Introduction to Data Structures and Algorithms
Prof. Kaushik Gopalan

IPL Data Analysis Project - Report

Team Name: IPL Geeks

Team Members: Pratik Patil, Sanjeev Kumar

Introduction

In the world of sports numbers, where stats help teams win, the Indian Premier League (IPL) shines bright. We love cricket and numbers, so we decided to dig deep into IPL data. Welcome to our project, where we use data and math to help fans of fantasy sports, especially those who play Dream11.

Choosing the Data

First, we needed the right data. We found a big set of IPL data on Kaggle, a place full of datasets. This data, put together by Raj Sengottaiyan, covers lots of IPL seasons and has information on matches and players.

The Goal

Our aim is to help Dream11 players. Fantasy sports, like IPL on Dream11, are really popular. We want to give players the information they need to pick the best teams. We want to make it easy for them to use data to make smart choices and win more often.

Overview of our IPL Data Analysis (Abstract Overview):-

Have you ever wondered which IPL team consistently makes it to the top? Or perhaps you're curious about the batting giants or bowling wizards who dominate the league? We embarked on a fascinating journey to explore the Indian Premier League (IPL) through the lens of data structures and algorithms. Our project culminated in a comprehensive program that empowers us to delve into the heart of the IPL.

Unveiling the Data:

Imagine a treasure trove of IPL statistics at your fingertips! Our program provides a user-friendly interface to navigate through this treasure. We can choose to explore data for a specific year, focus on a particular season, or delve into broader trends across multiple seasons.

The Heroes of the Game:

But data is just numbers without context. Our program brings these numbers to life by calculating batting and bowling points, potentially using formulas that capture the essence of on-field performance. We can identify the players who consistently rack up the most runs, dominate with their wickets, and boast impressive strike rates.

Team Dynamics:

No team wins without a strong foundation. Our program allows us to delve into the specifics of a chosen team. We can explore their performance throughout a specific year, including their position in the points table and how they've fared historically (depending on the program's features).

The Power of Organization:

Data analysis thrives in organizations. Our program utilizes the power of various data structures and algorithms, like merge sort, quick sort, and bucket sort. These algorithms efficiently arrange the data, making it easier to rank players, compare teams, and identify trends.

A Touch of Fun:

Cricket is all about entertainment, so why not add a dash of fun to our analysis? We've incorporated a "surprise game" module. This section analyzes historical data to identify teams that consistently finish in the top 3 positions. Armed with this knowledge, we can challenge the computer (or a friend!) to a "surprise game" where each team selection is based on historical performance—a unique way to test our IPL knowledge!

This IPL data analysis project is more than just numbers on a screen. It's a testament to the power of data structures and algorithms in unlocking the secrets of the game we love. By combining exploration, analysis, and a touch of fun, we've gained deeper insights into the captivating world of the IPL.

Overview of Our Project (Technical Overview):-

This is well-structured C++ code for an IPL data analysis program. Here's a breakdown of the functionalities implemented in the code you provided:

1. Display Functions:

displayMainScreen: This function creates the welcome screen with the IPL logo and prompts the user for input to enter the program.

dashboard: This function displays the main menu with options for IPL data analysis, team hub, fantasy team, heroes hub, surprise game, and algorithm analysis. It takes user input and returns the chosen option.

2. Data Analysis Functions:

iplDataAnalysis: This function prompts the user for the year of data they want to analyze and returns the chosen year as an integer.

yearData: This function retrieves data based on the chosen year and performs some initial processing.

batBowlPoints: This function calculates batting and bowling points based on the processed data.

display: This function displays the calculated batting and bowling points for the chosen year.

mostRuns: This function identifies players with the most runs in a particular year.

mostWickets: This function identifies players with the most wickets in a particular year.

pointsTableSummary: This function provides a summary of the points table for a particular year.

strikeRate: This function calculates the strike rates of batsmen for a particular year.

economicalFigures: This function calculates the bowling economy of bowlers for a particular year.

top4PointsTable: This function displays the top 4 teams from the points table for a particular year.

3. Team and Player Functions:

teamHub: This function prompts the user for a team name or displays a list of teams and returns the chosen team name.

getTeamDetails: This function retrieves and displays detailed information about a chosen team for a particular year.

heroesHub: This function allows users to explore player data based on batting or bowling.

getPlayerDetailsBatting: This function retrieves and displays batting statistics for a particular player.

heroesHubField: This function prompts the user to choose between batting or bowling data for player exploration.

getPlayerDetailsBowling: This function retrieves and displays bowling statistics for a particular player.

4. Surprise Game Functions:

top3Teams: This function reads data from a CSV file (points_table.csv) and identifies the top 3 teams for each season.

calculateTeamCounts: This function calculates the number of times each team appeared in the top 3 across all seasons in the data.

chooseComputerTeam: This function randomly selects a team from the available teams based on their top 3 appearance counts.

getUserTeamChoice: This function displays available teams with their counts and prompts the user to choose a team.

5. Sorting Algorithms:

merge: This function implements the merge step of the merge sort algorithm to merge two sorted sub-arrays.

mergesort: This function implements the merge sort algorithm recursively to sort a vector of data.

economyMerge: similar to merge, used for sorting based on bowling economy.

economyMergesort: Similar to mergesort, used for sorting based on bowling economy.

partition: This function implements the partition step of the quick sort algorithm to divide an array based on a pivot element.

quickSort: This function implements the quick sort algorithm recursively to sort a vector of data.

bucketSort: This function implements the bucket sort algorithm to sort data by distributing elements into buckets based on a chosen index value.

algorithms: This function prompts the user to choose a sorting algorithm (merge sort, quick sort, or bucket sort).

mergeSorting: This function calls the mergesort function to perform merge sorting on the data.

quickSorting: This function calls the quicksort function to perform quick sorting on the data.

bucketSorting: This function calls the bucketSort function to perform bucket sorting on the data.

Overall, this C++ program provides a well-structured framework for IPL data analysis with various functionalities. The commented sections indicate parts of the code not shown, likely containing the data processing and display logic for different analysis options.

Team meeting 1st on :- Apr 10, 2024 9:30 PM

After deciding on our project's key areas, we planned out six main things to work on. These are:

1. **IPL Data Analysis:** This involves digging deep into IPL data to find out interesting trends and patterns.

2. **Team Hub:** We're creating a central place where people can find all the important information about IPL teams.

3. **Fantasy Team:** Our aim is to help Dream11 players make better fantasy teams by using data analysis.

4. **Heroes Hub:** This is like a central hub to know the stats of your favourite hero in IPL.

5. **Surprise Game:** We want to find some hidden surprises or unexpected things in the data.

6. **Different Algorithm Analysis:** We'll be trying out different methods and algorithms to see which works best for our project.

Regarding the datasets, we're using IPL data covering seasons from 2008 to 2023. This data includes details about batting, bowling, and point tables. Each dataset has 15667 rows and 12 columns, but we noticed there were some missing values. So, our first task was to clean up the data and fill in these missing values. We also separated the data into batting, bowling, and points table categories for easier analysis.

- **Link to our dataset taken from Kaggle:-**
  Ipl Dataset of Kaggle

WRITTEN DESCRIPTION OF OUR PROJECT (FIRST BRAINSTORMING PROCESS):-

1. IPL Dream Team Analysis Dashboard - Written Description
Welcome Screen:

The application starts with a warm welcome message: "Welcome to the IPL Dashboard!"

2. Exploration Menu:

A menu is presented offering various exploration options:

Datasets: Users can delve into specific datasets related to IPL. This could involve player statistics, historical match data, or team information. Datasets will be represented by letters (a, b, c, d, etc.).

3. Subsets: Users can further explore subsets within a chosen dataset. This might allow for filtering data by specific criteria, like a particular season or team performance in a specific venue. Subsets will be denoted using the dataset letter followed by a period and a number (e.g., a.1, a.2, a.3).

4. Prediction Hub: This section will be dedicated to analyzing data and predicting future player performances using various algorithms.

5. Fantasy Team Selection: This is where users can create their dream team for Dream 11. The application will provide access to player data and functionalities to build a team within budget constraints and based on player positions.

6. Dream Comparisons: This exciting feature will allow users to compare their Dream 11 team with others, potentially providing insights into potential improvements.

7. Surprise Game: This section aims to identify potential breakout players who might exceed expectations in an upcoming match.

8. User Interaction:

Users navigate the menu by entering their choices. They can choose a specific dataset letter, explore subsets within a dataset, or navigate to dedicated sections for predictions, fantasy team selection, dream comparisons, or the surprise game feature. Additionally, users can always go back to the main menu or return home (considered the same action).

9. Data Display:

When a user selects a dataset or explores a subset, the application will display relevant information. This might involve a summary or a preview of the data, potentially using tables or charts for better visualization.

10. Future Enhancements:

This is just the initial framework. We plan to further develop each section with specific functionalities. Error handling, a user-friendly interface, and interactive data visualizations will be incorporated as we progress.

NEW INSIGHT

Data Refining - A Valuable Lesson Learned

- One of the key insights gained during our data analysis project was the importance of data refining. We encountered a challenge with our initial dataset - it contained a significant number of empty values. These missing values could have introduced errors and skewed our analysis.

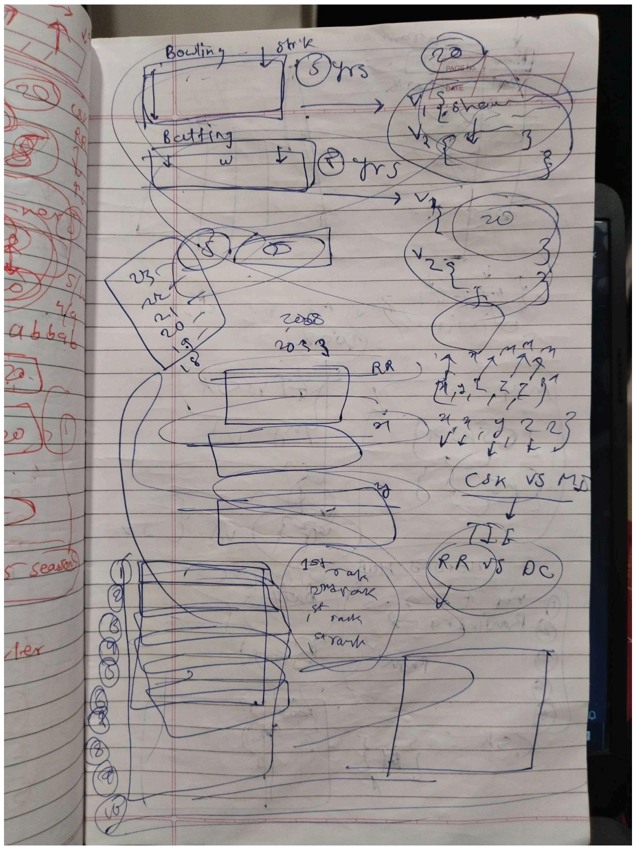
- To address this, we embarked on a data refining process. We learned how to effectively edit the CSV file using a spreadsheet program like Excel. By implementing formulas, we were able to replace the empty values with zeros. This ensured a cleaner and more consistent dataset, ultimately leading to more reliable results in our analysis.

- This experience highlighted the crucial role of data preparation in any data analysis project. Clean and well-structured data is essential for drawing accurate conclusions and deriving meaningful insights. We can consider this a valuable addition to our data analysis toolkit - the ability to identify and address data quality issues.

Team meeting 2nd on :- Apr 11, 2024 9:15 PM

Phase 1:-

Code Lines = 1 to 474



Low fidelity diagram (phase 1)

1. Project Kickoff:

- The project likely began with a kickoff meeting or discussion where our team defined the scope and objectives of the IPL dashboard or analysis tool.
- We identified the need for functionalities like data analysis, team management, fantasy team creation, and algorithm analysis.

2. Requirement Gathering:

- We gathered detailed requirements for each functionality, including user interactions, data sources, sorting algorithms to implement, and output formats.
- For example, we identified that users should be able to navigate through the program using a dashboard interface and choose different functionalities to explore.

3. Designing Functionality:

- Based on the requirements, we designed the functionalities and the overall flow of the program.
- We outlined the main functions needed for user interaction, data analysis, team management, and algorithm analysis.

4. Choosing Data Structures:

- We selected appropriate data structures to represent and manipulate the data.
- Since we're dealing with tabular data (e.g., IPL statistics), we chose vectors of vectors (`std::vector<std::vector<std::string>>`) to store the data rows.

5. Sorting Algorithms Selection:

- We decided to include multiple sorting algorithms to analyze their performance on the data.
- We chose merge sort, quick sort, and bucket sort for their efficiency and suitability for different scenarios.

6. Coding Iteration:

- We iteratively implemented and tested each functionality and sorting algorithm.
- Starting with basic functionality like displaying the main screen and dashboard, we gradually added more features.

7. Error Handling:

- We implemented error handling mechanisms to deal with invalid inputs or unexpected situations.
- For instance, we added try-catch blocks for parsing errors while converting string data to numerical values.

8. Integration and Flow:

- We integrated the individual functions into the main flow of the program, ensuring smooth navigation and proper data processing.
- We followed a modular approach, where each function performs a specific task, enhancing code readability and maintainability.

9. Testing:

- We conducted thorough testing of the code, including unit tests for individual functions and integration tests for the entire program flow.
- We tested various scenarios, including edge cases and boundary conditions, to ensure the robustness of the code.

10. Documentation and Comments:

- Throughout the development process, we documented the code and added comments to explain complex logic or algorithms.
- We aimed for clear and concise documentation to facilitate understanding and future maintenance of the codebase.

By following this approach, we developed a comprehensive IPL dashboard and analysis tool that meets the project requirements and provides valuable insights into IPL statistics and performance analysis.

NEW INSIGHT

Data Refining - A Valuable Lesson Learned

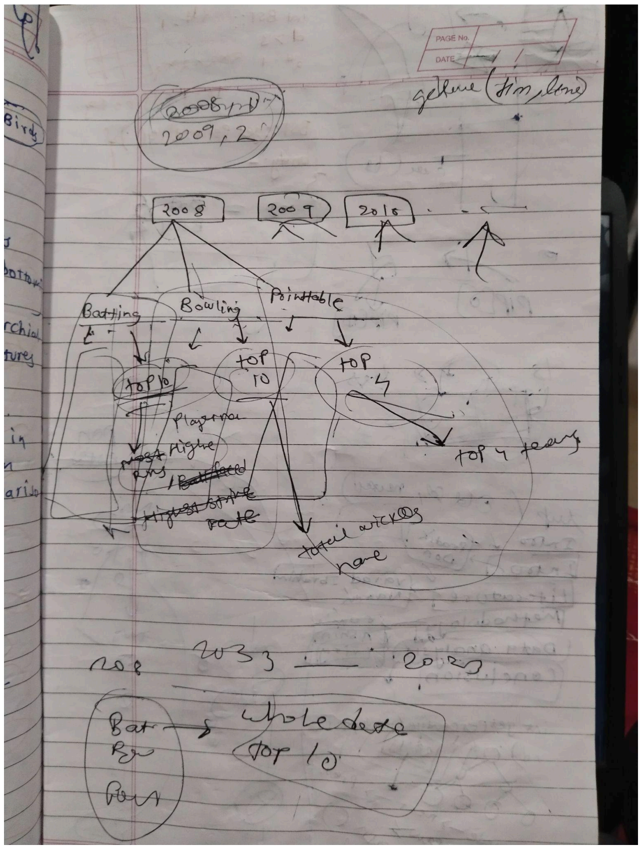
- **Handling Conversion Errors:** Through today's coding process, we encountered the concepts of STOI (String to Integer) and STOF (String to Float) errors. These errors occur when attempting to convert string data to numerical values, and they can lead to runtime issues if not handled properly. Learning how to effectively manage these errors ensures robustness in data processing and enhances the reliability of our code.

- **CSV File Reading in C++:** Today, we gained valuable experience in reading CSV (Comma-Separated Values) files within the C++ programming language. Understanding the intricacies of parsing CSV data allows us to efficiently extract information from external sources and integrate it into our applications. This skill broadens our toolkit for handling diverse datasets and strengthens our capabilities in data-driven development.

Team meeting 3rd on :- Apr 12, 2024 10:00 PM

Phase 2:-

Code Lines = 474 to 991



Low fidelity diagram (phase 2)

1. Project Kickoff:
- The project began with a kickoff meeting where our team discussed the objectives and scope of the IPL Analysis Tool.
- Key functionalities such as data sorting, team selection, and statistical analysis were identified during this phase.

2. Requirement Gathering:
- Detailed requirements were gathered for each functionality, including user interactions and data processing.
- Use cases were defined to ensure that the tool meets the needs of both casual users and data enthusiasts.

3. Designing Functionality:
- Based on the requirements, the team designed the structure and flow of the software.
- Modular design principles were followed to ensure scalability and maintainability.

4. Choosing Data Structures:
- Suitable data structures were selected to represent and manipulate the IPL data effectively.
- Vectors of vectors were chosen to store tabular data, providing flexibility for sorting and analysis.

5. Sorting Algorithm Selection:
- Multiple sorting algorithms were chosen for comparison, including insertion sort, merge sort, quick sort, and bucket sort.
- Each algorithm was implemented as a separate function to facilitate easy comparison.

6. Coding Iteration:
- The coding process involved iterative development and testing of each functionality.
- Error handling mechanisms were implemented to deal with invalid inputs and unexpected situations.

7. Integration and Flow:

- The individual functions were integrated into the main flow of the program.
- A user-friendly interface was designed to enable easy navigation and interaction.

8. Testing:

- Comprehensive testing was conducted to ensure the correctness and robustness of the code.
- Unit tests and integration tests were performed to validate the functionality of each component.

9. Documentation and Comments:

- The code was thoroughly documented, with comments added to explain complex logic and algorithms.
- Clear and concise documentation was provided to aid in understanding and maintaining the codebase.

NEW INSIGHT

Data Refining - A Valuable Lesson Learned

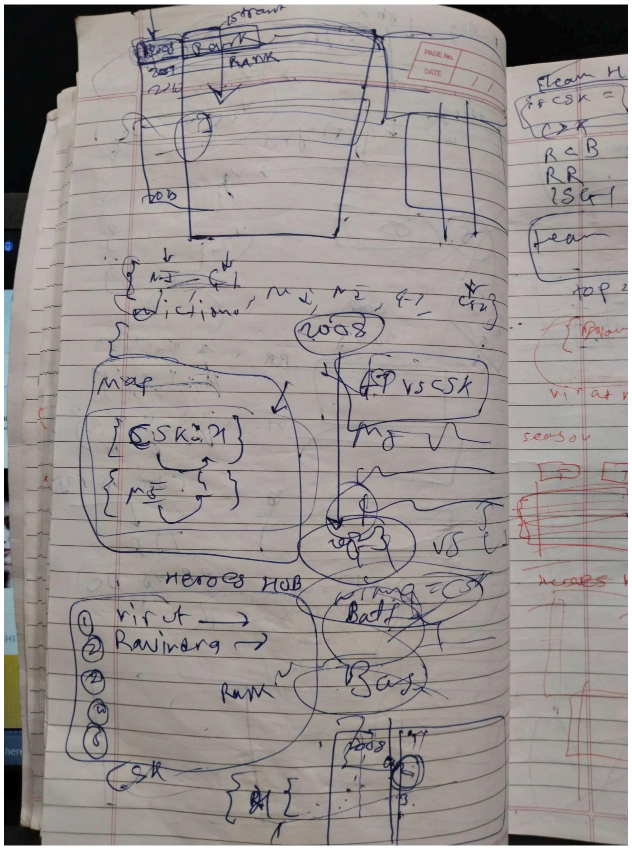
- **Utilizing Maps in C++:** Through today's coding process, we gained a deep understanding of how to effectively use maps in C++. Maps are powerful data structures that allow us to store key-value pairs, providing efficient lookup and retrieval of values based on their associated keys. This knowledge enables us to leverage maps for various applications, such as organizing data and implementing algorithms efficiently.

- **Exploring Map Functions:** Today, we delved into the different functions available for manipulating maps in C++. We learned about functions like insert(), erase(), find(), and at(), among others, which offer versatile capabilities for adding, removing, and accessing elements within a map. Understanding these functions equips us with the tools necessary to manipulate map data structures effectively in our coding projects.

Team meeting 4th on :- Apr 13, 2024 9:45 PM

Phase 3:-

Code Lines = 991 to 1421



Low fidelity diagram (phase 3)

Initial Conceptualization:

The brainstorming process likely began with identifying the key functionalities of the IPL data analysis system. These functionalities would include:

1. Providing a menu for users to choose between exploring batting statistics, bowling statistics, team information, and year-wise data analysis.

2. Implementing functions to handle user input and display relevant data based on the user's choice.

3. Reading data from CSV files containing batting and bowling statistics for different IPL seasons.

4. Presenting the data in a structured and user-friendly format.

5. Ensuring error handling for invalid user inputs and file operations.

Functionality Breakdown:

The code consists of several functions, each responsible for a specific aspect of the IPL data analysis system:

1. `heroesHubField()`:

- Displays a menu for users to choose between exploring batting or bowling statistics.

- Takes user input and returns the chosen field (batting or bowling).

2. `heroesHub(std::string n)`:

- Based on the user's choice of batting or bowling, displays a list of IPL players for the selected field.

- Prompts the user to choose a player and calls corresponding functions to display their statistics.

3. `getPlayerDetailsBatting(const std::string& playerName)`:

- Retrieves batting statistics for the specified player from a CSV file.
- Displays the player's batting statistics in a tabular format.

4. `getPlayerDetailsBowling(const std::string& playerName)`:

- Retrieves bowling statistics for the specified player from a CSV file.
- Displays the player's bowling statistics in a tabular format.

5. `teamHub()`:

- Displays a menu for users to choose an IPL team for which they want to explore information.
- Takes user input and returns the chosen team.

6. `teamHubYear()` and `iplDataAnalysis()`:

- Displays a menu for users to choose a specific year for IPL data analysis.
- Takes user input and returns the chosen year.

NEW INSIGHT

Data Refining - A Valuable Lesson Learned

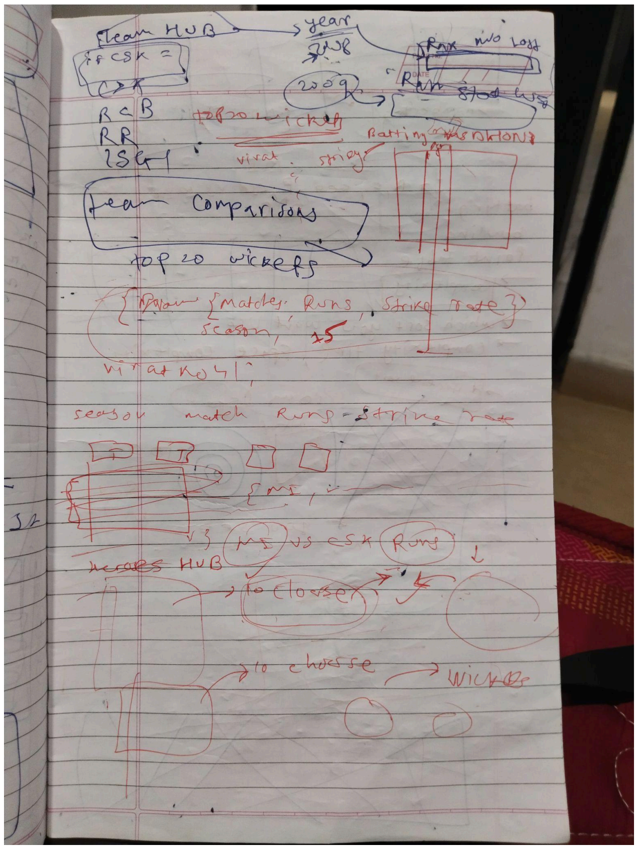
- **Understanding setw in C++:** Through today's coding process, we gained a deeper understanding of how to utilize the setw manipulator in C++. We learned how to apply it to format output, aligning data in columns to enhance readability. This insight empowers us to present data more effectively in our programs, improving user comprehension and overall user experience.

- **Exploring iomanip Library Functions:** Today's coding session also exposed us to the diverse range of functions available in the iomanip library of C++. We delved into functions like setprecision, setfill, and setw, each serving distinct purposes in manipulating output formatting. This newfound knowledge equips us with powerful tools to customize the appearance of our program's output, catering to specific presentation requirements and enhancing visual appeal.

Team meeting 5th on :- Apr 14, 2024 9:00 PM

Phase 4:-

Code Lines = 1422 to 1679



Low fidelity diagram (phase 4)

To create this IPL data analysis and fantasy team code, it seems like the developer followed a structured approach, likely going through the following steps:

1. Requirement Analysis: Before starting the implementation, there was likely a clear understanding of what the application should do. This includes analyzing the requirements, such as data analysis, displaying top performers, a points table summary, and creating a fantasy team.

2. Design Planning: Based on the requirements, a design plan was formulated. This would involve deciding on the overall architecture of the application, what functions are needed, and how they will interact with each other.

3. Functionality Implementation:

- Main Functionality: The core functionality includes analyzing data from CSV files, sorting the data based on different parameters like runs scored, wickets taken, and economy rate, and then displaying the top performers.

- User Interaction: Functions like `yearData`, `batBowlPoints`, and `display` handle user input and interaction. They prompt the user for choices, process the input, and call relevant functions accordingly.

- Data Processing: Functions like `mostRuns`, `mostWickets`, `pointsTableSummary`, `strikeRate`, and `economicalFigures` read data from CSV files, parse them, sort based on specified parameters, and then display the results in a formatted manner.

- Fantasy Team Generation: The `fantasyTeam` function generates a fantasy team based on the top performers in batting and bowling categories.

4. Code Modularity and Reusability: The code seems to be modular, with each function responsible for a specific task. This promotes code reusability and maintainability.

7. Documentation: Adequate comments and documentation were likely added to explain the purpose of each function, parameters, return values, and any important implementation details.

NEW INSIGHT

Data Refining - A Valuable Lesson Learned

- **Understanding Time and Space Complexity:** Through the use of the chrono library in C++, we gained insights into measuring time and space complexity in our algorithms. This understanding helps us assess the efficiency of our code and make informed decisions when optimizing for performance.

- **Exploring Chrono Library Functions:** We delved into the various functions provided by the chrono library, discovering their versatility in measuring time durations, calculating time points, and performing arithmetic operations on time. This exploration expanded our toolkit for time-related operations in C++ programming.

- **Exploring CTime Library:** Additionally, we explored the ctime library in C++, which provides functionality for manipulating time and date values. Understanding the capabilities of this library enhances our ability to work with time-related data and perform tasks such as formatting timestamps or calculating time differences in our programs.

In conclusion, our data analysis project underscored the critical importance of data refining in ensuring the accuracy and reliability of our analysis. By addressing missing values through a meticulous data refining process, we were able to enhance the quality of our dataset and mitigate potential errors that could have compromised our findings. This experience highlighted the significance of data preparation in facilitating robust analysis and deriving meaningful insights.

Moreover, our coding process provided valuable insights into handling conversion errors, reading CSV files, and utilizing maps in C++. We learned how to effectively manage conversion errors, parse CSV data, and manipulate map data structures, expanding our capabilities in data-driven development and enhancing our proficiency in C++ programming.

Exploring libraries such as iomanip, chrono, and ctime further enriched our understanding of formatting output, measuring time and space complexity, and manipulating time-related data in our programs. This exploration broadened our toolkit for implementing various functionalities and optimizing performance in our coding projects.

Overall, the knowledge and skills gained from this project and the coding process have equipped us with valuable tools and techniques for conducting data analysis and developing robust, efficient software solutions. By applying these insights to future projects, we can continue to drive innovation, solve complex problems, and deliver impactful results.

We wanted to take a moment to sincerely thank you for all of your help and support during the course. Thank you for your unwavering support, encouragement, and dedication to your students' success. Your impact extends far beyond the classroom, and we are grateful to have had the opportunity to learn from you.

THANK YOU!