**Progetto in C: variante del Blackjack**

**Obiettivo:**

Realizzare una semplice variante del blackjack tra due giocatori (un giocatore e il dealer) in codice C insieme all’implementazione dell’algoritmo Hi-Low per il calcolo delle probabilità.

**Contesto:**

Si hanno a disposizione n mazzi ciascuno dei quali è costituito da 52 carte.

Ogni carta, per semplificare i calcoli, assume il suo valore corrispondente ad esempio: la carta numero 1 vale 1, la carta numero 2 vale 2 fino ad arrivare alle seguenti figure:

* Jack: 11
* Queen: 12
* King: 13

All’inizio del gioco, il mazziere distribuisce due carte a ciascuno dei due giocatori; il primo di essi ad avere, una buona probabilità di vincita (calcolata mediante alcuni accorgimenti matematici) vince.

Nel caso in cui nessuno dei due giocatori raggiunge una probabilità vincente, si procede con un nuovo match ricevendo dal mazziere altre due carte e si continua a giocare.

Il giocatore, quindi, riprenderà a calcolare la probabilità.

**Il metodo di calcolo della probabilità: Hi-Low**

Gli accorgimenti matematici che vengono adottati per calcolare la probabilità di vincita sono le seguenti:

1. Ad ogni carta verrà assegnata un punteggio diverso da quello che rappresenta:

* Alle carte con valore rappresentativo < 7, si assegna il punteggio di 1
* Alle carte con valore rappresentativo > 9, si assegna il punteggio di -1
* Alle carte escluse dall’intervallo qui citato, si assegna il punteggio di 0

1. Una volta ricevute le carte si procede al calcolo aritmetico dei punteggi (il cosiddetto running cont)
2. Si calcola quante sono le carte rimaste ancora nel mazzo mediante la seguente espressione:

**truecont = (RunningCont / cR)**

Dove cR indica il numero di carte espresse in mazzi.

1. Infine, si calcola la probabilità con la seguente espressione:

**p = truecont \* 100**

**Esempio:**

Supponiamo che all’inizio del gioco ci siano 6 mazzi e che siano uscite le seguenti carte/figure:

7 8 K Q JACK 3 2 1 4 2

A questo punto il runningCont sarà:

runningCount = 0+0-1-1-1+1+1+1+1+1 = 2

Le carte rimanenti nel mazzo saranno pari a:

52(numero di carte per mazzo) \* 6 (numero di mazzi) – 10 (numero di carte uscite) = 302

Le carte espresse in mazzi sono:

302 / 52 = 5.80

Il truecont sarà uguale a:

truecont = runningCont / 5.80 = 0.34

Infine, la probabilità sarà data da:

p = 0.34 \* 100 = 34%

Ovviamente maggiore sarà il valore del truecont più alta sarà la probabilità di vincita.

**Come si è proceduti in C**

Ai due competitori che sono un giocatore e il banco, è assegnato un thread (consumer1, consumer2).

Al banco è poi stato affidato un altro thread ovvero il producer.

Analizzeremo ora alcune sezioni di codice.

**Funzioni chiamate durante il programma:**

Le funzioni chiamate durante l’esecuzione del programma sono:

* **conteggio**

è una funzione di tipo intera che possiede il compito di calcolare e restituire il runningCont.

Alla funzione vengono trasferite i seguenti parametri:

* Carta (valore della carta appena estratta)
* Truecont di ciascun consumer

Possiede alcune strutture decisionali basate sulla condizione if che consentono in base al valore delle carte il calcolo del runningCont.

* **calcolaprob**

Si tratta di una funzione di tipo float che permette di calcolare e restituire la probabilità trasferendo come parametro solamente il valore del runningCont.

* **produci\_elementi**

E’ una funzione di tipo void che contiene:

* Meccanismo di generazione dei valori (carte) e conseguente immissione nel buffer di memoria.
* Parte di codice “critica” dove è presente la variabile condivisa dai threads che tiene conto del numero di elementi presenti nel buffer *num\_elementi\_presenti\_buffer*
* **consuma\_elementi**

E’ una funzione di tipo void che contiene:

* Meccanismo di consumo dei valori (carte) e conseguente estrazione dal buffer di memoria.
* Parte di codice “critica” dove è presente la variabile condivisa dai threads che tiene conto del numero di elementi presenti nel buffer *num\_elementi\_presenti\_buffer*

**Main e sezione di codice associati ai threads**

* **Corpo del main**:

È costituito dalla definizione dei threads e dall’acquisizione del numero di mazzi.

* **Producer (parte di codice affidata al thread1)**

Ha il compito di generare casualmente dei numeri interi che vanno da 1 a 13 mediante la funzione rand( ).

Si è scelto appositamente di non utilizzare una procedura che eviti le ripetizioni dei numeri perché realisticamente, i mazzi possono contenere carte dello stesso valore (si pensi ad esempio di carte di seme diverso)

Nel codice associato al thread1 è presente:

* **Un controllo sul numero di elementi presenti nel buffer.**

Grazie al ciclo while possiamo verificare se il buffer è pieno tramite la seguente condizione:

num\_elementi\_presenti\_buffer == LUNGHEZZA\_BUFFER

se questa condizione è vera significa che il buffer è pieno e quindi la sezione del codice affidata al thread producer si “bloccherà” al ciclo while fintanto che uno dei due consumer estrae un valore dal buffer (carta) e di conseguenza modificherà il valore di *num\_elementi\_presenti\_buffer.*

* **Una duplice chiamata della funzione *produci\_elementi* racchiusa all’interno del meccanismo del mutex lock.**

La duplice chiamata della funzione assicura la produzione di due valori (carte) per volta.

Mentre il mutex lock assicura un controllo sulla variabile condivisa *num\_elementi\_presenti\_buffer.*

Dato che questa variabile viene a essere modificata continuamente dai thread, essa rappresenta una sezione di codice critica.

il consumer e il producer sono concorrentemente in esecuzione quindi entrambi i thread modificherebbero il valore del contatore nello stesso momento.

In questo modo si permette la modifica del contatore solo ad un thread alla volta.

* **Due strutture decisionali**

che obbligano ad arrestare il producer nel caso in cui uno dei due giocatori ottiene una probabilità vincente superiore al 100% o il producer ha esaurito i valori (carte) da produrre.

* **Consumer**

Il consumer rappresenta uno dei due giocatori partecipanti.

All’interno di questa sezione di codice affidata ai threads 2 e 3 è presente:

* **Struttura decisionale if**

Non permette di eseguire il codice affidato al thread consumer qualora il buffer sia vuoto o il producer ha finito di produrre valori (carte)

* **Ciclo while**

Non permette di consumare (eseguire il codice) qualora il buffer sia vuoto.

* **Una duplice chiamata della funzione *consuma\_elementi* racchiusa all’interno del meccanismo del mutex lock.**

La duplice chiamata della funzione assicura la consumazione/estrazione di due valori (carte) per volta.

Mentre il mutex lock assicura un controllo sulla variabile condivisa *num\_elementi\_presenti\_buffer.*

* **Altre strutture if alla fine del ciclo while (true)**

Terminano il thread nel caso in cui uno dei due consumer abbia una probabilità vincente (> 100%)

Oppure i due giocatori non hanno raggiunto una buona probabilità anche se si sono scartate tutte le carte disponibili.