ANALISI SARTORIA

Sommario

Programma

Realizzare un programma in linguaggio **C** per la gestione di una **sartoria**, focalizzato sul controllo e l'amministrazione dei **rotoli di tessuto** e dei **progetti di produzione**.

Il sistema deve consentire l'inserimento, la modifica e l'eliminazione di tessuti e progetti, la gestione dell'inventario secondo una logica **FIFO (First In, First Out)** e il calcolo dei costi e dell'usura dei materiali.

Analisi del contesto

La sartoria lavora con diversi rotoli di tessuto salvati in inventario[], ognuno con le sue caratteristiche: tipo, colore, fantasia, misure, codice fornitore, costo e usura.
Ogni scheda contiene anche la quantità disponibile, l'uso previsto, gli scarti e la data di acquisto.

I progetti usano più rotoli grazie alla struct:

```
struct rotoli {
    char rotolo_richiesto[MAXSTRING];
    float quantita_richiesta;
};
```

Ogni progetto contiene fino a MAXP rotoli richiesti, un costo, il tipo di capo, valori economici (paga e ricavi), e un flag per i mini-progetti che usano scarti.

Quando si taglia del materiale, l'usura aumenta, se supera il 50%, il rotolo non è più valido.
Se un rotolo finisce durante un taglio o un calcolo, viene ricomprato tramite riacquista(), usando lo stesso costo e le stesse dimensioni.

La sartoria ha un budget iniziale (BUDGETINIZIALE), il budget può scendere sotto zero, quindi la bancarotta è possibile.

Ci sono due file; Inventario.txt e Progetti.txt, questi file salvano tutti i dati in forma testuale usando fprintf() e fscanf().

L'organizzazione dell'inventario segue la **FIFO**: le schede vengono ordinate per data di acquisto, e la più vecchia è la prima a essere usata.

Il sistema deve gestire: tessuti, inventario, progetti, budget e scarti

Ipotesi

- Ogni rotolo ha un costo e, quando finisce, viene ricomprato con lo stesso costo e la stessa lunghezza originale.
- L'usura cresce a ogni taglio e arriva fino al 50% massimo.
- La sartoria ha un budget che può andare in negativo.
- Gli scarti sono salvati come pezzi interi nella scheda del rotolo.
- I mini-progetti consumano un certo numero di scarti.
- Il fornitore è salvato come stringa.
- La costante SCARTI viene usata per stabilire quanti scarti si producono dal taglio.

Struttura dati

Rotolo

- tipo tessuto
- colore
- fantasia
- lunghezza in metri
- larghezza in cm
- codice fornitore
- costo
- usura

Scheda inventario

- codice rotolo
- fornitore
- rotolo (struct rotolo)
- data_acquisto
- quantita_disponibile
- utilizzo_previsto
- scarti_utilizzabili (pezzi interi)

Progetto

- nome progetto
- array rotoli_richiesti[MAXP]
- rdim (quanti rotoli usa realmente)
- costo_approssimato
- mini (1 o 0) 0 = Normale, 1 = usa scarti
- scarti_richiesti
- tipoCapo

- paga
- ricavi
- valore

Requisiti

Requisiti Must:

1. gestione rotoli
2. gestione progetti multipli per ogni progetto
3. gestione inventario FIFO
4. calcolo costi
5. taglio dei materiali

Requisiti Should:

6. usura
7. salvataggio e caricamento file
8. calcolo costi

Requisiti May:

9. gestione budget
10. ricavi e paghe dei progetti
11. Informazioni fornitori

Costanti

MAXSTRING 30

- Dimensione massima delle stringhe

MAXTESSUTI 100

- Numero massimo di schede con i relativi tessuti

MAXPROGETTI 20

- Numero massimo di progetti

BUDGETINIZIALE 1000

- Budget iniziale

FILEINVENTARIO “Inventario.txt”

- Nome del file usato per salvare l’inventario

FILEPROGETTI “Progetti.txt”

- Nome del file usato per salvare i progetti

MAXP

- Max di rotoli del progetto

Scarti

- Calcolo degli scarti

Structs

Strutture dati:

```
struct rotolo{
        char tipo_tessuto[MAXSTRING];
        char colore[MAXSTRING];
        char fantasia[MAXSTRING];
        float lunghezza;
        float larghezza;
        char codice_fornitura[MAXSTRING];
        float costo;
        float usura;
};
struct scheda{
        char codice_rotolo[MAXSTRING];
        char fornitore[MAXSTRING];
        struct rotolo rot;
        int g,m,a;
        float quantita_disponibile;
        float utilizzo_previsto;
        int scarti_utilizzabili;
} inventario[MAXTESSUTI];
```

```
struct progetto{
        char nome_progetto[MAXSTRING];
        char rotolo_richiesto[MAXSTRING];
        float metraggio_richiesto;
        float costo_approssimato;
        int mini;                   // se è un mini progetto verranno usati gli scarti
        float scarti_richiesti;
        float paga;
        float ricavi;
        float valore;
} progetti[MAXPROGETTI];
```

Azioni sui dati

Gestione Rotoli

Inserimento

- crea una nuova scheda in inventario[]
- legge tutti i campi del rotolo
- inizializza usura a 0
- aggiunge data di acquisto
- incrementa RCount

Modifica

- cerca il rotolo tramite codice
- aggiorna i valori richiesti

Eliminazione

- elimina la scheda
- compatta l'array
- decrementa RCount

Gestione Progetti

- crea un nuovo progetto in progetti[]
- legge nome, tipo capo, mini (si o no)

- legge rdim (quanti rotoli usa)
- compila rotoli_richiesti[] con nome rotolo e quantità
- se mini legge scarti
- calcola costo approssimato e valore
- legge paga
- incrementa PCount

Modifica

- modifica nome, rotoli, quantità, paga
- ricalcola costo e valore

Eliminazione

- rimuove progetto
- compatta array
- decrementa PCount

Calcolo costi progetto

per ogni rotolo richiesto:

- trova rotolo nell'inventario
- calcola costo proporzionale al metraggio richiesto

se un rotolo finisce:

- chiama riacquista()
- diminuisce budget

aggiorna costo_approssimato

Avvio taglio

per ogni rotolo del progetto:

- scala le quantità seguendo FIFO
- aumenta usura
- assegna scarti con assegnaScarti()

se un rotolo finisce → riacquista()

se mini-progetto → scala scarti

aggiorna budget con paga/ricavi

elimina progetto completato

Controllo tessuti

Segnala rotoli con:

- usura alta
- poca quantità
- vecchia data (FIFO)

Rotazione scorta (FIFO)

- ordina inventario per data
- mantiene coerenza FIFO

Usura

- aumenta a ogni taglio
- se supera 50% → rotolo inutilizzabile

Gestione scarti

- assegnati con SCARTI
- salvati in scarti_utilizzabili
- mini-progetti consumano scarti

Gestione budget

- acquisti rotoli → uscita
- completamento progetto → entrata
- budget può diventare negativo

Salvataggio dati

- **salvaInventario()**
  - Salva *RCount*, *PCount*, tutte le schede, tutti i rotoli, tutte le variabili nei file .txt.
- **caricaInventario()**
  - Legge dai file e ricostruisce inventario e progetti.

Visualizzazione

- Stampa tessuti, stato rotoli, progetti, quantità e usura.

Funzioni

```
int menu(int filtro);                                   // mostra menu e gestisce scelta principale
int nuovoRotolo(int *RCount);                           // inserisce un nuovo rotolo in inventario
int modificaRotolo(int index, char codice[]);           // modifica un rotolo usando indice e codice
int eliminaRotolo(int *RCount, char codice[]);          // elimina un rotolo usando codice e aggiorna
                                                        conteggio
int nuovoProgetto(int *PCount, int RCount);             // crea un nuovo progetto
int modificaProgetto(int index, char nome[], int        // modifica un progetto usando nome e rotoli
RCount);                                                disponibili
int eliminaProgetto(int *PCount, char nome[]);          // elimina un progetto usando nome
float calcolaCostoProgetto(int index, int               // calcola costo del progetto indicato
RCount);
int mostraProgetti(int *PCount, int RCount);            // mostra tutti i progetti
int avviaTaglio(int *PCount, char nome[], int           // avvia il taglio per un progetto
RCount);
int mostraTessuti(int RCount);                          // mostra tutti i tessuti in inventario
int controlloTessuti(int RCount);                       // controlla rotoli (usura, quantità, problemi
                                                        vari)
int rotazioneScorte(int RCount);                        // ordina l'inventario secondo FIFO
float aumentoUsura();                                   // aumenta usura dei rotoli
void salvaInventario(int RCount, int PCount);           // salva inventario e progetti su file
void caricaInventario(int *RCount, int                  // carica dati dai file
*PCount);
void reset(int *RCount, int *PCount);                   // resetta inventario e progetti
void aggiorna(int RCount, int PCount);                  // aggiorna valori dopo operazioni
int assegnaScarti(float metraggio);                     // calcola scarti generati da un taglio
void riacquista(int index);                             // riacquista un rotolo esaurito
```

Variabili

Lista variabili (potrebbe cambiare durante la fase di sviluppo):

Dove	Nome	Cosa	Tipo	Valore
Globale	Finv	file usato per salvare l'inventario	FILE *	-
Globale	FProg	file usato per salvare i progetti	FILE *	-
Globale	budget	budget della sartoria	float	1000
main	RCount	tiene conto di quanti rotoli ci sono	int	0
main	PCount	tiene conto di quanti progetti ci sono	int	0
main	scelta	scelta effettuata in `menu()`	int	-
main	val	usata per leggere i dati per poi convertirli	char[ ]	10
main	err	riceve i codici di errore dalle funzioni	int	-
main	filter	usata come filtro in alcune funzioni	char[ ]	30
menu()	s1	variabile in `menu()`: prima scelta	int	-
menu()	s2	variabile in `menu()`: seconda scelta	int	-

Struttura file

La struttura dei file sarà organizzata in questo modo:

Dati

Rcount=1

Pcount = 1

inventario[0]:

- rotolo1
- Rotoli&Co
- dati rotolo…
- 10/08/2024
- 10 M^2
- 7
- 0

progetti[0]:

- progetto1
- rotolo1
- 5
- 30€
- 0
- 0

File:

```
1 1
rotolo1 Rotoli&Co [dati rotolo] 10 08 2024 10 7 0
progetto1 rotolo1 5 30 0
```

Le variabili verranno separate da spazi e successivamente lette dalla funzione fscanf()