PROGETTO MASTERMIND

Estevao Luis Costa Moura de Luna
estevaoluis.costamouradeluna@studenti.unimi.it
matr. 774158

DESCRIZIONE

L'obiettivo del gioco è indovinare una sequenza predefinita di 4 colori.
I colori disponibili sono: rosso, giallo, verde, blu, bianco e nero.
I colori presenti nella sequenza sono distinti per posizione e possono essere ripetuti.
Ad ogni tentativo dell'utente il programma dirà quanti colori si trovano nella giusta posizione e quanti sono presenti nella sequenza obiettivo, ma si trovano nella posizione sbagliata.
Il giocatore vince quando ha indovinato la sequenza obiettivo.

FUNZIONAMENTO DEL PROGRAMMA

Per iniziare a giocare, l'utente dovrà inserire la seguente meta:

```
?- gioca.
```

Successivamente, l'utente potrà inserire la sua sequenza di 4 colori. I colori andranno scritti uno per riga con un punto alla fine di ciascuna riga, come nel seguente esempio di input:

```
|: verde.
|: rosso.
|: giallo.
|: blu.
```

Dopo aver inserito la propria sequenza, il programma comunicherà quanti colori si trovano nella giusta posizione e quanti colori sono presenti nella sequenza obiettivo, ma sono fuori posizione.
Esempio:

```
Posizioni corrette: 1
Fuori posizione: 2
```

Se l'utente ha indovinato la sequenza obiettivo, il gioco termina, altrimenti l'utente potrà effettuare un nuovo tentativo.

ESEMPIO D'ESECUZIONE

Supponendo che la sequenza obiettivo sia: verde, giallo, blu, giallo

```
1 ?- gioca.
L'obiettivo del gioco è indovinare la corretta sequenza di 4
colori
I colori presenti sono: rosso, giallo, verde, blu, bianco e nero
I colori si distinguono per posizione e possono essere ripetuti
Per fare un tentativo scrivi il nome del colore nel seguente
formato: colore.
e premi INVIO. Inserisci 4 colori e osserva i risultati
Buona partita!
|: rosso.
|: bianco.
|: nero.
|: rosso.
Posizioni corrette: 0
Fuori posizione: 0

Prova ancora:
|: blu.
|: verde.
|: giallo.
|: blu.
Posizioni corrette: 0
Fuori posizione: 3

Prova ancora:
|: giallo.
|: blu.
|: verde.
|: giallo.
Posizioni corrette: 1
Fuori posizione: 3

Prova ancora:
|: verde.
|: giallo.
|: blu.
|: giallo.
Complimenti! Hai indovinato!
true
```

CODICE
Fatti

Colori usati nel gioco:

```
colore(rosso).
colore(giallo).
colore(verde).
colore(blu).
colore(bianco).
colore(nero).
```

Sequenza (fissa) da indovinare:

```
sequenza_obiettivo([verde, giallo, blu, giallo]).
```

Regola di inizio gioco

Il corpo della regola contiene le istruzioni per scrivere a video le istruzioni del gioco e far partire il caricamento dei dati di input.

```
gioca:-
     writeln('L''obiettivo del gioco è indovinare la corretta
sequenza di 4 colori'),
     writeln('I colori presenti sono: rosso, giallo, verde, blu,
bianco e nero'),
     writeln('I colori si distinguono per posizione e possono
essere ripetuti'),
     writeln('Per fare un tentativo scrivi il nome del colore nel
seguente formato: colore.'),
     writeln('e premi INVIO. Inserisci 4 colori e osserva i
risultati'),
     writeln('Buona partita!'),
     input.
```

Input dei quattro colori

Predicato senza accumulatore

```
input:-
     input(4,[]).
```

Caso base, in cui i 4 colori sono già stati caricati, ora si verifica se la sequenza inserita dall'utente è uguale alla sequenza da indovinare

```
input(0, L):-
     !,
     sequenza_obiettivo(S),
     verifica(L,S).
```

Caso ricorsivo, in cui ci sono ancora colori da caricare, per ogni colore letto, viene effettuato un controllo per vedere se il colore inserito è uno dei colori riconosciuti dal gioco oppure no.

```
input(K, L):-
     read(C),
     colore(C),
     conc(L,[C],L1),
     K1 is K-1,
     input(K1, L1).
```

Operazioni sulle liste

Concatenazione

```
conc([],L,L).
conc([X|L1],L2,[X|L3]):-
     conc(L1,L2,L3).
```

Cancellazione

```
del(X,[X|L],L).
del(X,[Y|L1],[Y|L2]):-
     del(X,L1,L2).
```

Uguaglianza

```
uguali(L,L).
```

Verifica dell'uguaglianza tra la lista inserita dall'utente e quella da indovinare

In T viene istanziato il tentativo inserito dall'utente che sarà confrontata con S che è la sequenza da indovinare.

Per prima cosa verifico se le due liste sono uguali, in tal caso, il gioco finisce e il giocatore ha vinto.

```
verifica(T,S):-
     uguali(T,S),
     write('Complimenti! Hai indovinato!'),
     !.
```

Se invece sono diverse, si conteranno quanti colori si trovano nella posizione corretta e quanti colori della sequanza tentativo sono presenti nella sequenza obiettivo in qualsiasi posizione. Verranno poi mostrati all'utente i risultati, cioè i numeri dei colori nella posizione corretta e quelli presenti, ma nella posizione sbagliata. Infine verrà chiesto all'utente di fare un nuovo tentativo, inserendo un'altra sequenza.

```
verifica(T,S):-
     posizione_giusta(T,S,N1),
     presenza(T,S,N2),
     risultati(N1,N2),
     input.
```

Regole per il calcolo del numero di colori nella giusta posizione

```
posizione_giusta(T,S,N):-
     posizione_giusta(T,S,N,0).
```

Abbiamo regole del tipo posizione_giusta(T,S,N,ACC) in cui:
- T sarà istanziata alla sequenza tentativo del giocatore
- S sarà istanziata alla sequenza da indovinare
- N è la variabile che verrà istanziata al numero di colori nella giusta posizione
- ACC è l'accumulatore

Si scorreranno contemporaneamente le liste S e P e ogni volta che gli elementi in testa coincidono, il contatore verrà incrementato

Caso base, in cui abbiamo finito di scorrere le liste

```
posizione_giusta([],[],N,N):-
     !.
```

Caso in cui gli elementi in testa coincidono, il contatore viene incrementato

```
posizione_giusta([X|T],[X|S],N,ACC):-
     !,
     ACC1 is ACC+1,
     posizione_giusta(T,S,N,ACC1).
```

Caso in cui gli elementi in testa sono diversi, si confronteranno gli elementi successivi

```
posizione_giusta([X|T],[Y|S],N,ACC):-
     posizione_giusta(T,S,N,ACC).
```

Regole per il calcolo del numero di colori presenti nella sequenza obiettivo

```
posizione_giusta(T,S,N):-
     posizione_giusta(T,S,N,0).
```

Abbiamo regole del tipo presenza(T,S,S1,N,ACC) in cui:
- T sarà istanziata alla sequenza tentativo del giocatore
- S ed S1 sono le liste con la sequenza da indovinare. Si scorre S1 e si ripristina il valore con S
- N è la variabile che verrà istanziata al numero di colori presenti nella sequenza obiettivo
- ACC è l'accumulatore

Considero un elemento alla volta di T e vedo se questo è presente in S. Se l'elemento è presente, devo anche eliminarlo da S, per evitare che sia ricontato erroneamente nel caso in cui l'elemento sia presente ancora in T.

Caso base, in cui abbiamo finito di scorrere T.
N contiene il numero di colori della sequenza tentativo presenti anche nella sequanza da indovinare

```
presenza([],S,S1,N,N):-
     !.
```

Caso in cui un elemento di T è presente anche in S.
Il contatore verrà incrementato e l'elemento sarà cancellato da S per evitare di essere ricontato erroneamente nel caso in cui l'elemento sia presente ancora in T.

```
presenza([X|T],S,[X|S1],N,ACC):-
     !,
     del(X,S,S2),
     ACC1 is ACC+1,
     presenza(T,S2,S2,N,ACC1).
```

Caso in cui gli elementi confrontati sono diversi, considero l'elemento successivo di S

```
presenza([X|T],S,[Y|S1],N,ACC):-
     presenza([X|T],S,S1,N,ACC).
```

Caso in cui ho finito di scorrere S, considero il successivo elemento di T

```
presenza([X|T],S,[],N,ACC):-
     presenza(T,S,S,N,ACC).
```

Stampa dei risultati

Conoscendo il numero di colori nella corretta posizione (Pos_corrette) e il numero di colori presenti in qualsiasi posizione (Presenze), posso calcolare il numero dei colori presenti nella sequenza obiettivo, ma nella posizione sbagliata (Fuori_posizione) facendo semplicemente una sottrazione:
Fuori_posizione = Presenze – Pos_corrette.
Posso quindi mostrare al giocatore i valori di Pos_corrette e di Fuori_Posizione

```
risultati(Pos_corrette, Presenze):-
     write('Posizioni corrette: '),
     writeln(Pos_corrette),
     write('Fuori posizione: '),
     Fuori_posizione is Presenze - Pos_corrette,
     writeln(Fuori_posizione),
     nl,
     writeln('Prova ancora: ').
```