

Gestione Inventario Prodotti (Pila e Array)

Prova d'Esempio — main.c

Università di Catania — Dipartimento di Matematica e Informatica

Corso di Laurea in Informatica

Programmazione 1 e Laboratorio (Canale O-Z)

Descrizione del Programma

Si scriva un programma in linguaggio C, chiamato `main.c`, per la gestione di un inventario di prodotti commerciali. Ciascun prodotto è caratterizzato da un codice identificativo numerico, un nome descrittivo, la quantità disponibile e il prezzo unitario.

Il programma deve ricevere i file di input e output tramite riga di comando, caricare i dati in una struttura dinamica di tipo pila, svuotare la pila trasferendo i dati in un array dinamico in cui effettuare ricerche e modifiche, e infine ordinare e salvare i risultati in un file CSV formattato, restituendo opportuni codici di terminazione.

Requisiti di Implementazione

Il programma deve essere strutturato nei seguenti punti:

PUNTO A: [3 punti] Gestione Parametri (CLI): Validare i parametri da riga di comando.

- Il programma deve ricevere esattamente due argomenti: il nome del file di input (es. `prodotti.csv`) e il nome del file di output (es. `report.csv`).
- Entrambi i file devono avere estensione `.csv`. In caso di errore (numero errato di parametri o estensione non valida), stampare un messaggio su `stderr` e terminare con exit code 1 (per numero parametri errato) o 2 (per estensione errata).
- Stampare a schermo i nomi dei file confermati.

PUNTO B: [8 punti] Struttura ADT Pila e Inserimento:

- **Punto 2a [4 punti]:** Definire la struttura `Prodotto` (ID intero, nome stringa max 100 caratteri, quantità intera, prezzo float) e il nodo `Node` per gestire una Pila dinamica (LIFO).

- **Punto 2b [4 punti]:** Aprire il file di input, leggerne le righe ed effettuare il parsing dei campi (ID, nome, quantità, prezzo), popolando la pila dinamica tramite operazioni di push nell'ordine in cui sono presenti nel file. Se il file non può essere aperto, stampare un errore ed uscire con exit code 3. Stampare a schermo il contenuto della pila dalla cima al fondo.

PUNTO C: [8 punti] Travaso su Array e Ricerca Min/Max:

- **Punto 3a [4 punti]:** Svuotare completamente la pila caricando e trasferendo tutti gli elementi (tramite operazioni di pop) in un array di prodotti allocato dinamicamente con `malloc` della dimensione esatta.
- **Punto 3b [4 punti]:** Scorrere l'array appena popolato per identificare e stampare a schermo i dati del prodotto con il prezzo minimo e di quello con il prezzo massimo.

PUNTO D: [8 punti] Manipolazione ed Ordinamento dell'Array:

- **Punto 4a [4 punti]:** Scorrere l'array di prodotti: per ciascun prodotto avente prezzo superiore a 100.0, appendere la dicitura " - PREMIUM" in coda al nome (tramite manipolazione della stringa), modificando il valore nell'array.
- **Punto 4b [4 punti]:** Ordinare l'array in-place in base al prezzo in ordine crescente utilizzando l'algoritmo **Insertion Sort**. Stampare a schermo l'array ordinato.

PUNTO E: [3 punti] Output Formattato su File: Scrivere gli elementi dell'array ordinato nel file di output.

- Scrivere nel file solo i prodotti aventi quantità superiore a 10 unità.
- Ciascuna riga del file deve essere formattata in modo tabulare perfettamente incolonnato (usando larghezze fisse in `fprintf`).
- Chiudere il file e stampare un messaggio di conferma. Terminare il programma con exit code 0.

Specifiche delle Funzioni

Il programma dovrà essere articolato in un unico file sorgente e contenere almeno le seguenti funzioni:

- `decodeParameters`: riceve `argc` e `argv`, valida i parametri e li restituisce tramite variabili puntatore o una struttura dedicata.
- `readFileAndFillStack`: legge il file di input e popola la pila.
- `stackToArray`: trasferisce gli elementi dalla pila a un array, restituendone il puntatore e la dimensione, svuotando la pila.
- `findMinMaxPrice`: trova e stampa i prodotti con prezzo minimo e massimo in un array.
- `insertionSort`: ordina l'array di prodotti in base al prezzo in ordine crescente.
- `writeOutput`: scrive il contenuto dell'array filtrato sul file di output con formattazione tabulare.

Note

- **Durata della prova**: 120 minuti.
 - È **VIETATO** usare variabili globali.
 - Si inseriscano i file sorgenti direttamente nella propria home directory.
 - Accesso alla documentazione disponibile tramite il browser all'indirizzo: <https://devdocs.io/c/>.
 - È possibile ritirarsi inserendo la scritta "RITIRATO" come prima riga di ogni file sorgente.
-

Output di Controllo (Esempio di Esecuzione)

Caso 1 — Esecuzione corretta con parametri validi ed elaborazione completa

Eseguendo il programma con il comando:

```
./main prodotti.csv report.csv
```

Un esempio di output stampato a schermo dal programma è il seguente:

```
===== Punto 1: Parametri =====
```

```
File di input: prodotti.csv
```

```
File di output: report.csv
```

```
===== Punto 2: Contenuto Pila (dalla cima) =====
```

```
1. ID: 510, Nome: Cuffie Wireless, Quantita: 95, Prezzo: 65.25
```

2. ID: 402, Nome: Monitor 24 pollici, Quantita: 30, Prezzo: 150.00
3. ID: 305, Nome: Webcam HD, Quantita: 50, Prezzo: 89.90
4. ID: 203, Nome: Mouse Gaming, Quantita: 120, Prezzo: 45.99
5. ID: 101, Nome: Tastiera Meccanica, Quantita: 80, Prezzo: 75.50

===== Punto 3a: Ricerca Min/Max Prezzo =====

Prodotto con prezzo minimo: ID: 203, Nome: Mouse Gaming, Quantita: 120, Prezzo:
 ↪ 45.99

Prodotto con prezzo massimo: ID: 402, Nome: Monitor 24 pollici, Quantita: 30,
 ↪ Prezzo: 150.00

===== Punto 4: Array Ordinato per Prezzo =====

1. ID: 203, Nome: Mouse Gaming, Quantita: 120, Prezzo: 45.99
2. ID: 510, Nome: Cuffie Wireless, Quantita: 95, Prezzo: 65.25
3. ID: 101, Nome: Tastiera Meccanica, Quantita: 80, Prezzo: 75.50
4. ID: 305, Nome: Webcam HD, Quantita: 50, Prezzo: 89.90
5. ID: 402, Nome: Monitor 24 pollici - PREMIUM, Quantita: 30, Prezzo: 150.00

===== Punto 5: Scrittura File =====

File report.csv scritto correttamente.

In questo scenario, il file `report.csv` generato conterrà un testo simile al seguente:

```
203  , Mouse Gaming           , 120  , 45.99
510  , Cuffie Wireless        , 95   , 65.25
101  , Tastiera Meccanica     , 80   , 75.50
305  , Webcam HD              , 50   , 89.90
402  , Monitor 24 pollici - PREMIUM, 30   , 150.00
```

(Exit code atteso: 0)

Caso 2 — Esecuzione con estensione parametri errata

Eseguendo il programma con il comando:

```
./main prodotti.csv report.pdf
```

Il programma stampa su `stderr`:

Errore: I file devono avere estensione .csv o .txt

(Exit code atteso: 2)

Appendici

Questo testo contiene le istruzioni e il codice di emergenza forniti per superare eventuali blocchi logici.

APPENDICE A: Dati di Test (Bypass Lettura da File)

Se non riuscite a completare la lettura da file (Punto 2b) o se riscontrate crash (Segmentation Fault) nel parsing, potete copiare ed incollare questa funzione nel vostro codice per popolare la pila con dati statici e proseguire. L'uso di questa funzione azzerà il punteggio del Punto 2b.

```
void carica_dati_mock(Node** top) {
    Prodotto test_data[] = {
        {101, "Tastiera Meccanica", 80, 75.50},
        {203, "Mouse Gaming", 120, 45.99},
        {305, "Webcam HD", 50, 89.90},
        {402, "Monitor 24 pollici", 30, 150.00},
        {510, "Cuffie Wireless", 95, 65.25}
    };
    for(int i = 0; i < 5; i++) {
        push(top, test_data[i]);
    }
    printf("\n[!] Dati di emergenza caricati (Punto 2b saltato).\n");
}
```

APPENDICE B: Strutture Dati Alternative (Downgrade ad Array)

Se l'implementazione delle strutture dinamiche (Punto 2a) o la loro elaborazione (Punto 3) causano crash bloccanti, potete decidere di aggirare l'ostacolo dichiarando nel main un array statico di supporto:

```
Prodotto array_emergenza[100];
int arraySize = 0;
```

Sviluppate i punti successivi (Ordinamento e Output) lavorando su questo array. In questo caso, i punti relativi alla struttura dinamica (Punti 2a, 2b, 3a, 3b) saranno valutati 0 punti, e il voto massimo dell'intera prova sarà bloccato a 18/30.