C programming Project

PROJECT REPORT

Project title:-Cricket scoreboard

Submitted By : Aditya
SAP ID: 590025474
Course Title: Programming in C
Course Code: CSEG1032
Semester: 1
Batch:-10

Submitted To : Mr. Rahul Prashad

1. ABSTRACT

This project implements a simple **Cricket Scoreboard System** using the C programming language. The program allows the user to enter details of a cricket team, including team name, number of players, and each player's batting performance. For every player, the runs scored, balls faced, number of fours, sixes, and strike rate are calculated and displayed in a formatted scoreboard.

The system also computes the team's total runs, total wickets lost, and total overs faced based on the total number of balls. It demonstrates the use of **structures**, **arrays**, **loops**, **user input handling**, and **basic arithmetic operations** in C. This project is suitable for beginners who want to understand how to model real-world problems, like sports scoring, using structured programming.

2. OBJECTIVE

The main objective of this project is to design a **console-based cricket scoreboard** application that:

- Accepts **team-level and player-level input** from the user.
- Calculates and displays **individual player statistics**, such as runs, balls faced, number of boundaries, and strike rate.
- Calculates and displays **team totals**, including total runs, total wickets, and overs.
- Demonstrates how to use **structures and arrays** to store related data in C.
- Provides a clear, readable **tabular scoreboard output**.

Through this project, the aim is to apply C programming concepts to a simple sports-based application and improve understanding of data structuring and formatted output.

3. PROBLEM DEFINITION

In many small matches, practice sessions, or school/college-level games, cricket scores are often written manually on paper. This has several limitations:

- Calculating **strike rate** and **overs** manually for each player is time-consuming.
- There is a chance of **calculation mistakes** in totals and averages.
- Data is not presented in a clean, structured format for easy understanding.

Problem:
There is a need for a basic computerized tool that can help students or scorers enter batting data for a team and automatically calculate and display a neat scoreboard with correct statistics.

Solution:
This C program solves the problem by allowing the user to input each player's performance and automatically generating a **complete scoreboard**. It calculates strike rate for each player, total team runs, total wickets, and overs, and then prints all details in a well-organized manner.

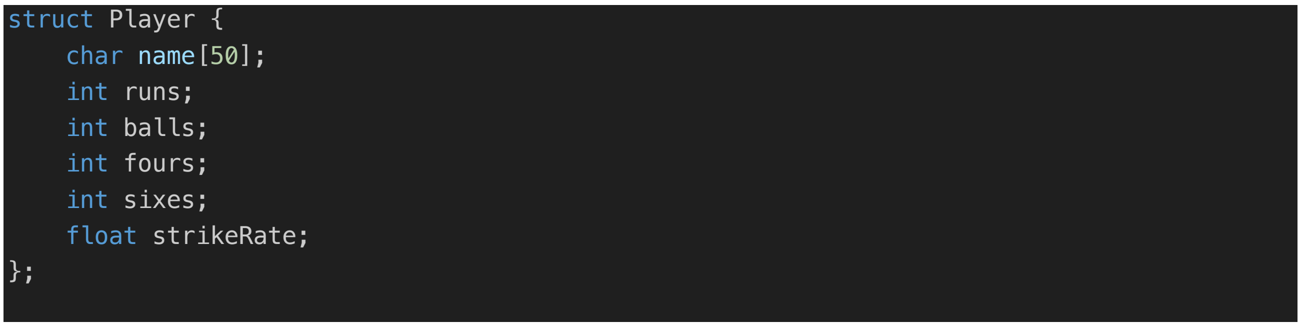
4. SYSTEM DESIGN AND ALGORITHM

Data Structures Used

Two struct types are used:

Player Structure

Code:-



```c
struct Player {
    char name[50];
    int runs;
    int balls;
    int fours;
    int sixes;
    float strikeRate;
};
```

This stores information about each batsman:

- name – player's name
- runs – total runs scored
- balls – balls faced
- fours – number of 4s hit
- sixes – number of 6s hit
- strikeRate – (runs / balls) * 100

Team Structure

Code:-

```
struct Team {
    char teamName[50];
    struct Player players[11];
    int totalRuns;
    int totalWickets;
    float overs;
};
```

This stores information about the whole team:

- teamName – name of the team
- players[11] – array of maximum 11 players
- totalRuns – sum of all players' runs
- totalWickets – wickets lost
- overs – overs played (calculated from total balls)

The system follows a modular architecture as required by the project guidelines:

File Structure:

"src/main.c" controls program flow and user interaction.
"src/scoreboard.c" handles CRUD (Create, Read, Update/Delete) operations.
"src/utils.c" provides helper functions (like clearing the screen).
"include/scoreboard.h" Contains structure definitions and function prototypes.

Algorithm / Program Logic

Step 1: Start the program.
Step 2: Declare a Team variable and other required variables
(n, i, totalBalls).
Step 3: Ask the user to enter the **team name**.
Step 4: Ask the user to enter the **number of players** (maximum 11).
Step 5: Initialize totalRuns = 0, totalWickets = 0, totalBalls = 0.
Step 6: For each player from 1 to n:

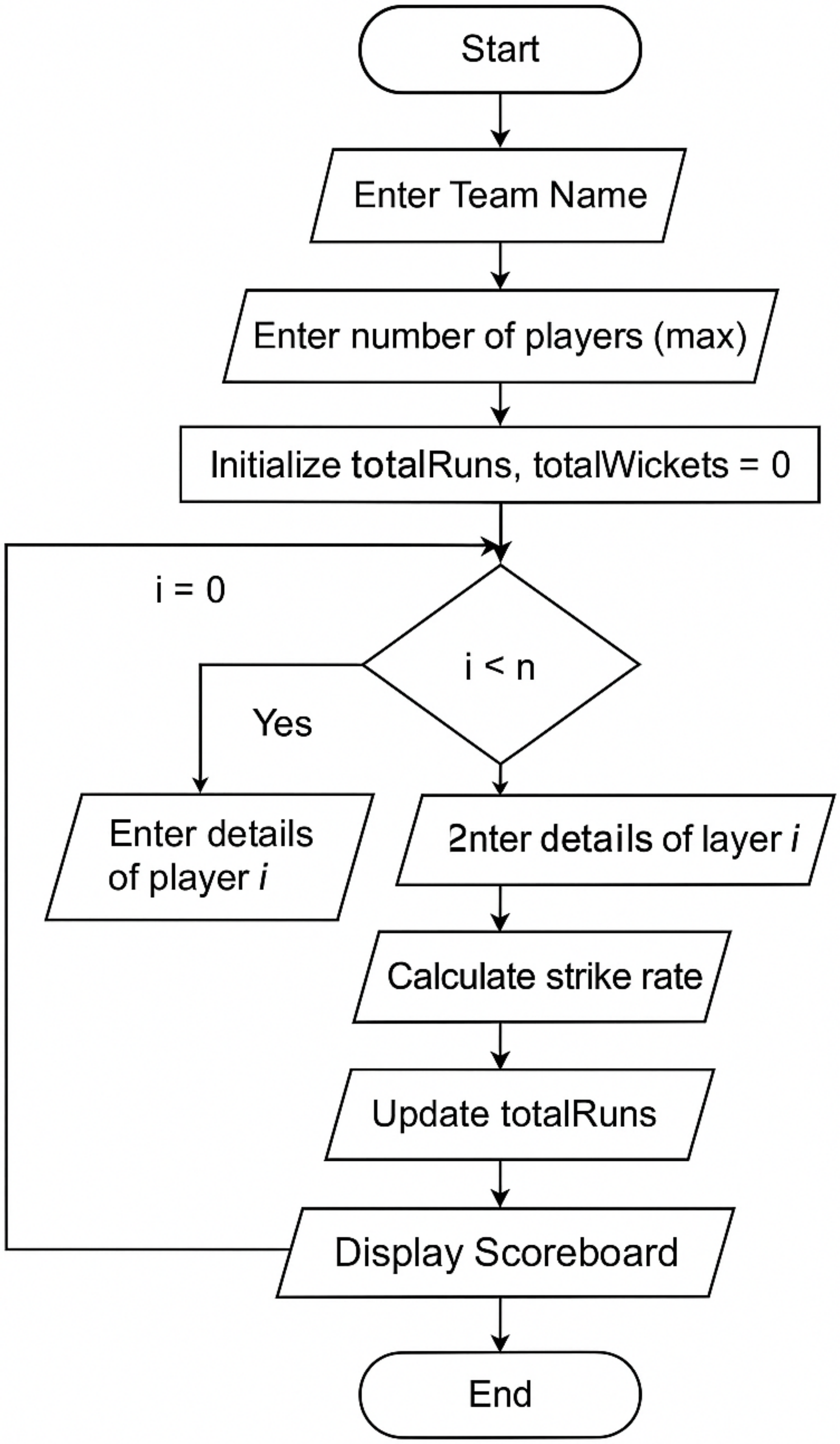
- Read player name.
- Read runs scored.
- Read balls faced.
- Read number of fours.
- Read number of sixes.
- If balls ≠ 0:
    - Calculate strikeRate = (runs / balls) * 100.
    - Else, strikeRate = 0.
- Add player's runs to team.totalRuns.
- Add player's balls to totalBalls.

Step 7: Ask for total wickets lost by the team and store
in team.totalWickets.
Step 8: Compute team.overs = totalBalls / 6.0.
Step 9: Display the formatted scoreboard:

- Team name
- Total score (runs/wickets) and overs
- Table of each player with runs, balls, fours, sixes, strike rate

Step 10: Display credits (Made by Aditya, SAP ID).
Step 11: End program.

Flowchart:-


Start
Enter Team Name
Enter number of players (max)
Initialize totalRuns, totalWickets = 0
i = 0
i < n
Yes
Enter details of player i
2nter details of layer i
Calculate strike rate
Update totalRuns
Display Scoreboard
End

5. IMPLEMENTATION DETAILS

The program is implemented in C using GCC. Structures are used to store player and team data, loops handle repeated user input for multiple players, a conditional block calculates the strike rate to avoid division errors, and formatted output is used to generate a clean scoreboard. The implementation demonstrates structured programming through the use of data models, arithmetic operations, and organized display formatting.

Key Language Features Used:

File Streams: Used FILE * pointers to manage data persistence.

Standard I/O: Used printf and scanf for the user interface.

Header Files: Created scoreboard.h to share definitions across .c files.

6. Source Code

```c
#include <stdio.h>
#include <string.h>
#include "utils.h"

struct Player {
   char name[50];
   int runs;
   int balls;
   int fours;
   int sixes;
   float strikeRate;
};

struct Team {
   char teamName[50];
   struct Player players[11];
   int totalRuns;
   int totalWickets;
   float overs;
};

int main() {
   struct Team team;
```

```c
  int n, i;
  float totalBalls = 0;

  printf("Enter Team Name: ");
  fgets(team.teamName, sizeof(team.teamName), stdin);
  team.teamName[strcspn(team.teamName, "\n")] = 0;

  printf("Enter number of players (max 11): ");
  scanf("%d", &n);

  team.totalRuns = 0;
  team.totalWickets = 0;

  for(i = 0; i < n; i++) {
    printf("\nEnter details of player %d\n", i + 1);

    printf("Name: ");
    scanf("%s", team.players[i].name);

    printf("Runs scored: ");
    scanf("%d", &team.players[i].runs);

    printf("Balls faced: ");
    scanf("%d", &team.players[i].balls);

    printf("Number of 4s: ");
    scanf("%d", &team.players[i].fours);

    printf("Number of 6s: ");
    scanf("%d", &team.players[i].sixes);

    // Calculate strike rate
    if (team.players[i].balls != 0)
      team.players[i].strikeRate = (team.players[i].runs /
(float)team.players[i].balls) * 100;
    else
      team.players[i].strikeRate = 0;

    team.totalRuns += team.players[i].runs;
    totalBalls += team.players[i].balls;
  }

  printf("\nEnter total wickets lost: ");
  scanf("%d", &team.totalWickets);

  team.overs = totalBalls / 6.0;
  printf("\n================================\n");
  printf("    CRICKET SCOREBOARD     \n");
  printf("================================\n");
  printf("Team: %s\n", team.teamName);
  printf("Total Runs: %d/%d in %.1f overs\n", team.totalRuns,
team.totalWickets, team.overs);
  printf("----------------------------\n");

  printf("Player\tRuns\tBalls\t4s\t6s\tSR\n");
  printf("------------------------------------------------\n");
  for(i = 0; i < n; i++) {
    printf("%s\t%d\t%d\t%d\t%d\t%.2f\n",
        team.players[i].name,
        team.players[i].runs,
        team.players[i].balls,
        team.players[i].fours,
        team.players[i].sixes,
        team.players[i].strikeRate);
  }

  printf("------------------------------------------------\n");
  printf("End of Scoreboard.\n");
  printf("made by aditya \n");
  printf("sap id 590025474 \n");


  return 0;
}
```

7. TESTING AND RESULTS

The system was tested with various inputs to ensure stability and correctness.

SYNTAX:-

Team Name: INDIA
Number of Players: 3

Player 1:
Name: Aditya
Runs: 45
Balls: 32
4s: 6

```
6s: 1

Player 2:
Name: Ankit
Runs: 60
Balls: 50
4s: 5
6s: 2

Player 3:
Name: Yash
Runs: 30
Balls: 20
4s: 3
6s: 1

Total Wickets Lost: 1
```

OUTPUT:-

```
================================
   CRICKET SCOREBOARD
================================
Team: india
Total Runs: 135/3 in 17.0 overs
---------------------------------------------
Player  Runs    Balls   4s     6s     SR
---------------------------------------------
aditya  45      32      6      1      140.62
ankit   60      50      5      2      120.00
yash    30      20      3      1      150.00
---------------------------------------------
End of Scoreboard.
made by aditya
sap id 590025474
```

Testing

The program was tested with:

- Different numbers of players (from 1 to 11).
- Cases where a player has **0 balls** (strike rate becomes 0 safely).

- Different wicket values.
- Different team names and player names.

The program behaves as expected and prints the correct totals and strike rates.

8. CONCLUSION AND FUTURE SCOPE

Cricket Scoreboard System developed in C successfully collects and displays detailed batting statistics for a cricket team. Using structures and arrays, it organizes the data of multiple players and provides a clear, formatted scoreboard that includes total runs, wickets, overs, and individual performance metrics like strike rate.

The project meets its objectives of applying C programming concepts to a real-world-style application, improving understanding of data structuring, loops, conditional logic, and formatted output.

Future Scope:

The current version of the program can be further enhanced in several ways:

- **Support for Two Teams:** Extend the program to handle both Team A and Team B, and display match results (who won, by how many runs/wickets).
- **Bowling Statistics:** Add another structure for bowlers to store overs bowled, runs conceded, wickets taken, and economy rate.
- **File Storage:** Use file handling to save scoreboards to a text file so that match records can be stored and viewed later.
- **Input Validation:** Add checks to ensure that runs, balls, and wickets are within valid ranges.
- **User-Friendly Menu:** Convert the program into a fully menu-driven system with options like “Enter Team Details”, “Show Scoreboard”, “Exit”.
- **Graphical Interface:** In the future, the logic can be reused in a graphical application for better presentation.

9. REFERENCES

Kernighan, B. W., & Ritchie, D. M. The C Programming Language.

Lecture Notes

Standard C Library Documentation: cppreference.com