Presentazione codice sorgente

1. Strategia algoritmica: Innanzitutto per la risoluzione del problema ho utilizzato la strategia “Divide et impera” cercando di dividere il problema in sotto problemi: prima la lettura delle informazioni da std input, poi la verifica dell’avvenimento delle reazioni e così via al fine di arrivare alla soluzione finale
2. Scelta della struttura dei dati: Sulla base dell’analisi del problema ho deciso di rappresentare la prima riga, riguardo alla configurazione iniziale, come un vettore di stringhe mentre nel caso delle reazioni ho definito un record utilizzando struct, i campi sono l’identificatore della reazione, i reagenti, l’inibitore, i prodotti, il numero di avvenimenti tot e il numero di avvenimenti ad un certo passo. (questi ultimi due campi risultano utili nel main). Per acquisire le informazioni sulla configurazione iniziale e sulle reazioni ho utilizzato due funzioni.
3. Spiegazione funzioni: la prima funzione da me creata utilizza la funzione fgets e una matrice. Il suo funzionamento è il seguente: i caratteri letti vengono assegnati alla prima riga finché non si incontra uno spazio, segnale di cambio riga, si utilizza infine la funzione atoi per convertire i caratteri in interi.

Per la lettura delle informazioni riguardo alle reazioni ho utilizzato una funzione pressoché identica per semplicità. In questo caso avrei potuto anche utilizzare la funzione sscanf a causa della struttura fissa delle reazioni.

Ho inoltre utilizzato una funzione per verificare se la reazione avviene o meno confrontando il contenuto della riga iniziale con le informazioni ottenute sulle reazioni.

1. Risultato finale, bubblesort e eliminazione doppioni: A questo punto sorge il problema del risultato finale inteso come prodotti ottenuti, in quanto essi possono essere rappresentati come elementi di un vettore il quale però ha lunghezza variabile e può contenere elementi ripetuti. Per far fronte a tale problematica ho deciso di implementare una funzione che innanzitutto riordinasse il vettore tramite un algoritmo del tipo “Bubblesort”. Tale algoritmo non è il migliore in termini di costo computazionale (è un O(n^2)) ma utilizza meno memoria di algoritmi quali quello di quicksort, ragione per cui ho deciso di utilizzare il bubblesort.

Dopodiché nella stessa funzione ho fatto in modo che qualora si presentassero dei doppioni venissero scambiati con l’ultimo elemento e si riducesse di uno la lunghezza del vettore eliminando il doppione e riallocando la memoria per il nuovo vettore. Tale funzione è ricorsiva in quanto possono essere presenti più doppioni.

1. Note sul main: Nel main ho utilizzato tutte queste funzioni in modo da ottenere il risultato corretto acquisendo da riga di comando il numero di step della simulazione utilizzando ancora una volta la funzione atoi.
2. Dettagli e accorgimenti: Si nota che si sarebbe semplificato di molto il problema qualora le informazioni fossero prese da un file esterno utilizzando la funzione fopen. Inoltre l’utilizzo dei doppi puntatori permette di modificare le variabili dentro una funzione, cosa che non sarebbe possibile altrimenti visto il passaggio degli argomenti per valore nelle funzioni.