**1. salvataggio\_punteggio/**

La cartella presentata sviluppa e gestisce la classifica dei giocatori. Permette di inserire e ordinare i dati relativi all’interno di un file di testo(“leaderboard.txt”), Di seguito, vengono descritte in dettaglio le parti più importanti del codice.

**2. Funzione write**

void write(char h[], char min[], char s[], char point[], char name[], char line[]){...}

Indubbiamente questa è la funzione principale (oltre a insert che tratteremo in seguito) per l'inserimento e l'aggiornamento dei dati nella classifica. Ha come parametri formali gli array di char che riguardano i dati relativi al giocatore. L’idea alla base per il salvataggio del giocatore nella classifica è:

utilizzo due file:

* **leaderboard.txt**: file in cui sono salvati i giocatori e ordinati in base al punteggio.
* **tmp.txt**: file temporaneo che utilizzo in appoggio (che alla fine della funzione elimino) a leaderboard.txt per ordinare e i giocatori tra loro.

Dopo aver formattato correttamente il tempo e recuperato i dati, creo il file tmp.txt, lo apro in scrittura per ricopiare i dati di leaderboard.txt che invece apro in lettura e successivamente inserisco nell’ordine corretto il nuovo giocatore (non c’è bisogno di pulire tmp.txt in quanto andrò sempre ad eliminarlo alla fine, l’eliminazione è finalizzata per avere solo il file della classifica nella cartella). Infine inseriti e ricopiati con gli opportuni aggiornamenti della posizione tutti i giocatori nel file tmp.txt, svuoto interamente leaderboard.txt e ricopio la classifica aggiornata al suo interno.

Il file leaderboard.txt però hanno al loro interno due particolarità entrambe sono molto utili per la parte che riguarda la lettura della classifica, ovvero la prima riga rappresenta un flag, il quale indica lo stato del file:

* new: il file è appena stato creato;
* tt: il file ha subito modifiche;
* ff: il file non ha subito modifiche

l’altra invece indica la riga dell’ultimo giocatore inserito (nel caso lo stato sia new il valore è -1 di default).

Nel while posso avere 3 casi:

1. il giocatore che devo inserire ha un punteggio minore rispetto al giocatore che sto confrontando ora (sarebbe la riga che sto leggendo dalla classifica), quindi la stampo nel file temporaneo.
2. Il giocatore che devo inserire ha un punteggio maggiore quindi lo inserisco, attivo il flag = true perché mi aggiornerà tutte le posizioni dei giocatori nella classifica e dopo non mi entrerà più nell’if perché è già stato inserito.
3. Il giocatore inserito ha un punteggio uguale, li inserisco entrambi senza aggiornare la posizione,attivo il flagI = true per inserire i dati senza aggiornare la posizione.

Is\_insert serve per evidenziare i dati appena inseriti quando andrà a leggere la classifica. Il funzionamento è semplice, fintanto che è falso conta altrimenti no.

else if(stoi(copy) <= stoi(point) && !flag){

      if(stoi(copy)==stoi(point)){

if(!is\_insert) count++;

         . . . . . . . .

 flagI = true;

           is\_insert = true;

           insert\_line(newFile, pos\_prec, pc, name, time, point, line);/\*Caso 3\*/

           insert\_line(newFile, pos\_prec, pc, Nm, Tim, copy, B);

       }

       else{

          flag = true;

            is\_insert = true;

            insert\_line(newFile, pos\_prec, pc, name, time, point, line); /\*Caso 2\*/

            pos\_prec++;

            insert\_line(newFile, pos\_prec, pc, Nm, Tim, copy, B);

         }

       }

       else{         /\* Caso 1

      insert\_line(newFile, pos\_prec, pc, Nm, Tim, copy, B); \*/

           }

**3. Funzione insert\_line**

void insert\_line(fstream &newFile, int pos\_prec, char pc[], char Nm[], char Tim[], char copy[], char B[]) { ... }

Questa funzione si occupa di scrivere una nuova linea nel file newFile, formattando vari dati ( posizione, il nome del giocatore, il tempo di gioco, i punti e le righe completate) in una stringa chiamata mess. La funzione viene utilizzata per aggiungere o aggiornare le informazioni di un giocatore nella classifica.

**5. Funzione del\_ch**

void del\_ch(WINDOW\* player, char text[], int i, int limit) { ... }

Questa funzione gestisce la cancellazione di caratteri nella finestra del terminale durante l'inserimento del nome del giocatore.

Nella funzione si passa anche la dimensione del bordo della finestra, perché quando elimino un carattere mi si “spostano” gli altri verso sinistra, per risolvere ciò, ridisegno il bordo con box(), nel mentre stampo tra il testo e il bordo(quindi la dimensione massima della finestra) degli spazi vuoti per eliminare caratteri indesiderati.

void del\_ch(WINDOW\* player, char text[], int i, int limit){

    text[i] = '\0';

    if(i > -1){

        mvwdelch(player, 13, i + 2);

        mvwprintw(player, 13, 2, "%-\*s", limit, text);

        box(player, 0, 0);

        wrefresh(player);

    }

}

**6. Funzione check**

bool check(char text[]) { ... }

Questa funzione controlla se nella stringa text (che rappresenta il nome del giocatore) sono presenti spazi vuoti (char(32)). Se trova un carattere non valido, restituisce true. Questa funzione serve a verificare se il nome è stato inserito o se esistono degli spazi.

**7. Funzione insert**

void insert(int p, int h, int min, int s, int row) { ... }

Questa funzione gestisce l'inserimento del nome del giocatore e dei relativi dati (tempo, punteggio, righe completate). Le principali operazioni svolte dalla funzione sono:

* **Creazione della Finestra**: Viene creata una finestra nel terminale dove l'utente può inserire (con il controllo sugli spazi e alcuni caratteri non ammessi) il proprio nome
* **Gestione dell'Input**: La funzione gestisce l'input dell'utente, permettendo di inserire il nome, cancellare caratteri e confermare l'inserimento.
* **Aggiornamento della Classifica**: Una volta inserito il nome, si chiama la funzione write per aggiornare la classifica con i nuovi dati.

### **8. Class\_menu.cpp**

Il codice è organizzato in diverse classi, ciascuna responsabile di una parte specifica dell'interfaccia del gioco: il menu principale, il menu di pausa, e la gestione della classifica ("leaderboard"). Tale codice implementa il funzionamento delle principali classi usate per il gioco.

1. Classe menu:

La classe menu è la classe principale responsabile della creazione e la gestione delle finestre di menu. Questa classe include metodi per inizializzare lo schermo (init), creare una finestra (create\_W), e gestire il menu (menu\_start e menu\_S).

2. Class:Read\_menu:

È una classe derivata della classe menu. Gestisce la visualizzazione delle informazioni dei giocatori.

La funzione tenta di aprire il file leaderboard.txt che contiene la classifica salvata, In caso di errore nell'apertura o nella creazione del file, viene gestita l'uscita dal programma:

fstream file;

    file.open("salvataggio\_punteggio/leaderboard.txt", ios::in); //mi apre il file in lettura

    char line[80];

    if(!file.is\_open()){ // is file doesn't exits i do an interrupt

        not\_exist = true;

        ofstream f("salvataggio\_punteggio/leaderboard.txt");

            f<<"new"<<endl;

        f.close();

        file.open("salvataggio\_punteggio/leaderboard.txt", ios::in);

        if(!file.is\_open())

        {

            cout<<"errore, file salvataggio non creato "<<endl;

            endwin();

        }

    }

La funzione legge il file riga per riga e gestisce diversi casi, a seconda del contenuto,se il file incontra un flag inizializza gli spazi all’interno della finestra,invece se incontra un “n”, va ad incremenetare le righe (in questo caso i). La classifica può avere un massimo di 30 giocatori salvati quindi dopo i primi 30 giocatori in classifica non stampa più ma continua a leggere il file.

    while(!file.eof()){          //è vero fin tanto che il file non sia finito

        file>>line;

        if(strcmp(line, "ff") ==0){

            c = 10;

            r = 31;

            i = 0;

            file>>line;

            pos\_h = stoi(line);

            fix = false;

        }

        else if(strcmp(line, "tt") == 0){

            c = 10;

            r = 31;

            i = 0;

            file>>line;

            pos\_h = stoi(line);

            fix = true;

        }

        else if(strcmp(line,"new")==0)

        {

            c = 10;

            r = 31;

            i = 0;

            fix = false;

        }

        else if(strcmp(line, "n") == 0){

            c++;

            r = 31;

            i++;

        }

        else{

            if(i<30){

            if(i == pos\_h-1){

                    wattron(win\_rank, A\_REVERSE);

                    mvwprintw(win\_rank, c, r, "%s", line);

                    wattroff(win\_rank, A\_REVERSE);

                }

                else mvwprintw(win\_rank, c, r, "%s", line);

                r = r + 20;

            }

        }

    }

    file.close();//chiude il file

Se il file è stato modificato, lo stato viene aggiornato a "ff" (file non modificato) per le future letture:

if(fix){         // se vero allora nella prima riga c'è tt quindi il file in precendenza è stato modificato

        fstream fileR;

        fileR.open("salvataggio\_punteggio/leaderboard.txt", std::ios::in | std::ios::out);

        fileR<<"ff"<<endl;

        //file>>endl;

        fileR.close();

        fix = false;

    }

### **9.Main.cpp**

È il file iniziale del gioco non a caso gestisce e implementa il menu principale che permette all'utente di scegliere tra diverse opzioni: avviare un gioco, visualizzare una classifica, modificare alcune impostazioni e uscire dall'applicazione.

Il menu è gestito da un ciclo principale che attende l'input dell'utente. Le frecce direzionali vengono utilizzate per navigare tra le opzioni del menu, mentre il tasto Invio conferma la selezione. Quando un'opzione viene selezionata, il programma esegue l'azione corrispondente: ad esempio, avviare il gioco richiede la funzione play(), che viene eseguita quando si seleziona l'opzione "Play".

Dopo ogni azione, il menu viene ricreato e l'interfaccia utente viene aggiornata. Il ciclo continua fino a quando l'utente non decide di uscire selezionando l'opzione corrispondente. e termina l'esecuzione.