Scarabeo

Inizializzazione variabili globali

All'inizio del programma vengono inizializzate le variabili globali a cui hanno accesso tutte le funzioni e che vengono deallocate solo al termine del programma.

Esse sono:

- una matrice 17x17 di tipo *char* che funge da tabellone, i cui elementi sono tutti carattere spazio vuoto;
- una matrice identica alla prima che funge da backup;
- un vettore di tipo *char* che contiene le lettere che verranno distribuite ai giocatori e man mano eliminate per rendere più efficiente la distribuzione;
- un *bool* che indica che quello giocato è l'ultimo turno;
- quattro variabili di tipo *giocatore*, ovvero: {
  string nome;
  int punteggio = 0;
  vector<char> lettere;
  }
- una variabile di tipo jolly così definita: {
  bool jolly;
  int cont;
  int usati;
  }

int main ()

All'inizio della funzione main viene chiesto di inserire in input il numero di giocatori.

Attraverso uno *switch case*, a seconda del numero inserito, vengono richiesti i nomi dei giocatori, vengono assegnate loro delle lettere tramite funzione **assegnalettere()**, il punteggio è settato a 0 e viene richiamata la funzione **turno()**.

bool turno (int numerogiocatori, int prossimogiocatore, int tabellone_pieno, bool inizio)

La funzione **turno()** riceve in ingresso tre variabili di tipo *int*, una che specifica quanti giocatori stanno partecipando alla partita, una che indica il giocatore corrente, una che indica quante caselle del tabellone sono occupate attualmente.

Prima di tutto la funzione controlla se non è più possibile inserire lettere nel tabellone e in tal caso fa terminare la partita. Altrimenti, controlla che il giocatore abbia otto lettere. Se non è questo il caso chiama la funzione **assegnalettere()**.

Stampa quindi a video il tabellone attraverso la funzione **stampa()** e salva la disposizione corrente delle lettere sul tabellone attraverso la funzione **backup()**.

Entra, quindi, in un ciclo *while* da cui può uscire solo una volta che il giocatore avrà dato in input una parola valida da inserire nel tabellone. All'interno del ciclo al giocatore vengono mostrate le sue lettere; nel caso in cui tra esse ci sia un jolly, viene rammentato al giocatore che lo potrà usare,

ma che gli costerà dieci punti. Viene inoltre data la possibilità di accedere alla funzione **suggerimento()**.

Una volta che è stata inserita la parola e le coordinate dove la si vuole scrivere, viene chiamata la funzione **controllocompleto()**.

Una volta fuori dal ciclo viene scritta la parola sul tabellone e viene fatto un ultimo controllo sugli incroci attraverso la funzione **contro_incroci()**.

Nel caso in cui esso risulti negativo, verrà richiamata la funzione **backup()** per cancellare la parola appena inserita e poi la funzione **turno()** con il numero identificativo del giocatore corrente.

Nel caso positivo, invece, la parola è scritta definitivamente sul tabellone, le lettere del giocatore vengono aggiornate e così il suo punteggio attraverso la funzione **conteggiopunti()**. Viene anche data la possibilità al giocatore (se in possesso di una lettere adatta) di sostituire un eventuale jolly presente sul tabellone con una delle proprie lettere.

Infine, viene aggiornata la variabile *tabellone_pieno* e richiamata la funzione **turno()** attraverso uno *switch case* che controlla quanti giocatori partecipano e quindi chi sarà il prossimo.

void vincitore (int numerogiocatori)

Questa funzione viene chiamata nel caso in cui le lettere da distribuire e già distribuite siano finite o nel caso in cui il tabellone sia pieno. Essa verifica quale dei giocatori abbia ottenuto il punteggio maggiore e se si sia verificato nel caso un pareggio. Stampa quindi a video un messaggio per il giocatore che ha vinto.

void suggerimento (int giocatore)

La funzione suggerimento è suddivisa in due parti.

- Nella prima viene creata una lista di codici che identificano tutte le combinazioni (dove quindi non importa l'ordine) delle lettere del giocatore. Per fare ciò si sfrutta *il teorema fondamentale dell'aritmetica per cui ogni numero naturale diverso da 0 e da 1 o è primo, o è il prodotto di fattori primi, e tale decomposizione in fattori primi è unica a meno dell'ordine dei fattori.*
  E' quindi possibile utilizzare un vettore di variabili di tipo *long long* per contenere tutte le combinazioni di lettere (un numero che identifica tutte e otto, otto numeri per identificare tutte le combinazioni da sette e così via …). Le combinazioni vengono codificate richiamando la funzione **codice()** che prende in ingresso una variabile di tipo *char* corrispondente a una lettera dell'alfabeto e restituisce un numero primo unico ad essa corrispondente.
  Il vettore viene quindi scorso per eliminare eventuali doppioni ed ordinato richiamando la funzione *std::sort* a cui vengono passati gli estremi del vettore.

- Nella seconda parte viene scorso il tabellone attraverso due *cicli for*. Quando si trova una lettera dell'alfabeto:
  - il file di testo contenente il dizionario viene preso in input attraverso *ifstream*;
  - si richiama la funzione **conteggiopunti_due()** per calcolare il punteggio di ogni parola del dizionario e confrontarla con quella salvata nella variabile di tipo *int punteggi* (a cui è inizialmente assegnato il valore 0);

- se la parola risulta avere un punteggio maggiore delle precedenti il codice (prodotto di numeri primi) ad essa corrispondente viene passato alla funzione **findnumber()** insieme al codice corrispondente alla lettera del tabellone identificata precedentemente;
- se si ottiene da findnumber() che il codice della parola corrente corrisponde a quello di una delle combinazioni delle lettere del giocatore moltiplicata per il codice della lettera del tabellone (ovvero se la parola corrente è formata dalle lettere del giocatore e dalla parola sul tabellone) si può procedere a controllare la validità della parola;
- se si ottiene che la parola può essere scritta sul tabellone (si usa **controllo_per_suggerimento()**) e la nuova parola viene salvata in una stringa a parte insieme al suo punteggio (che viene salvato nella variabile di tipo *int punteggi*) e alle coordinate e verso in cui può essere scritta

Alla fine della funzione viene stampato a video il suggerimento.

void fun_fact ()

La funzione fun_fact() viene richiamata periodicamente durante la funzione **suggerimento()** e stampa a video degli aneddoti sul gioco dello Scarabeo mentre la funzione scorre il tabellone. Essa usa la funzione rand() per ottenere un numero random identificativo dell'aneddoto da stampare.

bool findnumber (int first, int last, long long nuova, vector<long long> & cont, int codicex)

La funzione findnumber() prende in ingresso due variabili di tipo *int* (*first* e *last*) che indicano gli estremi del *vettore* di variabili di tipo *long long cont*, cont stesso (passato per *reference*), una variabile di tipo *long long* che va cercata all'interno del vettore e una variabile di tipo *int* corrispondente a un numero da moltiplicare agli elementi di *cont* prima che avvenga la ricerca.

La funzione utilizza la *ricerca dicotomica*: essa calcola un numero, detto *mediano*, e compara l'elemento di *cont* ad esso corrispondente con la variabile da trovare. Decide quindi se restituire *false* (l'elemento di *cont* corrisponde alla variabile che sta cercando) o se richiamare se stessa cambiando i parametri *first* e *last* con il valore del *mediano* (a seconda che la variabile da cercare sia maggiore o minore dell'elemento di *cont* corrispondente al *mediano*).

Se la ricerca si rivela infruttuosa restituisce *true*.

int codice (char c)

Prende in ingresso una variabile di tipo *char* corrispondente a una lettera dell'alfabeto e restituisce un numero primo unico ad essa corrispondente. Per fare ciò si utilizza uno *switch* case; nel caso in cui uno dei caratteri ricevuti sia '*' la funzione restituisce 1.

int conteggiopunti_due (string& par)

Questa funzione è virtualmente identica alla funzione **conteggiopunti()** ma omette di controllare i punteggi presenti sul tabellone.

controllolettere controllocompleto (string &parola, int riga, int colonna, bool verso, int numerogioc, bool vuoto)

Questa funzione prende in ingresso:

- una *stringa* (passata per *riferimento*) e corrispondente alla parola che si vuole controllare,

- un *vettore* di variabili di tipo *char* corrispondente alle lettere del giocatore (anche esso passato per *riferimento*),
- due variabili di tipo *int* corrispondenti alle coordinate dove si vuole scrivere la parola,
- una variabile di tipo *bool* che indica il verso in cui la si vuole scrivere,
- una variabile di tipo *int* che identifica il giocatore,
- una variabile di tipo *bool* che indica se il tabellone è vuoto.

Restituisce una variabile di tipo *controllolettere* così definita:

```
struct controllolettere{
        vector<char> nuovo;
        bool giusto;
        }
```

La funzione è divisa in tre *if* innestati:

- il primo verifica che la parola appartenga al dizionario richiamando la funzione **parola_diz()**;
- il secondo richiama invece **contrograg()** per controllare che la parola possa essere scritta sul tabellone e che incroci almeno una lettera già presente su esso. Salva quindi la *string*a ottenuta da contrograg nella variabile *temporaneabis*.
- il terzo verifica richiamando **controlloletteregiocatore()**, a cui passa *temporaneabis* (in cui sono salvate le lettere della parola da scrivere che non sono già presenti sul tabellone) se il giocatore possiede effettivamente le lettere per scrivere la parola passata in input.

Al termine della funzione, se tutti i controlli risultano positivi, la funzione restituisce le nuove lettere del giocatore (ovvero quelle iniziali a cui sono state sottratte quelle per scrivere la parola) e il *bool giusto* settato a true; se almeno un controllo risulta negativo, viene restituito false.

suggerimento_ controllo_per_suggerimento (string &parola, int riga, int colonna)

Questa funzione è molto simile alla funzione **controllocompleto()** eccetto per tre aspetti:

- Invece di controllare un solo set di coordinate essa controlla tutte le caselle presenti sulla riga presa in ingresso e poi tutte le caselle sulla colonna presa in ingresso. Essa fa cioè un controllo prima in verticale, controllando sempre la stessa colonna ma cambiando la riga, e poi uno in orizzontale, a riga fissa e colonna che varia. Una volta trovata una coppia riga-colonna in cui è possibile scrivere la parola la funzione ritorna immediatamente senza effettuare controlli su altre caselle.
- La funzione controlla anche se la parola crea incroci validi con le parole già presenti sul tabellone, compito affidato nell'altro caso alla funzione **turno().**
- La funzione restituisce una variabile di tipo *suggerimento_* così definita:

```
struct suggerimento_ {
        bool vero; //equivalente al bool giusto di controllolettere
        int riga;
        int colonna;
        bool verso;
        }
```

int conteggiopunti (string& par, int posizioneriga, int posizionecolonna, bool verso)

La funzione conteggiopunti prende in ingresso:

- una *string* passata come riferimento e corrispondente alla parola che si vuole scrivere sul tabellone,
- due *int* corrispondenti alle coordinate dove si vuole scrivere,
- un *bool* che indica il verso in cui si vuole scrivere.

La funzione salva in un vettore di variabili di tipo *int* (*punteggitemporanei*) i punteggi corrispondenti a ogni singola lettera della parola; fatto ciò scorre una *matrice* di *int* 17x17 chiamata *score* dove sono salvati i punteggi speciali. Essi vengono applicati ai punti già salvati in *punteggitemporanei*. Infine il contenuto di *punteggitemporanei* viene sommato in una singola variabile di tipo *int* chiamata *punti* e la matrice *score* viene scorsa di nuovo per vedere se vanno applicati dei punteggi speciali alla parola nel suo complesso.

Viene inoltre controllata la lunghezza della parola richiamando la funzione *size()* per assegnare ulteriori punti; un controllo finale conferisce cento punti in più nel caso in cui la parola sia "scarabeo".

Alla fine la funzione restituisce la variabile punti a cui viene sommato dieci se la variabile globale *jolly* è settata a false.

bool contro_incroci (int riga, int colonna, bool verso, int lunghezza)

La funzione contro_incroci viene richiamata una volta che una parola è stata scritta sul tabellone per osservare gli incroci che essa crea e deciderne la validità.

Essa è divisa in quattro parti:

- un controllo iniziale per le parole scritte in verticale
- un controllo iniziale per le parole scritte in orizzontale
- un secondo controllo per le parole scritte in verticale
- un secondo controllo per le parole scritte in orizzontale.

Nel primo tipo di controllo le caselle che precedono e che succedono quelle della parola scritta dal giocatore (cioè quelle sopra e sotto nel caso verticale e quelle a sinistra e a destra nel caso orizzontale) vengono conteggiate e se non sono libere (cioè non corrispondono al carattere vuoto) viene salvata la parola formata dalle lettere adiacenti e dalla parola scritta dal giocatore e viene passata a **parola_diz()**; nel caso in cui parola_diz() restituisca *false* la funzione *contro_incroci* restituisce immediatamente *false*.

Segue il secondo tipo di controllo, che scorre la parola scritta dal giocatore per tutta la sua lunghezza e controlla gli incroci nel verso opposto (orizzontale per verticale e verticale per orizzontale); anche durante questo controllo viene chiamata parola_diz() e nel caso in cui restituisca false si esce immediatamente dalla funzione restituendo false.

Alla fine di entrambi i controlli si restituisce true.

bool parola_diz (string& parola)

Questa funzione prende in ingresso una stringa passata come riferimento corrispondente alla parola che si vuole cercare sul dizionario.

Inizialmente è posto un controllo iniziale sulla lunghezza della parola che usa *size()* per restituire *true* in caso la parola sia formata da meno di tre caratteri.

Viene quindi usato *ifstream* per leggere in input il file di testo dizionario; ogni parola del file viene salvata in una stringa a sua volta salvata in un *vettore* di *stringhe* chiamato *dizio*.

Viene quindi usato *sort()* per ordinare le stringhe di *dizio* e viene richiamato **controdizionario()** che restituisce *true* se la parola cercata è presente nel vettore, *false* altrimenti.

bool controdizionario (int first, int last, string &parola, const vector<string>& dizio)
La funzione usa la ricerca dicotomica per cercare la *string* parola tra gli elementi del *vettore dizio* in modo non dissimile dalla funzione **findnumber()**. Per fare ciò viene sfruttata la possibilità di comparare fra di loro le stringhe usando gli operatori < e >.

void stampa ()
La funzione stampa() scorre la variabile globale tabellone stampandola a video.

controllolettere controlloletteregiocatore (string& tempbis, int numerogiocatore)
Questa funzione scorre le lettere della parola che si vuole scrivere sul tabellone e controlla che ognuna corrisponda ad almeno una lettera tra quelle disponibili per il giocatore; se trova una lettera della parola non "posseduta" dal giocatore restituisce immediatamente false; se invece arriva fino alla fine del controllo restituisce le nuove lettere del giocatore (quelle iniziali meno quelle usate per scrivere la parola) e un *bool* settato a true.

void assegnalettere (int lettere, int giocatore)
Alla funzione vengono passate due variabili di tipo *int*:
- Una che indica il numero di lettere da assegnare al giocatore
- Una che definisce il giocatore a cui assegnare le lettere

Viene chiamata *srand*() passandogli come seed *time*(0). Si entra quindi in un ciclo *while* determinato dal numero di lettere da assegnare; all'interno del ciclo si chiama *rand*() per ottenere un numero random. In una variabile di tipo *int* chiamata *random* viene salvato il resto della divisione tra il numero trovato con *rand*() e la dimensione del *vettore* di *char alfabeto* (inizializzato globalmente); il numero salvato in *random* viene usato come indice per identificare una lettera presente nel vettore alfabeto e assegnarla al giocatore.
Se il vettore risulta vuoto, la funzione setta la variabile globale *ultimo_turno* a true e la variabile di tipo int *lettere* a 0 per uscire dal ciclo.

vettore_bool controgrig (string temp, int riga, int colonna, bool verso)
La funzione controgrig() opera alcuni controlli sulla parola che si vuole scrivere sul tabellone:
- Controlla che la parola non ecceda lo spazio disponibile (controlla cioè che la somma della variabile di tipo *int riga* (o *colonna*) e del valore corrispondente alla lunghezza della parola sia minore di 17)
- Controlla che le caselle dove si vuole scrivere la parola siano libere, o che contengano già le lettere che vi si vuole scrivere
- Controlla che la parola incroci almeno con una lettera già presente sul tabellone (controllo attuato aggiungendo un contatore al controllo precedente)

La funzione restituisce una variabile di tipo *vettore_bool*, così definita:
{
bool vero_falso;
string tempbisx;
}

Nel caso in cui i controlli saranno andati a buon fine in *tempbisx* verranno salvate le lettere della parola che non sono già presenti sul tabellone.

void backup (bool versione)

La funzione backup permette di scrivere una parola sul tabellone potendo eventualmente tornare alla configurazione precedente.

Essa è distinta in due parti:

- Se le viene passato un parametro false, la funzione scorre la variabile globale *tabellone* e la copia in un'altra variabile globale chiamata *tabellone_backup.*
- Se le viene passato un parametro true, la funzione scorre la variabile *tabellone_backup* salvata in una chiamata precedente e copia il suo contenuto nella funzione *tabellone.*