Student Management System

Program to store Student details such as roll number name marks and grades allow operative add Student display all students search by roll number and calculate average marks

Understanding the Core Structure

Our Student Management System are global constants and arrays designed to store and manage student information. This setup allows for easy access and modification of data throughout the program. We define a maximum capacity for students and use parallel arrays to keep their details organised.

1

Constants & Global Variables

- `MAX_STUDENTS`: Defines the maximum number of students the system can handle (e.g., 100).
- `rollNumbers[]`: Stores unique identification numbers for each student.
- `names[]`: Holds the names of students.
- `marks[]`: Stores the marks obtained by each student.
- `grades[]`: Stores the calculated grades for each student.
- `studentCount`: Keeps track of the current number of students in the system.

Essential Utility Functions

Several helper functions are crucial for the system's functionality, simplifying complex operations and ensuring data integrity. These functions encapsulate specific tasks, making the code modular and easier to maintain. We will look at grade calculation, data saving, and data loading.

Calculate Grade

The `calculateGrade()` function dynamically assigns a letter grade (A, B, C, D, E, F) based on a student's marks. This logic ensures consistent grading across all students.

Save to File

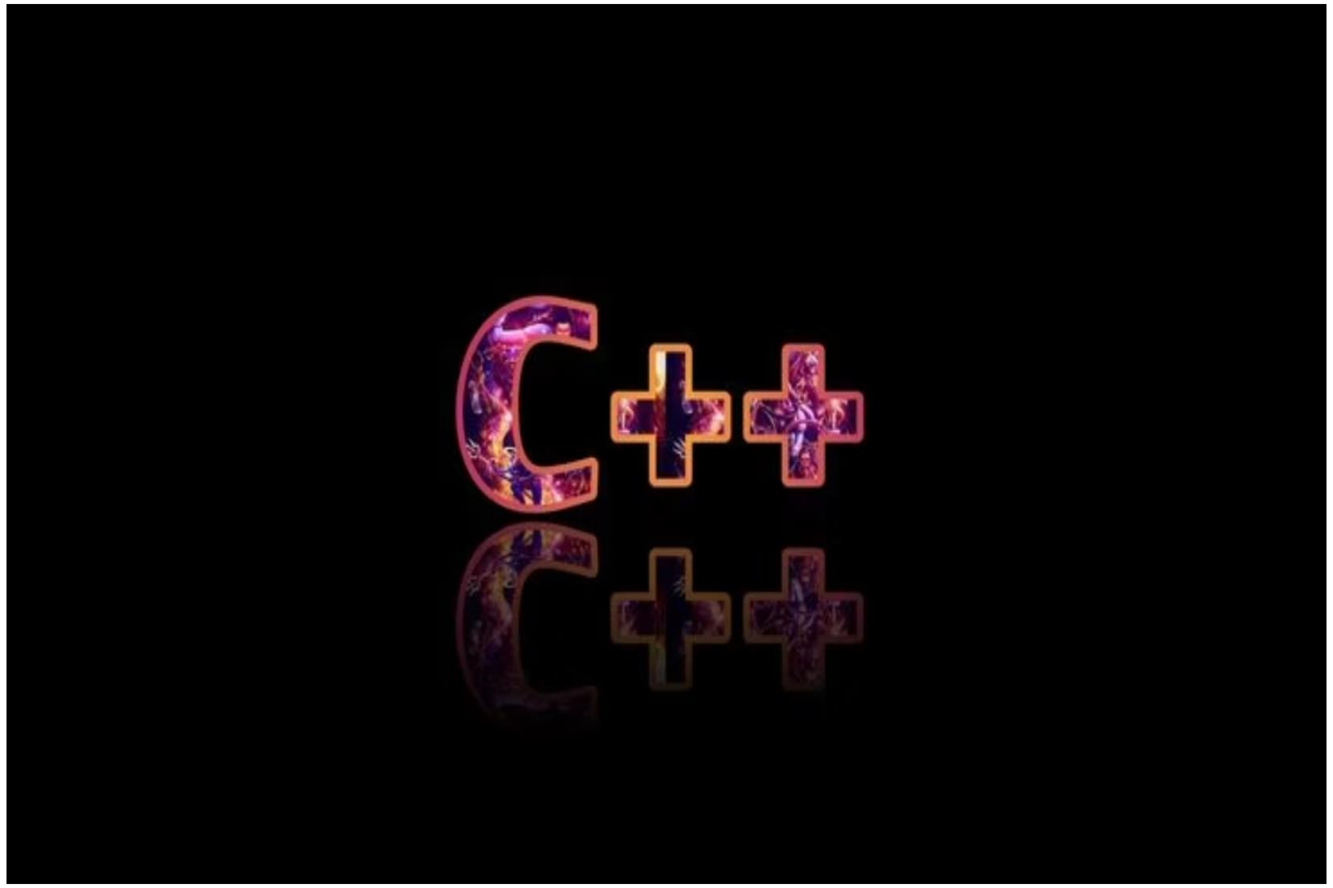
The `saveToFile()` function diligently writes all current student data to a CSV file named "students.csv". This feature is vital for persistent storage, preventing data loss when the program closes.

Load from File

Conversely, the `loadFromFile()` function reads student data from "students.csv". It ensures that upon program startup, any previously saved student records are restored, providing a seamless user experience.

Managing Student Records: Add and Display

The core of any student management system lies in its ability to add new records and display existing ones. These functions are designed to be user-friendly, allowing for straightforward data entry and clear presentation of information.



Adding a New Student

The `addStudent()` function prompts the user to enter details like Roll Number, Name, and Marks. It includes important validation checks to prevent duplicate roll numbers and ensure marks are within a valid range (0-100). The grade is automatically calculated, and the data is saved.

Displaying All Students

The `displayAllStudents()` function provides a clear, formatted table of all students currently in the system. It presents their Roll Number, Name, Marks, and Grade, along with a total count of students, making it easy to review all records at a glance.

Advanced Features: Search, Statistics, and Deletion

Beyond basic management, a comprehensive system offers functionalities to search for specific records, analyse academic performance, and maintain data hygiene by removing entries. These features add significant value and utility to the student management system.

1

Search by Roll Number

The `searchByRollNumber()` function allows users to quickly find a student's details by entering their unique roll number. It then displays all information for that student or indicates if the student is not found.

2

Calculate Marks Statistics

The `calculateAverageMarks()` function computes and displays key statistics such as the total number of students, average marks, highest marks, and lowest marks. This is invaluable for academic analysis.

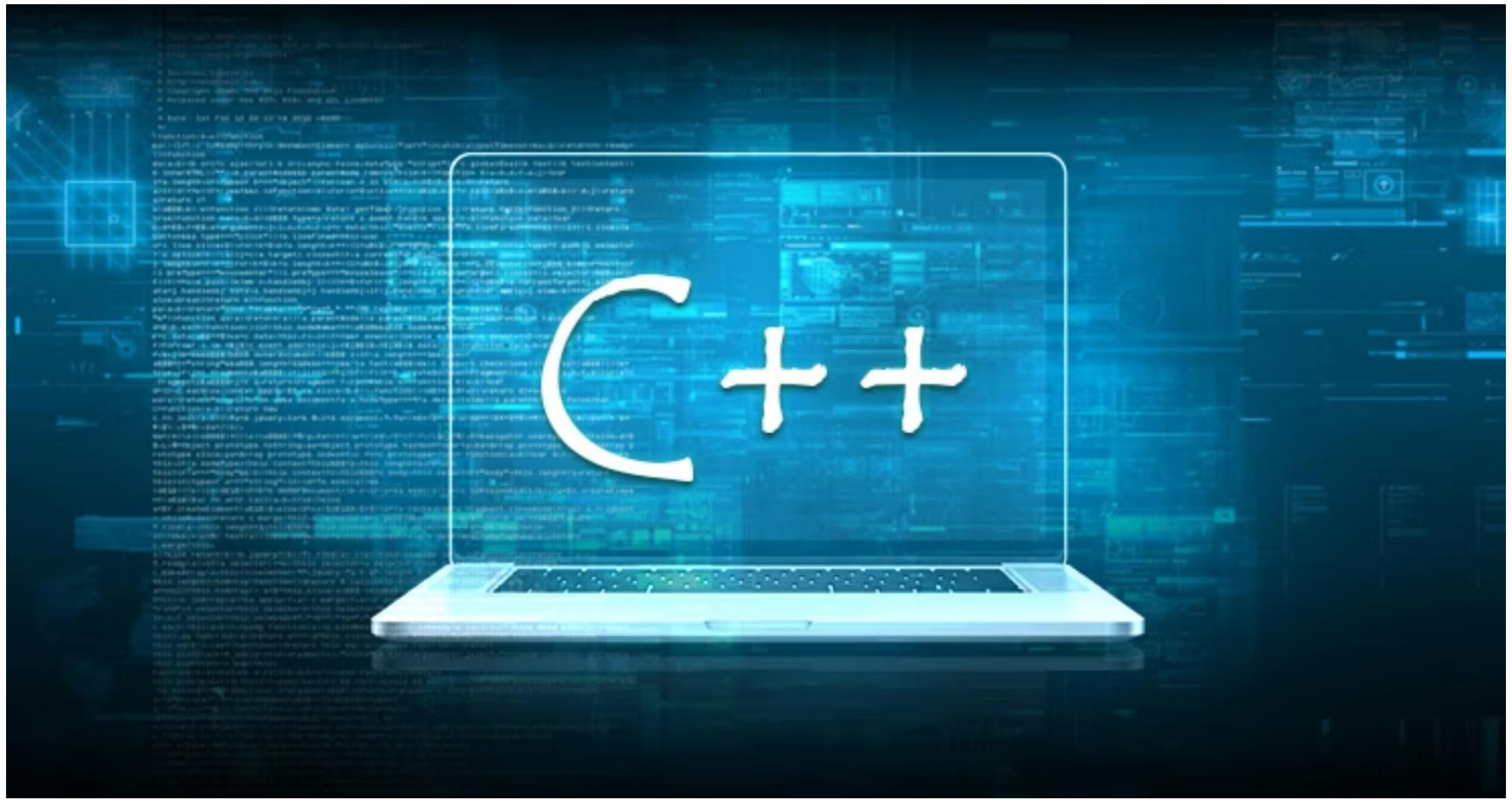
3

Delete Student

The `deleteStudent()` function enables the removal of a student's record from the system using their roll number. It ensures data integrity by shifting subsequent records to fill the gap, maintaining a contiguous data structure.

Putting It All Together: The Main Menu & Sample Data

The main function arrange all these components, presenting a user-friendly menu for interaction. It also includes an important feature for initial setup: adding sample data if no existing records are found.



The Interactive Menu

The `display Menu()` function is central to user interaction, offering options like adding, displaying, searching, calculating statistics, deleting students, and exiting the program. The `main()` function uses a `do-while` loop to continuously present this menu until the user chooses to exit.

Adding Sample Data

The `add SampleData()` function is a convenient feature for first-time users or during testing. If no "students.csv" file is found, it automatically populates the system with a few pre-defined student records, allowing immediate interaction without manual entry.

Key Takeaways & Next Steps

This C++ Student Management System provides a foundational understanding of data handling, modular programming, and user interaction. It's a stepping stone for more complex applications.

→ Modular Design

Functions like `calculateGrade`, `saveToFile`, and `loadFromFile` highlight the importance of breaking down a large program into smaller, manageable units.

→ Data Persistence

The use of CSV files for saving and loading data demonstrates how to achieve data persistence, ensuring information is not lost between program executions.

→ User Interaction

The menu-driven interface showcases effective ways to interact with users, making the program intuitive and easy to navigate.

→ Error Handling

Basic error checks, such as validating roll numbers and marks, are crucial for building robust applications.