Campionato calcistico - Progetto num. 6

Iacoangeli Francesco, Howladar Rafsan, Orti Nicolò

Descrizione progetto:

Il nostro obiettivo era di:
Gestire delle squadre con degli attributi stabiliti (nome, punti, gol fatti , gol subiti). L'utente dovrà inserire da tastiera i risultati delle partite generate dal programma e alla fine il programma dovrà trasferire la classifica ordinata per punti, e in caso di pareggio calcolare la differenza reti, in un file di tipo .txt come richiesto dalle estensioni.

Funzionamento del codice:

1. All'avvio del codice chiederà di inserire prima di tutto il numero delle squadre che parteciperanno al campionato. (**Attenzione!!**, se il numero che verrà inserito è **minore di 2** o un **numero dispari**, il programma darà un messaggio d'errore e l'**arresto immediato** del programma)

2. In seguito verrà chiesto di inserire il numero delle giornate che comporranno il campionato. (**Attenzione!!**, se il **numero** sarà **minore di 1** il **programma si arresterà** nell'immediato, lasciando un messaggio d'errore).

3. Se il programma non darà errore, si può incominciare ad inserire manualmente i risultati di ogni partita che genererà il codice: Prima si dovrà inserire il numero di gol della squadra di casa (il programma lo indicherà affianco), e in seguito chiederà il numero di gol della squadra ospite. (**Attenzione!!**, se verranno inseriti dei **valori** riguardanti i gol **inferiori a 0** il programma rilascerà un messaggio d'errore e il **programma si arresterà**).

4. Arrivati alla fine se verrà inserito **tutto correttamente**, il programma rilascerà un messaggio con scritto **"Dati caricati"**. Verrà creato sul desktop un file di tipo **dati.txt** dove verrà **riportata la classifica finale** con il vincitore in prima posizione, considerando i punti e in caso la differenza reti.

Struttura dei dati utilizzati:

Nel nostro codice abbiamo sviluppato le seguenti strutture dati:

- **Struct**: struct Squadra la struttura principale del nostro programma. Serve per la formazione di una squadra con all' interno i requisiti richiesti:
    - Nomi
    - Punti
    - Goal subiti (GS)
    - Goal Fatti (GF)

- **Array utilizzati**:
    - Squadra arr[n_squadre]; => Contiene tutte le squadre che parteciperanno al campionato. Utilizzato per mescolare, calcolare i risultati, ordinare e stampare la classifica finale

- **Variabili chiave:**
    - n_squadre => La variabile (inserita dall'utente) che passerà la dimensione all' array (Squadra arr[n_squadre]) per il numero di squadre inserito da tastiera.
    - n_giornate => La variabile (inserita dall'utente) che passerà alla maggior parte dei cicli for, il numero delle giornate.

Descrizione delle funzioni principali:

Nel nostro codice abbiamo provato a modulare il più possibile creando 7 tipi di funzioni. Ognuna di loro svolge un ruolo fondamentale per il funzionamento corretto del programma, e sono:

- void mescola_squadre => Funzione che mescola casualmente le squadre nell'array per evitare sempre le stesse partite.

- void selection_sort => Funziona che riordina in ordine decrescente, (per mettere la squadra con più punti in alto) per stilare una prima classifica.

- void verifica_input => Funzione che controlla che i numeri inseriti siano validi su:
    - Numero di squadre maggiore di 1;
    - Numero di giornate maggiore o uguale a 0;
    - I gol non devono essere negativi;

- void differenza_reti => Funzione che in caso di parità dei punti riordina la classifica usando la differenza reti (GF-GS).

- void input => Funzione che gestisce l'inserimento dei risultati delle partite per ogni giornata e aggiorna punteggi, gol fatti/subiti.

Scelte implementative e difficoltà affrontate:

Scelte implementative:

- Abbiamo scelto di utilizzare una struttura dati ("struct Squadra") per **raggruppare** tutte **le informazioni** relative a ciascuna squadra. Questo ci ha permesso di gestire in modo ordinato e coerente i dati durante tutto il campionato.

- Per ordinare le squadre in classifica abbiamo implementato un algoritmo di **selection sort**, così da poter controllare passo per passo l'ordinamento in base ai punti. In caso di parità, abbiamo introdotto una funzione specifica per confrontare la **differenza reti** (gol fatti - gol subiti).

- Per gestire correttamente **l'inserimento dei dati** da parte dell'utente, abbiamo creato la funzione **"verifica_input"**, che ci permette di **bloccare il programma** in caso di inserimenti non validi (es. numero di squadre dispari, giornate non positive, gol negativi).

Difficoltà incontrate:

- Una delle difficoltà iniziali è stata evitare che le stesse partite si ripetessero ogni giornata. Questo è stato risolto con la funzione "mescola_squadre", che rimescola casualmente l'ordine delle squadre prima di ogni giornata.

- abbiamo affrontato problemi con la scrittura su file, risolti controllando che il file fosse correttamente aperto prima di procedere alla scrittura della classifica finale.

Test effettuati:

1.Test:

Input

N. squadre 2
N. giornate 1
Nomi: Squadra a , Squadra b;
Goal: 0 - 2

Output atteso

----CLASSIFICA FINALE----

Squadra: b | Punti: 3 | GF: 2 | GS: 0 | DR: 2

Squadra: a | Punti: 0 | GF: 0 | GS: 2 | DR: -2

2.Test:

Input

N. squadre 3

Output atteso

Errore: non si può fare una partita con 3 squadre.

3.Test

Input

N. squadre 2
N. giornate -1

Output atteso

Errore: non si può fare una partita con -1 giornate.

4.Test

Input

N. squadre 2
N. giornate 1
Nomi: Squadra a , Squadra b;
Goal: 0 - -3

Output atteso

Errore: Una squadra non può avere -3 Gol.

Estensioni realizzate:

Come richiesto abbiamo inserito dentro il codice l'intenzione di trasferire i dati in un file esterno di tipo txt. Abbiamo utilizzato la seguente funzione:

- void carica classifica

Il suo ruolo è di caricare la classifica finale ordinata per punti e differenza reti nel file "dati.txt".
Per rendere fattibile questo abbiamo inserito la libreria "fstream", per creare un file di tipo txt abbiamo usato std::ofstream outFile("dati.txt").

Conclusione e riflessione:

Questo progetto ci è servito per iniziare a capire come potrebbe funzionare un ipotetico lavoro in futuro e come affrontare le difficoltà in gruppo. In questo progetto abbiamo imparato principalmente come mescolare in modo casuale un array, come trasferire dei dati in un file esterno, trovare dei bug che l'utente potrebbe inserire e come modulare e renderlo il più leggibile a tutti. Ci siamo molto divertiti a realizzarlo, perché abbiamo preso il compito come una sfida da portare al termine, in più la scoperta di nuove librerie e variabili che non abbiamo mai utilizzato che ci porteremo da qui in avanti se un giorno decidessimo di frequentare un università.