Magazzino G.C. 3BITI

Dividere il programma in sotto problemi più piccoli.

## Struttura del FILE di memorizzazione **magazzino.txt**:

Esempio struttura e dati memorizzati:

|  |  |  |  |
| --- | --- | --- | --- |
| Categoria | Nome prodotto | Prezzo | Quantità |
| Telefono | iPhone 12 | 999 | 10 |
| Scheda Video | Nvidia RTX 3080 | 2000 | 0 |
| Notebook | Asus ROG Strix | 2500 | 2 |

## Problemi subito da risolvere nella memorizzazione e lettura dati in ingresso:

Risolvere il problema degli spazi utilizzando le soluzioni sperimentate in precedenza per la lettura (aggiunta trattini \_ al posto degli spazi durante la scrittura su FILE o dopo lettura input utente, e rimozione nella lettura per mostrare i valori), per il resto la scrittura sarà effettuata per i vari elementi della struct successivamente.

Ad ogni riga sarà inserito un prodotto.

## Funzioni principali programma:

1. Inizializzazione nuovo FILE, questa opzione servirebbe per cancellare l’intero magazzino o crearne uno nuovo, nei log sarà salvato come -Reset-.
2. Aggiungi prodotto al magazzino, questa funzione servirebbe per aggiungere un nuovo prodotto al magazzino, includerebbe anche ricerca e in caso di aggiunta di un prodotto già aggiunto e quindi incremento della sua quantità precedente con quella nuova, nei log sarà salvato come -**Aggiunto**-.
3. Modifica prodotto magazzino, questa funzione servirebbe per modificare un prodotto nel magazzino, ove disponibile, in caso sia assente, comunicare all’utente della sua mancata esistenza, sarà comunicato come -----**Modificato-**, un menu sarà provvisto per la modifica.
4. Eliminazione prodotto magazzino, questa funzione servirebbe per eliminare un prodotto dal magazzino ove presente, in caso di assenza sarà comunicato all’utente, sarà aggiunta come nota nei log -**Eliminato**-.
5. Lettura valori nel magazzino intero.
6. Lettura prodotti per categoria nel magazzino, nel caso esso sia disponibile, altrimenti comunicherò con un messaggio la mancanza di prodotti nella determinata categoria.
7. Vendita prodotto, verifica se un prodotto esiste nel magazzino e se è disponibile e nel caso esista, sottrarre la quantità venduta e scrivere nei log/file modifiche l’opzione -**Venduto**-.
8. Imprevisto e perdita di un prodotto, è possibile aggiungere una nota personalizzata che andrà nei log per questa opzione e modificare il prodotto, previe verifiche, in caso di mancanza di nota personalizzata sarà scritto -**Imprevisto**-.
9. Leggi informazioni prodotto o verifica se è disponibile, non saranno modificati dati quindi questa operazione non sarà salvata nei log.

# Inserimento dati:

Saranno richiesti dopo aver scelto una funzione, determinati dati all’utente.

Categoria e Nome Prodotto utilizzeranno un getline per prendere valori anche con eventuali spazi.

Tutti gli altri elementi avranno bisogno di un semplice scanf per leggere un numero, di vario tipo come double per prezzo oppure int per quantità.

## FILE DI LOGS che salva la storia delle modifiche effettuate nel magazzino:

Ad ogni utilizzo con successo di una funzione, saranno salvate dei log su un file **modifiche.txt**, in una struct doppiamente grande rispetto a quella originale, ossia di 8 elementi, che salva i valori precedenti e i valori nuovi.

Per esempio, ho modificato DA -> A:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Telefono | iPhone 12 | 999 | 10 | Aggiunto |
| Telefono | iPhone 12 | 990 | 9 | Venduto |

Perché eventualmente un prodotto in meno è disponibile nel magazzino e c’è stato un piccolo calo di prezzo generale, ma possono anche essere modifiche maggiori.

La struttura può anche solamente essere un riassunto dell’operazione, in versioni più avanzate, il formato DA -> A sarà modificato in OPERAZIONE.

L’ultimo elemento sarà aggiunto, alla seconda struct, è opzionale e solamente in alcune opzioni sarà aggiunta.

Comunque sarà salvato nel FILE di log una riga, con 8 elementi, i primi 4 sono il valore iniziale, e gli altri 4 quelli nuovi.  
Eventualmente, sarà aggiunta una funzione che verifica i dati che sono cambiati rispetto al valore originale, e che quindi aggiunge un carattere speciale per identificare il valore che è cambiato, come un \*, o comunque qualcosa per identificare la modifica.