|  |
| --- |
|  |
| Scarabeo |
| Relazione elaborato di Fondamenti di Informatica |
|  |
| **Moscatelli Samuele – Pozzoli Andrea** |
| **AA 2016/2017** |

|  |
| --- |
|  |

**Relazione elaborato**

**Indice:**

* Analisi dei requisiti
* Progettazione della soluzione
* Note sulla realizzazione
* Prove di test

**Analisi dei requisiti**

Giocatori e campo di gioco

Il programma consente all’utente di giocare una partita al gioco dello Scarabeo con le regole ufficiali.

Scarabeo può essere giocato da due, tre oppure quattro giocatori. Il campo da gioco è costituito da una griglia quadrata di 17 caselle per lato, identificate con una coppia di coordinate riga-colonna. Alcune di queste celle hanno un valore speciale, cioè permettono di incrementare il punteggio ottenuto con l’inserimento di una parola. Esse sono:

* “2L”, che raddoppia il valore della lettera posta su quella casella;
* “3L”, che triplica il valore della singola lettera;
* “2P”, che moltiplica per due il punteggio dell’intera parola (senza contare “2L” e “3L”);
* “3P”, che triplica il valore della parola (sempre non tenendo conto dei “2L” e dei “3L”).

Nel gioco la prima tipologia di caselle speciali è contrassegnata dal colore grigio, la seconda dal colore blu, le celle “2P” sono colorate di verde, mentre le ultime di rosso.

Tessere

Il gioco comprende un “sacchetto” contenente 130 tessere, 128 rappresentanti le lettere dell’alfabeto italiano (quindi escludendo J, K, W, X, Y) e 2 jolly, indicati con il simbolo “$”.

Le lettere sono presenti proporzionalmente all’utilizzo nella lingua italiana, quindi si hanno:

* 12 A, E, I, O;
* 7 C, R, S, T;
* 6 L, M, N;
* 4 B, D, F, G, P, U, V;
* 2 H, Q, Z.

Regole del gioco

Il gioco comincia distribuendo otto lettere a ciascun giocatore, pescate dal “sacchetto” casualmente.

Il primo giocatore deve comporre una parola di senso compiuto, utilizzando le proprie lettere, e inserirla nel campo in modo tale che passi per il centro (che ha coordinate (8 ; 8)). Il giocatore deve specificare la parola che vuole inserire, le coordinate della cella di partenza e la direzione di inserimento (orizzontale o verticale).

Le parole possono essere inserite solamente in orizzontale da sinistra a destra e in verticale dall’alto al basso. Ogni tessera usata deve essere posizionata in una sola cella, e le lettere di una parola devono essere disposte in caselle contigue.

Una volta inserita la parola, si calcola il punteggio ottenuto (in base ai criteri spiegati nel prossimo paragrafo) e lo si somma a quello del giocatore. Egli pesca un numero di tessere dal “sacchetto” pari a quello delle lettere utilizzate, in modo da averne sempre otto, e passa il turno al giocatore successivo.

Nei turni successivi al primo, i giocatori devono inserire una parola intersecandone almeno una già presente sul campo (è possibile anche aggiungere lettere a una parola, per esempio anteporre una “S” alla parola “VANTAGGIO” per inserire “SVANTAGGIO”). Non è possibile inserire una parola se vi sono lettere nelle celle adiacenti alla testa e alla coda. Se inserendo la parola se ne vengono a formare altre di lunghezza maggiore o uguale a tre, nella direzione opposta all’inserimento, allora esse devono essere di senso compiuto; in questo caso, esse vengono contate nel calcolo del punteggio. Le parole consentite sono quelle incluse nel dizionario italiano, con l’esclusione dei nomi propri e geografici. Sono ammesse anche tutte le sigle (incluse le targhe automobilistiche) e i verbi declinati.

Le tessere jolly possono essere utilizzate in sostituzione a qualsiasi lettera dell’alfabeto italiano. Se il giocatore corrente possiede la lettera corrispondente a un jolly sul campo, egli può appropriarsene sostituendolo con la propria all’inizio del turno.

A disposizione dei giocatori è presente un comando che consente di visualizzare un suggerimento per la parola migliore (ovvero quella che permette di ottenere il punteggio massimo in base alle lettere possedute dal giocatore e presenti sul campo).

Il gioco termina nel momento in cui tutte le lettere del “sacchetto” sono terminate e uno dei giocatori finisce le proprie.

Calcolo del punteggio

Ogni lettera ha un proprio valore, in base alla frequenza di utilizzo nella lingua italiana (maggiore è il loro utilizzo e minore è il loro punteggio). Per ogni parola inserita il giocatore calcola la somma dei valori delle lettere che la formano, tenendo conto anche di quelle già presenti sul campo e intersecate. Il calcolo del punteggio avviene solo nel caso in cui la parola abbia lunghezza maggiore o uguale a tre (una parola da due lettere vale zero punti). Il punteggio di ogni lettera è il seguente:

* 1 punto per ogni A, C, E, I, O, R, S, T;
* 2 punti per ogni L, M, N;
* 3 per P;
* 4 per B, D, F, G, U, V;
* 8 per H, Z;
* 10 per Q.

In caso di utilizzo del jolly, esso assume il valore della lettera sostituita.

Il punteggio viene incrementato nel caso in cui si intersechino le caselle speciali (spiegate nel primo paragrafo). Vengono seguite le regole matematiche, quindi prima si moltiplica (nel caso di “2P” e “3P”) e poi si somma (per “2L” e “3L”).

Altri punteggi bonus sono:

* 10 punti bonus vengono sommati se non si utilizzano jolly;
* 10 punti bonus se si utilizzano 6 delle proprie lettere;
* 30 punti bonus se si utilizzano 7;
* 50 punti bonus nel caso si utilizzano tutte e 8.
* 100 punti bonus se si riesce a scrivere la parola “SCARABEO” oppure “SCARABEI”.

Determinazione del vincitore

Al termine del gioco, il giocatore che ha composto l’ultima parola, e quindi finito le proprie lettere per primo, ottiene il punteggio delle tessere rimaste “in mano” agli altri giocatori. Vince il giocatore che ha totalizzato il maggior numero di punti.

**Progettazione della soluzione**

Parte globale

Abbiamo deciso di avvalerci di strutture dati dichiarate globalmente affinché esse potessero essere visibili e quindi modificabili da tutte le funzioni implementate.

In particolare, abbiamo usato:

* Un vector di interi (“valorelettere”) che contiene il valore di ogni singola tessera in ordine alfabetico tenendo conto delle lettere straniere per comodità coi cicli for.
* Un vector di caratteri (“alfabeto”) che simula il sacchetto da cui vengono pescate le lettere; esso contiene ripetute con la loro frequenza tutte le lettere presenti nel gioco, inclusi i jolly.
* Un vector di interi (“punteggiogiocatori”) contenente i punteggi correnti dei giocatori in base al numero e alla posizione.
* Un vector di stringhe (“nomegiocatori”) in cui sono presenti i nomi dei giocatori chiesti a inizio partita, in ordine di inserimento;
* Un vector di stringhe (“letteregiocatori”) per le lettere correnti di ogni giocatore, viste appunto come stringhe.
* Un vector di caratteri (“letterejolly”) che tiene salvato quali lettere sostituiscono il primo e il secondo jolly quando utilizzati. Se i jolly non sono sul campo, essi sono salvati come “$”.
* Un vector di interi (“jolly1”) contenente la posizione nel campo del primo jolly; se il jolly non è presente esso è inizializzato a una cella esterna.
* Un vector di interi (“jolly2”) analogo al precedente, ma relativo al secondo jolly.
* Un vector di stringhe (“paroledizionario”) contenente le parole del dizionario di lunghezza maggiore o uguale a 3.
* Un vector di strighe (“paroledadue”) contenente le parole del dizionario di lunghezza minore di tre.
* Un intero (“punteggioturno”) che salva il punteggio ottenuto di volta in volta dal giocatore corrente, permettendo quindi di stamparlo.
* Un intero (“ultimoainserire”) che salva di volta in volta l’indice dell’ultimo giocatore che ha inserito una parola.
* Un array di 17 array di 17 stringhe (“campo”) che costituisce il campo da gioco. Inizialmente tutte le stringhe sono uguali a “vuota”, mentre quando le parole vengono inserite, le celle assumono il valore della lettera che contengono.
* Un array di 17 array di 17 stringhe (“caselle speciali”) che costituisce il campo da gioco. Tale ‘matrice’ non viene mai modificata in quanto serve a memorizzare la posizione e il tipo delle caselle (speciali e non).
* Un array di 17 array di 15 interi (“libereorizzontali”) che contiene il numero di celle libere alla destra di ogni casella (se maggiore di 9, contiene 9), così da consentire di determinare se è possibile inserire o meno una parola in orizzontale a partire da una certa posizione.
* Un array di 15 array di 17 interi (“libereverticali”) che contiene il numero di celle libere in basso verticalmente alla cella stessa (se maggiore di 9, contiene 9), così da consentire se è possibile inserire o meno una parola in verticale a partire da una certa posizione.

Main

Nel main sono dichiarati:

* Un intero “turno”, inizializzato a zero, necessario per contare il numero di turni. Esso viene incrementato di uno ogni fine turno.
* Un intero “numerogiocatori”, inizializzato a zero, che tiene salvato il numero di giocatori (2, 3 o 4) una volta che è chiesto in input.
* Un intero “lettererimanenti” che contiene il numero delle lettere ancora presenti nel “sacchetto”. Esso viene aggiornato ogni volta che il giocatore inserisce una parola nel campo e quindi gli vengono distribuite un numero di lettere pari a quelle utilizzate nel turno.
* Un booleano “fine” che indica se la partita è finita o se è ancora possibile giocare.
* Una stringa “passa”, che ogni turno torna a essere vuota, che serve per, una volta svuotato il “sacchetto”, chiedere al giocatore se vuole “passare il turno” se non gli è possibile comporre una parola.
* Una stringa “suggerimento” che serve per leggere la risposta dell’utente alla proposta di un suggerimento.
* Un vector di stringhe “parolefattibili” contenente le parole del dizionario che il giocatore corrente, in base alle lettere che possiede, può realizzare.

Il main è strutturato in modo schematico. Infatti, esso si limita a richiamare le funzioni implementate. Il procedimento, in ordine, si svolge nel seguente modo:

1. Chiede all’utente il numero di giocatori (salvandolo in “numerogiocatori”) e il loro nome richiamando la funzione “chiedinome()”.
2. Stampa per la prima volta il campo invocando la funzione stampaCampo().
3. Vengono riempiti i vettori “paroledizionario” e “paroledadue” tramite la funzione “riempidizionario()”.
4. Vengono distribuite le otto lettere iniziali a ciascun giocatore con “distribuzionelettere()”, tramite un ciclo for che ripete in base al numero dei giocatori.

A questo punto inizia il ciclo do-while rappresentante i turni del gioco, che termina in base al valore della variabile “fine”. Esso si sviluppa come segue:

1. La funzione “sistemamatrici()” viene invocata per aggiornare “libereverticali” e “libereorizzontali”.
2. Viene richiamata “sfoltiscidizionario()” che cambia il valore di “parolefattibili”.
3. Viene invocata la funzione “visualizzalettere” che stampa le lettere del giocatore corrente.
4. Si apre un ciclo for che controlla se il giocatore corrente possiede almeno un jolly; in caso affermativo assegna al vettore “parolefattibili” il vettore “paroledizionario”, in quanto, con il jolly, il giocatore, potenzialmente, potrebbe comporre tutte le parole presenti nel dizionario.
5. Con un do-while si chiede all’utente se vuole o meno il suggerimento, cambiando il valore di “suggerimento”. Il ciclo termina solo se quest’ultima variabile è uguale a si o a no.
6. Se “suggerimento” è uguale a si allora viene richiamata la funzione “parolamigliore()” che stampa il suggerimento.
7. Nel caso in cui il sacchetto sia vuoto, viene chiesto al giocatore corrente se vuole passare il turno; in caso affermativo il turno passa al giocatore successivo.
8. Se il giocatore non ha passato il turno viene invocata la funzione “inserisciparola()” che chiede al giocatore corrente la parola che vuole inserire ed esegue i dovuti controlli.
9. Viene aggiornato l’ultimo giocatore che ha inserito una parola (“ultimoainserire”).
10. Viene richiamata la funzione “system(“cls”)” che permette di “pulire” lo schermo una volta inserita la parola.
11. Viene poi invocata “stampapunteggio()” che stampa i punteggi aggiornati di tutti i giocatori.
12. Viene richiamata nuovamente la funzione “stampaCampo()” che stampa il campo di gioco aggiungendo l’ultima parola inserita.
13. La variabile “fine” assume il valore di ritorno della funzione “finegioco()” che ritorna falso se il gioco non è terminato, vero se invece si è giunti al termine della partita.
14. Al termine del turno, quindi dopo aver inserito la parola, il giocatore pesca le lettere necessarie tramite la funzione “distribuzionelettere()”.
15. La variabile “turno” viene incrementata.
16. Una volta usciti dal do-while, viene richiamata la funzione “stabiliscivincitore()” che determina quale giocatore ha vinto la partita.

Funzioni

1. Void chiedinome (int numerogiocatori): questa funzione prende in ingresso solo un intero indicante il numero di giocatori e si occupa di chiedere all’utente i nomi dei giocatori che partecipano alla partita. I nomi inseriti vengono salvati ciclicamente (a seconda del numero dei giocatori) all’interno del vettore di stringhe “nomegiocatori”, e saranno poi utilizzati per interagire con i giocatori durante i vari turni. Inoltre i vettori “punteggiogiocatori” e “letteregiocatori” acquisiscono un numero di celle pari al numero di giocatori.
2. Void distribuzionelettere (int posizione, int & lettererimanenti): questa funzione prende in ingresso due interi, uno indicante il giocatore corrente e l’altro il numero di lettere rimanenti. Il suo compito, come dice il nome stesso, è quello di distribuire le lettere mancanti al giocatore in questione e tenere aggiornato il numero di lettere rimaste nel “sacchetto”. In particolare si sviluppa nel modo seguente:
   1. Si apre un ciclo for che si ripete tante volte quante sono le lettere mancanti al giocatore.
   2. Se il sacchetto è vuoto il ciclo si interrompe e la funzione non modifica le lettere del giocatore.
   3. Se il sacchetto non è vuoto la funzione “pesca” casualmente ad ogni ciclo una lettera da aggiungere al giocatore per completare l’ottetto.
   4. Elimina di volta in volta le lettere pescate.
3. Void visualizzalettere (int posizione): questa funzione prende in ingresso un intero indicante il giocatore corrente e ha il compito di stampare le lettere di tale giocatore. Inoltre, essa verifica anche se il giocatore può o meno prendere un jolly dal campo; in caso affermativo chiede al giocatore se lo vuole scambiare con la sua lettera. Il procedimento si svolge come segue:
   1. Stampa le lettere.
   2. Verifica se il primo jolly è sul campo.
   3. In caso affermativo controlla se il giocatore possiede la lettera corrispondente al jolly ed eventualmente chiede al giocatore se lo vuole prendere.
   4. Verifica se il secondo jolly è sul campo.
   5. In caso affermativo controlla se il giocatore possiede la lettera corrispondente al jolly ed eventualmente chiede al giocatore se lo vuole prendere.
   6. Nel caso in cui abbia modificato le lettere del giocatore le ristampa.
4. Void stampaCampo (): questa funzione non riceve valori in ingresso; la sua funzione è semplicemente quella di stampare il campo con la relativa legenda:
   1. Stampa gli indici delle colonne.
   2. Stampa gli indici di riga e il campo distinguendo le celle di diverso tipo.
   3. Stampa la legenda del campo.
5. Void inserisciparola (int turno, int numerogiocatori): questa funzione prende in ingresso due interi, uno indicante il turno corrente e l’altro indicante il numero dei giocatori. Tale funzione costituisce il corpo principale del programma in quanto gestisce l’inserimento delle parole, e si fa quindi carico di tutti i relativi controlli necessari. Il procedimento si svolge come segue:
   1. Viene gestito prima il turno 0.
   2. Si chiede all’utente qual è la parola che vuole inserire e nel momento dell’inserimento si controlla che la parola sia presente all’interno del dizionario.
   3. Si conta il numero di jolly posseduti dal giocatore corrente.
   4. Attraverso due cicli for si verifica che il giocatore possegga le lettere necessarie a formare la parola.
   5. Se il giocatore non possiede le lettere necessarie la funzione stampa un avviso all’utente e riparte richiedendo la parola.
   6. Altrimenti la funzione procede chiedendo la riga e la colonna della cella da cui si vuole partire.
   7. La funzione inserisce automaticamente la parola di modo che passi dal centro, senza chiedere quindi all’utente la direzione di inserimento. Se dalla posizione di partenza inserita è impossibile passare per il centro la funzione stampa un avviso e riparte richiedendo la parola.
   8. Nel caso in cui la posizione inserita sia il centro stesso, la funzione chiede la direzione di inserimento.
   9. Se tutti i controlli sono andati a buon fine inserisce la parola nel campo.
   10. Aggiorna il punteggio del giocatore e stampa i punti appena fatti dallo stesso.
   11. A questo punto vi è il caso generico, che gestisce i turni successivi al turno 0.
   12. Esso inizia chiedendo al giocatore la parola che vuole inserire, sempre controllandone la presenza nel dizionario.
   13. Chiede poi le coordinate della cella di partenza e la direzione di inserimento.
   14. A questo punto controlla che la parola non esca dal campo; in caso affermativo stampa un avviso al giocatore e riparte richiedendo la parola da inserire.
   15. Controlla se sono presenti lettere nelle celle immediatamente prima e dopo la parola inserita; in caso affermativo stampa un avviso e riparte richiedendo la parola.
   16. Controlla se si formano parole e, in caso affermativo controlla che esse siano corrette (ovvero che siano presenti nel dizionario). Se sono corrette aggiunge i punti di queste al punteggio totalizzato dal giocatore in quel turno.
   17. Controlla che la parola intersechi almeno una delle lettere già presenti sul campo e verifica la corrispondenza tra parola e intersezione; verifica inoltre che il giocatore utilizzi almeno una delle proprie lettere. Se il controllo non va a buon fine stampa un avviso e richiede la parola da inserire.
   18. A questo punto controlla che il giocatore possegga le lettere necessarie per comporre la parola (escludendo ovviamente quelle che vengono intersecate sul campo). Se ciò non va a buon fine, stampa un avviso e ricomincia richiedendo la parola da inserire.
   19. Una volta superati tutti i controlli inserisce la parola nel campo, stampa il punteggio acquisito e aggiorna quello totale del giocatore.
6. Void eliminalettera (int posizione): questa funzione riceve in ingresso un intero rappresentante la posizione della lettera da eliminare; il suo compito è infatti quello di togliere dal sacchetto una lettera, che, evidentemente, è stata distribuita ad un giocatore. Essa procede come segue:
   1. Crea un vettore in cui inserisce tutte le lettere tranne quella da eliminare.
   2. Assegna tale vettore al vettore “sacchetto”, così da eliminare la lettera distribuita.
7. Int calcolapunteggio (string& parola, int const& x, int const& y, bool& orizzontale, int const& lettereusate, bool const& usoj): questa funzione riceve in ingresso un riferimento a una striga costituente la parola inserita, due riferimenti costanti a interi rappresentanti le coordinate della cella da cui viene inserita la parola, un riferimento a booleano per conoscere la direzione di inserimento, un riferimento costante a intero indicante il numero di lettere usate e un riferimento constante a booleano indicante se il giocatore ha o no usato il jolly. La funzione procede come segue:
   1. Si apre un ciclo for che si ripete tante volte quante sono le lettere della parola.
   2. Viene calcolata la posizione della lettera in esame, così da poterne conoscere il valore all’interno del vettore “valorelettere”.
   3. Verifica in quale casella è posizionata la lettera in questione, e nel caso in cui si trovi in una cella speciale, calcola il valore ottenuto.
   4. Calcola quindi il punteggio ottenuto.
   5. Aggiunge eventuali punti bonus nel caso si soddisfino le richieste del gioco per ottenerli.
   6. Restituisce il punteggio ottenuto.
8. Bool controllodizionario (string parola): questa funzione riceve in ingresso una stringa rappresentante la parola da controllare. Il suo compito è infatti quello di controllare se una parola sia presente o meno nel dizionario. Per fare ciò ci siamo avvalsi della ricerca dicotomica, che consente una riduzione cospicua dei tempi, visto che la quantità di informazioni da analizzare non è per niente ridotta. Essa procede come segue:
   1. Inizializza una variabile “start” a zero, una variabile “end” all’ultima posizione del dizionario e una variabile “centro” a zero.
   2. Pone la variabile “centro” a (“start”+”end”)/2.
   3. Controlla se in una delle posizioni “start”, “centro” e “end” è presente la parola.
   4. Se è così restituisce vero, altrimenti se la parola è maggiore di centro pone “start” uguale a “centro”, se minore a “centro” pone “end” a “centro”, ricalcola “centro” e ripete le operazioni già fatte nei punti precedente.
   5. Ciò viene ripetuto fino a quando o si trova la parola, e quindi si restituisce vero, oppure quando “end” diventa minore di “start”, che significa che la parola non è presente, e quindi si restituisce false.
   6. Se la parola ha lunghezza minore di tre, si cerca tramite una ricerca lineare nel vettore “paroledadue”.
9. Void stabiliscivincitore (int numerogiocatori): questa funzione prende in ingresso un intero rappresentante il numero dei giocatori e ha il compito di stabilire, a fine partita, il vincitore. In particolare, la funzione procede come segue:
   1. Stampa nome e punteggio del primo giocatore.
   2. Si apre un ciclo for che scorre il vettore “punteggiogiocatori” calcolando in questo modo chi ha totalizzato un maggior numero di punti.
   3. Durante il ciclo for si stampano i punteggi di tutti i giocatori.
   4. Viene verificato che il giocatore con il punteggio massimo sia solo uno.
   5. In caso affermativo viene stampato che è il vincitore, altrimenti si cercano gli altri giocatori che hanno raggiunto il punteggio massimo e vengono quindi stampati tutti i nomi dei vincitori.
10. Bool finegioco (int numerogiocatori, int turno, int lettererimanenti): questa funzione prende in ingresso tre interi: uno rappresentante il numero di giocatori, uno rappresentante il turno corrente e uno rappresentante le lettere rimanenti. Il suo compito è quello di stabilire, ad ogni turno, se il gioco è finito oppure no, e, nel caso fosse terminato, assegnare al giocatore che ha composto l’ultima parola il punteggio delle tessere rimanenti agli altri giocatori. Il procedimento si svolge come segue:
    1. Le condizioni per cui il gioco finisca sono due: la prima è che non ci siano più lettere nel “sacchetto” e la seconda che un giocatore abbia finito anche le proprie.
    2. Se le due condizioni non sono soddisfatte la funzione restituisce immediatamente falso.
    3. Se sono soddisfatte si apre un ciclo for che calcola il punteggio delle lettere che non sono state usate dai giocatori; somma poi il totale trovato al punteggio del giocatore che ha composto l’ultima parola.
    4. Terminato il ciclo for restituisce vero.
11. Void paroalamigliore (int turno, int numerogiocatori, vector<string> parolefattibili): questa funzione riceve in ingresso un intero rappresentante il turno corrente, un intero rappresentante il numero di giocatori e un vector di stringhe riempito di tutte le parole che il giocatore corrente può potenzialmente comporre con le lettere che possiede. Il compito di tale funzione è quello di fornire all’utente il miglior suggerimento possibile, ovvero la parola che gli consentirebbe di ottenere il più alto punteggio possibile, considerando le sue lettere e la disposizione del campo. La funzione si sviluppa come segue:
    1. Anche in questo caso, come in “inserisciparola ()” la funzione distingue il primo turno da tutti gli altri.
    2. Se il turno è il primo si apre un ciclo for che di volta in volta cerca di inserire una parola diversa del dizionario (o del vettore “parolefattibili” a seconda dei casi) in ogni cella della colonna centrale, per verificare quale sia la parola e la posizione che forniscano il punteggio più elevato.
    3. A questo punto iniziano tutta una serie di controlli per verificare che la parola possa essere composta e che possa essere inserita. Essi sono disposti in modo che i controlli più semplici e veloci siano i primi, mentre quelli più complessi e lunghi siano ultimi. Per ogni controllo, se non va a buon fine si passa direttamente alla parola successiva attraverso il comando “continue”.
    4. Si apre un ciclo for che servirà a scorrere tutte le celle della colonna centrale del campo di gioco.
    5. Si controlla che la parola passi per il centro.
    6. Si controlla che il giocatore abbia le lettere necessarie per comporre la parola.
    7. Si calcola il numero di lettere che si userebbero con la parola che si sta analizzando.
    8. Si calcolano i punti che tale parola fornirebbe.
    9. Se tale punteggio è maggiore del massimo finora trovato, la parola diventa la parola migliore (al momento) e vengono quindi salvati: il suo punteggio, la posizione e la direzione di inserimento.
    10. Nel turno generico si procede invece in modo inverso: per ogni cella del campo si provano ad inserire tutte le parole del dizionario (o del vettore “parolefattibili” a seconda dei casi).
    11. Anche in questo caso i controlli sono ordinati dal più semplice al più complesso. Se il controllo non viene superato si passa direttamente alla posizione successiva.
    12. Si sfruttano le matrici “libereorizzontali” e “libereverticali” per sapere se nella posizione che si sta analizzando è possibile inserire una parola.
    13. Sempre usando le due matrici “libereorizzontali” e “libereverticali” si verifica se la parola abbia la lunghezza necessaria per intersecare almeno una delle lettere già presenti sul campo.
    14. Si controlla che la parola non esca dal campo.
    15. Si controlla che la parola intersechi correttamente le lettere già presenti su campo.
    16. Si controlla che con quella parola in quella posizione il giocatore utilizzi ameno una delle sue lettere.
    17. Si controlla che il giocatore abbia le lettere necessarie per comporre la parola.
    18. Si controlla se l’inserimento di tale parola formi altre parole, e in tale caso si controlla che le parole formate siano valide (quindi presenti nel dizionario).
    19. Si contano le lettere che il giocatore userebbe per comporre tale parola.
    20. Si calcola il punteggio che tale parola fornirebbe.
    21. Se il punteggio che fornirebbe è maggiore del massimo finora trovato, la parola diventa la parola migliore (al momento) e vengono quindi salvati: il suo punteggio, la posizione e la direzione di inserimento.
    22. Si stampa infine la parola migliore finale, le coordinate della posizione di partenza e la direzione di inserimento (se non si è trovata nessuna parola si stampa che il giocatore non può comporre nessuna parola di lunghezza maggiore o uguale a tre).
12. Void sfoltiscidizionario (vector<string> & parolefattibili, int turno): questa funzione prende in ingresso un vector di stringhe vuoto per riferimento e un intero che indica a che giocatore ci si sta riferendo. Il suo compito è quello di riempire il vettore che gli viene dato in ingresso con le parole di cui il giocatore possiede almeno una lettera. L’obiettivo di tutto ciò è quello di accorciare i tempi di ricerca della parola migliore. La funzione si sviluppa come segue:
    1. Si apre un ciclo for che si ripete tante volte quante sono le parole del vettore “paroledizionario” (vettore contenente le parole del dizionario di lunghezza superiore o uguale a tre) per analizzare ogni parola contenuta in tale vettore.
    2. Si aprono altri due cicli for, uno per scorrere la parola in questione e uno per scorrere le lettere del giocatore.
    3. Se esiste almeno una lettera della parola posseduta anche dal giocatore allora tale parola viene inserita nel vettore che la funzione riceve in ingresso.
13. Void stampapunteggio (int numerogiocatori): questa funzione riceve in ingresso un intero indicante il numero di giocatori. Il suo compito è quello, dopo la “pulizia” dello schermo che segue ogni fine turno, di stampare i punteggi aggiornati di tutti i giocatori. Ciò avviene come segue:
    1. Si apre un ciclo for che si ripete tante volte quanti sono i giocatori.
    2. Per ogni giocatore stampa il nome seguito dal punteggio.
    3. In più, per il giocatore che ha appena finito il turno viene stampato anche l’ammontare dei punti appena acquisiti.
14. Void riempidizionario (): questa funzione non riceve in ingresso alcun parametro. Il suo compito è quello di leggere da un file di testo i vocaboli del dizionario italiano e inserirli uno a uno in un vettore. La funzione procede in questo modo:
    1. Viene aperto il file di testo.
    2. Si apre un ciclo for che si ripete tante volte quanti sono i termini del dizionario.
    3. Ogni termine, in base alla lunghezza, viene “preso” dal file di testo e inserito all’interno del vettore “paroledizionario” (se maggiore o uguale a tre) o nel vettore “paroledadue”.
    4. Viene chiuso il file di testo.
15. Void sistemmatrici (): questa funzione non riceve in ingresso alcun valore. Il suo compito è quello di riempire due array di array di interi che consentano di dire, per ogni cella del campo, se è possibile inserire una data parola in quella posizione oppure no; l’obiettivo è ancora una volta quello di velocizzare la ricerca della parola migliore. La funzione procede come segue:
    1. Vengono aperti tre cicli for, i primi due per navigare all’interno della matrice “libereorizzontali”, il terzo per contare quante celle libere ci sono a destra della cella corrente.
    2. All’interno di ogni cella viene inserito il numero di celle libere alla sua destra, fermandosi a un massimo di nove per velocizzare il ciclo.
    3. Analogamente, il procedimento viene ripetuto per la matrice “libereverticali” per contare le celle libere sotto a quella corrente.
16. Void selector (unsigned short color): questa funzione riceve in ingresso un intero senza segno, e sfruttando le API win32, in particolare con la funzione SetConsoleTextAttribute() si gestiscono i colori dell’output. Il codice ottiene un HANDLE (una "maniglia", un riferimento) alla console di output. Tramite quell’handle cambia le impostazioni di colore della console. Questo non è parte delle STL quindi è specifico per Windows.

**Note sulla realizzazione**

Librerie utilizzate:

1. <iostream>: è la libreria che definisce gli oggetti stream di input/output standard, come cin e cout.
2. <vector>: necessaria per utilizzare il tipo contenitore vector<T>.
3. <string>: necessaria per le operazioni di gestione delle stringhe.
4. <ctime>: questa libreria contiene definizioni di funzioni per ottenere e manipolare data e ora; noi ce ne serviamo per utilizzare la funzione time().
5. <cstdlib>: questa libreria definisce diverse funzioni di uso generale, compresa la generazione di numeri casuali grazie alle funzioni srand() e rand() che utilizziamo per simulare la pescata casuale di una lettera dal sacchetto.
6. <windows.h>: è un file header che “chiama” un insieme di librerie che servono a interfacciare il linguaggio con i sistema operativo, in particolare consente di usare le classi di API, che noi usiamo per gestire i colori dell’output.
7. <iomanip>: libreria che include i manipolatori parametrici; in particolare noi ci serviamo di setw per impostare la larghezza delle celle del campo nell’output.
8. <fstream>: questa libreria consente di comunicare con file esterni; noi utilizziamo la classe ifstream per leggere dal file di testo contenente il dizionario.

Per comodità utilizziamo “using namespace std;” per utilizzare il nome di determinati oggetti senza dover specificare e quindi ripetere il namespace di appartenenza, in questo caso la libreria standard.

Per generare un numero casuale abbiamo utilizzato la funzione rand() (contenuta in <cstdlib>) che genera un intero casuale compreso tra 0 e una costante RAND\_MAX (=2147483647). La funzione la chiamiamo dopo aver chiamato srand(time(0)); (per la quale bisogna includere <ctime>). I numeri così generati sembrano casuali, ma in realtà sono i millisecondi di distanza dal 1° gennaio 1970. Per ottenere dei numeri casuali da 0 a *n*, si utilizza rand()%n. srand(time(0)) è sufficiente dichiararla una volta sola.

Utilizzando la funzione selector si cambia il colore di testo e sfondo. In totale ci sono 256 combinazioni differenti tra colore dello sfondo e colore del testo. Noi abbiamo utilizzato:

selector (7) è il colore standard, cioè sfondo nero e testo grigio.

selector (12) sfondo nero e testo rosso.

selector (15) sfondo nero e testo nero.

selector (31) sfondo blu e testo bianco.

selector (47) sfondo verde e testo bianco.

selector (79) sfondo rosso e testo bianco.

selector (143) sfondo grigio e testo bianco.

system (“cls”); è la funzione che “pulisce” lo schermo, cioè cancella tutto ciò che è stato stampato fino a quel momento sulla finestra di output.

**Prove di test**

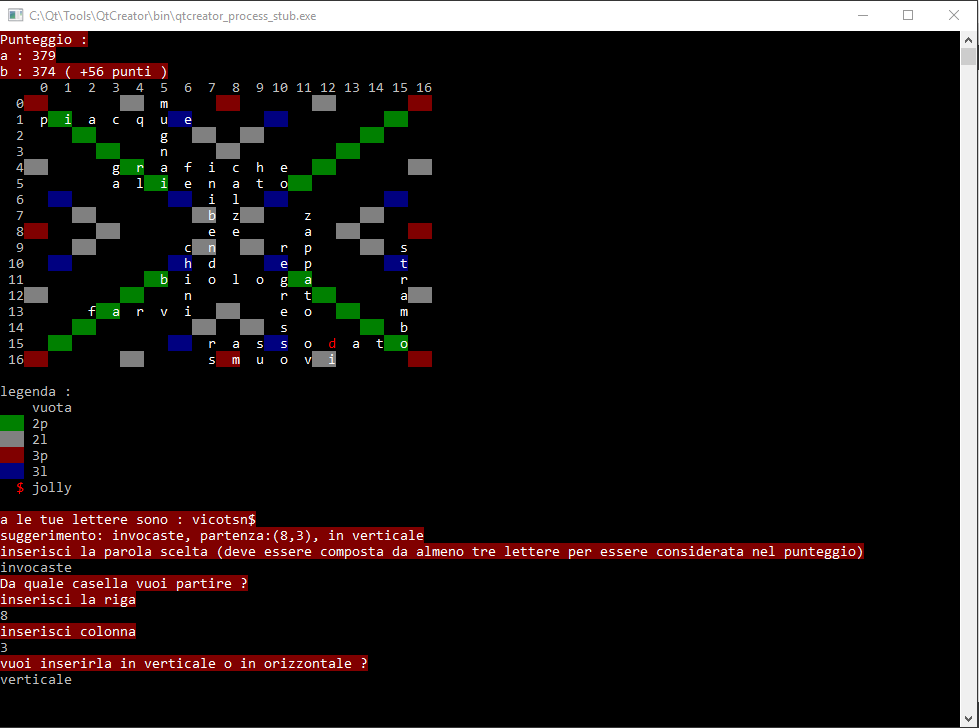
Ecco alcune schermate di esempio che mostrano particolari situazioni di gioco in cui ci si può venire a trovare durante l’esecuzione:

Prima situazione



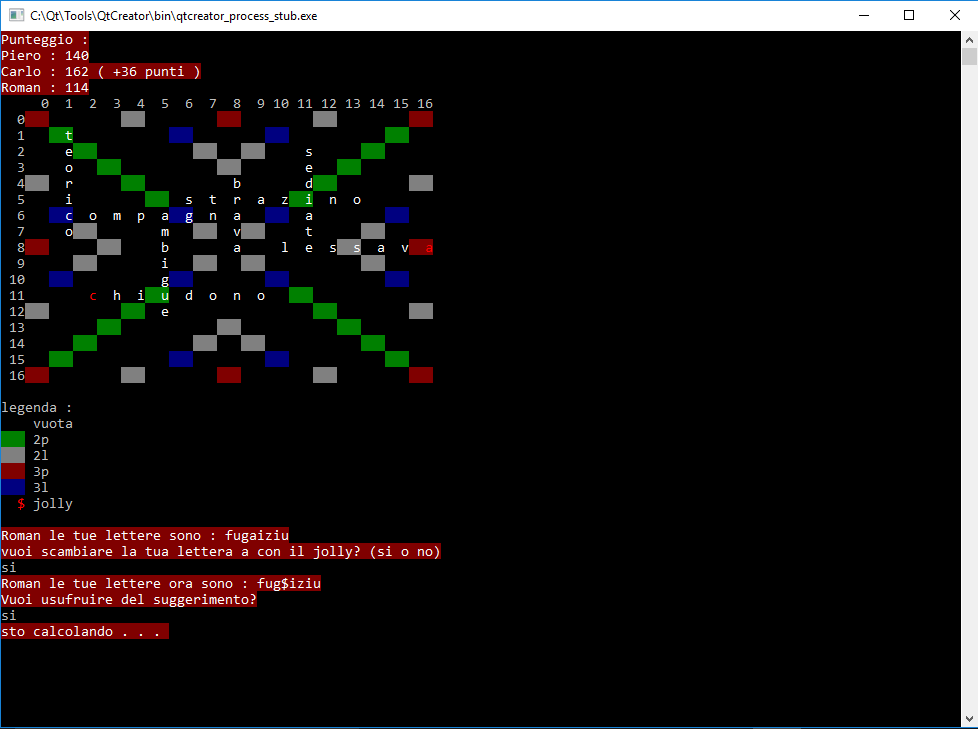
Questo è un esempio di come si presenta l’inizio del gioco e quindi il turno zero. Il calcolatore chiede all’utente quanti sono i giocatori e qual è il nome di ciascuno di essi. Stampa quindi il campo e dà inizio alla partita stampando le lettere del primo giocatore e chiedendogli se vuole usufruire del suggerimento.

Seconda situazione



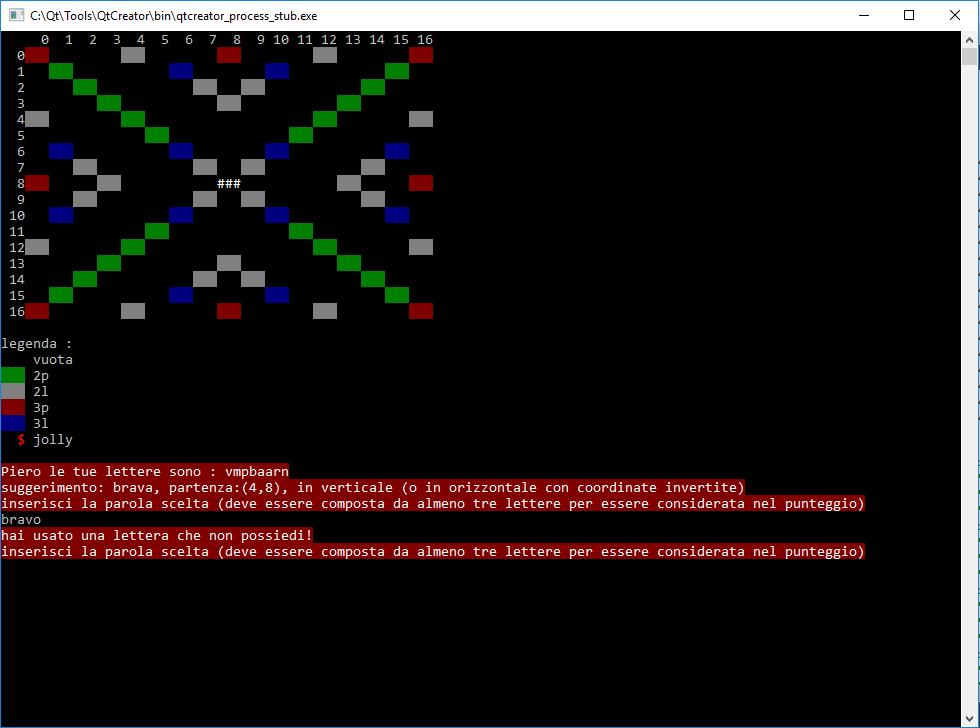
In questo caso vi è invece un turno generico della partita, in cui viene stampato prima il punteggio, poi il campo e infine le lettere del giocatore, il quale, in questo caso, ha deciso di usufruire del suggerimento. Procede quindi all’inserimento della parola che gli è stata consigliata.

Terza situazione



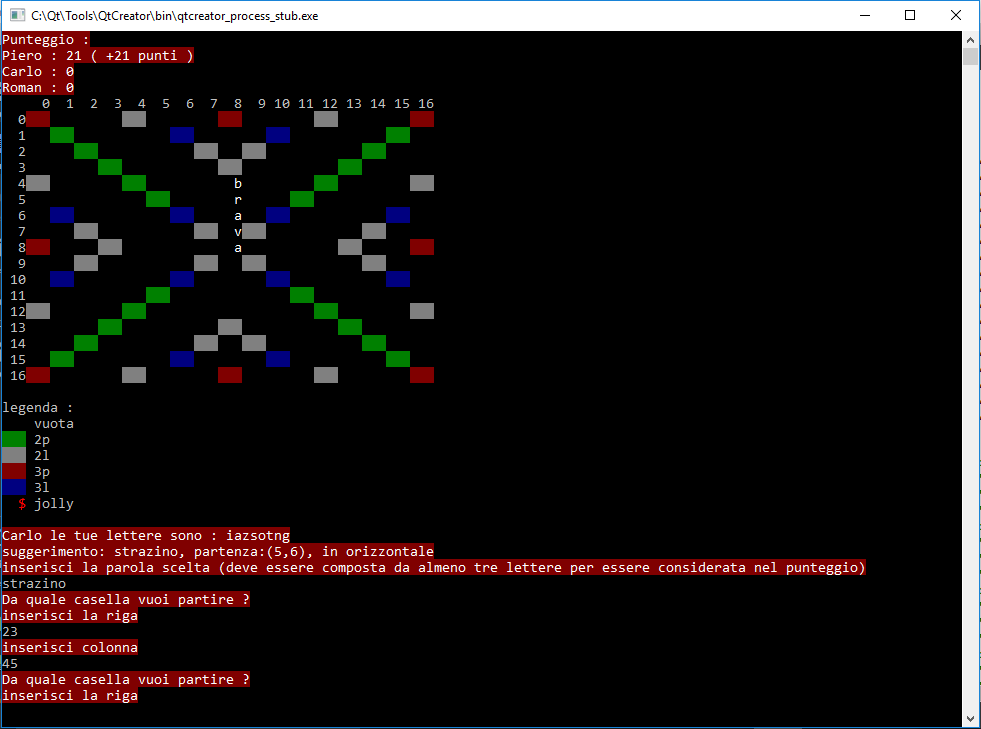
In questo caso si ha l’esempio di una situazione in cui il giocatore può appropriarsi di un jolly presente sul campo. Rispondendo “si” scambia la sua lettera con il jolly corrispondente; vengono perciò ristampate le sue lettere aggiornate. Decide poi di usufruire del suggerimento.

Quarta situazione



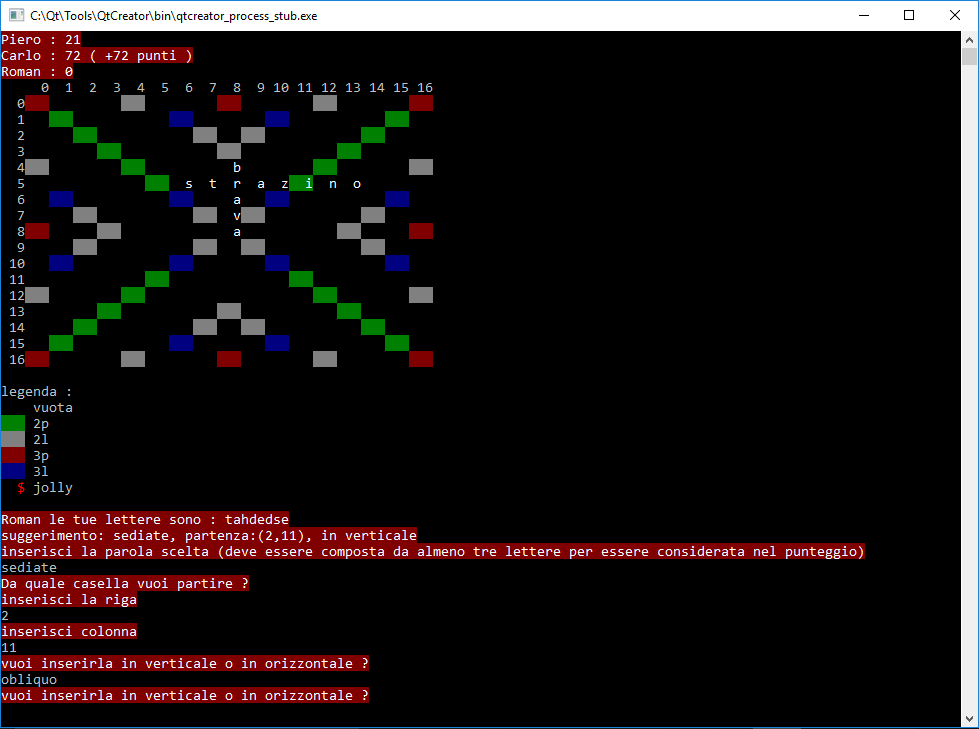
Ecco ora invece l’esempio di un errore che si può commettere durante il gioco, cioè quello di inserire una parola per cui non si hanno le lettere necessarie per comporla. Viene perciò stampato un avviso al giocatore e gli viene richiesta un’altra parola valida. Anche in questo caso il giocatore ha scelto di usufruire del suggerimento.

Quinta situazione



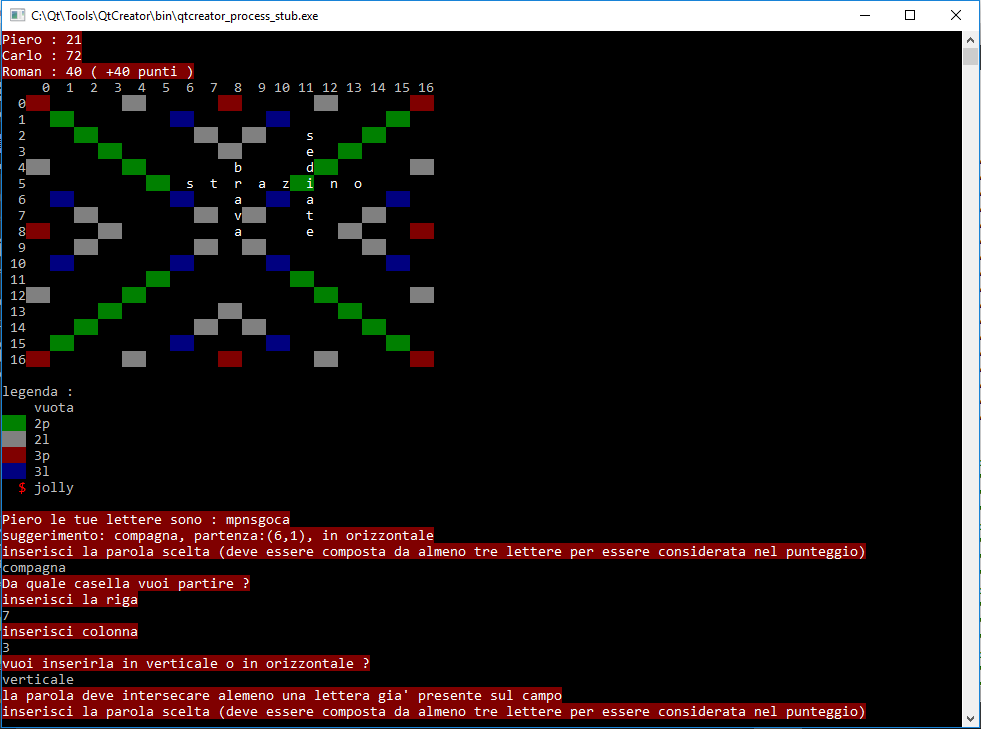
Un altro possibile errore può essere quello di inserire coordinate sbagliate, in questo caso il programma non ne permette l’inserimento e chiede quindi nuovamente le coordinate della cella da cui si vuole partire a inserire la parola.

Sesta situazione



In questo caso viene invece scritta una direzione sbagliata di inserimento, per questo motivo il programma, ancora una volta, non consente all’utente di procedere e richiede quindi una corretta direzione di inserimento.

Settima situazione



In questo caso il giocatore, seppur servendosi del suggerimento ha deciso di inserire la parola in un’altra posizione, tuttavia l’intersezione non è corretta. Viene quindi stampato un avviso e viene richiesta la parola da inserire con le coordinate corrette

Ottava situazione



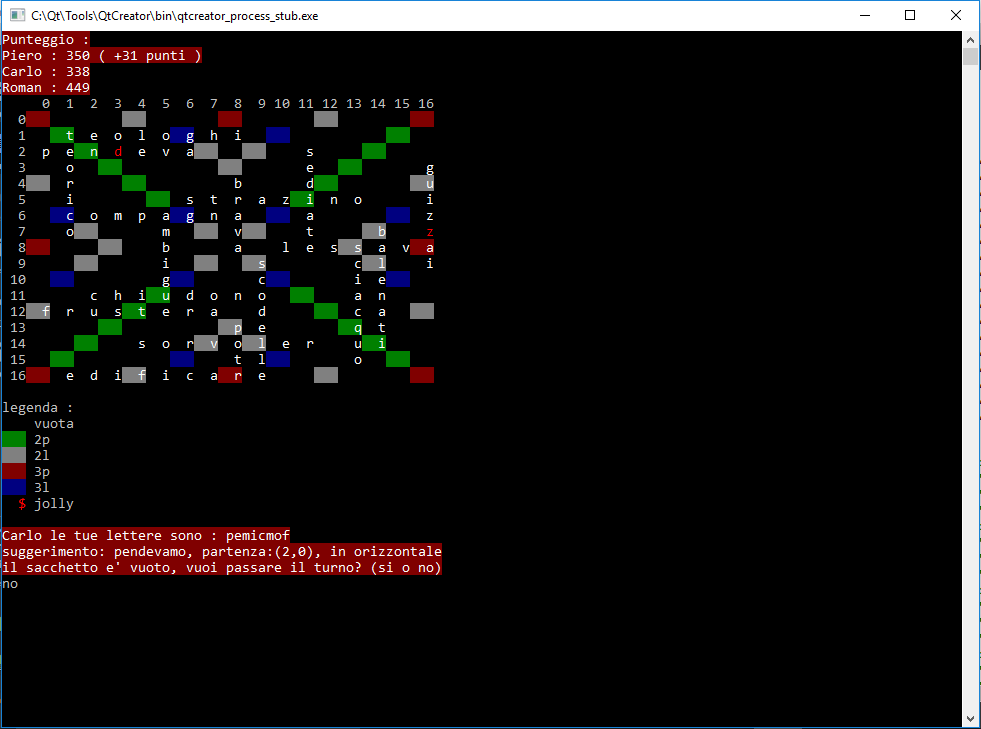
In questa situazione il giocatore ha provato a inserire una parola già presente sul campo e nella stessa posizione. Il programma ovviamente non glielo permette, perché per le regole del gioco, per inserire una parola, bisogna usare almeno una delle proprie lettere.

Nona situazione



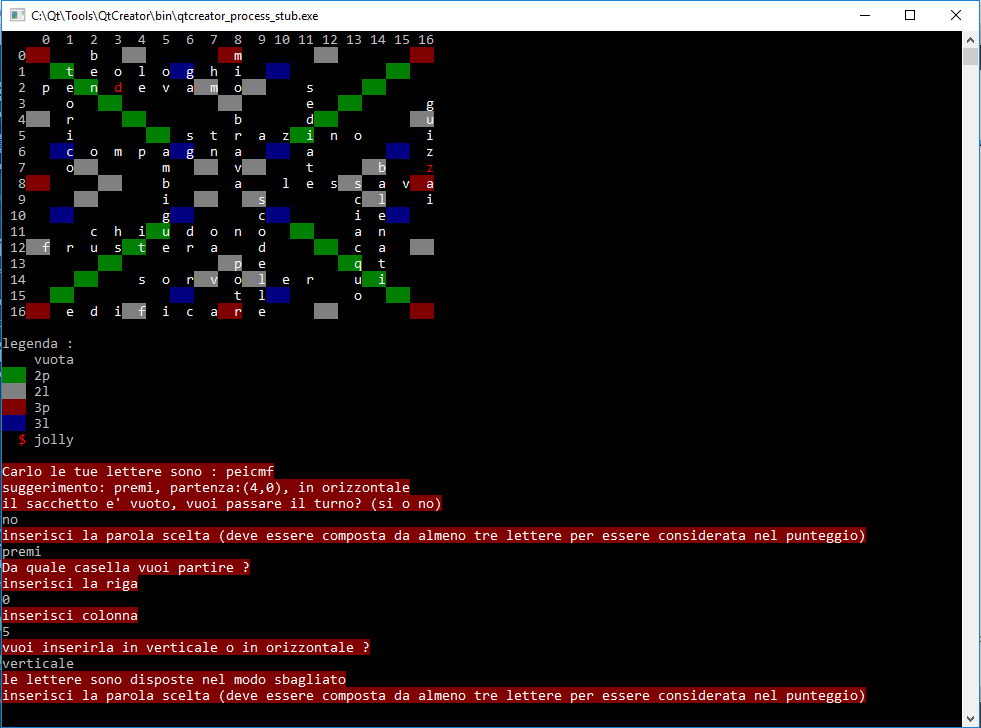
In questo caso il giocatore inserisce una parola in una posizione in cui se ne formano altre. Le parole che si formano a seguito dell’inserimento sono tuttavia errate in quanto non presenti all’interno del dizionario, perciò il programma non consente all’utente di proseguire, ma stampa un avviso e richiede la parola da inserire.

Decima situazione



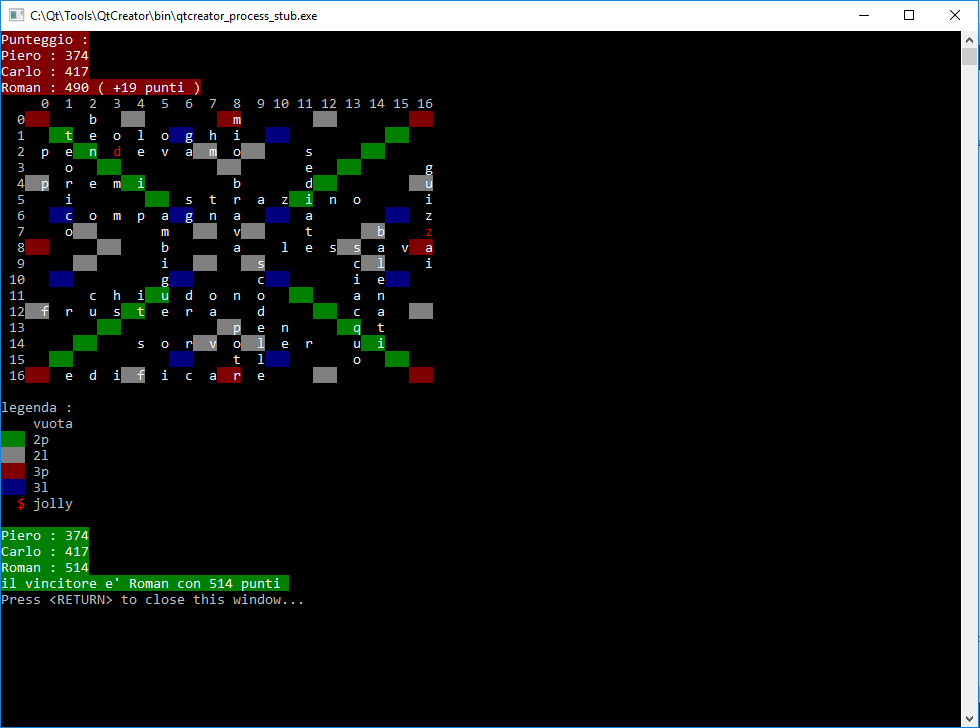
In questo caso la partita sta volgendo al termine in quanto il “sacchetto” è vuoto e quindi le lettere da distribuire sono finite. Da adesso in poi, ad ogni turno il programma chiederà al giocatore se vuole passare il turno in quanto è più probabile che non si riesca a comporre parole valide.

Undicesima situazione



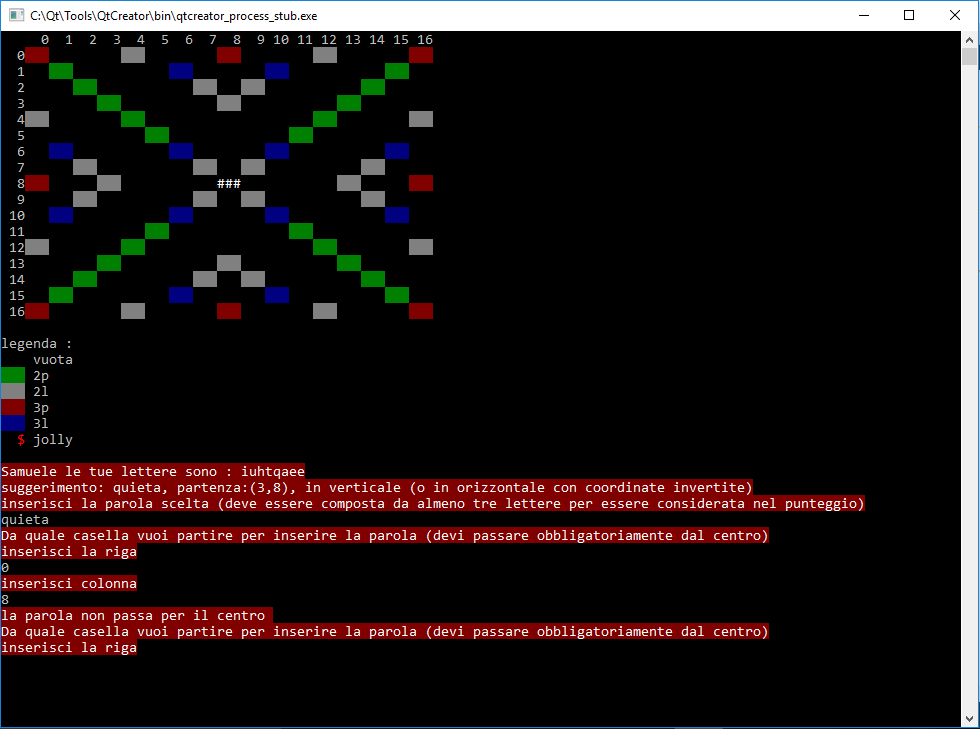
In questa situazione il gioco sta volgendo al termine e il giocatore sta cercando di inserire una parola in una posizione sbagliata, in quanto, come giustamente dice il programma, le lettere sarebbero disposte nel modo sbagliato; le intersezioni che si andrebbero infatti a formare sarebbero errate. Viene perciò richiesta la parola da inserire.

Dodicesima situazione



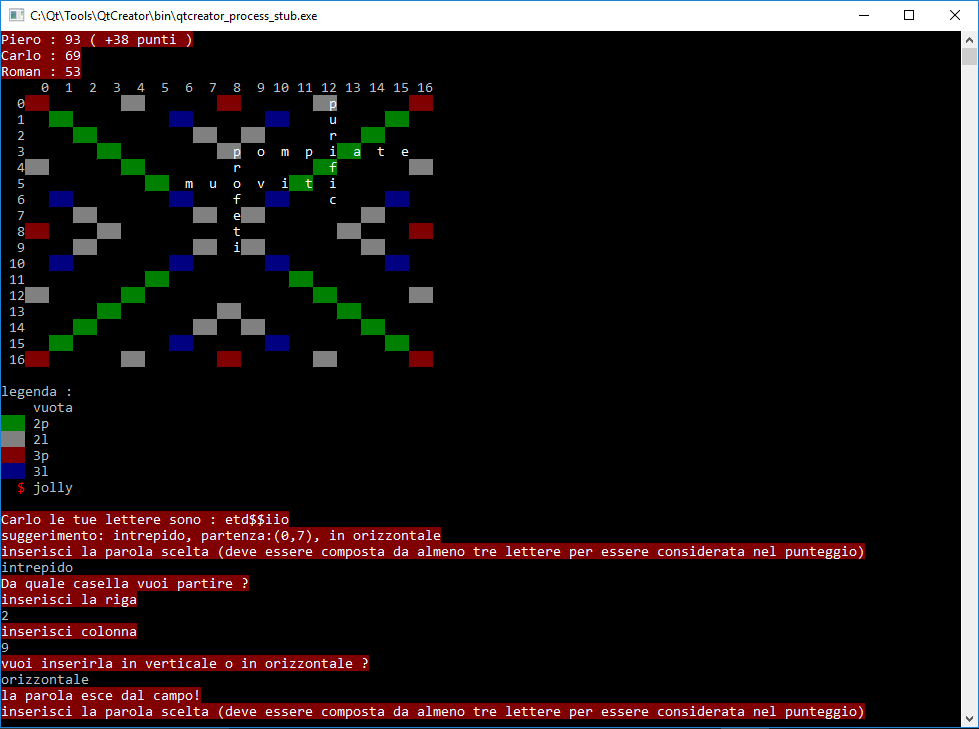
Questo è invece l’esempio della terminazione di una partita. Vengono infatti stampati i punteggi finali di tutti i partecipanti e infine viene stampato il nome del vincitore con il relativo punteggio.

Tredicesima situazione



Qui, ritornando all’inizio del gioco, vi è la situazione in cui il primo giocatore prova ad inserire una parola senza farla passare per il centro. Ovviamente ciò è vietato nelle regole del gioco, perciò il programma non glielo permette; stampa quindi un avviso e richiede le coordinate di inserimento della parola.

Quattordicesima situazione



In questo caso il giocatore inserisce una parola che, rispetto alla posizione di inserimento, risulta essere troppo lunga: essa uscirebbe infatti dal campo. Perciò il programma stampa un avviso e richiede al giocatore la parola da inserire.

Sedicesima situazione



In questo caso il giocatore riesce ad inserire una parola in una posizione che comporta la formazione di un’ulteriore parola valida. Il suo punteggio sarà quindi incrementato sia grazie ai punti forniti dalla parola da lui inserita, sia grazie ai punti forniti dalla parola che si forma.

Il risultato è il seguente:

