

Gradebook with Class Statistics

A C Language Mini-Project

Bipasha Roy

ERP ID: Ru-25-10407

Section: A (IBM)

Guided By:

Naina Ma'am

Project Overview

1

Introduction

2

Problem Statement

3

Tools & Technologies Used

4

Project Working

5

Code Logic Overview

6

Applications

Introduction: Streamlining Academic Assessment

In the dynamic environment of educational institutions, the manual management of student marks can be a time-consuming and error-prone process. This mini-project addresses these challenges by automating key assessment tasks.

Project Automation:

- **Marks Calculation:** Automates the aggregation of scores across various subjects.
- **Grade Assignment:** Applies predefined grading criteria to assign appropriate grades.
- **Class Statistics:** Generates insightful statistics about overall class performance.

Project Foundations:

- **C Programming Language:** Developed entirely using C, showcasing its capabilities for console-based applications.
- **Console-Based Application:** Designed for command-line interaction, making it lightweight and efficient.
- **Beginner-Friendly:** An ideal learning tool for understanding fundamental C concepts.



Problem Statement: The Need for Automated Gradebook Management

The core challenge addressed by this project is the efficient and accurate management of student academic records. Manually processing grades for a large number of students is not only tedious but also prone to human error. This project provides a robust solution by automating critical aspects of gradebook management.

Input Management

- Efficiently input marks for multiple students.
- Handles various subjects to ensure comprehensive record-keeping.

Core Calculations

- Automatically computes total marks for each student.
- Calculates percentage scores accurately.
- Assigns appropriate grades based on predefined criteria.

Insightful Analysis

- Determines subject-wise averages to identify areas of strength or weakness.
- Identifies the top scorer in the class, highlighting individual achievement.

Tools & Technologies Used: Building the Gradebook

This project leverages standard and widely-used tools and technologies, making it accessible and robust for academic and practical applications. The choice of C language emphasizes foundational programming concepts.

Development Environment

- **Programming Language: C** - Chosen for its efficiency, low-level memory management, and foundational role in computer science education.
- **Code Editor: VS Code** - A popular, lightweight, and powerful code editor that supports various programming languages, including C.
- **Compiler: GCC** - A free and open-source compiler system for C and C++ programming languages.

Key C Programming Concepts

- **Arrays:** Used to store collections of data, such as student marks for multiple subjects or a list of subject names.
- **Functions:** Modularize code for specific tasks like calculating totals, percentages, or assigning grades, promoting reusability and readability.
- **Loops:** Enable iteration over student data or subject marks, handling repetitive tasks efficiently.
- **Conditional Statements:** Implement decision-making logic, crucial for grade assignment based on percentage ranges and input validation.

Project Working: A Step-by-Step Overview

The Gradebook system follows a logical flow to process student information and generate reports. This section outlines the operational sequence from data input to statistical output.

01

User Input: Number of Students

The program begins by prompting the user to specify the total number of students whose data will be processed.

02

Student Name Entry

For each student, the user is prompted to enter their full name, ensuring proper identification in reports.

03

Marks Entry Per Subject

Marks are systematically entered for five core subjects: English, Hindi, Mathematics, Social Science, and Science. Input validation ensures marks are within a valid range (0-100).

04

Automated Calculations

Upon receiving all marks, the program automatically calculates the total marks, percentage, and assigns a corresponding grade for each student.

05

Class Statistics Display

After processing all students, the system displays comprehensive class statistics, including subject-wise averages and the top-scoring student.



Code Logic Overview: Behind the Scenes

The functionality of the Gradebook project is built upon fundamental C programming constructs. Understanding these elements provides insight into how the program efficiently manages and processes data.



Data Storage with Arrays

Arrays are critically utilized to store student marks for each subject and to maintain a list of subject names, allowing for organized data handling and easy iteration.



Modular Design with Functions

The project employs distinct functions for specific calculations: `calculateTotal()`, `calculatePercentage()`, and `calculateGrade()`. This modular approach enhances code readability, maintainability, and reusability.



Efficient Iteration with Loops

`for` loops are used extensively to process data for multiple students and iterate through subject marks, ensuring that calculations and data entry are handled systematically for every record.



Robust Input Validation

A do-while loop is implemented for marks input, ensuring that all entries are within the valid range of 0 to 100. This prevents incorrect data from corrupting calculations and reports.



Top Scorer Identification

Conditional logic is applied to compare each student's total marks, allowing the program to accurately identify and record the name of the student with the highest total score.

Output Screenshots & Code Snippets

This section provides a visual representation of the Gradebook in action, showcasing both the user interaction and the core code that drives its functionality.

Example Output:

The output clearly presents individual student reports, followed by overall class statistics. This includes:

- Student Name and individual subject marks
- Total Marks, Percentage, and assigned Grade
- Average marks for each subject across the class
- Identification of the class's top scorer

```
Enter number of students: 3

Enter name of student 1: Aniket
Enter marks for English (0-100): 67
Enter marks for Hindi (0-100): 78
Enter marks for Mathematics (0-100): 98
Enter marks for Social Science (0-100): 76
Enter marks for Science (0-100): 56

Student Report
-----
Name: Aniket
English Marks: 67
Hindi Marks: 78
Mathematics Marks: 98
Social Science Marks: 76
Science Marks: 56
Total Marks: 375
Percentage: 75.00%
Grade: B+
```

```
Class Statistics
-----
Average marks in English: 80.33
Average marks in Hindi: 78.00
Average marks in Mathematics: 90.67
Average marks in Social Science: 68.33
Average marks in Science: 56.67
Top Scorer: Sakshi (392 marks)
```

Key Code Snippets:

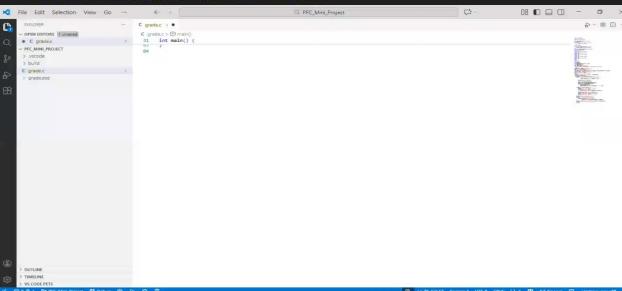
The full code illustrates the application of C concepts. Here's a glimpse into critical functions:

```
printf("\n Student Report\n");
printf("-----\n");
printf("Name: %s\n", name);
for (int i = 0; i < SUBJECTS; i++) {
    printf("%s Marks: %d\n", subjects[i],
marks[i]);
}

printf("Total Marks: %d\n", total);
printf("Percentage: %.2f%%\n", percentage);
printf("Grade: %s\n", grade);
printf("-----\n");
printf("\n Class Statistics\n");
printf("-----\n");
for (int i = 0; i < SUBJECTS; i++) {
    printf("Average marks in %s: %.2f\n",
subjects[i], (float)subjectTotal[i]/
students);
}

printf("Top Scorer: %s (%d marks)\n",
topScorer, highestTotal);
printf("-----\n");
return 0;
```

Video Demonstrating The Inputs and Outputs:



Video of Output

A screenshot of a code editor window titled "PFC_Mini_Project". The left sidebar shows an "EXPLORER" view with a project structure. Under "OPEN EDITORS", there is one unsaved file: "C grade.c". Under "PFC_MINI_PROJECT", there are files ".vscode", "build", "grade.c", and "grade.exe". The main editor area displays the "grade.c" file, which contains the following code:

```
C grade.c | *
C grade.c > main()
31 int main() {
32     ...
84 }
```

The status bar at the bottom shows the following information: Line 78, Column 52, Spaces: 4, UTF-8, CRLF, (1) C, (1) Go Live, windows-gcc-x64, and a refresh icon.

Applications & Advantages: Impact of the Gradebook System

The developed Gradebook system offers significant utility and benefits, extending its application beyond a simple academic exercise to real-world scenarios.

Practical Applications:

School Result Processing: Automates the cumbersome process of preparing student result cards and reports at the school level.

College Internal Marks System: Provides a streamlined solution for managing internal assessment marks for college courses.

Academic Performance Analysis: Facilitates quick analysis of student and class performance, helping educators identify trends and areas for improvement.

Key Advantages

Ease of Use: Designed with a simple, intuitive console interface, making it user-friendly for educators and administrators.

Reduced Manual Errors: Minimizes human errors associated with manual calculations and data entry, ensuring higher accuracy.

Fast Calculations: Processes large datasets quickly, significantly reducing the time required for result compilation.

Beginner Friendly: Serves as an excellent practical example for students learning C, demonstrating core concepts in a functional application.

Future Scope & Conclusion: Evolving the Gradebook

While the current Gradebook system provides robust functionality, there are numerous avenues for future enhancements to elevate its capabilities and user experience.

Potential Future Enhancements:

Add File Handling: Implement features to read and write student data to files, enabling persistent storage and retrieval of records without manual re-entry.

Graphical User Interface (GUI): Transition from a console-based application to a GUI, offering a more intuitive and visually appealing user experience.

Permanent Data Storage: Integrate with databases or advanced file systems to ensure data persistence, allowing for long-term record keeping and analysis.

Expand Subject Customization: Allow users to dynamically add or remove subjects, making the system more flexible and adaptable to different curricula.

Conclusion: A Foundation for Learning

Successful Implementation: The project successfully delivers a functional and efficient gradebook system, achieving all its primary objectives.

Enhanced C Understanding: It served as an invaluable platform for deeply understanding and practically applying core C programming concepts.

Real-Life Academic Utility: Demonstrates the practical applicability of programming skills in solving real-world academic challenges, making it suitable for various educational settings.

Thank You!