's academic achievements.
It helps students understand their strengths and areas where they need to improve. Additionally, CGPA is often a key factor in determining eligibility for scholarships, internships, and further educational opportunities.

This project involves creating a CGPA calculator that takes the grades and credits of different courses as input.
The program will calculate the Grade Point Average (GPA) for a specific term and the overall CGPA for the entire academic program.
By using this calculator, students can easily track their academic performance and plan their studies more effectively.

The Role of ‘struct’ in the CGPA Calcu-lator

In the CGPA calculator, the ‘struct’ is used to efficiently store and manage course information. The ‘course’ structure groups related data variables, such as grade points and credits, under a single entity. This approach provides several benefits:

1. Organization: by grouping course-related together, the ‘struct’ makes the code more organized and readable.
2. Data management: it simplifies the process of passing course information to functions and handling it within the program.
3. Scalability: the ‘struct’ allows for easy expansion by adding new attributes, such as course names or instructors, without significant code changes.

In essence, the struct in the CGPA calculator organizes, manages and scales course data, making it easy to calculate and handle academic performance.

HEADING

Structs: Organizing and Managing Course Data.

Array structure for storing & managing multiple courses

In the CGPA calculator, an array is used to organize and handle data for multiple courses. This array is a list-like structure that stores several ‘course’ objects, each representing a different course with its associated grade, points and credits.

The array allows for systematic storage and easy retrieval of course information. By indexing the array, the program can efficiently access, update, and manipulate the details for each course.
This method ensures that the calculator can handle varying numbers of courses, keeping the data organized and manageable for calculations such as GPA.

HEADING

Managing Course Data with Arrays.

Functions for input, GPA calculation, and CGPA calculation.

In the CGPA calculator, three main functions manage course data and calculate performance metrices:

1. Input Function:
   The input function collects course grades and credits from the user, storing them in an array of 'course' structures for organized processing.
2. GPA calculation function:
   The GPA calculation function computes the term GPA by averaging grade points, weighted by course credits, to measure academic performance for a specific term.
3. CGPA calculation function:
   The CGPA calculation function extends GPA calculations across multiple terms, providing a cumulative metric of overall academic achievement.

HEADING

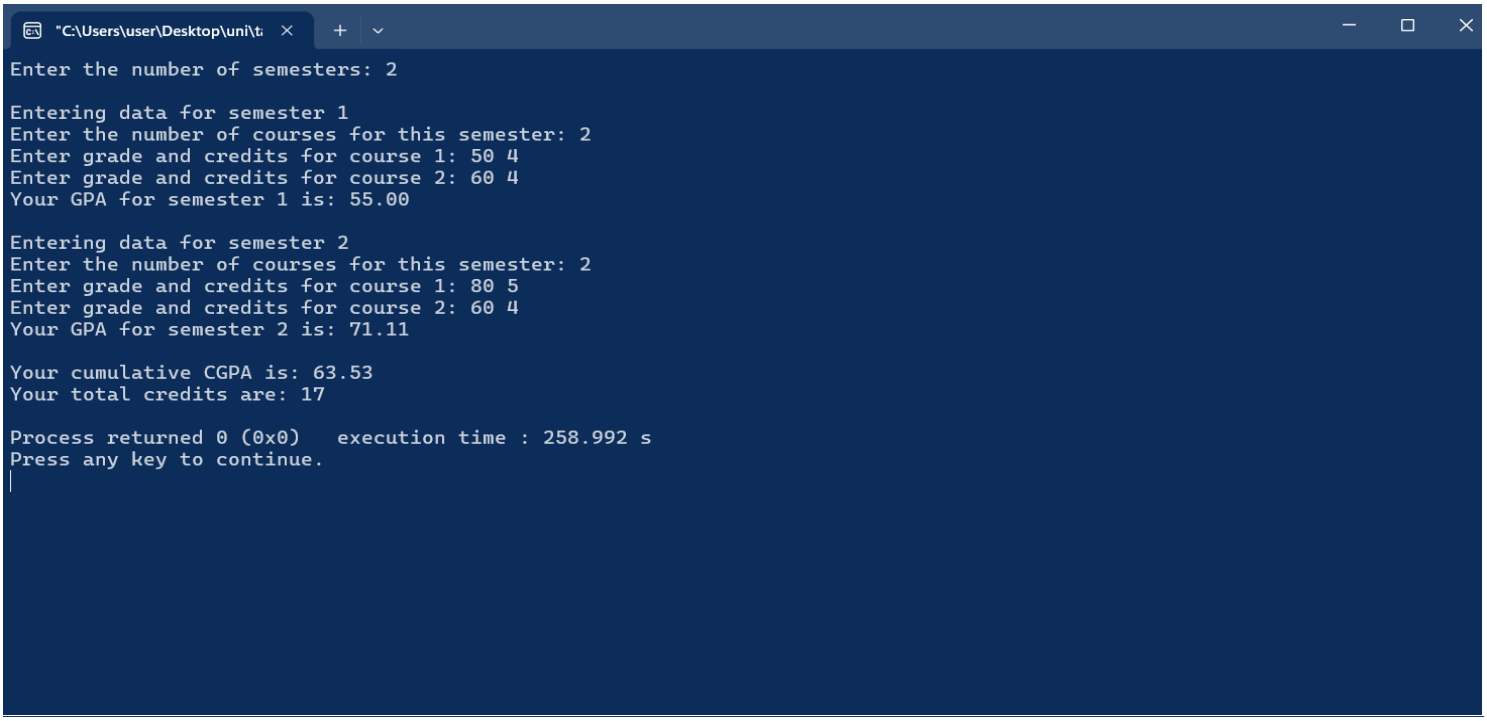
Key functions for managing course data and calculating GPA/CGPA.

The purpose of computing is insight, not numbers.

- Richard Hamming

Program calculations and Display outputs

- The number of semesters.
- Number of courses for each semester.
- Individual grades and credits for each course in a semester.
- GPA for the semester.
- Overall CGPA.
- Total credits earned.



Implementation

Start here | task 1.cpp

```cpp
#include <iostream> //Input/Output Stream Library
#include <cstdio> //C standard I/O library

```

In C++ we use #include<iostream> to include the Input/Output stream library. This Library allows us to perform basic i/p and o/p operations using objects like ‘std::cin’ and ‘std::cout’.

Including <iostream> is essential for interacting with the console in C++ programs.

In C++ we use #include<cstdio> to include the C standard i/p and o/p Library. This library provides C-styles functions for performing input and output operations (printf /scanf …).

Including <cstdio> is useful for performing lower-level I/O operations.

```cpp
// Define the Course structure to store grade and credits
struct Course {
    double grade; // The grade points for the course
    int credits;  // The number of credits for the course
};

```

This structure helps in organizing course data.

```cpp
// Function to collect course data from the user for one semester
void inputCourses(Course c[], int &numCourses) {
    printf("Enter the number of courses for this semester: ");
    std::cin >> numCourses;

    for (int i = 0; i < numCourses; ++i) {
        printf("Enter grade and credits for course %d: ", i + 1);
        std::cin >> c[i].grade >> c[i].credits;
    }
}

```

This function is essential for populating the c array with the necessary information.

```
21  // Function to calculate the GPA for a semester
22  double calculateGPA(Course c[], int numCourses) {
23      double totalGradePoints = 0.0;
24      int totalCredits = 0;
25
26      for (int i = 0; i < numCourses; ++i) {
27          totalGradePoints += c[i].grade * c[i].credits;
28          totalCredits += c[i].credits;
29      }
30
31      return totalGradePoints / totalCredits;
32  }
33
```

This function provides a measure of the student's performance for the semester.

```
34  // Function to calculate the CGPA considering multiple terms
35  double calculateCGPA(Course allCourses[], int totalCourses) {
36      double cumulativeGradePoints = 0.0;
37      int cumulativeCredits = 0;
38
39      for (int i = 0; i < totalCourses; ++i) {
40          cumulativeGradePoints += allCourses[i].grade * allCourses[i].credits;
41          cumulativeCredits += allCourses[i].credits;
42      }
43
44      return cumulativeGradePoints / cumulativeCredits;
45  }
46
```

This function aggregates grades and credits to calculate the student's overall academic performance.

```
47  // Function to calculate the total credits of all courses
48  int calculateTotalCredits(Course c[], int numCourses) {
49      int totalCredits = 0;
50
51      for (int i = 0; i < numCourses; ++i) {
52          totalCredits += c[i].credits;
53      }
54
55      return totalCredits;
56  }
57
```

This function helps in verifying the total academic effort taken by a student.

```
58  int main() {
59      const int MAX_COURSES = 100;
60      Course allCourses[MAX_COURSES * 10]; // Assuming a maximum of 10 semesters
61      int totalCourses = 0;
62      int numSemesters;
63
64      printf("Enter the number of semesters: ");
65      std::cin >> numSemesters;
66
67      for (int t = 0; t < numSemesters; ++t) {
68          Course c[MAX_COURSES];
69          int numCourses;
70
71          printf("\nEntering data for semester %d\n", t + 1);
72          inputCourses(c, numCourses);
73
74          // Calculate and display the GPA for this semester
75          double gpa = calculateGPA(c, numCourses);
76          printf("Your GPA for semester %d is: %.2f\n", t + 1, gpa);
77
78          // Copy courses to allCourses array
79          for (int i = 0; i < numCourses; ++i) {
80              allCourses[totalCourses++] = c[i];
81          }
82      }
83
84      // Calculate and display the CGPA
85      double cgpa = calculateCGPA(allCourses, totalCourses);
86      printf("\nYour cumulative CGPA is: %.2f\n", cgpa);
87
88      // Calculate and display the total credits
89      int totalCredits = calculateTotalCredits(allCourses, totalCourses);
90      printf("Your total credits are: %d\n", totalCredits);
91
92
93      return 0;
94  }
95
```

The main function manages the whole program. It processes multiple terms, gathers course data with `inputCourses`, computes GPA for each term with `calculateGPA`, combines course data into a cumulative array, calculates the overall CGPA with `calculateCGPA`, and totals the credits using `calculateTotalCredits`. Finally, it displays results and guides the End User on using the program.

Summary of Functionalities Achieved

The CGPA calculator program is designed to handle essential academic calculations:

1. **Data collection**: It collects grades and credits for multiple semesters from the user.

2. **GPA calculation**: it computes the grade point average for each semester based on the grades and credits provided.

3. **CGPA calculation**: it calculates the cumulative grade point average by averaging the GPAs across all semesters.

4. **Total Credits calculation**: It sums up the total number of credits from all courses and semesters.

Potential Improvements and Additional features.

1. **Historical data tracking:**
   - Add functionality to track and store academic performance over time, allowing students to monitor their progress across different years.

2. **Integration with Academic Systems:**
   - Integrate the program with academic management systems to automatically pull in grades and course data, streamlining the process.

3. **Mobile Accessibility:**
   - Develop a mobile app version of the calculator to enable students to calculate and check their GPA and CGPA from their smartphones.